Unstructured Decentralised Data Distribution in Wireless Sensor Networks

Ricardo Simón Carbajo

A thesis submitted to the University of Dublin, Trinity College

in fulfillment of the requirements for the degree of

Doctor of Philosophy (Computer Science)

September 2012

Thesis 9968

Declaration

I declare that this thesis has not been submitted as an exercise for a degree at this or any other university and it is entirely my own work.

I agree to deposit this thesis in the University's open access institutional repository or allow the library to do so on my behalf, subject to Irish Copyright Legislation and Trinity College Library conditions of use and acknowledgement.

I hear and I forget.

I see and I remember.

I do and I understand.

- *Confucius* -

Acknowledgements

First and foremost, my sincere thanks to Ciarán Mc Goldrick, my supervisor, for giving me the chance to do a Ph.D. and guiding me through this process. I truly enjoy debating with him and he has always been there for me. In tandem with Ciarán, I have to thank Meriel Huggard. She has been acting as my second supervisor, providing advice of all kinds. I believe their combined style of supervision has provided me with a deep insight into the academic and research world.

Support, support and more support: these words define what I have had from my dear parents, brother and sister. I can count on them unconditionally. I can not forget my aunts and grandmothers for all their assistance when I needed the most, as well as my uncle and cousin for their wise advice. A special thanks goes to Victor, a genuine friend who has been there for me from day one. He is always keen to listen and has the ability to cheer me up when I need it most. He is one of a kind.

I have collaborated with a variety of people during my years in Trinity and I am grateful to all of them. In particular I have to thank Michael Clear. He is a world-class researcher and it is always a pleasure to work with him. I would also like to thank all my colleagues in the lab and in the Distributed Systems Group who have made it a pleasure to study in Trinity. Special mention goes to Gianluigi Pibiri and Ritesh Shreevastav both of whom started the Ph.D. process with me and with whom I have shared many good moments.

I want to express my gratitude to each of my friends who are always there offering encouraging words; this includes my Irish family, the Sweetman's, with whom I have shared many happy moments.

I am also grateful to Stefan Weber and Cormac J. Sreenan for examining this thesis and providing valuable and constructive feedback.

Finally, thanks to Silvana. She has suffered the collateral damage of the Ph.D. process, when I had to reach deadlines or when I could not see the light at the end of the tunnel, or even the tunnel itself. She is special to me and I trust her judgement and good advice. Gracias a todos!

Ricardo Simón Carbajo

September 2012

Abstract

Next generation Wireless Sensor Networks will operate as self-regulated ad hoc networks of tiny devices that sense, actuate and communicate in a collaborative fashion. These networks will also be required to operate in an autonomous and decentralised manner. Data distribution capabilities will facilitate the replication of data elements to multiple, disparate devices in the network. To achieve these goals, a communications infrastructure capable of efficiently scaling and reliably disseminating data to a (sub)set of consumer nodes in a wireless sensor network is required.

This thesis presents the design, implementation and evaluation of TinyTorrents; a novel communications architecture for selective data dissemination in scalable, unstructured wireless sensor networks. The TinyTorrents framework supports mechanisms for selective data pushing and pulling within the sensor network, and to and from the Internet. The architecture is comprised of a data distribution layer, the TinyTorrents protocol, sitting on top of a reliable routing protocol. The cross-layer design employs data-centric dissemination and address-centric routing to enhance the versatility of the communications process. TinyTorrents employs peer-to-peer content distribution concepts to efficiently distribute pieces of data amongst the overlay of consumer and producer nodes. Network traffic burden is balanced amongst consumers and producers.

A reactive routing protocol (UMG) has been developed to complement the TinyTorrents protocol. UMG employs gradient routing concepts to provide reliable end-to-end delivery, data-centric routing capabilities and service advertisement and discovery. Additionally, UMG is robust to moderate mobility of nodes in the network.

The TinyTorrents system has been evaluated under a variety of network conditions and scenarios via simulation. TinyTorrents, using UMG as a routing substrate, has been shown to be reliable, scalable, and capable of distributing data in a fair and efficient manner across the network. The communication architecture has then been evaluated in a real-world testbed comprised of 64 telosB nodes with similarly successful results.

Publications Related to this Thesis

1. **Ricardo Simon Carbajo**, Meriel Huggard, Ciarán Mc Goldrick, "An End-to-End routing protocol for Peer-to-Peer communication in Wireless Sensor Networks", Proceedings of the 6th ACM Workshop on Middleware for Network Eccentric and Mobile Applications (MiNEMA), Glasgow, UK, pp.5-9, 1-2 April 2008

   CONTRIBUTION: This paper describes TinyHop, a lightweight reactive routing protocol for wireless sensor networks capable of discovering bidirectional routes and employing an upstream local repair mechanism. TinyHop provides reliability in the establishment of end-to-end routes and also in data delivery. TinyHop has been used as the initial routing substrate for the centralized data distribution process presented in this thesis.

2. Enrico Perla, Art O'Cathain, **Ricardo Simon Carbajo**, Meriel Huggard, Ciarán Mc Goldrick, "PowerTOSSIM z: Realistic Energy Modelling for Wireless Sensor Network Environments", 3rd ACM International Workshop on Performance Monitoring, Measurement, and Evaluation of Heterogeneous Wireless and Wired Networks (PM2HW2N 2008), Vancouver, Canada, pp.35-42, 31 October 2008

   CONTRIBUTION: This paper describes PowerTOSSIMz, an energy consumption plug-in for TOSSIM 2.x which is based on PowerTOSSIM and incorporates a realistic non-linear battery model for the purpose of tracing energy consumption in wireless sensor networks. The outcome of this work has been employed for the evaluation and comparison, in terms of energy consumption, of the protocols presented in this thesis. The author designed, supervised and contributed to the implementation of this work.

3. Ciarán Mc Goldrick, Michael Clear, **Ricardo Simon Carbajo**, Karsten Fritsche, Meriel Huggard, "TinyTorrents - Integrating Peer-to-Peer and Wireless Sensor Networks", 6th IEEE International Conference on Wireless On-Demand Network Systems and Services, WONS 2009, Snowbird, Utah, USA, pp.119-126, 2009

   CONTRIBUTION: This paper presents the centralized version of TinyTorrents, a data distribution system for wireless sensor network which transparently integrates to the Internet BitTorrent network. This work employs TinyHop as the routing protocol and evaluates TinyTorrents *within the TOSSIM simulator. This thesis further explores the decentralization of the TinyTorrents system in wireless sensor networks, compares its performance with the centralized version, and presents a fully scalable and unstructured data distribution solution. The author designed and contributed to the implementation and evaluation of the work presented in this paper.*

4. Clay Stevens, Colin Lyons, Ronny Hendrych, **Ricardo Simon Carbajo**, Meriel Huggard, Ciarán Mc Goldrick, "Simulating Mobility in Pervasive WSNs - Bridging the gap between ns-2 and TOSSIM 2.x", 13th IEEE/ACM International Symposium on Distributed Simulation and Real Time Applications, Singapore, pp.247-250, 2009

   CONTRIBUTION: This paper presents an enhancement of the TOSSIM 2.x simulator for the purpose of simulating mobility. The implementation allows simulation designers to change the gain (dBm) on any link between two nodes at a specified time. The mobility extension has been tested and calibrated against the ns-2 simulator. This work has served for the evaluation of the protocols employed in this thesis where nodes exhibit mobility. The author designed, supervised and contributed to the implementation of this work.

5. **Ricardo Simon Carbajo**, Meriel Huggard, Ciarán Mc Goldrick, "Opportunistic Detection of Relative Mobility in Wireless Sensor Networks", Wireless Days (WD), 2010 IFIP, Venice, Italy, pp.1-5, 20-22 October 2010

   CONTRIBUTION: This work describes a mobility estimation mechanism which detects if a node has changed its relative position with respect to its neighbourhood based on eavesdropped data. A probability-based model is employed to assess the mobility state of a node according to the information cached in a set of temporal shift Bloom filters which store overheard information. This mechanism has been incorporated in the routing protocol proposed in this thesis, i.e. UMG, to maintain a high level of data delivery when nodes exhibit mobility.

Contents

List of Figures

List of Tables

Glossary

- **Ad Hoc Networks** - The term refers to a collection of wireless devices which cooperate to establish network connections for specific purposes. In this type of network, nodes act as routers of packets.
- **Flash Crowd Problem** - This problem arises when a resource achieves sudden unexpected popularity, resulting in excessive demand on the provider, which might have an impact on the performance of the node.
- **Hot Spot Problem** - This problem arises when a node becomes popular in the network thus receiving a higher number of targeted packets than the rest of the nodes within its area.
- **Interest** - This term is employed in this thesis to refer to the type of data a node provides or wishes to acquire. The interest of a node is defined via the use of human-readable tags encapsulated in efficient memory structures (descriptors). A Torrent file contains a descriptor of the data.
- **Local Mobility** - The term refers to a node changing position within the boundaries of a limited area such that the wireless connectivity with its neighbours still remains or demonstrates an intermittent behaviour.
- **Mote** - The term was coined by researchers working on the NEST Project at UC Berkeley, including David Culler and Kristofer Pister, and is employed to refer to a wireless sensor node. The terms "Node" and "Mote" are used interchangeably in this thesis to refer to a wireless sensor device.
- **Network Partition** - The term refers to the situation where some nodes in an area of the network get disconnected from the rest of the network. This can occur when a node, or set of nodes, acting as the link between areas fail or their energy is depleted.

- **Network Traffic Load Balancing** - The term refers to the fair distribution of packets sent and received in the network by the subset of nodes participating in the communication. The goal is to more evenly distribute packets to reduce the burden in areas, routes or nodes. This can help avoid the Hot Spot and/or Flash Crowd Problems.

- **SensorNet** - The term is used as an alternative name for Wireless Sensor Network and describes a network comprised of sensor devices (Motes).

- **Service** - The concept of Service in this thesis is employed to define any resource provided by a node; this includes data files or resource availability information about other nodes.

- **Swarm (of a Torrent)** - This is the list of peers (nodes) in the network which contain pieces of the data file represented by a given Torrent.

- **Torrent** - In this thesis, the term refers to a specific type of structure in the TinyTorrents protocol employed to describe a particular data file using metadata. It resembles the torrent files employed in the BitTorrent protocol.

- **TT** - Refers to TinyTorrents. It can be used as an acronym for the TinyTorrents framework or the TinyTorrents protocol.

- **UMG** - Refers to the Ubiquitous Mobile Gradient routing protocol.

Chapter 1

Introduction

Wireless Sensor Networks (WSNs) are a particular realisation of wireless technologies characterised by the use of small, low power, low data rate, wireless devices with sensing and actuating capabilities. WSNs enable the monitoring of environmental conditions and infrastructures in a pervasive manner. In effect, the range of applications of traditional wired sensor technologies is extended through the use of small unobtrusive wireless devices which may also form ad hoc networks. However, WSN technologies raise issues that have to be carefully addressed for the technology to be fully adopted and integrated in our daily life. The main challenges come from the constrained characteristics of the wireless sensor devices, also known as "motes". Motes are devices with short communication ranges, low data transmission rates and limited processing power and memory. They are commonly energy constrained as they are battery powered. Additional constraints include the noisy nature of the wireless medium and the unattended nature of the devices and applications.

Extensive research has been carried out over the last decade which has proposed a variety of communication architectures, protocols and mechanisms at various layers of the network stack. However, research has not yet unlocked the full potential of the technology, as envisioned by many authors in the late 90's.

Next-generation Wireless Sensor Networks will involve self-regulated networks of static and transient motes behaving as autonomous colonies - sensing, acting and communicating in a collaborative way. Pervasive networks of co-operative tiny mobile wireless devices, capable of interacting with the environment is an emerging research challenge in ubiquitous computing and will definitely contribute to advancing the vision of ambient intelligence. "Ambient intelligence is the vision of a technology that will become invisibly embedded in our natural surroundings, present whenever we need it, enabled by simple and effortless interactions, attuned to all our senses, adaptive to users and context-sensitive, and autonomous [1]". In this regard, one of the main goals is to transparently connect WSNs to the Internet such that WSNs can also be globally inter-connected. When transparently connected to the Internet, WSN nodes will be easily accessible to any user in any network, including via mobile cellular networks.

WSNs should also be capable of operating autonomously in a decentralised distributed fashion. For this purpose, a Wireless Sensor Network, may operate as a distributed data storage system, preserving data for later upload to other nodes, or to the Internet, from different points in the network. For this purpose, data needs to be distributed to multiple points in the WSN to reduce the risk of data being lost in the event of devices or entire areas failing or becoming disconnected. This is something which could occur due to the error-prone wireless medium, the limited energy of motes, and/or the unreliable physical conditions of the environment where motes can be deployed. For instance, consider a scenario where sensors are tracking and monitoring glaciers which might suddenly break into smaller ice blocks. Climate scientists might be interested in periodically retrieving data from those small separated ice blocks. To achieve such a level of data distribution, a reliable data dissemination protocol is required which distributes data replicas over the network to a set of nodes. This goal necessitates a communications architecture composed of a set of reliable and efficient protocols capable of providing fault tolerant decentralised data distribution in unreliable wireless sensor networks. This thesis addresses the design, implementation and evaluation of such a communications architecture.

The remainder of this chapter is structured as follows: Firstly the main features of the proposed communications framework are introduced and the research domains, where the contributions of this thesis lie, are identified. The research rationale of the work presented is then explained in the context of the WSN domain. Next, the design issues addressed and underpinning assumptions are presented. The contributions of this thesis follow. Finally, a summary of the thesis is provided and directions for future work discussed.

1.1 A Novel Peer-to-Peer Data Distribution Framework

The TinyTorrents (TT) framework is presented in this thesis, a communications architecture which provides the substrate for the development of the next generation of co-operative decision-making applications for wireless sensor networks. TinyTorrents challenges the rather narrow, commonly held, perception of a WSN as a network of devices with sensing capabilities which retrieve data, fuse it, and send it to a network gateway to be stored or utilised. This is achieved by leveraging existing Peer-to-Peer (P2P) content distribution concepts from wired and wireless networks. TinyTorrents employs P2P data distribution systems to efficiently distribute data globally and inside the sensor network. In this sense, the TinyTorrents framework can be viewed as being composed of two elements:

- *the Gateway-Internet side*, which transparently integrates the WSN to the global BitTorrent [2] network, thus leveraging all the benefits that P2P content distribution solutions offer for global data dissemination, and
- *the Wireless Sensor Network side*, where a reliable and decentralised communications system for selective data dissemination over multi-hop WSN has been designed which employs concepts from the BitTorrent content distribution protocol.

The contributions of this thesis reside in the *Wireless Sensor Network side*. Here a selective data dissemination protocol for reliable data distribution in sensor networks is presented as the TinyTorrents protocol (see Chapter 5). The TinyTorrents protocol employs BitTorrent-like P2P communication mechanisms to replicate data amongst nodes in a efficient, reliable and scalable manner across the WSN. The TinyTorrents protocol cannot work in isolation - it needs to operate above a suitable routing mechanism. For this purpose two routing protocols have been designed to provide the wireless network communications required by the paradigm of data distribution in sensor networks. The first protocol is called TinyHop [3], which is an on-demand, end-to-end, flat-topology, reliable routing protocol. The second protocol is called Ubiquitous Mobile Gradient (UMG) (see Chapter 4) and introduces improvements with respect to TinyHop by using the concept of gradients in such a way as to achieve efficient P2P communication. UMG also introduces an efficient lookup mechanism to support content distribution techniques in TinyTorrents and is designed to work in an environment where nodes can experience moderate mobility.

The TinyTorrents framework transparently disseminates and replicates sensor data on a global scale via the Internet. For this purpose, the *Gateway-Internet side* of the TinyTorrents framework connects to the Internet BitTorrent network via the use of gateway devices implementing a customized plug-in for Vuze, a popular BitTorrent Client/Server application. When data is routed out of the WSN, all the benefits of P2P dissemination can be employed to distribute data over the Internet - including Web 2.0 services and RSS feeds. In disseminating data across networks on a global basis, TT conforms to the "Infrastructure as a Service" (IaaS) capability set required for cloud computing deployments. The TinyTorrents framework is intended to target a broad range of application scenarios where a generic, stable, robust and fault tolerant communications platform provides the framework for the development of tailored application domain solutions.

1.2 Research Rationale

Much of the research in wireless sensor networks tackles the problem of data dissemination from the perspective of the initiator of the communication: i) source-to-sink, i.e pushing or ii) sink-to-node i.e. pulling. A sink node pulls data by sending a request to the source node. However, a source node may decide to push data to the sink at any given time. Pushing data towards a sink, or set of sinks, can be seen as a data collection process. Sinks can be placed at different points in the network for continuous collection or can act as opportunistic or nomadic points of collection which move to another area once data has been retrieved. Nomadic mobile sinks are sometimes known as data mules. Sink nodes usually have more resources and can communicate with other networks such as the Internet. Hence they are employed as gateways to perform pulling activities over the WSN and to push data outside the WSN. For reconfigurable systems, pushing data into the sensor network is employed for programming purposes where the goal is to reliably deliver big chunks of data to all, or most of the nodes, in the network; this is traditionally achieved in a hop-by-hop epidemic manner.

Considering WSNs will cooperate as a set of sensor and actuator nodes taking decisions and performing actions based on distributed data, a versatile communications architecture is required. Pushing and pulling data from any point in the sensor network is necessary such that nodes receive up-to-date information on its surroundings for collaborative decision making. Moreover, data may need to be replicated and disseminated so that the sensor network acts as a distributed storage medium; this avoids data loss, increases data integrity, makes the network fault tolerant, allows for efficient and reliable data collection from any point, and reduces the appearance of the hot spot and flash crowd problems. Storing data at multiple points gives the sensor network a degree of autonomy when performing data collection, where data can be retrieved from different points at a later stage, while also enables decision making based on historical data. However, nodes should decide whether to participate in the dissemination and storage process according to their capabilities or the status of their resources. In addition, when a node needs to communicate to a distant area within the same WSN, or to another WSN, backbone networks, such as the Internet, should be employed when possible. For instance, nodes sensing and acting a wide area might require to be aware of certain events and data produced at 30 hops away. A backbone network could be used to transparently push data close or into the requesting area via gateway nodes placed at any point in the network. However, when a gateway is not accessible, a dissemination process will be required to reliably push data towards disparate areas.

Considering the above, this thesis presents the TinyTorrents communications architecture for selective data pushing and pulling within a wireless sensor network and to and from the Internet. Although the TinyTorrents framework is developed in the document, the major contributions of this thesis lie in the area of selective data dissemination and routing for wireless sensor networks. A routing protocol, UMG, and a data distribution protocol, the TinyTorrents protocol, are proposed as the communications architecture required to accomplish the WSN-side collaborative scenario presented above.

1.3 Assumptions and Design Issues

The following list of assumptions and design issues help to scaffold the design of the system rather than bound it.

- Common scenarios for WSNs may include a low number of nodes changing neighbourhood, that includes sink nodes, with a relatively low speed, e.g. a person walking or even jogging. Therefore, support for mobility needs to be in place taking into consideration that a WSN is not a vehicular ad-hoc network. However, vehicular networks could make use of the WSN for opportunistic data communication.
- Nodes do not employ a system to calculate or obtain their geographical coordinates, such as a GPS device, and coordinates are not hardcoded in the device. Nodes can be deployed in an unsupervised fashion if required.
- Every node can act as a gateway for the integration of the WSN with other networks such as the Internet. Connections can be established through any existing node and a new node may be deployed in a particular area which might not be stationary.
- Nodes can use the communication protocols to query the network or can passively wait for metadata to be received in order to decide whether to acquire the associated torrent data.

- A flat-network topology is adopted as a more flexible and adaptable structure which requires less management than a hierarchical approach where clusters need to be formed and maintained.
- The communications architecture presented in this thesis maps the decision process of some routing and data distribution factors to the application layer, such that the designer retains control over the involvement of the node in the dissemination process, communications efficiency and data storage.

1.4 Contributions of this Thesis

This section identifies the contributions of this thesis by highlighting the key features and behaviours of the protocols presented.

- The Ubiquitous Mobile Gradient (UMG) routing protocol is presented in Chapter 4 as an opportunistic, reactive, gradient-based routing protocol for data collection, point-to-point, multipoint-to-point and point-to-multipoint communication. The research contribution of UMG lies in the area of versatile routing protocols for low power and lossy networks. The goal was to develop a routing protocol capable of supporting efficient and reliable communication for collaborative application scenarios in WSNs. For this purpose UMG has been designed employing the gradient routing concept while offering reliable mechanisms for the creation, update and navigating of the gradient field. Local recovery mechanisms and efficient caching systems are employed to increase the reliability of the end-to-end communication. UMG is address-centric in the sense that node identifiers are employed for routing, but it also integrates support for data-centric routing by making use of Bloom filters as compression mechanisms of service description. Bloom filter-based descriptors are integrated into the gradient formation thereby providing a mechanism for data-centric searching. UMG has been designed to tolerate sink and source mobility by opportunistically detecting a node's relative change in neighbourhood and reacting by locally updating its gradient.
- In Chapter 4, Section 4.6, the opportunistic relative mobility detection algorithm employed by UMG is explained. The contribution of this approach lies in the area of efficient mobility detection mechanisms based on neighbourhood awareness. The mechanism defines a model, which can be parameterized by the user, to compute the likelihood that the node is changing its neighbourhood, only taking into account opportunistic communication. The approach, which

can be easily employed by any routing protocol seeks to avoid the proactive periodic beacon employed by neighbourhood detection systems in determining mobility.

- The TinyTorrents protocol is then presented in Chapter 5 as a decentralised data distribution protocol which sits on top of UMG operating in a cross-layer fashion. One of the research goals in the area of data dissemination is to reliably and efficiently deliver data to all, or a selection of nodes, in the network. Most of the approaches examined in the literature employ some sort of epidemic algorithm where data is diffused over the network in a hop-by-hop basis. Nodes either rely on their neighbours to acquire data or employ some sort of routing protocol to acquire data from a source node. One of the research goals of this work is to disseminate data only to those nodes interested in it while utilising other nodes for relay purposes in a process whereby the dissemination mechanism acquires control over the flow of data traffic. In this way, the traffic can be balanced, avoiding factors such as network partition, hot spots and flash crowds, which can adversely impact the performance of the network, up to a point of partial or total failure. The TinyTorrents protocol, operating in a cross-layer fashion with UMG, offers a selective data dissemination mechanism capable of managing the flow of data in the dissemination process. It employs peer-to-peer content distribution concepts from the BitTorrent protocol for data partition and distribution, with the goal of reducing the burden on any node in the network, including the source of data, by providing selective data redundancy and co-operative data distribution. The architecture has been designed to operate in a self-regulated, decentralised manner where every node has the capability of acting as a partial location tracker for proximate data. Unstructured mechanisms are employed for data discovery instead of structured (DHT-based) algorithms - the latter being sensitive to mobility whilst requiring more control and communication input. The unstructured discovery mechanisms employ UMG's Bloom filter descriptors to prune the scope of the candidate node's searching process. Nodes participate in the decision of which pieces of data to retrieve, and from which peers, according to different peer-piece selection mechanisms. The cross-layer architecture design of the TinyTorrents protocol and the UMG routing protocol employs data-centric dissemination and address-centric routing to enhance the versatility of the communications capabilities. The TinyTorrents protocol pushes data from any node into the network, employing a handshake-based scheme based on advertised metadata. More traditionally, the UMG routing protocol can also be employed as a mechanism for data pulling.

1.5 Thesis Roadmap

The remainder of this thesis is organised as follows:

Chapter 2: State of the Art

This chapter reviews the literature in the areas of the contribution of this thesis following a general-to-specific approach. Firstly, an overview of the Wireless Sensor Network technology is described. Then, a general review of the area of routing protocols is given. A comprehensive review of the state of the art in the area of gradient routing protocols for wireless ad hoc networks is provided. Finally, the state of the art in peer-to-peer content distribution protocols for wired and wireless networks is presented including P2P discovery mechanisms and dissemination protocols for WSNs.

Chapter 3: The TinyTorrents Framework: Data Distribution in WSN

This chapter presents the top-level component view of the TinyTorrents framework. The architecture is described which is divided into two main areas of communication: i) the Gateway-Internet communications architecture and ii) the WSN-Mote architecture. The work presented in the remainder of the chapters focuses on the design, implementation and testing of the WSN-Mote communications architecture, and is where the major research contributions are made.

Chapter 4: The Ubiquitous Mobile Gradient Routing Protocol

In this chapter, the UMG routing protocol is presented as a versatile communications protocol for the WSN-Mote architecture. UMG provides reliable bidirectional point-to-point, multipoint-to-point, and point-to-multipoint connectivity employing gradient concepts. A description of the operation of the protocol is given. Subsequently, each of the phases of the routing process are explained along with the mechanisms and algorithms involved, as well as the implementation and data structures utilised.

The last section of the chapter, Section 4.6, thoroughly describes the mechanism for opportunistic detection of relative mobility employed by UMG.

Chapter 5: The TinyTorrents Protocol

This chapter presents the TinyTorrents protocol as a selective data dissemination layer which utilises UMG to achieve multi-hop communication. The TinyTorrents protocol, which employs data partition and distribution concepts from the BitTorrent protocol, is designed to operate in a centralised and decentralised manner. Initially, the design and implementation of the common architecture and functionality of both versions of the TinyTorrents protocol are described in detail. Subsequently, the progression in the design from the centralised version towards the fully decentralised architecture is described, as well as the advantages and disadvantages arising. The remainder of the chapter focuses on the decentralised version following the idea of every peer being capable of acting as a partial tracker for peers in a proximate area. Unstructured mechanisms for discovering potential trackers are then proposed. Peer-piece selection strategies are explored by taking into account the constraints posed by the WSN domain. Finally, the mechanism employed by the TinyTorrents protocol for service discovery, i.e torrent data discovery, is also described.

Chapter 6: Evaluation

The Evaluation chapter introduces the operating system employed for the implementation and performance analysis of the protocols, i.e. TinyOS version 2.x, and describes its integrated simulator, TOSSIM. The specifications of the sensor devices employed in the evaluation are also given. The TinyTorrents protocol in its two modalities, centralised and decentralised, operating above the UMG routing protocol is evaluated. A performance analysis employing a variety of network conditions, potential real-world scenarios and protocol configurations is provided. The system is also evaluated in terms of scalability, both when the network and consumer-producer distributions scale. Additionally, TinyTorrents is compared against state-of-the-art dissemination protocols in the simulator. Finally, the decentralised version of the TinyTorrents protocol is evaluated in a real-world testbed comprised of 64 telosB motes.

Chapter 7: Conclusions & Future Work

The author summarises the work presented in this thesis and its contributions in the area of Wireless Sensor Networks. Future work is proposed to enhance the functionality of the TinyTorrents framework while indicating potential research avenues to explore in the area of data distribution in WSNs.

Chapter 2

State of the Art

This chapter reviews the state of the art in the areas of the protocols technologies presented in this thesis. The first section provides the reader with an overview of the Wireless Sensor Network technology and describes its main design principles. A description of Medium Access Control (MAC) solutions in WSNs then follows. The next section introduces the area of routing protocols in WSNs and includes a general classification. Thereafter, the mechanisms of gradient-based routing in wireless networks are explained. A comprehensive review of the most relevant gradient-based routing protocols in WSNs, which impact the design of the Ubiquitous Mobile Gradient (UMG) routing protocol described in Chapter 4, is provided. Finally, related work in the areas of i) peer-to-peer content distribution for wired and wireless networks (including the BitTorrent protocol), ii) structured and unstructured P2P discovery mechanisms in wireless networks, and iii) data dissemination protocols in WSNs, is explored in the last section. This final section aims to introduce concepts employed in the TinyTorrents protocol, described in Chapter 5, while identifying the research areas where the architecture presented in this thesis is placed.

2.1 Wireless Sensor Networks

Interest in Wireless Sensor Networks (WSN) has grown rapidly since Mark Weiser introduced the concept of "ubiquitous computing" in the paper "The Computer for the Twenty-First Century" [4] in 1991. While wired sensors have been employed since Cold War times and are widely deployed in everyday objects, they are still subject to cabling constraints and consequently are unsuitable in many deployment scenarios. Wireless Sensor Networks overcome many of these limitations by wirelessly interconnecting sensor devices. Wireless sensors can be deployed in an ad hoc, unstructured fashion and communicate with each other to form cooperative networks. To achieve this goal, a WSN device not only needs to have sensing and wireless communication capabilities, but also incorporate a microprocessing architecture capable of running distributed algorithms for in-network data processing, cooperative decision taking and multi-hop networking.

The main challenges for wireless sensor network technologies come from the constrained characteristics of the sensor devices, also known as "motes"; typically they are battery powered devices with constrained processing power, limited storage and short communications ranges achieving low data transmission rates. Furthermore, the distributed redundant nature of this technology necessitates special attention to issues such as sensor node deployment, density of nodes in an area, environmental interference, and amelioration of node failure. These challenges can affect the network fault tolerance and robustness, mainly by compromising coverage of the region/area where the phenomenon occurs. In addition, modern networks commonly enable Quality of Service (QoS) provisioning. In WSNs such provisions may need to be achievable in error-prone wireless environments. Robust WSN system design needs to anticipate such requirements. Whilst low-level QoS is realised using metrics such as bandwidth, delay, jitter and packet loss rate, in WSNs there is a parallel interest in the requirements at the application level, what is referred as high-level QoS. As many sensor platforms communicate using the unlicensed ISM bands, they may encounter considerable interference traffic from other devices.

While the technology is no longer in its infancy, we cannot say that it has attained its full potential. Much research has been undertaken in developing architectures and protocols which sensor devices can use to communicate amongst themselves and with other networks, but these have yet to be standardized or widely embraced by the community. Amongst these standards, IEEE 802.15.4 [5] for the PHY and MAC layers is the most widely adopted, employing 3 ISM frequency bands for operation, 2.4GHz (global), 915MHz (Americas) and 868MHz (Europe). Diverse communities and associations are working on creating specifications of protocol suites which layer on top of the IEEE 802.15.4 standard. For example, a free versatile specification of higher layer protocols has been developed by the ZigBee Alliance [6], while MiWi [7] comes as a simple proprietary protocol suite. There are also specifications fitting particular settings, such as the industry oriented ISA-SP100.11a [8] and WirelessHART [9]. These protocols seek to offer communication architectures which manage ad hoc networks of constrained devices and efficiently transport data within the WSN. There are also efforts on the integration of WSN with the Internet; in this regard, the Internet Engineering Task Force (IETF) is developing the IPv6 over Low power Wireless Personal Area Networks (6LoWPAN) standard [10]. In 2010, the IETF Routing Over Low power and Lossy networks (ROLL) Working Group was formed [11] to standardize a generic IPv6-compliant routing protocol for this type of networks, i.e. the Routing Protocol for Low power and lossy networks (RPL) [12].

In the following section, the main design issues which differentiate wireless sensor networks from other wireless technologies are discussed in detail.

2.1.1 Design Issues in WSN

When designing protocols and algorithms for Wireless Sensor Networks, some design issues which characterise this technology have to be considered. Next, some of the most relevant design issues are described in the context of wireless sensor networks.

2.1.1.1 Fault Tolerance/Robustness

A system is fault tolerant when it can continue correct operation in the event of a partial hardware or software failure. Redundancy is employed to make systems fault tolerant and reliable, and can be achieved in a variety of ways in WSNs. For instance, sensor devices can be placed into the same area to redundantly monitor a phenomenon. In this scenario, a given node might obtain the same, or closely correlated data for a particular phenomenon from more than one neighbour. This way multiple instances of the same data can be compared to assess the integrity and accuracy of the sensed data. In terms of communication, redundancy can be achieved by disseminating data to different points in the network. This enhances the network's robustness in case of a node failure or network partition.

Factors such as the type of deployment, network density, energy consumption, physical damage and environmental interference can affect the network fault tolerance, mainly by not supporting coverage and communication to the whole area where the phenomenon occurs.

Koushanfar et al. state that fault tolerance in WSN can be addressed at different levels [13]: Hardware, System Software, Middleware and Application. They also mention that heterogeneous networks increase the fault tolerance of the system as there are different sources of data with different properties which can be cross-checked to detect the presence of a faulty node.

Algorithms should be designed to be robust to changes in the topology, environmental problems and mobility of nodes such that the normal functioning of the systems is not affected, and if it does, minimize the time in solving the failure while risking minimum resources.

2.1.1.2 Scalability

In a wireless sensor network, the number of nodes can grow from hundreds to thousands, either by covering the same area, i.e. increasing the density, or by extending the coverage area. In both cases, the system has to be prepared to handle such issues. The same phenomenon can be detected by a big quantity of sensors, and data might need to be transported over large distances employing multihop communication. Performance has to be maintained in larger WSNs with techniques such as data fusion and aggregation to minimize the number of retransmissions. The use of clustering mechanisms to enhance the reliability when transmitting data over long distances is also proposed, despite its sensitivity to topological changes and the required coordination of the cluster.

2.1.1.3 Quality of Service

Quality of Service (QoS) can be divided into two categories for wireless sensor networks [14]: i) low-level QoS, related to the networking of the devices and ii) high-level QoS, which is usually imposed at the application layer and can be considered as a user-observable subjective metric; it has also been categorized as quality of experience, QoE.

While the low-level QoS refers to parameters like bandwidth, delay, jitter, packet delivery and loss rate, the high level QoS is focussed on application requirements. Different performance/attributes might be required by each specific application for a system to provide. For instance, specific QoS is required in applications belonging to the following areas: i) event detection/reporting probability, ii) event classification error, iii) event detection delay, iv) missing reports, v) approximation accuracy, vi) tracking accuracy.

When designing protocols tailored to each application, a balance between quality of service or experience and the required resources might need to be achieved. The nature of the application typically establishes this trade-off. For instance, health monitoring applications might require "real-time" data reporting, within less than a second, from a patient sensor device, while humidity control applications for home environments may not require such constrained time boundaries.

2.1.1.4 Mobility

The majority of wireless sensor networks work with stationary motes tracking static or mobile phenomenon. Currently, research is focusing on what is called Mobile Wireless Sensor Networks, where the sensor devices themselves might be transient.

Depending on the type of WSN, nodes might have different responsibilities or capabilities, i.e. heterogeneous networks. For instance, a node might be in charge of collecting data for the whole network or for a particular area; this node is denoted as the "sink". A WSN can also have multiple sinks at different points. When designing protocols, especially in routing, the type of node which is experiencing mobility needs to be taken into account as this can determine the performance of the system.

Mobility can appear as: i) node mobility, where the node itself moves due to some sort of device attached to it or due to the effect of the environment, ii) sink/gateway mobility, where a device with mobile capabilities gathers data from nodes (also known as the "data mule"), and can act as a gateway to other networks or iii) event mobility, where static sensor devices monitor and track transient objects or transient physical phenomenons. For network connectivity purposes, a node is considered to show mobility when a change is produced in its neighbourhood set; this is called relative mobility. In addition, it has to be noted that mobility of a sensor node might affect the area of the phenomenon being monitored.

2.1.1.5 Hardware Constraints

A sensor device, also known as a "mote", is comprised of 4 main units: i) processing unit, ii) transceiver, iii) sensing/actuating unit and iv) power unit. There are other components which might be attached and which depend on the application requirements such as GPS/localization unit, power generator, etc. These devices have to operate autonomously and fit in reduced spaces. Energy sources need to be unobtrusive even if that implies reducing their energy capacity. According to this, the rest of the components in a mote are constrained by the lack of energy and they need to minimize their power consumption. To do so, components operate in low duty cycle mode and go to sleep whenever possible. Limited size flash memories are employed and microprocessors tend to reduce the power consumption and operate at low frequencies (8 MHz). Radio transceivers consume the highest energy compared to the rest of the components, which is the reason why researchers seek to reduce communication activities in their protocols. Despite these constraints, the technology is evolving rapidly; new energy sources which last longer and scavenging systems like solar cells are being researched. The number of transistors that can fit in an integrated circuit has still not reached its limit and keeps on growing without strictly following Moore's law. A good example of the evolution of the hardware is the new Lotus mote [15] which integrates a 32bit 10-100MHz CPU, 64KB SRAM, 512KB Program Flash and 64MB Measurements Serial Flash. This mote supersedes previous technology like the micaZ mote [16] which incorporates a 8bit 16MHz CPU, 4KB SRAM, 128KB Program Flash and 512KB Measurements Serial Flash, representing a huge jump in memory size and computational capabilities.

The generic characteristics of this technology - i.e. low power, low processing capabilities, reduced size of memory - along with the high number of devices which are needed to cover an area, demand for the components to be cheap. In order to create networks of hundreds of nodes, they have to be priced at far less than € 1 to be feasible. Currently the existing hardware for research and commercial purposes is too expensive for the technology to be profitable.

2.1.1.6 Deployment/Topology/Connectivity/Coverage

There are two types of deployment in WSN: i) supervised and ii) unsupervised. The first one involves the creation of the network where motes are placed in strategic locations to efficiently cover the whole area to be monitored and to increase the network's fault tolerance. Once it is deployed, the network should have a full connectivity amongst all the motes and complete coverage of the phenomenon. On the other hand, the unsupervised deployment is an uncontrolled process where the position of the motes within an area is not determined or controlled. This can lead to the situation where some areas have no coverage, the network may not be fully connected, and/or the absence of redundancy might drastically affect the fault tolerance of the system. In this case, the network may be partitioned and data from affected areas might not reach the gateway. An example of unsupervised deployment occurs when nodes are dropped from a plane; this produces irregular density and areas might be uncovered or isolated in terms of communications. On the other hand, this type of deployment is not as cumbersome as the supervised approach where, for instance, full connectivity and coverage should be guaranteed to monitor every single corner of some secure premises. The mobility of nodes, the effect of the environment, and the malfunctioning of strategic motes could break down both classes of networks, supervised and unsupervised, if there is not enough redundancy of sensors within a given area. In this case, protocols should adapt to the new topology to try to maintain the normal operation of the network. A redeployment phase can be carried out to replace or distribute sensor devices in compromised areas [17].

Bulusu et al. calculate the network density $\mu(R)$ in terms of number of nodes per nominal coverage area like [18]:

$$\mu(R) = \frac{N\pi R^2}{A} \tag{2.1}$$

where N is the number of sensors within region A and R is the range of a particular sensor or the radio transmission range. In this paper, critical density λ is defined as the density required to achieve a given task. If the network density is greater than the critical density, i.e. $\mu(R) > \lambda$, then only a subset of λ nodes needs to participate. The number of possible subsets of nodes can be calculated with the binomial coefficient index by μ and λ. In addition the authors in [19] demonstrate that, with a uniform distribution, the probability that a node is connected reaches 1 when $\mu(R) = 6$.

2.1.1.7 Applications

Traditional sensor devices were restricted by wires and consequently could not operate in environments where the installation of wires was inefficient or simply not possible. With the revolution of WSNs, the range of environments which can be monitored and actuated go from environmental applications for early flood detection [20], volcano monitoring [21] or animal monitoring (e.g. the Great Duck Island project [22] and the ZebraNET project [23]) to military application for ad hoc estimation of the trajectory of a bullet and the sniper position [24].

In addition, underwater applications, referred as Underwater Wireless Sensor Networks (UWSN) [25], are capable of monitoring for example the quality of sea bays, while underground sensors, also known as Wireless Underground Sensor Networks (WUSN) [26], can assist geologists in unobtrusively measuring soil properties. Furthermore, the concept of Body Sensor Networks (BSN) has been coined to describe WSN operating in human bodies, for instance monitoring the health of patients with Parkinson's disease [27].

Materials are starting to be developed with wireless sensors embedded for unobtrusive monitoring; for instance wireless sensor devices integrated in wind turbine blades are being studied for structural health monitoring and damage detection [28]. Problems may arise when the conditions in the environment interfere with the communication process, when the devices cannot be accessed to replace the energy source, or even when the motes are subject to physical alterations from the surrounding medium; this needs to be carefully considered in the specification of the application domain.

2.1.1.8 Transmission Media

In the area of WSN, the wireless transmission medium which has been most widely adopted is the Radio Frequency (RF) based. Although optical and infrared are considered, due to their protection against interference from electrical devices, they need line of sight for communication which reduces the scope of environments where motes can be deployed. Underwater Wireless Sensor Networks mostly employ acoustic waves to form wireless links, while WSN transmitting underground (WUSNs) employ magnetic induction.

In the RF spectrum, the efforts are focussed on getting a radio frequency band which can be established as a standard for WSN transmission worldwide. Currently WSN technology employs different unlicensed bands for communications such as 868 MHz, 915 MHz, 2.4 GHz Industrial, Scientific, Medical (ISM) or the 5GHz Unlicensed Network Information Structure (U-NII) which are used for multiple purposes with different transmission power ranges. The energy constraint of motes makes them one of the weakest contenders for the channel. As an example, technologies like IEEE 802.11 (WiFi) and Bluetooth, which are usually implemented in devices with higher energy limitations, work in the 2.4 GHz band, same range in which microwave ovens may operate, causing interference in the wireless medium which can affect the reliability and integrity of the sensor network communications operating in the same band and channel. Despite using narrow-band and ultra-wide-band (UWB) transmission techniques, WSNs commonly employ spread spectrum techniques due to its resistance to interference. While Frequency Hopping Spread Spectrum (FHSS) is employed in Bluetooth technology, most of the WSN transceivers employ Direct Sequence Spread Spectrum (DSSS). In DSSS, the narrow-band signal and its energy is spread over a larger bandwidth such that it appears like noise. Indeed, the signal is multiplied by a stream of pseudo-noise codes, i.e. chips, employed to code and decode the signal. The pseudo-noise stream needs to be known beforehand by those devices involved in DSSS-based communication. DSSS is utilised in chips such as the Texas Instruments CC2420 Chipcon Transceiver [29], which is IEEE 802.15.4 [5] compliant, and is embedded in motes such as the micaZ [16] and telosB [30]. ASK, FSK and PSK and its variants are employed for digital modulation of the signal. For instance, the IEEE 802.15.4 standard employs Binary Phase Shift Keying (BPSK) modulation in the 816MHz (1 channel) & 915MHz (10 channels) bands and Offset Quadrature Phase Shift Keying (O-QPSK) in the 2.4GHz band (16 channels).

Researchers are working on cognitive and Multiple-Input Multiple-Output (MIMO) radios which can dynamically vary the transmission power, change the transmission band according to the prevailing interference level and use multiple channels to carry the signal.

2.1.1.9 Power Consumption

Currently motes operate with limited energy sources, usually a set of AA or AAA batteries providing 1.2-1.5 Volts. The goal is to further reduce the size of the power source such that it can be unobtrusively integrated in any environment and still provides enough energy capacity to keep motes running for years. Advances in materials, nanotechnology and even scavenging techniques will be key to develop such energy sources. In addition, the way in which electronic components make use of the energy is an important factor to consider. The wide range of components, like transceiver or microprocessors, draw energy according to their different operational states and can be turned on/off and, thus, operate at different energy levels. The energy levels of components can be controlled and combined to produce different operation states within a mote; that is referred as Dynamic Power Management (DPM) [31]. The mote can be in the "active" state in which most of the main components will be on, the "idle" state in which components like memory or microprocessors can be turned off, and in "sleep" state in which all components turn off for a period of time presuming the mote does not need to be operative. In addition, in most of the WSN platforms, communication activities - transmission, reception and idle state - consume higher energy than processing and sensing tasks. For instance, the telosB mote [30], which integrates the Texas Instruments CC2420 Chipcon Transceiver [29], consumes 32 mA in receiving mode while only drawing 1.8 mA in the MCU when in active mode. In addition, transceivers allow for different transmission powers to control the transmission range and energy. This is highly useful when deploying motes in order to establish different densities at different areas of the network. Moreover, dynamically adaptive protocols are capable of modifying the transmission power according to the current status of the network to optimise communications.

2.1.1.10 Heterogeneity

A wireless sensor network is formed by homogeneous motes when all the sensor devices retrieve the same type of data and have the same capabilities. On the other hand, a WSN can be comprised of devices with different capacities in terms of computation, power or communication, hence creating a heterogeneous network. In these type of networks, different devices can also measure different phenomena like temperature, humidity, capture images or track devices. In this regard, these networks are usually more difficult to manage as protocols should take into account the device capabilities for routing purposes and for data processing. On the other hand, the greater the diversity of data which can be gathered from an area, the richer the information for monitoring purposes and the higher the redundancy. An example of an heterogeneous device can be a cluster head node. This node can be used to perform data aggregation of data from nodes within its cluster and communicate the processed data to far away nodes. For instance, the IEEE 802.15.4 MAC layer [5] defines two types of devices: Reduced Function Device (RFD) and Full Function Device (FFD). RFD have limited capabilities and their communication is restricted to FFDs to achieve multihop end-to-end connectivity. However, FFD devices are equipped with higher capabilities and can act as Personal Area Network (PAN) coordinators.

2.1.1.11 Data Reporting Method

The data reporting method employed by a node depends on the nature of the application and its QoS requirements. Energy conservation and routing protocol designs depend on the type of data reporting method employed [32]. These methods can be classified as: i) time-driven, ii) event-driven, and iii) query-driven.

In time-driven reporting methods, motes are schedule to transmit in time intervals, hence if a phenomenon is monitored the mote has to wait until its allocated time to transmit. That is not suitable for critical applications like fire alarms or real time systems in which the data needs to be obtained rapidly. This method is energy efficient when the phenomenon occurs frequently.

On the other hand, event-driven methods report data as soon as the event is detected. It is usually employed in critical systems where the phenomenon needs to be monitored in real-time. If events occur frequently, the cost in energy of this method increases radically.

When either a base station or another mote requires some data, the method employed is query-driven, where motes reply to explicit queries from requesting devices. This method becomes energy inefficient when queries are issued and data is not available.

Depending on the application, a combination of the data reporting methods is usually a solution to reduce communication and decrease energy consumption.

2.1.1.12 Data Aggregation/In-network processing

Despite the benefits of redundancy of data and communication for network robustness, an excess of redundancy may be highly inefficient in terms of energy consumption and communications. In this case, techniques such as in-network processing and aggregation, where data is manipulated before forwarding, are employed. Aggregation of correlated data resulting from nodes monitoring the same phenomenon is one of the keys to reducing the excess of redundancy. In the process of aggregating data, there are some issues to consider like: i) does the aggregator have to wait for all its associated motes to transmit before aggregating data?, ii) which function is to be used to aggregate the data and what is its cost in terms of complexity and processing power?, iii) how is the aggregator identified and is there a need to create a special topology?, iv) what is the risk of losing redundancy with the aggregation and how would this affect the system fault tolerance?.

All these issues have to be taken into consideration at the time of designing the system. For instance, aggregation functions can range from duplicate suppression to complex algorithms which compose images from different mote locations. Cluster heads are usually employed as aggregators.

The efficacy of the data aggregation can be calculated according to the following metrics [14]: i) accuracy of the resulting value at the receiver, ii) completeness of the aggregation, iii) latency generated with the aggregation process, and finally iv) the amount of compression in the message overhead. Signal processing methods can be used for a mote to produce a more accurate output signal by combining the incoming signals (data fusion) and reducing the noise [32].

2.1.2 Medium Access Control in WSN

Medium Access Control (MAC) protocol design for Wireless Sensor Networks is influenced by the constrained factors that characterise this technology. These factors involve ad hoc wireless communication, energy-efficiency, low processing capabilities, duty-cycle activities and nature of mobility. The main classification for these types of protocols is based on how/when the medium is accessed; in this regard two main groups are defined for MAC protocols: i) contention-based and ii) contention-free.

The first group, contention-based, can also be defined as random access protocols, as there is no coordination amongst the nodes accessing the medium. This randomness produces packet collisions and consequently retransmissions which increase the latency of the communication potentially impacting the quality of service of the system. Earlier MAC protocols for wired networks included Pure Aloha or Slotted Aloha which were outperformed by the Carrier Sense Multiple Access (CSMA) and its Collision Detection (CD) mechanism used in Ethernet. CSMA was later enhanced for use in wireless communications with the Collision Avoidance (CA) mechanism, where collisions could not be detected but rather avoided. CSMA/CA employs binary exponential backoff mechanisms to space out the retransmissions in the presence of collision; this reduces the likelihood of packets colliding and reduces congestion. This protocol is widely used in IEEE 802.11 wireless LANs and Wireless Sensor Networks. However, transmission in the wireless medium gives raise to the hidden and exposed node problems. To mitigate or reduce the occurrence of these problems, two approaches have been employed. The first one is known as the busy tone, which requires the node to operate in duplex mode with two channels, one for data and another for control. The receiver will send a signal through the control channel to indicate to the neighbourhood that is busy receiving a packet. The second approach implements the handshake mechanism for the CSMA/CA based on Request to Send (RTS) and Clear to Send (CTS) messages, which does not fully solve the hidden node problem in all the scenarios. Examples of random access-based protocols, which are usually variants of the CSMA/CA can be classified as [33]: i) Multiple Transceivers [34], ii) Multiple Path [35], iii) Event-Centred [36] and iv) Encounter-based [37].

The second group, the contention-free MAC protocols, seeks to avoid collisions by scheduling the access to the medium. Scheduling can be based on time, Time Division Multiple Access (TDMA), on frequency, Frequency Division Multiple Access (FDMA), or on code, Code Division Multiple Access (CDMA). The TDMA type of protocol is based on scheduling time slots for each node in the neighbourhood to transmit in a contention-free fashion. Issues like synchronization have to be taken into account. Centralised solutions use cluster heads to synchronize neighbourhoods while distributed solutions usually depend on global synchronization. These type of protocols need an initial contention-based stabilization phase in order to schedule the slot frames amongst all the neighbours in a 2-hops radius while avoiding overlapping. Another type of protocol employs Polling and Reservation techniques in which slots are reserved according to the demand. Examples of schedule-based, or contention free, protocols are mainly based on a TDMA approach since those based on frequency and code might increase the cost and power requirements. They can be classified as [33]: i) Priority-based [38], ii) Traffic-based [39], iii) Clustering-based (Sensor MAC (S-MAC) [40]) and iv) TDMA (Self-Organizing MAC for SensorNets (SMACS) [41]).

There are other issues to take into account when designing MAC protocols for WSNs such as: i) the type of network and topology, e.g. flat vs. cluster topologies, ii) the dynamics of the network, e.g. self-stabilizing to topology changes, iii) the complexity of the protocol, for instance in terms of messages, or iv) whether sleep scheduled mechanisms are employed to save energy. In addition, latency, throughput, robustness, scalability and fairness in the allocation of the channel capacity also need to be considered.

2.1.3 Significance for this Thesis

This section has presented a general overview of the area of Wireless Sensor Networks, providing context for the technology behind the architecture proposed in this thesis. With an emphasis on networking, WSN technology has been characterized in terms of the advantages and disadvantages posed by the type of devices employed. These are small devices, constrained in terms of resources, such as processing, memory and energy, which communicate wirelessly and can be unobstrusively integrated in the environment for sensing and actuating purposes. The range of pervasive applications which this technology empowers has been identified. Additionally, the new networking paradigms, which have arisen from the design of protocols for this type of error-prone wireless distributed networking scenario, have been described. This section has also highlighted common mechanisms and design issues around wireless technologies, while noting the particularities of research in wireless sensor networks.

2.2 Routing Protocols in Ad Hoc Wireless Networks

Routing protocols for wireless sensor networks have been the focus of much effort in recent years. Many of the approaches proposed have evolved from routing techniques in wireless networks and have been adapted to suit the requirements, and more importantly, the constraints of WSNs. When designing a routing protocol for WSNs, most of the design principles explained in Section 2.1.1 must be taken into account.

Different routing protocol classifications exists which try to group WSN routing protocols according to their functionality and operation. Al-Karaki and Kamal [32] suggests routing protocols for wireless sensor networks can be classified according to a) the Network Structure, b) the Route Discovery Process and c) the Protocol Operation.

Arising from the Network Structure, routing protocols can be classified as:

1. Flat-based: all nodes in the network are peers with the same relevance, functionality and duties.
2. Hierarchical-based: special nodes in the network have different functionalities and roles, and are usually responsible for coordinating the cluster of nodes, representing the cluster within the whole network, or aggregating data from nodes. These protocols aim to tackle some of the scalability and energy-inefficiency problems which may arise in flat-based networks routing protocols by organising the network into areas.
3. Location-based: routing decisions are performed based on the location information to send data to a certain region rather than spreading it inefficiently over the network.

Arising from the Route Discovery Process, routing protocols can be classified following the time when routes are discovered as:

1. Reactive (On-Demand): Protocols start their route discovery mechanisms only when there is a need for a node to start communication with any other node/s. Thus routes are created on-demand when there is data to be sent and a route does not already exists or is not (known to be) working. This mechanism has an impact on data communication latency when routes are unknown or fail and need to be discovered.
2. Proactive (Table-Driven): Routes are created at some point in time, which might be the beginning of the network's life, before they are needed to route data. Routes are created and

updated every so often at the cost of periodic maintenance messages. This type of routing protocol are characterised by their rapid response in delivering data (low latency) as routes are discovered beforehand. They are also characterised by having a high degree of robustness to node mobility at the cost of frequent transmissions update. However, when nodes within the network or a specific area do not need to transmit data periodically, this approach proves to be highly expensive in terms of communication and energy, as routes are maintained but rarely used.

3. Hybrid: Routing protocols employ mechanisms from both Reactive and Proactive route discovery methodologies.

Arising from the Protocol Operation, routing protocols can be classified as: Negotiation-based, Multi-path-based, Query-based, QoS-based and Coherent-based. Furthermore, as different protocols are suited to different applications, features like mobility tolerance, energy awareness and reliability are of significance in WSN environments and have to be considered when designing routing protocols.

An alternative classification arises from the larger number of devices in a wireless sensor network when compared to traditional ad hoc wireless networks. The high number of devices, which might overlap each other in the monitoring of an area or phenomenon, gives this technology a different communications paradigm where data and metadata become more important than the devices which generate it. This concept is known as "data-centric" and was introduced with a protocol called Directed-Diffusion [42]. "Data-centric is different from the "address-centric" concept where the address of the device primes in the communication process. In this regard, data-centric routing protocols need to be aware of the type of application data and the network characteristics for the design of the routing mechanism. Most of the protocols employing "data-centric" routing are classified as data dissemination protocols, where the main goal is to push data through the network towards a sink, or set of sinks, in a process where intermediate nodes do not need to be known and act as pure relays. Data-centric Routing Protocols for data dissemination include Directed Diffusion [42], SPIN [43], Rumor Routing [44] and Gradient-Based Routing (GBR) [45] which are presented in Section 2.3.3 and Section 2.4.5.

These classifications establish points of reference for defining the behaviour of a routing protocol and, therefore, its suitability for a particular range of applications in the domain of wireless sensor networks. However most of the routing protocols fit in more than one type of each proposed classification. One of the main research goals is to build a generic routing protocol capable of adapting to multiple network scenarios and different applications.

For instance, SPIN [43] and Directed Diffusion [42] are classified as flat-network, data-centric, query-based and negotiation-based, while Rumor Routing [44] and GBR [45] are characterised as flat-network, data-centric and query-based. AODV [46], DSR [47] and DYMO [48] are examples of flat-network, address-centric and reactive route discovery, while DSDV [49] is one of the first protocols to be classified as flat-network, address-centric and proactive route discovery protocols. Examples of hierarchical-network routing protocols are LEACH [50] and PEGASIS [51], while GPSR [52] is a good example of location-based-network protocols, also known as geographical routing protocols. Finally, two protocols which explicitly address QoS metrics are SAR [53], which is classified as flat-network and multi-path-based, and SPEED [54] which belongs to the group of location-based-network protocols.

2.2.1 Flooding and Gossiping

Flooding and Gossiping are two of the most commonly employed techniques in routing, dissemination and searching in computer networks [55]. While they are used in wired networks, their operation is described in the domain of ad hoc wireless networks where the broadcast property of the wireless medium impacts their functionality.

Flooding [56] in wireless ad hoc networks is a dissemination mechanism whereby nodes broadcast packets on reception of a new, or duplicate, packet. The flooding process can be limited by including a Time-To-Live (TTL) parameter in the packets where the TTL value is decremented at each node according to node configuration. Each node keeps a record containing information on the packets which have been broadcast. Flooding the network typically has a high dissemination factor, i.e. the number of nodes in the network which receive the data, and the operation can be affected by the dynamics and density of the network topology. However, flooding encounters three main problems: 1) Implosion, where the same data is received multiple times, thus wasting resources, 2) Overlap, where sensors measuring the same phenomenon send redundant data, thus wasting resources due to data duplication, and 3) Resource Blindness, where nodes do not modify their activities according to the current state of their resources. These problems have been extensively studied for WSN, not only in terms of how to avoid them, but also how to benefit from them.

Gossiping [57] is a dissemination technique which forwards a packet to a randomly selected neighbour, thereby reducing the energy consumption. Unlike flooding, the information is disseminated slowly and may not reach all the nodes in the network. When adapting gossiping techniques to unreliable ad hoc wireless networks, where neighbour nodes might not be known or simply not available, a different perspective can be adopted. The authors in [57] took advantage of the broadcast property of wireless networks to define gossiping as the probability of a receiving node broadcasting a received message. Nodes on reception of a packet decide whether to broadcast it with a probability "p" or discard it with a probability "1-p". The probability of broadcasting a packet can be calculated as a function of the resource levels and overheard packets at the node. This way the main flooding problems are ameliorated in an efficient manner, although not completely solved.

2.2.2 Reactive Routing Protocols

Reactive routing protocols, like Ad hoc On-Demand Distance Vector (AODV) [46], Dynamic Manet On-Demand (DYMO) [48] and Dynamic Source Routing (DSR) [47], have been used for traditional wireless ad hoc networks. Such protocols work on an on-demand basis: communicating only when there is data in the network to be transmitted. Although these protocols are not suitable "as is" for wireless sensor networks (because of the differences between the technologies), the reactive approach offers a good solution for avoiding the energy-greedy, periodic beacons used by proactive protocols. These protocols need to be adapted to suit the constraints of WSN. In the area of address-centric reactive routing protocols for flat-networks, AODV, DSR and DYMO establish the basic mechanisms employed in the design of new on-demand protocols for WSN. However every new routing protocol tries to optimise the network performance according to the design principles of WSN presented in Section 2.1.1.

2.2.2.1 AODV

The Ad hoc On-Demand Distance Vector (AODV) [46] routing protocol follows the reactive paradigm of discovering routes when communication with a node is requested. In the route discovery phase (see Figure 2.1), the network is flooded with Route Request (RREQ) packets such that multiple paths towards the destination are created in the routing tables of the intermediary nodes. The flooding process is controlled by avoiding cycles while broadcasting only the first RREQ packet arriving. Duplicates are detected and discarded as every packet contains the source address and sequence identifier which is incremented with every RREQ packet issued. In addition, the destination address and the hop counter are carried in the packet as well as the most recent sequence number the node has for the destination. Intermediate nodes can reply to the RREQ only if they have a route to the destination with a greater or equal destination sequence number. When a RREQ packet arrives at the destination, or at an intermediate node which contains a route to the destination, a Route Reply (RREP) packet is sent on the reverse route towards the source node. The RREP packet is sent back to the node from where it was received (unicast), such that the routing table is queried for the next hop on the way back and also updated with the received packet address as the next node on the path to the destination. This process establishes a bidirectional route and sets a timeout value for the route entry to expire. If more than one RREP packet is received at the same node, only those with a higher sequence or those which have a smaller hop count are forwarded. Once the bidirectional route is established - note that AODV requires symmetric links - a route maintenance phase starts where packets can be sent over the path by relaying the packet according to the routing table information. Every time a packet traverses the route, the timeout is reset to indicate the route is still valid. In the event of an intermediate node in the path failing or changing position, the route will break. In this case a mechanism based on periodic hello messages detects the missing node and a special RREP packet, also known as Route Error (RERR), is unicast to alert the nodes along the upstream of the failing route towards the source/s node (see Figure 2.2). Another mechanism employed to detect whether the next node is available and has received the packet make use of the link-layer acknowledgements at every node in the downstream. When a RERR arrives to the source node, a new route discovery process (RREQ) is triggered. Another possibility is to locally repair the route such that a new RREQ message is broadcast, with a time to live (TTL) in terms of hops, in order to discover new routes from the failing point to the destination node. Once the destination replies to the intermediate node, the communication between the original source and the destination is resumed. Azzuhri et al. studied the performance evaluation of the local and source initiated repair mechanisms to compare the efficiency of each approach [58].

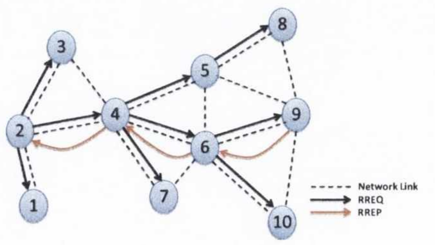


Fig. 2.1: AODV - Route Discovery

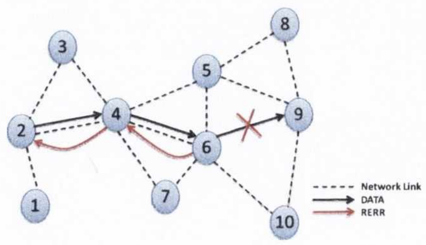


Fig. 2.2: AODV - Route Error

2.2.2.2 DYMO

The Dynamic Manet On-Demand (DYMO) [48] routing protocol operates under the same basic concepts of AODV. In fact, DYMO was conceived as a lighter, simplified version of AODV, for Mobile Ad hoc NETworks (MANETS). DYMO does not implement the local route repair mechanism employed in AODV, but rather repairs the route from the source when RERR packets are received. The option of intermediate nodes replying with a RREP when they know a fresh route to the destination is optional in DYMO, while for AODV it is the default mechanism. Even though DYMO does not impose a mechanism to monitor the state of the route, a timeout value for the routing table entries is employed. An RREP message is broadcast when the corresponding entry of an intermediary node, which is part of the path being traversed, has timed out or does not exist anymore. Timeout values are refreshed every time a packet is received or sent. When detecting a failing link, an RERR packet is broadcast to inform those nodes containing the failing node in their routing table entries (see Figure 2.4). Instead, AODV unicasts the RERR packet through the upstream path towards the sink/s by following the precursor list. DYMO has an option to accumulate the intermediate nodes addresses in the RREQ and RREP packets such that paths are transported in each packet and intermediate nodes can populate their routing tables with paths to different destinations (see Figure 2.3). This mechanism can be expensive in terms of message size as the number of hops increases. DYMO and AODV have been tested under different mobility conditions [59] and DYMO performs better in terms of packet delivery ratio, throughput, and control message overhead when the mobility speed increases. However AODV is more efficient in terms of energy consumption when the mobility speed is low.

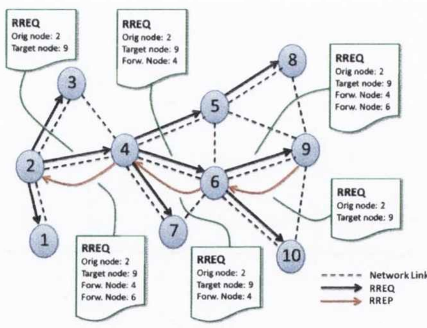


Fig. 2.3: DYMO - Route Discovery

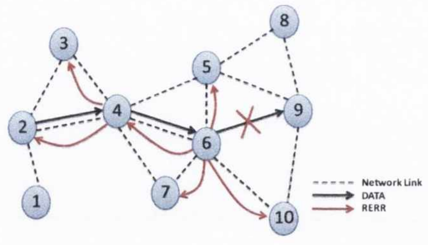


Fig. 2.4: DYMO - Route Error

2.2.2.3 DSR

The Dynamic Source Routing (DSR) [47] protocol describes some of the basic mechanisms of reactive routing for multi-hop wireless ad hoc networks; for instance the route discovery and route maintenance mechanisms employed in the AODV routing protocol. As opposed to AODV, DSR accumulates the path traversed by RREQ packets in the the packet itself (see Figure 2.5). The accumulated path also serves as a mechanism to detect loops and to discover routes to other nodes. The accumulation path mechanism is also optional in DYMO. Multiple RREP packets are issued by the destination node which traverse different routes back to the source node. The source stores the path/s to reach the destination; the decision on which path to store might be taken in terms of hops). When data needs to be sent, the packet contains the full path which needs to be traversed on the way to the destination and back (as bidirectionality is assumed). By using inter-node acknowledgements, end-to-end acknowledgements, or per-hop implicit acknowledgements, a broken link can be detected and an RERR packet is unicast towards the source node; this packet follows the reverse path contained in the packet (see Figure 2.6). Bisoyi et al. have evaluated the performance of DSR, AODV and DYMO in small scale networks under different number of nodes, speed and pause times [60]. DYMO performed better than DSR and AODV with respect to Quality of Service parameters such as throughput, packet delivery ratio, delay and normalized routing load. However AODV performs better in terms of routing overhead.

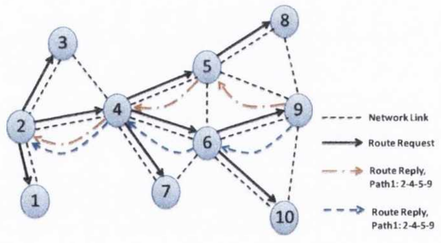


Fig. 2.5: DSR - Route Discovery

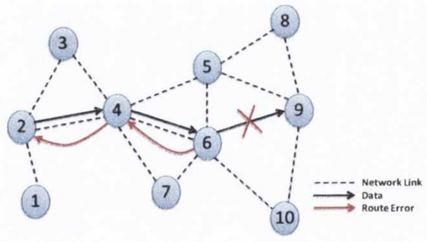


Fig. 2.6: DSR - Route Error

2.2.3 Energy Efficiency and Reliability in Routing Protocols for WSN

In Wireless Sensor Networks, two of the most important factors to consider when designing routing protocols are the reliability of the system in terms of communication and the efficiency of the protocol in terms of energy at node and network level.

Energy Efficiency is an important issue for wireless sensor networks due to the energy constraint of the sensor devices. Different techniques have been adopted to reduce the power consumption of both the motes and the WSN as a whole. Jayashree et al. define a taxonomy with three main schemes for energy management [61]: i) Battery Management, ii) Transmission Power Management, and iii) System Power Management. These include strategies such as sleep mode operation whenever possible, avoiding retransmissions, minimizing control packet size and using power efficient error control techniques [62]. In tandem with this, many solutions have been proposed to increase the life of the network by avoiding network partition [63]. These include: i) Routing oriented to traffic [64], ii) Probabilistic methods to select the next hop based on energy [65], iii) Awareness of the level of energy needed not only to perform routing but to keep routes working [66], and iv) Energy-based multipath routing to balance the network communication [67].

Furthermore, cross-layer designs are employed as a solution to optimise the use of energy across multiple layers, from data link and routing layers to application layer. Finally, the on-demand paradigm of routing protocols seeks to avoid the energy-expensive periodic beacon messages of proactive routing protocols [68] and it should be used when possible or in combination with the table-driven paradigm to reduce energy consumption.

Reliability is another concern in a WSN environment. The wireless medium implicitly gives rise to problems related to contention and fluctuations in the quality of data transmission. These can result in data delivery failures where links might be intermittently operative. Whilst some of these problems are addressed at the link layer, reliability mechanisms at the network layer are necessary for link connectivity and end-to-end packet delivery.

Protocols like Multipath On-Demand Routing (MOR) [69] offer reliability at the link layer by storing alternative paths in every node and performing local retransmission in the event of link failure. This approach uses a reactive mechanism that keeps track of the transmission status of every packet. It uses flooding mechanisms to discover routes and unicast data packets once a route is created. MOR has been layered over IEEE 802.11, where it performed reliably and in a more energy efficient way than AODV and a modified version of DSR. The drawback of this protocol is that multiple routes have to be retained in the routing table to provide reliability. Other protocols use multiple paths to help ensure reliability to some degree. Reliable Information Forwarding Using Multiple Paths in Sensor Networks (ReInForM) [70] sends the same packet over different routes to achieve a degree of reliability that is dependent on the importance of the packet. It utilises a reactive approach and floods the network periodically to identify the neighbours of each node.

Another protocol which achieves reliability, using local repair mechanisms, is Single-path With Repair routing scheme (SWR) [71]. In SWR a quick local repairing mechanism is launched when a broken link is detected. This uses information available in the immediate neighbourhood of the sender. The sender overhears the receiver transmission and detects whether the receiver sends the message following the route or not. If the receiver has not sent the packet after a certain period, the sender starts a neighbourhood discovery to find another node that has the routing information required to reach the destination. If no node is found, the packet is returned to the origin node to indicate that a new discovery process needs to be launched; otherwise the new node is used to reach the destination.

Reliable Energy Aware Routing (REAR) [72] protocol is described as using a reactive approach. REAR achieves reliability by creating two disjoint paths, between sink and source, in the discovery process. The first route created will be used as a service path which performs unicast transmissions. The second is created as a backup path. In the event of a failure in the service path, a message is broadcast to both the sink and the source to indicate that the backup path should be used and a new service path must be discovered. REAR is energy-efficient as it discovers paths using flooding mechanisms that determine if a node can be part of a route based on its levels of reserved energy. Using an estimation of the quantity of data to be transmitted from the source, nodes reserve the energy necessary to perform routing for each route they are part of. If the node is energy-sufficient, the flooding packet is forwarded. The path selected is usually the one with the largest available energy. REAR increases the overhead when links are not bidirectional as new service routes have to be discovered and it does not incorporate a link repairing system.

2.2.4 Significance for this Thesis

This section has introduced the different classifications of routing protocols for wireless sensor networks according to the network structure, the protocol's route discovery process and the protocol operation. A special type of classification for WSN has been presented which employs data and metadata for routing purposes, i.e. data-centric routing, as opposed to the traditional address-centric routing. The protocols presented herein can be identified as supporting both data and address centric routing paradigms, having a reactive route discovery process and forming a flat network structure. The key mechanisms and algorithms representing the flat-network reactive routing concept have been presented, with an emphasis on communications reliability and energy efficient design.

2.3 Gradient-based Routing in Ad Hoc Wireless Networks

2.3.1 Gradient Definition

The word "gradient" can be defined as:

According to the Oxford English Dictionary (OED):

- *"A continuous increase or decrease in the magnitude of any quantity or property along a line from one point to another; also, the rate of this change, expressed as the change in magnitude per unit change in distance."*
- *Of a road or railway: "Amount of inclination to the horizontal; degree of slope."*
- *In Mathematics: "A vector function whose components along the co-ordinate axes are the partial derivatives with respect to the corresponding variables of a given scalar function; it is denoted by ∇f or by grad f, where f is the scalar function."*
- *In Geometry: "The degree of steepness of a graph at any point, measured by the tangent of the angle between the horizontal axis and either the line (if straight) or the tangent to the curve."*

According to the Merriam-Webster Dictionary:

- *"The rate of regular or graded ascent or descent: inclination."*
- *"A part sloping upward or downward."*
- *"Change in the value of a quantity (as temperature, pressure, or concentration) with change in a given variable and especially per unit distance in a specified direction"*
- *"The vector sum of the partial derivatives with respect to the three coordinate variables x, y, and z of a scalar quantity whose value varies from point to point."*

The method of "Steepest Descent" also called "Gradient Descent", allows for the progression from any point in the gradient to the closest local minima. According to Wolfram MathWorld [73], the "Method of Steepest Descent" is defined as:

- *"An algorithm for finding the nearest local minimum of a function which presupposes that the gradient of the function can be computed. The method of steepest descent, also called the gradient descent method, starts at a point P_0 and, as many times as needed, moves from P_i to P_{i+1} by minimizing along the line extending from P_i in the direction of $-\nabla f(P_i)$, the local downhill gradient."*

If the gradient is formed in such a way that local minimum points do not occur, the steepest descent algorithm will progress until the global minimum is reached, i.e. the originator of the gradient.

2.3.2 Basis of Gradient-based Routing

In denoting routing protocols in computer networking, the "gradient" concept is employed to describe an overlay network in which every node has an associated quantity value with respect to a node known as "sink". Routing of packets is achieved by decreasing the quantity value at the next node in the path while descending the gradient towards the sink node, i.e. relaying the packet to those neighbour nodes which have lower quantity values with respect to the sink. For instance, Figure 2.7 gives a visual representation of the gradient flowing toward the originator at point (0,0). A set of vectors indicate the direction towards point (0,0) and the concentric contour lines shows the radius distance as the quantity value with respect to the origin of the gradient.

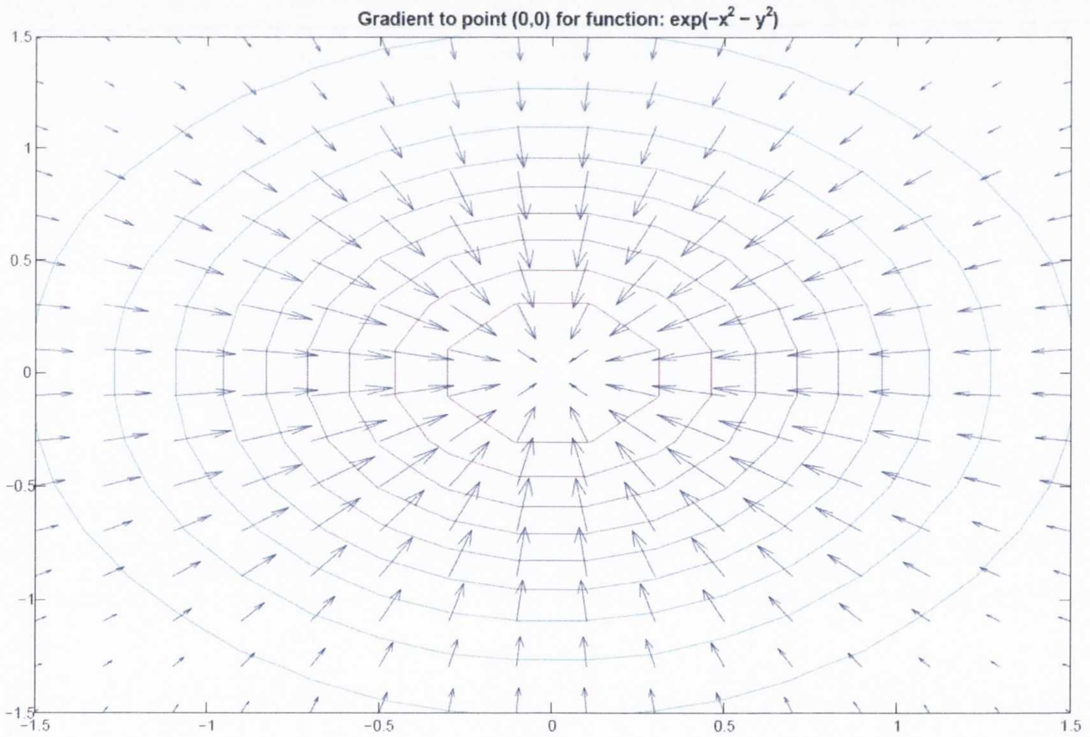


Fig. 2.7: Gradient field example with sink at position (0,0). Arrows indicate the direction towards the sink. Concentric circles indicate the radius distance with respect to the sink.

Gradient-based routing protocols are employed as mechanisms for data collection in multi-hop mesh networks where a source node, i.e. the sink, spreads its gradient through the nodes in the network. This process creates a Directed Acyclic Graph (DAG) where every participating node has an associated value with respect to the root node, i.e. the sink, which is ultimately reached by following the direction of the graph in a process called "gradient descent". The concept of sending data towards a central node (with higher capabilities) is frequently employed in ad hoc networks for network data processing or for the purposes of pushing data outside of the network.

One of the first protocols employing the gradient routing concept for ad hoc wireless networks was described in the year 2000 with the paper "Gradient Routing in Ad Hoc Networks" using the acronym "GRAd" [74]. Around the same time, three of the first protocols designed to operate in wireless sensor networks employing the concept of gradient as a way to disseminate data were presented as "Directed Diffusion" (DD) [42, 75], Gradient Based Routing (GBR) [45] and GRAdient Broadcast (GRAB) [76, 77]. Since then, gradient-based routing schemes have been one of the most widely used mechanisms for reporting data in multi-hop networks. For instance, the IETF Routing Over Low power and Lossy networks group (ROLL) was formed in 2010 to standardize a generic routing protocol for this type of networks. RPL, an IPv6-compliant protocol which stands for Routing Protocol for Low power and lossy networks [12], has been designed employing gradient-based routing concepts as the basic mechanism for routing data.

In gradient-based routing protocols, two phases are well defined: i) gradient setup, where the overlay network, i.e. the gradient, is formed, and ii) data forwarding or gradient descent, where packets are routed by a forwarding process which follows the gradient path towards the sink node.

The gradient setup phase deals with the formation of the gradient-based overlay network and needs to be launched before nodes can start sending packets to the sink node. Starting from the sink, packets are broadcast in such a way that every node receiving a packet, updates its routing table and (re)broadcasts the packet. The common mechanisms to discover paths in routing protocols such as flooding, periodic broadcast, and, to a certain extent, gossiping are employed. Depending on the nature of the broadcasting mechanism, a node might decide if and when the packet needs to be broadcast. When a broadcast packet is received, the routing table is populated with the address of the next node in the path to reach the sink, i.e. the node from which the packet was received, and also with the quantity value associated to the cost of selecting that node as the next hop. The cost value, known as the "height", is the key to forming the gradient and is calculated by a cost function at each node. The cost function calculates the height of the current node according to the received height. The new node's height is then encapsulated in the packet to broadcast. The different metrics determine the expression of the cost function which governs the gradient formation. For instance, metrics such as hop count, node's remaining energy, link quality indicators, physical distance or even fastest response are employed separately, or in combination, forming different routing overlay networks in terms of efficiency. Gradient-based networks are initially launched by the sink node and their structure can be updated when a neighbour node is found which reduces the current height value. Alternatively, the sink node can launch the formation of the gradient at any point in time to update the routing overlay network. When a gradient is spread, routing tables are populated with the unique combination of the sink node address and the gradient sequence value. This way, newer gradient formations can be identified over older entries in the routing table and updated accordingly. However, special mechanisms are required for those nodes which do not receive the new gradient packet; older and newer entries need to coexist for a loop-free routing process. Gradient-based routing protocols usually scale well as there is no need to store a large quantity of information in its forwarding table. On the other hand, the gradient-based network might be affected by an excess of transient nodes, demanding a new gradient formation. A "good" formation of the gradient implies avoiding local minima points which might produce a situation where a data packet can not progress towards the global minima, i.e. the sink. Backtracking and local repair methods are used to progress from local minima points. However, local minima points can be avoided or minimized in the gradient setup phase. In this phase, avoiding loops is one of the major issues to address. Still, the algorithm needs to be prepared for the situation where nodes fail thereby creating a local minima point. The ideal scenario where a gradient has no local minima and only one global minima, the sink, can be seen in Figure 2.8 with different points having heights in the range of 0 to 3.

Once the gradient is formed, packets can be routed from those nodes in the network which belong to the gradient overlay network of the sink. Whether the packet is originated from the node or the node is acting as a router, the packet is either sent to the next hop address according to the routing table or broadcast such that only neighbours with lower heights keep on broadcasting. This phase is known as the data forwarding phase or gradient descent as data is sent to the sink by descending its gradient via reducing the height to the sink in each routing hop. The height to the sink is contained in the routing table of each node and is employed in the data forwarding phase to decide whether the packet needs to be forwarded according to the cost value carried in the packet. The routing table could contain none, one, or more than one possible next hop neighbours for the same gradient. In this case a decision process based on the available information can be employed to select the most efficient next hop neighbour. On the other hand, the next hop address at the routing table might not be contactable or available to continue forwarding data. In this case, local recovery mechanisms

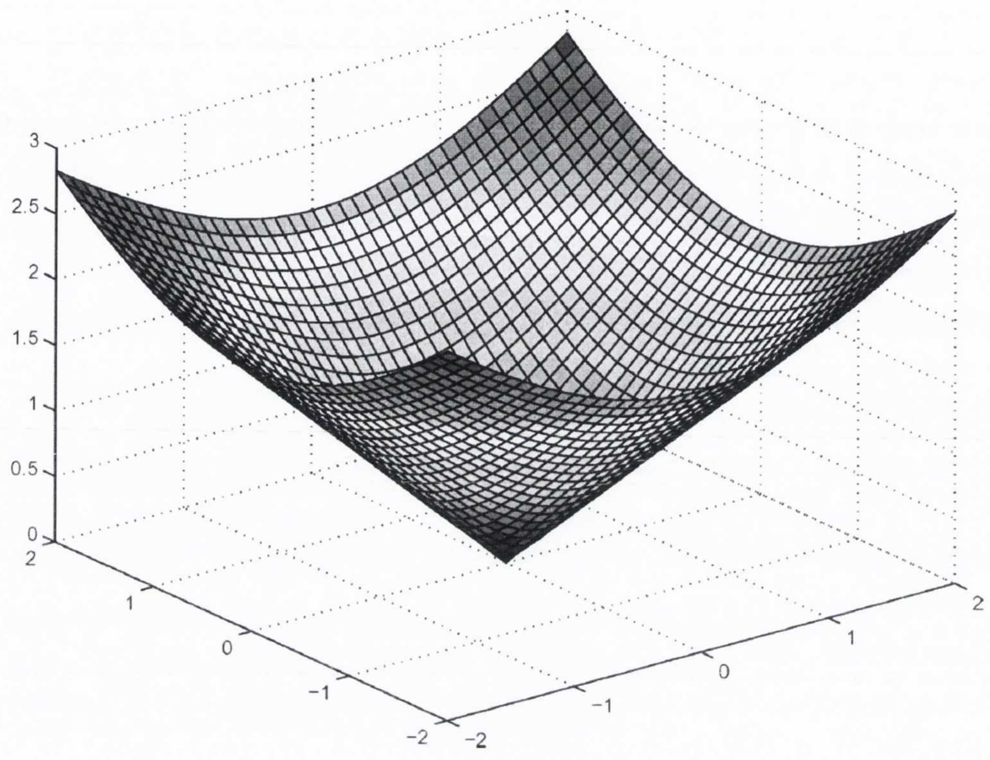


Fig. 2.8: Gradient field in 3D with no local minima points and one global minima, i.e. the sink. Height is represented in the axis which ranges from 0 to 3 [77].

are employed to locate a valid neighbour to replace the failing router node. These mechanisms use acknowledgement packets to detect node availability. Node unavailability might be due to the node not being capable of acting as a router; for instance this may happen when the level of its available resources - such as memory, processing capabilities, queue status or energy remaining - is below a certain threshold. Many gradient-based routing protocols broadcast the packet to their neighbours such that these are left to take the decision whether to keep on forwarding the received packet. Despite being a memory efficient solution, decision mechanisms at each node need to be implemented to avoid multiple redundant transmissions which increase the use of the medium and waste energy.

