Hard Real-Time Communication for Mobile Ad Hoc Networks

Barbara Hughes

A thesis submitted to the University of Dublin, Trinity College

in fulfillment of the requirements for the degree of

Doctor of Philosophy (Computer Science)

October 2006

Declaration

I, the undersigned, declare that this work has not previously been submitted to this or any other University, and that unless otherwise stated, it is entirely my own work.

Barbara Hughes

Dated: October 15, 2006

Permission to Lend and/or Copy

I, the undersigned, agree that Trinity College Library may lend or copy this thesis upon request.

Barbara Hughes

Dated: October 15, 2006

Acknowledgements

Writing a thesis is very much like running a marathon. There's the initial sprint after starting, then everything is progressing nicely, then things get rough and it seems like it's impossible to move on, despair and desperation sink in, and finally, the finish line is in sight and is met with relief and jubilation.

As in a marathon it's impossible to reach the finish without the help of many around you. I would like to thank my supervisor Professor Vinny Cahill for his time and support throughout. Also, I acknowledge the abundance of knowledge (and fun) in the Distributed Systems Group in Trinity College, Dublin, particularly Dr. Raymond Cunningham, who always listened to my questions even though I gave him headaches, Mr. Mark Gleeson, Mr. Anthony Harrington and Mr. Gregor Gaertner.

I would never have completed this marathon without the constant love and support of my friends, family, Mam and Dad. Special mention must be made to my Mam, Margaret, who (with a non-computing background) has listened to every facet of every program, project and finally thesis I have worked on, and has shown an incredible ability to listen, understand and give amazing insights which I value hugely. My aunt Mary will never see this thesis but I know she was with me every step of the way. All my family know how important Daisy and Dougal are. Lastly but most importantly, I wholeheartedly thank Dudley, who always knew exactly what to say to make things better, supplied a shoulder to cry on, a joke to get me through and without whom I would not be half the person I am. You have my heart.

Barbara Hughes

University of Dublin, Trinity College

October 2006

Abstract

The increasing availability of wireless local area networking, particularly ad hoc networking, has lead to the evolution of new application domains, such as inter-vehicle communication and communication between autonomous mobile robots. Real-time communication is essential to allow applications in these domains to be realised.

The characteristics of a mobile ad hoc network, typified by host mobility, unpredictable resource availability and time-varying connectivity, pose challenges for providing hard real-time communication guarantees in this domain. An approach adopted by previous real-time communication models is to adapt the communication time bounds to reflect the dynamics of the network. However, allowing time-bound adaptation implies that only soft real-time communication is available and, critically, that hard real-time communication is not.

This thesis describes a new communication model, the space-elastic model, to provide hard real-time communication for applications with guaranteed response-time requirements in wireless networks in general, and ad hoc networks in particular. In addition, a new real-time ad hoc routing protocol, the Space-Elastic Adaptive Routing (SEAR) protocol, is described, which provides the basis of a real-world implementation of the space-elastic model.

The contributions of this thesis are two-fold. Firstly, the space-elastic model is proposed to enable hard real-time communication by using specified geographical bounds to scope the area within which timely communication must be guaranteed in a wireless network. Due to network dynamics the space or actual coverage within which timely communication is guaranteed may be adapted over time with timely adaption notification to higher layers when a space adaptation occurs. No change is made to the specified communication time-bounds within the actual coverage. Secondly, a new location-aware real-time ad hoc routing protocol, SEAR, coupling time-bounded route discovery and maintenance with dynamic resource reservation has been designed and implemented. An evaluation of the space-elastic model, using SEAR, shows that time-bounded communication is possible within the actual coverage and that time-bounded notification can be provided if adaptation occurs.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

This thesis presents a new model, the space-elastic model, to address the problem of providing hard real-time communication guarantees in mobile wireless networks.

The space-elastic model uses geographical bounds to scope the region of the wireless network within which timely communication is guaranteed. The dynamics of such networks, particularly in the ad hoc case, imply that the geographical space within which real-time communication is guaranteed, referred to as the actual coverage, may change over time. If an adaptation of the actual coverage occurs, timely notification of the adaptation is communicated to space-elastic applications. Space-elastic applications must therefore be able to specify and interpret geographical bounds and adapt their behaviour to reflect the current actual coverage for real-time communication.