A considerable number of approaches to routing have been presented in the literature employing the concept of gradient formation and efficient forwarding mechanisms to route data towards the sink. Many of these protocols employ a unidirectional approach where data is routed from nodes to only one sink node. Some of them employ clustering algorithms to designate the group of nodes belonging to the sink gradient overlay network. In addition, routing protocols with multiple sinks are also proposed and mechanisms to evaluate the current network performance are employed to optimise the routing process. In most of the gradient-based protocols, a combination of the mechanisms of the gradient setup phase and the data forwarding phase impacts on the way the protocol operates. For instance network load balancing is addressed in gradient-based routing protocols by exploiting information and taking decisions at the gradient setup phase and/or the data forwarding phase. When balancing the network traffic, the use of the available resources from the nodes in the network tends to be evenly distributed. Balancing the traffic load according to the level of resources is highly beneficial in constrained wireless sensor networks where network partition can be produced when the energy of a set of strategic nodes is depleted, leaving areas of the network disconnected. In addition, traffic load balancing ameliorates the congestion and avoids higher latencies in multi-hop routes which might also produce low delivery ratios.

Two categories of techniques employed by gradient-based routing protocols can be defined according to the type of traffic load information to be exploited in the forwarding decision at each node [78]: i) 1-hop neighbourhood traffic load information and ii) cumulative traffic load information over the gradient climbing path. Decisions are typically made based on the energy remaining or forwarding queue length information. In the first category, this information comes from 1-hop neighbour nodes and the decisions are taken once the gradient is formed. The gradient is created based on the energy remaining or the hop count at each node. Examples of this category are SGF [79] and PB-routing [80]. However in the second category, a cumulative value is calculated in each previous hop of the path when the gradient is being established. The cumulative path cost at each node is calculated with the value from a received packet by typically following different weighted average formulas. In the first category of load balancing methods the decision is taken based on local information and affects the load balancing of a reduced scope in terms of hops. However, in the second category the impact on the traffic load balance broaden as the decision information is the result of an aggregation of single decisions in each of the previous nodes forming the gradient. However, when using the cumulative method, climbing down the best cost effective gradient path does not guarantee that one of its nodes is not one of the most energy depleted or highly congested in the network. Yoo et al. present GLOBAL [78] as a gradient-based routing protocol capable of balancing the traffic load by selecting the least-loaded path which does not contain the most overloaded sensor. GLOBAL employs a weighted average of the cumulative path and traffic load of the most overloaded nodes in the path to establish its gradient.

To clearly understand the basic mechanisms behind gradient-based routing, the following section presents a comprehensive list of routing protocols in this category covering a wide range of approaches to solve some of the most typical routing problems.

2.3.3 Gradient-based Routing Protocols

In this subsection a set of the most relevant gradient-based routing protocols are chronologically presented and their underpinnings explained. Earlier designs of gradient-based protocols established the basis for gradient routing and therefore are well described next. Later approaches focussed on optimising the performance of the basic mechanisms for gradient-based routing to adapt to different applications and scenarios. The majority of the gradient routing protocols presented here have been designed to operate in ad hoc wireless sensor networks.

2.3.3.1 GRAd (2000)

One of the first works to describe the basis for gradient-based routing protocols in ad hoc wireless networks was published by Robert D. Poor in 2000 under the name "Gradient Routing in Ad Hoc Networks" [74]. The gradient-based protocol "GRAd" belongs to the reactive category of ad hoc routing protocols where routing information is established on-demand. In this type of routing protocols, the general rule is "if you want to be spoken to, you must first speak". Grad assumes that links between nodes are symmetrical but it works well in wireless real-world scenarios where partial symmetry exists. Only broadcast packets are employed in the communication. Every packet is uniquely identified as it carries the originator address and a sequence value. In addition, it contains the message type (M_DATA, M_REQUEST), the target address, the "accrued cost" - which starts in 0 and is incremented by one "hop" with every broadcast at each relaying node - and the "remaining value" - an initial estimation of the number of hops to the destination which is decremented with every broadcast at each relaying node. Two phases are clearly differentiated, "Message Origination and Relaying" and 'Reply and Request". The "Message Origination and Relaying" (M_DATA) phase is performed when a node A wishes to communicate to a destination node B and it contains a cost value in the so called "cost table" with the number of hops to reach the destination. In this case, a packet is broadcast with the "remaining value" as the estimated cost value stored in the cost table. When a node receives a packet, the decision of forwarding the packet, i.e. broadcast the packet, is made based on the cost tables. If the cost value in the table is less than the "remaining value" carried in the packet, then the packet is broadcast. Loops are avoided as the "remaining value" is decremented by one before broadcast and it is also discarded when the value reaches 0. However, this mechanism proves to be expensive in terms of communications when there is a high density of nodes within range. All nodes with a lower cost value broadcast the received packet, increasing the medium access contention and wasting energy in the transmission and reception. On the other hand this increases the delivery ratio in the situation where some nodes are in sleeping mode. In addition, the author employs implicit acknowledgements to stop a node from broadcasting a packet when an instance of the same packet with a lower cost value is overheard. The "Reply and Request" (M_REQUEST) phase is triggered when a node A needs to send a message to a destination B for which routing information is unknown. By the process of flooding the network, where every node only broadcasts the first packet received, a gradient is formed in terms of hop distance cost to the originator by storing the "accrued cost" or height in the cost table. If another packet is received with a better "accrued cost", i.e. lower hop distance, the value is updated in the cost table but the packet is not broadcast again. Depending on the initial "remaining value", the packet may not arrive at the target id. If this occurs, timers are in place to try again with a higher "remaining value". When the packet is received by the target node, the "accrued cost" of the received packet is used to populate the "remaining value" of the packet to be broadcast with the originator id as the target. This way, the packet traverses the route back to the originator in the same way a data packet does, by the mechanism of broadcasting and decrementing the "remaining value". The propagation of the message through the network establishes and updates reverse path routing information to the originator and employs the "accrued cost" value to create a gradient to the destination node B; however the gradient information for B is only stored at the cost tables of those nodes participating in the route back to the originator node A, including those which overhear the packet. Each entry at the cost table has an expiration time which is refreshed every time information for this gradient is received. The authors simulated GRAd under dynamic network topologies and it proved to work well due to the opportunistic update of cost tables. This mechanism produces shorter and longer routes and the possibility of increasing the "remaining value" field when end-to-end acknowledgements are not received. GRAd was tested operating on top of a CSMA/CA IEEE 802.11 MAC layer with similar results. The authors suggest that a compromise between relaying packets to a fixed neighbour, solution employed in protocols such as AODV [46]), and to all the neighbours, solution employed in GRAd, shows some promise. Finally, the paper describes the possibility of using functional addressing in which a node's target is replaced by a predicate which is evaluated by those nodes receiving the request message (M_REQUEST).

2.3.3.2 GBR (2001)

One of the first papers to develop a routing protocol employing the gradient concept for ad hoc wireless sensor networks appeared in 2001 under the title "Energy Efficient Routing in Wireless Sensor Networks". In this work, the Gradient-Based Routing protocol (GBR) [45] is presented under the assumption that optimal routing in sensor networks is infeasible. GBR employs a special packet called "Interest" and a data packet "Data". The sink floods the network with "Interest" packets at the beginning of the network life. The flooding process establishes a gradient from every node towards the sink which is based on the hop count as a measurement of the height of each node. In the forwarding process every node forwards the packet to a neighbour node with a lower height value towards the sink. Nodes satisfying the requirements in the "Interest" packet send their "Data" packets to the sink by descending the established gradient. In this approach, energy efficient techniques like in-network data processing with data combination are used. To distribute the network load, the process of selecting the next node in the gradient descending phase is done in a stochastic fashion when two or more nodes with the same height are available. The protocol also implements another approach to balance the traffic load in which a node informs its neighbours, except for the one from which the stream is coming, if its height value has increased. This reduces the node's involvement in future data streams when its battery is running low, or if its stream load is too high, and diverts the future streams along other paths.

2.3.3.3 A Scalable Solution to Minimum Cost Forwarding in Sensor Networks (2001)

Ye et al. propose a scalable solution to minimum cost forwarding in large sensor networks [81]. The paper presents an algorithm to construct an efficient gradient in terms of forwarding cost which employs a controlled flooding where sensors only broadcast each unique gradient discovery packet once. In order to wait for your neighbourhood to finish transmitting, each node receiving an advertisement packet (ADV) sets a timer to wait before broadcasting its ADV packet. The ADV packet broadcast contains the best associated cost from all neighbours' packets previously received. A backoff-based scheme is employed which introduces a delay in the formation of the gradient and avoids multiple retransmissions of the advertisement packet by the same node in an effort to optimise the gradient formation. The backoff timer delay is a function of a constant multiplied by the cost value of the link between the sender of the packet and the neighbour receiver node. The cost value for each pair of nodes's link is calculated as the energy needed to transmit from the sender to the receiver according to the physical distance between them and the node's communications range. If the constant is too low, they will need to rebroadcast again if a packet with a lower cost value is received at a later stage.

When descending the gradient, the cost value for the optimal route calculated during gradient setup is encapsulated in the packet to be broadcast. By matching the received cost value in the packet, minus the cost added by the local node, with the local cost value to the sink, the receiving node knows if it is part of the optimal path and therefore must broadcast the packet. When ADV packets are lost in the gradient formation, the optimal path considering all network will not be formed. In the event of a node failure, the authors propose to increase the cost value at the packet in an effort to overcome the gap left in the optimal path.

2.3.3.4 GRAB (2001-2005)

In the paper "A Scalable Solution to Minimum Cost Forwarding in Large Sensor Networks" [81], Ye et al. proposed a backoff-based method for efficient gradient formation. An enhanced version of their protocol was presented later as Gradient Broadcast (GRAB) [76, 77], a gradient-based routing protocol which provides a robust mechanism for data forwarding when descending the gradient. While keeping the gradient setup mechanism according to the first design [81], where the energy consumed in each link is employed as the cost value, the gradient is also refreshed based on the variation of each source historic profile. Information for each packet received is stored in the sink to account for changes in variables such as source delivery ratio or source average budget value. These metrics indicate whether change in the topology has occurred and thus a new gradient setup is required.

GRAB employs a robust forwarding algorithm which makes use of the multiple number of paths available when descending the gradient. Nodes in GRAB contain an estimation of the energy cost to send a packet to a neighbour, which is computed in the gradient setup phase employing measurements such as the Signal to Noise Ratio, and therefore each node stores an energy cost to the sink. Each node receiving a broadcast data packet forwards the packet if the local cost value to reach the sink is lower than the value in the packet, which is updated at each router node. This mechanism makes use of multiple interleaved paths to forward the packet which increases the reliability at a cost in redundancy and therefore transmission and energy. Depending on the number of nodes and the energy costs, a "band" containing all nodes involved in the transmission from source to sink is formed (see Figure 2.9). GRAB controls the width of this band in order to increase the delivery ratio with a credit-based value generated at the source node and transported in the packet. The credit value is an extra budget which can be added to the energy cost at each node when deciding if the packet is to be forwarded. The packet can be broadcast over the width of the band and down to the sink for as long as the forwarding nodes energy cost is lower than the sum of the energy cost of

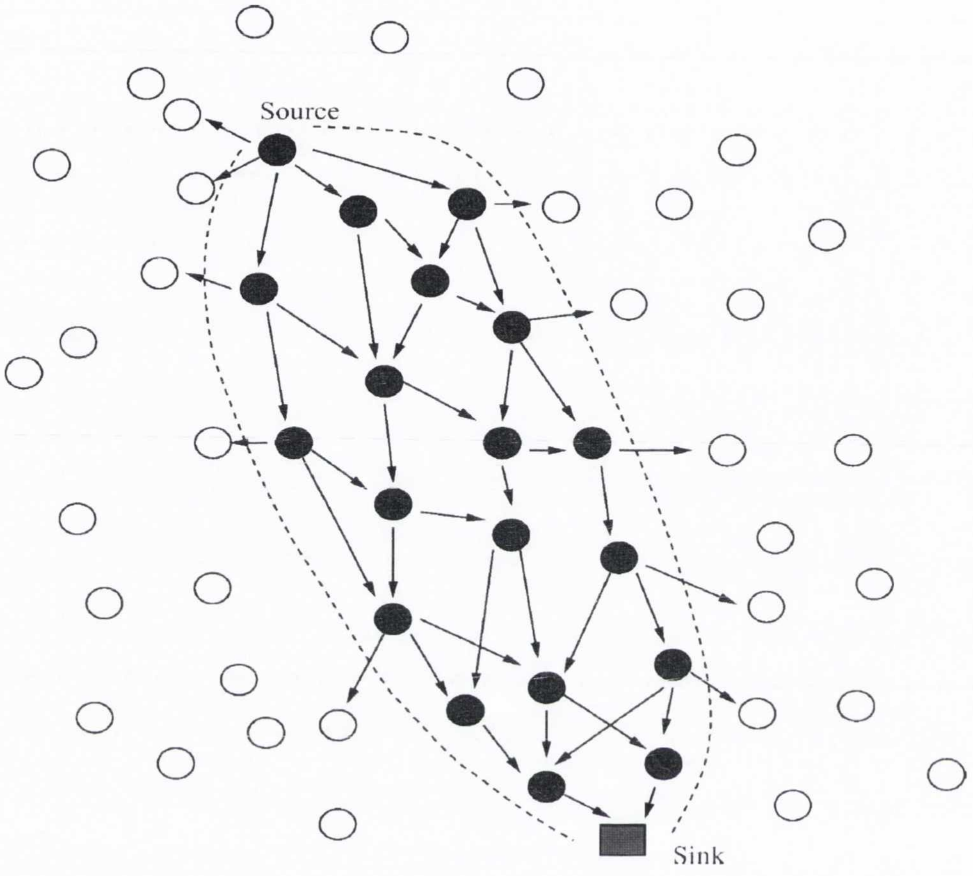


Fig. 2.9: Forwarding Band composed of Multiple Nodes which participate in the communication from Source to Sink in the GRAB routing protocol [77].

the packet and the credit value from the source node. Nodes can tune their transmission power to control the range of the broadcast. GRAB has been simulated under different node failure, packet loss, density and scalability conditions indicating a delivery ratio of around 90%, depending on the credit value, which might drop to 50% when transient nodes fail suddenly.

In works by Jaffres-Runser et al., three optimisations of the forwarding phase of GRAB have been proposed as "Probabilistic-GRAB" (P-GRAB) [82], "Utility-GRAB" (U-GRAB) [83] and "Utility and Probabilistic GRAB" (UP-GRAB) [84], which take into account interference and congestion metrics. In GRAB when descending the gradient a considerable number of redundant packets flowing through multiple paths contribute to increase the reliability of the packet delivery. However, Jaffres-Runser et al. decrease the number of forwarded data messages by employing cooperative (U-GRAB and UP-GRAB) and non-cooperative (P-GRAB) decision making. Decisions are made based on the available information at each node with the goal of improving the robustness-latency-energy trade-off. P-GRAB does not interact with neighbours - instead each node takes broadcasting/forwarding decisions based on the interference distribution of the neighbourhood. P-GRAB assigns higher forwarding probabilities to nodes which are part of an area where less interference has been produced. On the other hand, nodes interact in U-GRAB to gather information from the neighbourhood on energy level and channel occupancy. A heuristic utility function based on pricing concepts is employed for the decision, taking into account current channel congestion. UP-GRAB aggregates the benefits of both policies by combining the interference impact metric with the channel congestion. The schemes outperform GRAB in terms of the average transmission delay, reducing it by up to two times. P-GRAB is the best approach in terms of robustness and energy consumption in networks with high reliability of nodes. U-GRAB with its cooperative congestion-based approach is the best option for unreliable networks where collisions need to be avoided as much as possible.

2.3.3.5 GLIDER (2005)

In "Gradient Landmark-Based Distributed Routing for Sensor Networks" [85], GLIDER is presented as a gradient routing protocol where each node contains a lightweight global vision of the routing information. The authors assumed that nodes in a sensor network can disappear but the main global topology is maintained. The protocol creates an overlay of landmarks, which are nodes acting as reference points for its tiles, i.e regions of nodes with a landmark node virtually representing it. The network is partitioned into tiles which are represented by a landmark using the landmark Voronoi complex. Within each tile, traffic is balance using greedy routing or local coordinates based on the square distances of hops - using an average distance of the landmarks - with respect to a set of landmarks. A virtual routing overlay of landmarks is abstracted by applying combinatorial Delaunay triangulation. This creates an efficient graph representation of a higher overlay which is disseminated to the nodes in the network, thus every node has a global vision. In every tile, a node routes through another node by descending the gradient towards the next tile which belongs to a landmark point closer to the destination. Therefore, routing is executed at a higher level from tile to tile by employing greedy gradient descent over a path composed of a series of tiles; the gradient descent process utilised the lightweight global information contained in each node.

2.3.3.6 A New Gradient-Based Routing Protocol in Wireless Sensor Networks (2005)

Xia et al. employ gradient-based mechanisms with the goal of prolonging the network lifetime by optimising the energy consumption [86]. A combination of hop count and remaining energy at each node is employed to calculate the cost value when creating the gradient. In addition, a backoff mechanism waits for a fixed time before sending the broadcast packet. By delaying the broadcast, the best metric from all the messages received is chosen and the number of setup messages is reduced. Implicit acks are employed to acknowledge a packet that has just been relayed by snooping for the next broadcast packet - that with lower cost - in the path. They tackle the problem of a node changing area by broadcasting a request message to the new neighbours such that the new costs for the gradient can be recomputed based on the received height information from each neighbour.

2.3.3.7 GRASP (2005)

Lim et al. present a gradient-ascending stateless mechanism for gradient-based routing protocols called GRASP [87]. GRASP operates in stationary networks where the gradient has already been established. GRASP makes use of data packets which descend the gradient towards the sink. These data packets are used by GRASP to populate a space efficient data structure called a Bloom filter [88], a probabilistic bit-vector structure which acts as a membership-based filter capable of indicating whether the packet has been stored or not. The Bloom filter mechanism is explained in Section 4.4.1, nevertheless it has to be noted that GRASP employs a counting Bloom filter [89]. Counting Bloom filters are capable of inserting and deleting elements as they are multi-dimensional Bloom filters, thus they can keep count on how many items have been inserted within a certain probability. This adds more complexity and reliability to the probabilistic membership mechanism from a simple Bloom filter. GRASP stores packet footprints in the Bloom filter and exploits this information to ascend the gradient from the sink to the source node. If a node receiving a packet contains in its Bloom filter the footprint of a previous packet related to this one, then the node broadcasts the packet. This mechanism takes into account sequence, hop count, destination and source fields of the packet and avoid duplicates. The authors state that the number of packets from the same source will traverse a similar set of nodes in the network when descending the gradient. However, this might not be true depending on the number of source nodes sending packets and the network density. A situation where many of the bits are set to 1 in the Bloom filter will start to produce false positives. Nevertheless, they employ a Bloom filter per sink packet in each node and employ a metric to quantify how long ago a footprint was created or assessed. In ascending the gradient, GRASP employs reliable mechanisms such as implicit and explicit acknowledgement packets employing storing and rebroadcast mechanisms to increase the delivery success of the packet. GRASP is simulated in ns-2 [90] with the CSMA/CA IEEE 802.11 MAC layer.

2.3.3.8 HYPER (2006)

Hyper [91] was designed as a collection routing protocol offering support for mobile sinks. The protocol capitalises on the idea of collecting data at any given time from any given point in the WSN. In this situation, a mote is deployed which acts as a sink. The sink advertises its arrival to the neighbourhood and evaluates the link quality to each of its sensors with a mechanism called "fast connect". This is done by using the Expected Transmission Count (ETX) metric over a series of broadcast beacons and acknowledgements. The ETX metric estimates the link quality based on the number of successfully delivered unicast packets between two given nodes. Once the neighbourhood is properly assessed, Hyper builds the tree-based routes and creates a gradient towards the sink. The formation of the gradient delays the broadcast based on the cost value at each node which only broadcasts the best cost received during the waiting time. A periodic beacon is sent to maintain the tree structure connected. If a node, or set of nodes, get isolated from an area or do not have a sink to route to, they store their data in the flash memory until a new data sink is advertised. Every time a sink enters a new neighbourhood it needs to advertise its presence and setup the gradient. Hyper, as many other protocols which aim to increase reliability in the data delivery, bases its reliability mechanisms foundations in the good assessment of the link quality of its neighbours, in the appropriate management of the neighbourhood table and in the use of reliability-based cost metrics [92].

2.3.3.9 RCDR (2006)

The Reliable Cost-based Data-centric Routing Protocol for Wireless Sensor Networks (RCDR) [93] was designed to route data from an event zone towards the sink in dynamic environments where the sink node can change position. The sink needs to initially spread its global gradient. Then data packets from nodes capturing an event descend the gradient by multipath routing. RCDR exploits the event's footprint-based gradient created by the diffusion laws of a physical phenomenon. This is used as the local gradient for the event which overrides the global gradient for the sink, such that nodes report initially to the local minima node for the event. This node is then in charge of aggregating data and sending the packet towards the sink by progressing through the global gradient. The global gradient is formed in a similar fashion to GRAB [76, 77]. RCDR employs Data Query (DQ) messages to setup the gradient. A variable backoff time is applied in the setup phase at each node which depends on the cost. This avoids the rebroadcast storm with only one broadcast per node occurring which transports the minimum cost from those packets received. It also implements GRAB's mechanism to control the width of the multipath band between source and sink; they employ the concept of credit cost as premium cost. Only the first packet to be received in the forwarding phase is forwarded. Mobility is detected by periodically checking neighbours information from the MAC layer. If a node moves, and it is not the sink, it sends a query message to its new neighbourhood to update its cost table to the best relay node for the sink. When the sink detects a change in its neighbourhood, the setup phase is performed with a limit in the number of hops for the DQ to live. This avoids updating the whole network, only refreshing a closer area. Every time the sink updates its gradient in this manner, the initial cost of the DQ packet sent by the sink is decremented by 1, such that it keeps on updating the funnel, i.e. gradient, to negative lower values of cost - on the initial setup the sink has a cost of 0 - such that packets keep progress to the sink. The authors employ a CSMA/CA MAC protocol, the Sensor-MAC [40], although no reference to the use of a sleep duty cycle is made. RCDR has been evaluated against GRAB comprising simulations with speeds from 0 to 3 m/s in a 60 sensor network of 800x800 meters with a transmission range of 150 meters and a 2 packets/second stream from 1 node to the sink. Nodes move in this area according to the random way-point model (RWP) with random speed and waiting time. With dynamic scenarios where nodes move at speeds greater than 2 m/s, GRAB does not perform reliably and the expensive flooding mechanism doubles the energy consumed with respect to RCDR. A 75% success delivery ratio is achieved with RCDR when 100% of the nodes move (sink is static) at a speed of 3 m/s, while 90% is achieved when 20% of the nodes are moving. When only the sink moves, RCDR achieves 95% success delivery ratio at 3 m/s with GRAB achieving 28%.

2.3.3.10 SGF (2009)

In State-free Gradient-based Forwarding (SFG) [79], the gradient is established at the beginning of the network and it is updated via received data packets. By employing a backoff-based gradient formation mechanism, an improved version of the algorithm designed by Ye et al. [81], the gradient cost value is defined as the minimum total energy required to send a packet from the node itself to the sink; this takes into account energy consumed at every node both in the transmission and reception of the packet. The backoff-based scheme in "A Scalable Solution to Minimum Cost Forwarding in Large Sensor Networks" (see Section 2.3.3.3) introduces a delay in the formation but avoids multiple retransmissions of the advertisement packet by the same node in an effort to optimise the gradient formation. Improvements on the design of the backoff-based gradient formation algorithm have been carried out to avoid the situation where the gradient is not properly formed due to a miscalculation in the delay to broadcast at each hop. The delay introduced at every node before broadcasting the next ADV packet is a function of the energy cost received in the packet, i.e. a delay in the order of milliseconds is applied per watt of energy. However, the authors of SGF found that when higher delays occur when the packet is being transmitted, propagated and processed, the broadcast delay might not be large enough to wait for other packets to arrive before the timer expires. Therefore a minimum delay to broadcast needs to be larger than the sum of the transmission, propagation and processing delays for a packet plus the maximum delay which can be incurred by the MAC layer, taking account of the exponential backoff and the number of retransmissions. The improvement increases the delay in the formation of the gradient but guarantees an energy-efficient formation of the gradient with an optimal cost. SGF does not maintain a neighbours table but rather maintains the gradient as a cost-field value which indicates the direction toward the sink. When the gradient breaks, an opportunistic distributed contention process within the neighbourhood is launched to fix the gradient path towards the sink. A neighbour node wins the contention process depending on its gradient, channel condition and remaining energy, and gets elected as the next forwarder.

2.3.3.11 GRACE (2009)

Khan et al. in their paper "GRAdient cost establishment (GRACE) for an energy-aware routing in wireless sensor networks" [94] present a protocol which enhances network lifetime and reliable data delivery when compared to GRAB (see Section 2.3.3.4). In the gradient setup phase, the ADV packets carry the lower cumulated cost value of each node when broadcast. This is achieved by following the same mechanisms as GRAB for the setup phase which also includes a back-off time which is calculated as a function of the cost (same as in GRAB). The cost is calculated based on the cost of the link to the sender plus the energy of the node (receiver). The cost of the link is calculated as the transmission energy consumption divided by the reception energy consumption of the two nodes of the link. Once the setup phase is completed, the so called steady-state phase is launched where data packets flow in a unicast manner through the path with the smallest cost. GRACE offers different modes of operation to update the status information at the nodes, i.e. the costs. This way nodes with lower energy increase their cost thereby informing neighbours and reducing the likelihood of participation in the path. The first mode is based on updating the cost of the nodes via a new gradient setup when a node dies. The second mode employs unicast acknowledgements from data packets on the way to the sink to piggyback the current cost of each node. A third mode employs broadcast acknowledgements to inform all the neighbourhood of its current cost.

Finally, two correction modes are presented which act as reliable mechanisms employing end-to-end acknowledgements from the sink to the source and from an intermediate node back to the source; these mechanisms inform whether a packet has been lost or dropped. The authors also propose a modification to GRAB to enhance its energy-efficiency while maintaining the same level of reliability. This is achieved by reducing the number of broadcast packets in GRAB via the use of GRACE's acknowledgements and correction modes. Overall, GRACE offers a solution to extend the network lifetime which can be controlled by dynamically choosing the most suitable mode of operation to update the cost of nodes according to the density of the network or vicinity of the sink.

2.3.3.12 CTP (2009)

The Collection Tree Protocol (CTP) [95] is the de-facto standard in data collection protocols for Wireless Sensor Networks in the leading platforms: TinyOS 2.x [96] and Java SunSPOTS [97]. CTP is a tree-based protocol where a designated root, or set of roots, advertise themselves in a gradient setup manner creating routing trees. Root nodes establish their gradient using a cost metric called Expected Transmission Count (ETX). The ETX metric for each node estimates the link quality based on the number of successfully delivered unicast packets between two given nodes; that includes the successful reception of the acknowledgement packets. The ETX of a node is the ETX value received from its parent plus the ETX cost of the link from the node to its parent. The ETX at the root is 0 when the gradient is setup. This is an additive measure, much like the hop count, which helps to avoid loops when data packets descend the gradient towards the root by always progressing through a node with lower ETX. When a loop is detected, an inconsistency in CTP terms, the node broadcasts a beacon to inform the node which sent the data packet that it needs to adjust its routes. The data packet is then sent along the loop with the prospect that by the time the packet comes back to the node the routes have been adjusted and the loop cancelled. In the case where the network is partitioned, nodes could not progress towards the sink; in this case the ETX, which might be adjusted in an uncontrolled fashion forming loops, would reach a maximum ETX value. Legitimate duplicate packets, for instance those which circulate through the loop twice, need to be detected but not discarded. To detect this type of packet, a counter named "time has lived" (THL) is carried in the data packets. Packets are uniquely identified by their origin address, sequence number and an identifier for higher protocols. When the THL is also considered the instance of a packet can also be uniquely identified.

A parent node is in charge of detecting inconsistencies for its children, i.e. children have lower ETX than the parent. The parent node reacts to inconsistencies by sending a beacon frame to all its neighbours to inform them about the inconsistency. This node also needs to schedule an advertising process in the near future to fix those routes.

Two coupled mechanisms are employed to discover bidirectional links to enable computation of the ETX; every 5 transmissions the ETX value is computed based on the previous ETX and the successful packet delivery ratio. The first mechanism, called LEEP, employs beacon messages containing in-bound link estimations according to the transmissions for the nodes in the neighbourhood [98]. This way nodes receiving the beacon can calculate their out-bound link estimation. The periodicity of broadcasting the beacon is regulated based on the Trickle dissemination protocol [99]. Trickle self-regulates the beacon update period based on the changes in the neighbourhood according to some parameters in the packets. In the case of CTP, an inconsistency or a drastic decrease in its routing cost will reset the period of beacon transmission to the minimum value. A packet can also trigger the beacon, for instance if its routing table is empty. If the beacon transmission period is not reset, it doubles until it reaches a maximum value. Minimum values can be in the order of 60 ms while maximum values in the order of 1+ hours. In addition, the other mechanism employs the unicast data packets and its acknowledgement packets to update the ETX values. Information from both mechanisms is combined into an exponentially weighted moving average.

CTP provides best-effort delivery mechanisms with a high number of retrials and local repairs and it has been designed for relative low traffic rates. CTP counts on the link layer to provide an efficient local repair with synchronous acknowledgements for unicast packets; each packet carries single-hop source and destination addresses.

CTP has been designed with a set of core mechanisms for collection routing purposes while leaving some of the aspects open for different implementations. Gnawali et al. developed CTP Noe [100], an implementation of CTP which includes additional functionality to improve the protocol's performance. CTP Noe uses retransmit timers of up to 32 trials capitalising on the idea that dropping a packet from the queue would not fix the path for the next in the queue. This can decrease the performance for real-time applications where fresh data needs to be received. However, this is a best-effort implementation and the value is open to be modified. CTP employs hysteresis, i.e. the reaction of the system depends on past reactions, in the selection of the route such that a new route is selected if there is a considerable gain in the ETX value. Its beacon rate in Trickle has been set to a minimum value of 64 ms and a maximum value of 1 hour. This way changes in the topology (mainly link dynamics) can be detected rapidly and at the same time avoid redundant communication when the network is stable. CTP Noe delays the transmission of a data packet based on a randomized value within an interval which aims at minimizing collision with previous packets in the path. However this mechanism works well when there is only data flowing towards a path. It also implements a hybrid send queue where there is a pool of packets for as many higher-level clients of the routing protocol and a fix pool of packets for forwarding packets. The queue along with a small cache is checked for duplicates packets. In addition, CTP Noe employs a 4-bit link estimator metric as a composite value of the quality of the link from the physical, mac, and routing layer [101]. CTP Noe has been evaluated in 12 static WSN testbed with different link dynamics, mote platforms, density and number of motes. Four mac layer protocols have been employed with different low power schemes. One root node has been used with different inter-packet intervals, from every 16 seconds to 5 minutes, operating in channels with different levels of interference. One experiment aimed at reproducing sink mobility by changing the root every so often. Experiments measured the performance of CTP Noe in terms of robustness, agility to failure and energy consumption in scenarios with topology inconsistencies and with different transmit timers and cache sizes. CTP Noe offers 90-99.9% packet delivery in highly dynamic link topologies with 73% fewer control packets than existing approaches. The authors claim it can achieve duty cycles of less than 3% while supporting aggregate loads of 25 packets per minute.

2.3.3.13 BCP (2010)

The Backpressure Collection Protocol (BCP) [102] implements the backpressure routing concept for routing packets in multi-hop networks. This is achieved by employing queue congestion status in the forwarding of a packet. BCP needs to initially create a gradient, based on the ETX metric, similar to CTP [95], where routing trees are created with the root node as the sink collector. Based on the ETX value and the queue backlog of neighbour nodes, a weight is calculated for every link between the node and its neighbours. This weight changes as packets are queued or de-queued and it is the basic factor in selecting the next hop for the next packet in the queue to be forwarded. Thus, the protocol operates on the basis of pushing packets according to the gradient and the status of the queue. In the initial gradient setup phase, some packets get trapped in the queues and can not progress. This is due to the difference in queue size between a node and its neighbours. This difference does not create a positive weight required for the packet to be forwarded. However, when packets start to be sent in the direction of the sink, packets begin to flow from high to low congested queues in the direction of the gradient. In addition, FIFO and LIFO queues are employed as two mechanisms for selecting the packet to be forwarded.

The authors compared the protocol against CTP [95] in static and sink mobile scenarios, where another node is simply designated as the sink, rather than having physical mobility. With 0.25 packets per second, CTP has a 99.0% delivery ratio while BCP produces 96.9%. However, in mobile sink scenarios, the delivery ratio for BCP was 99.6%, considerably higher than CTP's 59%. It has to be noted that when using LIFO instead of FIFO, the system average delivery packet latency can be reduced by more than 98% at low data rates, while 75% at higher rates.

2.3.3.14 WARP (2010)

The Whirlpool Routing Protocol (WARP) [103] is a data collection protocol which works on the same basis as CTP [95], indeed WARP is an extension of CTP which includes an enhancement for routing to mobile sinks. A sink node in WARP detects its mobility by using periodic beacon messages. The beacon sending rate is variable as the node only sends beacons if a data packet has not been received within a certain period of time. Thus, data traffic is employed to validate the path stability to the sink node. Beacons are employed to inform nodes of the presence of the sink in the neighbourhood. However, if the sink moves and a data packet arrives to its old neighbourhood, a mechanism is launched to locate the sink and fix the routing tree from the new to the old area of the sink. This mechanism, the so called whirlpool routing, is initiated by the node which is unable to relay the packet to the sink. The whirlpool routing mechanism exploits the irregular concentric circles of increasing cost values which are formed around the sink when the gradient is setup. A packet is sent to a randomly selected node in each of the concentric circle areas up to a maximum distance which is probabilistically estimated. When nodes receive the packet, they forward it to nodes with a closer cost value, therefore packets start to be routed around the corresponding concentric circle. This is a speculative mechanism which sends packets to the last known location of the sink in order to locate its new position. When the sink receives or snoops the whirlpool packet, it sends a beacon to the neighbours in the area of the update. This packet cancels the whirlpool routing and progressively updates the cost tables of the nodes involved. WARP is compared against CTP, HYPER [91] and DYMO-TYMO [104] with metrics such as delivery ratio and average path length in a real topology of 60 motes with slow sink mobility and in the simulator TOSSIM [105–107] where mobility is induced by changing the id of the sink per time. Node speed ranges from 0.7 to 2.9 meters per second and different overall data rates from 4.3 to 21.3 packets per second are employed. Across all data rates and speed scenarios, WARP stays above 83% in terms of reliability and outperforms CTP and HYPER (60%). However, WARP performance in terms of reliability goes down to 48.8% when testing in high speed scenarios.

2.3.3.15 TABS (2010)

TABS (Try Ancestors Before Spreading) [108] is a gradient-based routing protocol which aims at descending the gradient while increasing reliability in the presence of lossy links. The protocol assumes that a gradient setup phase occurs which creates a gradient based on the hop count metric. This phase can be initiated reactively by the sink in a distance-vector manner or as an exchange of local information following the link-state approach. Each node maintains a cost to the sink. In addition duplicate packets can be detected. When descending the gradient towards the sink, a sender broadcast a packet which contains a value known as the "minimum progress limit". Receiving nodes compare their cost with that of the received packet and keep on broadcasting if the difference is higher or equal than the minimum progress limit. This way TABS controls how many nodes are involved in the forwarding process in the same way GRAB [76, 77] controls the width of the forwarding band. TABS uses implicit acknowledgements from overheard broadcast packets to cancel redundant broadcasts and also employs acknowledgements on received broadcast packets to increase the reliability. When an acknowledgement packet is not received within a predefined time, a retransmission is launched. If this fails, the "minimum progress limit" value is reduced in order to increase the number of opportunistic routes which can be used. The "minimum progress limit" is an inverse function of the retransmissions counter for the packet. TABS has been evaluated on a testbed having 56 telosB nodes over 1125 square meters achieving over 98% delivery ratio while link quality is highly dynamic.

2.3.3.16 DGR (2011)

Guo et al. present DGR in the paper "Dynamic Gradient-based Routing protocol for unbalanced and persistent data transmission in wireless sensor and actor networks" [109]. The protocol aims at balancing the energy consumption by detecting nodes reaching certain energy levels and consequently refreshing the gradient with new cost values. When gradients are updated, a parameter in the packet controls the number of nodes involved in the routing process; this is known as the "expansion strategy".

DGR has been developed based on concepts from the SGF [79] and GRAB [76,77] protocols. The protocol also assumes that nodes can change the communication range by varying the transmission power at any given time. All sensor nodes are initially in idle mode and, as soon as a sensor receives a transmission request, they turns to busy mode. The DGR protocol mechanism is comprised of three phases: the dynamic gradient setup phase, the routing path establishment phase, and the data transmission phase.

In the setup phase of the dynamic gradient, the hop metric is employed. Nodes accept only the first received ADV message and ignore all the others. In this phase the gradient cost value is calculated based on the hop count, the remaining energy of the node, and a coefficient which trades network balance for routing efficiency. The coefficient helps regulate how many nodes are involved in the routing process for this gradient, i.e. the expansion strategy. Once this phase is completed, source nodes launch the route establishment phase. A source node broadcasts a packet, at low transmission power, requesting neighbour nodes to reply. Neighbours delay their reply as a function of their gradient cost value. The first node replying will be selected as the next hop which also enables bidirectionality for reverse path communication. The fast packet to reply is selected as it forms an efficient gradient in terms of hops and energy. The process is repeated until the destination sink is reached. When the path is established, data can be sent in the data transmission phase. In this phase nodes send packets using full transmission power, rather than the lower transmission power employed in the route establishment phase. This way a node can reach other nodes in the path at a distance higher than one hop away. This will avoid having to route over a higher number of nodes. The distance which can be reached in terms of hops according to the full power transmission is calculated based on information of the power levels of all the nodes in the path. However, the node receiving the data packet needs to send an acknowledgement packet back to the sender using the full path at low power transmission. This mechanism increases the reliability of the communication. In addition, the routing path establishment phase is launched again when the energy level of a node falls below a threshold. This energy threshold is computed by the sink and disseminated in the ADV packet when the gradient is created. When this happens, the node sends a message to the source node so it stops transmitting data packets and initiates the route establishment phase. Nodes not participating anymore will go into idle mode. Now costs will be recalculated with the current energy values. Different costs will now change the path to be established, balancing the traffic load to those with higher costs and therefore higher energy. However, by varying the coefficient involved in the calculation of the cost, a tradeoff can be achieved between optimising the route in terms or hops and balancing the network in terms of energy.

2.3.3.17 RPL (2010-2011)

Recently, the IETF Routing Over Low power and Lossy networks (ROLL) Working Group has been formed [11] to standardize a generic routing protocol which satisfies most of the requirements and constraints from urban [110], industrial [111], home automation [112] and building automation [113] scenarios where low power and lossy networks are to be deployed. An IPv6-compliant routing protocol known as RPL has been designed by taking into consideration the applicability, scalability, constraints, usability and existing technology integration of this type of networks. RPL (Routing Protocol for Low power and lossy networks) [12] has been engineered to route data in networks of dozens to thousands of low power devices where interconnections are characterised by high loss rates, low data rates, and instability. The protocol has been proposed, with some modifications, as the routing layer for the advanced metering infrastructure in Smart Grid [114]. The design of the protocol describes the basis of routing such as packet processing and forwarding decisions while leaving the network traffic optimisation mechanisms open for implementation by the application developer.

The IETF ROLL group work indicated the necessity for the routing protocol to support point-to-point (P2P), point-to-multipoint (P2MP), and multipoint-to-point (MP2P) communication, to account for all the possible traffic flow application scenarios. The RPL routing protocol employs gradient-based routing concepts as the basic mechanism to accomplish most of the requirements outlined for the generic design of a routing protocol. Thus, RPL capitalises on the formation of Directed Acyclic Graphs (DAG) ending up in a DAG root, which is a node with no outgoing edges. A DAG is defined by 1) a set of sink nodes, 2) the set of metrics on each link, 3) the link cost formation according to the metrics and 4) multi-hop path cost formation from a combination of link's costs. The protocol uses the concept of "Up" and "Down" to route through the acyclic graph and a "Rank" value to assess a node's position relative to others with respect to the root node. A DAG can have multiple root nodes (gradients), useful for multicasting, while graphs with only one root node are known as DODAG. Forwarding mechanisms and Route Repair mechanisms are suggested and paths are established by sending the so called Destination Advertisement Object (DAO) messages. An on-demand mechanism to establish cost-efficient routes from a router or host to another router is available to guarantee P2P communication [115]. RPL is IPv6-compliant employing periodic IPv6 Router Advertisements to maintain the gradients in the network.

2.3.4 Other Gradient-based Routing Mechanisms

A number of protocols have been presented in detail in the previous section, which describe the basis of gradient-based routing and a variety of mechanisms employed in the gradient setup and data forwarding phases. This section extends the state of the art by briefly presenting other algorithms for gradient formation and forwarding decision taken.

Shah et al. calculate the gradient setup cost value as a combination of the link transmission energy and the residual energy of the node [65]. The algorithm does not aim at finding the optimal route but rather employing multiple sub-optimal paths to reduce traffic load and avoid network partition. To achieve this, a probabilistic forwarding mechanism calculates an average cost of the available neighbours to route data (excluding high cost values), and randomly selects the next neighbour from those with a cost equal or less than the average cost of the neighbourhood.

Chen et al. present an algorithm called Self-Selective Routing (SSR) [116] for wireless networks which employs the gradient cost variable as a delay in the data packet forwarding. In SSR when neighbour nodes receive a broadcast data packet, a back-off timer is set to a value proportional to the cost. When the first receiving node broadcasts the packet, this serves as an implicit acknowledgement to cancel the broadcasting of those nodes waiting. To cover those nodes out of the range of the node broadcasting, the previous sender (which has overheard the broadcast) sends an explicit acknowledgement packet to cancel the sending of those nodes in its range. Even though the mechanism is robust to failures and reduces the number of forwarded messages, the end-to-end delay is increased.

Cheng et al. propose a forwarding method which takes decisions based on 1-hop neighbourhood information to balance the traffic load according to a priority value which is calculated based on the remaining energy [117]. The approach employs a multi-sink network in which nodes route information to the nearest sink to save on the overall energy of the network; the goal is to push data from the network to the sinks in an energy efficient manner.

On the selection of the next hop to forward the packet, the Erdene et al. propose a set of selection schemes (random, deterministic and random probabilistic) to improve the resilience of the network against malicious attacks [118]. The next hop selection strategies were implemented and tested on the Gradient-Based Routing protocols (GBR) [45].

Han et al. employ the hop count as a metric to setup the gradient [119]. They also tackle the problem of a node changing its neighbourhood via sending periodic hello messages to discover the node with the lowest hop count to the sink; this node become its new parent.

Verbist et al. present a mechanism for good formation of the gradient which utilises a back-off time before broadcasting the gradient advertisement packet [120]. The FuRF (Furthest Responds First) algorithm employs the received signal strength indicator (RSSI) and the hop count to delay the single broadcast packet sent by each node.

A gradient setup mechanism for highly dynamic environments is described by Hhan et al. [121]. The mechanism enhances the ideas in the paper "A scalable solution to minimum cost forwarding in large sensor networks" [81] which was only evaluated to work well in static environments. This mechanism requires that all nodes in the network are synchronized and that the number of neighbours of a node is predefined. The algorithm is based on a delay computed at the sink which is then carried in the advertisement packets. The advertisement packet tells nodes the amount of delay before broadcasting. Only one broadcast is made with the best accumulated cost. In order to account for mobility, the sink performs statistical analysis on the received data packets in search of performance variations. If mobility is detected, the sink triggers a new gradient setup process with a new delay value which is calculated from statistical analysis of the data packets received amongst other factors. The idea aims at a proper gradient formation where, for instance, nodes at 1 hop away from the sink broadcast the packet before the first node at a distance 2 hops away does it.

2.3.4.1 Data-Centric Query and Searching by Gradient Navigation

Another class of gradient-based routing protocols exploit the potential field, i.e. the gradient, formed by the physical phenomenon diffusion laws when the phenomenon propagates away from the point of origin of the event, e.g. the RCDR protocol (see Section 2.3.3.9). This class of protocols are driven by queries of data which navigate the gradient, usually by employing greedy algorithms, selecting the next hop node according to the correlation of the phenomenon/s sensed at the nodes and the information encapsulated in the query.

In this line of research, Jabbed Faruque et al. proposed in their paper "Routing on fingerprint Gradients in Sensor Networks" (RUGGED) [122] a mechanism for routing of packets towards the central point of an event. This is achieved by following the event's fingerprints left in the sensor network according to the diffusion laws of any physical phenomenon. Their argument is based on the gradient created by a phenomenon which affects nearby sensors according to their distance from the event. The rationale is that queries initially employ some flooding mechanism to traverse the network until those nodes containing some information about the event are located. At this point, packets progress through the gradient in a multi-path greedy manner. Due to the possible distortion of the physical events over distance, local minima or maxima points can be given. The authors employ a technique based on simulated annealing, where a change in the status is based on probabilistic decisions, to escape from this points and progress towards the global maxima or minima, i.e. where the event occurs. This is not a gradient-based protocol per se. The algorithm is based on the concept of data-centric routing, where packets are forwarded according to the properties of the data, instead of using node identifiers. However, the algorithm provides evidence of the suitability and rationale behind the use of gradients for routing.

Similar data-centric approaches which employ the concept of navigation through the gradient formed by the phenomenon, or data from the event, have been employed [123–126]. These mechanism seek to query single events, or a composition of them, in an efficient manner by descending and climbing the potential field generated by data from the phenomenon. In-network processing and decision making at every forwarding node are made. These employ a combination of equations based on the way data is diffused and the analysis and comparison of both sensor data and query (meta)data.

2.3.5 Significance for this Thesis

This section has presented a chronological and comprehensive review of gradient-based routing protocols for wireless sensor networks. Common features of all the gradient-based routing protocols analysed, including the Ubiquitous Mobile Gradient (UMG) routing protocol, have been identified and are presented in Table 2.1. The table helps to characterise the behaviour and functionality provided by UMG when compared to existing protocols. UMG's design can be compared to that of DGR [109] or GRACE [94] as they have some commonality of feature set, including a reactive behaviour, unicast-based gradient descent and gradient ascent functionality. However, UMG does not incorporate energy in the routing metric and tolerates sink mobility.

While having some similarities to existing gradient protocols, UMG design integrates a set of specific mechanisms which enhance the reliability and versatility of the routing process. UMG incorporates a robust gradient formation mechanism based on a backoff delay as a function of the hop count. The protocol also employs a mechanism which avoids cycles and allows for the coexistence of cost table information from different gradient formation processes from the same sink node. It employs reliable end-to-end mechanisms utilizing a unicast-based gradient descent phase to transport data, while ascending the gradient with broadcast packets for acknowledgement purposes. UMG also employs a novel mechanism to detect relative mobility for triggering scoped sink gradient updates. In addition, UMG leverages the routing process to provide service advertisement to higher layers.

In summary, UMG has been designed as a reliable and versatile routing protocol, providing point-to-point, multipoint-to-point and point-to-multipoint communication for those scenarios where a node's application requires end-to-end, on-demand connectivity. UMG has been designed to satisfy the communications requirements of the content distribution protocol presented in this thesis, i.e. the TinyTorrents protocol.

Table 2.1: Comparison of Gradient-based Routing Protocols.

	Reac	Proac	Assy	Cost	Gradient Creation	Height Update	Descent	Ascent	Mobi
GRAd(2000) [74]	✓			Hops	Flood,1stPck	Setup	Broadcast	✓	
GBR(2001) [45]	✓	✓		Hops	Flood,1stPck	Neigh	Unicast		
A Scalable...(2001) [81]	✓			EnergyComms	Flood,BestPck,Delay	Setup	Unicast		
GRAB(2001-2005) [76,77]	✓		✓	EnergyComms	Flood,BestPck,Delay	Setup	Broadcast		
GLIDER(2005) [85]		✓		Hops	N/A	N/A	Unicast		
A New...(2005) [86]	✓		✓	Hops+Energy	Flood,BestPck,Delay	Setup	Broadcast		✓
GRASP(2005) [87]	✓			Hops	N/A	N/A	N/A	✓	
HYPER(2006) [91]		✓	✓	ETX+LQI	Flood,1stPck,Delay	Beacon	Unicast		✓
RCDR(2006) [93]	✓		✓	EnergyComms	Flood,BestPck,Delay	Setup	Broadcast		✓
SGF(2009) [79]	✓	✓		EnergyComms	Flood,BestPck,Delay	Setup,Data	Unicast		
GRACE(2009) [94]	✓		✓	EnergyComms	Flood,BestPck,Delay	Setup,Ack	Unicast	✓	
CTP(2009) [95]		✓	✓	ETX	Flood,1stPck	Beacon,Data,Ack	Unicast		
BCP(2010) [102]		✓	✓	ETX+QB	Flood,1stPck	Beacon,Data	Unicast		
WARP(2010) [103]		✓	✓	ETX	Flood,1stPck	Beacon,Data,Ack	Unicast		✓
TABS(2010) [108]	✓	✓	✓	Hops	N/A	Setup	Broadcast		
DGR(2011) [109]	✓		✓	Hops+Energy	Flood,1stPck	Setup	Unicast	✓	
RPL(2011) [12]		✓	✓	User-Defined	Flood,1stPck	Beacon	Unicast	✓	
UMG(2012)	✓		✓	Hops	Flood,1stPck,Delay	Setup,Ack	Unicast	✓	✓

Reac/Proac: Reactive/Proactive Route Discovery ||| **Assy**: Assymetric Link Tolerance ||| **Cost**: Routing Metric for Gradient Creation and Navigation (EnergyComm: Energy Radio Transmission; Energy: Device Energy; QB: Queue Backlog) ||| **Gradient Creation**: Gradient Creation Mechanism (Flood: Flooding; 1stPck: 1st Packet Received is Broadcast; BestPck: Best Cost Packet is Broadcast; Delay: Back-off Delay before Pck is Broadcast) ||| **Height Update**: Cost Table Updated (Setup: In the Gradient Setup Process; Beacon: With Periodic Beacons; Neigh: With Explicit Neighbour Pck; Data: With Data Packets; Ack: With Ack Packets) ||| **Descent**: Gradient Descent (Broadcast: Data Pck is Broadcast according to Threshold with Cost Decrement; Unicast: Each Router Selects Next Forwarder following the Routing Table or Local Repair) ||| **Ascent**: Gradient Ascent - Ack Pck from Sink to Source ||| **Mob**: Sink Mobility Tolerance

2.4 Peer-to-Peer Data Distribution in Wireless Networks

Global data dissemination networks and infrastructures can be realised using Peer-to-Peer (P2P) technologies over TCP/IP. P2P architectures are commonly characterised as moving the resource management to the edge of the network, i.e.: to the "peers". This is achieved by allowing direct communication among peers, which encourages them to cooperate amongst each other in managing the resource. The resource being managed is commonly bandwidth, data or computing power. Peers in a P2P system operate autonomously, which means that such systems exhibit greater reliability and availability, as they are more resistant to individual node failures. Peer autonomy also means that peers are often equipped with discovery protocols, which allow them to find functioning peers in the network when neighbouring peers have failed. This ability of "self-organisation" greatly adds to the reliability of P2P systems. It also means that the central node is relieved of much of the management burden. Ideally, a P2P network has no central nodes, with decentralised functionality being provided through structured Distributed Hash Tables (DHT) on all participating nodes or unstructured discovery mechanisms. For these reasons, P2P systems generally scale better than corresponding client-server systems. This is most clearly seen in file-sharing applications in which the cost of publishing and transferring data is spread across all the nodes in the P2P network.

2.4.1 Taxonomy of Peer-to-Peer Content Distribution Strategies

The concept of peer-to-peer networking enables distribution of the burden of centralised servers by turning networks into a set of self-organised devices capable of communicating with each other and sharing resources. In particular, content distribution is an area of P2P technology which deals with data sharing in a distributed way. The network acts as a distributed database where redundant data is published, searched and transferred by employing P2P communication amongst devices [127]. This approach offers the following advantages: i) network traffic load distribution, which contributes to avoid congestion and improve throughput by better using the resources in the network, ii) fault-tolerance, as data is replicated and traffic control is decentralised, iii) implicit security, which is achieved as multiple copies of data at disparate points in the network can be verified against each other for integrity and authenticity, iv) higher reliability, as there is availability of multiple copies of data from multiple sources, v) robustness to changes, due to a higher availability of data at multiple and distant points in the network, vi) scalability, as new devices entering the network do not incur a major system performance decrease, and vii) resistance to censorship, as there is no central unit of traffic control and anonymity can be achieved.

Ideally, P2P networks are fully distributed, however different systems require some degree of centralisation for coordination purposes. In this regard, a network can be classified as having a:

- Purely Decentralised Architecture

  All nodes have the same duties and functionality; there is no central entity.

- Partially Centralised Architecture

  Some nodes might perform special tasks within a local neighbourhood. These nodes are called supernodes, they are dynamically appointed and can be replaced making the system fault tolerant.

- Hybrid Decentralised Architecture

  There is a special peer which contains metadata information about what other peers contain and can also indicate how to access them. This peer acts as a central coordinator. As a result of this, there is a single point of failure which makes the system vulnerable and less scalable. However, the transmission of data obeys the peer-to-peer concept.

Another classification arises from the structure of the virtual overlay network employed by decentralised distributed systems for content discovery:

- Unstructured Overlay Networks

  Data or metadata files are distributed without following an overlay structure. Nondeterministic, inefficient searching mechanisms like flooding have been employed for simplicity. This type of architecture is suitable for networks with transient nodes where a structured overlay requires frequent update. On the other hand, scalability problems may arise if the searching mechanism proves to be inefficient in terms of latency and communication. However, the evolution of searching mechanisms has overcome some of the inefficiencies in the searching process. Techniques employed include: i) gossiping, ii) routing indices [128], iii) history and local data caching, for decision making in the forwarding of the packet [129], iv) multiple parallel random walks [130], v) intelligent forwarding decision based on a ranking of peer performance profile [131] and vi) heterogeneity node's exploitation for the formation of dynamic clusters while employing random walks for cluster nodes searching [132]. An example of this networks is Gnutella [133], which needs to bootstrap a list of nodes and then discover other nodes by pinging nodes from this list. Search is achieved with communication expensive nondeterministic methods, mainly variants of flooding. Singla et al. proposed a solution for efficient searching in Gnutella via the use of super-nodes, i.e. nodes hierarchically superior than other nodes in the network [134]. Super-nodes act as proxy nodes caching information for a set of client-nodes in such a way that a client-node has a supernode associated for querying. This approach ameliorates the traffic burden and makes the network scale by reducing the number of messages involved in the communication.

- Structured Overlay Networks

  In this type of infrastructure a virtual overlay network is created as a keyspace-based structure where virtual nodes store either a list of nodes containing data or the data itself. The structure provides an efficient mechanism to store and retrieve data from distributed nodes, called Distributed Hash Table (DHT). The idea is to employ consistent hashing to map data and node identifiers to keys in the keyspace structure. This way, data represented by closer keys will be managed by nodes with closer keys in the keyspace structure. Therefore the addition or removal of data, or node identifiers, will only incur an additional update to a small subset of the keyspace nodes; a dynamic rearrangement of the keyspace will be produced in proximate keys to the affected key of the structure. DHTs are mechanisms for storage and searching which scale well and are fault tolerant. However, in highly dynamic networks, where nodes may enter and leave the keyspace rapidly, the cost of updating virtual neighbours may be highly expensive in terms of communication. In addition, DHT-based peer-to-peer systems suffer from the Sybil attack [135], where a set of attack nodes might control part of the DHT structure. A solution to this is the application of trust-based systems in peer-to-peer networks. The main algorithms employing the DHT mechanism for data lookup are: Chord [136], CAN [137], Pastry [138], Tapestry [139] and Kademlia [140].

- Loosely Structured Overlay Networks

  This category can be classified as a structured architecture which, instead of maintaining a virtual overlay, locates data by combining data hash keys and routing level hints. An example of this special category is Freenet [141].

While centralised architectures offer efficient and reliable single-point lookup mechanisms for distributed data searching, they suffer from the hot spot problem and present a single point of failure. Pure decentralisation, partial centralisation and hybrid approaches are solutions which distribute both data and data location information over multiple nodes. Decentralised approaches scale better and can adapt to changes in the network from a local perspective. However, data, or service, discovery poses a challenging problem in decentralised P2P networks. In their paper "Ubiquitous and Pervasive Application Design" [142], Bakhouya and Gaber present a classification of service discovery systems according to their architectures and their operating mode. According to them, discovery systems can be classified as: 1) structured systems (indexation and DHT), 2) unstructured systems (flooding and random walks) and 3) self-organised systems (affinity networks). The first two discovery systems are covered in detail for wireless network in Sections 2.4.3 and 2.4.4. The latter approach is based on the formation of collaborative communities of peers according to affinity relationships; this can be seen as multiple overlay networks where a peer is a member of one or more virtual communities.

On the other hand, in the area of ad hoc wireless networks, data distribution and service discovery pose higher challenges than in traditional wired networks. This is mainly due to the unreliable wireless medium and the inefficiency of routing to distant nodes. Specifically in the domain of WSN, data dissemination algorithms are employed to distribute data over the sensor network for data collection and reprogramming purposes (see Section 2.4.5). Dissemination techniques have also been used in WSN to distribute data over the sensor network for distributed storage purposes. While some approaches employ local storage and need query mechanisms such as TinyDB [143], others distribute data replicas at remote locations in the network such as DISC [144] and TinyPEDS [145]. One of the major problems when transforming the sensor network into a distributed remote storage is the location of data. Some approaches employ cluster-based mechanisms for hierarchical searching (DISC and TinyPEDS) which may incur a high maintenance cost in terms of communications. In this regard, in the area of wireless sensor networks, a decentralised and flat-based network structure represents a more efficient and scalable solution, however, it poses greater challenges in the design of robust and reliable data discovery and dissemination mechanisms.

In the following section, the mechanisms of the BitTorrent protocol are presented as a reference point in the design of the data distribution protocol presented in this thesis. Proposed adaptations of BitTorrent to wireless networks are also identified. In later sections, solutions which employ structured overlay networks for data discovery and routing purposes, both in Mobile Ad Hoc Networks (MANETs) and Wireless Sensor Networks (WSN), are described. Some of the most relevant approaches for ad hoc networks, with a special emphasis in the area of wireless sensor networks, are presented in Section 2.4.3. Unstructured lookup mechanisms for ad hoc wireless networks are then described in Section 2.4.4 as convenient solutions for data searching in sensor networks offering a high degree of data replication. Finally, a review of the most influential dissemination protocols is given as mechanisms for data distribution in WSNs.

2.4.2 The BitTorrent Protocol for Wireless Networks

2.4.2.1 The BitTorrent Protocol

BitTorrent is a widely used file sharing protocol designed by Bram Cohen [146]. It is intended for efficient data dissemination on the Internet, creating a content distribution network based on the peer-to-peer model. Its key components are the peers, which cooperate in replicating content across the network, and the tracker, which provides a means for peers to discover each other. BitTorrent allows peers to cooperate directly in transferring data, and ensures fairness among peers using a tit-for-tat algorithm. Content is broken into pieces, each of which are treated as atomic units of data for transportation purposes. This allows different portions of the same file to be downloaded at the same time from different nodes in the network. The goal of BitTorrent is to replicate the content as quickly as possible among the peers, thereby reducing the burden on the original publisher of the content.

The main elements in the BitTorrent architecture are the torrent file, the tracker, the set of peers, and the seed peer. BitTorrent distinguishes peers into two groups: seeders and leechers. Peers which have a complete copy of the file being distributed are called seeders. Leechers are peers which do not yet have a complete copy of the content. A tracker is a centralised component which is responsible for keeping track of all peers currently involved in the distribution of a file. It is the means by which peers discover one another, using the tracker protocol which may be layered on top of HTTP. The tracker is not necessarily directly involved in distributing the content. BitTorrent does not include a file search mechanism as other P2P content distribution networks do. Instead, it relies on users acquiring a "torrent" file, which contains metadata about an item of content published through BitTorrent. The torrent file is acquired externally to the BitTorrent system. BitTorrent can be broken down into a number of key elements and behaviours:

- Publishing Data

  In order to publish content in BitTorrent, one needs to construct the ".torrent" file. This ".torrent" file is made available outside of the BitTorrent network, typically by hosting it on a standard web server. At least one peer, which has the full fileset, must be available. This peer is the seed for the torrent.

- Tracker Protocol

  The tracker protocol is used between peers and trackers, and signals the intent of a peer to join a torrent. The tracker keeps track of the peer membership in the torrent. Peers attempt to join a torrent using the information about a torrent contained in the ".torrent" file. In particular, the address of the tracker responsible for the torrent is provided. Once a peer has contacted a tracker, the tracker randomly chooses a set of peers from the torrent and communicates this back to the joining peer. Once a peer has joined a torrent, it will periodically inform the tracker of its status and progress. This ensures that the tracker's global view of the torrent is kept up to date.

- Peer Interaction
  Once a peer has joined the torrent and obtained a list of peers, it proceeds to establish TCP connections with its peer set, over which all peer to peer interaction will take place. All peer-to-peer connections are bidirectional. The peer interaction begins with a handshake process. The handshake messages contain a fixed header, followed by the SHA1 hash of a portion of the ".torrent" file, which uniquely identifies the content. If both peers do not agree on this hash, the connection is terminated. Once the handshake is complete, the peers begin exchanging data with each other. They do this by interchanging pieces of the file with each other. Whenever a peer completes a piece, it informs all the peers in its peer set. This is important, as it allows peers to keep track of the status of other peers and thus make better decisions regarding piece selection. Two important aspects of the peer protocol are now highlighted, piece selection and fairness.

- Piece Selection
  The piece selection strategy is the mechanism by which a peer decides in what order it should request pieces of the content. In order to select the piece to be downloaded next, peers in BitTorrent use a rarest first strategy. This means that peers choose to download the pieces which the fewest of their own peers already possess. Peers must inform each other when they obtain a new piece. The rarest first approach makes sure that peers have pieces which their peers are likely to want as well as ensuring that the pieces are rapidly duplicated across the network. This means that redundant copies of all pieces are made as quickly as possible, thus reducing the likelihood of content loss due to node failure. The rarest first approach also means that the burden on the initial seed is reduced much more quickly, thus reducing the flash crowd problem [147].

- Fairness: Choking and Unchoking
  In BitTorrent, peers essentially barter with each other for pieces of the content. Each peer-to-peer connection in BitTorrent is associated with two states at each end: choked or unchoked, and interested or not interested. The interested status indicates if the peer at the other end of

the connection has a piece that this peer still needs. Peers decide to whom they should upload using a tit-for-tat algorithm. They do this using "choking", which is a temporary refusal to upload to a peer. Once a connection is unchoked, then uploading can take place again. In this way peers reciprocate by uploading to peers which upload to them, thus encouraging fairness. Data transfer between two peers will only occur if the sending end of the connection is not choked, and if the receiving end is interested. Maintaining the choking state is key to BitTorrent achieving its goal of fairness as it allows peers to interact with each other and promotes fair participation in the network [148].

A series of initiatives to decentralise the BitTorrent protocol have been proposed which aim at replacing the tracker node (where information about the swarm is kept) by a trackerless mechanism based on Distributed Hash Tables. The most popular DHT versions, the one implemented by BitTorrent Inc., known as Mainline DHT [2], and that developed by Azureus-Vuze, known as Distributed Database [149], are both based on Kademlia [140]. An alternative strategy for peer discovery, known as Peer Exchange (PEX), has been implemented in some BitTorrent clients [150]. PEX entails querying present peers to find others. It is, however, susceptible to the formation of articulation points, whereby the failure of a peer partitions the swarm into disjoint components. It thus becomes necessary to flood in order to re-discover remaining peers. Anatomic P2P [151] is another attempt to decentralised BitTorrent employing a supernode caching system. Multiple supernodes act as trackers spread over the network. In addition, TRIBBLER [152] offers decentralised search capabilities based on social relationships between peers. By maintaining and exploiting social networks, TRIBBLER perform content discovery, including torrent files, and builds a peer list for data from its known peers. The system offers the possibility to recommend content to other peers and employs a gossiping protocol to search for those peers in the network.

2.4.2.2 BitTorrent in Wireless Ad Hoc Networks

The core design of the BitTorrent protocol offers some benefits to tackle a number of major problems faced when designing communication protocols for wireless ad hoc networks, specifically for Wireless Sensor Networks. Like other P2P content distribution protocols, BitTorrent helps to balance the communication load and avoid network partition. The latter features correspond to two of the key design goals in Wireless Sensor Networks. BitTorrent also employs the rarest piece algorithm which has been proved to introduce a higher degree of fairness in data replication. The BitTorrent mechanism of segmenting data files into pieces also adds an extra level of inner security to the system.

A node which does not contain all pieces of a data file provides only meaningless data when tampered with. Replication policies can be set up so as to avoid peers storing entire data files, thus achieving this added level of security. One additional advantage in using the BitTorrent design is that if every segmented data piece can be transported within a single packet, the probability of successful data delivery in the network increases and, hence, the reliability.

The BitTorrent tracker node contains the list of peers participating in the peer-to-peer overlay network, i.e. the swarm. While a central point of management is useful for traffic distribution control, it is a single point of failure with a high likelihood of suffering the hot spot and flash crowds problems. The inefficiencies produced by these two problems are exacerbated in multi-hop wireless networks as the hop distance to the central point increases. Thus, a decentralised approach is required which discovers closer peers capable of providing pieces of the data being searched.

In their works, Mario Gerla et al. proposed the SPAWN [153] swarming protocol as part of a project for cooperative downloading in vehicular networks called CarTorrent. SPAWN employs concepts from the BitTorrent protocol such as partial downloading and content sharing where pieces of data are acquired from closer vehicles in a peer-to-peer fashion. The mechanism has been designed to avoid the situation where a car needs to wait until getting in range with the next gateway to resume the download. Instead, SPAWN fosters collaboration amongst vehicles by interchanging pieces of data. Since content distribution protocols for vehicular wireless networks inherently have a high degree of churn, i.e. the continuous process of node arrival and departure in a network, peer discovery mechanisms need to be in place. To solve these issues, SPAWN employs a decentralised mechanism for peer discovery based on periodic gossiping [57] (see Section 2.4.5). A peer, from time to time, starts a gossiping process containing the peer list for a torrent. As the packet traverses the network, the intermediate node identifiers are aggregated in the packet as a mechanism to avoid cycles and provide an indication of hop distance between nodes. Nodes hearing the gossip packet forward the packet with a higher probability if they are interested in the torrent, such that the gossip process becomes more efficient. The protocol can also cache the last gossip heard from as many nodes as possible and select which gossip packet to rebroadcast at different rates. The selection of the packet to rebroadcast and the sending rate depends on the interest of the node in the torrent of the packet. These mechanisms act as peer list maintenance and discovery solutions, increasing the robustness when in the presence of high churn. However, the likelihood of finding a peer list, or a piece of data, in proximate nodes depends on the gossiping process. When a peer discovers another peer, it notifies its neighbours and includes the new piece bitvector. While UDP is employed for gossiping, TCP is used for content delivery employing the AODV [46] routing protocol for multi-hop transfers. In this type of networks peer proximity impacts the success and efficiency of the communications such that it needs to be taken into account in the selection of the peers from which to download pieces of data. The authors include the proximity of the peer in the peer-piece selection process and suggest the following policies: i) Rarest-Closest First, i.e. select the rarest piece and break ties with the closest peer, ii) Rarer-Closer, i.e. a weighted piece selection based on rarity and proximity, and iii) Closest-Rarest First, i.e. select the rarest piece amongst the closest. As expected for this type of network, the Rarest-Closest First policy produces the most efficient overall performance. Furthermore, the higher the swarm of peers for a torrent (popularity index), the better the performance; this indicates that collaboration should be fostered in this type of network for efficient distribution and scalability.

Around the same time as SPAWN, a cross-layer design was published by Rajagpalan et al. which adapts the BitTorrent protocol to operate in mobile ad hoc networks employing TCP and UDP [154]. The cross-layer approach employs a reactive routing protocol named ANSI for communication and searching purposes. The paper presents a centralised approach, BTI, which implements the basic functionality of BitTorrent where a tracker node keeps track of the swarm. In BTI, torrents need to be searched by employing the flooding process of ANSI in combination with the "expanding ring search" strategy if the search is not successful for the initial scope. The "expanding ring search" works by progressively increase the scope of the flooding process, in terms of hops, when the search is not successful. Both seeders and trackers reply if they contain the torrent. Once the tracker is contacted for the peer list, peers are contacted randomly. They employ a random piece selection strategy. Peers deciding to become seeders inform the tracker for how long and with what probability they should be selected for the peer list. An enhanced cross-layer decentralised version of BTI is proposed as BTM. It employs the ANSI routing protocol to collect and cache information about torrents and peers. The tracker node disappears at the cost of caching information and extra communications. In addition, torrents are rebroadcast using gossiping [57] from the initial seeder. It also uses the ANSI routing protocol for searching torrents and peers. Periodic scoped flooding packets are issued in search for peers, which reply with a response, to keep the peer list updated. Beacon packets are also employed to inform the neighbourhood that the node is a peer for a particular torrent. The authors employ a mechanism to increase data diversity by quickly distributing the content to a set of strategic nodes in the network, which immediately acquire the data from the initial seeder. These nodes, known as proxy seeders, are selected on reception of a torrent according to a hash function which assures that a full copy of the data is available at disparate points of the network. Experiments varying mobility, piece size and in the presence of network partitions proved that BTM outperforms BTI in terms of goodput and number of unique pieces received. However, these benefits come at the cost of higher MAC resource consumption. A recent paper [155] has implemented the BTM algorithm which enhances the protocol by implementing the next piece selection policies: rarest piece, random first piece plus rarest piece, rarest piece plus strict priority. The last strategy, based on completing sub-pieces of a piece before moving to download the next piece, was proved to perform better than the others, including the random policy, in terms of average goodput, energy consumption, control overhead and latency.

There have been approaches which employ a combination of the above mechanisms for cooperative content distribution enhancing proximity and diversity [156]. ElRakabawy et al. presented a similar decentralised mechanism which halves the time to download a file by avoiding the random-peer or closest-peer selection strategies [157]. Instead, they take into account the current traffic load and interference on the download paths to select peers, and thus download multiple pieces simultaneously by avoiding interference.

A later approach to adapt the BitTorrent protocol to wireless ad hoc networks was presented by Sbai et al. [158]. They focus in improving the efficiency of the content sharing by minimizing the download time and improving diversity and fairness. They adapt BitTorrent to operate using TCP for data communications and UDP for control packets. A decentralised mechanism for peer discovery is proposed where a peer initiates a flooding process for peer searching and advertisement purposes within a scope in the network. The Time-To-Live(TTL)-based flooding mechanism periodically sends "Hello" messages to discover peers containing pieces of the torrent; matching peers reply with a message. For ad hoc routing purposes, the Destination-Sequenced Distance-Vector (DSDV) [49] proactive protocol is employed. Proximity is taken into account when selecting peers for data transfer. Two tables are created, the Near Neighbours Table (NNT) and the Far Neighbours Table (FNT). The NNT contains nodes at two hop distance while the rest of the known nodes are stored in the FNT. Piece selection strategies - the local rarest first policy - operate only in the NNT table, and therefore peers send their piece update packets only to nodes in the NNT table. However, for every "q" peers selected from the NNT, a node is randomly chosen as a peer from the FNT. In addition, the piece which is going to be sent to the FNT node must not exist in the neighbourhood of the selected FNT node; the authors called this as "the absent piece strategy". This way, the mechanism increases piece diversity in the network while keeping the core of the download process in the vicinity, thus decreasing the completion time. The best "q" value, which establishes the number of peers selected from the NNT for each peer from the FNT, depends on the number of nodes in the network. These concepts were applied into the creation of BitHoc [159], a trackerless adaptation of the BitTorrent protocol employing TCP/IP over the 802.11 MAC Layer with the RTS/CTS mechanism. Another version of BitHoc is presented which instead employs the OLSR [160] proactive routing protocol [161]. Tracker agents running in nodes connect to each other to create a global distributed membership tracking service. In addition, the same authors explored the impact of peer selection strategies in the performance of BitTorrent over mobile wireless ad hoc networks [162]. In this case, their selection strategy of choosing a far neighbour in order to increase piece diversity proved to be very inefficient due to the high loss ratio of mobile networks when performing routing. Through simulations, near nodes are preferred as a solution for quick and reliable content delivery which exploit node mobility to increase piece diversity.

In a recent paper [163], Sbai et al. combine the above mechanisms to allow for content sharing in mobile wireless networks. They structure the overlay of peers of a torrent in a minimum spanning tree according to the hop distance. This way all peers are connected and efficient routing can be accomplished while increasing the chances that all peers complete the download of the file. However, this mechanism requires periodic messages to maintain spanning trees connected when peers enter or leave the overlay or when nodes move. Rather than only using the routing hop distance for its peer selection strategy, the logical hop distance at the overlay minimum spanning tree is also employed as a mechanism to select the list of near neighbours. As mentioned before, the authors include in the peer list a distant peer, i.e. a far neighbour, for every "q" near peers in order to increase piece diversity. However, too many seeders within the same area connecting to distant nodes would increase the overhead in routing providing a low degree of diversification. As a solution, the number of connections to far nodes was reduced according to the number of seeders in a particular area, i.e. the diversification area. The scope of the diversification area is calculated according to the number of successful pieces sent to far-away peers. They found that a higher number of concurrent torrents increases the routing overhead and thus the best strategy is to only connect to near peers. When the number of simultaneous torrents is low, the diversification area is self-regulated to cover a wider scope. In addition, mobility also contributes to piece diversification.

Most of the BitTorrent adaptations for ad hoc wireless networks in the literature place a strong emphasis on the concept of introducing proximity in the peer-piece selection strategies. A cross-layer design is a popular approach since routing information should be taken into account for efficient content distribution decision making.

2.4.3 Structured Lookup Mechanisms for P2P Wireless Networks

A range of mechanisms have been presented in the domain of Wireless Sensor Networks which create virtual overlays employing Distributed Hash Table (DHT) schemes for data search and retrieval. Many of them explore the impact that unreliable peers produce when entering and leaving the structure of the DHT. Solutions have been propose to ameliorate this effect by considering peer proximity thus enhancing DHT robustness. However, when facing lossy networks where peers suffer from short and long periods of connectivity disruption, and may even have mobility, DHT lookup schemes have proven to be ineffective.

One of the first approaches to distribute, store and query data in WSNs was published in the papers Data-Centric Storage in Sensornets (DCS) [164, 165] and Geographic Hash Tables for Data-Centric Storage (GHT) [166]. The authors presented the idea of distributing data by replicating it amongst nodes in the network such that a failure in the area will not affect the already gathered data. In order to distribute the data, a geographical routing protocol called Greedy Perimeter Stateless Routing (GPSR) [52] is employed. Geographical protocols route packets to a location rather than to an address, i.e. data-centric approach. Each node determines its own geographic coordinates, using for example GPS devices, and periodically announces its address and coordinates to its neighbours. In order to localize where data is stored, a Distributed Hash Table (DHT) is employed which hash data into a hash value which correspond to a location in the network. The hash function evenly generates keys amongst all the areas of the WSN, such that every key belongs to a location. In each location (zone) a node is responsible for the area, the home node, which is the closest to the coordinates of the hashed key. This node is responsible for the storage coordination in this area. To address node failures or topology changes, a refresh of each perimeter is periodically launched, which also confirms if the home node is still valid; if it is not valid, e.g. the node has failed or moved, a new home node is elected from those in the perimeter. The approach stores the same type of data in the same location. The location can become a hot spot or fail, therefore a technique called structured replication is employed. With structured replication, multiple locations can store the same data type and all of them are hierarchically structured so they update each other according to proximity. This technique adds extra redundancy at a higher communication cost.

Chord for Sensor Networks (CSN) [167] is proposed as a protocol which follows the principles of the DHT-based Chord algorithm [136] to provide a lookup mechanism capable of searching data with a logarithmic complexity. The authors claim it scales well as it maintains a finger table of $O(\log(n))$ entries and employs $O(\log(n))$ messages to locate data. Cluster hierarchies and overlays are formed such that virtual neighbours are closer geographically. The authors state that in mobile scenarios, a fast update method should keep the finger tables updated but it has not been evaluated. The same authors propose a case for P2P overlay networks in sensor networks [168]. This paper outlines the design goals to consider when applying overlays in sensor networks, proposing a DHT-based protocol known as Tiered Chord (TChord). This establishes a virtual overlay of powerful devices with slaves nodes, i.e. sensors with lower capabilities, hooked to them. A cluster structure is formed which can dynamically change and which takes into account physical proximity.

A solution for implementing a DHT in MANETS was proposed in Ekta [169]. The idea provides a peer-to-peer substrate by the integration of DHT-abstraction Pastry [138] in the network layer, employing DSR [47], a reactive routing protocol for multi-hop communication. Ekta updates its routing table and leaf set by snooping packets. A similar approach to Ekta, that integrates Pastry with the AODV [46] routing protocol, was later presented as MADPastry [170]. DHT tables are formed considering physical locality. It employs a set of random landmark keys spread evenly in terms of geographic coordinates which define temporal cluster nodes. Nodes do not hold a key, but rather landmark keys are assigned to new nodes in the cluster area by getting the cluster overlay prefix in addition to a random key. A node moving to another landmark cluster area is detected by periodic beacon messages. An approach to address mobility in MANETS using DHT was explained by Zahn et al. where a distributed segment tree (DST) is implemented on top of MADPastry [171]. In a paper by Zheng et al. [172], a DST is used to maintain the ranges of keys in a DHT, therefore obtaining the range query, i.e. all keys within a range, and the cover query, i.e. all ranges for a key. The same authors of MADPastry proved later in [173] that their DHT approach outperforms reactive and proactive routing protocols in MANETS. They state that there is no need to maintain a separate routing protocol for peer-to-peer communication when having a DHT-substrate. Another approach to implement Pastry at the routing layer has been studied with the OLSR routing protocol which does not take into account proximity in the overlay network formation [160].

Ghose et al. proposed a DHT approach with circular virtual overlays over a geographical routing protocol, the Greedy Perimeter Stateless Routing (GPSR) [52], which finds optimal routes by using local information at every hop [174]. This is achieved as every node has information about its position, typically by the use of GPS device; in a sensor node the use of a GPS might prove to be impractical due to its high energy consumption. Furthermore, the DHT overlay network is formed by "reference" nodes which represent events from an area, called event monitors. Bloom filters are employed for enabling attribute-based queries of the events at every event monitor node.

There has also been research on constructing DHT according to the topology of the network, such that virtual neighbours in the overlay are also close in terms of routing (hops). For instance, Topology-based DHT (T-DHT) [175] selects "reference" nodes by flooding mechanisms to offer support for other nodes to perform triangulation and to calculate the relative position of a node. Then, a virtual two-dimensional coordinate system is generated and mapped into a DHT which is used to store and retrieve data deterministically. A similar approach for data-centric storage and routing is also proposed in GEM [176], which uses graph embedding and tree routing, maintaining a two hop neighbourhood.

A routing protocol which applies DHT concepts on top of the link layer was presented as Virtual Ring Routing (VRR) [177]. The goals in designing the routing protocol were: i) to avoid flooding and ii) to avoid location-dependent addresses. On the other hand, scalability issues were not evaluated. Routing between virtual neighbours in the Pastry-based overlay ring is performed in such a way that there is knowledge about other nodes in the network, rather than only predecessors and successors. The complexity (in number of hops) to locate a virtual node can be reduced from $O(\log(n))$ to $O(\sqrt{n})$. This is achieved by taking into account link layer information of closer nodes when routing in the virtual ring. The protocol was compared in a sensor network of mica2dot against the Beacon Vector Routing (BVR) [178] protocol. BVR has been classified as a routing protocol for wireless networks and it has been evaluated in mobility scenarios with successful results. It should be mentioned that physical proximity is not considered when forming the overlay network and simulations have not taken into account power consumption.

Awad et al. presented Virtual Cord Protocol (VCP) as a DHT-based routing protocol which sits on top of the MAC layer [179]. It also takes into account geographical proximity (2 hops away) of virtual nodes to minimize communication. Like in VRR, routing is based on virtual and routing knowledge. Mobility scenarios were not evaluated in the approach.

Finally, ScatterPastry [180] is a DHT approach using Pastry on top, or integrated, at routing level. It is evaluated in a real WSN testbed called ScatterWeb. The authors try to minimize energy consumption and address scalability by using Pastry concepts both on top of, and in combination with, the Destination-Sequenced Distance Vector (DSDV) [49] routing protocol. Every node hashes its ID into a key and routing is performed by key prefix comparison such that a packet will be sent to the closest prefix in the routing table. By exchanging periodic messages, nodes failing or leaving an area can be detected thus refreshing table entries. A new node discovers its neighbourhood and finds a route to the node with its closer key in the network. Mobility is not simulated in this approach.

2.4.4 Unstructured Lookup Mechanisms for P2P Wireless Networks

The problem of data retrieval from a wireless sensor network acting as distributed data storage has also been explored employing unstructured lookup techniques. Research has been carried out on the concept of disseminating queries towards the area where data has been generated. For instance, the comb-needle mechanisms [181] are particularly renowned in the area of query dissemination. In this approach, an event is advertised or pushed into some of the nodes in the network forming a spike or needle shape where the source is the centre point; this is achieved by the use of geographically aware mechanisms. Data is then pulled by issuing queries from the sink forming a comb shape, with a distance in between the teeth of the comb equal to the length of the needles, such that all multiple queries sweep the area to eventually reach the event data. This type of discovery behaviour based on the dissemination of data/query was classified by the authors in the paper as: i) Push or Collection (event source nodes send data to the sink), ii) Pull or Querying (source nodes store data locally and sink nodes query it on-demand), and iii) Push-Pull (a combination of both approaches where source nodes push data in a way that can optimise the query/searching process). Alternative classifications for querying are based on the type of data being searched [182] as: i) Continuous Stream and ii) One-Shot Query. In this regard, the comb-needle mechanism is an unstructured one-shot Push-Pull approach where the searching process benefits from a previous data dissemination phase. This paradigm exploits the redundancy factor of information spread over the network where queries do not need to reach the node/area where the event is produced, but rather locate a near node which can provide either the data itself or information about the location of the data being queried. Hence, this problem becomes a distributed searching problem.

In Wireless Sensor Networks, unstructured lookup mechanisms come as a good solution for data searching in dynamic network topologies where structured lookup mechanisms (DHT-based) lose performance. The benefits of unstructured query systems are exposed in scenarios with a high degree of data replication over the network [183]. This situation makes the unstructured searching process efficient when compared to DHT-based methods at a much lower complexity and low maintenance cost.

Unstructured discovery systems can be classified in two categories [142]: i) Flooding and ii) Random Walks (RW). Moreover, random walks are subdivided in a) K-Parallel RW and b) Agent Cloning.

Flooding is a basic mechanism for searching based on Breadth First Search where nodes broadcast packets to all neighbours in a hop-by-hop basis (see Section 2.2.1). The broadcast behaviour of flooding increases the probability of reaching a high set of nodes in the search domain. The scope of the search is controlled with a Time-To-Live (TTL) counter which typically employs the hop count metric as a way to control the limits of the searching/dissemination process. Enhancements to flooding have been proposed to increase the efficiency of the searching process where an element of control is introduced, for instance duplicate packet suppression. Three types of enhancements can be defined [184]: i) Expanding Ring Search (ERS), Blocking Expanding Ring Search (BERS), and iii) Local Indices (LI). ERS floods the network with a TTL value - if no query reply is received after a given time, the TTL limit is increased progressively such that it covers a broader area. The so called Search Expansion Function (SEF) controls the TTL value and is defined as: f(x) = a + (x-1)b, where "a" is the TTL value at the start, "b" is a fixed increment, and "x" is the search attempt. The difference with BERS is that the latter starts the next flooding process from those nodes where the maximum TTL value is reached; this node acts on behalf of the source node. The source node is then responsible for sending a packet to cancel the process when a query response is received. A delay needs to be in place before attempting the next flooding. The authors in [185], introduced a shorter delay according to the TTL value in BERS to smooth the delay process, achieving lower latencies than ERS and BERS for number of hops greater than 3. Finally, Local Indices (LI) store the content of the nodes from an area X hops away such that the node acts on behalf of these nodes. Only a subset of these nodes are required to maintain local indexes with a separation of 2X+1 amongst them for efficient coverage. If the nodes enter and leave the network, or they change position to another area, an update is required.

Random walks are stochastic processes which aim at visiting the vertices of the network graph randomly in a unicast manner until the search query is satisfied [186]. A query packet, known as the walker, is sent to a randomly selected neighbour in a process which is independent from previous visited nodes. The walker traverses the network from node to node, thus the overhead in neighbours reception communication is low at the cost of potential high latencies. Parallel random walks increase the reliability of the searching process while decreasing the latency at the cost of a higher overhead in communications. However, the number of parallel random walks, k, needs to be estimated to control the trade-off. Additionally, agent cloning is a type of random walk where agents are software entities which travel the network in a random walk fashion until the search is complete. Agent packets accumulate information from each node visited in order to perform a backtracking routing process. An agent can be seen as an explorer packet which sets the path for future routing or subsequent biased random walks. At each node, an agent has the capability to clone such that the searching process is distributed, i.e. k-parallel random walks. For instance, Gaber et al. employ an intelligent cloning mobile agent in combination with a biased random walks technique for unstructured resource discovery [187]. The previously presented comb-needle algorithm can also be categorized as a biased k-parallel random walk technique. In the paper "Empirical evaluation of querying mechanisms for unstructured wireless sensor networks" [182], a random walk technique with a three phase handshake mechanism is employed to increase the reliability of the unicast packet delivery process. In addition, a memory-based self-avoiding random walk mechanism is also implemented to avoid the most visited nodes; the list of visited nodes is transported in the agent. The mechanism has been compared to Flooding and the authors convey that: "For isolated networks with little or no interference, flooding has high reliability and outperforms random walks in terms of delay and energy costs. For networks with considerable interference, the reliability of flooding is drastically reduced and random walk based approaches might be better suited".

Wireless Sensor Networks are characterised by a high degree of data redundancy over lossy networks and dynamic topologies which make unstructured discovery mechanisms a suitable choice for data searching. In the area of WSN, different approaches exist which employ the above methods as the basis for unstructured discovery.

In a paper from 2004 entitled "Asymptotics of Query Strategies over a Sensor Network" [188], the probability of a query being successful was studied in source-only scenarios (Pull), source-receiver driven scenarios (Pull-Push), and spatially-periodic caching scenarios (Pull-Push with Data Redundancy). In all these techniques the source node triggers random walks to query receiver nodes. In the Pull-Push strategy, the one employed in comb-needle approaches, the receiver also triggers a random walk which leaves a trail of information of the type of data provided. This idea has been employed for routing in the so called "Rumor Routing [44]". In the latter approach, the Pull-Push with Data Redundancy, the data itself, or information on where the data is located, is spatially stored at different points in the network on a periodic basis. The authors conclude that the probability of a source-only query being unsuccessful decays to zero only logarithmically fast, while it becomes polynomially fast when employing either a source-receiver driven search or the spatially-periodic caching approach. They also convey that the Pull-Push strategy requires less memory consumption than the caching approach, and thus is a good strategy for searching in unstructured networks. Furthermore, remote distributed storage with a high degree of data replication not only offers efficient queries but also enhances all the benefits outlined for data distribution systems such as robustness and hot spot avoidance.

Dimakis et al. proposed an unstructured solution on how to enable ubiquitous access to distributed data in WSN [189, 190]. The goal is to retrieve a set of distributed data packets by querying as many nodes as the number of data packets required. Storage nodes are randomly selected and data is pre-routed with a complexity per data node of $O(\ln(n))$. Distributed erasure codes are employed as a solution to achieve reliable distributed storage. An erasure code is a forward error correction code which can detect and correct "k" errors or erasures in a structure of "n" redundant bits/symbols, where $\frac{k}{n}$ is the code rate and is less than 1. With distributed erasure codes, each node can store, for instance, at most one data packet, such that when more than one packet is to be stored the data is diffused to accommodate it to the available size. The diffusion is performed with randomized linear codes such that by querying any "k" nodes it is possible to retrieve all the "k" data packets with a high probability. Data then is available to any collector in the network by querying "k" or more nodes and reconstruct the data from the set of "k" packets. It is assumed that a routing layer with geographical information like GPSR [52] is provided and packets are not lost.

Rachuri et al. presented Increasing Ray Search (IRS) [191], a highly scalable, density independent unstructured solution to achieve efficient search in high density scenarios when compared to ERS and Random Walks. This paper aims to minimize the number of messages employed in the query mechanisms in an unstructured network in a non-deterministic way. The paper employs a Pull technique sending one-query at a time until the destination is found. Each query path traverses a ray shape perpendicular to the sink, with a width equal to the transmission radius of a node. By varying the angle with respect to the sink, the query packet is only forwarded by those nodes which match the angle criteria. If the destination is not found, the angle is varied to cover another ray area. The nodes need to know their location to apply geometric calculations of the angle.

In ACQUIRE [192], a packet is injected containing an active query which employs random walks, biased random walks or even a predetermined path. Each node receiving a query packet performs, if information is not available, an on-demand request for information, within a scoped distance in terms of hops, in order to improve on the selection of the next neighbour. When the active query is completely resolved, the query reply is sent directly to the source.

A recent paper presents a combination of biased random walks for Pull-type unstructured search where sensor nodes are considered stationary [193]. Sensor nodes need to be aware of their neighbourhood and the distance in hops to the sink, i.e. the level. A set of protocols are presented which employs a combination of Several Short Random Walks (SSRW) and Level Biased Walks (LBW). In LBW a query is propagated from the sink in such a way that it is forwarded to a neighbour with a higher level until it reaches a limit. If the query is not successful, another is started. A query walk can alternate a sequence of steps using LBW and then perform SSRW. The idea of LBW is to increase the coverage by reducing the correlation of visited nodes. The protocols are efficient in terms of energy, communications and latency when compared to simple random walks and multiple random walks techniques.

In the category of biased random walks, searching based on Bloom Filters (BFs) [88] come as a probabilistic-based memory-efficient solution (see Section 4.4.1 for more detail about BFs). The Bloom Filter is an efficient structure for data compression which has been used as a query mechanism in peer-to-peer networks in the Internet. The filter acts as a probabilistic membership structure in the form of a bit vector which can be queried to find out, with a certain likelihood, whether an element has been previously stored. In this regard, a BF can produce false positives which could affect the decision-based mechanisms. In unstructured content discovery for peer-to-peer networks, BFs have been used as memory-efficient mechanisms to contain a synopsis of the content stored in a node. The BF-based search aims at disseminating the BF amongst the nodes in the network such that the complexity of the search is reduced, thereby saving resources and decreasing the latency of the query process. However, the BF structure needs to be periodically refreshed to update the network on the new content of the node. Different approaches to BF-based searching exist. The simple one is that where the BF storing a description of the content in a node is spread to all the nodes in the network. Queries are then resolved by checking the BF table and the use of a routing protocol. On the other hand, BFs have been employed as data-centric routing mechanisms [194–196] where queries are forwarded based on the BF information cached at a given node about its neighbours. The routing process is usually performed by a greedy approximation which selects the next node whose BF best matches the BF in the query. In this type of routing, BFs need to be disseminated over the network in such a way that a gradient of information is formed towards the BF source. This is achieved by applying a decay factor on the filter as the BF is disseminated; this is usually based on proximity, hop distance, and consists of the aggregation or the union of filters. Approaches such as the Attenuated Bloom Filter (ABF) [197] aggregates filters according to their proximity in terms of hops while the Exponential Decay Bloom Filer (EDBF) [198] applies a decay proportional to the hop count to introduce noise in the received BF.

In the area of WSN, some research has also been conducted employing Bloom filters for data-centric routing. Hebden et al. [195] employed a hierarchical structure where a cluster formation protocol assigns nodes to monitor an area via caching data from packets in a counting Bloom filter - a BF which keeps count on how many times an element has been inserted. Query routing is performed between cluster heads according to the BF structures. Li et al. adopted a flat-network approach presenting the Scope Decay Bloom Filter (SDBF) as a structure to disseminate event hints [196]. SDBF is based on a mechanism where a decay factor is only applied to the BF once it is at a distance larger than a predefined hop-based scope. Bloom Filter-based searching has been proved to increase the query success rate, reduce energy consumption and decrease latency when compared to flooding-based or random walks at the cost of BF dissemination overhead.

2.4.5 Data Dissemination in Wireless Sensor Networks

Data dissemination techniques over wireless sensor networks have been proposed to reliably distribute data with the aim of achieving a high data delivery and low latencies while minimizing communication and energy consumption. First approaches in the area of dissemination employed the flooding and gossiping mechanisms (see Section 2.2.1). Flooding and gossiping, in their basic forms, are considered as inefficient solutions with low complexity which do not optimise the process of data dissemination. For instance, controlled flooding employs prune mechanisms, duplicate and rebroadcast control, or expanding rings to make the process efficient. Comprehensive research has been conducted in the area of data dissemination to increase the efficiency of the process while maintaining low complexity.

One of the first approaches in the design of data dissemination mechanisms for wireless sensor networks was introduced with a protocol called Directed Diffusion [42]. In Directed Diffusion, source nodes describe data using attributes, introducing the concept of data-centric routing (replacement for the tradition address-centric approach). Around the same time, the SPIN [43, 199] protocol tackled the problem of data dissemination from a data-centric perspective with a negotiation-based paradigm in which nodes decide whether to acquire the data from neighbour nodes. These two protocols establish the concept of metadata packets, known as "interests", which travel the network to inform other nodes of the availability of data while preparing the path for data communication. In this regard, data dissemination protocols, where the sink propagates the interest and the source responds with data, might be classified as Sink Oriented Data Dissemination protocols. Examples of this type are Directed Diffusion [42, 75, 200], TTDD [201, 202], Declarative Routing Protocol (DRP) [203] and GRAB [76, 77]. The design of Sink Oriented Data Dissemination protocols revolves around the subscribe-publish paradigm where nodes disseminate the interest or query towards potential sources of data such that data is forwarded to nodes which manifest an interest. On the other hand, Source Oriented Data Dissemination protocols are characterised by source nodes initiating the dissemination of metadata towards the sink. On reception, the sink performs some data acquisition process which might be based on negotiations. A good example of this is SPIN [43, 199]. The Source Oriented Data Dissemination protocols are based only on source nodes publishing and disseminating metadata towards the network in order to alert nodes on the availability of data. This type of protocol can easily be enhanced with a subscribe mechanism if the dissemination process can be controlled.

The adoption of each scheme brings different benefits and drawbacks related to scalability, network topology dynamics, resource-efficiency and real-time data acquisition.

Moreover, dissemination protocols have been employed for multi-hop network reprogramming where large objects of data/code are disseminated over the network in a reliable manner. Examples of these types of protocols are Pump Slowly Fetch Quickly (PSFQ) [204], GARUDA [205] and Reliable Multi-Segment Transport (RMST) [206] which employ hop-by-hop retransmissions for reliability. Multihop Over-the-Air Programming (MOAP) [207] is another approach to network reprogramming over multihop networks which delivers the whole object to a node before the latter become a source for the next hop. In [208], the Multihop Network Reprogramming protocol (MNP) adds a sender selection algorithm which attempts to guarantee that at most one source is transmitting within a neighbourhood to minimize collisions. Other approaches rely on network coordination to minimize collision at a higher complexity cost. For network reprogramming in TinyOS [96], two data dissemination protocols are available: i) Deluge [209], and ii) Typhon [210] which is based on the basic mechanisms from Deluge and improves its performance. On the other hand, DIP [211] and DHV [212] are employed in TinyOS for small to medium data size dissemination. They operate by finding nodes in need of data updates in order to keep consistency over the network.

Deluge, Typhon, DIP and DHV employ the Trickle [99] protocol for maintaining code updates. Trickle employs beacons to disseminate data which fits in much less than a packet, and self-regulates the periodicity of the update beacon based on the changes in the neighbourhood. Trickle is efficient in terms of packets as it suppresses broadcasts if a recently up-to-date packet has been overheard. Trickle also adapts the beacon periodic time-out according to the network activity and the up-to-date status of the neighbours such that beacons might not be transmitted for an extended period, thus reducing resource consumption while scaling well. In doing so, Trickle exponentially increases the beacon interval to reduce communications when there are no changes in the neighbourhood, until it reaches a maximum value of T_h. Conversely, Trickle decreases the beacon interval towards a minimum value, T_l, when the neighbourhood is out-of-date. Setting T_l to a low value increases the overhead in communications while improving latency.

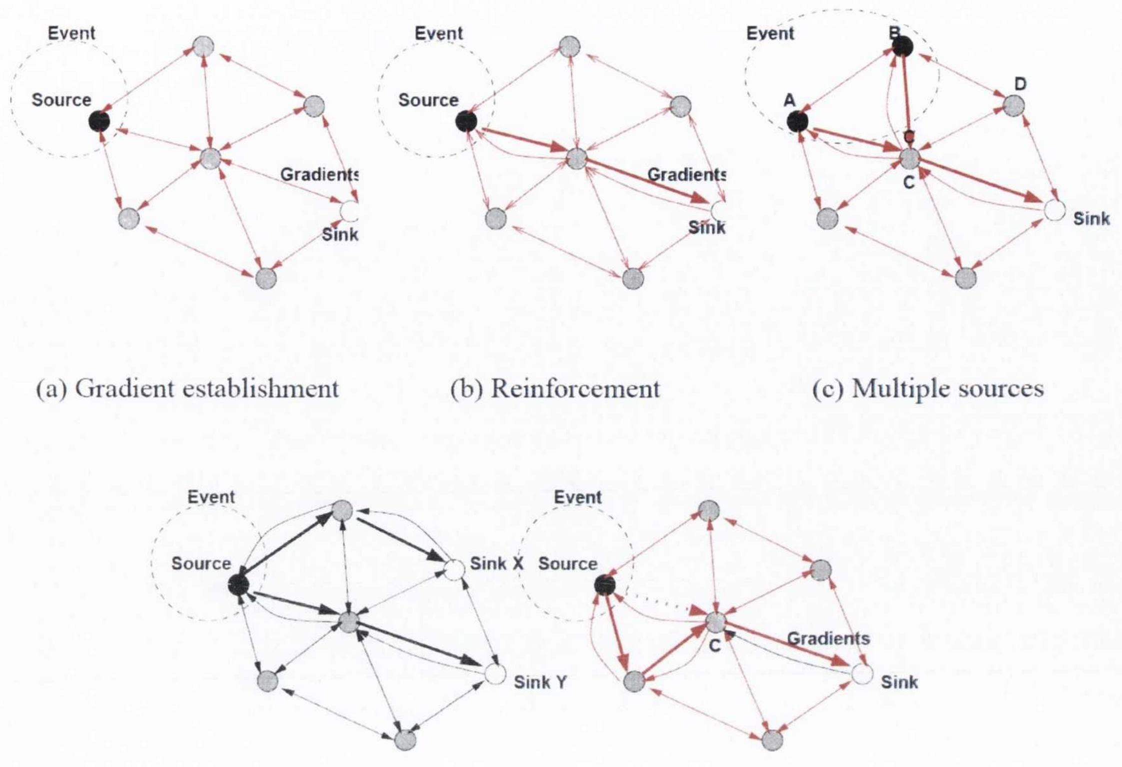
All of the above dissemination protocols are based on hop-by-hop data communication rather than supporting end-to-end connectivity and their reliable mechanisms operate at 1 hop distance. However, they aim for reliable data dissemination in an efficient manner which is of relevance for the work presented in this thesis. Next, an insight into some of the the most relevant data dissemination protocols is given.

2.4.5.1 Directed Diffusion

Directed Diffusion (DD) [42,75,200] was presented in the year 2000 as a communications paradigm for coordination of wireless sensor nodes to perform distributed sensing of environmental phenomena. DD design seeks to disseminate data in an energy efficient and robust manner while suiting in-network processing and scaling efficiently in large networks of different densities. In addition, DD is application aware and data-centric in the sense that data is represented, or named, with metadata in the form of attribute-value tuples which describe the properties of the data. Instead of routing data in an end-to-end address-centric fashion, DD is only aware of its neighbours to forward packets, therefore there is no global knowledge of the topology. The dissemination protocol does not aim at finding optimal paths, but rather trades off energy efficiency for robustness and scalability, while following the communications behaviour of ant colonies when building robust and scalable transmission paths.

In Directed Diffusion any node, called the sink, can start communication by transmitting an interest describing the type of data which needs to be acquired. The metadata packet is called "interest" and is propagated through the network via different mechanisms such as flooding or directional flooding based on location or cached data (see Figure 2.10 a). Since interests are not reliably transmitted, the packet is periodically resent by the sink to refresh and reach most of the nodes in the network. For control purposes, the metadata packet not only contains the type of data being queried (and associated attributes, operators or location information) but also incorporates parameters which indicate the interest timestamp, expiration time, and the "interest interval" at which those potential sources will be sending events to the sink. The "interest interval" value is employed to regulate the flow of information, to select paths, and to establish the data rate of packets from source to sink. The propagation of interest packets creates a gradient towards the sink node. At each intermediate node an entry for the gradient is described by the direction, i.e. the next hop address, and the data rate, i.e. the interest interval. In this regard, a node which receives an interest generates an entry if one does not exist already, and associates one or more gradients with this interest entry. The reason is that multiple sinks might be querying the same interest with different intervals, and using different paths.

Once a node receives the interest, different application-based local rules can be applied to decide whether to keep forwarding it. Nodes store interests in their cache. When a node detects an event, it searches its interest cache for matching entries. For those matching, the node (known as a source node) sends the so called "exploratory events" towards the sink node, containing the interest plus some extra parameters on the sensed phenomenon. The source node sends the packet to the relevant neighbours from which the interest was received. A node receiving the exploratory event checks its



(a) Gradient establishment (b) Reinforcement (c) Multiple sources

(d) Multiple sinks (e) Repair

Fig. 2.10: Directed Diffusion Routing Protocol Phases [200].

interest cache for matching entries. This mechanism allows for duplicate packets control and loop prevention, and it also regulates the exploratory events data rate which has been previously defined for the interest for each of the gradients according to their intervals. Exploratory events flowing towards the sink also establish a gradient towards the source node called exploratory gradient. Once exploratory events arrive to the sink, possibly along multiple paths, the sink selects one of its neighbours according to local rules to reinforce one of these paths (see Figure 2.10 b). Same rules are applied at intermediate nodes and therefore a single gradient path is chosen for which data from source to sink flows. An example of a local rule can be the selection of the first neighbour to reply which will selects a route with a low latency. The mechanism to reinforce the path is based on the previously mentioned "interest interval" which is initially set to a low data rate. The sink reinforces a neighbour by increasing the data rate, i.e. the interval value, in such a way that the receiving node must reinforce the next neighbour on the way to the source node. The path is then established from source to sink through local interactions according to node's local rules which might be defined by the application. Reinforcement also operates in scenarios with multiple sinks and multiple sources where data can flow from different paths or partially disjoint routes which might be inefficient in terms of communication and energy (see Figure 2.10 c,d). Reinforcement rules and mechanisms can also be triggered by intermediate nodes when degradation in the link or the node's energy level is detected. This way, reinforcement can be used for local repair (see Figure 2.10 e). Finally, DD counts on negative reinforcement mechanisms for these situations where more than one path is reinforced at different event periods. One mechanism employs soft state reinforcement where the path times out if it is not reinforced. In addition, DD employs the same mechanism employed for reinforcement but in this case reducing the interest data rate; a reduction in the "interest interval" indicates a receiving node that it needs to reduce its data rate and therefore that it is no longer included in the gradient for data transmission for that sink. A different local rule is employed by each node to select those nodes to receive negative reinforcement. For instance, those neighbours from which no new exploratory events have been received may be selected for negative reinforcement as compared to the rest of the neighbourhood. Loop removal is also possible with negative reinforcement.

Directed Diffusion has been compared against flooding and omniscient multicast - where all route information is available - in fields ranging from 5 to 250 nodes, varying the numbers of sinks and sources and employing the IEEE 802.11 MAC layer in the ns-2 simulator [90]. While maintaining the density, DD has been evaluated in terms of average dissipated energy, average delay, and event delivery ratio in networks with a 10% to 20% node failure with and without in-network aggregation. The authors claim DD outperforms the idealized omniscient multicast dissemination in terms of delay and energy dissipation.

2.4.5.2 SPIN

One of the most cited papers in the area of data dissemination in wireless sensor networks presents a family of adaptive protocols called Sensor Protocols for Information via Negotiation (SPIN) [43, 199]. The paper emphasizes the higher level of accuracy, robustness and sophistication in capturing the physical phenomenon which a collection of wireless sensor devices can achieve. In doing so, performing in-network processing and data transportation over the network are key elements. Thus, the authors exploit the idea that effective dissemination information strategies need to be in place for WSN which suit the constraints of this technology and effectively relay data. One of their arguments is based on the inefficiencies caused by simple approaches such as flooding or gossiping. According to this, SPIN was designed to operate efficiently and conserve energy. SPIN is based on a negotiation protocol composed of three stages. First, a node with data to disseminate advertises a smaller description of the data, the metadata, by sending an advertisement (ADV) packet. A neighbour node receiving the ADV packet decides whether to acquire the data from the sender. If so, the node sends a request packet, REQ, containing the metadata information for the requested data. On reception of a REQ packet, the node sends data to the requesting neighbour node. The negotiations and data transactions only occur within the neighbourhood of a node (1-hop). Not all the nodes might be interested or capable of acquiring the data.

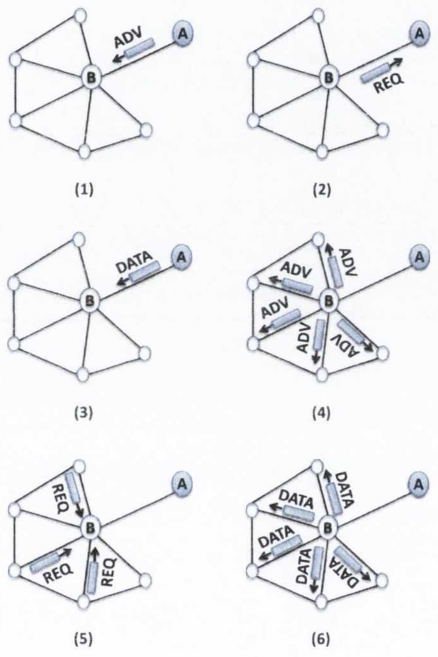


Fig. 2.11: SPIN-PP Protocol [199]

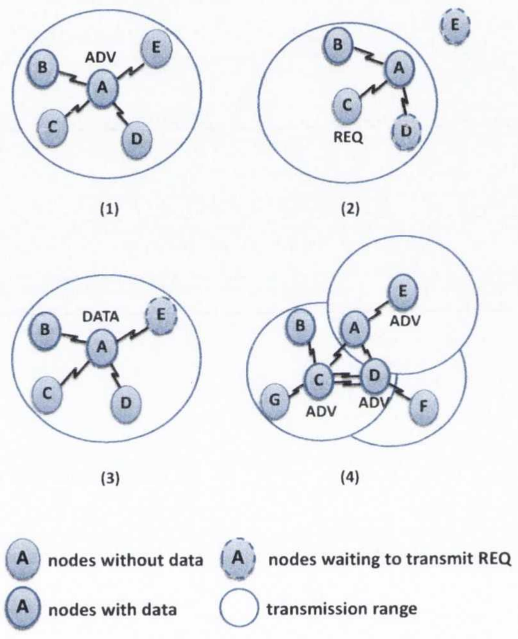


Fig. 2.12: SPIN-BC Protocol [199]

Four versions of SPIN have been designed: SPIN-PP, SPIN-BC, SPIN-EC and SPIN-RL. SPIN-PP operates the three stages of the protocol in a unicast point-to-point fashion (see Figure 2.11), therefore a mechanism to discover the neighbourhood needs to be in place. SPIN-BC employs broadcasting to send the ADV packet and data to all of its neighbours (see Figure 2.12). Since neighbour nodes receive broadcast packets, if they decide to request data while hearing a request for the same data, then they do not need to send the request as data will be broadcast and received by all neighbours. Re-trial mechanisms with implicit and explicit acknowledgements are in place to account for packet collisions. An energy-efficient enhancement of SPIN-PP, known as SPIN-EC, describes a resource manager interface which SPIN exposes to allow the node to account for the current level of resources. This way the node decides its involvement in the next communication activity according to the status of its resources. Therefore, a node with a low energy level may decide not to request data since it will not be capable of handling its reception. In addition, SPIN-RL enhances the reliability of SPIN-BC by keeping track of the ADV packets received and re-requesting data if a time is elapsed. Moreover, a random delay is applied before broadcasting the ADV packet to minimize congestion.

With its negotiation-based approach, SPIN solves the implosion problem and even the overlap problem of Flooding (see Section 2.2.1). The SPIN three stages protocol occur within 1-hop distance, therefore the dissemination of information depends on neighbour nodes requesting and receiving data. The SPIN family achieves high performance in data dissemination at low cost in terms of complexity, energy, computation and communication when compared to classical flooding and gossiping.

2.4.5.3 TTDD

Ye et al. in their paper "A Two-Tier Data Dissemination Model for Large-scale Wireless Sensor Networks" (TTDD) [201, 202] address the problem of data dissemination over large-scale sensor networks where multiple mobile sinks query data source nodes. A set of static data source nodes constitute a virtual grid overlay for query and data dissemination. TTDD divides the network into a grid of square cells by appointing the closer source node at each intersection of a cell as the representative for a dissemination point; this node is known as the dissemination node. Dissemination nodes proactively update each other by propagating announcement messages and storing information about the virtual tier. Each dissemination point is represented by a dissemination node. A mobile sink belongs to a cell at a time, so it has assigned a dissemination node. A sink floods the cell to reach the closest dissemination node. The dissemination node then forwards the query from the sink to its upstream dissemination node in the direction towards the source node. Therefore the query is forwarded either to the source node itself or to another dissemination node which is already receiving data from the source. The query packet leaves information in the nodes along the path for performing reverse path routing from source to sink when sending data packets. Each node in the grid is aware of its own location via GPS or other techniques and it is also static, such that greedy geographical forwarding is employed to construct the grid structure. There will be a grid structure for each data source. When compared to Directed Diffusion in a 200 node network, TTDD has similar success rates, while they both scale similarly to the number of sources and stationary sinks in terms of energy consumption. TTDD consumes less energy when the number of sinks is small while Directed Diffusion consumes less energy when aggregating queries from multiple sinks. TTDD experiences a lower delay than Directed Diffusion when the number of sources increases.

2.4.5.4 Deluge

Deluge [209] is an epidemic protocol for reliable dissemination of large data objects to many nodes in the network. It works by executing state machine rules locally at each node. The simple mechanism is based on nodes advertising their most recent version of the data object. Neighbour nodes receiving beacons of an older version respond with the profile of the newer version object (metadata). Out-of-date nodes then request data from neighbours and data is unicasted. A request to another neighbour is made if the link performs badly while transmitting data. This is a three phase handshake protocol which serves as a mechanism for testing bi-directional communication between neighbours. Eventually the new version of the object spreads over the whole network to those nodes which are out-of-date. The advertising rate is dynamically adjusted to trade propagation speed for resource consumption. Deluge is density-aware in the sense that redundant control packets are suppressed to minimize contention according to the number of neighbours participating. This is controlled with Trickle in an effort to minimize resource consumption and quickly propagate data in a reliable way. Objects are split onto pages which then are chunked into packets of fixed size for a controlled, efficient and reliable transmission which fosters spatial multiplexing. With spatial multiplexing, nodes do not need to have the whole object to keep on propagating pages. Pages are sequentially transmitted in such a way that higher transmission priority is assigned to lower sequences in the case where a request decision is to be made. In the data transmission, density-awareness is also achieved by using random backoff and snooping techniques to cancel overheard redundant data. Reliability is ensured with selective acknowledgements and bitvectors for missing packets with the latter being checked for integrity with CRCs. Packets are sent in round-robin order to provide fairness among requesters. The authors describe the importance of using suppression of duplicate control packets in the performance of the dissemination algorithm. Without using suppression the contention could be as high as to produce a deadlock. They also mention that limiting the number of senders affects the performance. Deluge has been evaluated in terms of dissemination completeness, energy consumption, propagation time, packet loss rate vs. distance, and messages sent and received. The performance evaluation has been carried out in TOSSIM [105–107] over a grid topology of 20 x 20. The authors claim that "Deluge can disseminate data with 100% reliability at nearly 90 bytes/second, one-ninth the maximum transmission rate of the radio supported under TinyOS [96]". They also argue that "dissemination is inherently slower than single path propagation and identified establishing a tight lower bound as an open problem".

2.4.5.5 Typhon

Typhon [210] has been designed following the basis of Deluge [209] to reliably disseminate large objects of data to nodes in the network for reprogramming purposes. The protocol divides the large objects, which can weight as much as 100KB, into small objects, called pages, which are themselves partitioned for transmission to fit the packet payload size. The aim is to disseminate the data as fast as possible trying to minimize collisions by simultaneous transmissions at non-overlapping nodes. Even though the authors aim at reducing the idle listening time for energy efficiency, they agree that duty cycling the node will incur a high complexity when trying to achieve reliability. The protocol injects large objects from a designated node, i.e. the sink or base station. Before sending an object, Typhon employs Trickle [99] to quickly disseminate a small object containing metadata information of the new object, such that nodes on reception decide whether the object is to be acquired. After hearing a metadata packet containing a new object id which needs to be acquired, a node start to broadcast packets requesting a page (page requests increment sequentially until the object is completed). Neighbours hearing the request packet reply with a packet offering the page if they already have it. These nodes then wait for the node to send a stream request packet asking for the receiver to start transmitting all the data for the page. Back-off mechanisms with acknowledgement packets and retrials are in place to minimize collision and increase reliability. Typhon employs the overhearing property of the wireless transceiver such that nodes acquire packets for a page which they do not contain if the opportunity arises. However, if all the data pieces are not received, the page is discarded. Typhon nodes control their timers in such a way that they increase the probability of handling a request for a page over requesting a page when it has recently acquired one. This way the dissemination process reduces its latency. Typhon also leverages spatial re-use to accelerate the propagation of data through the network by employing channel switching. Metadata and control packets operate in the default channel while data streams are sent on a randomly selected and agreed channel. However, nodes involved in getting data from a node may miss metadata packets. As a solution, nodes switch to the default channel as soon as data has been transferred and wait a period of time after a metadata packet is received to allow for this packet to get disseminated. Typhon has been implemented in TinyOS 2.x [96], simulated with TOSSIM [105–107] and tested in a real-testbed of 22 motes. Typhon has been tested in terms of completion time (dissemination), energy consumption, object size impact, density effect, overhead and packet loss impact. Typhon reduces the energy consumption and dissemination time by a factor of three as compared to Deluge [209].

2.4.5.6 DIP

Lin and Levis present DIssemination Protocol (DIP) [211] as an adaptable and scalable dissemination protocol which maintains data consistency for all nodes in the network. As in other dissemination protocols, data items are assigned tuples of the form (key,version), where "key" identifies the data item and "version" serves to indicate the freshness of data. DIP requires the set of keys to be predefined such that all nodes contain the whole range. DIP uses the Trickle [99] algorithm to broadcast advertisements containing versions of the data items a node stores. Trickle adapts to the neighbourhood conditions and suppresses redundant beacons, so can quickly adapt when data is out-of-date while keeping communications low when the network data exhibits consistency. Protocols can use Trickle to advertise versions of multiple data items, i.e. serial scan, or adopt a parallel approach where each data item is assigned a Trickle instance, i.e. parallel scan. Considering T as the total data items, the parallel scan broadcasts a beacon per data item and thus it has a latency of $O(1)$ and a cost in communications of $O(T)$. Conversely, the serial scan requires $O(T)$ intervals to transmit a given tuple, i.e. a latency of $O(T)$, while incurring in a communications cost of $O(1)$. DIP further proposes a new hybrid approach called "search" which achieves $O(Nlog(T))$ in communication cost and $O(log(T))$ in latency, where N is the number of new items. "Search" employs hash trees to aggregate hashes of versions for a subset of data items. If a node contains a different hash to that received for a range of data items, DIP reacts by sending sub-range hashes. In this way, by descending the hash tree levels, the advertisement complexity is reduced to $O(log(T))$ transmissions while the detection is $O(1)$. When the network is consistent, a hash of the entire range of nodes in the network is sent. 'Searching" is more efficient than scanning mechanisms if there are few data items to update. However if the number of data item updates is high a scanning method has less cost in transmissions. DIP takes this behaviour into account and adapts the advertisement mechanism from "Search" to scanning methods when the number of data items to update is high. Additionally, DIP employs an estimate value which indicates the probability of a data item being different. This value gets updated with information from all packets received, and also regulates the number of hash beacons sent and the hashed data items range. When the estimate value is high, DIP stops descending the hash tree and changes from sending hash beacons to employing scanning methods where beacons contain the actual (key,version) tuple to unambiguously identify the missing data. Once the missing data is identified, data messages are broadcast to the node with an old version, which in turn rebroadcast the data packet in case neighbours do not have it. Hashes are encapsulated in "summary" messages. Additionally, a Bloom filter structure is also encapsulated in the "summary" messages. The Bloom filter provides extra information which indicates whether a (key,version) tuple is a member and thus it can be used to circumvent some hash tree levels. This reduces advertisement messages and aids quickly identification of missing data. DIP has been shown to send 20-60% fewer packets and perform 200% faster than dissemination protocols employing randomized serial and parallel scanning techniques.

2.4.5.7 DHV

The authors of DHV [212] present a data dissemination protocol which maintains data consistency in all the nodes in the network. This protocol falls in the same category as DIP [211] and also uses the Trickle algorithm. DHV employs tuples of the form (key,version) to keep track of data item updates. The main improvement with respect to DIP is that DHV exploits the following observation: "if two versions are different, they often only differ in a few least significant bits of their version number rather than in all their bits"; this assumes that the version number is incremented by 1 with each update. Thus, it is not necessary to transmit all the version numbers for all the keys in the network to identify which version has changed. Rather, DHV creates a matrix with the bit vectors of the version numbers for all the keys of the data items, where the matrix must be sorted by key, i.e. rows, using the same algorithm for all the nodes in the network. By employing bit slicing, efficient searches on the matrix can identify the keys for which the version has changed. DHV phases are identified as: "Difference detection, Horizontal search, and Vertical search", and that is where the name arose from. The matrix is hashed and broadcast in a SUMMARY message via the use of Trickle. When the received hash at a node differs from the local hash, then a difference is detected. This node then enters an "horizontal search" phase, where the node broadcasts a checksum of all versions, known as the HSUM message. At the receiving node, the HSUM is used to identify which bit indices for all versions differ. At this time, the "vertical search" phase is started, where the node broadcasts packets containing the bit slice, i.e. VBIT, of the bit indice(s) differing, i.e. sends the differing column(s) in the matrix. In case more than one column differs, the bit slice for the least significant bit index is sent first. Upon receipt of the VBIT message, the node compares it to its local VBIT in order to identify the differing (key,version) tuple(s). Once identified, the tuple(s) are broadcast in a VECTOR message. On reception of a VECTOR message the node verifies if it has a newer version: in such a case it send a DATA message, otherwise it broadcasts its (key,version) tuple. DHV employs its own packet suppression mechanism as it decides whether to broadcast the packet when the Trickle broadcast interval is elapsed. If there are messages to send of the type 1-DATA, 2-VBIT or 3-HSUM, it sends them following the outlined priority. If there are not packets of this type then, if the node has heard two or more messages in the last interval, the broadcast is suppressed; otherwise the node sends VECTOR or SUMMARY messages. DHV achieves $O(N)$ in communications cost while $O(1)$ in latency. The authors claim that in simulations and on a real micaZ testbed DHV reduces by half the number of messages, converges in half the time, and reduces the number of bits transmitted by 40-60% when compared to DIP.

2.4.6 Significance for this Thesis

This section has reviewed research aligned with the ideas and mechanisms behind the TinyTorrents (TT) WSN communications architecture presented in this thesis. The system comprises a P2P data distribution protocol, the TinyTorrents protocol, functioning on top of the UMG routing protocol. An insight into, and taxonomy of, P2P data distribution protocols in wireless network has been initially provided. Here, the TT architecture is classified as purely decentralised, with an unstructured overlay network for content discovery.

Next, mechanisms of the BitTorrent protocol have been identified as key in the design of TT. A set of BitTorrent adaptations for wireless networks has been presented which exhibit parallels with TT. These are SPAWN [153], BTM [155] and BitHoc [159], protocols which employ TCP/IP communication. SPAWN follows the unstructured content discovery and advertisement paradigm employing gossiping and making use of a reactive routing protocol. However SPAWN is intended for vehicular networks with high churn and opportunistic connectivity. On the other hand, BitHoc and BTM are presented as decentralized BitTorrent solutions where searching of peers and tracker is based on flooding. TT employs similar peer and piece selection strategies and content discovery and advertisement mechanisms to BitHoc and BTM. However, TT employs the concept of "partial trackers" where every peer becomes a tracker for an un-defined proximate set of peers. Additionally, TT exploits information on the advertisement and dissemination of torrents both at a routing level and in a data-centric manner for discovery purposes. Following this metaphor, content discovery mechanisms for P2P wireless networks are presented as structured and unstructured. The TinyTorrents WSN solution employs a set of unstructured searching mechanisms to locate either metadata information, which can lead to the data, or the data itself. Unstructured lookup mechanisms, such as flooding and random walks, have been selected for use in TinyTorrents as: i) they do not lose performance in dynamic network topologies where structured lookup mechanisms (DHT-based) do, ii) they become more efficient as the data replication degree increases across the network, and iii) they are solutions of much lower complexity and lower maintenance cost.

Finally, some of the key data dissemination protocols in WSN have been summarised. Existing data dissemination protocols in WSN focus on i) pushing data into the network for the maintenance of data consistency, such as DIP [211] and DHV [212], and for reprogramming purposes, such as Typhon [210] and Deluge [209], or ii) pulling data from source nodes to a reduced set of (mobile) sink nodes, such as Directed Diffusion [42]. The TinyTorrents communications architecture targets those scenarios where data needs to be distributed in a selective manner to a particular set of nodes in the network. TT does not relay data in an epidemic manner such as SPIN [43, 199], Deluge or Typhon, but rather employs end-to-end, multi-hop connectivity amongst collaborative disparate nodes. In this sense, TinyTorrents fosters collaboration by downloading data from multiple points, thus reducing congestion, whereas Directed Diffusion only reinforces and aggregates the path(s) following the gradient's sink(s). Additionally, TT does not use physical location techniques, such as GPS devices, for discovery or routing purposes, unlike protocols such as TTDD [201, 202].

2.5 Summary

This chapter has introduced the current state of the art for each of the system concepts described in this thesis, i.e. the UMG routing protocol and the TinyTorrents data distribution protocol. A review of mechanisms and protocols which are related to, or impact the design of, the TinyTorrents WSN communications architecture proposed in this thesis has been provided.

Section 2.1 has introduced the field of Wireless Sensor Network research while identifying the key aspects which make WSNs different from existing wireless technologies. The subsequent section has provided an overview of routing protocols for WSN.

Next, section 2.3 has described the principles of gradient-based routing in wireless networks. This is the main research focus of the Ubiquitous Mobile Gradient (UMG) routing protocol (see Chapter 4). UMG operates as the main routing substrate for the TinyTorrents protocol. It supports point-to-point, multipoint-to-point and point-to-multipoint communication for those scenarios where a node's application requires end-to-end on-demand connectivity, and/or data advertisement and discovery support. To this extent, a comprehensive description of gradient-based and data collection routing protocols for wireless sensor networks has been provided which thoroughly covers the research field of UMG. In this review, a core set of mechanisms has been identified as the basis of the gradient routing protocols, including UMG. The key features of UMG have been expressed and compared to existing gradient-based routing protocols where UMG provides a set of functionalities for reliable and versatile communication which aim to serve as a routing solution for a wide spectrum of WSN applications.

Finally, section 2.4 has provided a detailed review of techniques for data distribution in ad hoc wireless networks, with an emphasis on the mechanisms of BitTorrent for content distribution. The TinyTorrents protocol presented in this thesis (see Chapter 5) employs concepts from the BitTorrent protocol which are adapted to support selective data dissemination in large networks of sensor nodes. In addition, the TinyTorrents protocol employs a combination of unstructured and self-organised mechanisms to discover peers in the swarm of a torrent. A review of structured and unstructured lookup mechanisms for decentralised content distribution systems is also included in this section. Furthermore, data dissemination protocols aligned with the TinyTorrents protocol concept are also addressed herein. The TinyTorrents WSN architecture is then compared with DIP and DHV, two of the the dissemination protocols presented in this section.

In the next chapter, the TinyTorrents communication system is described.

Chapter 3

The TinyTorrents Framework: Data Distribution in WSN

This chapter provides a description of the top-level component-oriented design of the TinyTorrents framework. The architecture is documented in terms of two functional entities: i) the Gateway-Internet communications architecture and ii) the WSN-Mote architecture. This chapter then provides context for the remainder of this thesis within the overall TinyTorrents framework.

3.1 The TinyTorrents Framework

The concept of TinyTorrents (TT) leverages existing P2P content distribution ideas from traditional wired networks to replicate and store data within a Wireless Sensor Network and on the Internet. Traditional P2P networks are characterised by the way they distribute the content amongst participants in the network, thereby balancing the load and ameliorating the burden on central nodes. Resources are spread amongst nodes forming self-organised, redundant, fault-tolerant and scalable networks. By the adoption of traditional P2P content distribution ideas, the TinyTorrents concept challenges the general paradigm of a WSN as a network of devices with sensing capabilities which retrieve data, fuse it, and send it to the gateway to be stored or utilised. This concept results from the vision of colonies of sensor devices, with mobility capabilities, which mutually cooperate to take decisions toward achieving a common task. The TinyTorrents framework has been designed to provide the next step in the future collaborative environments of WSN. It provides mechanisms to distribute data in such a way that the WSN can be converted into an autonomous decision-making system capable of operating independently and also interoperating with other networks via the Internet. The TinyTorrents framework provides a generic and versatile way of distributing data from the WSN to the Internet, interconnecting the networks in a transparent manner. The interconnection between the Internet and the WSN employs existing peer-to-peer (P2P) ideologies where data is encapsulated into torrents and stored at different nodes in the Internet. This is achieved via the use of the BitTorrent [146] protocol to create a distributed fault-tolerant database.

An initial version of the TinyTorrents concept was implemented by Karsten Holger Fritsche in his M.Sc. thesis entitled "TinyTorrent: Combining BitTorrent and SensorNets" [213]. The thesis presents an adaptation of the centralized version of the BitTorrent protocol for operation in wireless sensor networks. This work established the basis of P2P communication in WSN while incorporating primitive communication with the Internet BitTorrent network. This work was a simple proof of concept for TinyTorrents in that all sensor devices were placed at 1-hop distance with no routing functionality. The author of this thesis took over the TinyTorrents project and integrated support for multihop communication in small size networks while improving the peer selection process and providing a transparent and bidirectional integration with the Internet BitTorrent network [214]. This version was a multihop centralized BitTorrent adaptation for WSNs where the management of the central coordinator prevented for the system to scale and to be fault tolerant. This thesis further extends that concept to a fully decentralized, fault-tolerant, scalable version of the TinyTorrents system for cooperative and reliable data distribution within WSNs.

3.1.1 P2P Data Distribution in WSNs: A New Communication Paradigm

Traditional communication paradigms for WSNs focus on the collection of data from multiple sources by a reduced set of sink nodes. Sink nodes may be employed for further data processing and typically are connected to other networks, such as the Internet. In this paradigm, data is pushed out of the WSN. Conversely, data can be pushed into the WSN, mainly for reprogramming purposes. However, in these scenarios data flows in one direction towards a set of receiver nodes in the WSN. While this has been commonly accepted for many years, mainly due to the limited resources of sensor devices, more complex behaviours can arise from the use of WSNs as colonies of sensor and actuator nodes.

Consider a scenario where static nodes are deployed at some premises and other nodes with mobile capabilities, for instance miniature robots, are assigned to perform a given distributed common task. Robots would be interested in receiving data from fixed sensor nodes as input to their decision making process. It will also be highly important for a set of robots to receive data from each other for distributed decision making. This type of data flow requires a new communications paradigm for

WSNs where data is moved from a producer to a set of consumer nodes within the WSN such that the dissemination process is selective and efficient in terms of communications. Data will evolve as it flows between a set of nodes within the WSN where nodes can adopt consumer and producer roles. Predominantly, nodes will take the role of consumers when receiving and distributing a data file of interest, while they will become data producers less frequently, for instance when new data is created or a decision is made locally according to the governing application. In addition, the identity and number of consumers acquiring data files from a given producer might change dynamically according to the interest of the consumers in the data, thus generating virtual overlays of interest. The interest of a consumer in acquiring a data file, and the publication of data files by a producer, will depend on the application layer requirement. Thus, the application will be in charge of controlling the replication of the data files in the network and, consequently, the robustness of the network to the unavailability of nodes or data at any given time.

To provide a communications solution for the above scenario, existing dissemination mechanisms could be employed which: i) employ an epidemic process where most of the nodes in the network receive data, or ii) establish common paths for specific communication, typically towards a common sink. These solutions are inefficient mechanisms in terms of communications, providing a low degree of versatility for efficient and cooperative communication and for discovery of a subset of nodes in the network. Where these approaches fail to provide an efficient selective dissemination process, P2P content distribution concepts offer a more suitable solution for distributing the network traffic and fostering cooperation when performing a selective data distribution process. While this is a different dissemination paradigm for WSNs, the concept still follows the publish/subscribe strategy where a node publishes what type of services are offered and consumer nodes decide whether to subscribe to acquire the data. The core difference lies in the way data is transferred amongst those cooperative nodes which form a virtual community sharing a common interest.

This thesis explores the idea of distributing a file of data as opposed to the concept of continuous sensing. A file of data can represent sensed data for a period of time; a file includes metadata such as the timestamp when the data is created. Thus, the size of the file impacts the freshness of the data when received by other nodes. However, that does not stop use of the TinyTorrents communications architecture for continuous dissemination of data, where the oldest value of each file will have an extra delay according to the period of time the file represents.

Finally, the P2P content distribution system proposed in this thesis employs concepts and mechanisms from the BitTorrent protocol. While, there have been approaches to adapt the BitTorrent protocol to wireless networks, following the TCP/IP communications architecture (see Section 2.4.2.2), a straight forward adaptation to WSNs is not feasible. This is mainly due to the challenges posed by the WSNs technology: i) the potential large number of devices forming a WSN require a scalable approach which accommodates different neighbourhood densities, ii) the constrained sensor devices demand efficient data structures for communication and storage, iii) the unreliable multi-hop wireless communication requires efficient peer selection strategies which take into account the proximity of nodes, as well as reliable routing solutions which provide information to improve the efficacy and efficiency of the distribution process, even in the event of node failure. More specifically, while the centralised version of the BitTorrent protocol can be adapted to suit the WSN domain, the technology demands a decentralised, scalable and fault tolerant solution which requires a complete redesign of the peer discovery mechanism and peer selection strategies.

3.2 The TinyTorrents Architecture

The TinyTorrents framework comes from the idea of employing both existing technology and concepts from the BitTorrent [146] protocol to disseminate WSN data in a peer-to-peer fashion, having each sensor node acting as a logical peer in the Internet BitTorrent network. Hence, the TinyTorrents framework can be divided in two sides of operation which transparently integrate each other:

- The Gateway-Internet side: The architecture enables the connectivity of the WSN to the Internet via the use of the Vuze P2P content distribution client-server application [149]. The interconnection between Vuze and the WSN has been achieved through the development of the "Vuze TT Plugin", depicted in Figure 3.1. It deals with translating WSN torrents into BitTorrent format and making the data accessible in a variety of ways.
- The Wireless Sensor Network side: A communications architecture runs in each node of the WSN capable of providing selective data distribution over the WSN. The architecture for each sensor node, i.e. mote, is depicted in Figure 3.1 under the heading "Mote Architecture". The architecture provides functionality for each node to connect to the "Vuze TT Plugin" and thus act as a gateway or base station. This is achieve via the "Base Station" protocol usually employing serial communication. A decentralised BitTorrent-like protocol specifically adapted to suit the constraints of WSN, known as the TinyTorrents protocol is in charge of performing reliable and efficient data distribution in sensor networks. The TinyTorrents protocol needs to sit on top of a multi-hop routing protocol operating as a cross-layer fashion; the TinyHop and the UMG routing protocols have been designed for this purpose.

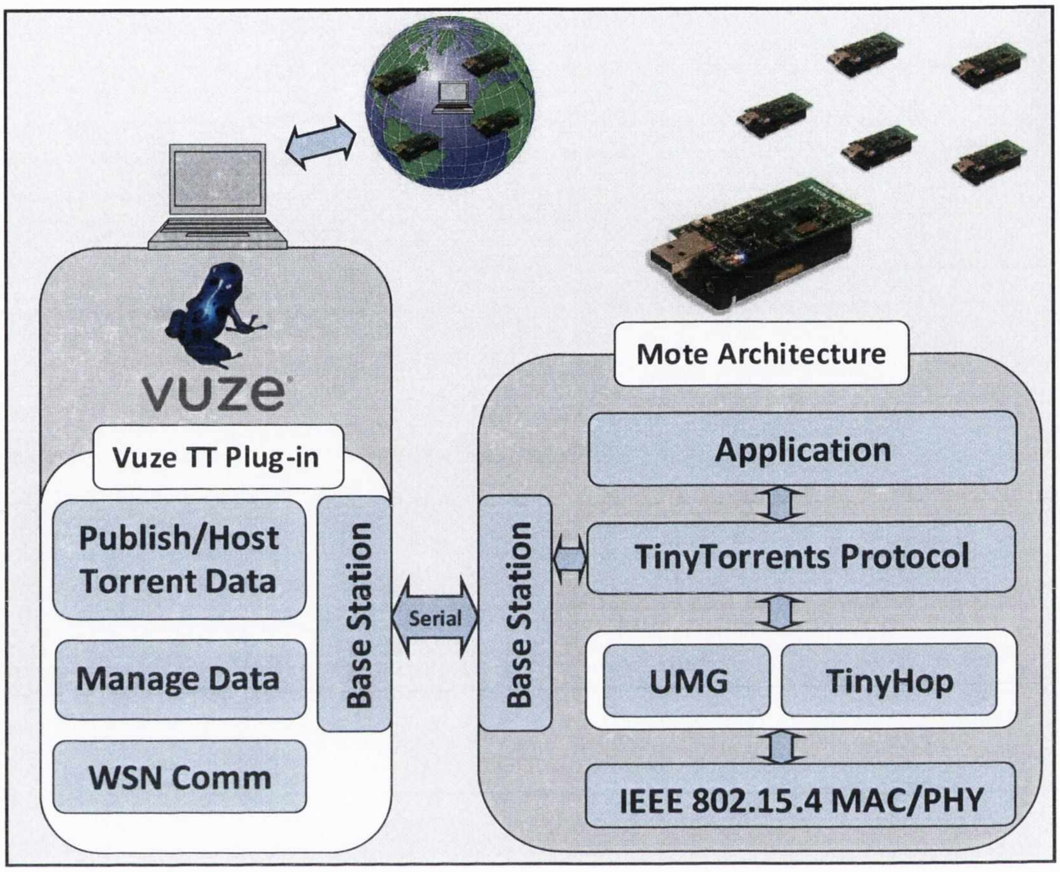


Fig. 3.1: The TinyTorrents Framework - Communications Architecture

3.2.1 Gateway-Internet Architecture: Vuze TT Plugin

The Vuze TT Plugin deals with making the sensor data available to the Internet by using the BitTorrent protocol [146]. The BitTorrent protocol is the most popular P2P content distribution mechanism currently being employed in the Internet. The concept is based on the cooperative distribution of data by participant nodes in the network. Data is represented via a metadata file known as the "torrent" which contains the control information required for the node to find other nodes to transfer data from and to. Torrents files are published and data files are transferred amongst nodes in a process called "torrenting". Nodes containing pieces of data for a data file belong to the swarm of the torrent. Nodes can join and leave a swarm. Thus, data is available to be fetched from multiple points in the BitTorrent network, therefore increasing the reliability, the security and the efficiency in the distribution of traffic load.

In order to access the Internet BitTorrent network, the Wireless Sensor Network needs a gateway at some point in the network. Thus, a gateway device with both Internet and WSN connectivity is required. Sensor devices will be accessed and represented in the Internet via the gateway. In fact, each mote can be connected via the gateway in a bidirectional manner. Furthermore data from motes is uniquely represented in the Internet BitTorrent network by the use of metadata. Metadata describing the BitTorrent torrent file carries descriptive information, including information about the gateway, the sensor device, the time at which the data was generated, and the metadata description in the WSN.

For the purposes of accessing the Internet BitTorrent network, the open source client-server application "Vuze", formerly "Azureus", has been selected [149]. This application implements the BitTorrent protocol. Vuze allows users to view, publish and share digital data content and it can be configured to work in a server mode, i.e. hosting data and torrents. In addition, Vuze can operate in a centralised manner, by using or providing tracker functionalities, or make use of distributed hash tables to decentralise the location of the peers. The Vuze application has been chosen over other existing ones because it offers a versatile API for the development of plugins. Customized plugins make use of the BitTorrent protocol to enhance their functionalities.

For the purposes of the TinyTorrents framework, a dedicated plugin has been developed which operates as a management component, bridging the WSN to the Internet BitTorrent network. The plugin, called "Vuze TT Plugin" (see Figure 3.1), employs the functionality provided by Vuze to create, publish and host data via the use of torrents. Moreover, the plugin manages data storage from the WSN and implements mechanisms to pull and push data from and to the WSN. This module allows for the transparent interaction with the WSN and provides an interface to perform queries. Moreover, the "Vuze TT Plugin" implements the WSN Communications component which is in charge of the connection to any mote in the network. This component performs handshakes and implements a data communication protocol for the transmission of the TinyTorrents data messages. Any node in the network is capable of acting as a base station for the transmission of data to and from the WSN.

Data is retrieved from the WSN in a peer-to-peer fashion translating WSN torrents into BitTorrent format, so that every sensor node can be seen as a functional peer of the BitTorrent network. Currently this architecture works with IPv4, using a single IP to represent the WSN on the Internet BitTorrent network. Future enhancements will see its integration with IPv6 over LoW Power wireless Area Networks (6LowPAN) to provide a single interface of communication per sensor node [215]. Regardless of the protocol used, IPv4 or IPv6, a gateway between the WSN and the Internet needs to be in place. This is essential in order to prevent uncontrolled access to sensor devices which can consequently drain their energy. When data is retrieved from the WSN into Vuze, all the benefits of P2P distribution can be employed to distribute data over the Internet; that includes Web 2.0 services like RSS.

As seen in Figure 3.1, the "Vuze TT Plugin" can be divided in three main areas of functionality grouped in modules. All of them are fully integrated, exposing user interfaces for the interaction of the user with the system and the WSN. The "WSN Comm" component deals with the communication with the WSN. The "Manage Data" component involves all the tasks regarding the creation, process, advertisement, storage and query of data. Finally, the "Publish/Host Torrent Data" component refers to the interaction of the Vuze TT Plugin with the Vuze application and all the mechanisms employed to advertise and host data.

3.2.1.1 WSN Communications Component

Connectivity to the Wireless Sensor Network is achieved via connection with any sensor device within the WSN. Every node in the sensor network implements functionality to operate as a base station and establishes connection to the "Vuze TT Plugin". The "Vuze TT Plugin" allows for the connection to multiple platforms providing the sensor devices implement a communications protocol known as the SensorNet protocol. The protocol is defined in both sides, at the "Vuze TT Plugin" and at the "Mote Architecture" (see Figure 3.1).

The SensorNet protocol deals with the connection to the sensor device and the transmission of all the TinyTorrents protocol messages between the gateway and the mote in a bidirectional way. The protocol utilises the same message structures defined in the TinyTorrents protocol (see Chapter 5) for the connection/disconnection and query data messages. The "Vuze TT Plugin" implements the TinyTorrents protocol functionality and injects and receives messages, to and from the WSN, in a complete transparent manner. This way, a higher degree of integration of the motes into the Internet BitTorrent network is achieved.

The SensorNet protocol handles the reception of messages from the WSN and acts accordingly. When receiving torrent messages, the SensorNet protocol holds the decision to proceed with the fetching of data. The base station node waits for the Vuze TT Plugin to reply to start the fetching process. On reception of data messages, the SensorNet protocol verifies the integrity of the data and stores the pieces of the data file - this is performed by the "Data Management Component". Once the data has been fully downloaded for a torrent, the process of publishing and hosting data takes place - this is managed by the "Publish/Host Torrent Data Component".

3.2.1.2 Data Management Component

Data coming from the WSN is stored on files sorted by the torrent description. The data and associated torrents are saved in a permanent file which is preloaded every time the Vuze TT Plugin boots up. The TinyTorrents framework implements a mechanism to describe data employing human readable metadata. Metadata comes from a dictionary of terms which the sensor devices employ to describe the generated data. Metadata terms are compressed and transported in a Bloom filter structure over the WSN. When the gateway receives the TinyTorrents torrent file, the terms are extracted from the structure to create the BitTorrent torrent file description.

The TinyTorrents protocol offers a query mechanism which employs the aforementioned metadata to search for particular data in the WSN. In this regard, the "Vuze TT Plugin" implements a mechanism to generate query messages according to user search criteria. The user employs a graphical interface to combine terms of metadata, time intervals of data generation, accuracy of the search, and target node id, in order to form a query message which is injected into the WSN network. Matching query replies are received and a list of available torrents is presented for the user to decide which to fetch. The query mechanism employs the service discovery mechanism provided by the TinyTorrents protocol.

3.2.1.3 Publish/Host Torrent Data Component

Once data files have been received from the WSN, tagged and stored, they need to be made available for download in the BitTorrent network. For this purpose, a BitTorrent torrent file is generated based on the metadata and data received from the WSN. Torrent files are tagged with the terms of the vocabulary extracted from the TinyTorrents torrent message. These terms, together with the data creation timestamp, the id of the generator node, and the description of the gateway, describe the torrent file and make it unique in the BitTorrent network. The "Vuze TT Plugin" enables the creation of agent applications customized to the user requirements. The agent application decides whether to publish certain data, or if data needs to be aggregated, to combine a digest file represented by a unique torrent file. The torrent file might be hosted locally or in another server to be retrieved by other interested peers. The torrent file contains descriptive information about the data which represents and also stores control information indicating the tracker node. Initially, the gateway representing the WSN is the only seeder of the data and, as peers show interest and fetch the data, the peer list grows and therefore the traffic load is distributed amongst participant peers for this particular data item. In addition to using the BitTorrent network to fetch data in a P2P manner, Vuze offers Web 2.0 instruments to retrieve data, e.g. Really Simple Syndication (RSS). Moreover a HTTP server which exposes web services has been implemented to offer a direct way to retrieve information about the torrenting status of the WSN. Multiple web services can be implemented and started by the agent application.

3.2.2 Mote Architecture

The top-level communications architecture which enables unstructured selective data distribution in wireless sensor networks as part of the TinyTorrents framework is depicted in the "Mote Architecture" in Figure 3.1. The remainder of this thesis focuses on the design and implementation of the protocols which shape the "Mote Architecture", and is where the contributions of this thesis lies. The "Mote Architecture" comprehends the TinyTorrents protocol which can not work in isolation and needs to sit on top of a routing protocol operating in a cross-layer fashion.

The TinyTorrents protocol employs P2P data distribution techniques to replicate data amongst peers in a fair, selective, efficient and reliable manner across the WSN. The protocol, which employ concepts from the BitTorrent protocol, make use of metadata files to describe data, and generates unique keys to identify data files. Similar to the BitTorrent protocol and other content distribution protocols, files are partitioned into pieces which are then distributed amongst other peers which, in turn, become distributors. This approach suits WSN where selective data replication is required as a mechanism to protect against network failures and to push data closer to consumers. In addition, the concept of every peer being able to become a distributor ameliorates the burden at the data source node and balances the network traffic to help avoid partitions. In addition, some security is inherently obtained with this approach as i) multiples copies of data can be compared to check for authenticity and ii) a policy where nodes would not contain all the pieces of a data file will protect the integrity of the data if a node gets tampered with.

Two increasingly complex designs of the "Mote Architecture" have been developed. Initially, in version 1, the TinyTorrents protocol operates with a single tracker, a central node which manages the list of peers containing the data, i.e. centralised version. The results of this work are summarised in [214]. However, this thesis focuses on the decentralised version of the TinyTorrents protocol, version 2, which has been designed to operate in a fully decentralised manner where the single point of failure, the tracker, is eliminated. One approach to achieve decentralisation is to employ distributed hash tables (DHT) which create a virtual overlay network for locating data. DHT's are not suitable for use in sensor networks in their current form. Due to the constraints of WSN, proposed algorithms in the literature take into account proximity of the sensor nodes to create the DHT structure. However, as seen in Section 2.4.3 nodes leaving and entering the DHT structure, and the presence of mobility in the network, have proved DHTs to be an expensive method of decentralisation in terms of communication for WSN. This is the reason why an unstructured approach is considered as a decentralised lookup mechanism to discover proximate peers holding pieces of data. The approach is efficient in terms of communication and does not require constant maintenance or updates. In addition, efficient peer-piece selection mechanisms are presented which distribute the traffic load in the network in a fair manner fostering diversity of the pieces of data files while seeking to reduce the overhead in communications.

In the routing layer, two routing protocols, UMG and TinyHop, have been designed to address the constraints of WSN while offering cross-layer support to the higher layer, i.e. the TinyTorrents protocol. The routing protocols suit the paradigm of torrenting in sensor networks. An initial protocol, called TinyHop [3], was designed as an on-demand, end-to-end, flat-topology, reliable routing protocol. The second protocol, called Ubiquitous Mobile Gradient (UMG), has enhanced and extended TinyHop by using the concept of gradients while tolerating environments where sinks can show mobility. UMG has been tailored to provide the type of end-to-end communication required amongst consumer and producer nodes while leveraging the routing process to provide the unstructured service advertisement and discovery mechanisms required for the data distribution process of the TinyTorrents protocol. Unlike existing routing protocols, UMG incorporates a set of properties and functionalities which suit the TinyTorrents data distribution paradigm. For this purpose, UMG provides for selected nodes to advertise data and create paths towards them by following the reactive paradigm where unstructured virtual clusters are created amongst nearby consumer and producer nodes. In this way, UMG performs routing amongst proximate consumer and producer nodes of the virtual cluster in a reliable manner, while employing the distribution mechanism of the TinyTorrents protocol to spread data to distant consumer nodes - thereby achieving scalability. Indeed, UMG has been designed to deliver data with end-to-end reliability over paths of 6 to 8 hops, under the assumption that routing at higher hop distances is inefficient and impractical for selective data dissemination. While the TinyTorrents protocol employs reliable mechanism for communication amongst consumer and producer peers, UMG also implements end-to-end reliability at the routing layer. Blending end-to-end reliability within the routing layer has been a practical decision for two reasons: i) the relative short length of paths which does not incur a high use of control messages, and ii) the faster and more certain awareness of peers status gives the TinyTorrents protocol a more effective and efficient response in the peer selection, data distribution and discovery processes.

3.3 Summary

This chapter has presented the TinyTorrents framework architecture; a communications system which transparently integrates Wireless Sensor Networks with the Internet BitTorrent network for efficient and global distribution of data. The design of the TinyTorrents framework is presented as 1) the Internet-Gateway side, which deals with the interconnection of the WSN with the BitTorrent network, and 2) the WSN-side, which implements an architecture for sensor devices capable of providing selective scalable data distribution in Wireless Sensor Networks. The main functionality of both the Internet-Gateway and the WSN communication systems have been documented.

The major contributions of this thesis arise in the WSN architecture of the TinyTorrents framework which is described in the remainder of this thesis as a two-layer communication system. For routing layer functionality, the UMG routing protocol has been developed. UMG has been designed to support and enhance the decentralized version of the TinyTorrents system and is fully explained in Chapter 4. The upper layer infrastructure, a selective data distribution algorithm known as the TinyTorrents protocol, is detailed in Chapter 5.

Chapter 4

The Ubiquitous Mobile Gradient Routing Protocol

This chapter presents the Ubiquitous Mobile Gradient (UMG) protocol, a reactive gradient-based dynamic routing protocol designed to provide the communications support required by the TinyTorrents protocol (see Chapter 5) to enable scalable and selective data distribution in WSNs. UMG evolves from the design of TinyHop in such a way as to provide a much more versatile communication solution, in terms of connectivity between nodes, by supporting moderate mobility while providing a more efficient and reliable protocol overall.

The novelty of the UMG routing protocol resides in the communications versatility provided and the collection of mechanisms which enhance the efficiency and reliability of the routing process. In this sense, UMG:

- provides reliable point-to-multipoint, multipoint-to-point and point-to-point communications while following the reactive paradigm,
- combines address-centric and data-centric routing concepts to provide service advertisement and discovery to higher layers,
- employs a backoff-based reliable controlled flooding mechanism for the progressive formation of gradients according to the hop distance,

- allows for the coexistence of multiple gradient updates from the same sink node thus facilitating nodes which could not get updated to participate in the forwarding process without incurring in loop formation,
- employs an efficient short memory cache to enforce reliability when descending the gradient, and to decide which acknowledgement packets are to be broadcast when ascending the gradient with the goal of providing end-to-end reliability,
- integrates an opportunistic relative mobility detection mechanism to signal when sink nodes change neighbourhood and consequently update their gradient within an estimated scope.

This chapter is organised as follows: The first section provides a description of the UMG routing protocol and explains the rationale of the key features which drive its design in the context of WSNs. The following section details the protocol operation and places the four main phases of the protocol into context. There then follows a detailed description of the design and implementation of the Gradient Spread phase. Next, a mechanism for service description, advertisement and discovery is presented as an integral element of the protocol. The design and implementation of the three data transport phases: Gradient Descent, Local Repair and Acknowledgement, are described in Section 4.5. In the following section, a relative mobility estimation mechanism for the memory-efficient and opportunistic detection of changes in the neighbourhood of a node is presented and fully explained. Finally, a summary of the chapter is provided and the main contributions of UMG are identified.

4.1 A Dynamic Gradient-based Routing Protocol for WSN

A dynamic gradient-based reactive routing protocol, known as Ubiquitous Mobile Gradient (UMG), has been designed for Wireless Sensor Networks. The UMG design supports efficient and reliable communication for a variety of application scenarios in the area of WSNs. It employs the gradient concept (see Section 2.3) while offering reliable mechanisms for the creation, update and navigation of the gradient field. It supports point-to-point, multipoint-to-point and point-to-multipoint communication. UMG is address-centric in the sense that node addresses are employed for routing, but it also integrates support for data-centric routing by making use of Bloom filters as compressed mechanisms for storing data descriptions. Bloom filter-based descriptors are integrated in the gradient formation thus providing a mechanism for nodes to advertise their services. Services range from sensed or fused data to resources available at the node. A service requested by a node is known as "Interest". In this sense, UMG acts as a service advertisement protocol and facilitates the data-centric searching process. The possibility of each node to be capable of describe itself allows for applications to create multiple overlays according to node's services or node's interests in data, thereby making possible the formation of dynamically changing communities of wireless sensor nodes. Each node might belong to multiple communities indicated by the node's descriptor acting as the node's profile.

One of the goals of UMG is to avoid the use of periodic messages, assuming that areas with no data communication are not updated, whilst network areas with activity benefit from the opportunistic communication in order to maintain its connectivity status up-to-date. In other words, UMG follows the reactive paradigm, employing eavesdropping to update neighbours rather than using periodic updates.

UMG has been designed to operate in networks where every node can take the role of consumer and/or producer. This way nodes can be sinks and sources at the same time. It has also been designed to tolerate sink and source mobility by opportunistically detecting a node's relative change in neighbourhood, and consequently reacting by locally updating its gradient. Therefore, detecting relative mobility promptly, while following the reactive paradigm, is paramount. In this regard, mobility is detected by using a structure of temporal shift Bloom filters for the efficient storage of historical overheard information. A probability-based model is employed to assess the mobility state of a node according to the Bloom filter's structure. In addition, a recursive estimation approach introduces a higher level of accuracy to the mobility assessment by evaluating the degree of mobility of the previous temporal states.

4.2 Ubiquitous Mobile Gradient Operation

The Ubiquitous Mobile Gradient (UMG) routing protocol is based on the idea that a node wishing to be contacted must spread its gradient first. The proper formation of the gradient is a key element in the quality of the routing process, i.e. avoid local minimum points and inefficient path lengths. In UMG, a new node in the network might decide at some stage to spread its gradient to create routes from other nodes toward itself. The node spreads its gradient either because it has taken a producer/consumer role in the network or because another node requests to contact the node. In the reminder of the document, a node spreading its gradient at any given time will be known as "gradient origin node" or "sink". Multiple sinks might exist in the UMG network which establish end-to-end communication. The rest of the nodes act as pure routers, i.e. relays of packets.

A node spreads (setup) its gradient by employing a reactive controlled flooding process which can be limited in scope by setting a maximum coverage distance in terms of hops, i.e. the "Gradient

Spread Phase". When flooding the network, the gradient origin node broadcasts a gradient message with its address and a descriptor; the descriptor contains the services/data which the node either provides or is interested in. This way, the gradient spreading process is also used as a service advertisement mechanism. Only the first gradient packet is forwarded from those received for a given gradient process from a gradient origin node. The use of the lowest hop count metric can easily be incorporated in UMG to create the shortest path; this is possible as UMG implements a backoff mechanism for the proper formation of the gradient, which delays the next broadcast based on the hop distance from the gradient origin node. However, UMG employs the fastest route metric by default as: i) it produces good results in terms of hops, despite not always being optimal, ii) it requires low complexity, and iii) it inherently considers the current congestion status of the nodes and the wireless medium. The key idea is that a node controls the number of packets to be broadcast such that the implosion problem is minimized. For this purpose, and in order to avoid loops, duplicate packets are detected. In the same line, the broadcast of a packet might get lost due to contention problems or inactivity of the node, e.g. due to duty cycling. In this case, the gradient might not get formed, for instance, if a packet does not reach a crucial node, i.e. a node which, by its unique position, serves as the gateway in between areas of the network. For this situation, UMG adds a delayed extra broadcast retransmission of the first gradient packet in order to increase the reliability of the gradient formation.

A node receiving the gradient packet creates or updates the entry for the gradient origin node in its routing table. The entry in the routing table contains the neighbour address from which the packet was received. The routing table is composed of one entry for every gradient origin node. An entry in the routing table for the gradient origin node indicates the next node address to reach the gradient origin node, the service descriptor of the gradient origin node and the distance in number of hops to the gradient origin node. The service descriptor is stored in the form of a Bloom filter which compresses descriptive information efficiently and can also be efficiently and quickly queried (see Section 4.4). The routing protocol provides functionality to search and compare descriptors. When combining descriptors with information on the distance to the gradient origin node, the UMG offers powerful information to higher layers for the decision making process.

Once initial gradients are established, communication with the gradient origin node can be started from any other participating node in the gradient. This is achieved by descending the gradient in a reliable manner, i.e. the "Gradient Descent Phase". Every data packet descending the gradient is acknowledged at each hop, either with an explicit packet or by snooping the next hop transmission. If after a maximum number of trials, the packet has not been acknowledged, a local repair mechanism is launched which looks for a valid candidate to keep on descending the gradient towards the gradient origin node, i.e. the "Local Repair Phase". If the local repair fails, as it does not find a suitable neighbour, the entry in the routing table is provisionally disabled. The next end-to-end message will launch another local repair process without considering nodes with disabled entries for the gradient. This mechanism can be seen as a special one hop backtracking in which the failing node is not selected in the next end-to-end gradient descending process.

UMG provides end-to-end acknowledgements to enhance the reliability of the data delivery process. End-to-end acknowledgement packets are issued by the gradient origin node and climb the gradient towards the originator of the communication (see "Acknowledgement Phase" in Section 4.5.3). For this purpose, UMG implements a short time-out cache structure which stores key information from received and sent packets in order to avoid cycles and identify whether the node forwarded a previous data packet (see Section 4.5.4.1). According to this, the acknowledgement packet is only broadcast by those nodes which previously participated in the sending of the data packet. This way, the acknowledgement packet climbs the gradient on a hop-by-hop basis. The entry for a particular data packet in the cache structure becomes invalid when an acknowledgement packet is successfully forwarded or when it times out. The protocol is aware of the successful delivery of the acknowledgement packet by snooping the broadcast of the next node on the way up to the originator. If an acknowledgement packet can not be delivered to the next node, and bidirectionality is therefore not achieved, the option of sending the acknowledgement packet via the gradient of the originator of the communication is enabled; for this purpose, the originator of the communication must have spread its gradient. In the worst case scenario, the originator node has the option to spread its gradient in requesting mode, i.e. requesting the gradient origin node to spread its gradient. A diagram providing an example of the operation of the main phases of UMG is depicted in Figure 4.1.

The UMG routing protocol has been designed to support dynamic changing topologies, where gradient origin nodes leave their current neighbourhoods to become part of other areas. UMG tends to minimize communication when there is no data communication activity in the neighbourhood, avoiding the use of periodic hello messages. The idea consists in letting the network update itself via the use of opportunistic communication, either received or snooped. Areas with frequent data communication activity will make use of overheard packets to detect when a node is changing its position with respect to its neighbours, i.e relative mobility. On the other hand, areas with no communication activity will not be updated even if mobility occurs. These areas where communication occurs infrequently might require on-demand gradient updates if the topology changes. However, they will require low maintenance in terms of communication and will exhibit a silent behaviour.

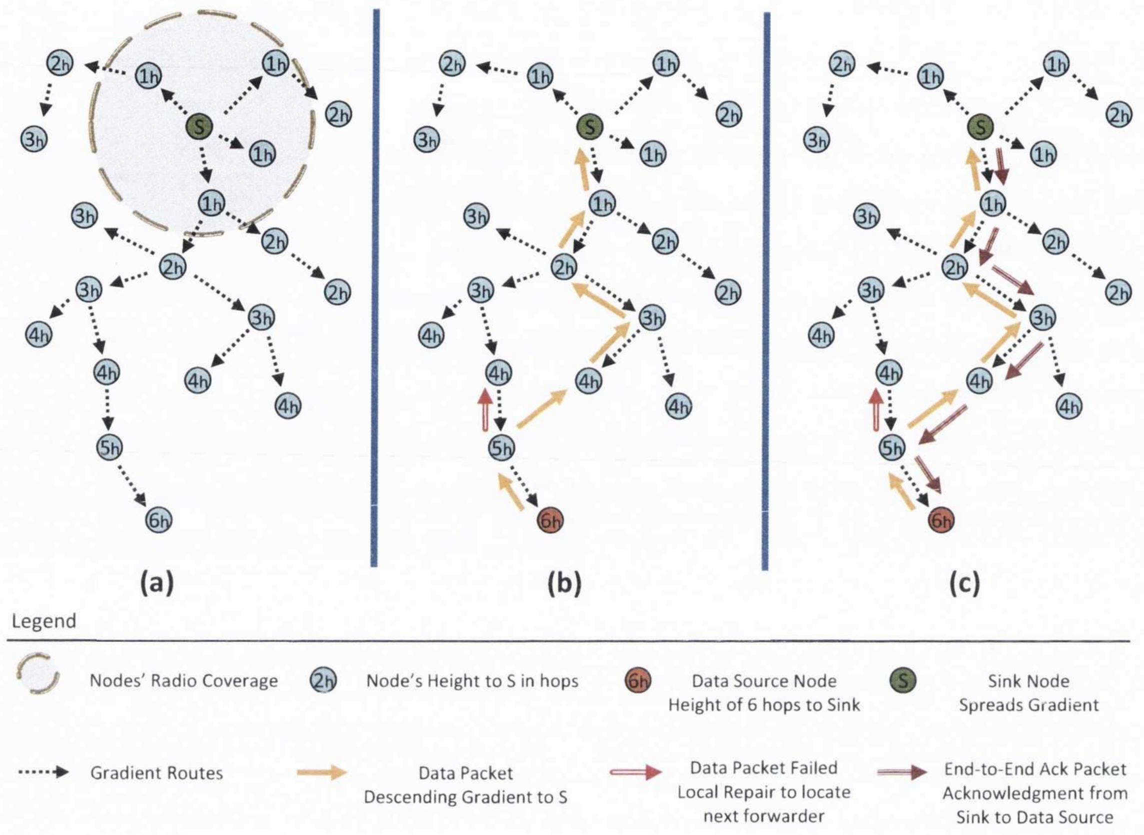

(a)
(b)
(c)
Legend
Nodes' Radio Coverage
Node's Height to S in hops
Data Source Node Height of 6 hops to Sink
Sink Node Spreads Gradient
Gradient Routes
Data Packet Descending Gradient to S
Data Packet Failed Local Repair to locate next forwarder
End-to-End Ack Packet Acknowledgment from Sink to Data Source

Fig. 4.1: UMG - Phases. Subfigure (a): Gradient Spread Phase. Subfigure (b): Gradient Descent Phase with Local Repair Phase. Subfigure (c): Acknowledgement Phase.