This thesis also describes a new real-time ad hoc routing protocol, SEAR, which is the basis for a real-world implementation and evaluation of the space-elastic model. Using SEAR, time-bounded transmission and adaptation notification within an adaptable space is experimentally validated.

This introductory chapter provides relevant background information on wireless networks, the requirements of hard real-time applications and the challenges of satisfying these requirements in a dynamic wireless network.

This chapter introduces the two main contributions of this thesis, namely a model supporting hard real-time communication in a dynamic wireless network, the space-elastic model, and a novel real-time ad hoc routing protocol, SEAR, whose implementation is used to evaluate that timely transmission and adaptation latency is achievable using the space-elastic model in the real-world.

Issues that will not be covered in this thesis are outlined in section 1.8. Towards the end of the chapter, a road-map for the remainder of this thesis is presented.

1.1 Wireless Networking

Wireless local area networks can be broadly classified into two types, infrastructure-based networks and ad hoc or infrastructure-less networks depending on the underlying network architecture (Murthy & Manoj 2004).

Infrastructure-based networks contain wireless hosts called access points (APs), who coordinate access to the wireless medium for all hosts under their control. The set of hosts under the control of an AP is called the basic service set. The AP has the ability to communicate with all hosts in the basic service set. The wireless hosts in the basic service set communicate with each other via the AP.

In contrast, an ad hoc network does not make use of a fixed infrastructure for communication between hosts. An ad hoc network may be created on the fly when required. The absence of a fixed infrastructure means that hosts themselves constitute the communication infrastructure. As hosts potentially move in and out of range of other hosts, the connectivity and network topology changes dynamically (Wang & Li 2002).

Communication between wireless hosts requires the received signal strength (RSS) to be adequate to connect to another wireless host. Variations in the RSS due to, for example, the movement of wireless hosts, different terrains which may include hilly or mountainous areas, wooded or forested rural areas, urban areas with multi-story buildings or low-density suburban areas (Karimi. & Krishnamurthy 2001), and transmission power which may depend on battery life (Gomez et al. 1999), impact the ability for a host to maintain a connection with another wireless mobile host or access point. The unrestricted host movement leads to unpredictable connection quality between hosts. Greater mobility increases the fluctuations in link connectivity, the volume of topological updates, the time spent processing the updates (e.g., for route discovery protocols), and congestion due to increased update transmissions and retransmissions. The impact of host mobility is exacerbated in the ad hoc case where the rate of link failure due to host mobility is the primary obstacle to routing (McDonald & Znati 1999).

Communication in wireless networks is an expensive operation both in terms of bandwidth and energy consumption. Any additional control packet overhead (e.g. for route discovery and resource reservation) must be kept to a minimum. Additional control packets increase the competition for network resources (e.g. bandwidth) for all transmissions. In mobile ad hoc wireless networks, in particular, the available bandwidth is very limited and some wireless devices have severe energy constraints limiting battery life and increasing the time-varying availability of routes and resources and the dynamics of the wireless network (Gomez et al. 2001).

1.2 Hard Real-Time Application Requirements

The real-time communication paradigm can be expressed as follows: *"The achievement of bounded and known message delivery delays, in the presence of disturbing factors such as other real-time traffic, variable load or faults"* (Veríssimo & Rodrigues 2001). There are two main classes of real-time communication: **hard real-time** and **soft real-time**. The criticality of the real-time communication is determined by the cost of timeliness failure.

The **hard real-time** class specifies that the communication *must always* be timely with failure to do so incurring a high cost. Some examples are, inter-vehicle communication for automated driving or vehicle-to-roadside communication in Intelligent Transport Systems (ITS). In both scenarios failure to communicate within known time-bounds is potentially life threatening. In contrast, in the **soft real-time** class, occasional failure to meet timeliness requirements is acceptable without critical consequences, e.g., video streaming for video-on-demand applications.