This scheme suits applications for wireless sensor networks where some areas might be silent for long periods of time and topologies are not subject to a high degree of variation.

In order to estimate whether a node has changed neighbourhood, the UMG routing protocol employs a probabilistic mobility assessment mechanism which is opportunistically fed with all packets received or snooped (see Section 4.6). The address of the sender of all the overheard packets is cached in memory efficient structures, Bloom filters (see Section 4.4.1). By employing a structure of temporal shifting Bloom filters, each filter stores the address of the neighbour nodes for a period of time before the next Bloom filter starts to be filled out. Periodically, a probabilistic estimation mechanism, employing a predefined table model, utilises the information stored in the Bloom filters to determine if the node is changing its position. This process compares previous states of temporal Bloom filters against a master Bloom filter. The master Bloom filter contains reliable information of the core neighbourhood obtained with a deterministic query process in a past period of time. This way, previous states of mobility can be recursively computed to decide on the probability of the node changing neighbourhood.

When mobility is detected, UMG employs explicit local update messages to confirm the neighbourhood mobility status, to find out the number of new and old neighbours, and to populate its master Bloom filter. All the entries in the routing table are disabled as they will be of no routing use in the new neighbourhood, however they are not deleted as they are employed for service lookup. In this case, the node will update its routing table opportunistically based on the communication being received. If the node is a gradient origin node, it needs to create routes from its old neighbourhood to its new position. To do so, the gradient is spread with a limited scope in terms of hops, depending on how far the node is from the farthest old neighbour. This mechanism reconfigures only nodes within the area of the scope such that they forward data packets towards the new position. An expensive global flooding will not be necessary due to the fact that the scope of the flooding only has to reach the old neighbours for the gradients to keep converging into the node.

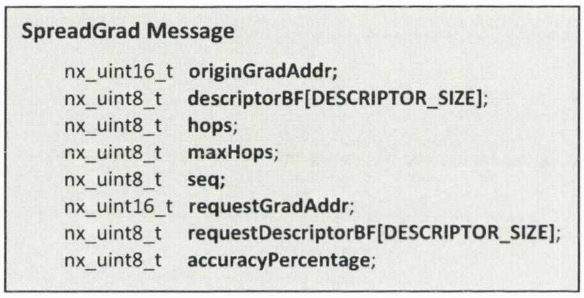
In the next sections of this chapter, the different phases and mechanisms of UMG are described.

4.3 Gradient Spread Phase

The gradient spread phase takes place when a node needs to inform other nodes in the network about both its existence and the type of service provided. This phase employs a controlled flooding process to establish a gradient towards the node, i.e. the gradient origin node, such that other nodes have the possibility to route traffic towards it. If a node does not want to be contacted or it has no service to offer, the node might choose not to spread its gradient. Instead, nodes might work simply as routers without providing or consuming data. A node can spread its gradient at any given point in time. The gradient spread phase can be launched at the beginning of the mote's life, which for most of the nodes will be the beginning of the network's life. Further gradient spread updates, which might affect only a limited scope of the network, would take place when i) a node changes its services, ii) when reliable mechanisms demand a gradient update, or iii) when a change in the neighbourhood of the node forces the gradient to be reconfigured. In addition, a node A has the option of requesting another node B to spread its gradient such that bidirectional data communication can occur between them; this is achieved while A is spreading its gradient.

4.3.1 Gradient Formation

In the initial network setup, i.e. at the beginning of the network's life, nodes wishing to advertise their gradients wait a random time to minimize collisions with ongoing gradient processes. This process gives time to all the nodes in the network to boot up and start their radio communication system, such that they can participate in the gradient setup process. The option of using a gradient message as a control message to start the formation of gradients once all nodes have booted up is also available. In this case nodes stay on a waiting state until the first gradient packet is received from a previously agreed node or with a reserved service description encapsulated; at this time nodes in need to advertise their gradients wait for a random bounded time. While this is a proactive process where gradients are created beforehand, nodes might also setup their gradients in a reactive manner when data communication is needed.

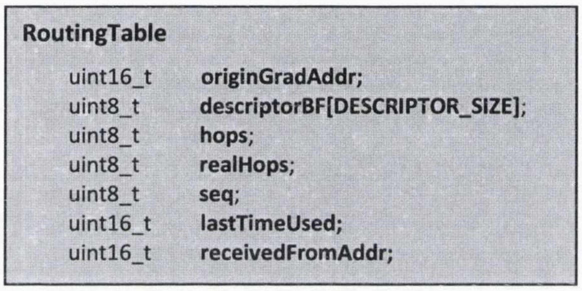


```
SpreadGrad Message
    nx_uint16_t  originGradAddr;
    nx_uint8_t   descriptorBF[DESCRIPTOR_SIZE];
    nx_uint8_t   hops;
    nx_uint8_t   maxHops;
    nx_uint8_t   seq;
    nx_uint16_t  requestGradAddr;
    nx_uint8_t   requestDescriptorBF[DESCRIPTOR_SIZE];
    nx_uint8_t   accuracyPercentage;
```

Fig. 4.2: UMG - SpreadGrad Message Structure

The spreading of the gradient consists in a controlled flooding process where the first unique packet received is sent twice for a "good" formation of the gradient. The message structure employed to spread the gradient is known as "SpreadGrad" message (see Figure 4.2). The address of the gradient origin node spreading the gradient is indicated in the SpreadGrad message as the "originGradAddr" field. Each node employs the SpreadGrad message to advertise its services. For this purpose, a memory efficient structure containing the service descriptor is carried in the SpreadGrad message as "descriptorBF" (see Section 4.4 for further explanation). In order to distinguish between different gradient updates from the same gradient origin node, a sequence control value is defined in the SpreadGrad message as "seq". The combination of "originGradAddr" and "seq" makes the SpreadGrad message uniquely identifiable. The sequence value is incremented each time a gradient origin node starts a gradient formation. The SpreadGrad message also contains a value which carries the height of the sender node with respect to the gradient origin node; by default UMG employs the hop distance as the height. The gradient setup limits the scope of the spreading process in terms of hop distance; this is indicated by the "maxHops" field in the SpreadGrad message.

The protocol also provides the option for a gradient origin node to request other node(s) to setup their gradient. By encapsulating requesting information in the SpreadGrad message, the gradient origin node can make requests in two ways: i) by addressing a particular node indicating the address of the node in the SpreadGrad message as the "requestGradAddr', or ii) by addressing a set of nodes whose service descriptor matches the descriptor "requestDescriptorBF" of the SpreadGrad message with an accuracy equals or higher than "accuracyPercentage". Upon reception of the SpreadGrad message, matching nodes hold the power to decide whether to spread the gradient.



```
RoutingTable
      uint16_t     originGradAddr;
      uint8_t      descriptorBF[DESCRIPTOR_SIZE];
      uint8_t      hops;
      uint8_t      realHops;
      uint8_t      seq;
      uint16_t     lastTimeUsed;
      uint16_t     receivedFromAddr;
```

Fig. 4.3: UMG - Routing Table

In UMG, every node receiving a SpreadGrad message only takes into account the first packet received for a given gradient origin node and sequence "seq". The first gradient message received by a node populates or updates the Routing Table (see Figure 4.3) with the address of the gradient origin node as the key index (see "originGradAddr" in the SpreadGrad message and in the Routing Table). In addition, the address of the node from which the message has been received is stored in the Routing Table as the next node in the gradient to reach the gradient origin node (see "receivedFromAddr" in the Routing Table). Other values from the SpreadGrad message are also stored in the Routing Table such as the hop distance ("hops"), the sequence value ("seq"), the maximum hop distance ("maxHops") and the service descriptor ("decriptorBF"). In addition, a local counter at each node is employed to timestamp the creation or update of each Routing Table entry; the "lastTimeUsed" field in the Routing Table stores the current local counter value for the associated "originGradAddr" entry. The local counter is incremented every time the node receives a gradient update or a data packet is relayed through it. This provides an indication of the freshness of information for each gradient entry, as compared to the rest of the entries, in the event of the Routing Table gets full and a "originGradAddr" entry needs to be replaced by a new one. The "lastTimeUsed" field is also employed as a mechanism to enable or disable the Routing Table entry for a gradient. A node might be interested in canceling its routing activity for any given gradient by temporarily disabling the entry ("lastTimeUsed" equals to 0). For instance, the node might require to save battery, ameliorate the congestion, or simply indicate that the Routing Table entry is not valid as a result of mobility or unresponsive behaviour. Finally, the "hops" field is populated when the gradient is created and acts as a control variable together with the "seq" field. However, further gradient updates, e.g. when mobility is detected (see Section 4.6), or the launching of local repair processes (see Section 4.5.2) might change the real end-to-end hop distance when descending the gradient. To account for the real end-to-end hop distance to the "originGradAddr", the "realHops" field has been added to the Routing Table. The update of this field occurs in the Acknowledgement Phase (see Section 4.5.3).

On the formation of the gradient, routing protocols employ different mechanisms to create efficient paths in terms of hops. Many implementations employ an evaluation function which decides from a set of received packets which one is to be forwarded in the creation of the gradient. This function selects the most suitable packet based on metrics like hop count, battery level, link quality or signal strength. This process consumes memory and computational resources while demanding an extra backoff time to wait for a set of packets to be received. In UMG, the first packet arriving from the flooding process is forwarded and used for update purposes. Every node only forwards the first unique packet received for a gradient origin node. The rest of the SpreadGrad messages are discarded; this contributes to avoid cycles, decrease the communication activity and thus reduce the network contention. This is a simple mechanism which aims to reduce the complexity in the gradient process. The assumption is that creating and maintaining optimal gradients, in any terms, incurs a high use of resources at relatively low benefit in dynamic networks such as WSN. Instead, UMG exploits the creation of well-formed (sub-optimal) gradients by increasing the likelihood of all the neighbours receiving the SpreadGrad message. In this regard, the successful reception of the message by all neighbours depend on factors such as the density and topology of the network, the contention in the wireless medium, the congestion in the queues, and the nodes' sleeping policies. These factors produce packet loss and might contribute to the inefficient formation of the gradient. While the cost of not forming efficient gradients can be translated into an increase in the number of hops of a route, the true risk comes when posterior gradient updates only reach a subset of the nodes. In this scenario, different gradient updates for the same gradient origin node need to coexist such that loops and local minima points are not produced. As an observation, gradient routing protocols which unicast data packets following the gradient path, such as UMG, are more susceptible to this problem than protocols which do not set paths and rely on nodes with lower height values to broadcast the packet.

To solve the above issues, avoid loops and add reliability and efficiency to the gradient path formation, two spaced retransmissions of the same packet are employed to increase the likelihood of all neighbours receiving the packet. This acts as a countermeasure against packet loss due to network contention or the short unavailability of a node to receive a packet. In addition, a back-off time is applied to the first and second retransmission of the broadcast packet. The back-off time is computed as a function of the hop distance to the gradient origin node, such that the delay in the retransmission increases linearly with respect to the distance from the gradient origin node. This mechanism introduces a delay in the gradient setup at the benefit of a progressive formation of the gradient in terms of distance from the origin. This can be seen as a wait and forward controlled mechanism which expands the gradient in a step-based process where each hop iteration increases the scope of the discovery progressively in a semi-uniform concentric manner with centre at the gradient origin node.

The backoff-based gradient spread mechanism works as follows: When a new unique SpreadGrad message is received for the first time at a node, an initial backoff time is calculated as:

$$\text{BackOff\_Initial} = (\text{hops} * \text{DelayPerHop}) + \text{RDDelay} \tag{4.1}$$

where "hops" is the number of hops received in the SpreadGrad message, "DelayPerHop" is a constant which establishes the incurred delay in milliseconds per hop unit, and "RDDelay" is a randomly generated delay in the range of [0, Delay]; "Delay" is a constant time in milliseconds. The higher the "Delay" upper limit, the higher the likelihood of an increase in the time to create the gradient and the lower the likelihood of colliding with other packets in each neighbourhood. By the same token, the higher the "DelayPerHop" value, the higher the latency in the formation of the gradient. However, the "DelayPerHop" value affects the probability of packet collision and the proper formation of the gradient. Experimentally, the probability of collision increases when the "DelayPerHop" is set to a value equal or less than 3 ms, while the probability of collision remains bounded between proximate values when the value is greater than 5 ms. Nevertheless, the probability of collision is highly dependent on the number of neighbours and the ongoing communication in the neighbourhood; these factors can be constantly changing. This is also one of the main reasons why the delay and retransmissions of the SpreadGrad message are required for the proper formation of the gradient.

Once the backoff time elapses, the packet is broadcast. A second shorter backoff time, which does not depend on the hop count, is applied to delay the retransmission of the SpreadGrad message as:

$$\text{BackOff\_Retransmission} = \text{RDDelay} + \text{MinDelay} \tag{4.2}$$

where "MinDelay" is a constant which guarantees a minimum backoff time in the event of the "RDDelay" random value is zero. The BackOff_Retransmission is employed as a mechanism to increase the likelihood of all the neighbours receiving the packet. It is set to a lower value than the BackOff_Initial such that the second broadcast of the packet occurs before a packet with a higher hop value is broadcast.

While this process improves the gradient formation, it also adds a cumulative linear growth to the latency of the gradient setup, which might affect the QoS requirements of high layers. The next equation calculates the cumulative latency in the worst case scenario (where the first SpreadGrad message is lost) as:

$$\text{LatencyWC} = \sum_{\text{hops}=0}^{\text{n}} (\text{hops} * \text{DelayPerHop} + \text{Delay} + \text{Delay} + \text{MinDelay}) \tag{4.3}$$

solving the series in Equation 4.3, the worst case latency can be calculated as:

$$\text{LatencyWC} = \frac{1}{2}(\text{hops}+1)((\text{hops} * \text{DelayPerHop}) + ((2 * \text{Delay} + \text{MinDelay}) * 2)) \tag{4.4}$$

By the same token, the best case scenario, where the first packet is always received and the random value of the "RDDelay" is always 0, can be calculated as:

$$\text{LatencyBC} = \sum_{\text{hops}=0}^{\text{n}} (\text{hops} * \text{DelayPerHop}) \tag{4.5}$$

solving the series in Equation 4.5, the best case latency can be calculated as:

$$\text{LatencyBC} = \frac{1}{2}(\text{hops}+1)((\text{hops} * \text{DelayPerHop}) \tag{4.6}$$

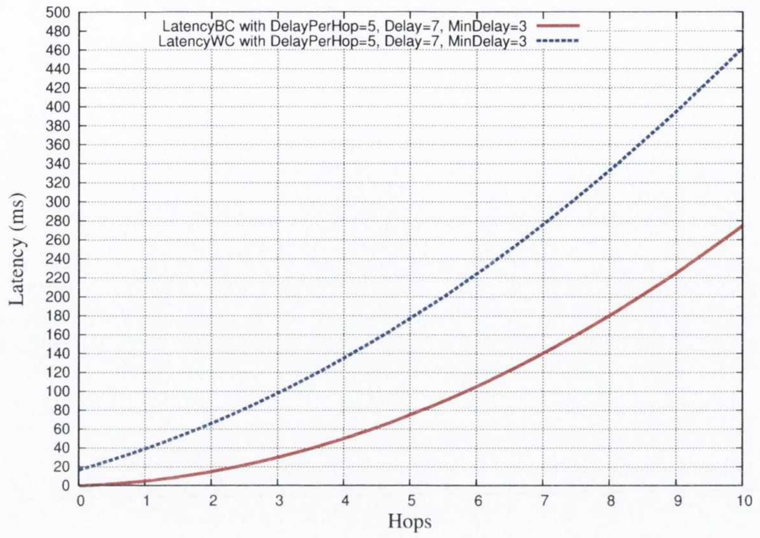


Fig. 4.4: UMG - Gradient Formation Latencies (WC=Worst Case Scenario see Equation 4.4) (BC=Best Case Scenario see Equation 4.6)

By default, UMG employs the next combination of values for the time delays in milliseconds: "DelayPerHop = 5 ms", "Delay = 7 ms", and "MinDelay = 3 ms". These parameters have been empirically selected as they have proven to establish efficient gradients in terms of hop distance at low latencies for a variety of scenarios with different traffic loads. Following these values, the latency for the worst and best case scenarios is shown in Figure 4.4 for a maximum hop distance of 10 hops. Note that the latencies correspond to the delay in the formation of a gradient, this does not take into account the delay introduced when other packets are in the queue, the time to send a broadcast packet ($\sim$ 11 ms in TinyOS) and each packet MAC backoff time (300 μs - 9.8 ms in TinyOS).

Figure 4.5 plots the heights of each node in the network with respect to a selected gradient origin node at the initial state of the network when multiple nodes are simultaneously spreading the gradient. The gradient origin node is placed at position (100,100). Its gradient is spread progressively according to the hop distance over a 20 by 20 nodes grid. The semi-uniform concentric circles formed can be seen in the contour map, while the smoothness of the gradient shape is identified by the dots which represent the height in terms of hops of every node at each location. In addition, the figure gives an idea of the transmission range of the nodes and the degree of asymmetry in their links.

The gradient formation process has a two-fold functionality. Firstly, it creates the gradient paths and advertises the service. Secondly, the flooding process updates nodes on the current state of their neighbourhood, thus increasing the accuracy of the mobility assessment mechanism (see Section 4.6).

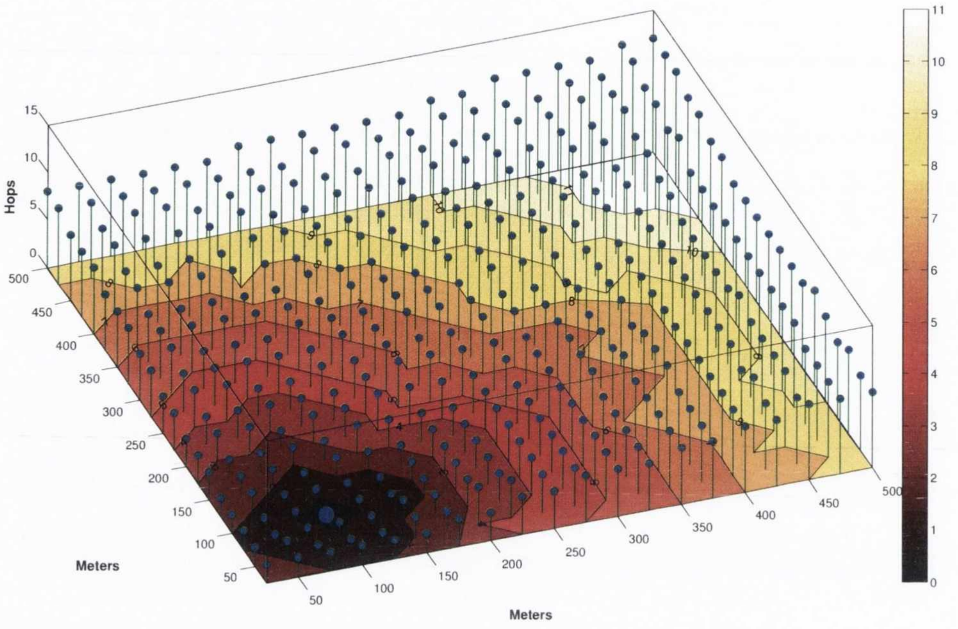


Fig. 4.5: UMG - Gradient Formation Contour Map Example (400 Nodes). Gradient Origin Node at position (100,100). Colour map corresponds to the distance in hops, i.e. height, of each node from the gradient origin node.

Additionally, the number of messages sent and received in the gradient formation process can be estimated beforehand. The number of messages broadcast in a gradient formation process depends on the scope of the flooding and the number of nodes in the scope. However, knowing the number of participant nodes, the maximum number of messages broadcast will be 2 per node, which gives an idea of the communication impact of this process. The number of broadcast packets received at each node for a particular gradient formation depends on the network density, the transmission power range and the receiver sensitivity of the transceiver. In other words, it depends on the average number of neighbours for each node as this varies dynamically due to the instability of the wireless links. In addition, the scalability factor in UMG is directly proportional to the number of routing table entries which a node can accommodate. In other words, how many consumer/producer nodes are going to employ a given node to perform routing for their gradients. In this regard, a given node might only be responsible for a set of gradients within a limited scope. The rationale behind this is that, in large sensor networks, nodes would limit communication within a certain hop distance for efficient routing; after a certain number of hops, routing might be inefficient and infeasible.

Finally, a new node arriving to an up and running network where gradients are already formed is of no use to the network until it learns from the network on how to act as a router. The cost of re-calculating the gradients of a new node's neighbourhood to make it work as a functional router for all the existing gradients is expensive in terms of communication. Therefore, in UMG a new node will progressively operate as a router when its routing table is updated with new gradient updates. Gradient setups can be triggered due to an unresponsive end-to-end connectivity or when mobility detection demands an update. On the other hand, if a new node manifests interest in being connected to a particular node or set of nodes, the UMG routing protocol provides query mechanisms to prompt specific nodes for gradient formations.

4.3.2 Implementation and Structures

A single component manages the gradient setup process and the Routing Table, while exposing an interface to control the gradient process, receive events occurring in the component, and query the Routing Table. A single-purpose packet for spreading the gradient has been created, the SpreadGrad message (see Figure 4.2). The SpreadGrad message acts as a service advertiser as it carries the descriptor for the services provided by the gradient origin node, i.e. the "descriptorBF" field. The size of the descriptor in UMG, i.e. the DESCRIPTOR_SIZE, has been set to 4 bytes. When the SpreadGrad message serves as a query mechanism for nodes matching the service, i.e. the "requestDescriptorBF" field, a comparison against the local descriptorBF at each node is performed (see Section 4.4 for details on the comparison of the Bloom filter structures).

The Routing Table structure defines the fields for each unique gradient origin node entry, i.e. "originGradAddr" (see Figure 4.3). Thus, an array of Routing Table structures is created which contains an entry for each gradient for which the node is a router. The size of the Routing Table array depends on the expected number of nodes advertising their gradients. This has a cost in terms of RAM memory - currently 13 bytes are allocated for each entry - and therefore UMG lets the user application control this parameter. However, a mechanism to replace infrequently used entries is in place when the number of gradient advertising nodes exceeds the limit; this was mentioned in the previous section which is based on the "lastTimeUsed" field in the Routing Table as an indicator of the activity of the entry. A global variable to control the roll over of the "lastTimeUsed" counter is in place; the 0 value is reserved as an indicator of the disable status of the entry in the Routing Table.

Moreover, the component implements a dedicated queue to manage SpreadGrad messages. Packets are processed following a FIFO policy. Each packet is timestamped when added to the queue in order to account for the time elapsed in the queue when calculating the initial backoff time (see Equation 4.1). The queue size depends on an estimation of the number of concurrent gradient setup processes and the backoff time applied per packet. For all of the simulations and testbed experiments presented in this thesis, a queue size of 10 packets has proved to be successful. Gradients of up to 10 hops have been achieved, with 60 nodes spreading their gradients. A random time over an interval of 5 seconds is calculated by each node to initially start their gradient. In addition, a minimum time of 3 seconds has been set for a node to wait for its gradient to be spread before updating the gradient again.

4.4 Ubiquitous Lookup

Many applications in WSN require some form of node description in terms of data or service provided. Protocols have been designed to specifically achieve such service advertisement and discovery functionality. Most of them require a high number of messages to achieve such functionality which, in WSN, results in a expensive resource consuming activity that can not be afforded. The idea of introducing a service advertisement and discovery mechanism at the routing layer has been implemented in UMG. The process of service advertisement/discovery is implemented as part of the routing process by making use of every SpreadGrad message traversing the network. When creating the gradient, the SpreadGrad message carries a descriptor of the service/data provided by the gradient origin node (see "descriptorBF" in Figure 4.2). The metadata set, M, is a collection of human readable words in a dictionary which is common to the whole network. Items in M are employed to define the type of service or the behaviour of a node in terms of data acquisition, i.e. interest. The "descriptorBF" structure contains a subset of the metadata, D, such that $D \subseteq M$, and D defines the service of a gradient origin node. The idea lays in the possibility of extending the universe of M while not affecting the "descriptorBF" of the nodes in the network. To achieve this, the "descriptorBF" field operates as a Bloom filter [88], i.e. an array of bits set to 1 and 0, which store the membership of an item in the universe of M, rather than storing the item itself. Due to the fact that storing every item (word) would cost too many bytes, and associating one bit of a bitvector to each item restricts the scope of M, Bloom filters were selected as a compressing structure to store and advertise the metadata/services.

On reception of a SpreadGrad message, a node updates its routing table with the "descriptorBF" for its gradient origin node entry (see Figure 4.3). This mechanism allows for each node to have a descriptive map of the type of services offered by those nodes for which the local node is a router, along with the distance in hops to reach them. This is a powerful tool for the decision making process of applications working on top of the routing protocol. UMG provides an interface to the upper layers to create and query Bloom filters. An application might contain a set of keywords, i.e. a dictionary, which defines the data offered/contained/stored by the node. With this approach, an implicit ubiquitous lookup system is available which provides hints in the searching process.

In order to populate the "descriptorBF" structure before spreading the gradient, a set of keywords from M are selected which describes the behaviour of the node. A set of hash functions are applied to each keyword to produce a set of positions in the array of bits of the Bloom filter which are to be set to 1. Once created, Bloom filter structures are easy and fast to query and can be compared using a "XOR" bitwise operation. This provides a powerful tool in areas like distributed data fusion, discovery service, distributed storage or swarm decision making. Moreover, the Bloom filter descriptor can be seen as a profile descriptor which describes the node and thus could serve as a mechanism to create overlay virtual communities of sensors.

4.4.1 Bloom Filter Functionality

A Bloom filter [88], is a probability-based compressing structure which identifies whether a data item is a member of the filter or not, rather than storing the data item itself. The Bloom filter structure is represented as an array of bits. By switching bits from 0 to 1 in a reduced set of positions in the bitvector, an element is declared as a member of the filter. Every combination of positions represents a data item stored. In order to calculate the combination of positions which represent a particular data item, hash functions are employed. Each hash function is applied to the data item producing an integer number which is trimmed/scaled to the number of bits in the Bloom filter - thereby generating a position in the bitvector. Applying X hash functions to the same data item produces Y positions set to 1 in the bitvector where $X \geq Y$. The collection of hash functions applied over a data item will always produce the same combination of positions for that data item. By the same token, a data item can be hashed to check whether all its positions in the Bloom filter are set to 1, i.e. to check membership.

The Bloom filter structure has a drawback. The higher the number of data items hashed in the Bloom filter, the higher the probability of obtaining false positives. A false positive occurs when a data item has not been hashed in the Bloom filter, but the bit positions which correspond to the data item are set to 1 as a combination of positions from other data items previously hashed. The selection of the number of hash functions, together with the number of bits of the Bloom filter and the number of elements inserted in the Bloom filter, establish the probability of getting false positives.

On the other hand, the Bloom filter guarantees that there will be no false negatives, i.e. if there is not a combination of positions set to 1 which satisfies the hashed data item, then it is guaranteed that the data item was not inserted in the Bloom filter. Bloom filters can easily be compared with a "XOR" bitwise operation and merged with an "OR" bitwise operation.

The probability for a false positive error can be calculated with the next equation:

$$E_{fp} = \left(1 - (1 - 1/m)^{kn}\right)^k \tag{4.7}$$

where "m" is the number of bits in the Bloom filter, "n" the number of elements inserted and "k" the number of hash functions [216]. The error E_{fp} can be minimized for the number of hash functions according to the next equation [216]:

$$k = (m/n) \ln 2 \tag{4.8}$$

For instance, for a bit array 10 times larger than the number of entries, the probability of a false positive is 1.2% for k=4 hash functions, and 0.9% for the optimum case of k=5 hash functions. If the data set is known "a priori" then the Bloom filter can be designed to avoid false positives. Figure 4.6 depicts the activity of inserting an element into a Bloom filter, and also includes a practical example showing how a set of keywords, which describe a sensor device, are inserted in the structure.

4.4.2 Implementation

The Bloom filter has been implemented as a generic component which needs to be parameterised with the number of hash functions and the size of the filter in bits. Up to ten common implementations of hash functions have been selected and their computational performance tested in real sensor devices and are available to be applied sequentially. Interfaces are provided for the operation of the Bloom filter, for instance: data item insertion and membership checking. Other functionalities are provided like Bloom filter reset and union.

A function to compare the similarity of two Bloom filter structures, according to a percentage value of accuracy, is also implemented. The accuracy percentage is obtained as the fraction of the number of matching positions set to 1 in both Bloom filters divided by the number of positions set to 1 in the Bloom filter with the highest number of positions set to 1. This functionality is employed by UMG to query the routing table of other nodes descriptors for the discovery of services. For instance, the local descriptor is compared against the query descriptor "requestDescriptorBF" of the SpreadGrad message (see Figure 4.2); if the accuracy percentage of similarity is greater or equal than

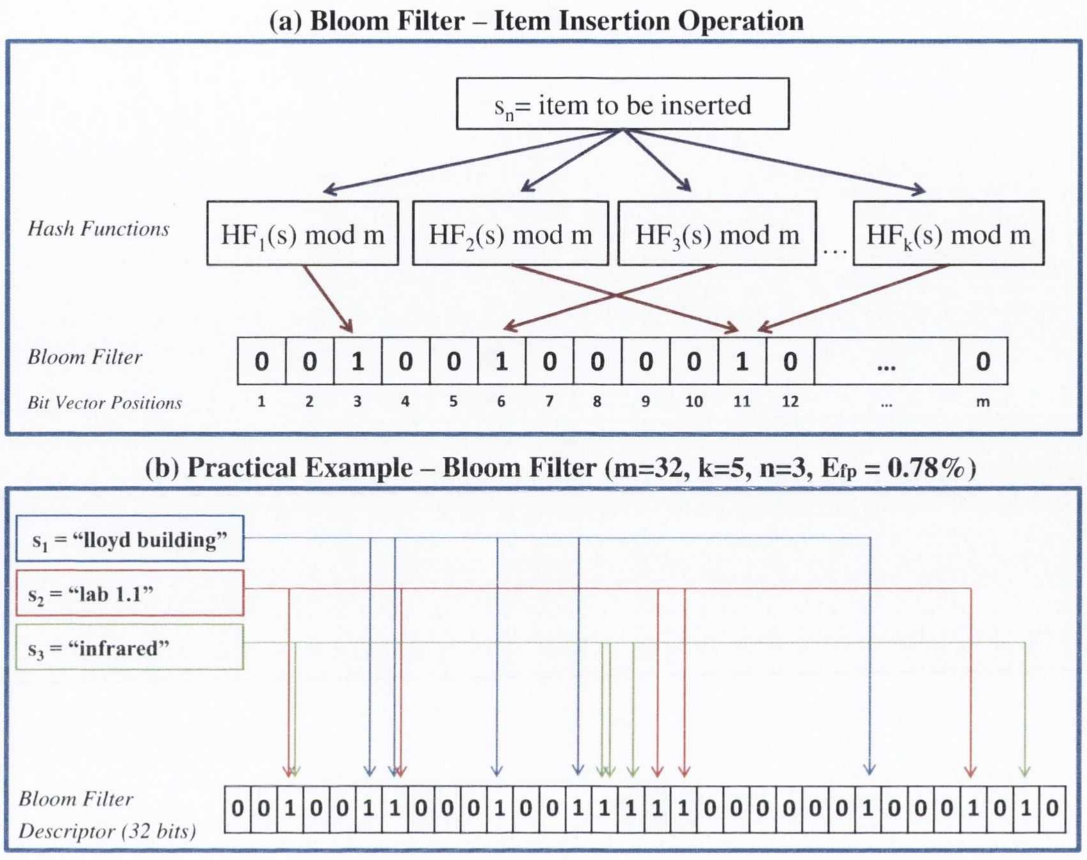


Fig. 4.6: UMG - Bloom Filter Operation. Subfigure (a):

the "accuracyPercentage" of the SpreadGrad then the node should spread its gradient. Moreover, this mechanism provides support for applications which require searching or aggregation of different types of data.

In UMG, the default parameters employed to configure the descriptor Bloom filters are: size equals to 32 bits, i.e. DESCRIPTOR_SIZE, and number of hash functions equals to 5. If the application contains a set of predefined keywords M, i.e. a metadata dictionary, the Bloom filter can be firstly tested to reduce the probability of obtaining false negatives on selected subsets.

4.5 Data Transport Phases

This section describes the design and implementation of the three phases employed in the transportation of the data. These phases are launched when the gradient for the destination, and even the source, node has been spread. The three data transport phases are: 1) Gradient Descent Phase, 2) Local Repair Phase, and 3) Acknowledgement Phase.

4.5.1 Gradient Descent Phase

This phase is executed when a node needs to communicate with another node which has already spread its gradient. The Routing Table is consulted for the entry corresponding to the gradient origin node address. The Routing Table can also be searched by descriptors, "descriptorBF", searching for nodes which can satisfy the requested service (see Figure 4.3). If the entry in the Routing Table can not be found and the local node demands to communicate with a specific node, then the local node needs to setup its gradient with the option of requesting a particular address, or a set of nodes matching the "requestDescriptorBF", such that the requested node/s spread the gradient (see Section 4.3.2). When there is a Routing Table entry for the destination node which is marked as enabled, the local node prepares the "DataGrad" message (see Figure 4.7). Data from higher layers is encapsulated in the "data" field to be transported. A new sequence value is generated by the originator node as "seq" in the "DataGrad" message. Together with the originator node address encapsulated in the "originAddr" field and the destination node address as the "originGradAddr", the packet is uniquely identifiable. The packet is sent to the address indicated by the "receivedNodeAddr" field in the Routing Table. Intermediate nodes receiving the "DataGrad" message keep on forwarding the data packet to the address of the node in the "receivedNodeAddr" field of their Routing Tables for the corresponding gradient origin node entry, i.e. the "originGradAddr" field. The number of hops is increased in the "DataGrad" message ("hops" field) with every new node visited. The "hops" field is also used as an incremental hop counter when climbing the gradient towards the originator of the communication in the Acknowledgement Phase (see Section 4.5.3). This way, the "hops" field is employed to update the "realHops" field of the Routing Table in the intermediate nodes, which indicates the most recent real hop distance.

4.5.1.1 Achieving Reliability when Descending the Gradient

The wireless medium is prone to errors in the packet transmission due to collisions, noise, or environmental condition changes which can last for an arbitrary length of time. Communication between

```
DataGrad Message
    nx_uint16_t  originGradAddr;
    nx_uint16_t  originAddr;
    nx_uint8_t   hops;
    nx_uint8_t   seq;
    nx_uint8_t   data[0];
```

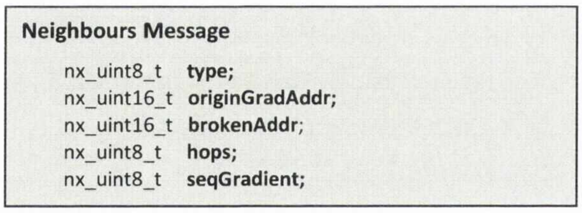
Fig. 4.7: UMG - DataGrad Message Structure

neighbours might not be established for short periods of time due to congestion at nodes, the node being in a sleeping mode, or the medium being contended. The connection can also be blocked for a long period of time, for example an obstacle could be temporarily interrupting the communication or local mobility might be placing nodes too far apart for the link to be stable. Moreover, a link might be broken as the next node might have died or changed position. In any of these situations, reliability mechanisms are employed to achieve end-to-end communication in the path.

In this regard, UMG employs acknowledgement packets, timers, and retrials for the end-to-end reliability of the path. If an explicit end-to-end acknowledgement packet is not received by the originator of the communication after a predefined time, the packet is sent again. This procedure is repeated for a number of trials until the packet is acknowledged. The number of trials is subject to be changed and by default has been set to 2. When the maximum number of trials is reached, the option of employing the gradient creation phase, with or without the option of requesting for the destination node to spread its gradient, can be launched to create routes. To reduce the number of gradient formations, which are expensive in terms of communication, UMG enables the acknowledgement mode at the MAC layer. In this mode, a unicast packet issued by a node must be explicitly acknowledged by the next hop receiver. This mechanism allows UMG to implement a procedure for which a node waits for a period of time for an acknowledgement packet and resends the packet up to a maximum number of trials if the acknowledgement packet is not received. In addition, UMG employs snooping functionality to eavesdrop the data packet being sent by the next hop node as a backup mechanism to implicitly acknowledge the packet. Both mechanisms increase the reliability of the end-to-end packet delivery in situations where short time communication problems occur. Additionally, when the communication between two neighbour nodes repeatedly fails, the maximum limit of retrials will be reached and no acknowledgement will be received by the next node when descending the gradient. In this situation a local repair mechanism is started which finds the most appropriate neighbour node to keep on descending the gradient.

4.5.2 Local Repair Phase

The local repair mechanism is launched when i) the local node needs to send a message to a node for which there is no entry in the Routing Table or ii) the maximum number of trials is reached and still the data packet has not been acknowledged by the next neighbour in the gradient descending path. In this phase, the node waiting to receive the acknowledgement packet needs to look for an alternative neighbour node which can keep on relaying the packet towards the gradient origin node. For this purpose, a neighbourhood discovery process is launched by broadcasting a "Neighbours" message (see Figure 4.8). The message is typed as LOCAL_REPAIR and contains the address of the node which has failed to acknowledge the packet ("brokenAddr"), and the distance in number of hops ("hops") from the sender node to the gradient origin node ("originGradAddr") (see Figure 4.8). The "brokenAddr" and "hops" values are included to restrict the replies to those neighbours which can provide a valid route towards the "originGradAddr" node. A candidate node must not contain the address of the sender of the "Neighbours" message, nor the value of the "brokenAddr", in the "receivedFromAddr" field of the "originGradAddr" entry in its Routing Table. In other words, the next node to reach the "originGradAddr" in the candidate node's Routing Table can not be the node which failed to acknowledge the packet, nor the sender itself. In addition, the candidate node must be at least equally close, in terms of hops, to the "originGradAddr" node than the sender node is. These checking procedures avoid the formation of cycles which might incur a higher number of local repair mechanisms being launched, increasing the contention in the medium and decreasing the performance.

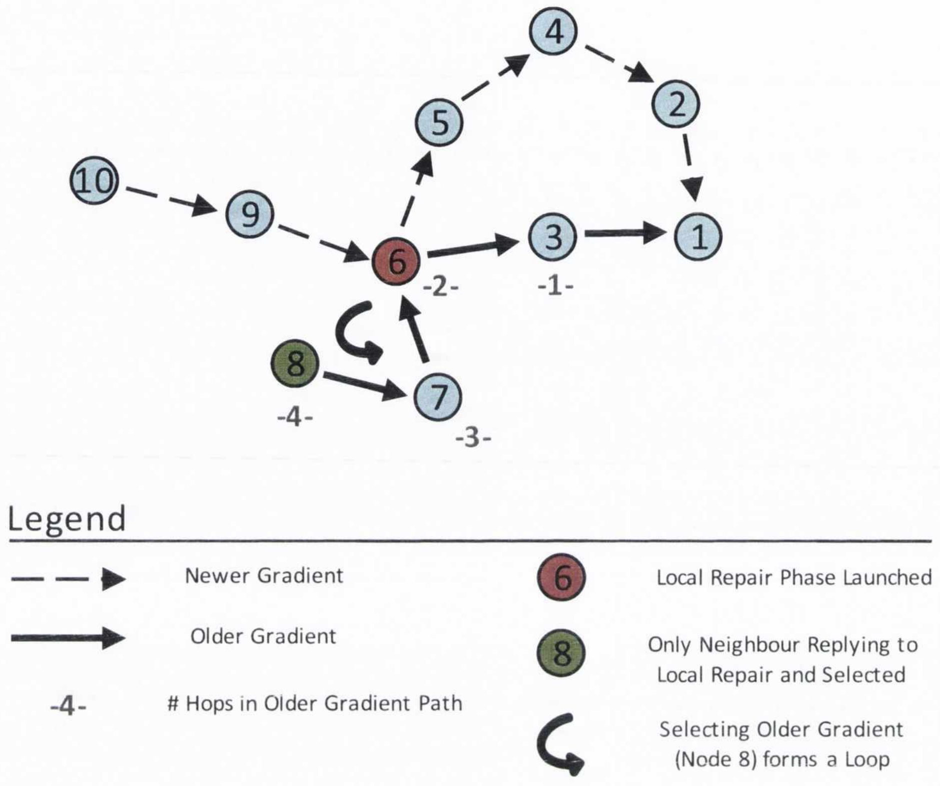

Neighbours Message

```
nx_uint8_t    type;
nx_uint16_t   originGradAddr;
nx_uint16_t   brokenAddr;
nx_uint8_t    hops;
nx_uint8_t    seqGradient;
```

Fig. 4.8: UMG - Neighbours Message Structure

However, when gradient updates for the same gradient origin node occur, it might be possible that some nodes will not get updated, thereby containing the address of the next hop ("received-FromAddr" field) in their Routing Tables corresponding to older gradient sequences "seq". In this situation, multiple gradient updates, with different sequences, need to coexist such that loops are not formed when performing local repairs. In Figure 4.9, this effect can be seen where a new gradient setup has been received which only updates some of the nodes in the network while the rest of the nodes still contain old gradient entries in their Routing Tables. In the diagram, for instance, node 10 issues a packet to descend the gradient towards node 1 via the new gradient. When descending the gradient, node 6 relays the packet to node 5 which does not acknowledge the packet after a number of trials. Thus, node 6, which is at 4 hops distance to node 1 via the new gradient, starts a local repair phase to discovery a valid next hop node. The only reply to the LOCAL_REPAIR packet sent by node 6 comes from node 8, which is also at 4 hops distance from node 1 but under the old gradient setup. Node 8 is a valid next hop node, as its next hop address is node 7 which does not pose any restriction. Nevertheless, node 7 relays packets to node 6 under the old gradient setup; this creates a loop. In order to avoid loops in this situation, UMG restricts a neighbour node from being a valid candidate if the sequence value "seq" of the entry "originGradAddr" in its Routing Table is older than that of the sequence of the node starting the local repair phase. The sequence of the initiator of the local repair phase is transported in the "Neighbours" message in the field "seqGradient" when the packet is typed as LOCAL_REPAIR. Nodes receiving the LOCAL_REPAIR message reply if their sequence for the "originGradAddr" is newer than the sequence received in the packet ("seqGradient"). Since sequence values roll over when reaching the maximum value of 2^8, a way to identify the new sequence when compared to others is required. For this purpose, the sequence counter is considered as a cycled structure where the newest sequence value is that with the highest distance to the other sequence value being compared against - the rationale behind this is that the probability of a small number of gradient setups updating all the nodes in the scope is higher than the probability of a node not getting updated when a larger number of gradient setups occur. Loops are not formed when packets progress from a node belonging to an old gradient setup to a node involved in a newer gradient formation. Thus, when selecting a node to repair a gradient, newer sequences will overlap older sequences even if the hop distance is higher. In addition to this, and as a precaution measure against man-in-the-middle attacks, UMG also implements a mechanism to detect cycles (see Section 4.5.3).

After a LOCAL_REPAIR message is sent, valid candidate neighbours reply with a "Neighbours" message with the "type" field set to REPLY_LOCAL_REPAIR. The latter message contains the number of hops to reach the "originGradAddr" node in the field "hops". From the list of candidates, nodes with the newest sequences are considered. From them, the closer one, in terms of hops to the gradient origin node, is selected. Other metrics like signal strength or link quality could enhance the decision. However, UMG randomly selects a node from those closer to the "originGradAddr" - this process is efficiency in terms of complexity, processing and memory. The gradient descending process is resumed employing the selected candidate node as the next hop neighbour. The Routing



10
9
6
5
4
2
3
1
8
7
-2-
-1-
-4-
-3-
Legend
Newer Gradient
Older Gradient
-4-
# Hops in Older Gradient Path
Local Repair Phase Launched
Only Neighbour Replying to Local Repair and Selected
Selecting Older Gradient (Node 8) forms a Loop

Fig. 4.9: UMG - Local Repair Phase - Risk of Loop Formation when Coexisting Multiple Gradient Updates

Table is updated accordingly, with the candidate neighbour as the next neighbour to reach the "originGradAddr", i.e. the "receivedFromAddr" field (see Figure 4.3). If a candidate neighbour is not found, instead of doing a backtracking process, the node disables the Routing Table entry for the "originGradAddr" and discards the packet. When the end-to-end timer expires at the originator of the communication, and the maximum number of trials is not reached, the data packet descends the gradient again. In this case, a local repair process will be launched just one hop before the node which could not repair the gradient previously. This can be seen as an induced backtracking process which is always initiated by the originator of the communication. This way, if the maximum number of end-to-end trials is reached, UMG opts to update its gradient since the gradient is experiencing a high degree of instability and failure.

4.5.3 Acknowledgement Phase: Short Memory Cache

The acknowledgement phase starts when the destination node is reached, i.e. the "originGradAddr". The destination node provides the data to the higher layer and is in charge of issuing an acknowledgement packet back to the originator of the end-to-end communication, i.e. the "originAddr" field in the "DataGrad" message. The acknowledgement packet is a "DataGrad" message with no data enclosed (see Figure 4.7). Rather than having a dedicated field for the packet type, the acknowledgement packet is identified by its length as it only contains the header fields of the routing protocol. The acknowledgement packet is configured with the same values of the data packet being acknowledged. The "hops" field is reset to 0 and is increased at each intermediate node when climbing the gradient, in order to update the "realHops" field of the Routing Table (see Figure 4.3). It has to be noted that the "lastTimeUsed" field of the Routing Table is also updated. Both the "realHops" and the "lastTimeUsed" fields provide an indication on the status and distance of the end-to-end communication, useful for decision making in higher layers.

The acknowledgement phase depends on a caching mechanism (see Section 4.5.4.1) which caches information about received and sent data messages when descending the gradient. Every time a data packet is received, the key values defining a unique message are stored in the caching structure. The structure caches the key fields of the "DataGrad" message in a small array of "CacheMemory" type structure (see Figure 4.11). When a packet arrives, this is stored as "status" equals to "MEM_RECEIVED" and the "timeStamp" is set to the actual time of the mote. This mechanism is used to control cycles and avoid the reception of duplicate packets. When the node sends the message to the next node in the gradient, the entry for the message changes its "status" to "MEM_SENT". This mechanism differentiates received packets in the "CacheMemory" from those which are also waiting to be acknowledged by a packet coming from the final destination (gradient origin node).

When the acknowledgement packet is broadcast on the way back to the source node, neighbour nodes receiving the packet check their "CacheMemory" array for a match. If the packet is found and its status is set to "MEM_SENT", the node increases by one the number of hops in the acknowledgement packet ("hops" field), updates the "realHops" field in the Routing Table, and broadcast the acknowledgement packet. The only nodes broadcasting the acknowledgement packet are those which have previously forwarded the data packet when descending the gradient.

Furthermore, the link might fail due to a series of short and long time communication problems (see Section 4.5.1.1). Therefore, a reliable mechanism is in place to deliver the acknowledgement packets in each gradient ascending hop. Due to the fact that broadcast packets are not acknowledged at the MAC layer, the UMG routing protocol snoops packets in search for the broadcast packet with a higher value in the "hops" field. If the packet is snooped within a certain time, the entry in the "CacheMemory" array is disabled, i.e. "status" changes to "MEM_DISABLED". Contrary, if the timer expires, the acknowledgement packet is broadcast again. This is done for a maximum number of times. If after that, the message could not be acknowledged, i.e. snooped, it is assumed that bidirectionality in the path can not be achieved and the packet is discarded. However, the gradient spreads in the direction of the climbing process and the likelihood of establishing communication in the ascending direction should be higher than when descending the gradient. In the situation when a packet is discarded, the end-to-end timer at the source node will fire; the node either sends the data packet again or spreads its gradient while requesting the destination node to do the same. Additionally, UMG offers the possibility to utilise the gradient of the originator of the communication to keep on forwarding the acknowledgement packet, providing there is an entry in the Routing Table for the gradient origin node. If there is an entry, the procedure is the same as in Section 4.5.1, including the local repair mechanism in Section 4.5.2. For this special purpose, the "hops" value in the acknowledgement packet is set to its maximum, thereby acting as a flag and missing the counting functionality ("realHops"). This flag indicates to the UMG protocol that the packet is an acknowledgement message thus memory caching is not performed and the end-to-end acknowledgement at the destination node will not issue an acknowledgement packet on reply. Instead, when the originator of the communication receives a data packet with this flag, it will be treated as an acknowledgement, stopping the end-to-end timer. When bidirectionality does not exist and there is no gradient setup for the originator node, the end-to-end communication would not be achieved. In this situation the originator node spreads its gradient by querying the destination node, both gradients will be setup and end-to-end communication will occur one way or another.

4.5.4 Implementation

A single component manages the data packet delivery mechanism through all its phases: gradient descent, local repair and acknowledgement. In this component three queues are employed to manage the flow of packets in the different phases of the data transport process. The main queue ("SendQueue") stores all packets to be sent, either from the received interface or from higher layers, and operates under a FIFO policy. By default, UMG employs a size of 10 packets which has been proved to be an efficient limit providing reliability for a large range of high traffic scenarios presented in the evaluation. A small queue is employed to store packets to be sent by higher layers which are to be delivered to destination nodes. Messages in this queue are removed when a corresponding end-to-end acknowledgement packet is received; the message is then acknowledged to higher layers.

This queue has been set to 1 packet by default, thus requiring for a higher layer to implement a queue for the management of its outgoing messages. A third queue is employed to store those packet from the "SendQueue" for which a local repair process of their gradient is occurring, i.e. the "LocalRepairQueue". Once the gradient is fixed packets are dequeued from this queue and place in the "SendQueue" to be sent; if the local repair is not successful, then the packet is dequeued from the "LocalRepairQueue" and discarded. This queue has been set by default to 3 packets and has also been proved to be a sufficient limit for the stress testing performed in the evaluation section. When the component dequeues a packet from the "SendQueue" to be forwarded via a gradient which is being repaired, the packet is enqueued again in the "SendQueue". This way, it waits for the gradient to be fixed and does not obstruct the sending process of the rest of the packets in the queue. If the packet is the only element in the "SendQueue", then a delay before checking the queue is applied to avoid a repetitive inefficient checking process. However, when a new packet is enqueued, the "SendQueue" is checked immediately.

Furthermore, a dedicated component manages the sending and reception of the "Neighbours" message, depicted in Figure 4.8. This is a multi-purpose message for the discovery and query of nodes in the neighbourhood. The message is utilised in the local repair phase (see Section 4.5.2) to find alternative nodes to keep descending the gradient. In this phase, the type of the "Neighbours" message (see "type" in Figure 4.8) is set to "LOCAL_REPAIR" for the discovery message; neighbours reply with the message typed REPLY_LOCAL_REPAIR. A small dedicated queue, by default set to 4 packets - trading the sending of a second discovery message for RAM memory - is in charge of handling the flow of "Neighbours" messages. The component stores the information for the REPLY_LOCAL_REPAIR messages in a short table called "NeighboursTable" (see Figure 4.10). The table stores the address of the candidate nodes and their distance in hops to reach the requested gradient origin node (this value is stored by the neighbours in the "hops" field of the "Neighbours" message). The component sets a timer which gives a limited time for the neighbours to reply; when the timer expires the component informs the rest of the components of the availability of fresh information from the neighbourhood. This component also provides a function to select the best candidate based on a random selection over those nodes with the lowest hop distance to the gradient origin node.

The "Neighbours" message is also employed by the mobility support component (see Section 4.6) to discover the core neighbourhood of a node. For this purpose, the "Neighbours" message is typed as "NEIGHBOURS_DISCOVERY" and "REPLY_NEIGHBOURS_DISCOVERY". In this case, the rest of the fields are not relevant since the packet only informs of the presence of a neighbour node.

```
NeighboursTable
    uint16_t    neighbourAddr;
    uint8_t     hopsToRequestedGrad;
```

Fig. 4.10: UMG - Neighbours Table

4.5.4.1 Messages Caching Mechanism

For the caching of received and sent messages, an array of "CacheMemory" type structure is employed (see 4.11). The structure stores the next fields of the "DataGrad" message: "originGradAddr", "originAddr", "seq" and "hops" (see Figure 4.7). These fields make the packet unique. When a node receives a data packet in the gradient descent phase, these values are stored in the "CacheMemory" structure. In addition, the "status" field is set to "MEM_RECEIVED" and the time at the node is stored in the "timeStamp" field. When a message is sent to the next node in the gradient, the "status" field changes to "MEM_SENT". This mechanism controls duplicates in the reception of the packets and also identifies those packets which have been forwarded; this is useful for acknowledgement purposes (see Section 4.5.3).

```
CacheMemory
    uint16_t    originGradAddr;
    uint16_t    originAddr;
    uint8_t     hops;
    uint8_t     seq;
    uint32_t    timeStamp;
    uint8_t     status;
```

Fig. 4.11: UMG - Message Cache Memory Structure

The "CacheMemory" array gets accessed rapidly so it needs to be short and entries need to be freed when they are not in use. For this reason, a timer updates the "CacheMemory" array at intervals of time defined by the cached messages. A predefined time value is set for a "MEM_RECEIVED" entry to expire. The time value is calculated according to the maximum end-to-end round trip time. When the "status" of an entry is set to "MEM_SENT" and an acknowledgement broadcast packet is received which matches the entry, the "timeStamp" field is updated to the actual time. At this moment, the MEM_SENT "status" becomes a numerical variable which starts at 0 and keeps count of the broadcast trials of the acknowledgement packet. When the entry "status" is equal to numerical values (0.."MAX_BROADCAST_TRIALS"), the node waits a backoff time for an acknowledgement to be snooped. When the time is elapsed, the "timeStamp" field is set to the actual time and the "status" field is increased by one. A maximum number of trials (broadcast of the acknowledgement packet) are performed before the entry is finally disabled. If the broadcast acknowledgement with a higher value in the "hops" field is snooped, the entry is also disabled, i.e. status is assigned to "MEM_DISABLED", to indicate that the acknowledgement packet is not to be broadcast anymore by the node.

A timer is employed to control the next update of the "CacheMemory" type structure. The timer is set to fire according to the first entry due to expire. This can be calculated using the "timeStamp" field, the "status" field, and the time expiration values for each "status" field.

In the event of the "CacheMemory" array gets full, i.e. none of the entries "status" fields are set to "MEM_DISABLED", a mechanism based on message priorities decides which entry is dispensable and can be replaced. The mechanism follows the following order for entries to be replaced: 1st) first entry to expire with "MEM_RECEIVED" status with the highest number of hops and 2nd) first entry to expire with "MEM_SENT" status with the highest number of hops. "MEM_SENT" entries have a higher priority due to the fact that are waiting for an acknowledgement packet. However, a way of increasing the likelihood of selecting a "MEM_RECEIVED" entry over a "MEM_SENT" entry when the time to expire between them falls below a predefined value has been implemented. This is achieved by incrementing the "MEM_SENT" entry expiration time with a predefined threshold value, the DELAY_SENT_ENTRY constant. Therefore, for the "MEM_SENT" entry to be disabled over the MEM_RECEIVED entry, the quickest "MEM_SENT" entry to expire has to be DELAY_SENT_ENTRY milliseconds closer to expire than the quickest "MEM_RECEIVED" entry to expire. In addition, the number of hops ("hops" field) is also a factor which introduces a delay in the expiration time of an entry. The higher the number of hops, the better to remove the entry due to the fact that the closer the node to the destination, the faster the entry had to be disabled by an acknowledgement packet. Thereupon, a predefined delay time ("DELAY_HOPS") multiplied by the number of hops ("hops" field) is subtracted from the expiration time of the entry; note that an upper limit in the number of hops is set to avoid excessive subtraction. Entries with the "status" field equals to integers (0.."MAX_BROADCAST_TRIALS") are not to be considered in this process due to their high priority.

The "CacheMemory" array could also be implemented as two separate queues, one for "status" equals to "MEM_RECEIVED" and the other for the rest of the status. Therefore, the entries will be expiring in a FIFO order, which suits the active queue model. This way, Active Queue Management (AQM) schemes could be applied at the queue with "MEM_RECEIVED" status.

4.6 Mobility Support

Dynamically changing topologies in ad hoc networks commonly necessitate a proactive approach to maintain the routing connectivity of the network. If the degree of mobility is high and data is transmitted frequently in most of the areas of the network, then the periodic update comes as a good solution. However, if we consider that many applications in ad hoc networks, more specifically in wireless sensor networks, transmit data intermittently, then the volume of control traffic required in the proactive approach to maintain the network updated might be expensive in terms of communication. Specifically for those scenarios where a network is composed of a few transient nodes, where communication activity might occur only in some parts of the network, and where data communication will be infrequent, the periodic proactive approach becomes inefficient. Furthermore, in many application scenarios for WSN, the topology does not change particularly rapidly. Wireless transmission ranges are in the order of tens to hundreds of meters and a great percentage of nodes are static. Some nodes might move at different speeds and some of them will only have local mobility. In tandem with this, a set of neighbour nodes may not be moving with respect to each other - even though they are physically changing position. In all of these situations, communication and energy may be conserved by opportunistically updating only areas with activity, while avoiding the proactive beacon approach. Nevertheless, the latency in the communication might increase when a route discovery process is launched to update the routing connectivity in areas with no activity.

One of the features in the design of the UMG routing protocol is to avoid the periodic beacon messages by assuming that areas with no communication activity do not require recurrent updates, whilst areas with activity exploit opportunistic communication to update their connectivity status. In other words, UMG follows the opportunistic and reactive paradigms, employing eavesdropping mechanisms rather than using periodic updates. The opportunistic behaviour of this approach might produce disconnected areas where no communication activity has been occurring during a long period of time, at the benefit of reducing communication activity and therefore network contention.

In UMG, every message received or snooped is analysed to determine the source of the packet, i.e. the neighbour address. Areas with no communication will not be updated until communication takes place. In this case, if mobility has occurred and the topology of the network has changed, the use of local repair mechanisms (see Section 4.5.2) or even the creation of a new gradient (see Section 4.3.1) will be required. Detecting when a node is changing neighbourhood is the key to update its routing status. In UMG, there are two situations when mobility of a node occurs: i) when the node is only a router and ii) when the node is also a gradient sink. In both situations when the node leaves its neighbourhood, its routing table becomes non-functional, therefore all the gradient entries are disabled, although not deleted. The cost in messages of integrating the node as a fully working router in the new neighbourhood might be too high if there are a lot of gradient entries. In addition, if the mote keeps moving, routing table updates become highly inefficient.

In the situation where the node changing neighbourhood is a sink, meaning that it has already spread the gradient, the node is responsible to update its gradient so packets can still descend its gradient towards its new position. A global gradient setup is expensive in terms of messages and, unless the network topology has changed drastically - indicated by multiple nodes requesting the gradient origin node to spread its gradient due to a poor end-to-end delivery ratio -, it might not be required. The efficient solution proposed in UMG consists in a scoped gradient setup. This mechanism performs a scoped gradient update in terms of hops such that the old neighbourhood of the moving sink node receives the update. In this regard, the scope needs to be over-estimated according to the new position such that all the old neighbours are reached. The new gradient setup will have a newer sequence value which will differentiate from old gradient setups from the same sink node. Once the gradient is spread, packets will be forwarded from any node in the network by progressing through the next node in the gradient, whether the node has a new sequence value or the number of hops is higher. For instance, a node A containing an old sequence entry would relay the packet to its next node B which has been updated with the new sequence (as it was within the scope of the gradient update). In this situation, node A would have a lower number of hops since node B has been updated with the hop distance to the new position of the moving sink. This situation will not affect the descending of the gradient but it will affect the end-to-end calculation of the hop distance. To overcome this problem, the acknowledgement packet climbing the gradient in the Acknowledgement Phase (see Section 4.5.3) updates the "realHops" field of the Routing Table of the intermediate nodes with the real hops value carried in the "hops" field of the packet. In addition, when selecting a neighbour node to repair a gradient (see Local Repair Phase in Section 4.5.2), the restriction of selecting the node with the newer sequence value (even if the number of hops is higher than old sequences) will enable the descending towards the new position of the moving sink. Nevertheless, inefficiencies in the gradient descending process can be produced when packets at closer nodes to the new position of the gradient origin node need to route away from it, via descending an old gradient, before starting to descend the new gradient.

This effect can be seen in Figure 4.12 where the gradient origin node, also known as the sink S, moves from position S to S' and then to S". When in position S, the gradient with sequence 1 ("seq" in the Routing Table) is spread over all the nodes in the network, i.e. a global gradient setup.

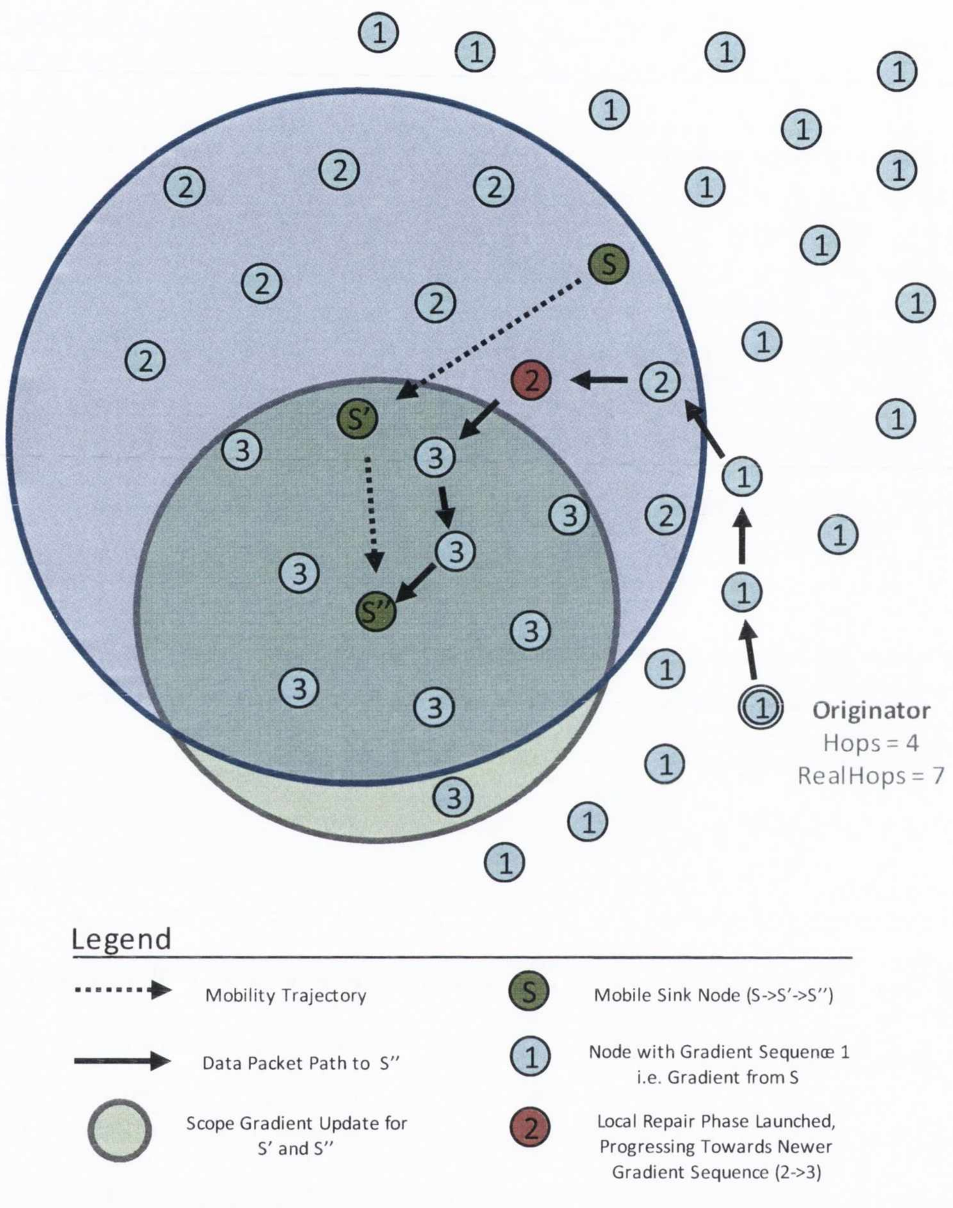

Originator
Hops = 4
RealHops = 7
Legend
Mobility Trajectory
Data Packet Path to S''
Scope Gradient Update for S' and S''
Mobile Sink Node (S->S'->S'')
Node with Gradient Sequence 1 i.e. Gradient from S
Local Repair Phase Launched, Progressing Towards Newer Gradient Sequence (2->3)

Fig. 4.12: UMG - Mobility Support Example

However, when mobility is detected at position S', the sink spreads its gradient with sequence 2 with a scope of 2 hops such that it reaches its old position. Another scoped gradient update with sequence 3 is triggered when the sink moves to S''. At this stage, the "Originator" node sends a packet towards the sink node. The packet is unicast to the next hop node in the gradient which in this case has a gradient sequence of 1. The packet keeps on descending the gradient with sequence 1 until it reaches a next hop node which belongs to gradient sequence 2. At some point, a node in the gradient with sequence 2 does not have a valid next hop node and therefore a Local Repair Phase is launched. In this case, neighbour nodes reply with sequence 2, providing a hop distance of 1, however the node with the highest sequence is chosen, i.e. sequence 3. Now the packet keeps on descending the gradient with sequence 3 until it reaches the sink at position S". It has to be noted that the shortest route from Originator to S" is of 3 hops, however the packet has traveled 7 hops. The "hops"' value contained at the Originator node corresponds to the hop distance to the sink node at position S, which has not been modified by any gradient update. When the acknowledgement packet climbs the gradient towards the Originator, the "realHops" value is updated in the Routing Table of the intermediate nodes. This is of high interest to the UMG routing protocol as the difference between the "hops" and the "realHops" at the Routing Table would indicate the degree of mobility of the sink node and the inefficiency of the routing process. According to these values, a global gradient update can be requested from the sink node.

For the scoped gradient update to be launched, the node needs to promptly detect when a node has changed its relative position with respect to its neighbourhood, i.e. relative mobility. For this purpose, a mobility estimation mechanism which detects if a node has changed its relative position based on eavesdropped data has been designed [217]. The approach builds a set of temporal shift Bloom filters (see Section 4.4.1) to store historical overheard neighbour addresses at continuous steps in time. A probability-based model is employed to assess the mobility state of a node according to the information cached in the set of Bloom filters. A recursive estimation approach enhances the accuracy of the mobility assessment by incorporating previous expressions of mobility certainty. The model accommodates most of the uncertainty scenarios where wireless connectivity might be temporarily disrupted and where the relative position of the node(s) may or may not have changed.

4.6.1 Recursive Mobility Estimation with Temporal-Shift Bloom Filters

The creation of a historical record of eavesdropped neighbours addresses is key to evaluating a node's relative mobility, i.e. changing neighbourhood. The core neighbourhood of a node is formed by a group of static, in relative terms, neighbours. Over time, other nodes can come and go including those which, by being at the edge of the communication range, may have local mobility. Nodes seen for a short period of time can distort this representation. To accommodate this, overheard information is split into a number of consecutive chronological intervals. The time window for overhearing packets, prior to launching the mobility evaluation process, is defined from the average speed at which a node will move and the mean transmission range of a node. The transmission range is a function of the transmission power and the environment where nodes are placed, and needs to be calculated beforehand through testing. These two factors provide an indication of the average time that a moving node would take to leave its neighbourhood area radius. This time, known as the "Evaluation Time", is employed as the mobility evaluation interval and is given by the equation:

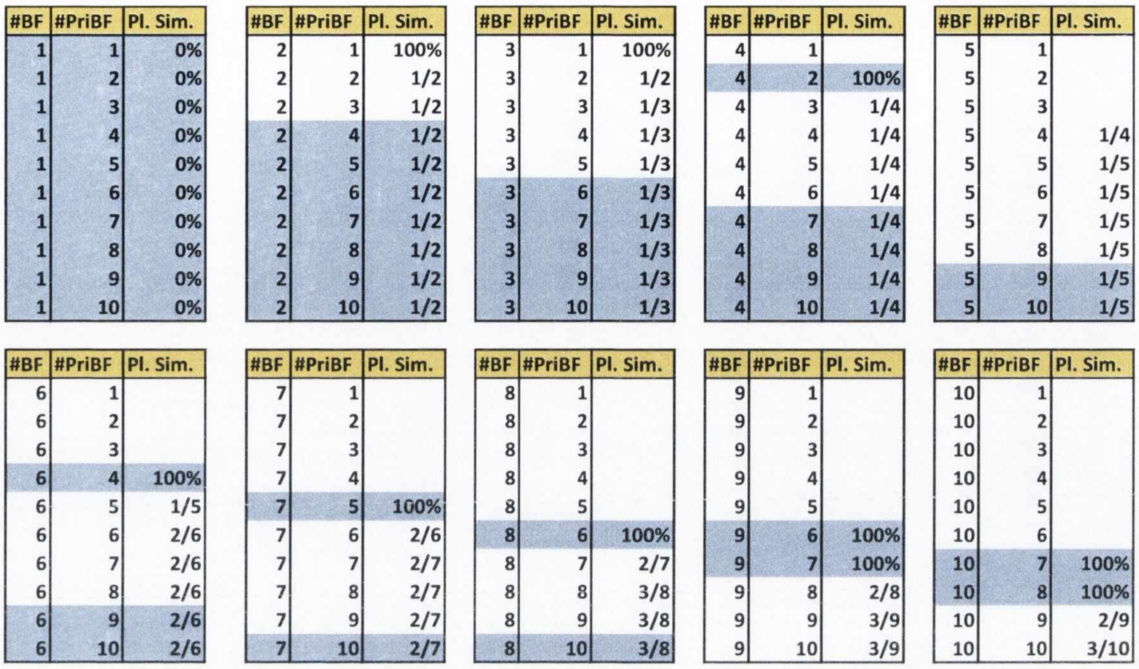
$$\text{Evaluation Time(s)} = \frac{\text{Transmission Range(m)}}{\text{Speed(m/s)}} \tag{4.9}$$

The mobility evaluation interval, i.e. the Evaluation Time, defines the interval prior to launching the mobility evaluation process. In this interval, all the overheard neighbour addresses are hashed into a set of Bloom filters which act as independent consecutive memory structures. Initially, an independent Bloom filter, the "PrimaryBF" (PriBF), will hash the addresses of the core neighbourhood. The initial neighbourhood can be established by explicitly requesting the neighbours to reply, or by eavesdropping for an initial period of time. The source of an overheard packet is hashed into the Bloom filter. Once the "PrimaryBF" is populated, every overheard packet will be stored in the "Active" Bloom filters. The number of "Active" Bloom filters is determined by the "split factor". The shifting time to start storing information in the next "Active" Bloom filter is calculated by:

$$\text{Shift BF} = \frac{\text{Evaluation Time(s)}}{\text{Split Factor}} \tag{4.10}$$

If no data communication activity occurs, the evaluation mechanism will not be launched as the "ActiveBF" will be empty. A maximum time can be preset such that, if no mobility evaluation process takes place, a "Hello" message discovery process starts; however, UMG does not set this time as it works on a reactive fashion. Each "ActiveBF" and "PrimaryBF" keep a counter of the number of different neighbour addresses which have been stored, i.e. #BF and #PriBF respectively. When the last "ActiveBF" is populated, the evaluation process starts by comparing each of the individual "ActiveBF" with the "PrimaryBF". This comparison calculates the similarity of the "PrimaryBF" and the "Active" Bloom filter as:

$$\text{Similarity(BF, PriBF, \#BF, \#PriBF)(\%)} = \begin{cases} \dfrac{\text{\# bits in ``BF'' matching ``PriBF'' 1's}}{\text{\# bits in ``PriBF'' set to 1}} & \text{if \#BF} \geq \text{\#PriBF}, \\ \\ \dfrac{\text{\# bits in ``BF'' matching ``PriBF'' 1's}}{\text{\# bits in ``BF'' set to 1}} & \text{if \#BF} < \text{\#PriBF}. \end{cases} \tag{4.11}$$



#BF	#PriBF	Pl. Sim.
1	1	0%
1	2	0%
1	3	0%
1	4	0%
1	5	0%
1	6	0%
1	7	0%
1	8	0%
1	9	0%
1	10	0%

#BF	#PriBF	Pl. Sim.
2	1	100%
2	2	1/2
2	3	1/2
2	4	1/2
2	5	1/2
2	6	1/2
2	7	1/2
2	8	1/2
2	9	1/2
2	10	1/2

#BF	#PriBF	Pl. Sim.
3	1	100%
3	2	1/2
3	3	1/3
3	4	1/3
3	5	1/3
3	6	1/3
3	7	1/3
3	8	1/3
3	9	1/3
3	10	1/3

#BF	#PriBF	Pl. Sim.
4	1	
4	2	100%
4	3	1/4
4	4	1/4
4	5	1/4
4	6	1/4
4	7	1/4
4	8	1/4
4	9	1/4
4	10	1/4

#BF	#PriBF	Pl. Sim.
5	1	
5	2	
5	3	
5	4	1/4
5	5	1/5
5	6	1/5
5	7	1/5
5	8	1/5
5	9	1/5
5	10	1/5

#BF	#PriBF	Pl. Sim.
6	1	
6	2	
6	3	
6	4	100%
6	5	1/5
6	6	2/6
6	7	2/6
6	8	2/6
6	9	2/6
6	10	2/6

#BF	#PriBF	Pl. Sim.
7	1	
7	2	
7	3	
7	4	
7	5	100%
7	6	2/6
7	7	2/7
7	8	2/7
7	9	2/7
7	10	2/7

#BF	#PriBF	Pl. Sim.
8	1	
8	2	
8	3	
8	4	
8	5	
8	6	100%
8	7	2/7
8	8	3/8
8	9	3/8
8	10	3/8

#BF	#PriBF	Pl. Sim.
9	1	
9	2	
9	3	
9	4	
9	5	
9	6	100%
9	7	100%
9	8	2/8
9	9	3/9
9	10	3/9

#BF	#PriBF	Pl. Sim.
10	1	
10	2	
10	3	
10	4	
10	5	
10	6	
10	7	100%
10	8	100%
10	9	2/9
10	10	3/10

Fig. 4.13: UMG - Predefined Model of the Plausible Similarity for each combination of cardinality of the "ActiveBF" and "PrimaryBF". The Plausible Similarity (Pl. Sim.) value is indicated either with a percentage or with a fraction of $\frac{\#Tolerance}{\#BF}$ according to Equation 4.12. Shadow combinations indicate UNCERTAIN state.

The "Similarity" percentage value in Equation 4.11 represents the ratio of the number of neighbours seen by an "ActiveBF" which are core neighbours; that is the number of neighbours also stored in the "PrimaryBF" to the total number of neighbour's addresses hashed in the filter ("ActiveBF" or "PriBF") with the lowest cardinality of neighbours' addresses. Each "ActiveBF" (BF) is intended to hash addresses of nodes that are static, mobile or appear within the neighbourhood subsequent to the formation of the "PrimaryBF". Then, the "Similarity" ratio seeks to capture an indication of the proportion of the nodes in the "ActiveBF" which belong to the core neighbourhood represented by the "PrimaryBF", bounded to the minimum cardinality of nodes hashed from the two filters.

$$\text{Plausible Similarity}(\#\text{BF},\ \#\text{PriBF},\ \#\text{Tolerance})=$$

$$\begin{cases} 100 - (\frac{\#Tolerance}{\#BF} \times 100) & \text{if } \#\text{BF} \geq \#\text{PriBF}, \\ 100 - (\frac{\#Tolerance}{\#PriBF} \times 100) & \text{if } \#\text{BF} < \#\text{PriBF}. \end{cases} \tag{4.12}$$

In addition, the Plausible Similarity (Pl. Sim.) in Equation 4.12 indicates the minimum level of "Similarity" that must be achieved for a node's neighbourhood to assume it remains unchanged. Plausible Similarities are precalculated according to Equation 4.12 creating a predetermined table model to be queried. A predetermined table model (generated for a maximum of 10 neighbours) with their associated percentages of plausible similarity and thresholds for the number of neighbours in the "ActiveBF" and the "PrimaryBF" has been created (see Figure 4.13). In Equation 4.12, the #Tolerance parameter is used to establish the level of allowable ("plausible") change within the Similarity expression according to the cardinality of the filter in the denominator. Indeed, the #Tolerance parameter indicates the number of neighbour nodes which are allowed not to form part of the core neighbourhood to still consider the neighbourhood of the node has not change. The Plausible Similarity depends on the relationship between #BF and #PriBF, giving rise to two different formulations of the expression.

Equation 4.13 establishes criteria which map the numeric Similarity and Bloom Filter measures into the following Mobility States:

- MOBILITY, mobility is estimated.
- STATIC, node has seen sufficient core neighbours to estimate that there is not mobility.
- UNCERTAIN, there is not enough information to evaluate whether the node is moving.

$$\text{Mobility Evaluation State(Sim., \#BF, \#PriBF, Pl.Sim.)=}$$

$$\begin{cases} \text{MOBILITY} & \text{if } \#\text{BF} \ggg \#\text{PriBF}, \\ \text{MOBILITY} & \text{if } (\#\text{BF} \gg \#\text{PriBF}) \; \&\& \; (\text{Sim.} < 100\%), \\ \text{UNCERTAIN} & \text{if } (\#\text{BF} \gg \#\text{PriBF}) \; \&\& \; (\text{Sim.} = 100\%), \\ \text{STATIC} & \text{if } (\#\text{BF} \geq \#\text{PriBF}) \; \&\& \; (\text{Sim.} \geq \text{Pl.Sim}), \\ \text{MOBILITY} & \text{if } (\#\text{BF} \geq \#\text{PriBF}) \; \&\& \; (\text{Sim.} < \text{Pl.Sim}), \\ \text{STATIC} & \text{if } (\#\text{BF} < \#\text{PriBF}) \; \&\& \; (\text{Sim.} \geq \text{Pl.Sim}), \\ \text{MOBILITY} & \text{if } (\#\text{BF} < \#\text{PriBF}) \; \&\& \; (\text{Sim.} < \text{Pl.Sim}), \\ \text{UNCERTAIN} & \text{if } (\#\text{BF} \ll \#\text{PriBF}) \; \&\& \; (\text{Sim.} \gtrsim \text{Pl.Sim}), \\ \text{MOBILITY} & \text{if } (\#\text{BF} \ll \#\text{PriBF}) \; \&\& \; (\text{Sim.} < \text{Pl.Sim}). \end{cases} \tag{4.13}$$

Each relevant "Mobility Evaluation State" (see Equation 4.13) is established for consecutive temporal states of "ActiveBF" and is associated with the interval of time in which the state was assessed. In order to improve the reliability of the estimation, the number of temporal Bloom filters ("ActiveBF") determined by the Equation 4.10 represents temporal states which are recursively estimated. A consecutive combination of states increases confidence in the estimation of the final mobility state for the evaluation time. For instance, if the "MOBILITY" state appears in 3 out of 4 consecutive states of the evaluation time, then mobility is assumed; a change from "STATIC" state to a set of consecutive "UNCERTAIN" states indicates the lack of definitive information. The combination of the number of states, and the order and consecutive states required, is defined within the model which provides a final mobility state for the evaluation time. These parameters can be changed to control the mobility detection speed versus the reliability of the detection.

When mobility is detected, a neighbourhood discovery process is launched to update the set of core neighbours in the "PrimaryBF" for future estimations. Based on the result, the distance in number of hops to reach all the nodes in the old neighbourhood is over-estimated from preestablished values.

4.6.1.1 Implementation

The mechanism has been designed as a separate component to operate with other routing protocols. The "Split Factor" (see equation 4.10), which determines the number of "Active" Bloom filters, has been set to 5. The size of the Bloom filter has been set to 64 bits. This helps to reduce the number of false positives through establishing that the maximum number of elements in the filter can be 12, i.e. 12 different neighbours. According to Equation 4.7, the probability of getting a false positive error in the Bloom filter with a size of 64 bits, for 12 data items inserted, and 4 hash functions, is 0.07915. The number of hash functions employed is close to the optimal number for minimizing the error, which according to Equation 4.8 is 3.69.

4.7 Summary

A new routing protocol designed to provide versatile communication support to higher layers in the domain of Wireless Sensor Networks has been presented in this chapter. UMG is a gradient-based routing protocol supporting reliable P2P communications. Its design is a combination of mechanisms for creating, descending and climbing the gradient in a reliable and end-to-end manner. UMG provides an algorithm for the proper formation of the gradient, i.e. avoiding loops and local minima. The protocol efficiently tolerates moderate mobility of nodes through the use of a scoped gradient setup mechanism which updates the routes towards a moving node for a specific area. UMG employs the gradient setup to efficiently advertise node services and thereby enhances the data-centric searching process. In addition, UMG integrates a probabilistic mobility estimation mechanism which employs opportunistic communication to detect when a node has changed neighbourhood.

UMG's design inherently supports the unstructured creation of clusters where some consumer and/or producer nodes might spread their gradient with a limited scope within a virtual cluster. Specially selected nodes, such as cluster heads, can spread the gradient with a wider scope thereby creating overlay networks of nodes which require communication within different area sizes in terms of hops.

Furthermore, UMG operates as the main routing substrate for the TinyTorrents protocol, providing a peer-to-peer cooperative data distribution framework for WSN. It provides end-to-end communication between consumers and producers of data. While UMG transports data within the scope of the gradients, the TinyTorrents protocol employs a selective data dissemination approach to push data to distant nodes in the network. The rationale behind this approach is that nodes in large sensor networks tend to limit communication within a certain scope for efficient routing, as routing at higher hop distances has proved to be inefficient and impractical. In this way, scalability is achieved in a reliable cross-layer fashion where the UMG routing protocol provides up-to-date information of a scoped area to the TinyTorrents protocol for scalable and efficient distribution of data. In addition, the TinyTorrents protocol has been designed to advertise its services via descriptions of data. In this context, UMG's advertisement mechanism provides support for describing the "interest" of data producer and consumer nodes. Moreover, descriptors can be used for searching purposes in content distributed algorithms. This can be exploited by higher layers in areas like distributed data fusion, service discovery, distributed storage or swarm decision making.

Chapter 5

The TinyTorrents Protocol

This chapter presents the TinyTorrents (TT) protocol, a selective data dissemination protocol for Wireless Sensor Networks. The TinyTorrents protocol leverages a routing protocol which provides reliable point-to-point communication, i.e. the TinyHop or UMG routing protocols. Two versions of the protocol exist: i) the TinyTorrents Centralised version, where a central node coordinates the distribution process, and ii) the TinyTorrents Decentralised version, a fully decentralised and scalable solution for the distribution of data in WSN.

The novelty of the TinyTorrents architecture, i.e. the TinyTorrents protocol operating on top of the UMG routing protocol, arises from the challenge of disseminating data to a subset of nodes placed at disparate points in the network in a cooperative, reliable and efficient manner such that traffic is balanced. In this regard, the TT protocol is a novel selective data dissemination protocol for WSNs which employs BitTorrent-based P2P content distribution concepts. The TT architecture:

- provides fair and efficient cooperation amongst a subset of consumer nodes in the network for the equitable distribution of data while balancing the traffic load,
- employs a set of peer selection policies which seek to balance the network traffic and foster data dispersion, mainly based on the proximity of the peer and the time when the consumer shows interest in acquiring the data,
- allows for application layers to configure the degree of data replication in the network, where nodes choose whether to participate in the distribution process, thereby making possible the creation of overlays of communities,

- implements a novel decentralized P2P data distribution solution which increases the degree of robustness and fault tolerance of the network by having every consumer node acting as a partial tracker of data for its nearby area,
- provides unstructured service advertisement and discovery mechanisms at both layers for efficient and robust lookup of nearby consumer nodes and for querying activities, enabling the network to function as a data storage vehicle,
- enables efficient scalability in the data distribution process - only limited by the inter-consumer distribution in the network,
- succeeds in providing a versatile, cooperative, reliable, scalable and decentralized communications architecture for selective data dissemination where as similar solutions only employ epidemic-based dissemination approaches for reprogramming or data collection purposes.

This chapter is organised as follows: The first section introduces the TinyTorrents protocol and its main features. Next, a functional description of all the various phases comprising the behaviour of the TinyTorrents protocol is provided which describes the flow of messages involved in the distribution process. The TinyTorrents mote architecture is depicted and described in the following section, including the storage structures employed in each component and phases of the protocol. The strategies employed to select peers in the data distribution process are then presented. Progressing into the following section, the centralised and decentralised versions of the TinyTorrents protocol are explained and their scalability discussed. The subsequent section describes a set of unstructured discovery mechanisms which are employed to locate partial trackers in the decentralised version of the TinyTorrents protocol. Finally, service discovery in the TinyTorrents protocol is explained, and the chapter is concluded.

5.1 A Data Distribution Protocol for WSN

The TinyTorrents protocol has been designed to operate as a P2P data distribution layer sitting above a reliable end-to-end routing protocol in a Wireless Sensor Network. The ideas behind the most popular Internet peer-to-peer content distribution protocol, the BitTorrent protocol, have been the corner stone of the design of the TinyTorrents protocol. However, the different behaviour of WSNs, the constraints of sensor devices, and the ad hoc multihop nature of these networks, impede the straightforward adaptation of the BitTorrent protocol to fully operate in WSNs. In this regard, the TinyTorrents protocol has been designed to benefit from P2P content distribution concepts while tackling the problems presented when distributing data in multihop wireless networks of constrained devices.

The TinyTorrents protocol offers an efficient mechanism for data publication and selective data distribution over the sensor network. Applications using the TinyTorrents protocol layer benefit from a series of reliable algorithms which allow for a collaborative distribution of data by employing application-level decision policies. TinyTorrents provides service advertisement and discovery mechanisms which allow for data tagging using human readable vocabulary. It offers a simple interface to tag, publish and distribute application layer data. Like in the BitTorrent protocol, data is split into small units called "pieces". The data is represented with a file called "TinyTorrent" which, in the remainder of the document, will be interchangeably called "torrent". A torrent contains a description of the data and some data control information, for instance the number of pieces in which the data is divided, data integrity values, and the next node to contact in the discovery of other nodes in the swarm of the torrent, i.e. the tracker node. Peer-to-peer data communication is achieved amongst nodes belonging to the swarm of the torrent, seeking to balance resource consumption and to dissipate the burden of data transfers and network overhead fairly.

The TinyTorrents protocol has been designed to operate in both a partially centralised and a decentralised manner. In the centralised approach, a central node, i.e. the tracker, keeps control of the peers involved in the swarm of each torrent. On request from a peer wishing to join the swarm, the tracker node selects the list of peers which the requesting node should employ to fetch the data. While a central tracker node reduces the system fault tolerance, the tracker node is capable of regulating the data distribution process while maintaining some degree of fairness in the process. However, in the decentralised approach, the central tracker node disappears to increase the system fault tolerance, robustness and autonomy. In this modality, each peer belonging to the swarm of a torrent is capable of acting as a partial tracker for that particular file. The concept of "partial tracker" arises from the fact that the node only keeps track of a subset of the peers in the swarm. The partial tracker discovers peers from new peers involved in its own data acquisition process or when acting as a tracker. This way, each peer is potentially a partial tracker which manages the swarm of the torrent within a scope of the network; the scope of the swarm depends on the location of both the peer and the peers involved in the data fetching process for the torrent. The decentralised approach can be seen as a scalable selective data distribution protocol which efficiently disseminates data files to a set of interested nodes in the network.

To achieve data communication in the network, the TinyTorrents protocol needs to sit on top of a reliable routing layer which also provides networking status information, such that both layers operate in a cross-layer fashion in order to enhance the protocol behaviour and its overall efficiency. Two routing protocols have been designed. The first one, TinyHop [3], is a reactive end-to-end routing protocol which implements local recovery mechanisms to achieve bidirectional reliable paths. The second routing protocol, the Ubiquitous Mobile Gradient (UMG) (see Chapter 4), employs gradient-based routing techniques with robust and fault-tolerant mechanisms for the creation and navigation of the gradient field and seeks to deliver data packets in a reliable end-to-end fashion. The protocol integrates mechanisms for node service advertisement and discovery which are employed by the TinyTorrents protocol for the unstructured discovery of partial trackers. The TinyTorrents protocol, employing UMG as the routing substrate, provides a communications framework for the development of a new range of versatile applications which foster the collaboration among wireless sensor and actuator devices and the autonomous behaviour of the sensor network

5.2 Functional Description of the TinyTorrents Protocol

This section initially introduces the TinyTorrents protocol phases in a functional overview of the protocol. Next, each phase is described in detail.

The TinyTorrents protocol is comprised of the following phases which are executed when a node in the network needs to publish or to acquire some data file:

- Publish Phase: This phase is initiated when a node, known as the "initial seeder", contains data to be published, i.e. the node is a "producer" in the network. A metadata file is generated, i.e. the "TinyTorrent" file or torrent, which describes and represents the data file in the network. Every node in the network potentially participates in the dissemination of the torrent file. When a torrent is received at a node, the application layer decides whether to acquire the data file associated with the torrent and whether to forward the torrent file. Nodes starting to fetch the data file for a received torrent become "peers" of the swarm of the torrent.

- Peer List Request Phase: A node wishing to fetch the data file associated with a received torrent, thus becoming a "consumer" of the torrent, needs to request a list of peers, known as the "peer list", by contacting a designated tracker node. A tracker node, whether in the centralised or decentralised version of the TinyTorrents protocol, maintains a list of peers participating in the data fetching process of each torrent. At any given time, the complete list of peers for a torrent is known as the "swarm of the torrent". A node might decide to acquire multiple torrents thereby becoming a peer in multiple swarms. The tracker is in charge of selecting a small subset of the swarm of peers to send back to the requesting node. A node requesting a peer list from a tracker node automatically becomes a peer in the swarm of the torrent.

- Handshake Phase: Once a node receives a list of peers from which to acquire the data file, a handshake process is performed with each of the peers in the list. The handshake process is responsible for finding out i) whether the peer is still contactable and, ii) the list of pieces of the data file contained by each peer. The data file is segmented into smaller "pieces" of data which become the atomic data unit in the TinyTorrents protocol; a piece of data fits in a single message for efficient and reliable transportation.

- Piece Request Phase: Once the handshake phase is completed, the node starts to request pieces of data from those contactable peers which contain the piece/s. The node performs a piece selection process in order to choose the rarest piece from all the pieces contained in the list of peers. The rarest piece from the remaining pieces is acquired first in order to foster a uniform piece distribution and a quick piece dispersion over the network. This minimizes the risk of a data file not being completed, i.e. all the pieces received. A peer selection process is then launched to select the peer from which to acquire the piece. Different peer selection policies can be applied to increase fairness and make the data distribution process more uniform and efficient. A node which contains all the pieces of data for a torrent is known as a "seeder" of the file.

```
TinyTorrent
    uint8_t     key[KEY_SIZE];
    uint8_t     length;
    uint8_t     descriptorBF[BF_SIZE];
    uint32_t    timeCreatedWRTLocalTime;
    uint16_t    tracker;
    uint8_t     checksum[MAX_PIECES];
```

Fig. 5.1: TT - TinyTorrent File Structure (also called "torrent")

The main structure of the TinyTorrents protocol is known as the "TinyTorrent" file, also called torrent (see Figure 5.1), and acts as a representative entity of a data file in the network, providing descriptive and control information. Each torrent is uniquely identified by its "key" field which has been defined as a combination of 4 bytes, i.e. KEY_SIZE is 4. The first two octets contain the identifier of the node, i.e. $2^{16} = 65535$ node addresses available in the network. The next byte is assigned to a sequence value which is incremented with every new torrent generated by the local node, while the last octet stores a randomly generated number. The size of the data file associated with the torrent is stored in the field "length" as the number of bytes. A description of the data file employing human-readable tags is stored in the field "descriptorBF", a Bloom filter structure (see Section 4.4 for more detail). Currently the size of the descriptor Bloom filter (BF_SIZE) is set to 4 bytes, but can be easily modified to decrease the likelihood of getting false positive descriptions, particularly when the descriptor of the file contains a high number of tags. In addition, the time when the torrent was generated is stored in the "timeCreatedWRTLocalTime" field of the torrent in binary millisecond precision. The timestamp field is 4 bytes long and thus is capable of counting up to 2^{32} milliseconds, i.e. 48.5 days before the timer rolls over. This value puts a limitation on the time before a torrent in the network is retrieved by a gateway node capable of matching the network timestamp of the torrent against the atomic clock time. This field acts as a reference timestamp value with respect to the time at a sender node for the purposes of performing soft real-time synchronization when the torrent is disseminated over the network. Moreover, the torrent structure contains a field ("tracker") for the purposes of designating the address of the node acting as a tracker. Finally, a byte array ("checksum") contains the checksum value of each of the pieces in which the data file is divided. The checksum value is computed for data integrity purposes as packets can get corrupted in the wireless transmission. It is calculated employing the "Adler-32" checksum algorithm [218] which trades reliability for efficiency in terms of computational speed. The 4 bytes resulting from the Adler-32 algorithm are combined into a single byte by applying a bitwise exclusive-OR (XOR) checksum. The checksum array length matches the maximum number of pieces per file (MAX_PIECES), which has been defined as 16.

5.2.1 Publish Phase

This phase is triggered by the application layer when a data file needs to be published to other nodes in the network. The application interacts with the TinyTorrents protocol via the TinyTorrents interface. This way when a torrent is received by a node, the application layer decides if and when to keep on disseminating the torrent to neighbours, and if and when to fetch the data. Therefore, the application has control over the behaviour of the data distribution process. In addition, the application layer inputs the vocabulary of tags (keywords) through the TinyTorrents interface and selects, for each generated torrent, the combination of keywords which defines its data. The application layer calls the TinyTorrents interface to create the torrent file, indicating the data file to be published. For reliability and efficiency in terms of packet communication, the torrent file is transported in a single packet known as the "PublishTinyTorrent" message (see Figure 5.2). In addition to the torrent fields, the field "fromId" of the message contains the identifier of the node which generated the torrent, i.e. the initial seeder. Despite the fact that the initial seeder of the torrent is also encapsulated in the first two bytes of the "key" field, the redundant "fromId" field is in place as a backup in case the torrent key might be generated in any other unrelated manner in a future version.

```
PublishTinyTorrent Message
    nx_uint16_t   fromId;
    nx_uint8_t    key[KEY_SIZE];
    nx_uint8_t    length;
    nx_uint8_t    descriptorBF[BF_SIZE];
    nx_uint32_t   timeWRTSender;
    nx_uint32_t   currentTimeAtSender;
    nx_uint16_t   tracker;
    nx_uint8_t    checksum[MAX_PIECES];
```

Fig. 5.2: TT - PublishTinyTorrent Message Structure

As mentioned earlier, the TinyTorrent structure is timestamped with the time at the local node (see "timeCreatedWRTToLocalTime" in Figure 5.1). However motes in the WSN are unsynchronized due to the fact that each sensor device has its own clock starting at different times. A variety of synchronization mechanisms exist which employ periodic beacon messages to keep the sensor network synchronized. However, these proactive mechanisms introduce extra congestion in the wireless medium. For this reason, the TinyTorrents protocol implements a soft real-time synchronization mechanism for the purpose of keeping the torrent creation time in sync with the clock at the nodes receiving the torrent. The mechanism works as follows: When a "PublishTinyTorrent" message is received, it contains both the torrent creation time at the sender, i.e. "timeWRTSender", and the current time at the sender when the message was sent, i.e. "currentTimeAtSender" (see Figure 5.2). The receiving node is able to calculate from these values the creation time for the torrent with respect to its local time, which is then stored in the torrent structure as "timeCreatedWRTLocalTime" (see Figure 5.1). This mechanism is performed in a single hop fashion, such that the average time elapsed from the time the "PublishTinyTorrent" message is timestamped and sent, to the time when the message is received and processed, can be calculated within a certain delay error threshold. The delay introduced depends on the congestion level at the routing and mac layers in both the sender and the receiver nodes and the packet propagation time. While the delay error is in the order of tens of milliseconds, and can be previously approximated, a small cumulative error over a high hop distance can produce a greater inaccuracy. Depending on the application, millisecond accuracy will require the adoption of additional synchronization algorithms, however accuracy in the order of a second might be sufficient for calculating the time when a torrent was generated for most applications. This mechanism saves messages and establishes a soft real-time data-level synchronization.

When a node receives the "PublishTinyTorrent" message and decides to keep forwarding the torrent, the protocol waits for a period of time indicated by the application layer before broadcasting the message. Due to the fact that the broadcast packet is not acknowledged, the protocol defines the number of times the message is broadcast in order to increase the delivery success. The successful dissemination of the torrent message over the network depends on the number of nodes participating in the forwarding process as well as the average number of neighbours per node.

In the centralised version of the TinyTorrents protocol, the torrent file is initially sent to the node appointed as the central tracker before being disseminated to the nodes in the network. The tracker is in charge of keeping records of the status of each torrent file in the network, which includes the identity of the nodes which contain some, or all, the pieces for the data file associated with the torrent. However, in the decentralised design, the central tracker is replaced by multiple partial trackers where the first tracker of a torrent is always the initial seeder itself.

Both in the centralised and decentralised versions of the TinyTorrents protocol, generated and acquired torrents, as well as their associated data, are stored in memory for seeding purposes.

5.2.2 Peer List Request Phase

This phase is initially launched when a node receives a torrent and the application layer instructs the TinyTorrents protocol to start the data fetching process. The consumer node needs to retrieve a peer list from which to acquire all the pieces of the data file. The consumer contacts the tracker node indicated in the "PublishTinyTorrent" message (see "tracker" in Figure 5.2). In the centralised version, the address of the tracker node remains constant during the dissemination of the torrent over the network. However, in the decentralised version, the address of the tracker node changes according to the dissemination of the "PublishTinyTorrent" message which enables the selection of a nearby partial tracker. The "tracker" field of the "PublishTinyTorrent" message is updated with the address of the local node before it is forwarded only if the node is to become a consumer, i.e. a potential partial tracker of the torrent. A node chooses the closest tracker address in terms of hops from all received "PublishTinyTorrent" messages before the torrent is due to be forwarded or its data fetched. The "tracker" field of the "TinyTorrent" structure (see Figure 5.1) is updated with the address of the closest partial tracker. The hop distance is obtained from the routing layer, which can be 0 when the node is out of the routing discovery scope. Other mechanisms are employed in the discovery of an efficient tracker in terms of proximity which are presented in Section 5.6.

A "PeerListRequest" message (see Figure 5.3) is sent to the tracker address on request of a valid peer list. The message contains the identifier of the requesting node ("fromId") as well as the key of the torrent ("key") from which the peer list is to be obtained.

```
PeerListRequest Message
      nx_uint16_t   fromId;
      nx_uint8_t    key[KEY_SIZE];
      nx_uint16_t   pieceBitVector;
      nx_uint8_t    numPeersFailed;
      nx_uint16_t   peerListFailed[MAX_PEERS];
```

Fig. 5.3: TT - PeerListRequest Message Structure

Tracker nodes are designed to employ selection schemes to construct a peer list according to information such as the chronological position when the node enters the torrent swarm, or the received piece bit vector of a peer. This strategies aim to regulate the traffic in the network by achieving high degrees of fairness. Peer list selection policies at tracker nodes are covered in Section 5.4.1.

```
PeerList Message
      nx_uint16_t   fromId;
      nx_uint8_t    key[KEY_SIZE];
      nx_uint8_t    numPeers;
      nx_uint8_t    position;
      nx_uint16_t   peerList[MAX_PEERS];
```

Fig. 5.4: TT - PeerList Message Structure

The tracker replies with a peer list from its swarm of peers for the requested torrent via the use of the "PeerList" message (see Figure 5.4). The message contains the address of the tracker node ("fromId") and the key of the torrent ("key"). The size of the peer list (MAX_PEERS) is configurable but by default has been set to 4, however the tracker might not contain as many peers in its swarm at a given time. For this reason, the message contains the number of peers ("numPeers") transported in the peer list array ("peerList"). Additionally, the "PeerList" message contains a field which indicates the chronological position ("position") at which the requesting node entered the swarm of the torrent for the tracker. This is employed in the next phases as an input in the peer selection strategy.

In the event of peers in the list can not be contacted or when the combination of pieces from all the peers can not satisfy the completeness of the data file, successive "PeerListRequest" messages are sent to the tracker. The idea is to retrieve another set of peers capable of providing the missing pieces. In this regard, the requesting peer informs the tracker of the list of missing pieces by including a piece bit vector in the message ("pieceBitVector"). In addition, the peer sends information to the tracker about the number of peers which failed to be contacted and their addresses ("numPeersFailed" and "peerListFailed"), such that the tracker avoids their inclusion in the new peer list (see Figure 5.3).

5.2.3 Handshake Phase

The Handshake phase of the TinyTorrents protocol begins when a valid "PeerList" message is received. In this phase, the protocol starts to contact sequentially each of the peers in the peer list with the goal of obtaining the list of available pieces at each of the peers. The "Handshake" message (see Figure 5.5) is employed in the communication from and to the requesting peer and serves as a mechanism to update both of the peers on their respective list of pieces. Thus the requested peer must reply with another "Handshake" message. The message contains the address of the node which issues the message ("fromId"), the torrent key ("key"), and the piece bit vector ("pieceBitVector") indicating the pieces of data stored at the peer for the given torrent key. The position of the peer in the swarm of the torrent ("position"), which was previously provided by the tracker node, is also sent in the "Handshake" message. This value is employed in the peer selection process of the next phase. In addition, a control field ("initiator") indicates whether the message has been issued by the originator of the handshake process.

```
Handshake Message
    nx_uint16_t  fromId;
    nx_uint8_t   key[KEY_SIZE];
    nx_uint16_t  pieceBitVector;
    nx_uint8_t   position;
    nx_uint8_t   initiator;
```

Fig. 5.5: TT - Handshake Message Structure

Once all handshakes are performed, the next information is stored in memory: i) a list of peers which are valid, i.e. contactable, ii) their position in the swarm of the torrent for the tracker, and iii) a list of available pieces from each peer. If some of the peers in the handshake process are not contactable after a defined number of attempts (2), they are marked as failed. In the event of none of the peers are contactable, or the list of peers can not provide all the pieces of the data file, the *Peer List Request Phase* is launched. In such a case, the addresses of the peers which previously failed to handshake are transported in the "peerListFailed" field of the PeerListRequest message (see Figure 5.3). On reception of the message the tracker avoids the inclusion of these peers in the selection of the new peer list.

5.2.4 Piece Request Phase

Once the handshake phase is completed, the consumer node starts the process of selecting which pieces are to be acquired first and from which peers. Like in the BitTorrent protocol, the rarest piece is selected; that is the piece with the least number of occurrences in the set of bitvectors of the peers in the list. Particularly in wireless sensor networks, the rarest piece selection strategy greatly improves the piece dispersion over the network, reducing the number of peers lacking of the same piece. This policy contributes to minimize the situation where the file can not be completed; seeders might be unreachable and the set of available peers could not provide the totality of pieces for the data file. Therefore, the TinyTorrents protocol calculates the rarest pieces from the set of available pieces to be retrieved and randomly selects one.

When more than one peer in the list contains the piece which is going to be acquired next, a selection algorithm decides which peer is to be contacted. Peer selection strategies are discussed in Section 5.4.2 which take into account information such as position of the peer in the swarm of the tracker, proximity of the peer, and number of pieces in the bit vector of the peer.

```
RequestPiece Message
    nx_uint16_t   fromId;
    nx_uint8_t    key[KEY_SIZE];
    nx_uint8_t    pieceIdx;
    nx_uint16_t   pieceBitVector;
```

Fig. 5.6: TT - RequestPiece Message Structure

Once a peer is chosen to provide the current rarest piece, a "RequestPiece" message (see Figure 5.6) is sent. The message carries the requesting peer address ("fromId"), and the key of the torrent ("key") for which the piece is being acquired. The identifier of the peer being requested is carried in the field "pieceId". In addition, the piece bit vector of the requesting peer is also sent in the message ("pieceBitVector") as an opportunistic mechanism to update receiving peers acquiring data for the same torrent.

A peer receiving the "RequestPiece" message replies with the data for the piece being requested by employing the "Piece" message (see Figure 5.7). The data for the piece identified as "pieceId" is transported in the "piece" array of the message. By default the piece size (MAX_PIECE_SIZE) has been set to 16 bytes. However, the piece size can be easily increased at the cost of increasing the transmission time of the message which also increases the probabilities of failure due to collisions. In addition, the "Piece" message is also employed to update the requesting peer, and other peers en route, on the piece bit vector ("pieceBitVector") of the sending peer.

```
Piece Message
    nx_uint16_t  fromId;
    nx_uint8_t   key[KEY_SIZE];
    nx_uint8_t   pieceId;
    nx_uint8_t   piece[MAX_PIECE_SIZE];
    nx_uint16_t  pieceBitVector;
```

Fig. 5.7: TT - Piece Message Structure

The TinyTorrents protocol has been designed to snoop both the "RequestPiece" and "Piece" messages en route to the destination node. A snooping peer opportunistically acquires pieces if they are on demand in its current torrent transfer process, and also updates the piece bit vector of the peers list of the torrent if appropriate. This mechanism enhances the performance of the protocol by reducing the number of messages in the communication while maintaining updated information which improves the peer selection process.

When a piece can not be provided due to the fact that the peer is not longer contactable or the peer does not contain the piece anymore - this could happen if the peer suddenly discards the torrent data -, the next peer selection process does not take into account the failing peer. If the peer does not contain the piece being requested, a "Piece" message is sent to the requesting node with the "pieceId" field set to the 0xFF flag value. In the event of all peers are failing to provide a piece, either the next rarest piece is attempted or a new peer list is retrieved from the tracker.

When receiving the "Piece" message, the peer stores the piece in the right position of the data file buffer. The mechanism keeps checking for the rarest piece for the remaining pieces which have not yet been retrieved. If the combination of the bitvectors from of all the peers in the list does not contain all the pieces of the file, the *Peer List Request Phase* is launched in order to retrieve a new peer list which can provide the remaining pieces. The node requests a new peer list from the tracker containing the missing pieces; that is indicated in the bitvector of the "PeerListRequest" message (see Figure 5.3). This mechanism keeps working until the data is completed for the torrent or a timer expires thus canceling all the current processes for the torrent and proceeding to fetch the data for the next torrent in queue. When the data file for a torrent is successfully retrieved, both the torrent and the data are stored in memory and the peer becomes a seeder. While a consumer node acts as a peer for the torrent being processed, it also acts as a seeder, and in the decentralised versions as a partial tracker, for all its stored torrents.

5.3 Architecture of Components

The TinyTorrents architecture for a sensor node, i.e. mote, is depicted in Figure 5.8. The TinyTorrents protocol is comprised of the "Peer", "Tracker" and "TinyTorrentsCore" modules which expose local interfaces for the communication amongst them, i.e. the "TorrentFetcher" and "Tracker" interfaces.

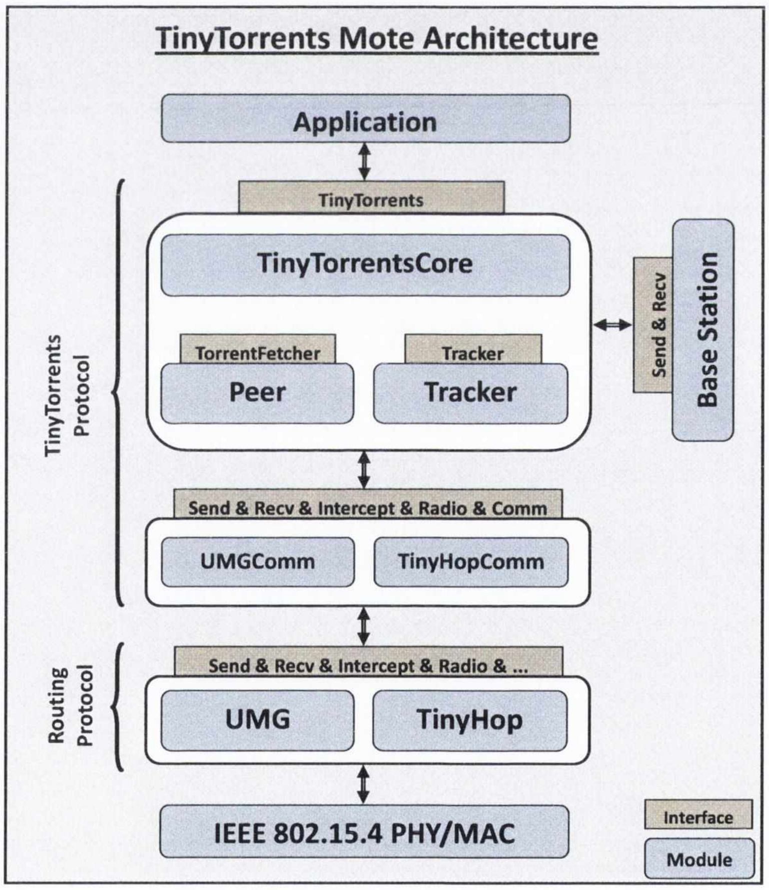


Fig. 5.8: TT - TinyTorrents Mote Architecture

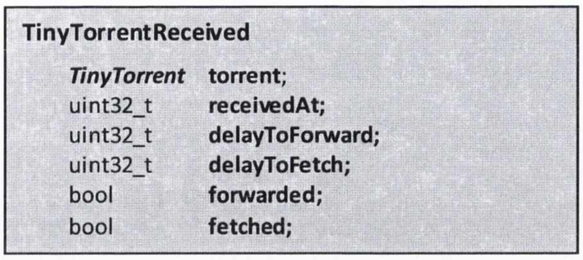
The "TinyTorrentsCore" module is the central point of coordination of the TinyTorrents protocol and provides the "TinyTorrents" interface for interaction with higher layers. The protocol utilises the "BaseStation" module to relay messages in both directions when the mote is acting as a gateway (see Section 3.2.1). The TinyTorrents protocol needs to work on top of the routing layer. Two routing protocols have been engineered to be easily interchanged according to the TinyTorrents protocol requirements. For the purpose of enabling the TinyTorrents protocol to operate on top of any routing protocol, an intermediate communications layer has been implemented which defines a set of interfaces and functionalities to be provided by the routing protocol. This layer needs to be slightly configured to connect the interfaces provided by the different routing protocols. In this regard, two intermediate layers, "UMGComm" and "TinyHopComm", have been created for each of the proposed routing protocols. This intermediate layer acts as a structure for message control, in charge of encapsulating the type of the message in the payload before it is relayed to the routing protocol. It exposes a set of common routing interfaces to the TinyTorrents protocol: "Send", "Receive", "Intercept", "Radio" and "Comm". The first three interfaces provide the basic message transfer mechanisms, the "Radio" interface is employed to initialize and disable the radio communications, and finally the "Comm" interface aggregates functionalities for the rest of interfaces provided by the routing protocol.

5.3.1 TinyTorrentsCore Module

The "TinyTorrentsCore" module is the main component of the TinyTorrents protocol which exposes the "TinyTorrents" interface for higher layers to interact with the protocol. The "TinyTorrents" interface declares the main commands and events which are employed by the application layer to: i) publish data, ii) generate torrents, iii) control the torrent dissemination process, and iv) decide if and when to fetch the data file for a torrent.

The "TinyTorrentsCore" module utilises the "TorrentFetcher" interface provided by the "Peer" module to generate local torrents and to initiate the data fetching process of a torrent. Moreover, the "TinyTorrentsCore" module manages the functional status of the "Peer" module thereby holding the responsibility to: i) retry a failed data fetching process, ii) cancel it, and iii) decide which torrent should be selected for data fetching from the queue.

The module is in charge of receiving "PublishTinyTorrent" messages (see Figure 5.2) from the network layer and stores each unique received torrent in the queue of "TinyTorrentReceived" structures (see Figure 5.9).



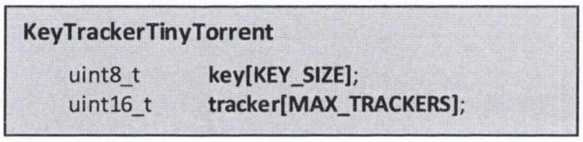
```
TinyTorrentReceived
    TinyTorrent  torrent;
    uint32_t     receivedAt;
    uint32_t     delayToForward;
    uint32_t     delayToFetch;
    bool         forwarded;
    bool         fetched;
```

Fig. 5.9: TT - TinyTorrentReceived Structure (see also Figure 5.1). Store received torrents and information about their status.

The "TinyTorrentReceived" queue stores torrents (see Figure 5.1) which are pending of being forwarded or their data fetched. Information on the local time at which the torrent is received is also stored in the field "receivedAt". When a new torrent is received, an event is triggered to inform the application layer which must decide if and when the torrent is to be forwarded and its data fetched. The application layer returns the delay in milliseconds for i) the torrent to be forwarded ("delayToForward") and ii) the fetching process of the data file to be started ("delayToFetch"). When the value is 0, the torrent is not to be forwarded nor fetched. Every time a torrent is successfully fetched or cancelled, its entry in the queue of "TinyTorrentReceived" structures is marked as fetched, i.e. "fetched" field equals to true. Furthermore, the queue is queried according to the time when the next torrent is to be forwarded or its data file fetched. The time for the next query process is calculated according to the time when the torrent was received ("receivedAt") and the corresponding delays. This mechanism guarantees that torrents are not removed from the queue until they are forwarded and its data is fetched according to the application layer decision. Torrents are selected for data fetching one at a time, with longer torrents in the queue having higher priority. Once a torrent at the top of the queue has both flag fields equal to true ("forwarded" and "fetched"), it is dequeued from the FIFO queue.

In addition to the "TinyTorrentReceived" queue, another queue has been implemented which acts as an index of the last X number of unique torrents received. The structure of each element of this queue is depicted in Figure 5.10 as "KeyTrackerTinyTorrent". It contains the key for the received torrents ("key") along with an array of the nearest tracker addresses ("tracker"). In the centralised version, the array has only one address, however in the decentralised modality up to 4 partial tracker addresses can be stored by default ("MAX_TRACKERS"). A tracker address is added to the array when a "PublishTinyTorrent" message is received. If the "tracker" array is full, the farthest address is replaced by the received partial tracker address if it is closer in terms of hops;

the newest entry is replaced first in the event of a tie. The routing layer is queried to find out the hop distance to both candidate tracker nodes. When a torrent is selected for data fetching from the "TinyTorrentReceived" queue, the tracker address is updated with the closest in the array of the "KeyTrackerTinyTorrent" queue. This is an opportunistic mechanism to discover the nearest set of partial trackers. The array of partial tracker addresses is utilised when the selected partial tracker can not provide a peer list. This way, a list of potential partial trackers is available which are contacted in a round-robin fashion until one of them provides a valid peer list. Furthermore, the "KeyTrackerTinyTorrent" queue is employed as a filter to prevent duplicate torrents to be processed, i.e. fetched and forwarded, in the situation when torrents have been already processed and dequeued from the "TinyTorrentReceived" queue. Hence, the "KeyTrackerTinyTorrent" queue should be large enough to act as a memory for the last received torrents in order to prevent delayed received torrents from being processed, particularly when there is a high number of torrents in the network.



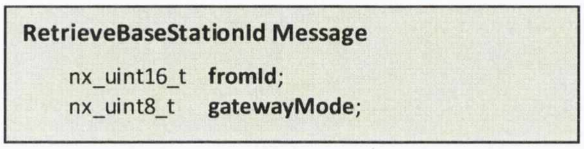
```
KeyTrackerTinyTorrent
    uint8_t     key[KEY_SIZE];
    uint16_t    tracker[MAX_TRACKERS];
```

Fig. 5.10: TT - KeyTrackerTinyTorrent Structure. Store the key and the closest set of trackers from a unique received torrent.

When a torrent is selected for data fetching, the "TinyTorrentsCore" module is in charge of instructing the "Peer" module to start the fetching process by pointing out the torrent in the "TinyTorrentReceived" queue and the address of the buffer where the data is to be stored. When the fetching of the data file is completed, both the torrent and the data are stored in memory for seeding and tracking purposes (see Tracker Module in Section 5.3.3).

5.3.1.1 Gateway Communication: BaseStation Module

The "TinyTorrentsCore" module is also responsible for the communication with the Vuze TT Plugin gateway (see Section 3.2.1) via the serial port. It employs the "BaseStation" module which acts as a bidirectional message relay component. This module implements two FIFO queues for the incoming and outgoing message communication with the gateway via the serial port. A protocol for the communication with the Vuze TT Plugin gateway is implemented here. This protocol enables the configuration of a node to act as a gateway. For a node in the WSN to become a functional gateway, the Vuze TT Plugin gateway must send a requesting connection message via the serial port. For this purpose, the "RetrieveBaseStationId" message (see Figure 5.11) is employed for the bidirectional communication. In the message, the field "fromId" is utilised to find out the address of the node which is going to act as the gateway. Additionally, the field "gatewayMode" is used to change the status of the node with respect to its functionality as a gateway. In this regard, the node can be instructed to be in the next gateway modes: "PEER", "GATEWAY" or "RELAY". By default all nodes are in "PEER" mode, which indicates the node is working autonomously with no gateway attached.



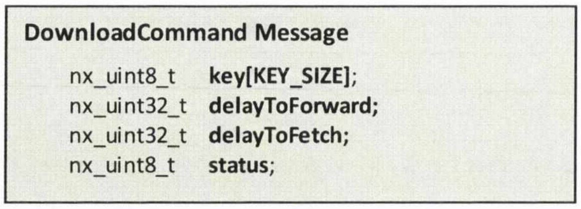
RetrieveBaseStationId Message

```
nx_uint16_t  fromId;
nx_uint8_t   gatewayMode;
```

Fig. 5.11: TT - RetrieveBaseStationId Message (Connection to the Vuze TT Plugin Gateway).

In the "RELAY" mode, the node is configured only to relay messages from the routing layer to the serial port and vice versa. The TinyTorrents protocol is also implemented at the Vuze TT Plugin in order to control the process. This mode allows the Vuze TT Plugin gateway both to push data out and into the sensor network in a transparent way. When no message activity between the Vuze TT Plugin and the node occurs for a period of time, a "RetrieveBaseStationId" message is also sent to check on the status of the connection. This mechanism changes the gateway mode in the node to "PEER" in the event of the serial communication has been incorrectly disrupted.

On the other hand, when the node is configured in "GATEWAY" mode, the "TinyTorrentsCore" module handles the control of the data fetching process which in this case is regulated by an application running in the Vuze TT Plugin. The application layer at the node passes on the control to the application at the Vuze TT Plugin to handle the distribution process. This way, "PublishTinyTorrent" messages (see Figure 5.2) are relayed to the Vuze TT Plugin when received. The application decides if and when to forward and fetch the data for the torrent. This values are sent to the gateway node in the "DownloadCommand" message (see Figure 5.12). The message contains the torrent key, the delay time to forward and the delay time to start the data fetching process. When the gateway node receives the message, a reply must be sent, employing the "DownloadCommand" message, to indicate the status of the torrent ("status"). If the torrent is stored in the queue waiting to be processed, the field "status" is set to "WAITING". When the node forwards the torrent or starts the fetching of the data file, another "DownloadCommand" message is sent to the Vuze TT Plugin to indicate the new status as "FORWARDED" or "FETCHING" respectively. In the "GATEWAY" mode, the "TinyTorrentsCore" module instructs the "Peer" module to relay messages containing pieces of data (see "Piece" message in Figure 5.7) to the Vuze TT Plugin.



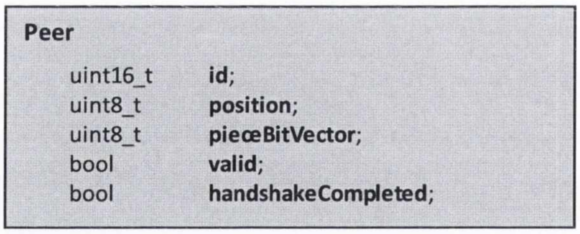
```
DownloadCommand Message
    nx_uint8_t    key[KEY_SIZE];
    nx_uint32_t   delayToForward;
    nx_uint32_t   delayToFetch;
    nx_uint8_t    status;
```

Fig. 5.12: TT - DownloadCommand Message Structure (Communication between the Vuze TT Plugin Gateway Application and the Sensor Node for Fetching and Forwarding decisions.)

In addition, when the node is in "GATEWAY" mode, the "TinyTorrentsCore" module is also in charge of receiving, sending and relaying service discovery messages coming from the Vuze TT Plugin (see Section 5.7 for Service Discovery in the TinyTorrents protocol).

5.3.2 Peer Module

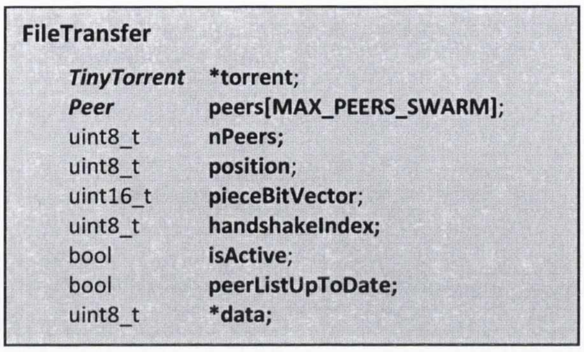
The "Peer" module handles the following phases of the TinyTorrents protocol: i) *Peer List Request Phase* ii) *Handshake Phase* and iii) *Piece Request Phase*. The phases are executed until the data file is completed or the "TinyTorrentsCore" module cancels the data fetching process. In this regard, the rarest piece selection process is implemented in this module as well as the different peer selection strategies discussed in Section 5.4.2. The "Peer" module provides the "TorrentFetcher" interface which implements a set of commands to create torrent files and to start the data fetching process of a torrent.



```
Peer
    uint16_t    id;
    uint8_t     position;
    uint8_t     pieceBitVector;
    bool        valid;
    bool        handshakeCompleted;
```

Fig. 5.13: TT - Peer Storage Structure

The "Peer" module maintains a structure which keeps the status of the active torrent data fetching process (see "FileTransfer" in Figure 5.14). The structure contains pointers to the torrent file being fetched ("*torrent") (see Figure 5.1) and to the buffer where pieces are to be stored ("*data") in the "TinyTorrentsCore" module. The list of peers received from a tracker in the "PeerList" message (see Figure 5.4) is stored in the array of "Peer" structure (see Figure 5.13). The number of peers which can be stored in the "Peer" array (MAX_PEERS_SWARM) is equal to the peer list



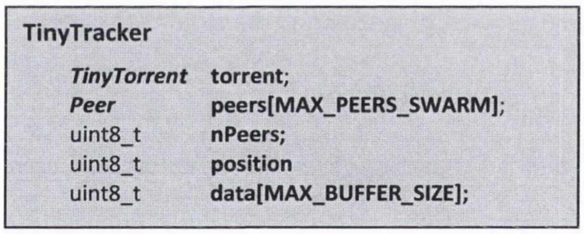
```
FileTransfer

    TinyTorrent   *torrent;
    Peer          peers[MAX_PEERS_SWARM];
    uint8_t       nPeers;
    uint8_t       position;
    uint16_t      pieceBitVector;
    uint8_t       handshakeIndex;
    bool          isActive;
    bool          peerListUpToDate;
    uint8_t       *data;
```

Fig. 5.14: TT - FileTransfer Storage Structure (see Figures 5.1 and 5.13)

size (MAX_PEERS) for the centralised version of the protocol. Nevertheless, in the decentralised version, MAX_PEERS_SWARM is greater than MAX_PEERS due to the fact that the peer can act as a partial tracker (adding requesting peers to the swarm) while the torrent data transfer is ongoing. The "Peer" structure stores the address of each peer ("id") along with the position of the peer in the swarm of the torrent ("position"). Also, the "pieceBitVector" field of each stored peer is updated when a "Handshake" message is received from the peer (see Figure 5.5). In addition, two flag variables are stored in each "Peer" structure to identify: i) if the *Handshake Phase* has been successfully completed with the peer ("handshakeCompleted") and 2) whether the entry for the peer corresponds to a valid peer list or contrary the protocol is still waiting for a "PeerList" message to be received ("valid"). In the "FileTransfer" structure, the number of peers stored in the "peers" array is kept in the field "'nPeers". This indicates whether a new peer can be added to the peer list when a non-initiated "Handshake" message is received. The position of the local peer in the swarm of the torrent at the tracker node is stored in the field "position". The record of the list of pieces stored in the data buffer is kept as a bit vector array in the "pieceBitVector" field. When the protocol is executing the *Handshake Phase*, the "handshakeIndex" field acts as a counter variable which indicates the entry in the "peers" array for which the handshake process is ongoing. Two boolean variables control the status of the transfer: i) "isActive" indicates whether the torrent transfer is active and therefore all receive messages are to be processed or otherwise discarded, ii) "peerListUpToDate" is set to false when a new peer list is being requested and therefore the current peer list in the "peers" array is invalid.

5.3.3 Tracker Module

The Tracker module has a two-fold approach. Firstly, it serves as a storage for torrents and data files when the "Peer" module completes a torrent data file. The storage of multiple torrent data files enables the possibility of having multiple torrent data transfers in the network at the same time while transforming the sensor network into an autonomous storage system where nodes might cooperate for the purpose of maximising the network capacity. Secondly, the "Tracker" module implements the mechanisms for tracking the swarm of peers of each torrent. In the decentralised version of the TinyTorrents protocol all consumers of a torrent can act as partial trackers by storing a subset of peers in the swarm of the torrent. In the centralised version a node is designated as the tracker which keeps a record of the whole swarm of peers for each torrent and acts as the central coordinator. This node does not perform any other peer functionality but only tracking duties. In the two versions, both for storage and tracker purposes, the "TinyTracker" structure is employed (see Figure 5.15) in this module. However, in the central version of the protocol, the central tracker does not perform storage of data files and thus the "TinyTracker" structure does not include the data buffer ("data"), nor the "position" field (as the tracker node does not belong to the swarm of another tracker).



```
TinyTracker
    TinyTorrent  torrent;
    Peer         peers[MAX_PEERS_SWARM];
    uint8_t      nPeers;
    uint8_t      position
    uint8_t      data[MAX_BUFFER_SIZE];
```

Fig. 5.15: TT - TinyTracker Structure: Torrent File & Data Storage. Employed for Tracking and Torrent-Data Storage Purposes in the TT Centralised and Decentralised versions (see Figures 5.1, 5.13).

The storage mechanism is implemented for all the consumer nodes in both the centralised and decentralised approaches. As soon as a torrent is either locally generated or its data successfully fetched, it is stored along with its data in the "TinyTracker" structure (see Figure 5.15). In addition, the list of peers stored in the "peers" array of the "FileTransfer" structure (see Figure 5.14) for the torrent file are also stored in the "peers" array. If the torrent is locally generated, the only peer stored is the local node. The size of the "peers" array (MAX_PEERS_SWARM) is, by default, set to 10 when a node acts as a partial tracker. However, the tracker node in the centralised version demands to record the whole swarm of peers for each torrent, and thus MAX_PEERS_SWARM is set to a high value which depends on the maximum potential consumers for a torrent and the available memory. The current number of peers in the swarm is also stored ("nPeers"). Additionally, the position of the peer in the swarm of the torrent is also stored ("position"). An array of the "TinyTracker" structure is in place which stores as many elements as memory permits with a policy of replacing the oldest entry when the array gets full.

In the centralised version of the TinyTorrents protocol, the tracker employs the "nPeers" field of the "TinyTracker" structure to indicate the position of a new peer joining the swarm of the torrent. Nonetheless, in the decentralised version, the position of a new peer joining the swarm of a partial tracker depends on the position of the local peer in its partial tracker, and the position of the other peers in its swarm. New peers entering the torrent swarm of a partial tracker are assigned the highest position from all the peers in the swarm (including the position of the local peer) plus one, - the peer position value in the swarm is employed as a weighting factor in the peer selection process (see Section 5.4) and this mechanism increases the likelihood of new peers being selected thereby fostering fairness in the distribution. It has to be noted that the *Handshake Phase* must be completed before the local peer can start acting as a partial tracker. Once the torrent data transfer process is completed and the peer becomes a seeder, the highest position is stored in the "position" field of the "TinyTracker" structure. This field indicates the position of the latest peer joining the swarm of the torrent and gets incremented when new peers join the swarm. This way, new peers are guaranteed a higher position value for peer selection purposes from the existing peers in the swarm. It has to be noted that the "position" field is limited to 255 values (1 byte). In the centralised version this value corresponds to the maximum number of consumers of a torrent before the counter rolls over and affects the selection process. However, in the decentralised version, the position value grows according to the distribution of the consumers in the network which depends on the branch of the spanning tree formed amongst the consumers where the initial seeder is the root with a position value of 1.

The "Tracker" module handles the reception of "PeerListRequest" messages (see Figure 5.3), and consequently updates the swarm of peers for the torrent being requested. In the centralised version, the module also handles the reception of the "PublishTinyTorrent" message (see Figure 5.2) which instructs the tracker node to create a swarm for the new torrent with the node as the initial seeder in position 1. In the decentralised version this process is not required since the first partial tracker for a generated torrent is the peer itself which automatically stores the torrent and data file in the "TinyTracker" structure.

In the "TinyTracker" structure, each element of the "peers" array is of the type "Peer" structure (see Figure 5.13). However, only the fields which contain the address of the peer ("id") and the position in the swarm ("position") are really employed for tracking purposes. The rest of the fields are not used and could be removed for memory efficiency but have been kept for consistency purposes thus maintaining a unique "Peer" structure for multiple purposes. Nevertheless, the "pieceBitVector" field is updated when peer list request messages are received from the peer - this happens when the previous list of peers fail to deliver all the pieces of the data file. This way, the "pieceBitVector" field is used by the tracker as an additional indicator of the necessity of recurring peers to obtain a peer list which satisfy the completeness of the data file. In this regard, updating tracker nodes regularly with the current torrent's piece bit vector at each peer gives updated information on the torrent piece status in the swarm. This greatly helps in the selection of a complete peer list for a requesting node. Nonetheless, this solution has been avoided as i) it has a great cost in terms of messages, ii) it increases the wireless medium contention level, and iii) it contributes to the risk of overflowing the message admission capacity of tracker nodes, mainly in the centralised version. On the other hand, the position of each peer in the swarm gives an estimation of the likelihood of the peer containing most of the pieces and can be used as a factor to include peers with lower positions in the peer list which have a higher likelihood of containing most of the pieces. Peer list selection strategies at tracker nodes are implemented in this module and are discussed in Section 5.4.1.

```
SwarmRemove Message
    nx_uint16_t  fromId;
    nx_uint8_t   key[KEY_SIZE];
```

Fig. 5.16: TT - SwarmRemove Message Structure

When a peer needs to replace an entry from the array of "TinyTracker" structures, or when a peer fails to fetch the data for a torrent file, the tracker is informed that the node is no longer a peer in the swarm of the torrent. The removal of a peer from the swarm of a torrent is achieved through the sending of a "SwarmRemove" message (see Figure 5.16). However, in the decentralised version of the protocol, this message is only sent to the last tracker node from which the peer list was retrieved, rather than trying to update all the connected partial trackers which contain the peer in their swarms. The lack of peer removal updates amongst partial trackers might require future extra peer list requests and handshake messages at the benefit of saving the extra messages required to keep all partial trackers updated. Additionally, the "PeerListRequest" message (see Figure 5.3) carries peers which have failed to handshake such that they are not include in the peer list.

5.3.4 Routing Communication Modules: UMGComm and TinyHopComm

An intermediate layer between the TinyTorrents protocol and the chosen routing protocol is implemented to adapt the interfaces provided by the routing layer to the generic interfaces used by the TinyTorrents protocol. This layer is in charge of multiplexing messages to the corresponding receive and send interfaces of the TinyTorrents protocol according to the type of packet. Thus, each layer is tailored for each routing protocol and acts as a single point of message control. This way messages can be intercepted and decisions on which messages are to be relayed, in both directions, can be made. For instance "Piece" messages are snooped for efficiency purposes and "PeerListRequest" messages can be intercepted when the local node is a valid partial tracker. This layer handles the communication with the routing protocol, thus it reacts to status messages coming from the routing layer and make use of the available routing functionalities. It also implements reliable mechanisms in order to make a best effort in the end-to-end delivery of each message. Reliable mechanisms such as retrials, backoff delays, or partial flooding are employed in response to the routing layer. For this purpose a queue of messages is employed to regulate the sending process. In this line, duplicate message suppression is performed and the design offers the possibility of implementing a priority management scheme in the sending process. Finally, "UMGComm" is the communications layer tailored for the use of the Ubiquitous Mobile Gradient (UMG) routing protocol (see Chapter 4), while "TinyHopComm" connects the TinyTorrents protocol layer to the TinyHop routing protocol.

5.4 Peer Selection Policies

In the TinyTorrents protocol appropriate selection of the peers from which to gather pieces of data impacts the efficiency of the traffic flow in terms of fairness, throughput, resource consumption and communications reliability. Peer selection strategies contribute to make the system scalable, to foster data dispersion, and to avoid the flash crowd problem at initial seeders.

Two peer selection processes are performed in the TinyTorrents protocol. The first one occurs at the tracker when a node requests a peer list. A tracker node is in charge of selecting a peer list from the swarm of peers for the requested torrent. The second selection process occurs at the requesting node when the peer list is received from the tracker. Here, the node selects, for every piece of data, a peer from the received peer list which can be contacted and also contains the piece to be acquired.

The combination of different peer selection strategies, both for peer list selection in a tracker node, and for peer selection at the requesting node, highly impacts the overall performance of the TinyTorrents protocol, both in its centralised and decentralised versions. A range of indicators of the status of the peers with respect to the requesting node can be utilised in the peer selection process. However, a balance between the quantity of information for each peer and its relevance in the selection process needs to be achieved. Employing too many status indicators might not enhance the selection process but rather consume valuable resources and increase the computational complexity. Accordingly, the TinyTorrents protocol employs the following indicators in the selection process: i) the chronological position of each peer in the swarm of its torrent (position-based selection), ii) the proximity of a peer in terms of hops (location-based selection), and iii) the piece bit vector of each peer (piece-based selection). This information is updated before each selection process and is opportunistically acquired and piggybacked in the messages, which does not incur an overhead in communication.

5.4.1 Peer List Selection Strategy at the Tracker

In a tracker node, both in the centralised and decentralised version, the selection of a peer list from the swarm of a torrent is mainly performed according to the position of each peer in the swarm (see "position" in the Peer Structure in Figure 5.13). In addition, the piece bit vector of every peer is only updated when successive peer list requests are received (see "pieceBitVector" in Figure 5.13). This only occurs when the peer list of the requesting peer can not satisfy the completion of all the pieces of the data file. Thus, the reception of a "PeerListRequest" message (see Figure 5.3) with a piece bit vector different from zero, indicates to the tracker that the requesting peer needs to obtain a list of peers which contain the missing pieces of data. Since piece bit vectors in a tracker node are not periodically updated, the protocol can not rely on this information to select a peer. Instead, the selection process takes into account the fact that the piece bit vector has suffered a change and it increases the likelihood of including peers with lower position values in the peer list in an attempt to select peers with higher probabilities to contain a higher number of pieces. Additionally, the "PeerListRequest" message might include a list of peers which failed to handshake (see "peerListFailed" in Figure 5.3). Failing peers are not including in the selection of the peer list when alternative peers are available. It also has to be noted that information on the proximity of each peer will not be of any use in this selection process unless peers are close to the tracker node.

Thus, the peer selection strategy utilised in the tracker is based on a stochastic process where the distribution of probabilities amongst electable peers is based on their position in the swarm of the torrent. It is a weighting scheme where the probability of a peer "i" being selected for inclusion in the peer list "$P_{selPos}(i)$" can be calculated according to Equation 5.2,

$$TW_{Pos} = \sum_{i \in S} Position(i)^{log(\alpha)} \tag{5.1}$$

$$P_{selPos}(i) = \frac{Position(i)^{log(\alpha)}}{TW_{Pos}} \tag{5.2}$$

where "Position(i)" is the position at which the peer "i" entered the swarm of the torrent in the tracker. The denominator is computed as the sum of the weighted positions "TW_{Pos}" according to Equation 5.1 where "S" is the set of peers in the swarm except from those in the "peerListFailed" array of the received "PeerListRequest" message. Additionally, in order to increase the probability of selecting peers with higher positions, i.e. recently received peers, over peers which have been longer in the swarm, a coefficient α is utilised as a weighting parameter to control the growth of the position-based probability distribution. In the situation where a peer can not complete the data file with the first requested peer list and needs to request another peer list, the tracker can lower the α coefficient to the minimum value, where α can take values from 1 to 255. A lower value of α increases the likelihood of peers with lower positions being selected under the assumption that, the lower the position in the swarm, the higher the likelihood for the peer to contain most of the pieces of data. However, when the peer list is requested for the first time for a given torrent, the value of α is set high in order to increase the probabilities of including recently joined peers in the peer list. While this is employed to achieve fairness in the data distribution process, it also contributes to include recently joined peers in the peer list which is provided to potential partial trackers in the decentralised version of the protocol. Including peers which recently joined the swarm in the peer list is of interest in the decentralised version since most of the torrent dissemination strategies would expand concentrically from the producer. This means that the likelihood of two nodes being in close proximity would increase as peers have a close value of position in the swarm. Currently, a value of 255 is assigned to α when the peer list is initially requested. Subsequent peer list requests for a give node change the value of α to 1.

5.4.2 Peer Selection Strategies

A set of different strategies are proposed for the peer selection process when a peer is in the *Piece Request Phase*, i.e. it has already received a peer list from the tracker, *Peer List Request Phase*, and has also performed the *Handshake Phase* with all, or some, of the peers in the list. In this phase, the peer has stored updated information on a set of peers from which to acquire pieces. For each contactable peer, the requesting peer contains the following information for the torrent: i) the piece bit vector and ii) the position in the swarm. While the position in the swarm and the piece bit vector of each peer are initially acquired in the *Handshake Phase*, the piece bit vector of a peer is updated with every "RequestPiece" and "Piece" messages received in the *Piece Request Phase*. Thus, the peer selection process is executed every time a new piece is to be retrieved as the current piece status of peers might have changed. This way, information about peers in the list is updated as the data fetching process progresses, which enhances the selection process in order to achieve fairness in the data distribution. Additionally, proximity to each of the contactable peers in terms of hops is known via the routing protocol; this value is up-to-date due to the fact that "Handshake" or "Piece" messages are received from the peers before the start of the peer selection process.

While the position in the swarm of a peer is a relevant factor to achieve fairness in a distributed manner and to alleviate the burden in initial seeders, the hop distance to each peer is the key factor to control the communications overhead and therefore the overall performance of the protocol. In single channel wireless sensor networks, a peer at a hop distance of 1 should always be chosen over 2 hops or more, to reduce the contention factor in the neighbourhood. However, when the peer selection process involves peers at a distance greater or equal to 2 hops, the unknown spatial distribution of the peers in their areas and the unknown contention conditions are insufficient information to assert that the closer peer in terms of hops becomes the most efficient choice. In this situation, the position in the swarm of the peer should be taken into account in the peer selection process.

With all of the above taken into consideration, a set of peer selection strategies have been proposed for comparison under different configurations of the TinyTorrents protocol.

5.4.2.1 Position-based Peer Selection

This strategy takes into account only the peer position in the swarm as the factor to calculate the probability of each peer to be elected; this is the same strategy employed at the tracker for peer list selection. The peer selection process is based on a stochastic process where high position peers have higher probability to be selected. This way, new peers entering the swarm have higher probability of being selected. This mechanism mitigates the presence of flash crowds and seeks to distribute the traffic load amongst peers in the swarm of the torrent in a fair manner.

The probability of selecting a peer "i" according to the position in the swarm "$P_{selPos}(i)$" can be calculated by following Equation 5.2. In this case, "TW_{Pos}" in Equation 5.1 considers "S" as the set of contactable peers which contain the piece being requested. The α coefficient is also utilised here to adjust the position-based growth probability distribution.

5.4.2.2 Position-Location-based Peer Selection

This strategy is analogous to the *Position-based Peer Selection* but combines both the position of the peer in the swarm and the location in terms of hops of the peer with respect to the local node in order to calculate the probability of each peer in the distribution. The probability of a peer "i" being selected according to the position (Position(i)) and distance in hops to peer "i" (HopsToPeer(i)) can be calculated according to Equation 5.4 as "$P_{selPosLoc}(i)$". In this equation, the denominator is calculated according to Equation 5.3 as "TW_{PosLoc}".

$$TW_{PosLoc} = TW_{Pos} + \sum_{i \in S} TW_{Pos}/HopsToPeer(i) \tag{5.3}$$

$$P_{selPosLoc}(i) = \frac{Position(i)^{log(\alpha)} + (TW_{Pos}/HopsToPeer(i))}{TW_{PosLoc}} \tag{5.4}$$

The equations take into account the sum of weighted positions in Equation 5.1 and increase the possible set of elementary outcomes (sample space) by a factor which is inversely proportional to the hop distance to each peer in "S", where "TW_{Pos}" is the constant of proportionality. Again, "S" is the set of contactable peers which contain the piece being requested. The α coefficient is employed as a mechanism to increase the probability of selecting peers both with high positions and low number of hops. This strategy seeks to increase the probabilities of selecting closer peers in the position-based probability distribution.

5.4.2.3 1-Hop First Position-Location-based Peer Selection

This strategy operates on the same basis as *Position-Location-based Peer Selection* when the peers in "S" are at a distance greater than 1 hop. If at least one peer is at 1 hop distance, then this is selected over the rest in a deterministic fashion. In the event of more than one peer is at a distance of 1 hop, the *Position-based Peer Selection* strategy is employed to select the peer from the subset of peers at 1 hop distance. This strategy seeks efficiency in terms of communication while balancing the network load of the swarm of the torrent.

5.4.2.4 Closest First Position-based Peer Selection

Initially, this strategy includes the closest peers in “S”, i.e. peers with the minimum number of hops from the list of contactable peers which have the piece to be requested. Then, the *Position-based Peer Selection* strategy is applied to the peers in “S”. This selection mechanism mainly seeks efficiency in terms of communication while it balances the traffic load amongst nearby peers.

5.4.2.5 Closest First Piece Remaining-based Peer Selection

This mechanism operates in the same way as the *Closest First Position-based Peer Selection* strategy but the swarm position factor is replaced by the number of pieces in the bit vector of the peer. For this purpose, the formula employed in the *Position-based Peer Selection* (see “$P_{selPos}(i)$” in Equation 5.2) is utilised which replaces “Position(i)” for “PieceRemaining(i)”. PieceRemaining(i) indicates the remaining number of pieces to be acquired by peer “i” according to its piece bit vector. The higher the number of pieces to be acquired, the higher the probabilities for the peer to be selected. This strategy mainly seeks efficiency in terms of communication while at the same time fosters the update of the piece bit vector at peers in need of pieces. Updating the piece bit vector of peers in need of pieces reduces the chances of peers requesting another peer list due to missing pieces in the current one. In addition, this strategy balances the load of traffic within the near proximities due to the fact that the piece bit vector of nearby peers is also updated at the requesting node when the piece is received; this increases the chances of nearby peers being selected to provide the remaining pieces of data.

5.4.2.6 PieceRemaining First Position-Location-based Peer Selection

Initially, this strategy includes in the set of electable peers “S” those peers with the most number of remaining pieces to be acquired, according to their piece bit vectors, from the list of contactable peers which have the piece to be requested. Then, the *Position-Location-based Peer Selection* strategy is applied to the set of peers in “S”. The strategy primarily seeks to balance the flow of traffic towards those peers with a low number of pieces in order to reduce the load on seeders. From those peers, the strategy increases the probabilities of peers which are closer and have recently join the swarm.

5.4.3 Implementation of Peer Selection Policies

The peer list selection strategy performed at the tracker is implemented at the "Tracker" module of the TinyTorrents protocol (see Figure 5.8). This strategy is employed both for the centralized and decentralized version. In the centralized version, only the tracker node inlines the strategy, while in the decentralized version every node can act as a consumer and thus implements this functionality.

On the other hand, the set of peer selection strategies are implemented at the "Peer" module of the TinyTorrents protocol (see Figure 5.8). One selection policy is enabled at each test case in the simulator for comparison purposes. When compiling the code for real devices only one strategy is inlined for memory efficiency purposes. However, strategies can be multiplexed at runtime if required both for testing or when a strategy has been shown not to perform well under certain conditions.

5.4.4 Reducing Peer Participation

The selection strategies presented above only employ information from peers with respect to a given torrent and thus they do not consider the overall status information of the device itself. For instance, a node might be lacking in resources such as energy or memory and might need to reduce its participation in the swarm of the torrents. Rather than being greedy, the node might be required to reserve its resources for situations of higher priority, for example when it acts as a key relay in the network or when it contains a rare piece of data.

For this purpose, the application layer at a node makes the decision whether to fetch a received torrent. In addition, a peer which is part of a torrent swarm should be able to reduce its participation without completely ceasing its service. For this purpose, the peer can control its participation by reducing its position value in the swarm when sending the "Handshake" message to other peers. For instance, reducing the value to 1 decreases the probability of the peer being selected when a Position-based peer selection strategy is operating. At the same time, the peer can still be selected, particularly in the cases when i) other peers are in the same situation, ii) the number of peers containing a particular piece is limited, and iii) the node is at 1 hop distance. By the same token, in strategies involving the PieceRemaining factor, a peer can reduce its participation by indicating that the peer does not store any piece of data - i.e. all bits in the pieceBitVector set to 0 - when sending messages containing the piece bit vector.

5.5 Scalability of the TinyTorrents Protocol: Centralised vs. Decentralised

In this section, the behaviour of the TinyTorrents protocol, both in its centralised and decentralised versions, is described with a sample network scenario and the factors affecting its scalability are explained.

Figure 5.17 illustrates the data distribution process in the centralised version of the TinyTorrents protocol for a torrent generated at a producer node "P". The central version of the TinyTorrents protocol relies on a central tracker to maintain the swarm of peers for all the torrent files in the network, identified as "T". Nodes interested in becoming peers, i.e. acquiring the data for the torrent, are identified as consumers "C".

Initially, the producer "P" sends a "PublishTinyTorrent" message containing the torrent (see Figure 5.2) to the tracker "T". Once the tracker is aware of the new tracker, the producer broadcasts the "PublishTinyTorrent" message to start disseminating the torrent across the network. Nodes receiving the message decide if and when to forward (broadcast) the torrent (see "delayToForward" in Section 5.3.1). Consumer nodes also decide if and when to fetch the torrent data (see "delayToFetch" in Section 5.3.1). In Figure 5.17, the arrows represent the dissemination process of the "PublishTiny-Torrent" message issued by the producer node ("P"). On reception of the message containing the torrent file, consumer nodes ("C") wait for a period of time ("delayToFetch") and then request a peer list from the tracker node ("T"). The position at which each consumer sequentially enters the swarm of the torrent is indicated as ("C#"). The peer list selection strategy at the tracker selects a peer list for each requesting consumer. One of the drawbacks of the centralised approach is the lack of a general map of the location of the consumers in the network, despite hop distance information at the routing layer. Thus, closer nodes to the consumer are not necessarily included in the peer list which is inefficient in terms of communications. For example, "C8" contacts the tracker which, by employing the position-based peer list selection strategy, replies with a random peer list including for instance peers "C7", "C6", "C5" and "C2". While the peer selection process in "C8" improves the selection of peers based on proximity and/or position, the peer list received from the tracker is already inefficient in terms of proximity, mainly when considering that peers "C3" and "C4" are closer to "C8" and were not included in the peer list. To ameliorate this problem, "C8" can request another peer list from the tracker when some of the peers exceed a defined routing scope in terms of hops. In the next peer list request, the node indicates that all the current peers as not valid (see "peerListFailed" in Figure 5.3). The tracker will produce a new peer list which does not contain

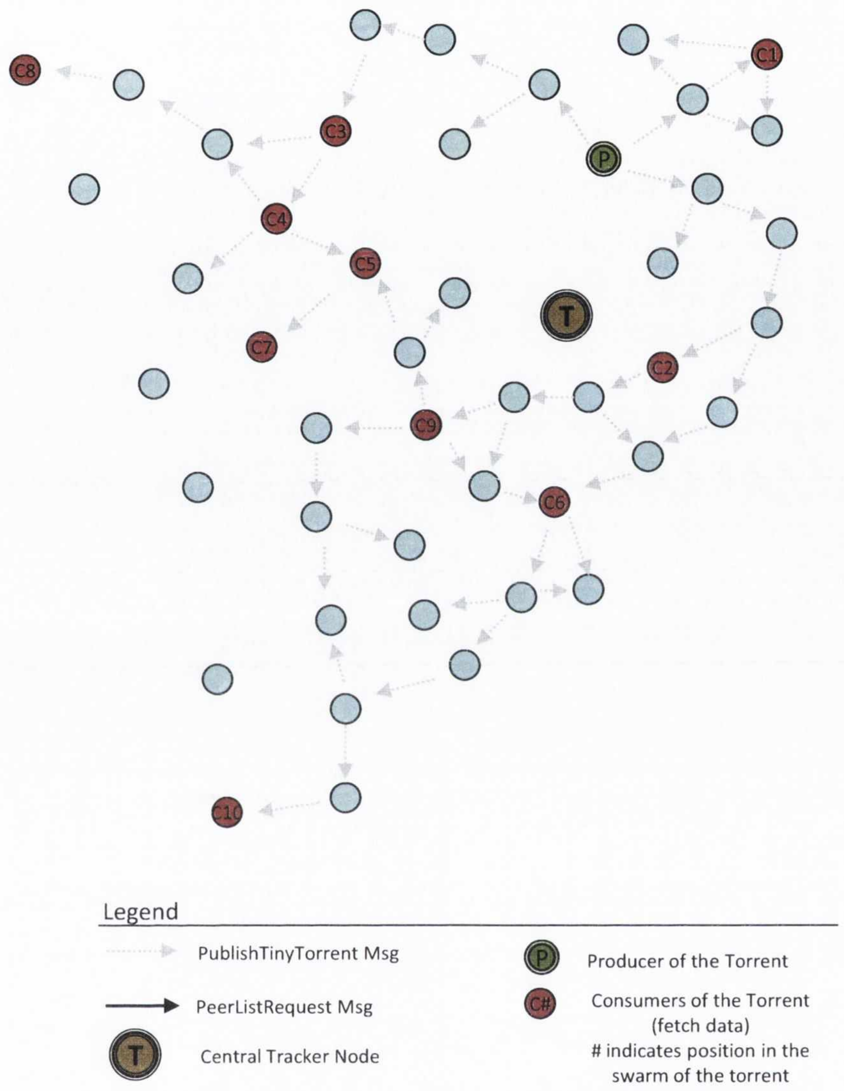


Fig. 5.17: TT - Centralised Version of the TinyTorrents Protocol

the previous peers. On reception of the new peer list, "C8" compares the old peer list with the new one in terms of proximity - information is available at the routing layer - and thus selects the closer set of peers defined by "MAX_PEERS". This mechanism improves the peer selection process in terms of proximity at the cost of an extra peer list request message. However, when the network scales and consumers are placed at a high distance from the tracker in terms of hops, the overhead in communications and the risk of packet loss greatly increases. Thus, the centralised version of the TinyTorrents protocol is targeted to small-medium size networks where the tracker node can be reached efficiently and reliably in terms of routing communications from any consumer node in the network.

The design of the decentralised version of the TinyTorrents protocol tackles some of the issues encountered in the centralised version with the use of "partial trackers". The central tracker, in charge of maintaining the swarm of peers for all the torrents in the network, is replaced by consumer nodes acting as partial trackers only for their stored torrents. A consumer acts as a partial tracker for a given torrent by keeping track of a list of peers which do not necessarily need to comprise the whole swarm of the torrent. A partial tracker discovers and updates its list of peers when i) the torrent data is being fetched, ii) peer list request messages are received, or iii) messages in the protocol are intercepted which contain the key of the torrent. Each consumer node is capable of acting as a partial tracker for nearby consumers. Thus peer lists are formed from a set of peers which are likely to be close to the requesting node. This behaviour can be seen in Figure 5.18 where consumer nodes send their "PeerListRequest" messages to nearby consumers acting as partial trackers.

In Figure 5.18, the cardinality in the consumer label ("C#") sorts consumers according to the time when they start fetching the torrent. Consumer nodes discover and select nearby partial trackers employing a set of unstructured discovery mechanisms. For instance, "C2" requests a peer list from the initial seeder "P" which only contains "C1" and itself as peer for the torrent. By the same token, "C8" selects "C5" as its partial tracker. "C5" contains in its swarm, at least, the peer list provided by "C4" which surely includes "C3", "C2", "C1" and "P', as the default size of the peer list is 4 (see "MAX_PEERS" in "PeerList" message Figure 5.4) and "C2" and "C1" are in the swarm of "P" when "C3" requests the peer list. In addition, "C5" has been acting as a partial tracker for "C7" which has also been included in its swarm of the torrent. It has to be noted that, for memory efficiency purposes, the maximum number of peers in the swarm of a partial tracker for a torrent is MAX_PEERS_SWARM which by default is set to 10. When the swarm is full and new peers are discovered for the torrent, the current policy replaces the furthest peer in terms of hops if the new peer is closer. This mechanism tends to populate the swarm of each partial tracker with closer peers. In this regard, a consumer needs to locate a nearby partial tracker to receive an efficient peer list in terms of proximity. Hence, the scalability of the system only depends on the distribution of consumers over the network, such that the highest distance between any two closest consumers is equal or less than the discovery scope of the routing protocol. For instance, in Figure 5.18, "C2" is within the routing scope of its partial tracker, i.e. "P". However, "C10" is an isolated consumer

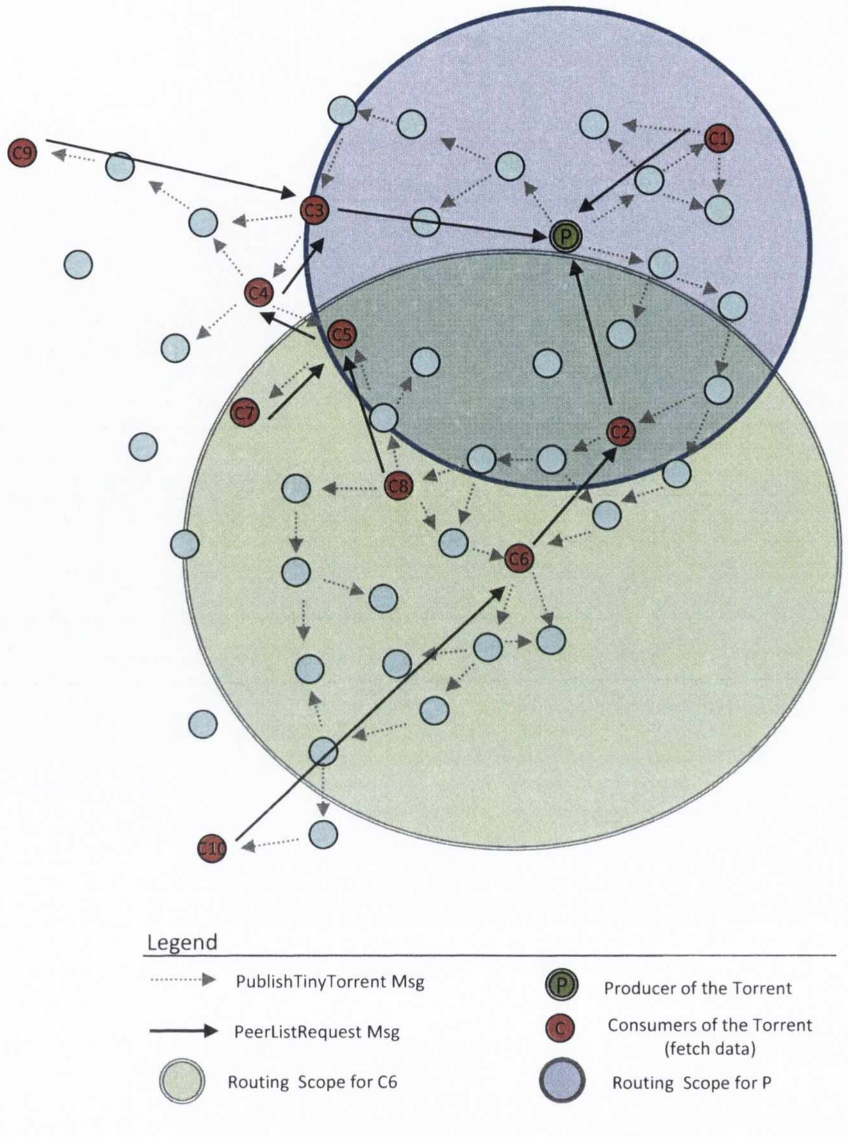


Fig. 5.18: TT - Decentralised Version of the TinyTorrents Protocol

due to the fact that the closer partial tracker is "C6" which is not within its routing scope; in other words, "C10" does not contain routing information about "C6" in its routing table. Therefore, the successful discovery of a nearby partial tracker is the key mechanism for the efficient operation of the decentralised version of the TinyTorrents protocol. In this regard, a set of mechanisms to discover partial trackers are presented in Section 5.6.

5.6 Unstructured Discovery of Partial Trackers

A mechanism to discover partial trackers, i.e. consumer nodes which can provide a peer list for a torrent, is required for the scalable and efficient operation of the TinyTorrents protocol in its decentralised version. For this purpose, unstructured discovery mechanisms have been developed which, as opposed to structured mechanisms, do not maintain an overlay network. Thus, unstructured mechanisms are more suited for discovery in unreliable wireless networks where nodes might be intermittently available and exhibit transient behaviour.

Two network environments need to be considered when discovering partial trackers: 1) "High Distribution Activity Environments" in which the data distribution process is ongoing thus the torrent is being disseminated and the fetching of the data occurs within a short interval of time since the torrent is generated at the producer and received, and 2) "Low Distribution Activity Environments" in which there is little data distribution activity occurring for the torrent, either due to a high delay in the fetching of the received torrent, or due to the acquisition of the torrent at a later stage, e.g. a gateway node arriving to the area/network queries the WSN for a torrent generated and distributed in the past (see Service Discovery in TinyTorrents in Section 5.7).

In "High Distribution Activity Environments", "PublishTinyTorrent" messages (see Figure 5.2) are being disseminated across the network. The data fetching process is occurring within a short enough interval of time to consider the routing scope of the consumer to be static from the point of view of the TinyTorrents protocol. This means that nodes can still be contacted and there is a high probability that data files, or pieces of data, are still stored in the memory of most of the consumers. However, in "Low Distribution Activity Environments", the data distribution process for the torrent is not ongoing and thus the likelihood of the status of the peers in the swarm having changed is higher than in "High Distribution Activity Environments". Peers' geographical location could have changed and the data file might have been removed from memory to accommodate data for other torrents. In this regard, peers which remove a data file from memory are required to inform their tracker. The "SwarmRemove" message (see Figure 5.16 in Section 5.3.3) is employed for this matter. In the central version of the TinyTorrents protocol the update of the swarm of a torrent is centralised and the tracker keeps records of peers joining and leaving the swarm. However, the decentralised version of the protocol requires to update all partial trackers which contain the peer to be removed in their swarm. This requires a distributed update where each consumer updates its partial tracker in a process which navigates the tree of relevant consumers. Nonetheless, this mechanism has an overhead in communication and complexity, and depends on intermediate consumers to still be contactable in order to reach all the relevant partial trackers. Therefore, the TinyTorrents protocol does not perform the distributed update and instead relies on the peer list request mechanism for obtaining different peer lists if nodes can not be contacted, i.e. handshakes fail (see Section 5.2.2). In this regard, each partial tracker can keep a counter to control the number of times the peer has been reported as failed (see "peerListFailed" in Section 5.2.2)) and remove it from the swarm once a threshold is reached. However, the latter approach has not been implemented as it constitutes a potential flaw - for instance a set of consumers can report a failed peer as the handshake was not successful, as it was out of the routing scope, but the peer still contains the torrent. Additionally, more than one request of a peer list for the same torrent is not guaranteed to happen if some peers can provide all the pieces of data.

Thus, the discovery of partial trackers is paramount in the eventual acquisition of a list of peers which can provide the data file. For this purpose, a set of unstructured discovery mechanisms for the location of nearby partial trackers both in "High Distribution Activity Environments" and "Low Distribution Activity Environments" have been designed and are presented next. These mechanisms are applied in the order presented such that the next is employed if the opportunity arises or when the previous mechanism can not discover a partial tracker.

1. Torrent Dissemination Control: This discovery mechanism can be classified as One-Shot Push-Pull according to Section 2.4.4, where initial seeders push data to the network in a way that can optimise the query/searching process of requesting consumers. It employs the delay variable at the application layer ("delayToForward" see Section 5.3.1) to regulate the dissemination process of the "PublishTinyTorrent" message. The application layer controls the flooding process of the torrent by deciding at each receiving node if and when to forward the message. In this sense, the first "PublishTinyTorrent" message received is forwarded only when the delay time ("delayToForward") is elapsed, and thus a reception window is opened for the admission of further messages of the same torrent type. This way, the reception of duplicate torrent messages is leveraged to update the consumer on the set of closer tracker addresses as messages come from spatially separated consumers. Additionally, consumer nodes assign a smaller value of delay as compared to the rest of the nodes, which increments the likelihood of potential consumers receiving a higher number of instances of the message coming from disparate partial trackers.

   Currently, the "delayToForward" value at the application layer is required to be higher than 70 ms and the TinyTorrents protocol layer automatically sets this value to 30 ms when the node is a consumer. The node is a consumer when the "delayToFetch" value is not 0. On reception of "PublishTinyTorrent" messages, a consumer node checks the queue of "KeyTrackerTiny-Torrent" structure for a matching torrent key (see Figure 5.10). If found, the array of partial tracker addresses (see "tracker[MAX_TRACKERS"]) is searched for the received tracker address. If the address does not exists, then it is added. If the array of partial tracker addresses is full, then the farthest address is replaced by the received partial tracker address if it is closer in terms of hops; the newest entry is replaced first in the event of a tie. The routing protocol provides information on the hop distance to the tracker address if this is within the routing scope. When the torrent is selected to be fetched or forwarded, the address of the closest tracker in the array is employed as the partial tracker. However, the closest partial tracker can be a consumer which is due to start the fetching process at a later stage and thus it can not provide a peer list yet. This is the reason why the "KeyTrackerTinyTorrent" structure stores an array of potential partial tracker addresses. The node contacts the addresses of the "tracker" array in a round-robin fashion until one of them provides a valid peer list. A delay (by default 2 seconds) is introduced in between requests to i) wait for these nodes to become partial trackers, and ii) avoid overloading the network.

Thus, the dissemination process of the "PublishTinyTorrent" message is utilised as a mechanism to discover nearby partial trackers.

2. "PeerListRequest" Message Interception: This mechanism enhances the previous mechanism by intercepting the "PeerListRequest" message (see Figure 5.3) en route to the previously discovered partial tracker. Every relay node in the route towards the partial tracker checks whether it can act as a tracker for the requested torrent key. In this case, if the node can provide a peer list for the torrent, then it replies with a "PeerList" message (see Figure 5.4) and becomes the new partial tracker of the requesting node. The intermediate layer of communication with the routing protocol (see Section 5.3.4) is in charge of intercepting the message received from the routing layer. This layer takes appropriate actions to acknowledge the "PeerListRequest" message sent at the requesting node when a "PeerList" message is received from a different address.

3. Initial Seeder Contact: When the appointed partial tracker can not be contacted, this mechanism leverages the information contained in the key of the torrent (i.e. the generator of the torrent) to request the peer list from the initial seeder. However, this mechanism relies on the *"PeerListRequest" Message Interception* mechanism to intercept the message and locate a nearby partial tracker. If the initial seeder can not be contacted, as the routing protocol does not have an entry in its routing table, then the mechanism can not be employed.

4. Routing Discovery Scope Increment: This mechanism is employed when neither the appointed partial tracker nor the initial seeder (*Initial Seeder Contact*) can be contacted, either because the end-to-end routing connectivity fails, or because the node is out of the routing discovery scope (i.e. its address is not in the routing table). In this situation, the TinyTorrents protocol instructs the routing layer, through the intermediate routing communications layer (see Section 5.3.4), to increase its default discovery scope while searching for the partial tracker node. The default routing discovery scope is incremented by a factor of X hops to try to reach the node. Currently, the scope is increased by 3 hops. When and if a partial tracker is discovered, the "PeerListRequest" message is sent towards the node.

5. UMG Service Discovery: When neither the appointed partial tracker nor the initial seeder can be reached and the *Routing Discovery Scope Increment* mechanism fails or is not employed, then the TinyTorrents protocol make use of the Ubiquitous Mobile Gradient (UMG) routing protocol service advertisement and discovery functionality (see Chapter 4, Section 4.4) in the search for potential partial trackers. Two mechanisms are available:

   - Local Discovery: This approach exploits local information in the routing table to provide a potential target node in the peer list search process. Rather than using random walks algorithms, this mechanism follows the route towards a potential consumer of the torrent while at the same time leveraging the benefits of the *"PeerListRequest" Message Interception* mechanism. For the location of potential trackers, the Ubiquitous Mobile Gradient (UMG) routing protocol provides a service advertisement mechanism which spreads a description of the general interest of each consumer node. In other words, a description of the type of torrents which each consumer is interested in acquiring/storing is initially spread to the nodes within the gradient spreading scope of each consumer. The description includes human-readable tags which are defined by the application layer. Thus, a given node contains in its routing table a list of contactable nodes and their associated descriptors, which serve as a list of potential candidates when searching for a peer list of a particular type of torrent. Currently up to a maximum of three nodes from the routing table are selected to be contacted which best match the description of the torrent; more than three nodes can be considered to increase the success in the discovery, however a communications overhead and delay is introduced when contacting each node.

- Network Discovery: When the *Local Discovery* mechanism can not find potential trackers within the local routing table, the Ubiquitous Mobile Gradient (UMG) routing protocol leverages the gradient spreading process to discover nearby nodes with a similar service description. Nodes receiving the gradient packet spread their own gradients if their service descriptor matches the descriptor of the torrent in the packet within a certain degree of accuracy in the comparison of the descriptors; the accuracy is also transported in the packet (see Sections 4.4 and 5.7). Eventually, the requesting node receives gradient updates from nodes matching the request, thereby generating a list of potential consumers of the torrent for the *Local Discovery* mechanism to take place. The scope of the gradient is incremented when the discovery is not successful up to a maximum number of hops (the default scope is 5 with an increment of 3). This approach operates as a descriptor-based multicast *Routing Discovery Scope Increment* mechanism.

6. TinyTorrents Service Discovery: While the *UMG Service Discovery* mechanisms are provided by the UMG routing layer to advertise and discover consumer nodes, the TinyTorrents Service Discovery mechanism employs descriptor-based query messages at the TinyTorrents layer to search for torrent files in the network. This mechanism, which is described in Section 5.7, is employed when a node is searching for a specific torrent or a description of a torrent which has not been received. For instance, this could have happened due to the node not being in the network at the time of the torrent message dissemination, or simply due to the torrent message not being successfully received. The mechanism is independent of the routing layer and thus it can operate with TinyHop or any other routing protocol. In the TinyTorrents centralised version, the mechanism targets only the central tracker node to discover torrents. However, in the decentralised version, the mechanism is employed to target specific nodes, or to query a scope of the network, in search for torrents matching a description. Thus, the mechanism is also employed as a way to locate partial trackers, i.e. consumers of a specific node description, mainly in "Low Distribution Activity Environments".

The combination of the above mechanisms seeks to increase the likelihood of finding a peer list for a given torrent in an unstructured manner. In this regard, some of the mechanisms can be avoided, prioritized, or their configuration parameters changed, e.g. the routing discovery scope, the delay time to forward, or the maximum nodes to contact, such that a balance between communication costs and peer list discovery success is achieved.

5.7 Service Discovery in the TinyTorrents Protocol

Service discovery and advertisement mechanisms have been introduced in the TinyTorrents protocol as an enhancement which allows for the localization of nodes providing services. In the TinyTorrents context, a service is defined as a torrent, a data file, or a torrent peer list, which is provided by a sensor node for an interval of time. A node could advertise a combined description of its stored torrents or the type of torrents it is interested in acquiring according to their meta-tag descriptors. In this regard, a sensor node should be able to describe itself according to its services. In addition, a mechanism to disseminate service descriptors needs to be in place which allows for an efficient service discovery in sensor networks.

For the purposes of describing data in memory constrained sensor devices employing human-readable tags, a space-efficient structure which allows for the storage of multiple combination of tags has been implemented, i.e. a Bloom filter. A Bloom filter is a probabilistic data structure in the form of a bit vector which stores the membership of an element by setting strategic bit positions to 1 (see Section 4.4.1 for more detail on the Bloom filter).

Currently the TinyTorrents protocol defines 32-bit Bloom filters as descriptors ("descriptorBF") employing 4 hash functions which produce 4 strategic positions in the bit vector for each stored item. False positives can occur when querying a Bloom filter; this occurs when positions which have been set to 1 match a combination for an item which has not been previously stored. False positives could be avoided in advance by checking for possible collisions amongst combinations of the range of elements to be stored, i.e. vocabulary of tags. Usually a maximum of 4 tags is sufficient for a descriptor in a torrent to characterise a data file in a sensor network. According to Equation 4.7 in Section 4.4.1, the probability of a false positive in a Bloom filter of 32 bits in size, employing 4 hash functions and inserting 4 elements is equal to 0.025. This probability of false positives is acceptable for the purpose of service discovery in TinyTorrents while false negatives are not given. However, different combinations of tags can be tested beforehand to minimize the presence of false positives.

Service advertisement is realised in TinyTorrents by encapsulating a Bloom filter descriptor in the torrent file which is transported in the "PublishTinyTorrent" message (see "descriptorBF" in Figures 5.1 and 5.2). The Bloom filter descriptor is employed to describe the data file associated with the torrent. Service discovery is achieved by querying the list of torrents stored in the local node or in other nodes in the network. The query process searches torrents matching a given descriptor and a period of time when they were created. For network querying purposes, a special message, the "Query" message, has been defined (see Figure 5.19).

```
Query Message
    nx_uint16_t   fromId;
    nx_uint16_t   destinationId;
    nx_uint8_t    hopsToLive;
    nx_uint8_t    descriptorBF [BF_SIZE];
    nx_uint8_t    accuracyBF;
    nx_uint32_t   currentTimeAtSender;
    nx_uint32_t   timeCreatedFrom;
    nx_uint32_t   timeCreatedTo;
```

Fig. 5.19: TT - Query Message Structure for the Torrent File Service Discovery

The "Query" message contains the torrent descriptor ("descriptorBF") which is to be retrieved. When the "Query" message is received, the node search its list of stored torrents and returns those matching the description in the "descriptorBF" field (see Figure 5.19) with a degree of tolerance indicated by the "accuracyBF" field. The calculation of the similarity between Bloom filter descriptors is performed at bit level, such that the query descriptor needs to match the queried descriptor bit vector with a percentage of bits set to 1 equal or greater than the "accuracyBF" value. The queried descriptor could contain a higher number of bits set to 1 and match the desired "accuracyBF" percentage. This indicates that the torrent is described with a higher number of tags, which in principle should not be a drawback in the selection of the torrent as a candidate. The query system also checks the torrent creation time which needs to fall within the period of time defined by the "timeCreatedFrom" and "timeCreatedTo" fields (both in milliseconds). Like in the *Publish Phase* of the TinyTorrents protocol (see Section 5.2.1), the comparison between the time at the sender and the receiver is not synchronized and thus the local time at the sender is transported in the "Query" message ("currentTimeAtSender") to calculate the relative time against the local time in the receiver. This is a soft real-time synchronization mechanism which introduces a degree of inaccuracy in the time calculation which increases as the number of hops in the query process gets higher; a synchronization accuracy of less than 500 ms was achieved in a network of 30 motes with distances of up to 4 hops, however further testing is required as this was out of the scope of this thesis.

The "Query" message is sent to a node providing it is contactable. For the location of potential nodes which can provide torrents matching a particular description, the Ubiquitous Mobile Gradient (UMG) routing protocol (see Chapter 4) advertisement and discovery mechanism is employed which is also based on Bloom filter descriptors (see Section 4.4). UMG enables the application layer to describe the type of services the local node provides or is interested in, and leverages the gradient creation process to advertise them. This way, the routing table of the UMG routing protocol provides a list of service descriptors which can be queried to find out potential consumers with a high likelihood of containing the torrent descriptor. This will narrow the query space while providing information on the proximity of the potential consumer.

When the TinyTorrents protocol operates on top of other routing protocols, like TinyHop, the query process can be achieved by querying nearby neighbourhoods, i.e. by flooding the neighbourhoods X hops away with the "Query" packet. The flooding process of the "Query" message is limited in scope with the "hopsToLive" field of the packet (see Figure 5.19). The "hopsToLive" field acts as a decreasing counter which when reaching 0 indicates to the receiving node not to forward the query packet. While expanding ring strategies can be implemented to keep on increasing the scope of the flooding process when the search is not successful, they increase the communication overhead. When a torrent discovery process fails after contacting a set of potential consumers, the TinyTorrents protocol assumes the torrent is either too farther away or it does not exist. In this case, it will be inefficient in terms of communication to try to acquire the torrent. Depending on the nature of the application, the communications overhead can be accepted. Nevertheless, if a mobile gateway is being employed for query purposes, its geographical relocation will be another option to consider.

On reception of a "Query" message, nodes reply with each matching torrent file. For this purpose, the "QueryResponse" message is employed which has the same structure as the "PublishTinyTorrent" message (see Figure 5.2). The "accuracyBF" field and the available list of matching torrents at each node determines the number of torrent replies. Once matching torrents are received, the data fetching process can begin.

The torrent discovery mechanism can be employed by any node in the network both at the application layer and at the Vuze plugin gateway via the "TinyTorrentsCore" module. Consequently, the wireless sensor network can also acts as a distributed data storage.

5.8 Summary

This chapter has presented the TinyTorrents protocol, a selective data dissemination protocol for cooperative P2P data distribution in wireless sensor networks. The protocol employs existing P2P content distribution concepts, specifically from the BitTorrent protocol, to tackle the problem of providing a fair and selective data dissemination capability in scalable multihop wireless sensor networks.

A detailed description of the operation of the protocol has been provided through detailed explanation of each of the phases from a functional and architectural perspective. The TinyTorrents protocol has been presented as a data distribution layer offering two modes of operation: i) partially centralised and ii) decentralised.

In the centralised version, the benefits of having a central point for data distribution management are diminished by the reduced scalability of the protocol and the lack of information about the localization of peers. Having a central node which tracks the identity of the peers involved in the distribution process is of great benefit when aiming to balance the network traffic of a relatively small sensor network. However, a central point of failure diminishes the fault tolerance of the system and poses a security risk, mainly in wireless networks of unreliable constrained devices. As a consequence of this, the decentralised version of the TinyTorrents protocol has been designed and presented as a fault-tolerance solution which eliminates the unique central tracker and makes the system scalable.

The decentralised version of the TinyTorrents protocol employs a set of unstructured discovery mechanisms for efficient location of partial trackers capable of providing a list of peers from which to download the data file. Different peer selection strategies have been explored for the retrieval of a list of peers from tracker nodes and for the selection of peers from which pieces of the data file are acquired. The mechanisms operate at the TinyTorrents layer and leverage UMG routing information and functionality. In addition, a service/data discovery mechanism is provided for the efficient querying of data files based on human-readable descriptors.

To conclude, the decentralised version of the TinyTorrents protocol presented in this chapter offers a data distribution system for sensor networks with the following properties: i) a high degree of fault tolerance and robustness, ii) efficient scalability in terms of communication (which depends on the consumer distribution), iii) a distributed self-regulated mechanism which seeks to balance data traffic distribution in the network and foster fairness, iv) a set of unstructured discovery mechanisms which, when combined, provide an efficient and effective solution for the scalable data distribution process and enable the use of the WSN as a distributed data storage medium.

Chapter 6

Evaluation

This chapter studies the performance of the protocols presented in this thesis. Initially, the operating system (TinyOS), employed for the implementation and evaluation of the protocols, is described, as well as the simulator (TOSSIM). Next, the specifications of two wireless sensor platforms employed in the evaluation are provided. The experimental settings and metrics employed in the performance analysis of the protocols are then presented. These initial sections provide information on understanding the evaluation environment, before progressing to the performance evaluation.

The following section focuses on the evaluation of the TinyTorrents protocol operating with the UMG routing protocol. The TinyTorrents architecture is evaluated in its centralised and decentralised modalities under the set of peer selection policies proposed in scalable scenarios of up to 400 nodes. The behaviour of the system under different conditions, such as torrent dissemination policies or number of consumers and producers, is also studied. At the end of the section, the performance of the decentralised TinyTorrents protocol is compared against two state-of-the-art dissemination protocols, DIP and DHV, and is also evaluated in a real-world testbed comprised of 64 motes. Finally, a summary of the chapter is provided which highlights the weaknesses and strengths of the operation of the protocols and their contribution to the sensor networks data dissemination process.

6.1 The TinyOS Operating System for WSN

The set of protocols presented in this thesis have been implemented and evaluated in TinyOS v.2.1.1 [96], the latest version of the most commonly used operating system for WSN. TinyOS 2.x. is a clean redesign and re-implementation of version 1.x, which overcomes some fundamental limitations of its previous version. The stability of the system and its usability have been improved with: i) a three layer hardware abstraction hierarchy to decouple components, ii) higher booting and initialization control, and iii) an optimised scheduler which only allocates one instance of a task at a time in the FIFO queue.

From an architectural point of view, TinyOS is an event-driven, component-based framework: independent components are linked together to build the final application to be loaded into the mote, and higher level components communicate with lower level ones by issuing "commands" and waiting for "events" to be signalled. Commands and events are decoupled and non-blocking, so that the command returns immediately and the response is signalled afterwards. This behaviour is usually defined as "split-phase": the invocation (call) and the completion (signal) of an operation are distinct with two different execution times. Long term operations, or tasks, form a third abstract computational unit and are scheduled on a non-preemptive FIFO basis. A task, which always runs to completion, can be posted to the scheduler queue by a command, event, or another task, where a task can also post itself. This design allows for very long computations to be split into multiple tasks that will post themselves.

The programming language employed to build TinyOS is known as "nesC" [219] and is a contraction of the "C" language which has been tailored and optimised for devices with low power and constrained resources. NesC is also an architecture description language which employs "configuration" components to interconnect interfaces. Additionally, "module" components provide, use and implement the functionality of these interfaces. Application layer components are compiled and statically linked with lower layer components in a process called "wiring" which produces a monolithic image of the required components in the system. The binary image is then loaded into the motes.

TinyOS applications are characterised by a small memory footprint, both in RAM and ROM, which varies according to the functionality and the number of components required. TinyOS provides support for a list of sensor network platforms in a modular way where low-level components implement the functionality for each chip. For instance, the micaZ [16] and telosB [30] wireless sensor platforms integrate the same transceiver but different microcontrollers and flash memory chips.

6.2 The TOSSIM Simulator

TOSSIM [105–107], now in version 2.1.1, is an open source system simulation environment included in the TinyOS suite [96]. It can capture a wide range of network interactions, and is capable of scaling up to over one thousand nodes by simulating using a fine-grained, bit-level granularity [105]. TOSSIM is a discrete event simulator, such that the operation of the system is driven by the execution of chronologically ordered events which react to, and change, the state of the system [220]. For instance, the sending of a packet is represented as an event in the events queue which occurs at a specific instant in time and lasts the transmission time of the packet. The simulator calculates the reception time of the message according to the sending of the packet and creates an event in the queue which is pulled and executed right after the previous event has been processed. Events are ordered by the time when they occur and how long they take and are executed instantaneously. In this sense, TOSSIM is a simulator, not an emulator. However, it does not capture the behaviour of the mote at the instruction level. Thus, if an application is CPU intensive, it is very hard to capture the exact amount of time the microcontroller spends in active state.

TOSSIM is integrated in the TinyOS suite as a library which enables the user to control and configure the simulation environment, such as topology, noise floor, radio and MAC models. It makes use of most of the code in the stack of a TinyOS application to generate the executable and replaces low level chip components with an implementation of their behaviour. Currently, TOSSIM supports simulation of the main chips integrated in the micaZ sensor platform (see Section 6.3.2), the Atmel ATmega 128L microprocessor [221] and the Texas Instruments CC2420 Chipcon Transceiver [29]. The CC2420 transceiver is also integrated in other platforms such as the telosB mote (see Section 6.3.1) and thus simulations at network level in TOSSIM can also be employed to test the networking activity of these type of sensor devices. Since TOSSIM maps directly to the TinyOS code, compiling an application for the simulation environment is as simple as compiling it for the real mote. This approach reduces the gap between the simulator and the real environment. By replacing low level components, a high level of fidelity between the functionality of the system in the simulator and in the real device is achieved. This also enables testing the codebase of an application both at node and network levels before testbed deployment. For this purpose, TOSSIM offers a debugging output system (dbg) which allows printing the state variables of each node at execution time. In addition, GDB, the GNU Project debugger, can be used to debug the code.

Good scalability is a natural consequence of the TinyOS design: mote based applications are usually small in size and each internal component has its own private and static frame, thus simplifying the simulation overhead and allowing simulation of hundreds of nodes in memory. Levis et al. [106] tested the effects of scalability on TOSSIM v.1 with respect to activity on each of the nodes and the number of nodes in the system. A simulation comprising 8192 motes over 12.5 virtual seconds took about 2.75 hours, with the overhead mainly due to the simulation of radio model and the MAC scheme. However, depending on the radio communication activity and the complexity of the protocols, the number of nodes has a great impact on the simulation execution time.

The MAC object in TOSSIM controls variables such as backoff, packet preamble length, radio bandwidth, etc., and by default is configured according to the CC2420 transceiver chip operation. On the other hand, the radio propagation model in TOSSIM is based on the propagation strength of each node and the noise floor trace introduced by the user. Other parameters such as the receiver sensitivity are configured according to the CC2420 transceiver specification. The radio propagation model employs a Signal to Noise Ratio (SNR) curve derived from experiments with two micaZ motes. Additionally, noise floor and interference from the environment and other nodes are included in the model via the use of the Closest-fit Pattern Matching (CPM) algorithm. As described by Lee, Cerpa, and Levis [107], "the CPM model significantly increases the accuracy of the simulation in terms of packet delivery by acknowledging the time-dependence of wireless noise". "This model can capture bursts of interference and other correlated phenomena". The CPM model exploits the non-linear behaviour of the relation between the packet reception ratio (PRR) and the signal to noise ratio (SNR). The user is responsible to create the topology in a connectivity file where unidirectional gain values between sender and receiver nodes in the network are assigned. Each gain value indicates the signal strength in dBm at which the destination receives the signal. Additionally, CPM generates a statistical model from the noise floor trace file selected by the user. Three noise floor files are available in TOSSIM which have been obtained from different environments. In Table 6.1, the average and standard deviation of a subset of 5000 samples from the noise trace files are included, which provides an idea of the background noise in the network. Depending on the path attenuation between two nodes, indicated in the topology file, the SNR can be calculated from the noise floor, and according to the SNR/PRR curve, the packet drop rate is obtained. In Figure 6.1, the "Casino-lab" noise floor has been obtained from measurement at a laboratory in the Colorado Schools of Mines, which produces a lower level of noise than the very noisy environment in the Meyer library from Stanford, "Meyer-heavy". These files are employed to test and compare networking protocols operating on a specific topology under different levels of attenuation.

Table 6.1: Statistical Comparison of Noise Floor (dBm) Traces in TOSSIM for 5000 samples.

Noise Trace	Casino-lab	TTX4-DemoNoiseTrace	Meyer-heavy
AVG	-97.69	-95.55	-93.21
STDV	1.34	2.89	8

6.3 Wireless Sensor Platforms

6.3.1 TelosB

The telosB mote (TPR2420CA) [30] (see Figure 6.1) is a wireless sensor network platform developed by the University of California, Berkeley, which can be programmed with the TinyOS open-source operating system. While it was initially commercialized by Crossbow Technologies Inc., currently is being provided by Memsic Corporation. The telosB platform has been employed in testbed experiments presented in this thesis.

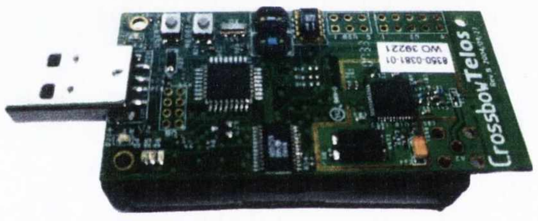


Fig. 6.1: TelosB Mote

The device integrates the ultra-low power Texas Instruments 16-bit MSP430 microcontroller with an 8 MHz CPU. It comes equipped with a 10 KB RAM module, 48 KB programming memory and a 1 MB external flash memory for data logging. The current consumption of the microcontroller in Active mode is of 1.8 mA, while only 5.1 μA in Sleep mode [30].

It incorporates the Texas Instruments 2.4 GHz CC2420 Chipcon Transceiver [29], which is IEEE 802.15.4 [5] compliant. The CC2420 transceiver transmits in the ISM 2.4 GHz band (2400 - 2483.5 MHz) with a transmission data rate of 250 Kbps. The IEEE 802.15.4 specifies 16 channels within the 2.4 GHz band, in 5 MHz steps, numbered 11 through 26. For experimental purposes, single channel communication has been employed where channel 26 has been selected due to the low degree of overlap with IEEE 802.11 channels. The device integrates an on-board antenna capable of reaching up to 100 meters outdoors and 30 meters indoors. The RF output power of the CC2420 transceiver is programmable, which not only enables dynamic transmission power control but also facilitates the testbed deployment for multi-hop testing purposes. The output power register in the CC2420 transceiver varies from level 3 at -25 dBm and a consumption of 8.5 mA, to level 31 at 0 dBm with a consumption of 17.4 mA. Accordingly the transmission range also varies which also depends on the environment where the mote is placed. On the other hand, when the transceiver is in receiving mode, the current drawn is 23 mA. Therefore, it can be observed that the most expensive activity in terms of power in the transceiver, and also in the whole architecture of the device, is the reception of a message (RX mode). In contrast to the transmission mode (TX mode) which is only enabled for the time it takes to transmit the message, the reception mode needs to be active not only for receiving a packet but also while listening. This is the main reason why most of the efforts to save energy in wireless sensor devices focus on the design of low power listening policies which duty cycle the transceiver from RX mode to Sleep mode with a current consumption of 1 μA.

The mote comes with a USB port for programming purposes and gateway connectivity employing the serial RS-232 standard. In addition, the telosB integrates a set of sensors: visible light, visible to IR light, humidity and temperature sensors. The device is powered by a set of 2 AA batteries.

6.3.2 MicaZ

The micaZ [16] mote (see Figure 6.2), like the telosB, was originally developed by the University of California, Berkeley, and can be programmed with the TinyOS open-source operating system. While they were initially commercialized by Crossbow Technologies Inc., currently they are being developed by Memsic Corporation. The micaZ mote is employed as the reference sensor platform for the implementation of the low level components in the TOSSIM simulation.

The micaZ platform incorporates an 8-bit Atmel Atmega128L microcontroller [221] with an 8 MHz CPU. It integrates the following memory modules: 4 KB SRAM, 128 KB flash memory for program code, and 512 KB flash memory for measurement/data storage. The current consumption of the microcontroller in Active mode is 8 mA, while less than 15 μA in Sleep mode [16]. The micaZ mote integrates the 2.4 GHz Texas Instruments Chipcon CC2420 [29], the same radio transceiver incorporated in the telosB device (see Section 6.3.1).

The micaZ is powered by two AA batteries and it comes with a 51-pin UART expansion connector for communicating with expansion and gateway interfacing boards, e.g. the MIB520CB board [222] employed for programming and gateway connectivity purposes. The micaZ sensing board interfaces through the 51-pin expansion connector to provide add-on sensing capabilities.

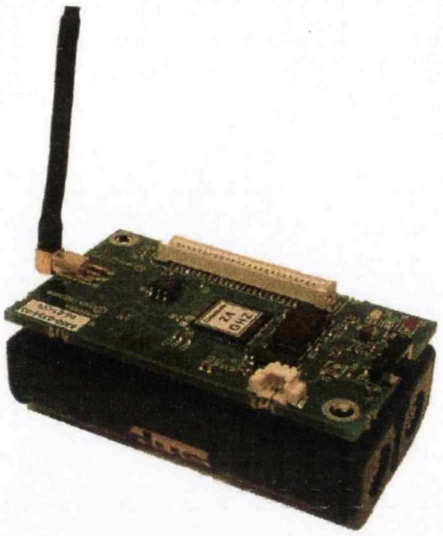


Fig. 6.2: MicaZ Mote

6.4 Experimental Settings and Performance Metrics

For evaluation purposes, a series of settings and metrics have been employed to assess the performance of the protocols under a wide range of test case scenarios.

Five topologies have been employed in the evaluation. The reference topology is a regular mesh layout comprised of nodes evenly distributed with an inter-distance of 25 meters; this type of topology is known as Square Grid (SG). Three scalable SG topologies have been created comprised of 64, 256 and 400 nodes with network sizes of 200x200 meters (see Figure 6.3), 400x400 meters (see Figure 6.5) and 500x500 meters (see Figure 6.7) respectively. Additionally, two irregular mesh topologies of 64 and 256 nodes have also been created where each node is placed randomly within its associated 25x25 meters area following the SG topology layout (see Figure 6.4 and 6.6); this type of topology is known as Random Uniform (RU).

Different network densities have been achieved by the variation of the transmission range - in real devices this is achieved through the node's transmission power. For this purpose, the link layer model generator from Zuniga and Krishnamachari [223] produces a TOSSIM compliant gain-based topology based on the geographical coordinates of nodes in the network. The tool allows to configure the parameters involved in the log-normal shadowing path loss propagation model which establishes the attenuation factor as a function of the inter-node distance. For the purpose of this evaluation, two configurations have been selected through experiments carried out in TOSSIM. The

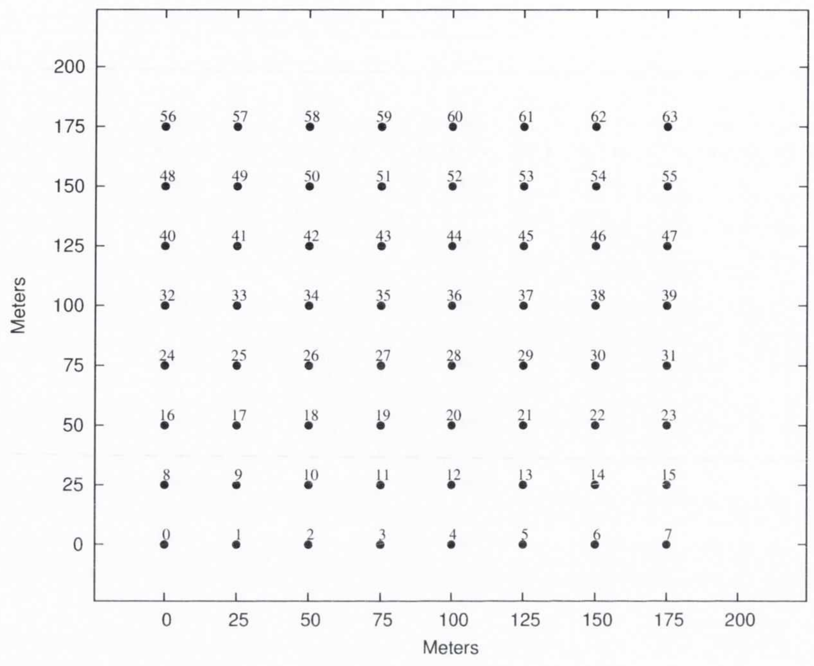


Fig. 6.3: Topology with 64 Nodes in a Square Grid (SG) (200x200 meters)

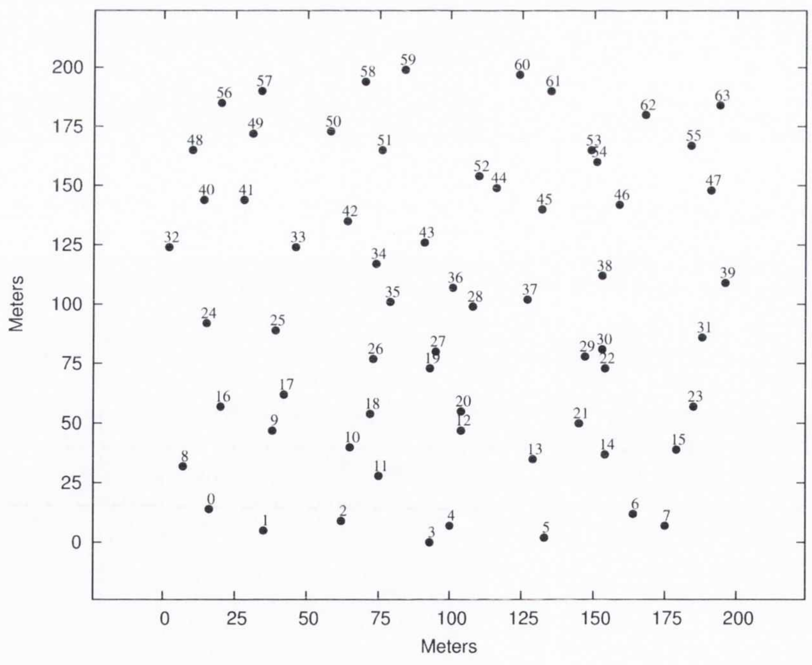


Fig. 6.4: Topology with 64 Nodes in a Random Uniform (RU) Grid (200x200 meters)

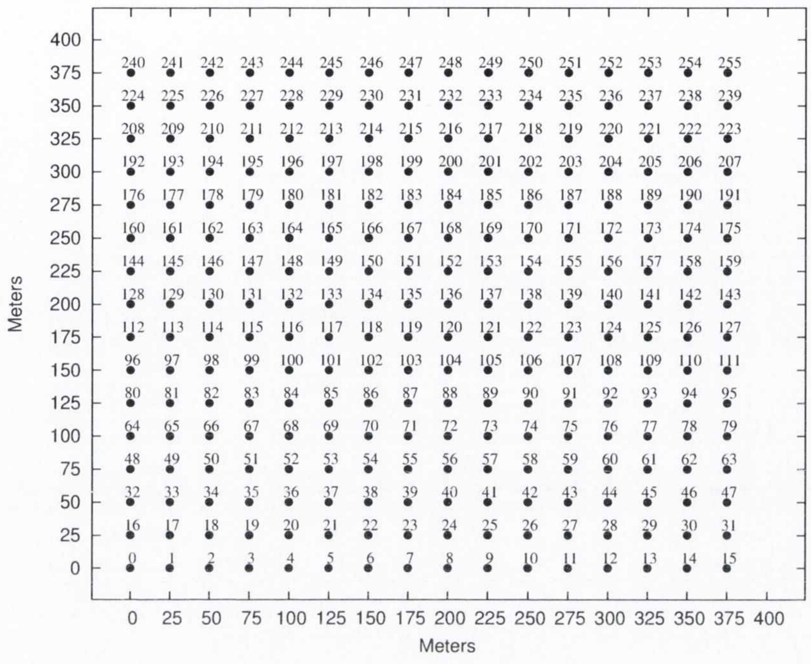


Fig. 6.5: Topology with 256 Nodes in a Square Grid (SG) (400x400 meters)

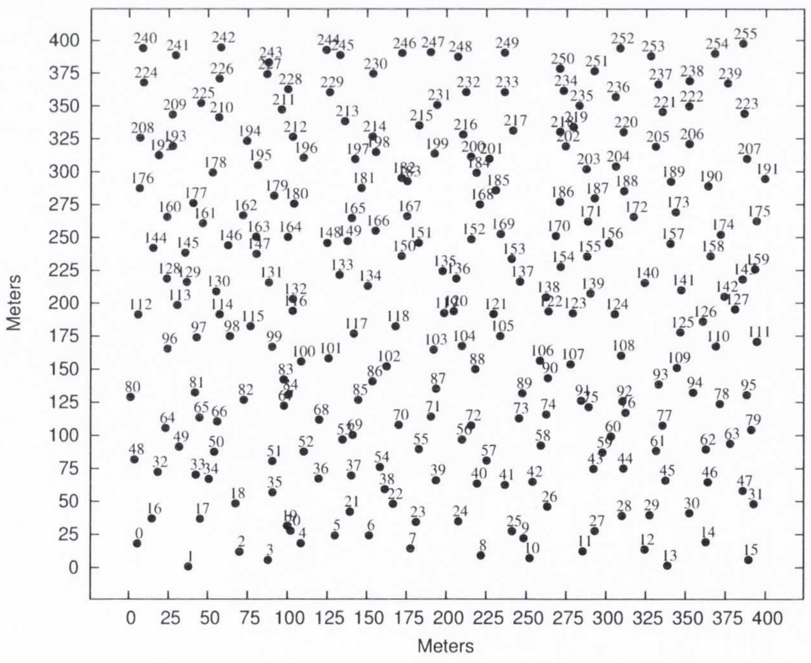


Fig. 6.6: Topology with 256 Nodes in a Random Uniform (RU) Grid (400x400 meters)

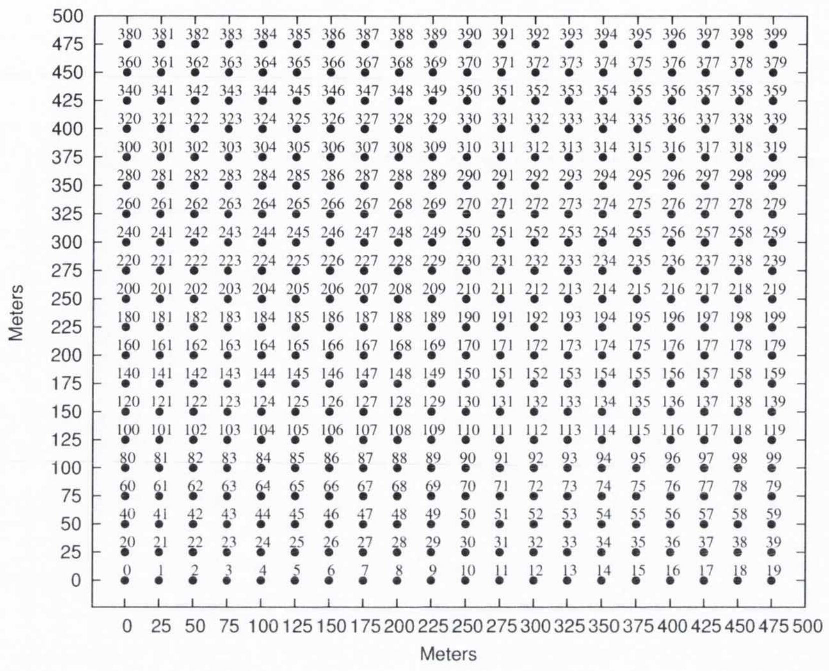


Fig. 6.7: Topology with 400 Nodes in a Square Grid (SG) (500x500 meters)

experiments calculate the average transmission range in meters when packets cease to be received by a node moving away from the sender. Employing the "Casino-lab" noise floor trace file in Table 6.1, the first configuration averages a radio transmission range of 37 meters while the second configuration produces a range of 56 meters.

The noise floor in the network has also been varied in the experiments. Two noise floor traces have been employed which are included in the TOSSIM simulator (see Figure 6.1). They differ from each other in the average noise floor value and the standard deviation of the sample distribution which impacts the packet reception ratio and its stability. The "Casino-lab" noise trace has been used in most of the test case scenarios, while the "Meyer-heavy" noise trace has been employed to produce scenarios with a high degree of packet loss.

At the Medium Access Control (MAC) layer, the contention-based Carrier Sense Multiple Access (CSMA) protocol with Collision Avoidance (CA) has been utilised both in the simulator and in the real testbed of TelosB devices. This is the default MAC protocol in TOSSIM which calculates backoff and preamble times according to the specifications of the CC2420 transceiver. Software acknowledgements have been enabled while hardware acknowledgements have been disabled.

6.4.1 Performance Metrics

The list of metrics employed in the performance evaluation of the TinyTorrents protocol operating on top of UMG are now presented and defined.

1. Total Packets Sent (TPS): The number of packets sent by a node at the MAC layer.
2. Total Packets Received (TPR): The number of packets received by a node at the MAC layer.
3. Network Total Packets Sent (NTPS): The sum of the Total Packets Sent by all the nodes in the network.
4. Network Total Packets Received (NTPR): The sum of the Total Packets Received by all the nodes in the network.
5. Average Network Total Packets Sent (A-NTPS): The sum of the Total Packets Sent by all the nodes in the network over the number of nodes in the network.
6. Average Network Total Packets Received (A-NTPR): The sum of the Total Packets Received by all the nodes in the network over the number of nodes in the network.
7. Torrents Received: The number of unique torrents (PublishTinyTorrent Message) received at each consumer node.
8. Torrents Completed: The ratio of the number of unique torrents for which the data has been successfully acquired to the Torrents Received metric.
9. Average Time Data File Completion (A-TDFC): The average time it takes for a torrent received at a node to acquire the whole data file from the moment the torrent is successfully selected from the queue of received torrents. A torrent might be selected and then returned to the queue when a valid peer list can not be retrieved from any tracker and there are more torrents waiting in the queue. The average is computed for all the torrents received at a node.
10. Piece Messages Sent: The total number of data messages (Piece Message) sent by each consumer/producer node. This metric quantifies the involvement of a consumer/producer node in the data distribution process of a torrent, or set of torrents, as compared to the rest of the consumers in the swarm. For comparison purposes, all the consumers/producers in the swarm must be involved in the data distribution of the same list of torrents.

11. Jain's Fairness Index (JFI): The Raj Jain Fairness Index [224, 225] is a metric employed to assess the degree of fairness in the resource utilisation within a set of nodes in the network. The JFI is computed by equation 6.1,

$$JFI(x_1, x_2, ..., x_n) = \frac{(\sum_{i=1}^{n} x_i)^2}{n \sum_{i=1}^{n} x_i^2} \tag{6.1}$$

where "n" is the number of nodes, and "x(i)" the resource value being studied. In our experiments, "x(i)" corresponds to the total number of packets sent or received. The interpretation of the JFI result ranges from: i) $\frac{1}{n}$ (worst case scenario) to ii) 1 (best case scenario) where all nodes are allocated the same resource.

6.5 Evaluation of the TinyTorrents System

This section provides an evaluation of the TinyTorrents protocol presented in Chapter 5 operating above the Ubiquitous Mobile Gradient (UMG) routing protocol (see Chapter 4). The UMG routing protocol has been selected over TinyHop due to its design which better enhances the data distribution and the unstructured discovery processes of the TinyTorrents protocol.

The performance evaluation seeks to demonstrate the capability of the TinyTorrents distribution system to disseminate data from producer nodes to a set of consumer nodes in a reliable and scalable manner where the network traffic is fairly distributed. The two modalities of the TinyTorrents protocol, i.e. centralised and decentralised, are evaluated and compared in terms of the metrics presented in Section 6.4. A performance comparison of the two modalities operating with the list of peer selection policies is presented in Section 5.4 which shows the degree of fairness achieved in both the network traffic and the data traffic amongst consumer nodes. A set of scenarios have been developed to test the performance of the system under different network conditions where the density and distribution of consumer nodes in the network has been varied.

6.5.1 Methodology

An application employing the exposed "TinyTorrents" interface (see Section 5.3.1) has been developed which controls the distribution process of the TinyTorrents framework. At any given time, the node can start distributing locally generated data, thus becoming a producer. A producer node indicates the data file which is to be distributed and instructs the TinyTorrents protocol to start the dissemination process. Currently, the system is capable of distributing files of up to 255 bytes; this limit has been established considering the available RAM memory of the motes in the real testbed, i.e. 10 KB for the telosB [30]. A data file of 255 bytes is subsequently divided by the TinyTorrents protocol into 16 pieces of data; by default the size of each piece of data is 16 bytes which has proved to achieve an efficient balance between reliability in the transmission of the message and amount of data being transported. The majority of the experiments have been carried out with one producer (node address 1), publishing a data file of 255 bytes every 100 seconds for a total number of 20 files (see Table 6.2). With this configuration, the effects of the distribution process can be clearly observed as a function of the number and the position of consumers in the network. When a producer node starts the distribution process, the torrent file is disseminated through the network. In this phase, the application layer decides, when receiving a torrent message, whether the node is acting as a consumer of the data file and whether the torrent file is to be disseminated (broadcast).

For this purpose, the TinyTorrents interface fires two events to the application layer when a new torrent file is received. The application layer decides i) if and when the fetching process of the data needs to be started ("delayToFetch" event) and, ii) if and when the torrent file is to be broadcast ("delayToForward" event); see Section 5.3.1 for a detailed explanation. Thus, based on the received torrent file, the node decides whether to become a consumer of the data file. For testing purposes, a set of strategies have been created which control the distribution of consumers in the network (see Section 6.5.1.2). In the same line, the strategy for the dissemination of the torrent file through the network is explained in Section 6.5.1.1 as a key artifact in the decentralised distribution process.

Table 6.2: TinyTorrents Protocol Simulation Parameters

Topology Layout	64 Nodes Square Grid (SG) (200x200m) ; 64 Nodes Random Uniform (RU) Grid (200x200m) ; 256 Nodes Square Grid (SG) (400 x 400m) ; 400 Nodes Square Grid (SG) (500 x 500m)
Transmission Range	37 m ; 56 m
Noise Floor Trace	Casino-lab ; Meyer-heavy
Data File Size	255 bytes (16 pieces)
Data File Producer	1 Producer (Node 1)
Producers Torrents Published	20
Producers Publish Interval	100 seconds
UMG Gradient Scope	5 hops

The same topologies employed for the evaluation of the routing protocols have been selected. The 64 nodes Square Grid topology (SG) and the 64 nodes Random Uniform (RU) Grid topology. In addition, 256 (SG and RU) and 400 nodes (SG) topologies have been created to test the scalability of the architecture. Two noise floor traces have been employed: i) the Casino-lab, which produces moderate levels of noise and has been employed in the majority of the simulation scenarios and ii) the Meyer-heavy, which has been included to demonstrate the effects of high noise environments in a busy network traffic scenario. The variation of the network density has been achieved through the transmission range of the nodes, 37 meters or 56 meters. A variation in the network density has an impact on: i) the wireless medium access contention, ii) the packets flow in the routing protocol queues and iii) the number of potential candidate consumers in the peer selection process within the default routing scope. In this regard, the UMG routing protocol gradient advertisement and discovery scope (routing scope) has been set to 5 hops by default. A consumer can retrieve data from other consumers within its routing discovery scope; if a consumer is out of the scope of other consumers, the "Routing Discovery Scope Increment" mechanism, part of the unstructured searching mechanisms (see Section 5.6) increments the scope by 3 hops on request for consumers placed outside of the default routing scope. However, only a subset of the consumer nodes might be required to increase the scope while advertising their gradient. This depends on the distribution of consumers in the network and the inter-consumer distance.

Table 6.2 displays the list of the simulation parameters employed in most of the experiments, however the UMG Gradient Scope and the number of Data File Producers have been modified for a set of specialized experiments which test the unstructured discovery mechanisms. The simulation runs until the distribution process for all the torrents is completed and there is no activity in the network. Metrics are collected every 100 seconds and before the simulation is due to finish.

In the remainder of this section, the centralised version of the TinyTorrents architecture is compared against the decentralised version for topologies with 64 nodes. The comparison evaluates the benefits of using a central tracker as opposed to multiple partial trackers. Node 28, placed at the centre of the network topology, has been selected as the central tracker node since it can be reached from any node in the network with the current value of UMG gradient routing scope, while providing a much fairer comparison with the decentralised version. The range of peer selection strategies proposed are also compared in both of the TinyTorrents modalities in order to identify the most suitable policy which fairly distributes: i) the load of data traffic amongst consumers and ii) the overall traffic amongst all the nodes in the network, while at the same time achieving a low communications overhead. The remainder of the experiments focus on the performance evaluation of the decentralised version of the TinyTorrents architecture configured to operate with the best peer selection strategy. The most reliable and efficient version of the TinyTorrents protocol is evaluated under a variety of environment and network conditions, network size, consumer distribution strategies, and realistic scenarios where multiple producer nodes simultaneously publish torrent files. Finally, the performance evaluation of the TinyTorrents protocol is validated on a 64 telosB testbed.

6.5.1.1 Torrent File Dissemination Strategies

The strategy to disseminate the torrent file through the nodes in the network is a key mechanism in the discovery of partial trackers in the decentralised version of the TinyTorrents protocol (see "Torrent Dissemination Control" in Section 5.6). When a torrent is received at a consumer node, the "tracker" field of the message is replaced with the local node such that nodes are aware of closer consumers acting as partial trackers. The idea is that the first received new torrent waits in queue for a small period of time before it is forwarded. Nodes, on reception of the first instance of a torrent message, delay their broadcast according to the value returned from the event "delayToForward" by the application. Only one instance of a torrent message is broadcast by each node. The rest of the received instances of the torrent message populate a short array of 4 potential trackers. When the torrent message is due to be forwarded by a non-consumer node, the closest tracker from those received is included as the new tracker in the message. For this reason, in order to locate closer trackers, consumer nodes introduce a lower delay when broadcast the torrent message than the rest of the nodes. This is a requirement imposed by the TinyTorrents protocol upon the application layer. However, the application layer decides the delay introduced by nodes and consumers in the forwarding of the torrent message. In this regard, a minimum delay of 1 second in the forwarding of the torrent message is imposed on the application layer. This is due to the fact that the TinyTorrents protocol employs a number of sending attempts to increase the torrent message getting received. By default, the producer broadcasts the torrent message three times while the rest of the nodes broadcast it twice. In between broadcasts a delay of 300 milliseconds is introduced. In addition, a randomized time of 200 ms is added to minimize the likelihood of collisions.

For evaluation purposes the delay to forward the torrent message has been set to 1 second in consumer nodes while non-consumer nodes delay the forwarding for 3 seconds; this configuration has been selected as it produces a progressive dissemination of the torrent messages through the network containing the address of closer partial trackers.

6.5.1.2 Consumer Distribution Strategies

The number and the distribution of consumers in the network is also a key factor which impacts the data distribution process. Having a high density of consumers will not exploit the benefits of the TinyTorrents selective data dissemination mechanism, but rather would suggest the use of epidemic mechanisms for data dissemination. On the other hand, if consumers are at a distance higher than the routing discovery scope (UMG Gradient Scope), the TinyTorrents system launches its unstructured discovery mechanisms (see Section 5.6). Expanding the routing discovery scope increases the communications cost and, at high distances, the packet delivery ratio starts to drop. Thus, the strategy for selecting which nodes become consumers of a given torrent are paramount for the TinyTorrents system to be fully scalable. A balance needs to be achieved which takes into account the default routing discovery scope and the scope increment value employed by the unstructured discovery mechanisms to expand the searching scope.

Next, the consumer distribution strategies created for the performance evaluation of the TinyTorrents system are defined:

1. **"64_8"**: 8 consumers have been selected in the 64 Nodes Topologies (SG & RU) according to Figure 6.8.
2. **"64_3ID"**: 22 consumers have been selected in the 64 Nodes Topologies (SG & RU) according to Figure 6.9.
3. **"256_25"**: 25 consumers have been selected in the 256 Nodes Topologies (SG & RU) according to Figure 6.10.
4. **"256_3ID"**: 86 consumers have been selected in the 256 Nodes Topologies (SG & RU) according to Figure 6.11.
5. **"400_21"**: 21 consumers have been selected in the 400 Nodes Topology (SG) according to Figure 6.12.
6. **"400_3ID"**: 134 consumers have been selected in the 400 Nodes Topology (SG) according to Figure 6.13.
7. **"_RAND_X"**: a node becomes a consumer with a random likelihood of X% with the first torrent message received from a producer. The first torrent message received from a producer (sequence value is 0) establishes the consumer distribution in the network.

Most of these strategies have been selected to enable comparison with different density of consumers and inter-consumer distance. However, the "_RAND_X" strategy is employed as a non-deterministic consumer distribution which can be used in unsupervised deployments where the inter-consumer distance might be greater than the routing discovery scope. When this occurs, the unstructured discovery mechanisms are launched to locate a closer partial tracker; this might require to increment the routing discovery scope.

The TinyTorrents protocol sets a maximum time to complete the acquisition of a torrent data file before it is cancelled and discarded. Once a torrent is discarded or completed, the data fetching process for the next torrent stored in the queue is launched. Currently, on reception of a torrent, a consumer node delays the start of the data fetching process ("delayToFetch" event) between 2 and 3 seconds. The time since the fetching process starts until the torrent is completed, or dropped, is computed with the Average Time Data File Completion (A-TDFC) metric for all torrents. Since the torrent file is disseminated through the network with a delay of 1 or 3 seconds at each hop, consumers will receive the torrent and start their data fetching process before a closer partial tracker is capable of providing a valid peer list. Thus, consumers need to wait and delay the acquisition of

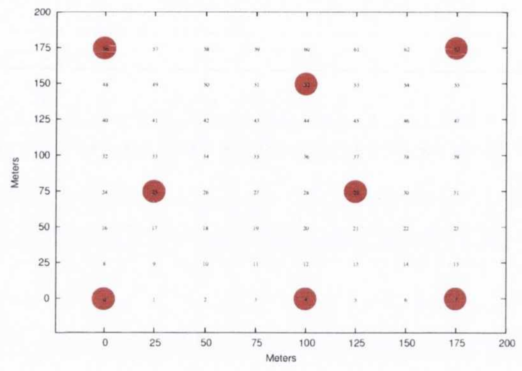


Fig. 6.8: 64_8 Consumer Distribution Strategy for the 64 Nodes Topologies

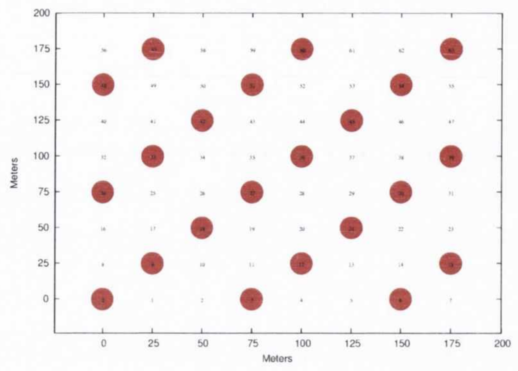


Fig. 6.9: 64_3ID Consumer Distribution Strategy for the 64 Nodes Topologies

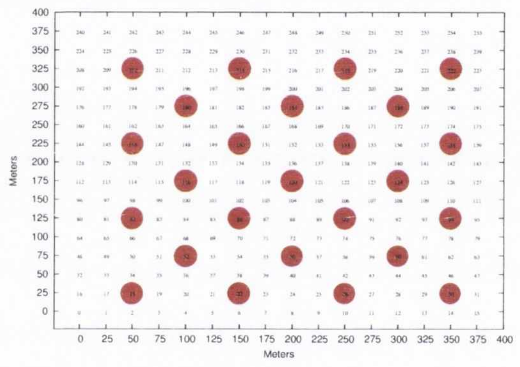


Fig. 6.10: 256_25 Consumer Distribution Strategy for the 256 Nodes Topologies

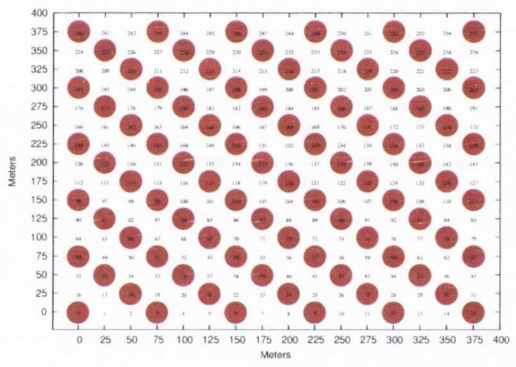


Fig. 6.11: 256_3ID Consumer Distribution Strategy for the 256 Nodes Topologies

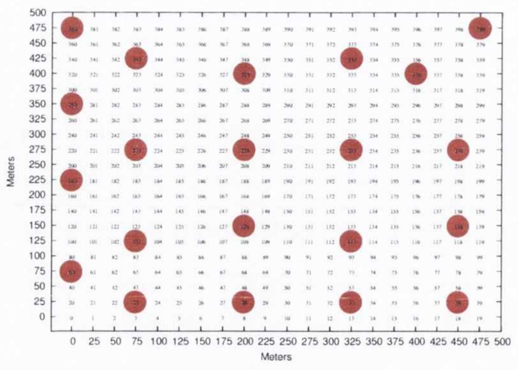


Fig. 6.12: 400_21 Consumer Distribution Strategy for the 400 Nodes Topology

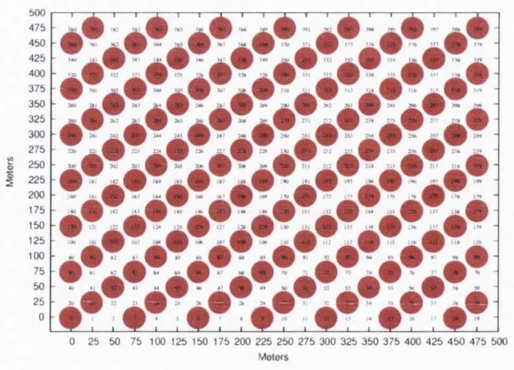


Fig. 6.13: 400_3ID Consumer Distribution Strategy for the 400 Nodes Topology

the data file until a valid peer list is retrieved from one of the potential partial trackers; this has an impact in the A-TDFC since this metric calculates the time since the torrent is successfully selected from the queue of received torrents to start its data fetching process.

In this respect, consumers query the central tracker, or the list of partial trackers known, on a regular basis (every 2 seconds) until a valid peer list is retrieved. In the decentralised version, partial trackers are queried on a round-robin fashion, and a partial tracker can only reply with a valid peer list when it has become a seeder of the torrent being requested. This restriction acts as an admission control mechanism which regulate the data distribution process of each consumer in such a way as to increase the likelihood of communication with closer consumers which contain pieces of the data file. As a consequence, the A-TDFC at each consumer provides an indication of the speed of the distribution process to reach the consumer area. However, when more than one torrent is in the queue to be processed, the protocol does not wait until a valid peer list is received from a tracker, but rather it selects the next torrent in the queue until it finds one for which a valid peer list can be retrieved. The queue is checked cyclically. When a selected torrent is returned to the queue due to the failure to retrieve a valid peer list from any of its potential partial trackers, the TDFC value is reset. In this sense, having more than one torrent in the queue will reduce the likelihood of achieving higher values of A-TDFC, and thus the A-TDFC value could no longer be used to study the agility of the distribution process. Additionally, a maximum value of TDFC must be set in the protocol to prevent consumer waiting for long periods of time. This value has been set to 300 seconds, such that the results never reach it and thus the agility of the distribution process can be studied. The 300 seconds maximum completion time is a sufficient high boundary for all the experiments to result in a **100% "Torrents Completed" ratio**, i.e. all data files for the received torrents are successfully acquired by all the consumers in all the scenarios presented.

6.5.1.3 Peer Selection Strategies

The peer list selection strategy presented in Section 5.4.1 is utilised by a node acting as tracker to select a peer list. The same strategy has been employed in both the centralised and decentralised versions of the TinyTorrents protocol, which employs the position of arrival of the peers into the swarm of the torrent at the central tracker or at a partial tracker. However, a partial tracker only replies with a peer list of a torrent for which it is a seeder. When a peer list message from a tracker provides a peer list with the maximum number of peers (by default 4), a second peer list request is sent to the tracker. The second request seeks to obtain a different set of peers and thus transports in its payload the list of peers which have already been received (see Section 5.2.2). On reception of the second peer list, the closest peer list (4 peers) are selected from the total set of peers retrieved (maximum 8). In this configuration, the α coefficient of the position-based peer list selection strategy in Equation 5.2 (see Section 5.4.1) is set to its maximum value (0xFF) to favour the selection of peers with high positions.

Table 6.3: TinyTorrents Protocol Evaluation - Peer Selection Strategies

RD	Random Selection
PFPL	PieceRemaining First Position-Location-based Peer Selection
PO	Position-based Peer Selection
PL	Position-Location-based Peer Selection
CFPO	Closest First Position-based Peer Selection
1FPL	1-Hop First Position-Location-based Peer Selection
CFP	Closest First PieceRemaining-based Peer Selection

In addition, the list of peer selection strategies proposed in Section 5.4.2 have been evaluated. They are employed to select the peer of the list from which to acquire a given piece of data. Each strategy has an impact in the: i) network traffic load, both in terms of received and sent packets, ii) the fair distribution of pieces of data amongst consumers, and iii) the average time to complete the data file (A-TDFC). A performance comparison of the range of peer selection strategies is shown in the following section where the acronyms for the strategies are shown in Table 6.3.

6.5.2 Impact of Peer Selection Strategies: Centralised vs. Decentralised

This section presents an evaluation of the performance of the set of proposed peer selection strategies in scenarios of 64 nodes and 256 nodes with the following consumer distribution strategies: 64_8, 64_3ID, 256_25 and 256_3ID. In addition, the performance of the centralised (Central) and decentralised (Decentral) versions of the TinyTorrents protocol have been analysed and compared for the 64 node consumer strategies (64_8, 64_3ID). The topology layout selected for this evaluation is the Square Grid (SG), with a node transmission range of 37 meters and the Casino-lab noise floor trace. This evaluation identifies the most suitable peer selection strategies for data distribution in wireless sensor networks while showing the efficiency of the decentralised version over the centralised version of the TinyTorrents protocol. All figures show results computed as the average of 5 repetitions for each scenario.

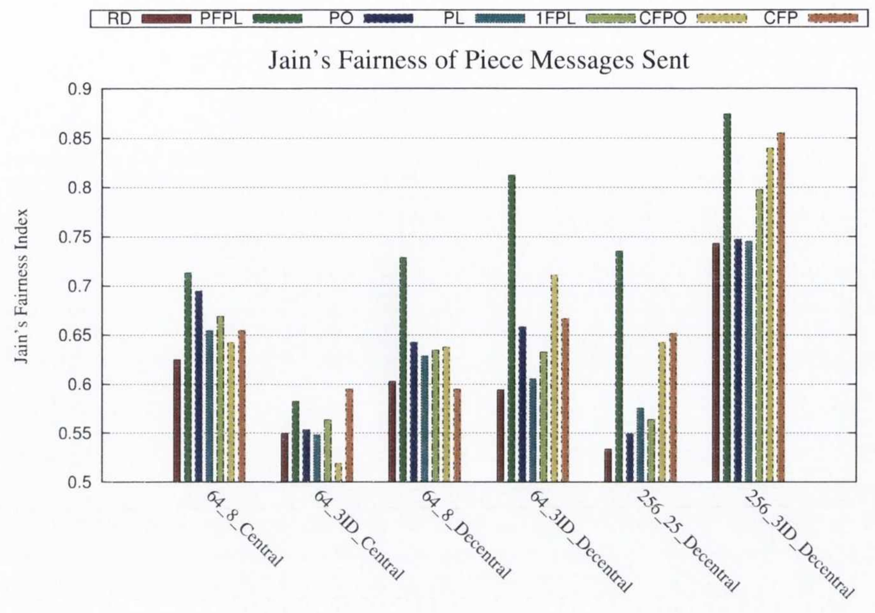


Fig. 6.14: Impact of Peer Selection Strategies - Jain's Fairness Index (JFI) of Piece Messages Sent - Average of 5 repetitions - Topologies: Square Grid, Transmission Range: 37 meters, Noise: Casino

The Jain's Fairness Index (JFI) in equation 6.1 has been employed to calculate the data traffic load of the set of nodes in the swarm of a torrent(s) from a given producer. This has been achieved by computing the JFI of the number of Piece Messages Sent by each of the consumers, including the producer, in each scenario. The index provides the degree of fairness in the data piece distribution process; the metric does not take into account the routing traffic at each node. The JFI worst case scenario is computed as $\frac{1}{n}$, where n is the number of nodes in the swarm of the torrent capable of sending pieces of data (i.e. the number of consumers plus the producer). According to the number of consumers in each of the strategies proposed, the JFI worst case - best case interval is: i) 64_8 [0.11,1]; ii) 256_25 [0.038,1]; 64_3ID [0.043,1]; 256_3ID [0.011,1]. However, values close to 1 are not expected mainly due to the fact that consumer nodes at the edges of the network, or at the end of the range of the data distribution process, might not even be required to provide pieces of data.

The JFI of the Piece Messages Sent for the set of scenarios is shown in Figure 6.14. Overall, values of the JFI are above 0.5 which indicates the appropriateness of the position-based peer list selection strategy at the tracker node, even for the random peer selection strategy. The PieceRemaining First Position-Location-based (PFPL) peer selection strategy outperforms the rest of the strategies in most of the scenarios. As expected, the number and distribution of consumers has an impact in the JFI value. For the centralised version of the TinyTorrents (Central), a sparse consumer distribution

(64_8) improves the degree of fairness in the data piece distribution as compared to a highly dense consumer distribution (64_3ID). Having a high number of consumers requesting a peer list within a relatively short period of time, in combination with the position-based peer list selection algorithm at the tracker, increase the likelihood of including consumers in the peer list which have not yet acquired a substantial number of pieces. Consequently, this has a negative impact on the fairness of the overall data piece distribution process. An admission control mechanism could be in place to delay the sending of the peer list when receiving multiple requests from consumers within a short period of time. However, the lack of topology information at the central tracker could delay consumers which might be key to provide pieces of data to farther consumers. This is one of the reasons why the decentralised approach is preferred over the centralised one. In fact, the decentralised (Decentral) version of the TinyTorrents protocol provides higher JFI values when the consumer density is higher (3ID) in all the scenarios for the majority of the peer selection strategies. This is a consequence of the intrinsic fair behaviour of the decentralised version by which each partial tracker provides peer lists containing proximate consumers. This is also the reason why the proximity-based peer selection strategies, CFPO and CFP, produce a good degree of fairness, specifically in high density of consumers (3ID), despite not reaching the level of the PFPL strategy.

According to the JFI results in Figure 6.14, the PFPL is the best strategy when seeking to distribute the load of data traffic among consumers. However, the total number of packets sent and received in the network need to be analysed to determine the efficiency in terms of communication at network level. For this purpose, Figure 6.15 and 6.16 show the Average Network Total Packets Sent (A-NTPS) and Received (A-NTPR) respectively for each of the scenarios being studied. In this regard, the ratio of the A-NTPR to the A-NTPS, which indicates the average number of neighbours receiving a packet from a neighbour node in the network, ranges between 5 and 6. Results indicate the inefficiency of the TinyTorrents centralised solution as compare to its decentralised version in scenarios with a high density of consumers (64_3ID), while showing similar results when the number of consumers is relatively low (64_8). In all the scenarios, the two proximity-based peer selection strategies (CFPO and CFP), prove to be the most efficient in terms of packets sent and received at network level, with similar results. This is due to the Closest-First policy which initially selects the set of nearest nodes (in terms of hops) before refining the selection according to the nodes position in the swarm (CFPO), or the number of pieces remaining to be acquired in the nodes (CFP). Moreover, the PFPL strategy shows slightly higher values of packets sent and received in low density consumer distributions (64_8 and 256_25), while producing higher results in scenarios with high density of consumers (3ID), mainly in the Centralised version.

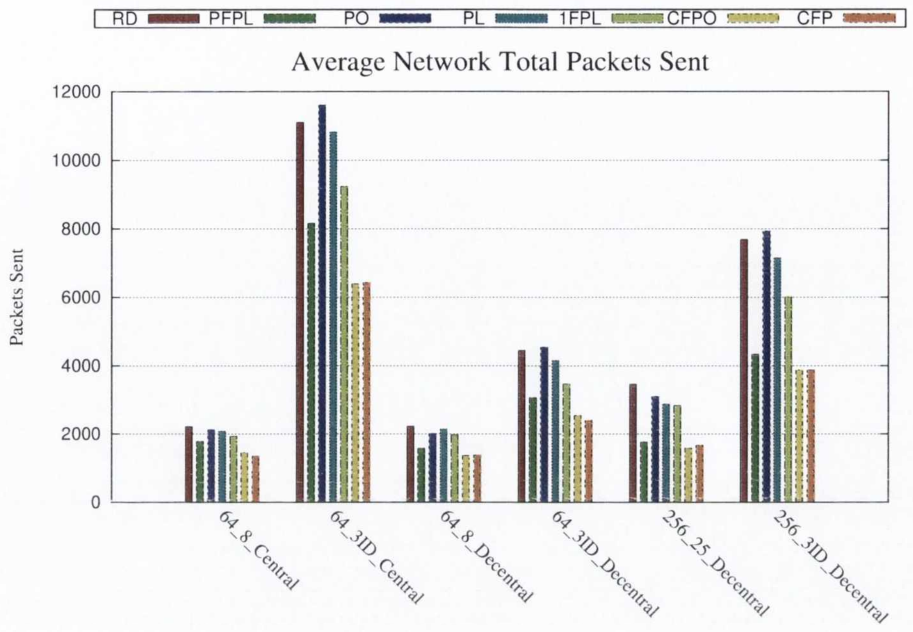


Fig. 6.15: Impact of Peer Selection Strategies - Average Network Total Packets Sent - Average of 5 repetitions - Topologies: Square Grid, Transmission Range: 37 meters, Noise: Casino

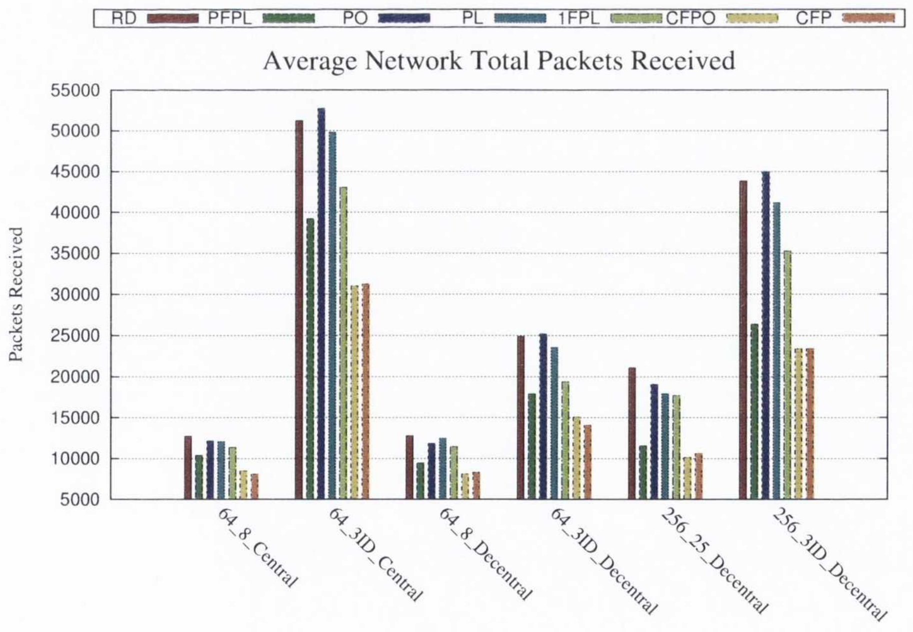


Fig. 6.16: Impact of Peer Selection Strategies - Average Network Total Packets Received - Average of 5 repetitions - Topologies: Square Grid, Transmission Range: 37 meters, Noise: Casino

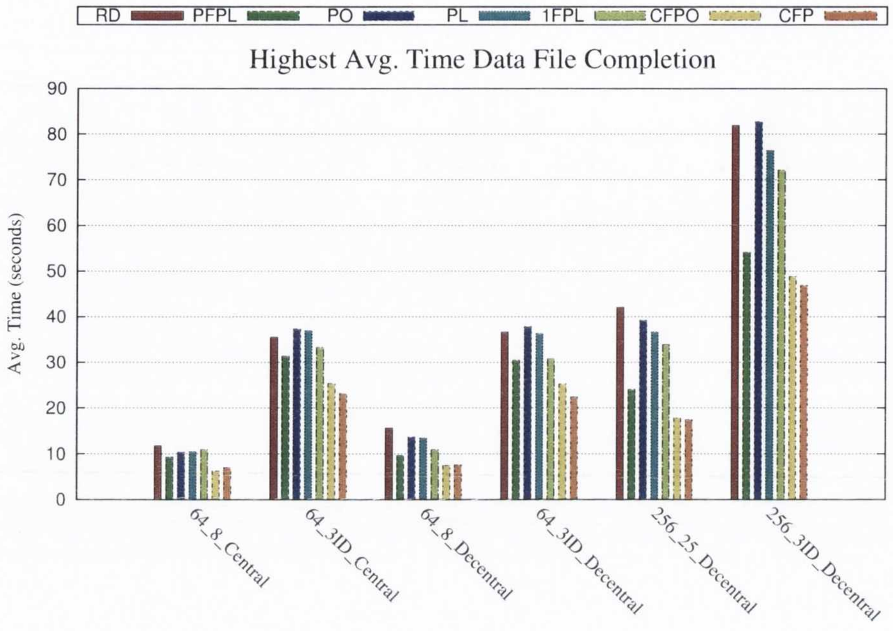


Fig. 6.17: Impact of Peer Selection Strategies - Highest Average Time Data File Completion - Average of 5 repetitions - Topologies: Square Grid, Transmission Range: 37 meters, Noise: Casino

On the other hand, the average time a node receiving a torrent takes to acquire the whole data file needs to be considered on the selection of the most appropriate peer selection strategy. In Figure 6.17, the highest time value of the Average Time Data File Completion (A-TDFC) for all the nodes in the network is shown. This metric gives an indication of the time the distribution of the data file in the network takes to complete. The CFPO and CFP strategies show the lowest times, with CFP performing slightly better for most of the cases. PFPO shows the third best time in all the experiments, with a higher impact than CFPO and CFP in high density consumer distribution scenarios (3ID). It has to be remarked that the highest A-TDFC achieved by the centralised and decentralised versions both for 64_8 and 64_3ID consumer strategies is similar. This time corresponds to the node which takes the longest to acquire the data file, and is based on the availability of pieces from nodes in the peer list. Peers in the list must be within the routing scope of the node. In these scenarios, the latest node to acquire the data file is not within the routing scope of node 1 or the set of first seeders. Therefore, the node needs to wait before the second request of a peer list from the central tracker. By the time the node can start the distribution process in the centralised version, the distribution process of the decentralised version has progressed in the direction of the node and a partial tracker and peers are also available within the routing scope of the node (5 hops).

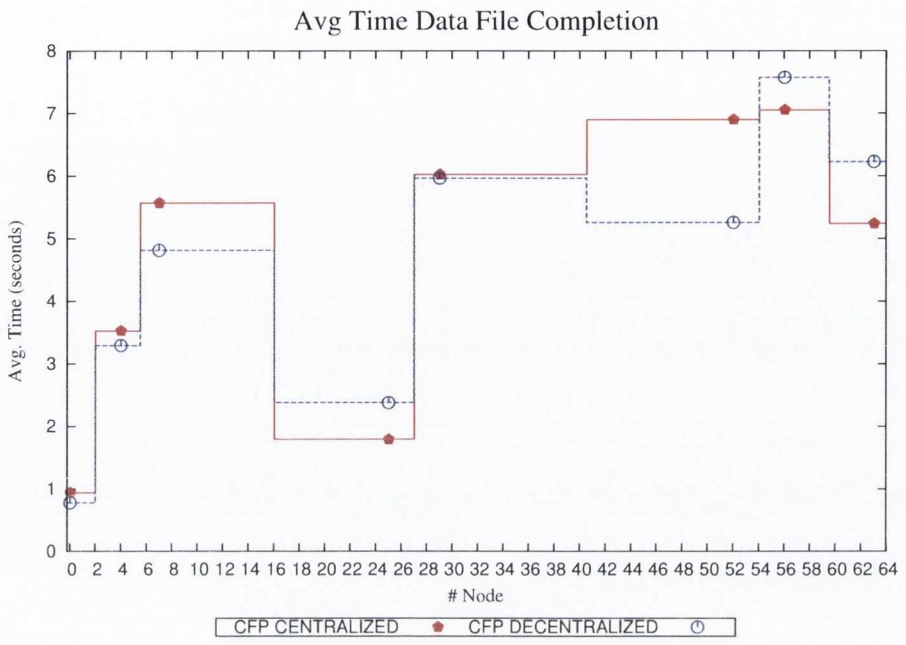


Fig. 6.18: Comparison of TT Centralised vs. Decentralised on Average Time Data File Completion - CFP, 64_8

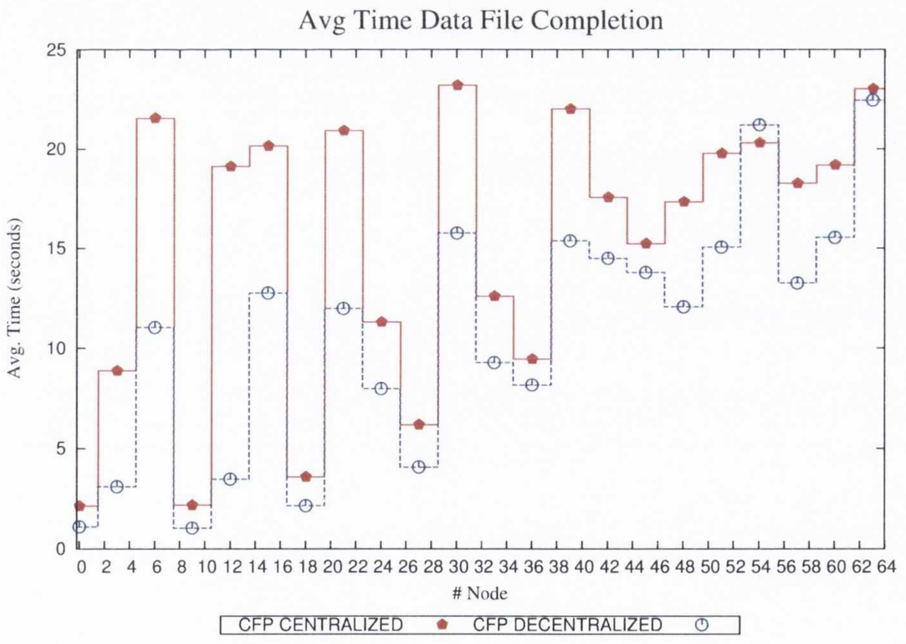


Fig. 6.19: Comparison of TT Centralised vs. Decentralised on Average Time Data File Completion - CFP, 64_3ID

This effect can be clearly seen in Figures 6.18 and 6.19 which show the Average Time Data File Completion of the 20 torrents received per consumer for the centralised and decentralised versions of the protocol. The experiment employs the CFP peer selection strategy with the 64_8 and 64_3ID consumer strategies.

Figure 6.18 shows a variation in the average time of consumers to acquire the data file of at most 2 seconds between the performance of the two versions of the protocol. This is due to the effect of randomness in the acquisition of data and the peer list selection process. On the other hand, Figure 6.19 shows the results for the 64_3ID scenario where 22 consumers acquire the data file. Results indicate that the decentralised protocol exhibits a smooth progression in the average time to acquire the data file according to the position of the nodes in the grid with respect to the producer node (node 1). The centralised protocol produces longer times for nodes which are not within the routing scope (5 hops) of the producer. This behaviour is due to the fact that the central tracker is unaware of inter-consumer distance when populating the peer list. This problem is ameliorated in the decentralised version of the protocol by enabling consumers to act as partial trackers for nearby consumers. Thus, the decentralised version fosters a progressive dissemination and tends to minimize situations of high traffic congestion.

Overall, the decentralised version performs better than the centralised modality in the majority of the scenarios, while at the same time providing a higher degree of fault tolerance. Thus, the remainder of the chapter studies the performance of the decentralised version of the TinyTorrents protocol under a range of different conditions. Results also indicate that the PieceRemaining First Position-Location-based (PFPL) peer selection strategy provides a high degree of fairness, while the Closest First Position-based (CFPO) and the Closest First PieceRemaining-based (CFP) are more efficient in terms of average network total packets sent and received. While these metrics provide an indication of the performance of the network on average, information of the traffic load amongst all the nodes in the network is key to identifying the most suitable peer selection strategy for the decentralised distribution process. In this regard, the number of Total Packets Received by each node has been plotted in the topology layout which enables clear visual inspection of the network traffic load as well as the identification of the areas with higher communication activity and hot spots. In addition, the number of Piece Messages Sent has also been plotted using the same format thereby providing a visual representation of the degree of fairness represented by the JFI metric. An experiment from the scenario with the 3_ID consumer distribution (64_3ID_Decentral) has been selected due to the large number of consumers which emphasizes the impact of the peer selection strategies. Figures 6.20, 6.22 and 6.24 show the Total Packets Received per node while Figures 6.21, 6.23 and 6.25 show the Piece Messages Sent per node for the CFP, CFPO and PFPL peer strategies respectively. Each metric value has been normalized to fit the right-hand side scale which employs both the colour and the diameter of each node to represent the value.

By visual inspection, the distribution of Piece Messages Sent per node in Figures 6.21 (CFP), 6.23 (CFPO) and 6.25 (PFPL) clearly show the fair data distribution achieved by the PFPL strategy (0.8125 JFI) as compared to CFP (0.6667 JFI) and CFPO (0.7112 JFI). The lower number of Piece Messages Sent by nodes farther away from the producer (node 1) which are placed mainly at the edges of the network needs to be noted. This effect is expected as nodes at the edges become seeders of pieces when most of the nodes within their partial tracker swarm have already received most of the pieces. Since PFPL selects peers with the lower number of pieces, nodes at the edges show a higher number of Packets Sent than those in the CFP and CFPO scenarios. This is also the reason why the PFPL strategy has a higher degree of fairness than the proximity-based strategies (CFP and CFPO).

On the other hand, it can be clearly noted that the distribution of Total Packets Received in the network tends towards the centre of the grid. This is expected as central nodes are surrounded by a higher number of neighbours and are also aware of a higher number of nodes at the routing layer which is determined by the routing discovery scope (5 hops). Both CFPO (see Figure 6.22) and CFP (see Figure 6.20) strategies show a similar traffic load balance in terms of total packets received mainly due to the Closest-First policy which selects nearby nodes. In Figure 6.24, the PFPL strategy produces a higher number of packets received than the CFPO and CFP strategies, specifically in nodes placed towards the centre of the network. Again, this effect is a consequence of the 5 hops routing scope which makes central nodes aware of a higher number of nodes as compared to nodes at the edges; this increases the routing cost in terms of packets sent and received. When the network scales in scenarios with even distribution of nodes (SG) and consumers (3ID), the traffic load balances out for central nodes as a result of these nodes being aware of a similar number of nodes within the routing scope of 5 hops (see Section 6.5.4).

Despite the high Jain's Fairness Index of the PFPL strategy, the number of packets sent and received by a node are the key metrics which impact the status of the communications in a wireless sensor network in terms of traffic congestion and load balancing. For this reason, CFP and CFPO have been selected as the most suitable peer selection strategies for the decentralised version of the TinyTorrents. Despite CFPO performing better in JFI terms for the 64_Decentral scenarios, CFP has been chosen for evaluation in the remainder of the section as it provides a higher fairness index in scalable scenarios with 256 nodes.

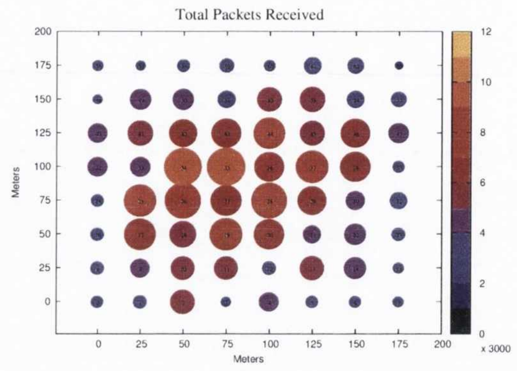


Fig. 6.20: Total Packets Received - CFP, 64_3ID , TT Decentral, SG, 37m, Casino

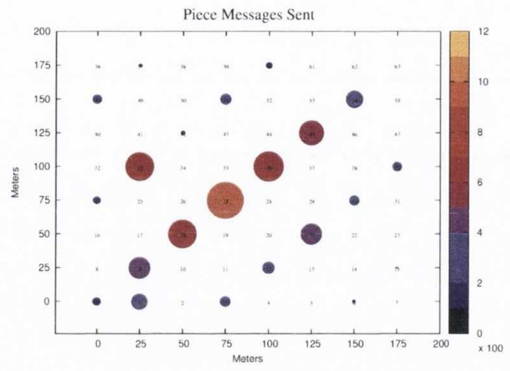


Fig. 6.21: Piece Messages Sent - CFP, 64_3ID , TT Decentral, SG, 37m, Casino

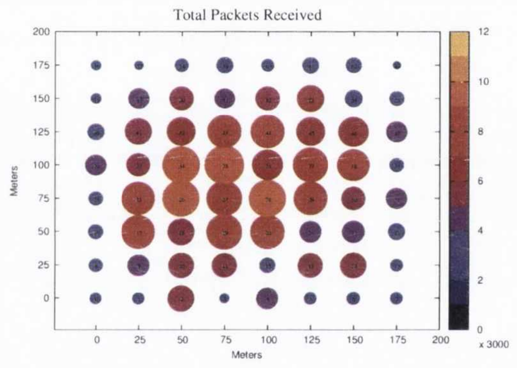


Fig. 6.22: Total Packets Received - CFPO, 64_3ID , TT Decentral, SG, 37m, Casino

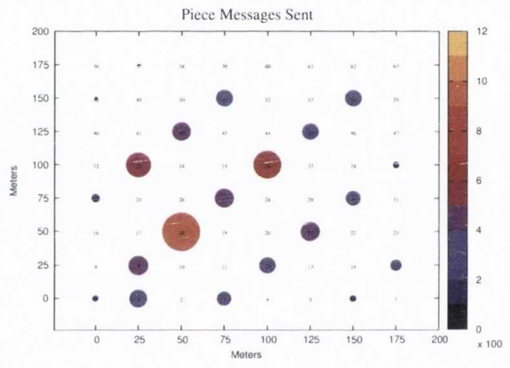


Fig. 6.23: Piece Messages Sent - CFPO, 64_3ID , TT Decentral, SG, 37m, Casino

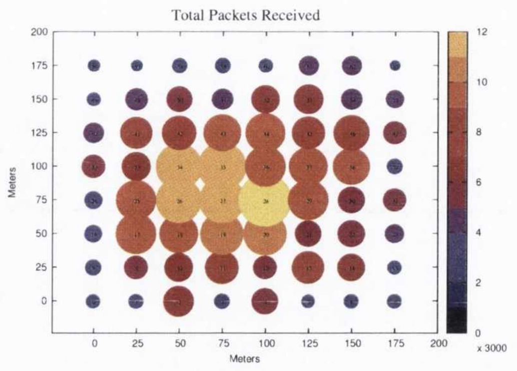


Fig. 6.24: Total Packets Received - PFPL, 64_3ID , TT Decentral, SG, 37m, Casino

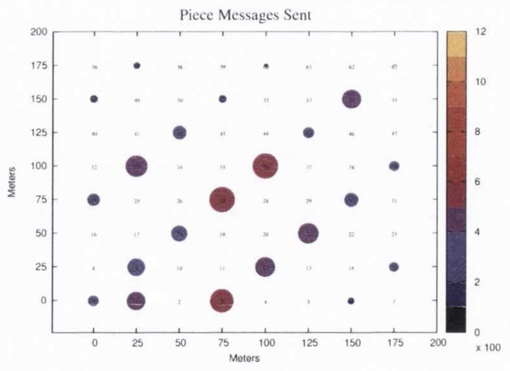


Fig. 6.25: Piece Messages Sent - PFPL, 64_3ID , TT Decentral, SG, 37m, Casino

6.5.3 Impact of Noise Floor and Network Density

The TinyTorrents decentralised version operating with the Closest First PieceRemaining-based (CFP) peer selection strategy has been tested under high levels of noise and network density in terms of Torrents Completed, JFI of Piece Messages Sent, Average Network Total Packets Sent and Received, and Highest Average Time Data File Completion. The 64 and 256 Random Uniform (RU) grid topologies have been employed as irregular layouts together with a node transmission range of 37 meters (37_RU). Higher network density in the Square Grid (SG) topology has been achieved by increasing the nodes transmission range from 37 meters to 56 meters (37_SG, 56_SG). For each scenario, the low-density consumer distribution, i.e. 8 consumers for 64 node scenarios and 25 consumers for 256 node scenarios, and the high-density consumer distribution (3ID) have been tested. All the above scenarios have been configured with the Casino noise floor. Additionally, the impact of high noise has been studied for the set of 64 node scenarios employing the Meyer noise floor trace. Figures show results grouped in sets of 3 scenarios - from now on called "density sets" - which correspond to the variation in network density and irregular layout (37_SG, 56_SG, 37_RU). For 64 node scenarios, the density sets have been compared with the 64_ 8 and the 64_3ID consumer distributions both for Casino and Meyer noise floors. The last two density sets correspond to the 256 nodes topologies operating under the Casino noise floor, for the 256_25 and 256_3ID consumer distributions.

Figure 6.26 shows the Jain's Fairness Index of Piece Messages Sent. As in previous results, the higher the number of consumers in the network, the fairer the process of data distribution. However, the inter-consumer distance and the routing scope influence this assertion. High noise floor (Meyer) has a negative impact in the JFI, mostly in irregular and high network density scenarios with a high number of consumers (64_3ID). In general, a clear pattern does not arise from the results which indicates a relationship between network density and fairness. Nevertheless, higher density scenarios (56 meters transmission range) provide equal or higher values of fairness for all the scenarios except for the 256_3ID. This is due to the higher number of nodes at the partial tracker in the peer list selection process and the availability of a greater number of partial trackers in the 256_3ID scenario; there is a high number of consumers at the same closest distance and not all of them are required to complete the whole data file. On the other hand, a higher value of JFI is obtained in the 256_37_3ID scenario with respect to the higher network density scenario, 256_56_3ID, which indicates that a large number of peers in the swarm of partial trackers might have a negative impact in the overlapping of the distribution processes. Still, scenarios of 256 nodes provide high values of fairness (JFI). Additionally, irregularity in the network layout (RU) does not directly impact the JFI as much as the distribution of the consumers in the layout does.

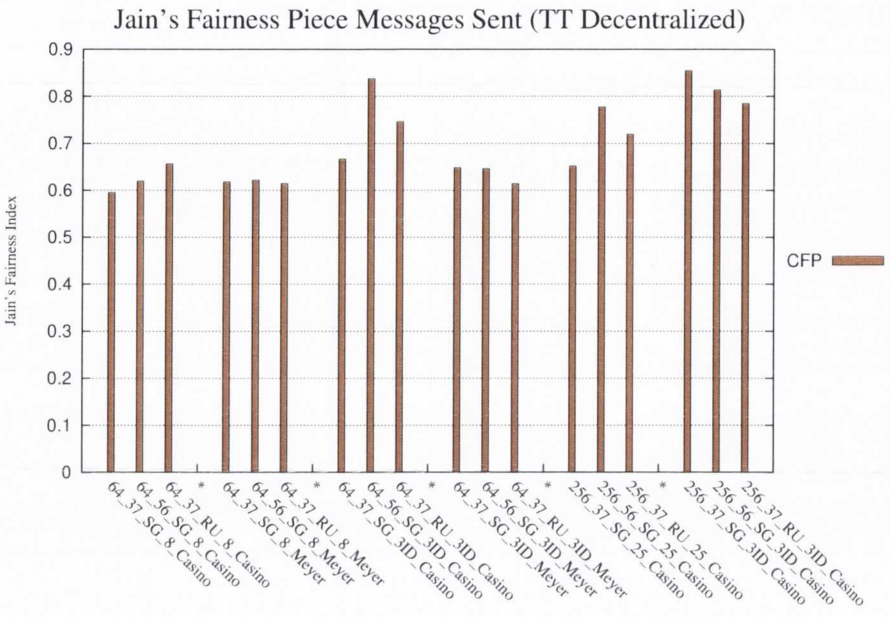


Fig. 6.26: Impact of Noise Floor and Network Density - Jain's Fairness Index of Piece Messages Sent - Average of 5 repetitions

While the Jain's Fairness index of the Piece Messages Sent is a metric to consider for the comparison of peer selection strategies, the Average Network Total Packets Sent and Received provide more relevant information on the performance of the protocol under different noise and network density conditions.

Figures 6.27 and 6.28 show the overall increase of the average total number of packets sent and received in the network in the presence of high noise floor (Meyer). As compared to the moderate noise floor (Casino) scenarios, the Average Network Total Packets Sent (A-NTPS) in high noise floor (Meyer) scenarios increases by a factor in the range of 6.15 to 7.8 while the Average Network Total Packets Received (A-NTPR) increases by a factor in the range of 3.3 to 4.3.

High network density scenarios (56 meters transmission range) produce the highest average of network total packets sent in scenarios of a high number of consumers (64_3ID and 256_3ID), while the lowest average of network total packets sent in scenarios of a low number of consumers (64_8 and 256_25). In terms of overall number of received packets, a higher network density (56 meters transmission range) produces the highest number of packets received in most of the cases, however the irregularity in the layout of a topology can have a higher impact in the reception (64_37_RU_8). The ratio of the Average Network Total Packets Received to the Average Network Total Packets Sent

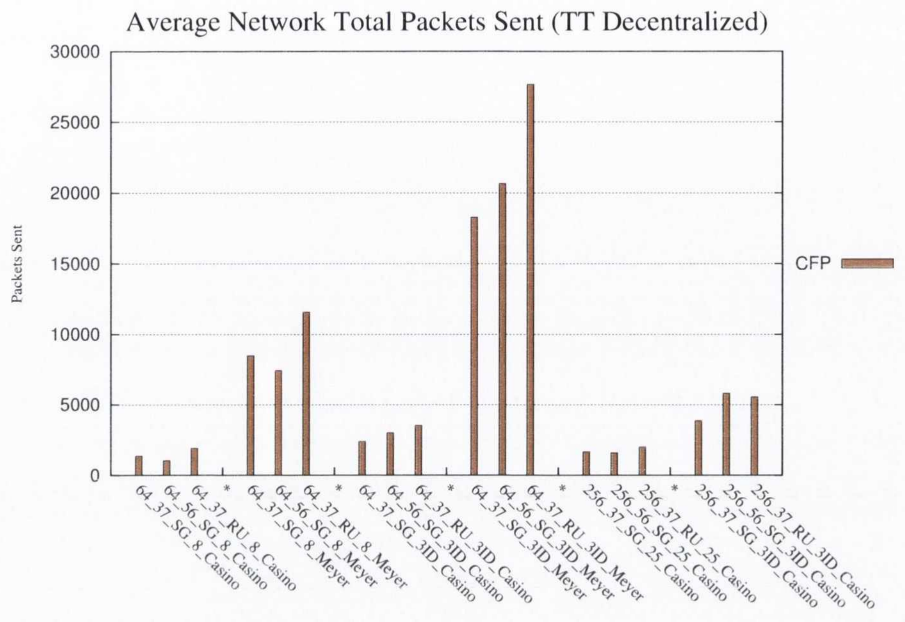


Fig. 6.27: Impact of Noise Floor and Network Density - Average Network Total Packets Sent - Average of 5 repetitions

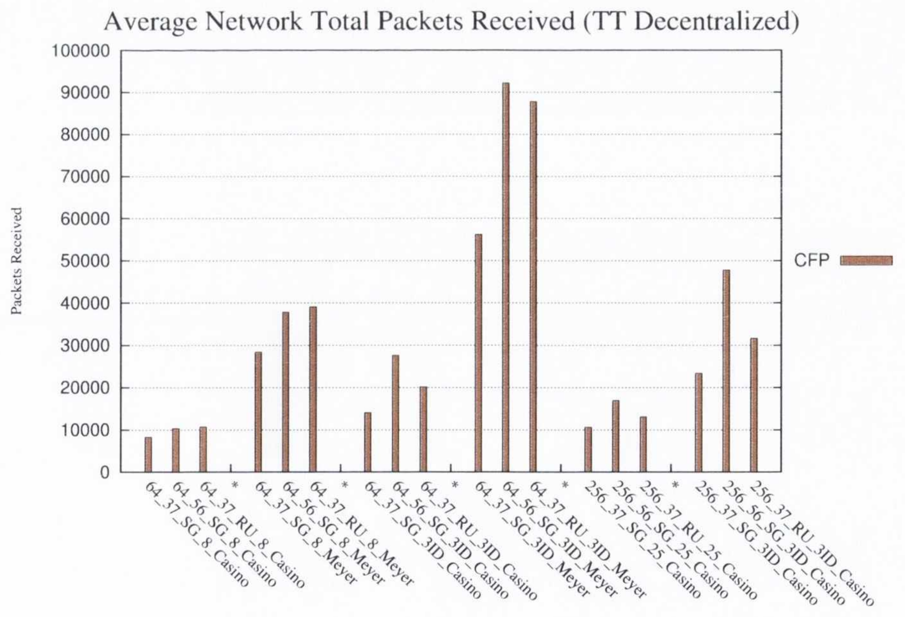


Fig. 6.28: Impact of Noise Floor and Network Density - Average Network Total Packets Sent - Average of 5 repetitions

is approximately 6 for scenarios of Casino noise floor and low network density (37 meters transmission range), while the ratio drops down to 3 when the noise floor is higher (Meyer). The ratio increases by a factor in the range of 1.5 to 4 for high density scenarios (56) within each density set, thus reaching ratios of 10 (6+4). This ratio serves as a partial indicator of the average number of neighbours per node in the network.

Furthermore, when the network scales from 64 to 256 nodes, an increment in the packets sent and received is produced mainly due to a higher number of central nodes acting as consumers, which are also aware of a higher number of nodes within the default routing scope. In scenarios of high number of consumers (3ID), the Average Network Total Packets Sent and Received increase respectively by i) 55% and 57% for 37_SG scenarios, ii) 92% and 72% for 56_SG scenarios, and iii) 57% and 57% for 37_RU scenarios. The cost of scalability in terms of average network total packets decreases with the number of consumers, despite the higher inter-consumer hop distance. The effect of scalability is further studied in Section 6.5.4 with larger networks of 400 nodes.

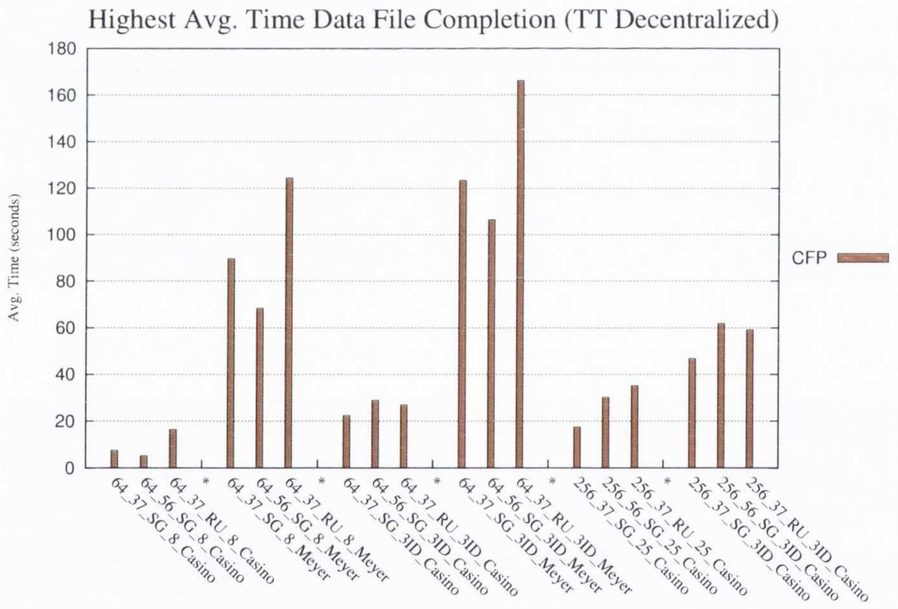


Fig. 6.29: Impact of Noise Floor and Network Density - Highest Avg. Time Data File Completion - Average of 5 repetitions

The impact of noise floor and network density on the average time to complete the acquisition of all the pieces of the data file has also been studied. The highest Average Time Data File Completion (A-TDFC) from all the nodes in the network is plotted in Figure 6.29. As expected, a high noise floor

(Meyer) impacts the time to complete the distribution process as a consequence of the lower delivery ratio, the routing delays, and the number of attempts (retrials). In absolute terms, a high number of consumers (3ID) produces higher A-TDFC values. However, in relative terms, when comparing moderate and high noise floor scenarios (Casino vs. Meyer), a higher increment in the average time to complete the data file is produced for low-density consumer scenarios with an average increment factor of 10.2, as opposed to high-density scenarios (3ID) with an average increment factor of 4.7. This is due to the higher inter-consumer distance in scenarios with a low number of consumers. High levels of noise highly increase the end-to-end latency as the path length grows. For this reason, increasing the range to 56 meters reduces the path length thus obtaining lower values of A-TDFC, despite increasing the network density. In moderate noise floor scenarios (Casino), increasing the transmission range does not have the same impact despite reducing the average path length. The irregular layout (RU) increases the average time to complete the data file when only a torrent is being distributed in the network and the impact gets exacerbated when the noise floor increases; nevertheless this highly depends on the consumer distribution in the network.

Finally, a 100% Torrents Completed ratio for all the experiments under different network density, irregular topology and high noise floor has been achieved. Again, this shows the high reliability levels of the TinyTorrents protocol.

6.5.4 On Scalability: TinyTorrents Decentralised

One of the advantages of the decentralised version of the TinyTorrents protocol is the capability to distribute the data file to distant nodes in networks of different sizes and densities. However, this scalability factor depends on the distribution of consumers in the network and the maximum routing scope. A consumer at a larger distance than the maximum routing scope from the closest consumer will be disconnected from the network of consumers and thus will not be able to fetch the data file. Moreover, a consumer will not be able to fetch the data file until at least one of the consumers within its routing scope acquires the data file. Thus, when scaling the data file distribution process, the distribution and number of consumers impact the Torrents Completed ratio, as well as the distribution of Total Packets Sent and Received by each node in the network.

For the purpose of comparing the scalability performance of the decentralised version of the TinyTorrents protocol operating with the CFP peer selection policy, the Square Grid topology layout, the Casino noise floor, and the 37 meters transmission range have been selected. The topology has been scaled from 64 to 256 and up to 400 nodes. The 3ID consumer distribution strategy has been selected as it produces a similar consumer distribution structure and enables comparison when

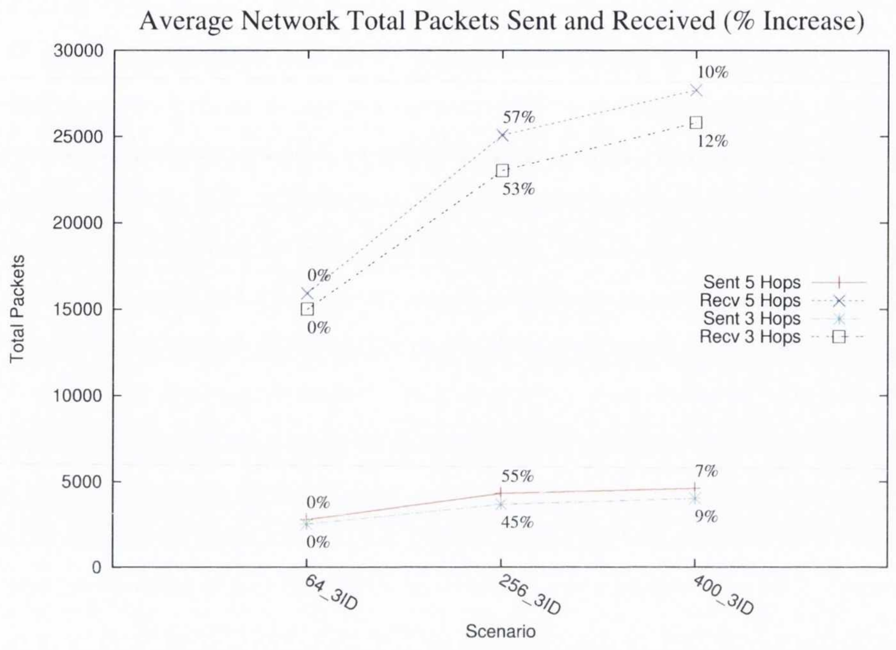


Fig. 6.30: On Scalability - Average Network Total Packets Sent and Received (% Increase)- Routing Scope of 3 and 5 Hops - CFP, TT Decentral, SG, 37m, Casino - Avg. of 5 repetitions

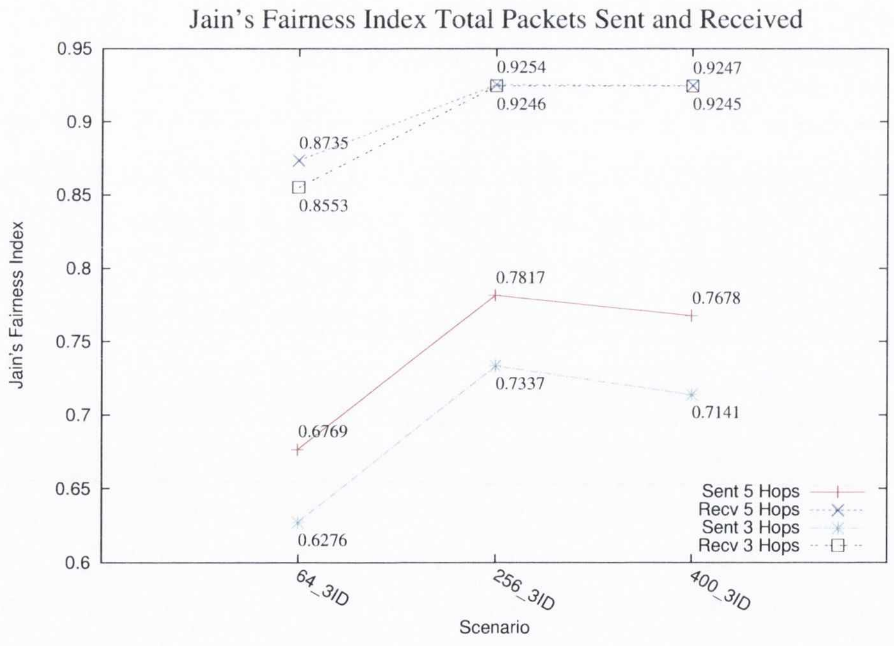


Fig. 6.31: On Scalability - Jain's Fairness Index of the Total Packets Sent and Received - Routing Scope of 3 and 5 Hops - CFP, TT Decentral, SG, 37m, Casino - Avg. of 5 repetitions

the network scales. Each scenario has been configured with two routing scopes, 3 and 5 hops, to evaluate its impact when the network scales. In order to compare the cost in packets of scaling both the network and the consumer distribution, the Average Network Total Packets Sent and Received metrics have been plotted in Figure 6.30. The figure also shows the percentage increase with respect to the previous value. Due to the high density of consumers in the 3ID strategy, the average network total packets sent and received increases with scalability. However, a higher percentage increase is obtained when scaling from the 64_3ID to the 256_3ID scenario - 57% (Recv), 55% (Sent) -, as compared to scaling from the 256_3ID to the 400_3ID scenario - 10% (Recv), 7% (Sent). This is due to the lower proportion of nodes at the edges of the square topology as the network scales. Nodes at the edges do not get as much participation in the network as central nodes and thus their proportion in the network impacts the average network total packets sent and received. In addition, a higher number of consumers fetching the data file within the routing scope increases the overall network total packets received and sent. In this regard, lowering the routing scope from 5 to 3 hops, lowers the average network total packets sent and received (see Figure 6.30) by a percentage decrease in the range of 6-9% (Recv) and 8-16% (Sent).

Additionally, Figure 6.31 shows the Jain's Fairness Index of the Total Packets Sent and Received calculated as an average of 5 repetitions of each scalable scenario (64_3ID, 256_3ID and 400_3ID) with a routing scope of 3 and 5 hops. The JFI has been computed for all the nodes in the network in terms of total packets sent and received to assess the degree of fairness in the overall network communication process for each scenario. However, it has to be noted that the JFI of the Total Packets Sent and Received is impacted by the number and distribution of consumers in the network. Thus, the metric is not a conclusive indicator of the overall network fairness in the distribution process, but it serves as a metric to evaluate the scalability of a network when it grows with a scalable consumer distribution, such as the 3ID. The 3ID consumer distribution strategy grows in a fair and scalable manner suitable for comparison, despite not producing the exact same geometric consumer distribution in all the scalable topologies - strategies 64_3ID and 400_3ID produce the same consumer distribution shape, while 256_3ID generates a different layout (see Figures 6.9, 6.11, 6.13). As expected, Figure 6.31 shows higher index of fairness for the total packets received than for the total packets sent. This is due to the broadcast nature of the wireless medium. In addition, higher values of JFI are obtained for the 256 and 400 nodes topologies as compared to the 64 nodes topologies. This is mainly a consequence of the lower proportion of nodes at the edges and the higher number of consumers at the centre of the network in the 256 and 400 nodes topologies. The high values of JFI in the total packets received (Recv) are due to the high density of consumers in the network, but they also confirm that the number of packets received is balanced. The JFI of the total packets sent (Sent) has lower values due to the fact that the 3ID consumer distribution strategy places a great proportion of consumers at 1 hop distance and consequently reduce the participation of non-consumer router nodes and hence their packet sending ratio. By the same token, the higher the routing scope, the higher the likelihood of non-consumer nodes acting as routers and thus contributing to increase the Jain's fairness index of the Total Packets Sent (see "Sent 5 Hops" vs. "Sent 3 Hops" in Figure 6.31).

Overall, results indicate that the decentralised version of the TinyTorrents protocol scales at a low cost in terms of total packets received and sent, which depends on the routing scope. It also exhibits high values of fairness in the communications process in the network, despite the network scales.

Additionally, the distribution of Piece Messages Sent, Total Packets Received and Average Time Data File Completion per node in the network has been plotted for two randomly selected experiments for the 256_3ID and 400_3ID scenarios. The distribution of Piece Messages Sent in the 256_3ID and 400_3ID scenarios is shown in Figures 6.32 and 6.33 respectively. Both of the scenarios show high values of fairness in the piece message distribution with a JFI of 0.85 for the 256_3ID scenario and 0.79 for the 400_3ID scenario. In both scenarios, a reduced number of consumers produce a higher number of Piece Messages Sent as compared to the rest of the consumers. These consumer nodes are popular candidates in the Closest-first Piece (CFP) peer selection process due to the stochastic process employed to generate the topology. The process can generate a connectivity map where some of the nodes connect to a higher set of neighbour nodes, despite forming a square grid layout. These nodes, rather than being a hot spot where congestion is produced, are points of high connectivity which distribute a high number of pieces of data to proximate nodes for overall efficiency in terms of communication. Furthermore, these nodes do not necessarily make an impact in the distribution of the Total Packets Received as it can be seen in Figure 6.34 (256_3ID) and Figure 6.35 (400_3ID). Indeed, the intermittent connectivity behaviour of some of the neighbours at the edge of communication of a node produces a higher impact in the Total Packets Received by a node within the area; this effect can be noted in some of the areas in the network indicated by a higher number of Total Packets Received. Nevertheless, the network balances out in terms of packets received, mainly due to the high density of the consumer distribution (3ID), and the fact that the decentralised version of the TinyTorrents protocol scales by distributing pieces of the data file locally, within a routing scope. On the other hand, the impact of scaling the network on the average time to complete the data file (A-TDFC) is shown in Figure 6.36 (256_3ID) and Figure 6.37 (400_3ID). Consumer nodes placed within the triangle area formed by coordinates (0,0), (375,0), (0,375) take

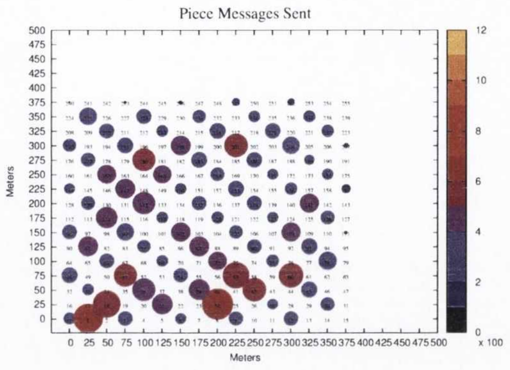


Fig. 6.32: Piece Messages Sent - 256_3ID, CFP, TT Decentral, SG, 37m, Casino - JFI 0.85

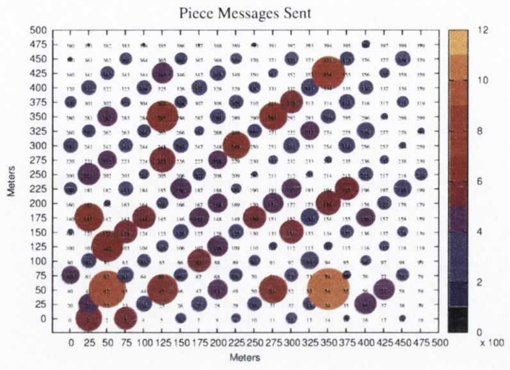


Fig. 6.33: Piece Messages Sent - 400_3ID, CFP, TT Decentral, SG, 37m, Casino - JFI 0.79

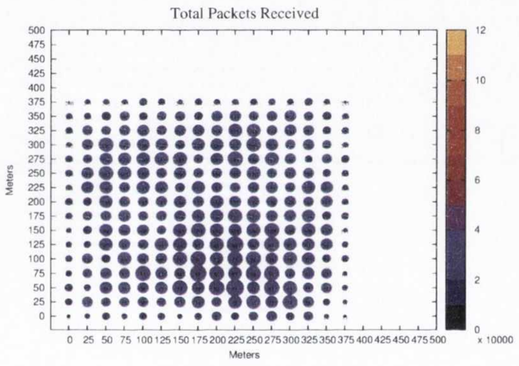


Fig. 6.34: Total Packets Received - 256_3ID , TT Decentral, SG, 37m, Casino

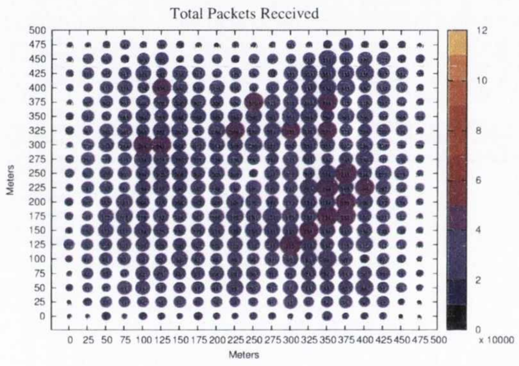


Fig. 6.35: Total Packets Received - 400_3ID , TT Decentral, SG, 37m, Casino

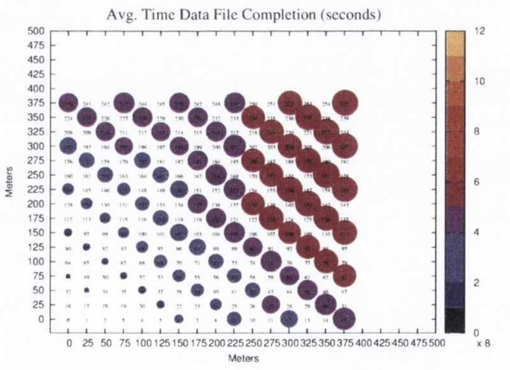


Fig. 6.36: Avg. Time Data File Completion (seconds) - 256_3ID , TT Decentral, SG, 37m, Casino

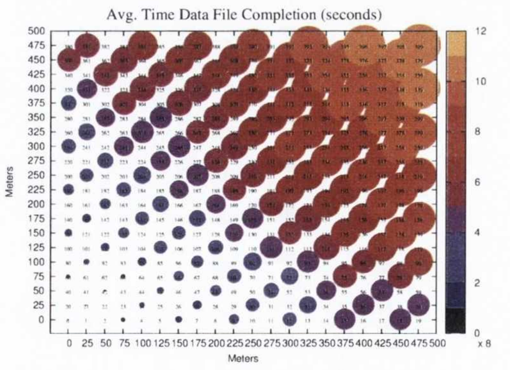


Fig. 6.37: Avg. Time Data File Completion (seconds) - 400_3ID , TT Decentral, SG, 37m, Casino

on average similar times to complete the data file in both scenarios of scalability. However, nodes outside of this area show higher average times in the 400 node scenario due to the higher number of consumers fetching the data simultaneously. For instance, in the 256 nodes topology, node 255 takes on average 45 seconds to complete the file, while the node at the same position in the 400 nodes topology, node 315, employs on average 65 seconds. Moreover, the figure shows the direction and the smooth progression of the data file distribution process. This is achieved via the use of i) the dissemination strategy of the torrent file (which enables the discovery of nearby trackers), and ii) the partial tracker admission mechanism (which only provides the peer list when the node itself is a seeder of the data). The latter mechanism enables the distribution process to self-regulate, thereby providing a progressive and localized distribution of data which seeks to minimize the overall number of messages sent and received. This reduces node congestion and wireless contention, and consequently reduces the overall time for data file completion in the network.

Finally, the impact of scalability has also been studied in networks with a low density of consumers. For this purpose, two experiments have been carried out with the 256_25 and 400_21 consumer distribution strategies (see Figures 6.10 and 6.12). While the 256_25 and the 400_21 strategies have a low number of consumers as compare to the 3ID, the 400_21 strategy presents a long inter-consumer distance of up to 5 hops as compared to the 256_25. This difference enables testing of the decentralised distribution protocol under long inter-consumer path lengths where communication is less reliable and the CFP strategy has a higher impact in the selection of closer peers. For this purpose, the distribution of Piece Messages Sent in the 256_25 and 400_21 scenarios is shown in Figures 6.38 and 6.39 respectively. With a Jain's Fairness Index of Piece Messages Sent of 0.65, the 256_25 scenario exhibits a high degree of fairness which gets reduced by the high number of Piece Messages Sent by node 52. This node acts as a hot spot due to its location in the consumer distribution and the direction in which the distribution process expands. Node 52 acts as a key closer consumer and partial tracker for a set of consumer nodes which will in turn foster the distribution of the data file to the rest of the consumers in the network. This indicates the importance of the consumer distribution strategy in the efficiency of the data file distribution process. On the other hand, the large inter-consumer path length of the 400_21 strategy in Figure 6.39 produces a high degree of fairness with a JFI value of 0.72. The appropriateness of selecting closer consumers of the CFP peer selection strategy, together with the layout distribution of consumers, increases the fairness in the distribution process. Having a fair distribution of Piece Messages Sent when the inter-consumer path length is relatively high indicates the effectiveness of the routing protocol, mainly in providing accurate information on the path length. In contrast, the distribution of Total Packets Received in

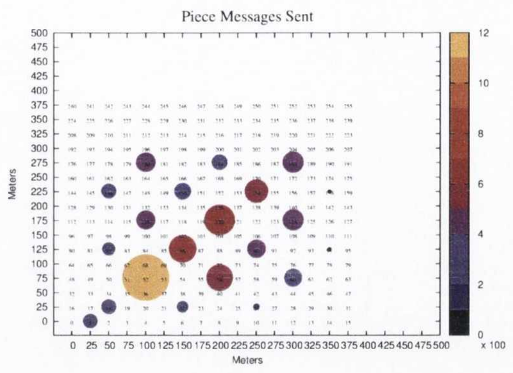


Fig. 6.38: Piece Messages Sent - 256_25 , CFP, TT Decentral, SG, 37m, Casino - JFI 0.65

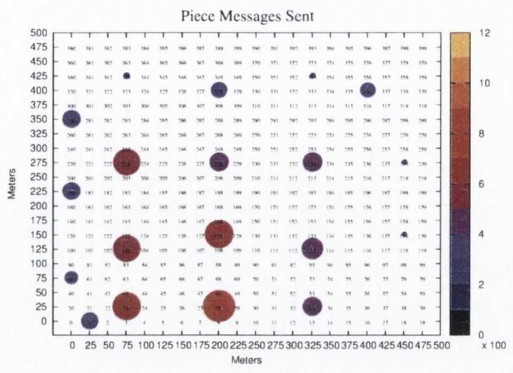


Fig. 6.39: Piece Messages Sent - 400_21 , CFP, TT Decentral, SG, 37m, Casino - JFI 0.72

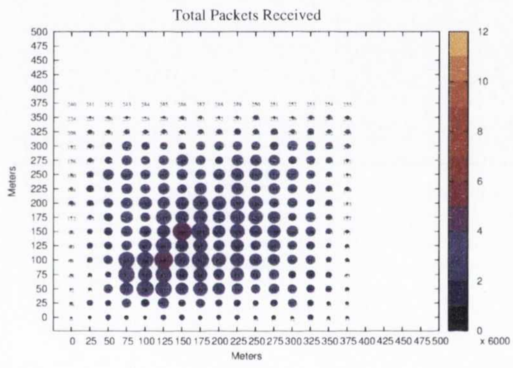


Fig. 6.40: Total Packets Received - 256_25 , CFP, TT Decentral, SG, 37m, Casino

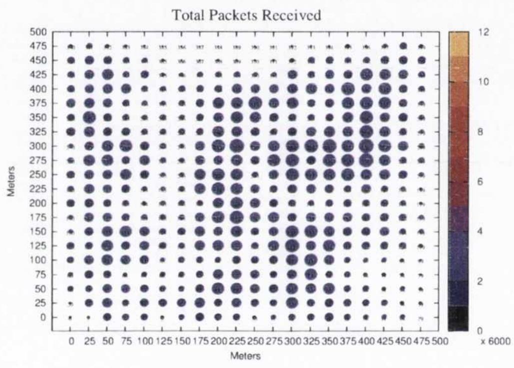


Fig. 6.41: Total Packets Received - 400_21 , CFP, TT Decentral, SG, 37m, Casino

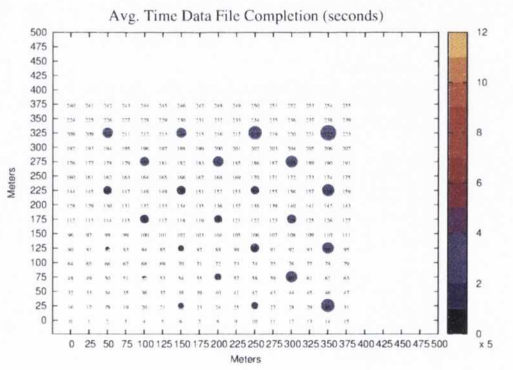


Fig. 6.42: Avg. Time Data File Completion (seconds) - 256_25 , CFP, TT Decentral, SG, 37m, Casino

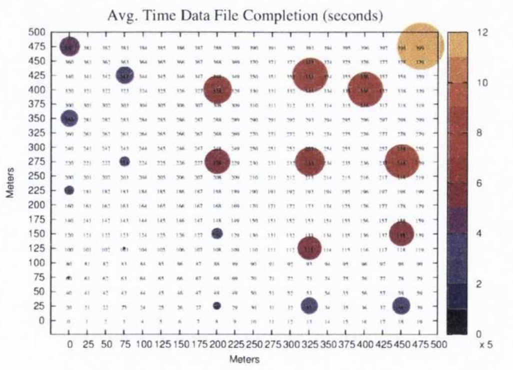


Fig. 6.43: Avg. Time Data File Completion (seconds) - 400_21 , CFP, TT Decentral, SG, 37m, Casino

the network in Figures 6.40 and 6.41 indicate that areas with a higher number of packets received do not necessarily correspond to the position where the node/s with higher number of Piece Messages Sent are placed, but to the areas where the density of consumers is high. This can be clearly seen in the figures, where the 400_21 scenario has lower levels of Total Packets Sent than the 256_25 scenario due to the lower density of consumers and the higher inter-consumer path length. This also proves the capability of the TinyTorrents protocol to scale when the density of consumers is low, only limited by the routing scope. Nevertheless, a higher average time in completing the data file is produced in the 400_21 scenario. This is mainly due to the long path lengths, which increases the unreliability and latency in the routing process, and the low density of consumers. For comparison, node 399 takes an average time to complete the data file (since the torrent file is received) of 52 seconds in the 400_21 scenario (see Figure 6.43), while in the 400_3ID scenario (Figure 6.37) the node takes 75 seconds. In scenarios of high consumer density, the data file takes on average a higher time to reach the same point/area in the network than in scenarios of low consumer density, despite the closer proximity amongst consumer nodes.

6.5.5 Effect of Multiple Producers and Random Consumer Distribution

Heretofore, the TinyTorrents decentralised distribution process has been evaluated from the point of view of having only one producer in the network. This has enabled the study of i) the degree of fairness of each peer selection strategy, ii) the scalability factor of the system, and iii) the reliability and efficiency of the protocol in isolation under different conditions. However, having simultaneous distribution processes of torrent files, either coming from the same or from different producers, can introduce a delay in the data file acquisition process. This will occur when consumers are forced to wait to acquire pieces of data from consumers which are in the process of acquiring other torrent files. While a consumer can act as a seeder and partial tracker for a completed data file, as long as this is stored in memory, consumers handle torrent files one at a time and consequently might delay other consumers in their data acquisition processes. To minimize this delay, a consumer can progress through fetching the next torrent file in queue if no peer list can be obtained for the current torrent file. This way, the consumer can participate in another torrent file distribution process for which data is available in nearby consumers, while avoiding those torrent files for which the distribution process has still not reached the node's routing scope vicinity. Once a torrent retrieves a valid peer list, the distribution process for the data file runs until completion, or until a time threshold is reached. For this reason, the effect of having simultaneous distribution processes for different torrent files and from producers placed at distant points in the network needs to be evaluated.

For this purpose, the TinyTorrents decentralised protocol operating with the CFP peer selection strategy has been evaluated in the next scenario: 400 nodes in a square grid (SG) layout, a node transmission range of 37 meters and a moderate noise floor (Casino). To introduce randomness in the distribution of consumers in the network and to test the reliability of the unstructured discovery mechanisms, the default routing scope has been set to 3 hops with a maximum increment of 2 hops, and the RAND_10 consumer distribution strategy has been employed. On reception of a torrent file, the RAND_10 strategy appoints the node as a consumer with a 10% likelihood. Only the first torrent file received from node 1 has been used to define which nodes act as consumers in the network. Once selected, the set of consumers remain constant for the rest of the simulation, acquiring data files from all the torrent files received from all the producers. A total of 5 producers (nodes 1, 57, 209, 342, 357), placed at the corners and in the centre of the topology, generate a data file every 5 minutes for a total of 20 files.

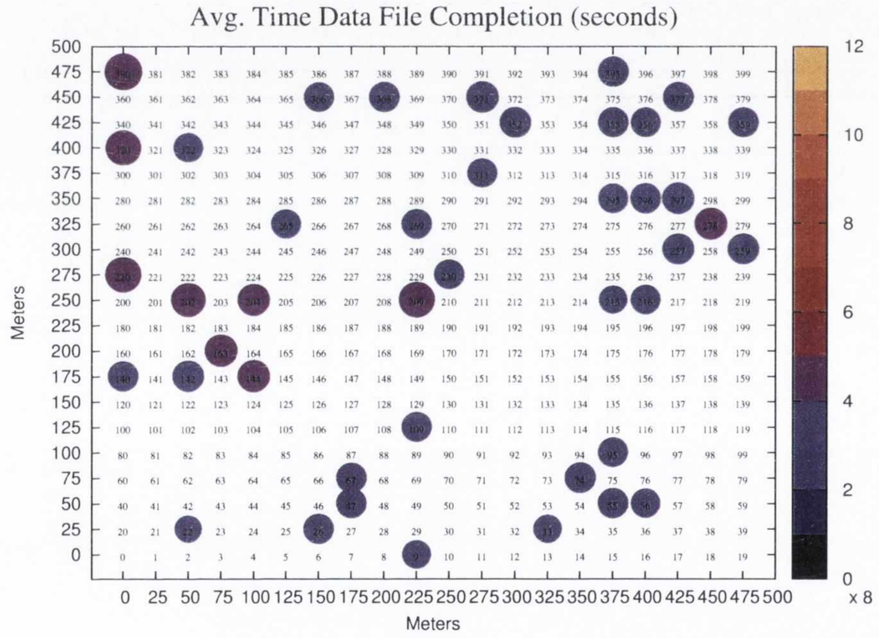


Fig. 6.44: Avg. Time Data File Completion (seconds) - 400_RAND_10 , CFP , 5 Producers , TT Decentral, SG, 37m, Casino

Figure 6.44 shows the average time each node takes to acquire each data file (A-TDFC). This takes into account the time elapsed between the selection of the torrent from the queue (for which a peer list is retrieved), and the completion of the data file; the average time for all the torrents received is calculated. On average, consumers take between 27 and 36 seconds to complete data files. The low average time to complete the data file, as compared to one producer scenarios such as the 400_21 in Figure 6.43, is a result of having multiple torrents in the queue and selecting one for which data pieces are available within the routing scope. This mechanism avoids incurring an overall delay in the distribution process, thereby increasing the likelihood of completing data files before the torrent cancellation time threshold is reached. However, the number of producers and their positions in the network impact the average time to complete the data file. Additionally, Figure 6.44 also shows the distribution of consumers in the network produced by the RAND_10 strategy which, in this particular experiment resulted in a total of 43 consumers, one of them being also a producer (node 209). For all consumers, a 100% Torrents Completed ratio was obtained; all consumers received 100 torrent files in total (20 from each producer), with the exception of consumer-producer node 209 which received 80 - as it does not receive and acquire its own generated torrent data files.

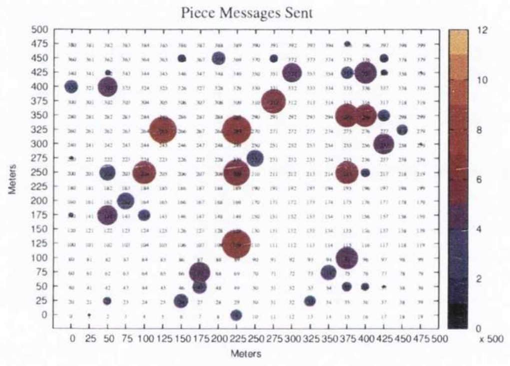


Fig. 6.45: Piece Messages Sent - 400_RAND_10 , CFP , 5 Producers , TT Decentral, SG, 37m, Casino - JFI 0.77

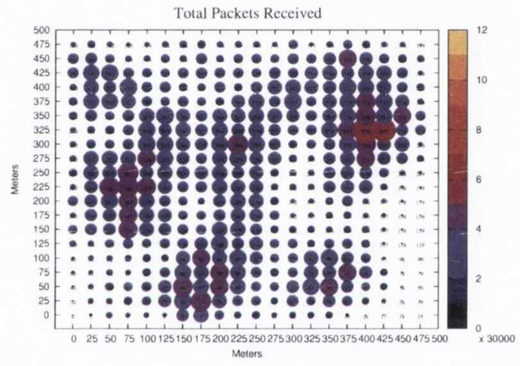


Fig. 6.46: Total Packets Received - 400_RAND_10 , CFP , 5 Producers , TT Decentral, SG, 37m, Casino

Figures 6.45 and 6.46 show the distribution of Piece Messages Sent and Total Packets Received in the network respectively. As in previous experiments, the distribution of Piece Messages Sent for all the torrent files in the network tends to balance according to the position of the consumers with respect to each other. However, in this scenario, both the distribution of consumers and producers impact the overall number of Piece Messages Sent. Each torrent distribution process has its own direction when expanding which impacts the set of partial trackers and peers from which a consumer acquires the data file. In the current scenario, the set of consumers remains the same, and therefore placing producers at disparate points in the network increases the likelihood of a consumer selecting different partial trackers and peers for each distribution process. This contributes to increase the overall fairness in the participation of consumers in all the distribution processes; this argument is supported by the high value of 0.77 obtained for the Jain's Fairness Index of Piece Messages Sent in the current experiment. Additionally, consumers having a higher number of Piece Messages Sent act as link points for a group of consumers within their proximities. For instance, the position of node 95 and 109 make them key points for the distribution of data coming from producers placed at the top of the topology. The effect of having a group of consumers together in the same area increases the number of Total Packets Received (see Figure 6.46) and thus contributes to increase the wireless medium congestion.

A second set of experiments have been carried out for the purpose of testing the effect of multiple producers in the network transmitting at the same interval of time, thus generating simultaneous torrent data acquisition processes. These tests have investigated the impact of having similar number of consumers and producers, where all consumers acquire data generated by all the producers. For this purpose, the TinyTorrents decentralised protocol, operating with the CFP peer selection strategy, has been evaluated in the following scenario: 256 nodes in a square grid (SG) layout, a node transmission range of 37 meters and a moderate noise floor (Casino). The default routing scope has been set to 3 hops with a maximum increment of 2 hops, and the RAND_10 consumer distribution strategy has been employed to randomly select the consumers. Additionally, instead of appointing producer nodes beforehand, at the beginning of the simulation all nodes decide with a 10% likelihood whether to become producers for the remaining duration of the simulation. Figure 6.47 shows a resulting distribution of consumers and producers from applying the process; note that some nodes become both consumers and producers. Producers randomly publish a file every 5 minutes to a total of 10 files.

Figure 6.48 shows the average time each node takes to acquire each data file (A-TDFC). On average, consumers take between 8 and 18 seconds to complete data files. When compared to the previous experiment in Figure 6.44, where consumers take between 27 and 36 seconds, a significant latency improvement can be observed. However, this improvement is a consequence of having a higher number of torrents in the queue available when the active torrent fails to be completed; this does not take into account the time from which a failing torrent is put back in the queue until it is selected again for data fetching. While the latency reduction is mainly due to the effect of having multiple torrents at the queue, the different consumers and producers distribution also impacts this result.

Finally, Figure 6.49 shows the distribution of Piece Messages Sent in the network where a low JFI value of 0.47 can be observed. This indicates the impact that multiple producers have on the

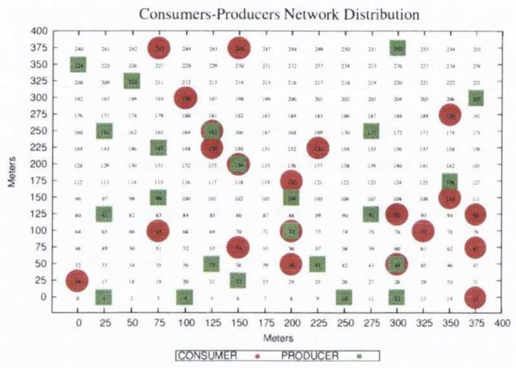


Fig. 6.47: Consumer-Producer Distribution - 256_RAND_10 , CFP , 23 Producers , TT Decentral, SG, 37m, Casino

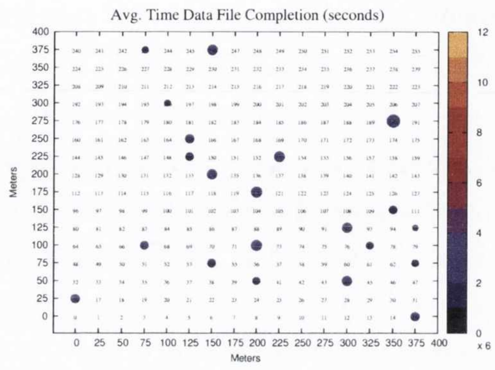


Fig. 6.48: Avg. Time Data File Completion - 256_RAND_10 , CFP , 23 Producers , TT Decentral, SG, 37m, Casino

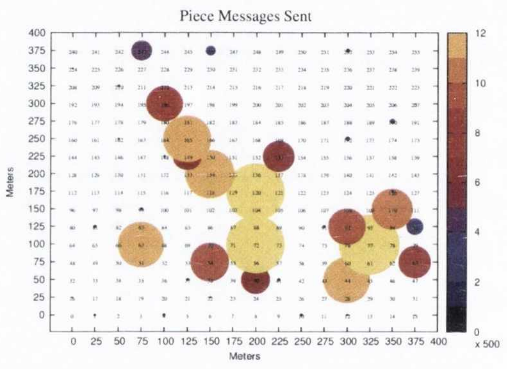


Fig. 6.49: Piece Messages Sent - 256_RAND_10 , CFP , 23 Producers , TT Decentral, SG, 37m, Casino - JFI 0.47

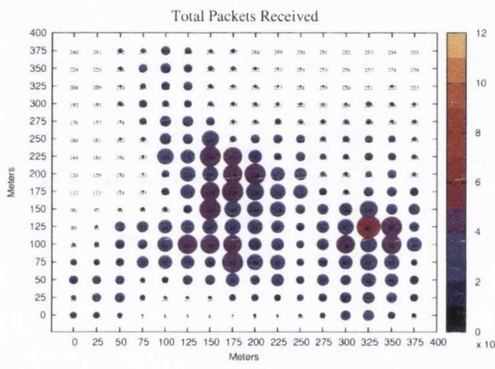


Fig. 6.50: Total Packets Received - 256_RAND_10 , CFP , 23 Producers , TT Decentral, SG, 37m, Casino - JFI 0.66

data traffic activity of key consumer nodes, which serve as gateways to connect different consumer-populated areas in the network. The low fairness index is due to the CFP peer selection policy, which employs proximity as the primary metric in the selection at the benefit of achieving better communications efficiency. The later can be seen in Figure 6.50 which shows the distribution of Total Packets Received in the network. The effect of having consumers and producers grouped in the same area increases the number of Total Packets Received, which increases further if they are positioned at key locations serving as gateways between different areas. Nevertheless, the efficacy of having a selective data dissemination protocol is clearly visible, which concentrates the data traffic amongst the web formed by the set of consumers and producers.

6.5.6 Performance Comparison of Dissemination Protocols

This section analyses the performance of TinyTorrents when compared to two of the most popular dissemination protocols for Wireless Sensor Networks, i.e. DIP [211] and DHV [212] (see Section 2.4.5). DIP and DHV have been designed with the goal of reprogramming the network in an epidemic fashion such that consistency of data is achieved amongst all the nodes in the network. This mechanism of dissemination does not follow the selective data dissemination approach that underpins the design of TinyTorrents. While DIP and DHV are good solutions for network reprogramming purposes, TinyTorrents provides for the distribution of data amongst a subset of consumer nodes in the network which express interest in the data, and thus the traffic load in the network depends on the consumer distribution. The higher the number of consumer nodes in the network, the more efficient the use of an epidemic dissemination protocol. However, consider a scenario where consumers and producers are placed along a defined area in the network for the purposes of performing some sort of sensing-actuating activity, for instance at the edge of the network. Nodes in other areas would not need to receive data from these producers and consumers, and the data transfer should only occur within this area and, most specifically, amongst the overlay of consumers and producers. TinyTorrents provides this type of selective data dissemination where router nodes do not necessarily need to receive all the pieces of a file of data, thus also increasing the security of the process. These advantages need to be taken into consideration when comparing TinyTorrents with epidemic dissemination protocols which have been shown to be very efficient in terms of communication.

For the purpose of providing a fair comparison, an application running on top of DIP and DHV has been developed in TinyOS 2.1 which has been configured to disseminate the same amount of data as the TinyTorrents application. One producer, node 1, has been selected to disseminate 20 files of data, each of size 256 bytes. The Piece message (see Figure 5.7 in Section 5.2), which contains the data in the TinyTorrents protocol, has been encapsulated in the payload of these protocols. The Piece message has been configured to contain 16 bytes of data and has been fitted in a packet by increasing the payload to that used for TinyTorrents packets. DIP and DHV operate by updating versions of data items where the most up-to-date version is disseminated with the goal of maintaining consistency in the network. Data packets corresponding to an old-version are not disseminated when a new version, i.e. an update, for the data item is, or has been, received at the node. Taking this into consideration, the application assigns each of the 16 pieces of a file a different key, such that they are described as different data items. In this way, different versions of each piece are injected into the network when a new file is disseminated. The same delay employed in TinyTorrents for the the dissemination of files by the producer is also in place which guarantees that no pieces of data are being transferred when a new version, i.e. a new file, is starting to be disseminated. This mechanism enables comparison with TinyTorrents where each piece of data is distributed as a different data item. Note that one of the drawbacks of DHV and DIP is the requirement for defining the type of data items to be disseminated at compilation time, such that all the nodes know beforehand what type of data items are to be received.

Additionally, DIP and DHV employ the Trickle algorithm [99] to regulate the periodicity of broadcasting packets. Trickle exponentially increases the packet broadcast interval to reduce communications according to the activity in the neighbourhood, while decreasing the interval towards a minimum value, T_l, when updates need to occur. Setting T_l to a low value will increase the communications at the benefit of reducing the latency in the dissemination process. T_l has been set to 30 ms; a lower value produces similar results in terms of latency and employs more packets. In this way, both DHV and DIP have been configured to achieve a very high performance in terms of latency with a low overhead in communications. To further improve latency on DIP and DHV, pieces are sequentially published by the producer with a small delay of 50 milliseconds. On the other hand, the TinyTorrents protocol employs a set of control messages, such as handshakes and peerlist request messages, to regulate the dissemination process. This enables control of the distribution of the traffic but also increases the latency and packet overhead in the dissemination of data.

The scenario selected for the comparison comprises 64 nodes in a square grid (SG) layout, a node transmission range of 37 meters and a moderate noise floor (Casino). To show the benefits of the selective data dissemination process of the decentralised version of the TinyTorrents operating with the CFP peer selection strategy, two consumer distributions have been employed: i) the 64_CONS8 distribution which employs 8 consumers placed across the network, and ii) the 64_CONS8E distribution, a new distribution created for this purpose which places 8 consumers along the bottom edge of the network (nodes 2, 4, 7, 10, 12, 14, 15, 19) (see Figure 6.52). Consequently, 4 protocols configurations have been evaluated: i) TinyTorrents with the CONS_8 distribution, ii) TinyTorrents with the CONS_8E distribution, iii) DIP with $T_l = 30$, and iv) DHV with $T_l = 30$. Note that, as DHV and DIP are epidemic dissemination protocols, they deliver data to all nodes in the network and thus all nodes are consumers.

Latency in the dissemination process has been studied with the Average Time Data File Completion (A-TDFC) in Figures 6.51 and 6.52 for TinyTorrents, in Figure 6.53 for DIP and in Figure 6.54 for DHV. For DIP and DHV, the A-TDFC for the furthest nodes from node 1, i.e. the producer, is of the order of 3 to 4 seconds, where DHV shows faster file completion times than DIP. TinyTorrents also shows a faster data file completion time for closer nodes such as 0 and even 4 in Figure 6.51,

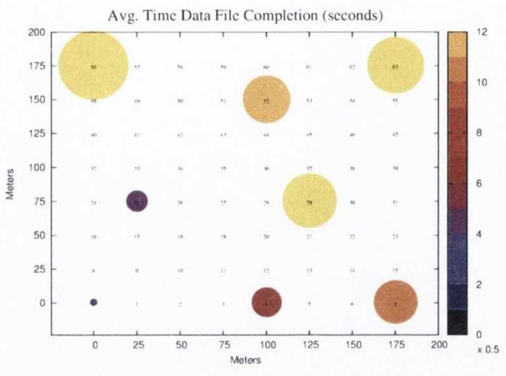


Fig. 6.51: TT Decentral - Avg. Time File Completion (seconds) - CONS_8

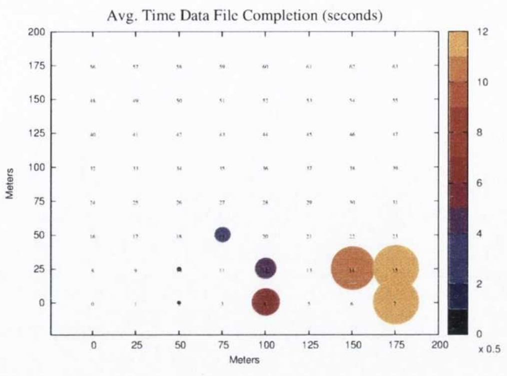


Fig. 6.52: TT Decentral - Avg. Time File Completion (seconds) - CONS_8E

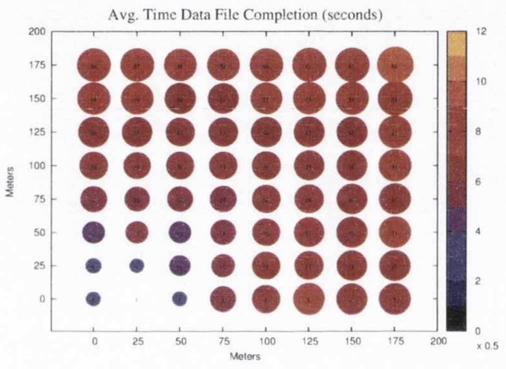


Fig. 6.53: DIP - Avg. Time File Completion (seconds) - $T_l = 30$

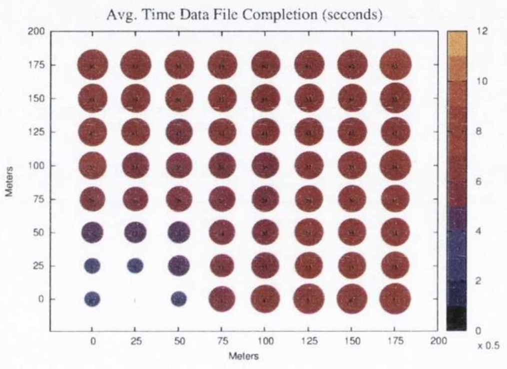


Fig. 6.54: DHV - Avg. Time File Completion (seconds) - $T_l = 30$

and nodes 2, 10, 12 and 19 in Figure 6.52. However, as the file is disseminated through the network, the data file completion time increases, producing higher values than with DIP and DHV. This is mainly due to the delay introduced by the consumers when waiting for other consumers within their scope to retrieve the file. For example, node 7 takes double the time to acquire the file than it takes with DIP and DHV. This result demonstrates that epidemic algorithms can offer faster solutions for delivering data, while employing less control packets, at the cost of disseminating the file throughout the whole network.

Figures 6.55 and 6.56 show the Total Packets Sent and Received respectively for each of the 64 nodes in the network for the 4 protocols configurations. DIP and DHV show evenly distributed network traffic, both for Total Packets Sent and Received. This is due to the Trickle algorithm which regulates packet transmissions according to the neighbourhood activity while leveraging broadcast packets for the update of data items. DHV requires less packets than DIP, while showing similar

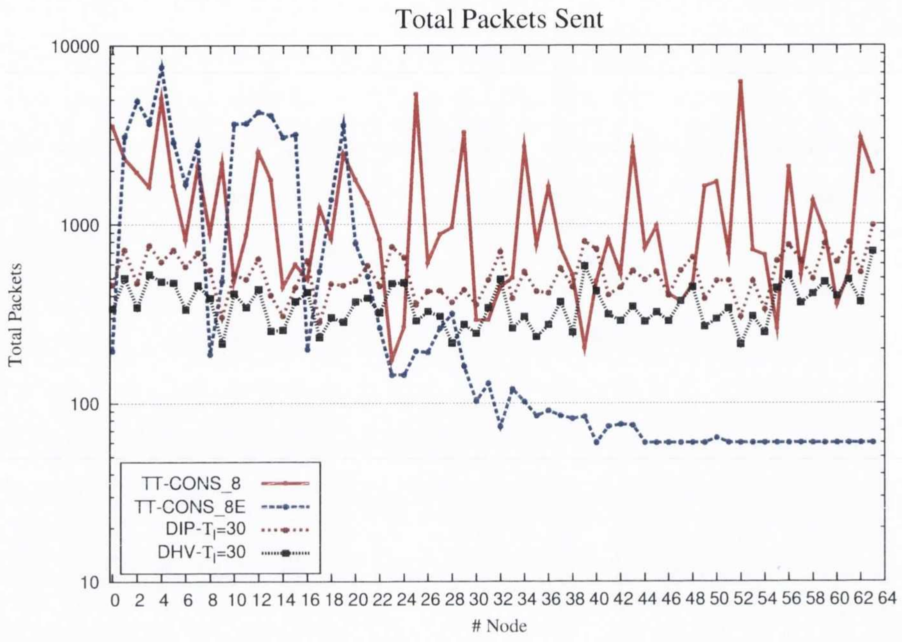


Fig. 6.55: Dissemination Protocols Comparison: TT, DHV and DIP - Total Packets Sent - 64 Nodes, 1 Producer , SG, 37m, Casino, TT Decentral, CFP

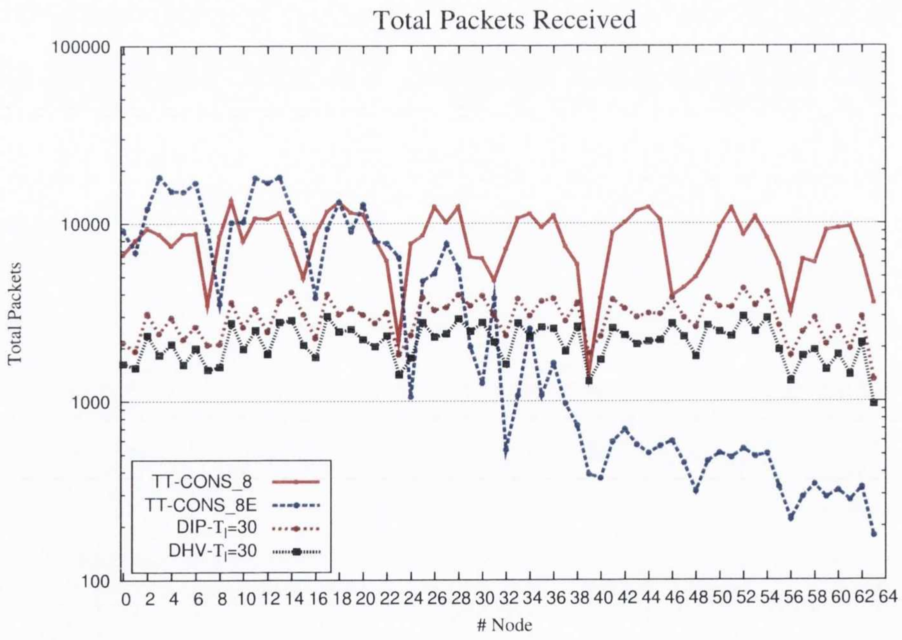


Fig. 6.56: Dissemination Protocols Comparison: TT, DHV and DIP - Total Packets Received - 64 Nodes, 1 Producer , SG, 37m, Casino, TT Decentral, CFP

behaviour. By comparison, TinyTorrents employs higher number of Total Packets Sent, mainly by those nodes which fully participate in the data dissemination process, both router and consumer nodes. However, TinyTorrents sends fewer packets than DIP and DHV from those nodes placed in areas where there is a lower concentration of consumers. This effect is more clearly visible with the CONS_8E consumer distribution, where nodes are placed at the edge, with the middle top area of the network (node 30 to 63) not receiving data and only receiving torrent messages. This shows the efficiency of TinyTorrents as a selective data dissemination protocol and the unsuitability of epidemic dissemination in this case.

6.5.7 Performance Evaluation on a 64-Node Testbed

This section explores the performance of the decentralised version of the TinyTorrents protocol in a testbed of 64 telosB [30] wireless sensor devices deployed in a home environment (see Figures 6.57 and 6.58). In order to achieve multi-hop communication in the test environment, a lower bound on the transmission power of the transceiver was set. According to the CC2420 chip specifications [29], the RF output power register can be set to a minimum nominal value of 3 (delivering -25 dBm). However, the TinyOS libraries provide for setting the minimum real value to 1, delivering an output power of less than -25 dBm. The telosB motes were configured to operate with an RF output power setting of 1 which reduced the communication range to a distance of, at most, 40 centimeters when they where placed on top of a non-conductive surface, i.e. carpet over cement (see Figures 6.57 and 6.58). It was noted that the positions of both the antenna and the usb metal connector with respect to the receiving neighbour node/s impact the propagation of the signal and thus the packet reception rate. This irregularity may be due to i) the antenna being integrated at the edge of the

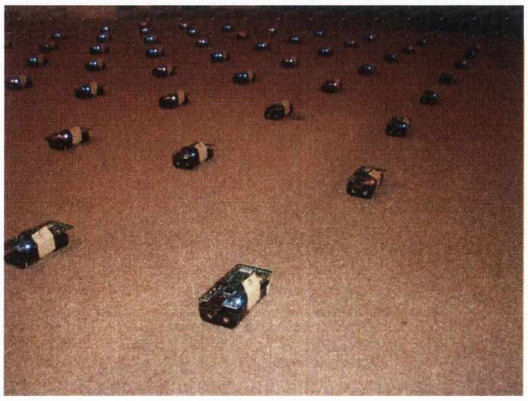


Fig. 6.57: Testbed View 1

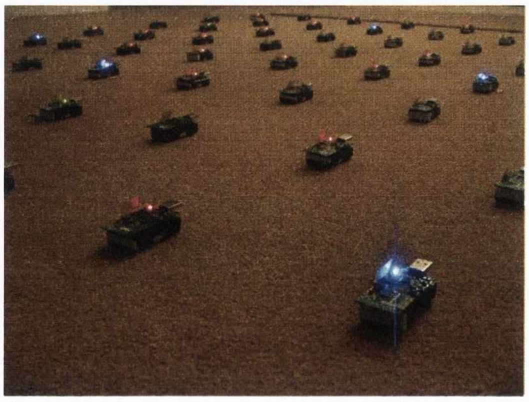


Fig. 6.58: Testbed View 2

telosB board and ii) the increase in the signal propagation produced by the usb metal connector. Taking these factors into account, two different topologies of 64 telosB motes were formed: i) the Square Grid (SG) layout, and ii) the Irregular Grid (IG) layout. The Square Grid layout (see Figure 6.59) follows the same square structure employed in previous tests in the simulator. Motes in the topology have been placed at 1-hop range to immediately adjoining nodes. On the other hand, the protocol has been evaluated operating on a Irregular Grid (IG) topology (see Figure 6.60) where nodes have been shifted randomly from the initial Square Grid layout, leaving three clearly defined holes (gaps). While this topology has a similar structure to the Random Uniform (RU) layout, the latter exhibits a lower degree of irregularity.

The decentralised version of the TinyTorrents protocol has been evaluated with the Closest First PieceRemaining-based peer selection algorithm (CFP). Three producer nodes have been appointed to periodically publish data files of 255 bytes length (the maximum permitted). Node 1, 38 and 49 have been selected as the three producers (3PRO) due to their distant location from each other in the network. Each producer publishes 20 torrents (data files) every 300 seconds. Producers are randomly scheduled to start publishing the next torrent within the first 20 seconds once the 300 seconds interval is elapsed. This generates simultaneous data file distribution processes for each of the 20 torrent intervals. Two deterministic consumer distribution strategies have been employed, the 64_8 and the 64_3ID, which enable comparison under different consumer densities, network traffic burden and inter-consumer hop distances. The size of the queues which store torrent information and data files have been reduced to fit the available RAM memory in the telosB architecture, i.e. 10KB. A maximum of 4 data files are stored at each consumer/producer. This could impact the success of the distribution process for a consumer/s seeking to fetch an old torrent which has been deleted/discarded from the queue of other consumers in order to accommodate a new torrent (i.e. more than 4 torrents being distributed at the same time). However, this has been avoided with the 300 seconds inter-torrent publication time which establishes a sufficient time for all the consumers in any distribution to acquire the 3 data files from nodes 1, 38 and 49. Each experiment took, from the time the producer motes were started until the acquisition of the last torrent, an average of 6300 seconds (1h 47'). Three repetitions of each experiment have been carried out at different times of the day - morning (MORNI), evening (EVENI), overnight (NIGHT) - to account for different scenarios of background radio activity. As in previous tests, the default discovery scope has been set to 5 hops. The length of the transmitted packet, MAC Protocol Data Unit (MPDU), is of 57 bytes, comprised of the CC2420 header (11 bytes) and the payload (46 bytes). The payload contains the UMG routing protocol control parameters and the TinyTorrents protocol message structures with

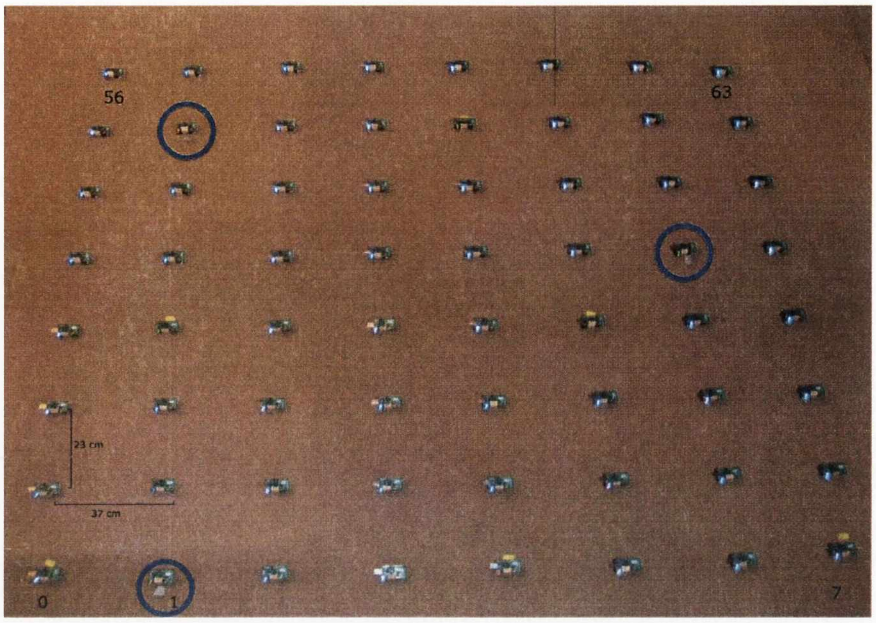


Fig. 6.59: Testbed - Square Grid - Producer nodes (1,38,49) are highlighted with a circle.

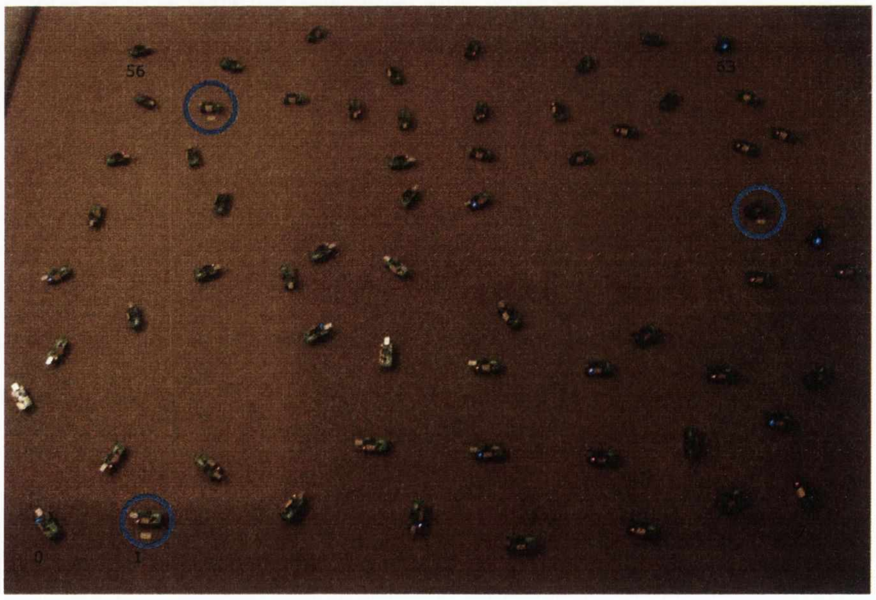


Fig. 6.60: Testbed - Irregular Topology - Producer nodes (1,38,49) are highlighted with a circle.

the corresponding data. At the end of each experiment, a base station node sequentially queries each of the motes in the network at full RF transmission power to acquire the set of metrics.

The experiment scenarios have been performed in the 64 telosB testbed at different times of the day (morning, evening, overnight) on consecutive days:

1. Experiment with 64_8 consumer distribution and Square Grid (SG) topology,

2. Experiment with 64_3ID consumer distribution and Square Grid (SG) topology, and

3. Experiment with 64_3ID consumer distribution and Irregular Grid (IG) topology.

Statistics are computed for the 9 experiments performed in the testbed. Figure 6.61 shows the Jain's Fairness Index (JFI) of the number of Piece Messages Sent. For all scenarios, JFI values above 0.6 are obtained which indicates a good degree of fairness in the distribution process between consumers, in line with results obtained in the simulator. The JFI of Piece Messages Sent for the 64_8 consumer distribution are close to those for the 64_3ID consumer distribution for the Square Grid (SG) topology. This situation only occurred in the simulator experiments when testing was done under heavy noise (Meyer). On the other hand, experiments with the Irregular Grid (IG) topology produce higher degree of fairness (JFI of Piece Messages Sent), which indicates the effect that the distribution of consumers in the network has for the fair distribution of pieces of data. No major difference is produced in the JFI of Piece Messages Sent at different times of the day. Furthermore, Figure 6.62 shows the highest average time achieved in the completion of the data file. As expected, the higher the number of consumers, the higher the average to complete the data file; this is due to the fact that torrents received from the other producers are waiting in the queue until the current one is processed (i.e. data file acquired). In this figure, a difference of 5 seconds is not a significant variation to draw conclusions on the impact of the noise at different times of the day, mainly due to the fact that producers publish torrents at random times within the first 20 seconds of each sequential torrent. Furthermore, Figures 6.63 and 6.64 show the number of network total packets sent and received respectively. The highest number of network total packets sent for each experiment is achieved in the evening (EVENI) in all the experiment scenarios. Conversely, the same effect occurs with the network total packets received, except for the 64_SG_3ID_3PRO_REAL_NIGHT experiment. This could be caused by a variation of the noise conditions since experiments at the same time of the day were performed at consecutive days. Moreover, the experiments on the Irregular Grid (IG) topology produce the highest number of network total packets received, which is due to the higher density of nodes in some areas of the network as compared to the even distribution of nodes in the Square Grid layout.

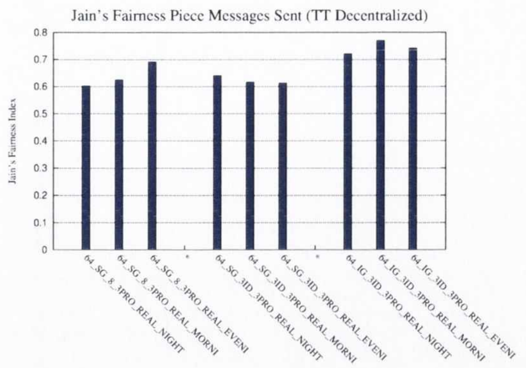


Fig. 6.61: Testbed Evaluation - JFI of Piece Messages Sent - TT Decentral, CFP

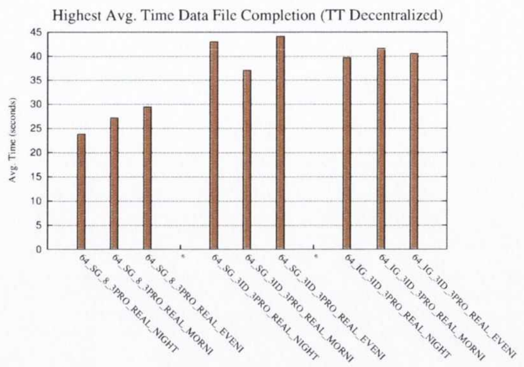


Fig. 6.62: Testbed Evaluation - Highest Avg. Time File Completion - TT Decentral, CFP

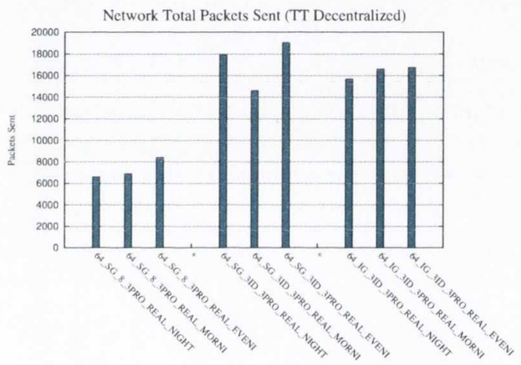


Fig. 6.63: Testbed Evaluation - Network Total Packets Sent - TT Decentral, CFP

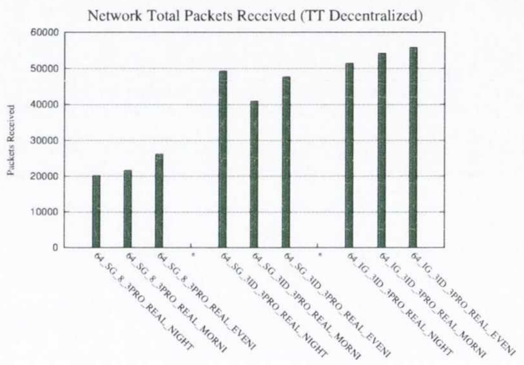


Fig. 6.64: Testbed Evaluation - Network Total Packets Received - TT Decentral, CFP

When comparing these results with those achieved with the simulator under a Casino noise floor, the testbed produces the highest number of network total packets sent with an average increment by a factor of 1.79, while obtaining the lowest number of network total packets received with an average decrement by a factor of 0.90 for all the scenarios. By the same token, the testbed produces the highest average times to complete the data file, increased by a factor of 3.6 (64_8) and 2.3 (64_3ID) when compared to the simulator results. Potential causes of this variation can be due to: i) the irregular signal propagation at the motes in the testbed which produces a higher variation in the path lengths when the routes are being discovered, and ii) the Casino noise floor trace having lower values of noise than the environment where the test was carried out. The efficacy and reliability of the TinyTorrents decentralised protocol operating in a network of 64 telosB devices in a home environment has been shown as it achieved a 100% Torrents Completed ratio for all the experiments.

To better understand the performance of the TinyTorrents protocol in the testbed, three experiments have been analysed in terms of distribution of network traffic and average time to complete the data file. For this purpose the testbed topologies have been mapped and are plotted along with the corresponding metrics.

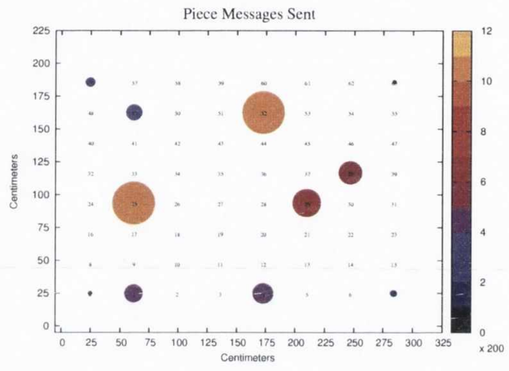


Fig. 6.65: Piece Messages Sent - 64_8 , CFP, TT Decentral, SG, TestBed - JFI 0.69

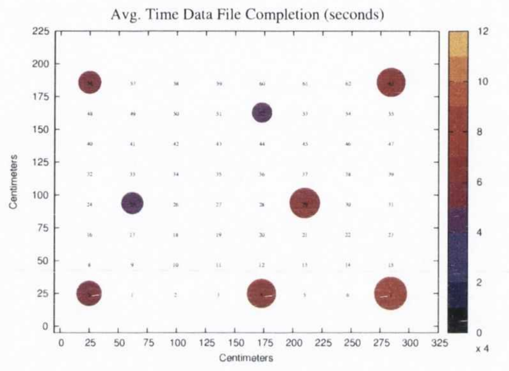


Fig. 6.66: Avg. Time Data File Completion (sec.) - 64_8 , CFP, TT Decentral, SG, TestBed

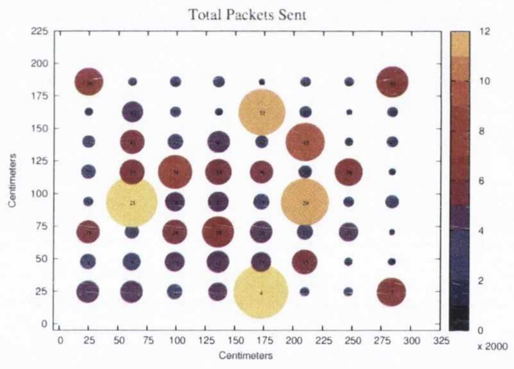


Fig. 6.67: Total Packets Sent - 64_8 , CFP, TT Decentral, SG, TestBed, - JFI 0.74

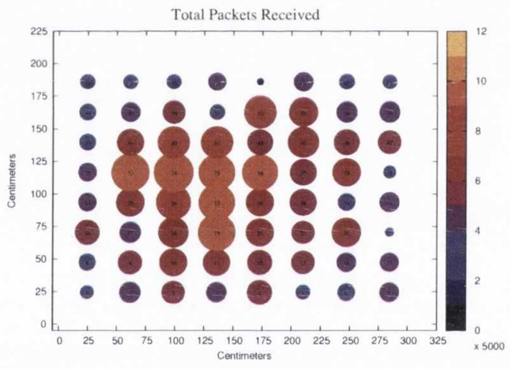


Fig. 6.68: Total Packets Received - 64_8 , CFP, TT Decentral, SG, TestBed, - JFI 0.91

Figures 6.65 - 6.68 show results for the experiment with the 64_8 consumer distribution in the Square Grid topology performed in the evening (EVENI). In terms of Piece Messages Sent (Figure 6.65), a high degree of fairness is achieved (JFI 0.69), with nodes 25 and 52 acting as main distributors of pieces of data from torrents generated by the three producers (nodes 1, 38 and 49). Nodes 25 and 52 are main distributors due to their proximity and key positions with respect to the producers and the other consumers; this effect occurs in the three experiments at different times of the day. These two nodes obtain the lowest average times to complete the data file (16 and 17 seconds) as compared to the rest of the consumers which take between 20 and 30 seconds (see Figure 6.66).

Furthermore, Figures 6.67 and 6.68 show the number of total packets sent and received by each of the motes in the network respectively. As expected, consumer nodes send the highest number of packets followed by nodes placed towards the centre of the network which act as routers between key consumers. It has been noted that consumer nodes at the edges send a high number of total packets when compared to the number of piece messages sent. One example of this case is node 4 which, despite not contributing as many pieces as node 25, sends a similar number of total packets. It has been observed that some of the motes at the edges have a lower packet delivery ratio which require a higher number of route repairs. This may be due to their location with respect to their neighbours such that these nodes are placed at the edge of the communication range of their neighbours. On the other hand, the total number of packets received exhibits a fair distribution with a JFI of 0.91, where centre nodes surrounded by key consumers (i.e. a high number of piece messages sent) receive the highest total number of packets.

Figures 6.69, 6.71 and 6.73 show results for the experiment with the 64_3ID consumer distribution in the Square Grid topology, and Figures 6.70, 6.72 and 6.74 show results for the same consumer distribution in the Irregular Grid topology. Both experiment sets were performed in the evening (EVENI). The Irregular Grid layout is conducive to a fair distribution of pieces of data when compared to the Square Grid layout; this can be seen by visual inspection and is also indicated by the JFI values (0.63 vs. 0.74). This also occurs for the other two experiments performed in the morning and overnight. This is a consequence of the Irregular topology which makes the distribution of consumers more suitable to achieve fairness than the consumer layout achieved in the Square Grid topology. This also confirms the impact the consumer distribution has in the efficiency of the dissemination process. Furthermore, in the Irregular Grid experiment (see Figure 6.70), node 18 is a hot spot in the distribution of pieces of data. This is expected due to its key position, acting as a central gateway node between consumers in the north and south of the network. In terms of total packets received, both topologies depict the same expected pattern where central nodes receive most of the packets; central nodes are surrounded by a higher number of neighbours and are required to route more packets. Specifically in the Irregular Grid layout (see Figure 6.72), the highest number of total packets received correspond to central nodes with a high number of close consumers, but also to those nodes in the gateway path where node 18 is placed. Some nodes at the edges of the Irregular Grid layout exhibit a high number of packets received when compared to nodes at the edges in the Square Grid layout. These nodes are employed as key routers to reach certain areas; for instance node 8. The JFI of the Total Packets Received is high in both experiments - 0.89 (SG) and 0.91 (IG) - which is expected for this type of scenario where a high density of consumers and three producers placed at

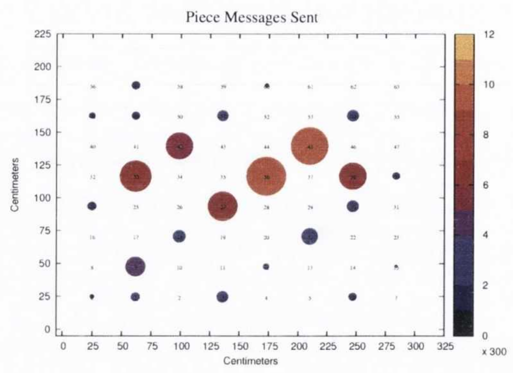


Fig. 6.69: Piece Messages Sent - 64_3ID , CFP, TT Decentral, SG, TestBed - JFI 0.63

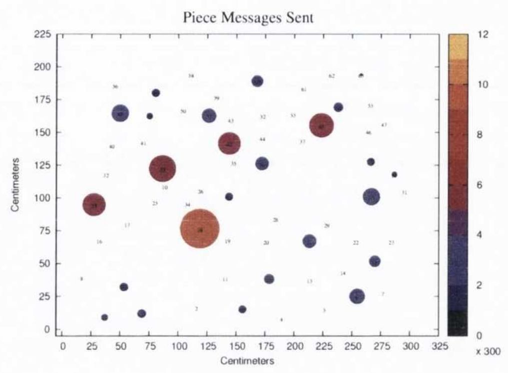


Fig. 6.70: Piece Messages Sent - 64_3ID , CFP, TT Decentral, IG, TestBed - JFI 0.74

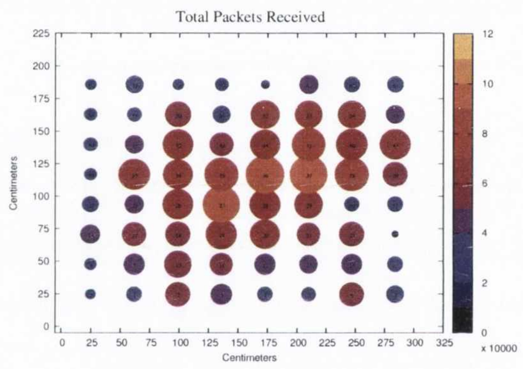


Fig. 6.71: Total Packets Received - 64_3ID , CFP, TT Decentral, SG, TestBed, - JFI 0.89

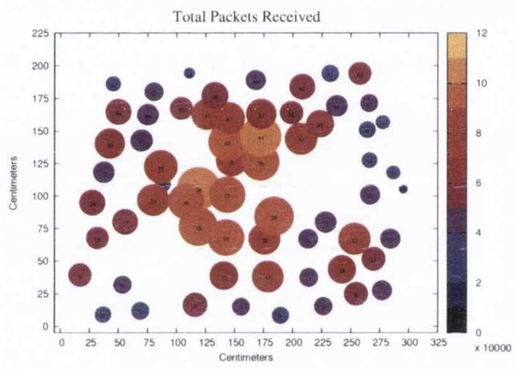


Fig. 6.72: Total Packets Received - 64_3ID , CFP, TT Decentral, IG, TestBed, - JFI 0.91

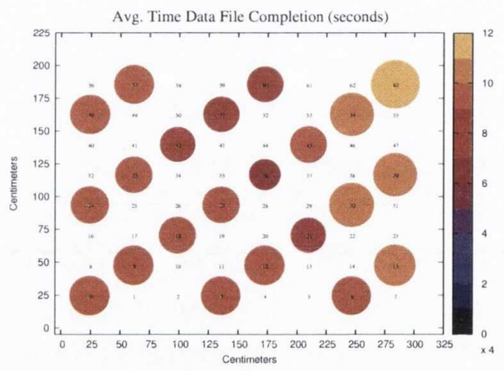


Fig. 6.73: Avg. Time Data File Completion (seconds) - 64_3ID , CFP, TT Decentral, SG, TestBed

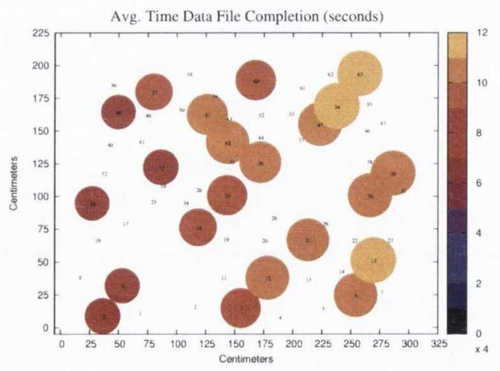


Fig. 6.74: Avg. Time Data File Completion (seconds) - 64_3ID , CFP, TT Decentral, IG, TestBed

distant points generate traffic in most of the areas of the network. The traffic load generated by the high number of consumers has an impact in the average time to acquire a data file (see Figure 6.73 (64_3ID,SG) as compared to Figure 6.66 (64_8,SG). While in the 64_8 experiment the average time to acquire a data file ranges from 18 to 29 seconds, depending on the position of the node, the 64_3ID experiment takes on average between 27 and 44 seconds. The 64_3ID experiment performed in the Irregular Grid topology presents similar average times than the same experiment in the Square Grid layout (see Figures 6.74 and 6.73). However, the lowest average times in the Square Grid topology correspond to central nodes while in the Irregular Grid topology the quickest nodes to acquire the data file are placed at the west side of the network. This is explained by noting that two of the producers are placed in the west side of the network (1 and 49) and the other producer (node 38) is placed at the east side, together with the fact that central nodes are farther from node 38 in terms of routing in the Irregular Grid topology. For all the experiments in the testbed, the average time to complete the data file is higher than that obtained for the same experiments in the simulator. This is a consequence of: i) the different noise level, ii) the topology formation (inter-node distance in the Square Grid topology), iii) the irregular transmission coverage of the motes (antenna position and effect of the usb connector), and iv) TOSSIM being a discrete event simulator which does not consider the real computational time.

In order to provide some insight of the RF conditions in the test environment of the testbed, the RF spectrum activity in the 2.4 GHz band has been characterised using the Wi-Spy spectrum analyser [226]. The measurements have been acquired for the evening experiment with the 64_3ID consumer distribution in the Square Grid topology. Figure 6.75 shows the screen capture of the Chanalyzer Pro (Wi-Spy software), capturing the signal received power from the beginning of the experiment, at 21:00, until the conclusion, at approximately 22:50. The Amplitude (y-axis) represents the RF power level received at the Wi-Spy device in dBm, where the middle line (gray colour) represents the average value. The telosB motes have been configured to operate at channel 26 of the IEEE802.15.4 specification (2480 MHz). This has been the selected channel for the experiments as it does not overlap with Wi-Fi channels and produces the highest packet reception ratios. At channel 26, values of Amplitude close to -90 dBm were obtained. 33 Wi-Fi access points were found interfering at different power levels. Additionally, the microwave was started at around 21:18:30 until 21:24:30 in order to introduce interference in the testbed. The interference effect of the microwave can be clearly seen in the waterfall as the horizontal line with higher density at the mentioned times. Despite this interference environment the ratio of Torrents Completed remained at 100%.

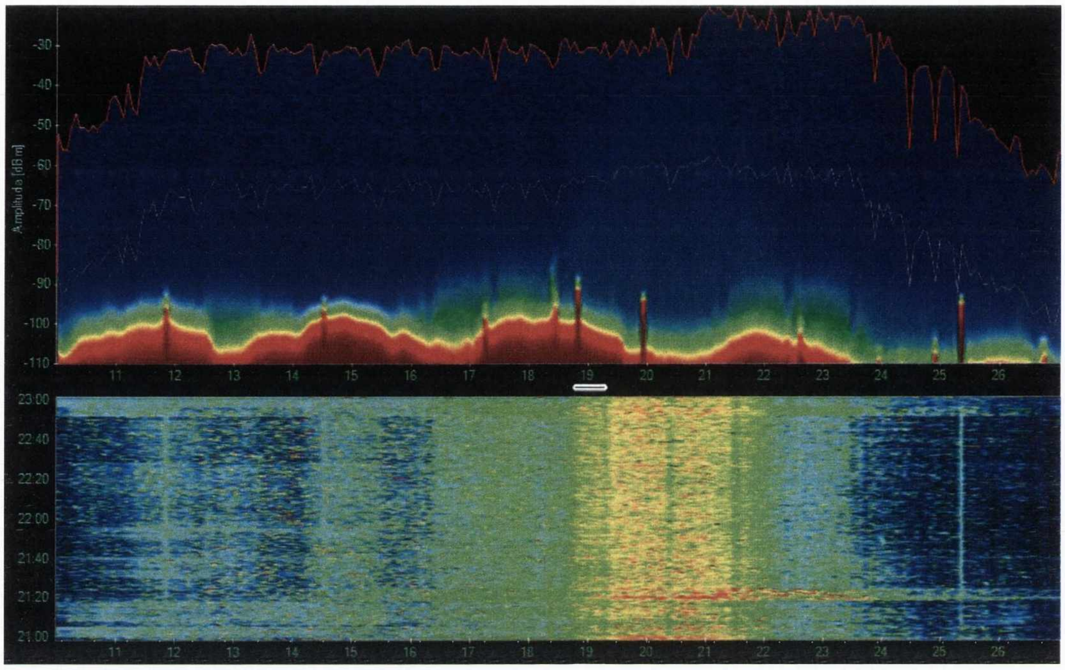


Fig. 6.75: RF Spectrum Activity at the Testbed - Measured with Wi-Spy - Experiment 64_3ID, SG

The testbed performance evaluation has confirmed the efficacy and high reliability of the Tiny-Torrents protocol in the distribution of data over a medium size network. The results obtained exhibited the same general patterns as in the simulated study. A more complete analysis of the differences between the practical and simulation study is proposed for future works.

6.6 Summary

This section has evaluated the performance of the TinyTorrents protocol in its centralised and decentralised versions operating with the Ubiquitous Mobile Gradient (UMG) routing protocol. The system disseminates data files generated by producer nodes to a set of interested, geographically dispersed consumer nodes. At first, a torrent message is disseminated over the network to inform the nodes in the network of the type of data available for distribution. On reception of a torrent, nodes have the capability to become consumers of the data file. A reliable strategy for the dissemination of the torrent file, along with a set of consumer distribution strategies for different network densities of 64, 256 and 400 nodes have been proposed for the evaluation of the performance of the system.

Initially, the efficiency of the set of peer selection strategies proposed to distribute the traffic load amongst consumers in the network have been compared for the centralised and decentralised versions of the TinyTorrents protocol under a variety of scenarios. The Closest First PieceRemaining-based (CFP) peer selection strategy has produced the best combined performance in terms of Total Packets Sent and Received and a high degree of fairness in the distribution of pieces of data. Accordingly, the TinyTorrents decentralised version has been configured with the CFP peer selection strategy for further testing. This version has been evaluated under different scenarios by varying the node density and noise floor in both regular and irregular mesh topologies. In these tests, the system has been shown to be robust to variations in the sensor network environments where nodes might be unavailable for relatively long periods of time. The scalability of the system has also been tested in scenarios of up to 400 nodes using different density consumer distribution strategies, while varying the routing discovery scope in the UMG routing protocol. The average number of packets received and sent grows logarithmically as the network scales and depends on: i) the number of nodes and shape of the topology, ii) the routing scope, and iii) the consumer distribution strategy. Additionally, the system has been tested in large networks of multiple producers in order to study the impact of having simultaneous data distribution processes. The distribution of consumers has been randomly assigned, with a 10% likelihood of a node becoming a consumer. In this scenario, the system has been shown capable of achieving a high degree of fairness in the distibution of pieces of data, despite the random distribution of consumers in the network. Moreover, the low average time to acquire the data file, as compared to scenarios with one producer, has confirmed i) the efficacy of selecting torrents from the queue for which the data file is contained in proximate consumers, and ii) the effectiveness of the unstructured discovery mechanisms in the location of partial trackers which can provide a list of consumers within routing discovery scope. The performance of TinyTorrents has been compared against DIP and DHV, two well-established data consistency maintenance dissemination protocols for wireless sensor networks. This evaluation has shown the efficacy of TinyTorrents in providing selective data dissemination as opposed to epidemic data dissemination where DIP and DHV perform better. Finally, TinyTorrents has proven to reliably disseminate data in a real-world testbed comprised of 64 micaZ motes.

For all the experiments in the scenarios presented in the evaluation, a ratio of 100% Torrents Completed has been achieved with a 100% Torrents Received ratio from each producer. This confirms the high reliability of the system in delivering the data file to all consumers and the suitability of the proposed torrent dissemination strategy, even in scenarios of high noise. The scenarios have been configured with a routing discovery scope of 5 hops and a set of consumer distribution strategies which did not leave any consumer isolated; this should be the case if the system is to maintain a 100% Torrents Completed ratio.

To conclude, the system has proved to be reliable, scalable and capable of achieving a fair and efficient distribution of data files under a variety of network scenarios in the simulator and in a real world testbed. However, the efficiency of the distribution process is also linked to the distribution of consumers and producers in the network. This opens a new research direction which focuses on studying the impact that the distribution of consumers and producers in the network has in the data file distribution process. In this regard, the trade-off between the inter-consumer distance and the routing scope in terms of reliability and fairness of the data file distribution process is of interest.

Chapter 7

Conclusions & Future Work

This chapter summarises the research contributions documented in this thesis. The contributions made with respect to the research problem posed in the wireless sensor networks domain are highlighted and the impact of the outcomes assessed. Finally, potential research avenues are identified in furtherance of the research described in this thesis.

7.1 Conclusions

This thesis addresses the areas of data dissemination and routing in Wireless Sensor Networks. A novel communications architecture providing scalable, selective data dissemination for the development of cooperative applications is presented. The architecture validates a solution to the following research problem: How to reliably disseminate data to a sparse subset of interested nodes in an unstructured, scalable wireless sensor network whilst distributing the traffic load. This problem has been tackled in the literature via the use of epidemic algorithms which push data in a hop-by-hop manner to many of the nodes in the network. While this approach is efficient for network reconfiguration purposes, it is an inefficient solution for scenarios where a subset of nodes in the network are producers and/or consumers which generate and/or consume data and the remaining nodes act as pure relays. The solution proposed in this thesis addresses the problem by employing peer-to-peer content distribution concepts where data is distributed to a subset of nodes in the network in a collaborative way. Existing P2P content distribution strategies, such as the BitTorrent protocol, have been augmented to operate on wireless networks. However, the unreliable multihop nature of wireless sensor networks, and the constrained and ubiquitous characteristics of its devices, demand novel, domain-aware solutions. In addition, the system needs to transport data within the network while operating in a decentralised manner and balancing the traffic load. This thesis has presented such a system - the TinyTorrents communications architecture for wireless sensor devices.

TinyTorrents is a cross-layer communications system composed of a data distribution layer operating above a routing protocol. The rationale behind the approach is based on the data distribution layer, i.e. the TinyTorrents protocol, coordinating data acquisition and distribution amongst the overlay of proximate consumer/producer nodes interested in a data file, whilst employing the routing protocol for data transportation between these nodes. The design employs concepts from the BitTorrent protocol, such as data splitting and the use of torrent files for data file identification and description. The centralised version of the architecture employs a single node to track the swarm of consumers and producers. Scalability and fault tolerance requirements incentivised evolution towards a decentralised version. In the decentralised version, every consumer/producer node is partially responsible for tracking a nearby partial swarm of consumer/producer peers. Consumers need to locate other consumers acting as partial trackers which also provide a list of consumer nodes from which to download parts of the data. For this purpose, a set of unstructured discovery mechanisms which leverage both routing information and the torrent message dissemination process have been created. Consumers select other consumers to download data pieces by following a set of peer selection policies which seek to balance the traffic load and foster data dispersion. Peer selection policies are mainly based on the proximity of the peer and the time when the consumer shows interest in consuming the torrent data, i.e. position in the swarm. Locating close consumers and selecting both the consumer to get data from and the data piece to acquire at each time are key roles of the protocol in the achievement of an efficient and scalable data distribution.

For multihop communication, the UMG routing protocol has been created which provides the point-to-point reliable communication paradigm required by the TinyTorrents protocol. UMG has been designed to provide end-to-end acknowledgements and operate in a reactive manner, avoiding control message overhead when there is no data to route. The routing protocol incorporates service discovery and advertisement functionalities that enable the unstructured discovery required by the decentralised version of the TinyTorrents protocol. UMG employs gradient-based routing which better accommodates the TinyTorrents protocol cross-layer requirements as consumer nodes can spread their gradients to contact other consumers, or to be contacted. The gradient spreading process not only populates nodes with the direction towards a consumer node but also disseminates the "interest" of the node. The interest of a node aggregates a description of the type of data contained or wishing to be acquired, and is a factor employed in the unstructured discovery of potential partial trackers (consumers). Gradients are spread within a limited hop scope of each consumer, which can be dynamically adjusted according to the distribution of the nearest consumers in the network. Each node needs to maintain routing information only for those consumers within a limited scope, which makes the distribution of consumers in the network the defining scalability factor of the architecture. Moreover, UMG implements an opportunistic mechanism for the detection of relative mobility of a node with respect to its neighbourhood. The TinyTorrents architecture is not designed to operate in networks of a high number of transient nodes, and leverages the idea of sensor networks composed of a core set of nodes monitoring and actuating the environment. However, the mobility detection mechanism of UMG enables efficient reactive routing for a small set of mobile nodes, which may act as data mules.

The evaluation section has studied the behaviour of the TinyTorrents protocol operating above the UMG routing protocol in terms of successful data file retrieval by each consumer, traffic load at each consumer, and overall routing cost. The two variants of the architecture, centralised and decentralised, have been evaluated with the set of peer selection policies proposed. However, most of the tests have been carried out with the decentralised version which has proved to be more scalable and achieves a higher degree of fairness in the network load. In tandem with this, peer selection policies considering the location of the peer as the main factor, and the remaining number of pieces as the subsequent factor, have produced the most efficient results in terms of routing cost and traffic load fairness. Additionally, a high impact on the performance of the protocol is derived from the torrent dissemination and acquisition policies which include: i) the number of times each torrent message is broadcast, ii) the average inter-consumer distance in hops, iii) the delay to broadcast according to the node involvement, i.e. producer, consumer or relay, and iv) the delay to start fetching the data. While multiple policies can be configured at the application layer, certain thresholds need to be established to avoid network overload and situations where the partial tracker of a node is too far away.

Overall, the TinyTorrents architecture has proved to be an effective communication substrate for the development of cooperative applications in sensor networks. These type of applications can be characterised by consumer and producer nodes belonging to multiple overlays of interest, akin to social networks today, where data needs to be distributed to the members of each overlay. Complex behaviours could arise from networks of sensor and actuators performing basic tasks on data which is either acquired from sensing or received from consumers/producers in an overlay of interest. In this context, the TinyTorrents framework provides mechanisms for data description and identification which have been incorporated at both layers of the communications architecture. In summary, the

TinyTorrents system provides the communication architecture required to enable the creation of an evolving ad hoc and ubiquitous data storage network which contains, and provides, historical and contemporaneous information from the sensed environment to static and mobile (gateway) nodes in the surroundings.

7.2 Future Work

The TinyTorrents architecture is fully operational and provides decentralised data distribution within the sensor network. Any given node can act, at any time, as a gateway which enables communication with the Internet BitTorrent network through the Vuze TinyTorrent Plugin. This provides global data query and distribution of the sensor network data. The system can also operate in passive mode where data is acquired when torrents are received and published in the BitTorrent network. Further testing and evaluation of the whole TinyTorrents system under complex deployment scenarios are being carried out in order to further stabilize the data distribution process.

As a communications framework for the development of cooperative applications, the TinyTorrents architecture provides interfaces for the application layer to control the torrent dissemination and the data distribution process. In this sense, the application developer is free to explore different data distribution policies, while testing the efficacy and efficiency of the communications framework under specific scenarios. However, a set of policies which cover a great variety of realistic scenarios should be provided such that the application developer can select the most appropriate according to the desired performance of the distribution process. An intermediate layer, which dynamically regulates the parameters of the data dissemination and distribution process and adapts to the current network performance in terms of successful data delivery, consumer distribution in the network, and latency of data acquisition, will be a valuable enhancement to the TinyTorrents framework.

The evaluation section of this thesis did not comprehensively explore the performance of the full set of mechanisms provided for unstructured discovery of partial trackers. The current evaluation has focussed on the rapid acquisition of data on reception of a torrent file, where the network information leveraged in the searching process has not been subject to much variation. The unstructured discovery mechanisms provide the capability to search for "old" torrents as though the network was operating as a distributed data storage system. While this was not a primary aim of this thesis, it is an inherent functionality of the system which deserves to be fully tested. In this regard, the storage of torrent data in the motes flash memory and the efficient and selective mapping of this data to the RAM memory, are key elements for fast distributed random access. The creation of a searchable descriptor-based hierarchical file system for torrent and data storage in the flash memory of sensor nodes is an ongoing project. The search mechanism offers a lightweight efficient search engine based on Bloom filter descriptors of torrent files. The application layer could benefit from descriptor-based searching of stored data in order to perform data fusion activities, or to compute the membership of consumer nodes to an overlay of interest according to the type of data stored in the node.

Related work arises in exploring sleep scheduling policies which suit the cross-layer architecture of the TinyTorrents system, thereby providing energy saving when communication is not occurring, while minimizing radio duty cycle constraints when the data distribution process is taking place. For this purpose, a novel adaptive cross-layer sleep scheduling approach is under development that is purely non-cooperative. A low power listening layer controls the radio duty cycle with a lightweight fuzzy-based inference engine which, instead of relying solely on input from the MAC layer, also receives information from higher layers of the stack. The fuzzy engine needs to be parameterized with the operation of the protocols at each layer, i.e. the type of phases and their sleeping priority policies, as well as the cross-layer relationship of the phases. This is achieved via the use of a set of interfaces which allow creation of a sleeping model from which to derive the correct duty cycle according to the input received from each layer at a given instant. The goal is to maximise the sleeping time while minimizing the risk of packet loss due to unavailability of the radio. This will maintain the performance of the TinyTorrents system at acceptable levels while significantly prolonging the life of unattended ad hoc wireless sensor networks.

Bibliography

[1] W. Weber, J. Rabaey, and E. Aarts, *Ambient Intelligence.* Springer Verlag, 2005.

[2] BitTorrent Inc, "The BitTorrent Protocol." `http://www.bittorrent.com/`, September 2012.

[3] R. Simon Carbajo, M. Huggard, and C. Mc Goldrick, "An End-to-End routing protocol for Peer-to-Peer communication in Wireless Sensor Networks," *MiNEMA Workshop, Proceedings of the Eurosys, Glasgow, UK*, pp. 5–9, April 2008.

[4] M. Weiser, "The computer for the 21st century," *Scientific American*, vol. 265, no. 3, pp. 94–104, 1991.

[5] IEEE, "IEEE Std. 802.15.4-2003: Wireless Medium Access Control (MAC) and Physical Layer(PHY) Specifications for Low-Rate Wireless Personal Area Networks (LR-WPANs)." `http://standards.ieee.org/getieee802/download/802.15.4-2003.pdf`, February 2003.

[6] ZigBee Alliance, "ZigBee Specification." `http://www.zigbee.org/en/spec_download/download_request.asp`, 2007.

[7] Flowers, D. and Yang, Y., "MiWi Wireless Networking Protocol Stack." `http://www.microchip.com`, Microchip Technology Inc. application note AN1066, 2007.

[8] Instrument Society of America, "ISA100.11a." `http://www.isa.org`, September 2012.

[9] Hart Communication Foundation, "WirelessHart Communication." `http://www.hartcomm2.org/hart_protocol/wireless_hart/wireless_hart_main.html`, September 2012.

[10] C. Bormann and G. Mulligan, "IETF IPv6 over Low power Wireless Personal Area Networks (6lowpan) Working Group." `http://tools.ietf.org/wg/6lowpan`, September 2012.

[11] M. Richardson and J. Vasseur, "IETF Routing Over Low power and Lossy networks (ROLL) Working Group." `https://datatracker.ietf.org/wg/roll/charter/`, September 2012.

[12] T. Winter, P. Thubert, and R. Team, "Rpl: Ipv6 routing protocol for low power and lossy networks," *IETF ROLL, IETF Internet-Draft*, 2010.

[13] F. Koushanfar, M. Potkonjak, and A. Sangiovanni-Vincentell, "Fault tolerance techniques for wireless ad hoc sensor networks," *Proceedings of IEEE Sensors*, vol. 2, pp. 1491–1496, 2002.

[14] H. Karl and A. Willig, *Protocols and Architectures for Wireless Sensor Networks.* Wiley, 2005.

[15] Memsic Corporation, "LOTUS Datasheet." `http://www.memsic.com/products/wireless-sensor-networks/wireless-modules.html`, September 2012.

[16] Memsic Corporation, "MicaZ Datasheet." `http://www.memsic.com/products/wireless-sensor-networks/wireless-modules.html`, September 2012. Previously manufactured by Crossbow Technology Inc.

[17] I. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "A survey on sensor networks," *Communications Magazine, IEEE*, vol. 40, no. 8, pp. 102–114, 2002.

[18] N. Bulusu, D. Estrin, L. Girod, and J. Heidemann, "Scalable coordination for wireless sensor networks: self-configuring localization systems," *International Symposium on Communication Theory and Applications (ISCTA 2001), Ambleside, UK, July*, pp. 1797–1800, 2001.

[19] L. Kleinrock and J. Silvester, "Optimum transmission radii for packet radio networks or why six is a magic number," *Proceedings of the IEEE National Telecommunications Conference*, vol. 4, pp. 1–4, 1978.

[20] E. Basha, S. Ravela, and D. Rus, "Model-based monitoring for early warning flood detection," *Proceedings of the 6th ACM conference on Embedded network sensor systems*, pp. 295–308, 2008.

[21] G. Wener-Allen, K. Lorincz, M. Ruiz, O. Marcillo, J. Johnson, J. Lees, and M. Walsh, "Deploying a wireless sensor network on an active volcano, Data-Driven Applications in Sensor Networks (Special Issue)," *IEEE Internet Computing*, pp. 18–25, 2006.

[22] A. Mainwaring, D. Culler, J. Polastre, R. Szewczyk, and J. Anderson, "Wireless sensor networks for habitat monitoring," *Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications*, pp. 88–97, 2002.

[23] P. Zhang, C. Sadler, S. Lyon, and M. Martonosi, "Hardware design experiences in ZebraNet," *Proceedings of the 2nd international conference on Embedded networked sensor systems*, pp. 227–238, 2004.

[24] J. Mazurek, J. Barger, M. Brinn, R. Mullen, D. Price, S. Ritter, and D. Schmitt, "Boomerang mobile counter shooter detection system," *Proceedings of SPIE - Sensors, and C3I Technologies for Homeland Security and Homeland Defense IV*, vol. 5778, pp. 264–282, 2005.

[25] I. Akyildiz, D. Pompili, and T. Melodia, "Underwater acoustic sensor networks: research challenges," *Ad Hoc Networks*, vol. 3, no. 3, pp. 257–279, 2005.

[26] I. Akyildiz and E. Stuntebeck, "Wireless underground sensor networks: Research challenges," *Ad Hoc Networks*, vol. 4, no. 6, pp. 669–686, 2006.

[27] S. Patel, K. Lorincz, R. Hughes, N. Huggins, J. Growdon, D. Standaert, M. Akay, J. Dy, M. Welsh, and P. Bonato, "Monitoring motor fluctuations in patients with Parkinson's disease using wearable sensors," *IEEE Transactions on Information Technology in Biomedicine*, vol. 13, no. 6, pp. 864–873, 2009.

[28] R. Simon Carbajo, A. Staino, K. Ryan, B. Basu, and C. McGoldrick, "Characterisation of Wireless Sensor Platforms for Vibration Monitoring of Wind Turbine Blades," *22nd IET Irish Signals and Systems Conference, ISSC 2011*, pp. 171–176, June 2011.

[29] Texas Instruments, "CC2420 Chipcon 2.4 GHz Transceiver." `http://focus.ti.com/lit/ds/symlink/cc2420.pdf`, September 2012.

[30] Memsic Corporation, "TelosB Datasheet." `http://www.memsic.com/products/wireless-sensor-networks/wireless-modules.html`, September 2012. Previously manufactured by Crossbow Technology Inc.

[31] A. Sinha and A. Chandrakasan, "Dynamic power management in wireless sensor networks," *Design & Test of Computers, IEEE*, vol. 18, no. 2, pp. 62–74, 2001.

[32] J. Al-Karaki and A. Kamal, "Routing techniques in wireless sensor networks: a survey," *Wireless Communications, IEEE [see also IEEE Personal Communications]*, vol. 11, no. 6, pp. 6–28, 2004.

[33] K. Kredo and P. Mohapatra, "Medium access control in wireless sensor networks," *Computer Networks*, vol. 51, no. 4, pp. 961–994, 2007.

[34] S. Singh and C. S. Raghavendra, "PAMAS—power aware multi-access protocol with signalling for ad hoc networks," *SIGCOMM Comput. Commun. Rev.*, vol. 28, no. 3, pp. 5–26, 1998.

[35] I. Chatzigiannakis, A. Kinalis, and S. Nikoletseas, "Wireless sensor networks protocols for efficient collision avoidance in multi-path data propagation," *Proceedings of the 1st ACM international workshop on Performance evaluation of wireless ad hoc, sensor, and ubiquitous networks*, pp. 8–16, 2004.

[36] M. Vuran and I. Akyildiz, "Spatial correlation-based collaborative medium access control in wireless sensor networks," *IEEE/ACM Transactions on Networking*, vol. 14, no. 2, pp. 316–329, 2006.

[37] J. Polastre, J. Hill, and D. Culler, "Versatile low power media access for wireless sensor networks," *Proceedings of the 2nd international conference on Embedded networked sensor systems*, pp. 95–107, 2004.

[38] L. Bao and J. Garcia-Luna-Aceves, "A new approach to channel access scheduling for Ad Hoc networks," *Proceedings of the 7th annual international conference on Mobile computing and networking*, pp. 210–221, 2001.

[39] V. Rajendran, K. Obraczka, and J. Garcia-Luna-Aceves, "Energy-Efficient, Collision-Free Medium Access Control for Wireless Sensor Networks," *Wireless Networks*, vol. 12, no. 1, pp. 63–78, 2006.

[40] W. Ye, J. Heidemann, and D. Estrin, "An energy-efficient MAC protocol for wireless sensor networks," *Proceedings of the Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM'02)*, vol. 3, pp. 1567–1576, 2002.

[41] K. Sohrabi and G. Pottie, "Performance of a novel self-organization protocol for wirelessad-hoc sensor networks," *Vehicular Technology Conference, 1999. VTC 1999-Fall. IEEE VTS 50th*, vol. 2, pp. 1222–1226, 1999.

[42] C. Intanagonwiwat, R. Govindan, and D. Estrin, "Directed diffusion: a scalable and robust communication paradigm for sensor networks," *Proceedings of the 6th annual international conference on Mobile computing and networking*, pp. 56–67, 2000.

[43] W. Heinzelman, J. Kulik, and H. Balakrishnan, "Adaptive protocols for information dissemination in wireless sensor networks," *Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking*, pp. 174–185, 1999.

[44] D. Braginsky and D. Estrin, "Rumor routing algorithm for sensor networks," *Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications*, pp. 22–31, 2002.

[45] C. Schurgers and M. Srivastava, "Energy efficient routing in wireless sensor networks," *Military Communications Conference, MILCOM 2001. Communications for Network-Centric Operations: Creating the Information Force. IEEE*, vol. 1, pp. 357–361, 2001.

[46] C. Perkins and E. Royer, "Ad-hoc on-demand distance vector routing," *Proceedings of the 2nd IEEE Workshop on Mobile Computing Systems and Applications*, vol. 2, pp. 90–100, 1999.

[47] D. Johnson and D. Maltz, "Dynamic Source Routing in Ad Hoc Wireless Networks," *Kluwer International Series in Engineering and Computer Science*, pp. 153–179, 1996.

[48] I. Chakeres and C. Perkins, "Dynamic MANET On-demand (DYMO) Routing," *draft-ietf-nanet-dymo-10, IETF Internet Draft*, July 2007.

[49] C. Perkins and P. Bhagwat, "Highly dynamic Destination-Sequenced Distance-Vector routing (DSDV) for mobile computers," *ACM SIGCOMM Computer Communication Review*, vol. 24, no. 4, pp. 234–244, 1994.

[50] W. Heinzelman, A. Chandrakasan, and H. Balakrishnan, "Energy-efficient communication protocol for wireless microsensor networks," *Proceedings of the 33rd Annual Hawaii International Conference on System Sciences*, pp. 3005–3014, 2002.

[51] S. Lindsey and C. Raghavendra, "PEGASIS: Power-efficient gathering in sensor information systems," *IEEE Aerospace Conference Proceedings*, vol. 3, pp. 1125–1130, 2002.

[52] B. Karp and H. Kung, "GPSR: greedy perimeter stateless routing for wireless networks," *Proceedings of the 6th annual international conference on Mobile computing and networking*, pp. 243–254, 2000.

[53] K. Sohrabi, J. Gao, V. Ailawadhi, and G. Pottie, "Protocols for self-organization of a wireless sensor network," *Personal Communications, IEEE*, vol. 7, no. 5, pp. 16–27, 2000.

[54] T. He, J. Stankovic, C. Lu, and T. Abdelzaher, "SPEED: A stateless protocol for real-time communication in sensor networks," *23rd International Conference on Distributed Computing Systems*, pp. 46–55, 2003.

[55] S. Hedetniemi, S. Hedetniemi, and A. Liestman, "A survey of gossiping and broadcasting in communication networks," *Networks*, vol. 18, no. 4, pp. 319–349, 1988.

[56] H. Lim and C. Kim, "Flooding in wireless ad hoc networks," *Computer Communications*, vol. 24, no. 3-4, pp. 353–363, 2001.

[57] Z. Haas, J. Halpern, and L. Li, "Gossip-based ad hoc routing," *INFOCOM 2002. Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, vol. 3, pp. 1707–1716, 2002.

[58] S. Azzuhri, M. Portmann, and W. Tan, "Evaluation of parameterised route repair in AODV," *4th International Conference on Signal Processing and Communication Systems (ICSPCS)*, pp. 1–7, 2010.

[59] M. Wister, P. Pancardo, F. Acosta, and D. Arias-Torres, "Performance Evaluation of AODV and DYMO as a Plattform for Rescue Task Applications in MANETs," *IEEE Workshops of International Conference on Advanced Information Networking and Applications (WAINA)*, pp. 670–675, March 2011.

[60] S. Bisoyi and S. Sahu, "Performance analysis of Dynamic MANET On-demand (DYMO) Routing protocol," *Special Issue of the International Journal of Computer & Communication Technology Vol.1 Issue 2, 3, 4 for International Conference [ACCTA-2010]*, August 2010.

[61] S. Jayashree and C. S. R. Murthy, "A Taxonomy of Energy Management Protocols for Ad Hoc Wireless Networks," *Communications Magazine, IEEE*, vol. 45, pp. 104–110, April 2007.

[62] D. Sumy, B. Vojcic, and J. Xu, "An Overview of Routing Protocols for Mobile Ad Hoc Networks," *Ultra Wideband Wireless Communication*, pp. 341–427, 2006.

[63] J. Chang and L. Tassiulas, "Maximum lifetime routing in wireless sensor networks," *IEEE/ACM Transactions on Networking*, vol. 12, no. 4, pp. 609–619, 2004.

[64] A. Michail and A. Ephremides, "Energy efficient routing for connection-oriented traffic in ad-hoc wireless networks," *The 11th IEEE International Symposium on Personal, Indoor and Mobile Radio Communications, PIMRC 2000*, vol. 2, pp. 762–766, 2000.

[65] R. Shah and J. Rabaey, "Energy aware routing for low energy ad hoc sensor networks," *Wireless Communications and Networking Conference, WCNC2002*, vol. 1, pp. 350–355, 2002.

[66] H. Chen, H. Mineno, and T. Mizuno, "An Energy-Aware Routing Scheme with Node Relay Willingness in Wireless Sensor Networks," *Proceedings of the First International Conference on Innovative Computing, Information and Control*, pp. 397–400, 2006.

[67] Y. Chen and N. Nasser, "Energy-balancing multipath routing protocol for wireless sensor networks," *Proceedings of the 3rd international conference on Quality of service in heterogeneous wired/wireless networks*, 2006.

[68] V. Raghunathan and C. Srivastava, "Energy-aware wireless microsensor networks," *Signal Processing Magazine, IEEE*, vol. 19, no. 2, pp. 40–50, 2002.

[69] E. S. H. C. Biagioni, "A reliability layer for ad-hoc wireless sensor network routing," *Proceedings of the 37th Annual Hawaii International Conference on System Sciences (HICSS04)*, 2004.

[70] B. Deb, S. Bhatnagar, and B. Nath, "ReInForM: reliable information forwarding using multiple paths in sensor networks," *28th Annual IEEE International Conference on Local Computer Networks, LCN'03.*, pp. 406–415, 2003.

[71] D. Tian and N. Georganas, "Energy efficient routing with guaranteed delivery in wireless sensor networks," *IEEE Wireless Communications and Networking (WCNC 2003)*, vol. 3, pp. 1923–1929, 2003.

[72] H. Hassanein and J. Luo, "Reliable Energy Aware Routing In Wireless Sensor Networks," *Proceedings of the 2nd IEEE Workshop on Dependability and Security in Sensor Networks and Systems*, pp. 54–64, 2006.

[73] E. W. Weisstein, "Method of Steepest Descent (from Mathworld - A Wolfram Web Resource)." `http://mathworld.wolfram.com/MethodofSteepestDescent.html`, September 2012.

[74] R. Poor, "Gradient routing in ad hoc networks," *Media Laboratory, Massachusetts Institute of Technology Cambridge, MA*, vol. 2139, 2000.

[75] D. Estrin, R. Govindan, J. Heidemann, and S. Kumar, "Next century challenges: Scalable coordination in sensor networks," *Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking*, pp. 263–270, 1999.

[76] F. Ye, S. Lu, and L. Zhang, "Gradient broadcast: A robust, long-lived large sensor network," *UCLA Internet Research Lab Papers*, 2001.

[77] F. Ye, G. Zhong, S. Lu, and L. Zhang, "Gradient broadcast: A robust data delivery protocol for large scale sensor networks," *Wireless Networks*, vol. 11, no. 3, pp. 285–298, 2005.

[78] H. Yoo, M. Shim, D. Kim, and K. H. Kim, "GLOBAL: A Gradient-based routing protocol for load-balancing in large-scale wireless sensor networks with multiple sinks," *IEEE Symposium on Computers and Communications*, pp. 556–562, 2010.

[79] P. Huang, H. Chen, G. Xing, and Y. Tan, "SGF: a state-free gradient-based forwarding protocol for wireless sensor networks," *ACM Transactions on Sensor Networks (TOSN)*, vol. 5, no. 2, pp. 1–25, 2009.

[80] A. Basu, A. Lin, and S. Ramanathan, "Routing using potentials: a dynamic traffic-aware routing algorithm," *Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*, pp. 37–48, 2003.

[81] F. Ye, A. Chen, S. Lu, and L. Zhang, "A scalable solution to minimum cost forwarding in large sensor networks," *Tenth International Conference on Computer Communications and Networks.*, pp. 304–309, 2001.

[82] K. Jaffres-Runser and C. Comaniciu, "A probabilistic interference and energy aware gradient broadcasting algorithm for wireless sensor networks," *3rd International Symposium on Wireless Pervasive Computing, ISWPC 2008*, pp. 1–5, 2008.

[83] K. Jaffres-Runser, C. Comaniciu, J. Gorce, and R. Zhang, "U-GRAB: a utility-based gradient broadcasting algorithm for wireless sensor networks," *IEEE Military Communications Conference, MILCOM 2009*, pp. 1–7, 2009.

[84] K. Jaffrès-Runser, C. Comaniciu, and J. Gorce, "Interference and Congestion Aware Gradient Broadcasting Routing for Wireless Sensor Networks," *CoRR 2009 - Arxiv Preprint arXiv:0902.0746*, 2009.

[85] Q. Fang, J. Gao, L. Guibas, V. de Silva, and L. Zhang, "GLIDER: Gradient landmark-based distributed routing for sensor networks," *INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies*, vol. 1, pp. 339–350, 2005.

[86] L. Xia, X. Chen, and X. Guan, "A new gradient-based routing protocol in wireless sensor networks," *Embedded Software and Systems*, pp. 318–325, 2005.

[87] J. Lim and K. Shin, "Gradient-ascending routing via footprints in wireless sensor networks," *Proceedings IEEE Real-Time Systems Symposium (RTSS '05)*, pp. 298–307, 2005.

[88] B. Bloom, "Space/time trade-offs in hash coding with allowable errors," *Communications of the ACM*, vol. 13, no. 7, pp. 422–426, 1970.

[89] L. Fan, P. Cao, J. Almeida, and A. Broder, "Summary cache: a scalable wide-area web cache sharing protocol," *IEEE/ACM Transactions on Networking (TON)*, vol. 8, no. 3, pp. 281–293, 2000.

[90] "The network simulator (ns-2)." `http://www.isi.edu/nsnam/ns/`, September 2012.

[91] T. Schoellhammer, B. Greenstein, and D. Estrin, "Hyper: A routing protocol to support mobile users of sensor networks," *Technical Report, Center for Embedded Network Sensing (CENS), Univ. of California*, 2006.

[92] A. Woo, T. Tong, and D. Culler, "Taming the underlying challenges of reliable multihop routing in sensor networks," *Proceedings of the 1st international conference on Embedded networked sensor systems*, pp. 14–27, 2003.

[93] J. Wu and P. Havinga, "Reliable Cost-based Data-centric Routing Protocol for Wireless Sensor Networks," *Seventh ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing (SNPD 2006).*, pp. 267–272, 2006.

[94] N. Khan, Z. Khalid, and G. Ahmed, "GRAdient cost establishment (GRACE) for an energy-aware routing in wireless sensor networks," *EURASIP Journal on Wireless Communications and Networking*, vol. 2009, pp. 1–15, 2009.

[95] R. Fonseca, O. Gnawali, K. Jamieson, S. Kim, P. Levis, and A. Woo, "TEP 123: The Collection Tree Protocol." `http://www.tinyos.net/tinyos-2.x/doc/html/tep123.html`, 2006.

[96] TinyOS Alliance, "TinyOS version 2.x (T2)." `http://www.tinyos.net`, September 2012.

[97] Oracle Labs, "SunSPOT World." `http://www.sunspotworld.com`, April 2012. Previously manufactured by Sun Microsystems.

[98] O. Gnawali, "TEP 124: The Link Estimation Exchange Protocol (LEEP)." `http://www.tinyos.net/tinyos-2.x/doc/txt/tep124.txt`, 2006.

[99] P. Levis, N. Patel, D. Culler, and S. Shenker, "Trickle: A self-regulating algorithm for code propagation and maintenance in wireless sensor networks," *Proceedings of the 1st conference on Symposium on Networked Systems Design and Implementation*, pp. 2–2, 2004.

[100] O. Gnawali, R. Fonseca, K. Jamieson, D. Moss, and P. Levis, "Collection Tree Protocol (CTP)," *Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems*, pp. 1–14, 2009.

[101] R. Fonseca, O. Gnawali, K. Jamieson, and P. Levis, "Four-bit wireless link estimation," *Proceedings of the Sixth Workshop on Hot Topics in Networks (HotNets VI)*, 2007.

[102] S. Moeller, A. Sridharan, B. Krishnamachari, and O. Gnawali, "Routing without routes: The backpressure collection protocol," *Proceedings of the 9th ACM/IEEE International Conference on Information Processing in Sensor Networks*, pp. 279–290, 2010.

[103] J. Lee, B. Kusy, T. Azim, B. Shihada, and P. Levis, "Whirlpool routing for mobility," *Proceedings of the eleventh ACM international symposium on Mobile ad hoc networking and computing*, pp. 131–140, 2010.

[104] R. Thouvenin, "Implementing and Evaluating the Dynamic Manet On-demand Protocol in Wireless Sensor Networks," Master's thesis, University of Aarhus (Denmark), June 2007.

[105] P. Levis, "Tossim system description," tech. rep., University of California, Berkeley, 2002.

[106] P. Levis, N. Lee, M. Welsh, and D. Culler, "TOSSIM: accurate and scalable simulation of entire tinyOS applications," *Proceedings of the 1st international conference on Embedded networked sensor systems*, pp. 126–137, 2003.

[107] H. Lee, A. Cerpa, and P. Levis, "Improving wireless simulation through noise modeling," *Proceedings of the 6th international conference on Information Processing in Sensor Networks (IPSN '07)*, pp. 21–30, 2007.

[108] J. Faruque and A. Helmy, "TABS: Link Loss Tolerant Data Routing Protocol for Multi-hop Wireless Sensor Networks," *2010 IEEE International Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing*, pp. 11–18, 2010.

[109] Y. Guo, Z. Xu, C. Chen, and X. Guan, "DGR: dynamic gradient-based routing protocol for unbalanced and persistent data transmission in wireless sensor and actor networks," *Journal of Zhejiang University-Science C*, vol. 12, no. 4, pp. 273–279, 2011.

[110] M. Dohler, T. Watteyne, and D. Barthel, "Urban wsns routing requirements in low power and lossy networks," *IETF ROLL, IETF Internet-Draft*, March 2009.

[111] K. Pister, P. Thubert, S. Dwars, and T. Phinney, "Industrial Routing Requirements in Low-Power and Lossy Networks," *IETF ROLL, IETF Internet-Draft*, June 2009.

[112] A. Brandt, J. Buron, and G. Porcu, "Home Automation Routing Requirements in Low Power and Lossy Networks," *IETF ROLL, IETF Internet-Draft*, January 2010.

[113] J. Martocci, P. DeMil, W. Vermeylen, and N. Riou, "Building Automation Routing Requirements in Low Power and Lossy Networks," *IETF ROLL, IETF Internet-Draft*, January 2010.

[114] D. Wang, Z. Tao, J. Zhang, and A. Abouzeid, "RPL Based Routing for Advanced Metering Infrastructure in Smart Grid," *2010 IEEE International Conference on Communications Workshops (ICC)*, pp. 1–6, 2010.

[115] M. Goyal and E. Baccelli, "Reactive Discovery of Point-to-Point Routes in Low Power and Lossy Networks," *IETF ROLL, IETF Internet-Draft*, 2010.

[116] G. Chen, J. Branch, and B. Szymanski, "Self-selective routing for wireless ad hoc networks," *IEEE International Conference on Wireless And Mobile Computing, Networking And Communications (WiMob'2005)*, vol. 3, pp. 57–64, 2005.

[117] R. Cheng, S. Peng, and C. Huang, "A Gradient-Based Dynamic Load Balance Data Forwarding Method for Multi-sink Wireless Sensor Networks," *IEEE Asia-Pacific Services Computing Conference, APSCC'08*, pp. 1132–1137, 2008.

[118] O. Erdene-Ochir, M. Minier, F. Valois, and A. Kountouris, "Toward Resilient Routing in Wireless Sensor Networks: Gradient-Based Routing in Focus," *Fourth International Conference on Sensor Technologies and Applications*, pp. 478–483, 2010.

[119] K. Han, Y. Ko, and J. Kim, "A novel gradient approach for efficient data dissemination in wireless sensor networks," *Vehicular Technology Conference, 2004. VTC2004-Fall. 2004 IEEE 60th*, vol. 4, pp. 2979–2983, 2004.

[120] F. Verbist, N. Festjens, K. Steenhaut, and A. Nowe, "Hop count discovery protocol for gradient based routing in wireless sensor networks," *First International Conference on Communications and Electronics, ICCE'06.*, pp. 102–105, 2006.

[121] S. Khan, N. Khan, and M. Ahmed, "A robust and energy efficient global gradient setup mechanism for gradient based routing in Wireless Sensor Networks," *14th Asia-Pacific Conference on Communications, APCC 2008.*, pp. 1–5, 2008.

[122] J. Faruque and A. Helmy, "Rugged: Routing on fingerprint gradients in sensor networks," *IEEE/ACS International Conference on Pervasive Services, ICPS 2004*, pp. 179–188, 2004.

[123] R. Sarkar, X. Zhu, J. Gao, L. Guibas, and J. Mitchell, "Iso-contour queries and gradient descent with guaranteed delivery in sensor networks," *INFOCOM 2008. The 27th Conference on Computer Communications.*, pp. 960–967, 2008.

[124] H. Lin, M. Lu, N. Milosavljevic, J. Gao, and L. Guibas, "Composable information gradients in wireless sensor networks," *Proceedings of the 7th international conference on Information processing in sensor networks*, pp. 121–132, 2008.

[125] W. Wei and Y. Qi, "Information Potential Fields Navigation in Wireless Ad-Hoc Sensor Networks," *Sensors*, vol. 11, no. 5, pp. 4794–4807, 2011.

[126] J. Faruque and A. Helmy, "PBS: A virtual grid architecture for gradient-based active querying in sensor networks," *International Symposium on Collaborative Technologies and Systems, CTS 2008.*, pp. 9–18, 2008.

[127] S. Androutsellis-Theotokis and D. Spinellis, "A Survey of Peer-to-Peer Content Distribution Technologies," *ACM Computing Surveys*, vol. 36, no. 4, pp. 335–371, 2004.

[128] A. Crespo and H. Garcia-Molina, "Routing Indices for Peer-to-Peer Systems," *International Conference on Distributed Computing Systems*, vol. 22, pp. 23–34, 2002.

[129] B. Yang and H. Garcia-Molina, "Improving Search in Peer-to-Peer Networks," *International Conference on Distributed Computing Systems*, vol. 22, pp. 5–14, 2002.

[130] Q. Lv, P. Cao, E. Cohen, K. Li, and S. Shenker, "Search and replication in unstructured peer-to-peer networks," *Proceedings of the 16th international conference on Supercomputing*, pp. 84–95, 2002.

[131] V. Kalogeraki, D. Gunopulos, and D. Zeinalipour-Yazti, "A local search mechanism for peer-to-peer networks," *Proceedings of the eleventh international conference on Information and knowledge management*, pp. 300–307, 2002.

[132] Y. Chawathe, S. Ratnasamy, L. Breslau, N. Lanham, and S. Shenker, "Making gnutella-like P2P systems scalable," *Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*, pp. 407–418, 2003.

[133] M. Ripeanu, "Peer-to-Peer Architecture Case Study: Gnutella Network," *Proceedings of International Conference on Peer-to-peer Computing*, vol. 101, 2001.

[134] A. Singla and C. Rohrs, "Ultrapeers: Another step towards gnutella scalability." `http://rfc-gnutella.sourceforge.net/Proposals/Ultrapeer/Ultrapeers.htm`, 2002.

[135] J. Douceur, "The Sybil Attack," *Peer-to-peer Systems*, pp. 251–260, 2002.

[136] I. Stoica, R. Morris, D. Karger, M. Kaashoek, and H. Balakrishnan, "Chord: A scalable peer-to-peer lookup service for internet applications," *Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications*, pp. 149–160, 2001.

[137] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Schenker, "A scalable content-addressable network," *Proceedings of the 2001 SIGCOMM conference*, vol. 31, no. 4, pp. 161–172, 2001.

[138] A. Rowstron and P. Druschel, "Pastry: Scalable, Decentralized Object Location, and Routing for Large-Scale Peer-to-Peer Systems," *Lecture Notes In Computer Science*, vol. 2218, pp. 329–350, 2001.

[139] B. Zhao, J. Kubiatowicz, and A. Joseph, "Tapestry: An Infrastructure for Fault-tolerant Wide-area Location and Routing," *Computer*, vol. 74, 2001.

[140] P. Maymounkov and D. Mazieres, "Kademlia: A peer-to-peer information system based on the XOR metric," *Proceedings of the 1st International Workshop on Peer-to-Peer Systems (IPTPS02)*, vol. 258, p. 263, 2002.

[141] I. Clarke, O. Sandberg, B. Wiley, and T. Hong, "Freenet: A Distributed Anonymous Information Storage and Retrieval System," *Lecture Notes in Computer Science*, pp. 46–66, 2001.

[142] M. Bakhouya and J. Gaber, "Ubiquitous and pervasive application design," *Encyclopedia of Mobile Computing and Commerce, Edited by: D. Taniar. Idea Group Pub*, 2007.

[143] S. Madden, M. Franklin, J. Hellerstein, and W. Hong, "TinyDB: an acquisitional query processing system for sensor networks," *ACM Transactions on Database Systems (TODS)*, vol. 30, no. 1, pp. 122–173, 2005.

[144] C. Jardak, E. Osipov, and P. Mahonen, "Distributed Information Storage and Collection for WSNs," *IEEE Internatonal Conference on Mobile Adhoc and Sensor Systems, MASS 2007.*, pp. 1–10, 2007.

[145] J. Girao, D. Westhoff, E. Mykletun, and T. Araki, "Tinypeds: Tiny persistent encrypted data storage in asynchronous wireless sensor networks," *Ad Hoc Networks*, vol. 5, no. 7, pp. 1073–1089, 2007.

[146] B. Cohen, "Incentives Build Rebustness in BitTorrent," *Workshop on Economics of Peer-to-Peer systems*, May 2003.

[147] L. G. et al., "Measurements, Analysis and Modeling of BitTorrent-like Systems," *Internet Measurement Conference*, 2005.

[148] R. Thommes and M. Coates, "BitTorrent fairness: analysis and improvements," *Proc. Workshop Internet, Telecom and Signal Proc, Noosa, Australia*, 2004.

[149] Vuze Inc, "Vuze BitTorrent Client (former Azureus)." `http://azureus.sourceforge.net/`, September 2012.

[150] Vuze Inc, "Peer Exchange Plug-In at Vuze BitTorrent Client." `http://wiki.vuze.com/w/Peer_Exchange`, September 2012.

[151] Berlios Developer, "Anatomic P2P." `http://developer.berlios.de/projects/anatomic`, September 2012.

[152] J. Pouwelse, P. Garbacki, J. Wang, A. Bakker, J. Yang, A. Iosup, D. Epema, M. Reinders, M. Van Steen, and H. Sips, "TRIBLER: a social-based peer-to-peer system," *Concurrency and Computation: Practice and Experience*, vol. 20, no. 2, pp. 127–138, 2008.

[153] A. Nandan, S. Das, G. Pau, M. Gerla, and M. Y. Sanadidi, "Co-operative Downloading in Vehicular Ad-Hoc Wireless Networks," *International Conference on Wireless on Demand Network Systems and Service*, vol. 0, pp. 32–41, 2005.

[154] S. Rajagpalan *et al.*, "A cross-layer decentralized BitTorrent for mobile ad hoc networks," *2006 3rd Annual International Conference on Mobile and Ubiquitous Systems-Workshops*, pp. 1–10, 2006.

[155] N. Gaddam and A. Potluri, "Study of BitTorrent for file sharing in Ad Hoc networks," *Fifth IEEE Conference on Wireless Communication and Sensor Networks (WCSN)*, pp. 1–6, 2009.

[156] P. Michiardi and G. Urvoy-Keller, "Performance analysis of cooperative content distribution in wireless ad hoc networks," *Fourth Annual Conference on Wireless on Demand Network Systems and Services, WONS'07.*, pp. 22–29, 2007.

[157] S. ElRakabawy and C. Lindemann, "Peer-to-peer file transfer in wireless mesh networks," *Fourth Annual Conference on Wireless on Demand Network Systems and Services, WONS'07.*, pp. 114–121, 2007.

[158] M. Sbai, C. Barakat, J. Choi, A. Hamra, and T. Turletti, "Adapting BitTorrent to wireless ad hoc networks," *Ad-hoc, Mobile and Wireless Networks*, pp. 189–203, 2008.

[159] M. Sbai, C. Barakat, J. Choi, A. Al Hamra, and T. Turletti, "BitHoc: Bittorrent for wireless ad hoc networks," tech. rep., inria-00196313, France, 2008.

[160] P. Jacquet, P. Muhlethaler, T. Clausen, A. Laouiti, A. Qayyum, and L. Viennot, "Optimized link state routing protocol for ad hoc networks," *IEEE INMIC*, vol. 1, pp. 28–30, 2001.

[161] A. Krifa, M. Sbai, C. Barakat, and T. Turletti, "BitHoc: A content sharing application for Wireless Ad hoc Networks," *IEEE International Conference on Pervasive Computing and Communications, PerCom 2009.*, pp. 1–3, 2009.

[162] E. Salhi, M. Sbai, and C. Barakat, "Neighborhood selection in mobile P2P networks," *Algotel conference, Carry-Le-Rouet, France*, 2009.

[163] M. Sbai, E. Salhi, and C. Barakat, "P2P content sharing in spontaneous multi-hop wireless networks," *Second International Conference on Communication Systems and Networks (COMSNETS 2010)*, pp. 1–10, 2010.

[164] S. Shenker, S. Ratnasamy, B. Karp, R. Govindan, and D. Estrin, "Data-centric storage in sensornets," *ACM SIGCOMM Computer Communication Review*, vol. 33, no. 1, pp. 137–142, 2003.

[165] S. Ratnasamy, B. Karp, S. Shenker, D. Estrin, R. Govindan, L. Yin, and F. Yu, "Data-centric storage in sensornets with GHT, a geographic hash table," *Mobile Networks and Applications*, vol. 8, no. 4, pp. 427–442, 2003.

[166] S. Ratnasamy, B. Karp, L. Yin, F. Yu, D. Estrin, R. Govindan, and S. Shenker, "GHT: a geographic hash table for data-centric storage," *Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications*, pp. 78–87, 2002.

[167] M. Ali and Z. Uzmi, "CSN: A Network Protocol for Serving Dynamic Queries in Large-Scale Wireless Sensor Networks," *2nd Annual Conference on Communication Networks and Services Research (CNSR'04)*, pp. 165–174, 2004.

[168] M. Ali and K. Langendoen, "A Case for Peer-to-Peer Network Overlays in Sensor Networks," *Proceedings of WWSNA / 6th IPSN'07, Cambridge, MA*, vol. 6, 2007.

[169] H. Pucha, S. Das, and Y. Hu, "Ekta: An Efficient DHT Substrate for Distributed Applications in Mobile Ad Hoc Networks," *Proc. 6th IEEE Workshop on Mobile Computing Systems and Applications (WMCSA)*, pp. 163–173, 2004.

[170] T. Zahn and J. Schiller, "MADPastry: A DHT Substrate for Practicably Sized MANETs," *Proc. of ASWN*, 2005.

[171] T. Zahn, G. Wittenburg, and J. Schiller, "Towards efficient range queries in mobile ad hoc networks using DHTs," *Proceedings of the 1st international workshop on Decentralized resource sharing in mobile computing and networking*, pp. 72–74, 2006.

[172] C. Zheng, G. Shen, S. Li, and S. Shenker, "Distributed Segment Tree: Support of Range Query and Cover Query over DHT," *Proc. IPTPS*, 2006.

[173] T. Zahn and J. Schiller, "DHT-based Unicast for Mobile Ad Hoc Networks," *Proceedings of the 4th annual IEEE international conference on Pervasive Computing and Communications Workshops*, 2006.

[174] A. Ghose, J. Grossklags, and J. Chuang, "Resilient Data-Centric Storage in Wireless Ad-Hoc Sensor Networks," *Lecture Notes in Computer Science*, pp. 45–62, 2003.

[175] O. Landsiedel, K. Lehmann, and K. Wehrle, "T-DHT: Topology-Based Distributed Hash Tables," *Proceedings of the Fifth IEEE International Conference on Peer-to-Peer Computing (P2P'05)*, pp. 143–144, 1931.

[176] J. Newsome and D. Song, "GEM: Graph EMbedding for routing and data-centric storage in sensor networks without geographic information," *Proceedings of the 1st international conference on Embedded networked sensor systems*, pp. 76–88, 2003.

[177] M. Caesar, M. Castro, E. Nightingale, G. O'Shea, and A. Rowstron, "Virtual ring routing: network routing inspired by DHTs," *Proceedings of the 2006 conference on Applications, technologies, architectures, and protocols for computer communications*, vol. 36, no. 4, pp. 351–362, 2006.

[178] R. Fonseca, S. Ratnasamy, J. Zhao, C. Ee, D. Culler, S. Shenker, and I. Stoica, "Beacon vector routing: Scalable point-to-point routing in wireless sensornets," *Proceedings of NSDI 2005*, pp. 329–342, 2005.

[179] A. Awad, R. German, and F. Dressler, "P2P-based routing and data management using the virtual cord protocol (VCP)," *Proceedings of the 9th ACM international symposium on Mobile ad hoc networking and computing*, pp. 443–444, 2008.

[180] A. Al-Mamou and H. Labiod, "ScatterPastry: An Overlay Routing Using a DHT over Wireless Sensor Networks," *The 2007 International Conference on Intelligent Pervasive Computing, IPC.*, pp. 274–279, 2007.

[181] X. Liu, Q. Huang, and Y. Zhang, "Combs, needles, haystacks: balancing push and pull for discovery in large-scale sensor networks," *Proceedings of the 2nd international conference on Embedded networked sensor systems*, pp. 122–133, 2004.

[182] J. Ahn, S. Kapadia, S. Pattem, A. Sridharan, M. Zuniga, J. Jun, C. Avin, and B. Krishnamachari, "Empirical evaluation of querying mechanisms for unstructured wireless sensor networks," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 3, pp. 17–26, 2008.

[183] W. Acosta and S. Chandra, "On the need for query-centric unstructured peer-to-peer overlays," *IEEE International Symposium on Parallel and Distributed Processing, IPDPS 2008.*, pp. 1–8, 2008.

[184] H. Barjini, M. Othman, H. Ibrahim, and N. Udzir, "Shortcoming, problems and analytical comparison for flooding-based search techniques in unstructured P2P networks," *Peer-to-Peer Networking and Applications*, pp. 1–13, 2011.

[185] I. Pu and Y. Shen, "Enhanced blocking expanding ring search in mobile ad hoc networks," *3rd International Conference on New Technologies, Mobility and Security (NTMS)*, pp. 1–5, 2009.

[186] C. Gkantsidis, M. Mihail, and A. Saberi, "Random walks in peer-to-peer networks: algorithms and evaluation," *Performance Evaluation*, vol. 63, no. 3, pp. 241–263, 2006.

[187] J. Gaber and M. Bakhouya, "Mobile Agent-Based Approach for Resource Discovery in Peer-to-Peer Networks," *Agents and Peer-to-Peer Computing, Lecture Notes in Computer Science*, vol. 4461, pp. 63–73, 2008.

[188] S. Shakkottai, "Asymptotics of query strategies over a sensor network," *INFOCOM 2004. Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies*, vol. 1, 2004.

[189] A. Dimakis, V. Prabhakaran, and K. Ramchandran, "Ubiquitous access to distributed data in large-scale sensor networks through decentralized erasure codes," *Fourth International Symposium on Information Processing in Sensor Networks, IPSN 2005.*, pp. 111–117, 2005.

[190] A. Dimakis, V. Prabhakaran, and K. Ramchandran, "Decentralized erasure codes for distributed networked storage," *IEEE/ACM Transactions on Networking (TON)*, vol. 14, pp. 2809–2816, 2006.

[191] K. K. Rachuri and C. S. R. Murthy, "Energy Efficient and Scalable Search in Dense Wireless Sensor Networks," *IEEE Trans. Comput.*, vol. 58, pp. 812–826, June 2009.

[192] N. Sadagopan, B. Krishnamachari, and A. Helmy, "Active query forwarding in sensor networks," *Ad Hoc Networks*, vol. 3, no. 1, pp. 91–113, 2005.

[193] K. Rachuri and C. Siva Ram Murthy, "Energy efficient and low latency biased walk techniques for search in wireless sensor networks," *Journal of Parallel and Distributed Computing*, 2010.

[194] D. Guo, Y. He, and P. Yang, "Receiver-oriented design of Bloom filters for data-centric routing," *Computer Networks*, vol. 54, no. 1, pp. 165–174, 2010.

[195] P. Hebden and A. Pearce, "Data-centric routing using bloom filters in wireless sensor networks," *Fourth International Conference on Intelligent Sensing and Information Processing, ICISIP 2006.*, pp. 72–77, 2006.

[196] X. Li, J. Wu, and J. Xu, "Hint-based routing in WSNs using scope decay bloom filters," *International Workshop on Networking, Architecture, and Storages (IWNAS'06)*, pp. 111–118, 2006.

[197] S. Rhea and J. Kubiatowicz, "Probabilistic location and routing," *INFOCOM 2002. Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, vol. 3, pp. 1248–1257, 2002.

[198] A. Kumar, J. Xu, and E. Zegura, "Efficient and scalable query routing for unstructured peer-to-peer networks," *INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE*, vol. 2, pp. 1162–1173, 2005.

[199] J. Kulik, W. Heinzelman, and H. Balakrishnan, "Negotiation-based protocols for disseminating information in wireless sensor networks," *Wireless Networks*, vol. 8, no. 2/3, pp. 169–185, 2002.

[200] C. Intanagonwiwat, R. Govindan, D. Estrin, J. Heidemann, and F. Silva, "Directed diffusion for wireless sensor networking," *IEEE/ACM Transactions on Networking*, vol. 11, no. 1, pp. 2–16, 2003.

[201] F. Ye, H. Luo, J. Cheng, S. Lu, and L. Zhang, "A two-tier data dissemination model for large-scale wireless sensor networks," *Proceedings of the 8th annual international conference on Mobile computing and networking*, pp. 148–159, 2002.

[202] H. Luo, F. Ye, J. Cheng, S. Lu, and L. Zhang, "TTDD: two-tier data dissemination in large-scale wireless sensor networks," *Wireless Networks*, vol. 11, no. 1, pp. 161–175, 2005.

[203] D. Coffin, D. Van Hook, S. McGarry, and S. Kolek, "Declarative ad-hoc sensor networking," *Proceedings of SPIE*, vol. 4126, p. 109, 2000.

[204] C. Wan, A. Campbell, and L. Krishnamurthy, "Pump-slowly, fetch-quickly (PSFQ): a reliable transport protocol for sensor networks," *IEEE Journal on Selected Areas in Communications*, vol. 23, no. 4, pp. 862–872, 2005.

[205] S. Park, R. Vedantham, R. Sivakumar, and I. Akyildiz, "A scalable approach for reliable downstream data delivery in wireless sensor networks," *Proceedings of the 5th ACM international symposium on Mobile ad hoc networking and computing*, pp. 78–89, 2004.

[206] F. Stann and J. Heidemann, "RMST: Reliable data transport in sensor networks," *IEEE International Workshop on Sensor Network Protocols and Applications*, pp. 102–112, 2003.

[207] T. Stathopoulos, J. Heidemann, and D. Estrin, "A Remote Code Update Mechanism for Wireless Sensor Networks," tech. rep., CENS-TR-30, University of California, Los Angeles, Center for Embedded Networked Computing, November 2003.

[208] S. Kulkarni and L. Wang, "MNP: Multihop network reprogramming service for sensor networks," *Proceedings of the 25th IEEE International Conference on Distributed Computing Systems, ICDCS 2005.*, pp. 7–16, 2005.

[209] J. Hui and D. Culler, "The dynamic behavior of a data dissemination protocol for network programming at scale," *Proceedings of the 2nd international conference on Embedded networked sensor systems*, pp. 81–94, 2004.

[210] C. Liang, R. Musăloiu-E, and A. Terzis, "Typhoon: A reliable data dissemination protocol for wireless sensor networks," *Wireless Sensor Networks*, pp. 268–285, 2008.

[211] K. Lin and P. Levis, "Data discovery and dissemination with dip," *Proceedings of the 7th international conference on Information processing in sensor networks*, pp. 433–444, 2008.

[212] T. Dang, N. Bulusu, W.-c. Feng, and S. Park, "Dhv: A code consistency maintenance protocol for multi-hop wireless sensor networks," *Wireless Sensor Networks*, vol. 5432, pp. 327–342, 2009.

[213] H. Fritsche *et al.*, "Tinytorrent: Combining BitTorrent and Sensornets," *M.Sc. Thesis in Computer Science, Trinity College Dublin.*, 2005.

[214] C. McGoldrick, M. Clear, R. Simon Carbajo, K. Fritsche, and M. Huggard, "TinyTorrents-Integrating Peer-to-Peer and Wireless Sensor Networks," *Sixth International Conference on Wireless On-Demand Network Systems and Services, WONS 2009.*, pp. 119–126, 2009.

[215] B. Cody-Kenny, D. Guerin, D. Ennis, R. Simon Carbajo, M. Huggard, and C. Mc Goldrick, "Performance evaluation of the 6LoWPAN protocol on MICAz and TelosB motes," *Proceedings of the 4th ACM workshop on Performance monitoring and measurement of heterogeneous wireless and wired networks*, pp. 25–30, 2009.

[216] S. Cohen and Y. Matias, "Spectral bloom filters," *Proceedings of the 2003 ACM SIGMOD international conference on Management of data*, pp. 241–252, 2003.

[217] R. Simon Carbajo, M. Huggard, and C. McGoldrick, "Opportunistic detection of relative mobility in wireless sensor networks," *Wireless Days (WD), 2010 IFIP*, pp. 1–5, October 2010.

[218] J. Gailly and P. Deutsch, "Zlib compressed data format specification version 3.3," *rfc1950, IETF Internet Draft*, May 1996.

[219] D. Gay, P. Levis, R. von Behren, M. Welsh, E. Brewer, and D. Culler, "The nesC language: A holistic approach to networked embedded systems," *Proceedings of the ACM SIGPLAN 2003 conference on Programming language design and implementation*, pp. 1–11, 2003.

[220] J. Banks and J. s. Carson, "Introduction to discrete-event simulation," *In Proceedings of the 18th Conference on Winter Simulation*, pp. 17–23, 1986.

[221] Atmel, "ATmega128L MicroController." `http://www.atmel.com/dyn/resources/prod_documents/doc2467.pdf`, September 2012.

[222] Memsic Corporation, "MIB520CB Datasheet." `http://www.memsic.com/products/wireless-sensor-networks/wireless-modules.html`, September 2012. Previously manufactured by Crossbow Technology Inc.

[223] M. Zuniga and B. Krishnamachari, "Analyzing the transitional region in low power wireless links," *First Annual IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks, IEEE SECON 2004.*, pp. 517–526, 2004.

[224] R. Jain, D. Chiu, and W. Hawe, "A quantitative measure of fairness and discrimination for resource allocation in shared computer systems," *DEC Research Report TR-301*, 1984.

[225] D. Chiu and R. Jain, "Analysis of the increase and decrease algorithms for congestion avoidance in computer networks," *Computer Networks and ISDN systems*, vol. 17, no. 1, pp. 1–14, 1989.

[226] MetaGeek LLC, "Wi-Spy 2.4x: Spectrum Analyzer for 2.4GHz ISM Band." `http://www.metageek.net`, April 2012.