This thesis addresses support for hard real-time communication in wireless networks, therefore only the properties of hard real-time communication are of interest here.

Approaches to supporting hard real-time communication in static networks (Kaiser & Mock 1999, Veríssimo & Casimiro 2002), have assumed the following conditions are satisfiable (Veríssimo & Rodrigues 2001):

1. *Known network load* - implying both the number of participants in a real-time communication, e.g., the members of a group-based communication, and the maximum system load are known and bounded;

2. *Guaranteed connectivity* - implying that the network medium is sufficiently reliable to guarantee accessibility and communication with all known network participants and network partitions are controlled;

3. *Deterministic communication latency* - given condition 1, the number of participants and the maximum network load are known in advance, thus, medium-access latency can be known and bounded by the worst-case load conditions, in the absence of faults. Given condition 2, connectivity is guaranteed, and thus, the end-to-end communication latency can be known.

4. *Guaranteed resource availability* - given condition 1, the worst-case load can be known and offline static scheduling is used to reserve resources to satisfy hard real-time transmissions based on this worst-case bound. Given condition 2, participants have guaranteed connectivity, and therefore, accessibility of the schedules resources when required.

These conditions for hard real-time communication have been satisfied in static networks, where the assumptions of the network, i.e., known bounds on network participants, guaranteed accessibility, known and bounded network resources and the static scheduling of resources, can be guaranteed. For example, the TAO Real-time Event Service (Schmidt et al. 1998) and the Real-time Event Channel Model for the CAN-Bus (Kaiser & Mock 1999), use static schedules to perform real-time medium-access scheduling off-line using a reservation-based scheme to avoid collisions by statically planning the transmission schedule. In addition, communication models that use centralised intermediate components, such as, brokers (OMG-CORBA 1995) or dispatchers (G.Cugola et al. 2001), rely on the assumption that connectivity is guaranteed.

The assumptions on which the support for hard real-time communication in static networks rely are not applicable for wireless networks where the dynamics of the wireless network is the primary obstacle to providing hard real-time communication, and is described in the next section.

1.3 Challenges to Hard Real-Time Communication in Wireless Networks

The previous two sections describe the characteristics of wireless networks and the conditions for achieving hard real-time communication guarantees, respectively. In this section, the influence of the characteristics of wireless networks on achieving hard real-time communication guarantees are presented.

The movement and limited battery life of wireless hosts lead to time-varying connectivity between hosts in the network. For example, in an ad hoc network, the lack of a fixed infrastructure and the limited power of wireless mobile hosts limits their transmission range and means that wireless hosts are designed to serve as routers if needed. The result is a distributed multi-hop network with a time-varying topology where routes are typically short-lived (Wang & Li 2002). The impact of a dynamic ad hoc network topology on hard real-time communication is illustrated in Figure 1.1.

- A, - host A moves out of communication range of a forwarding host and is no longer included on the route from the source S. Hosts that depend on host A for forward propagation of hard real-time communication from S will not receive hard real-time transmissions from S for a non-deterministic period until a new route including these hosts is discovered.

- B, - link B between two hosts fails which means that for a non-deterministic period the forward propagation of hard real-time communication from the source S is no longer possible.

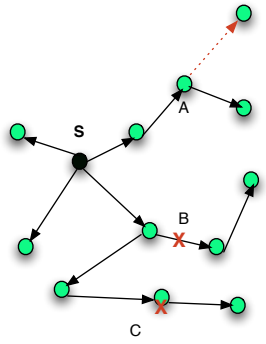


Fig. 1.1: The affect of ad hoc network dynamics on support for hard real-time communication

- C, - the failure of a forwarding host, C, means that for a non-deterministic period all hosts on the route dependent on host C will no longer receive hard real-time communication.

Given the dynamicity of a wireless network neither of the conditions: known network load, nor guaranteed connectivity, for the availability of hard real-time communication in section 1.2 are applicable in this domain.

In infrastructure-based wireless networks, access to the wireless medium is controlled by the AP. For example, using the Point Coordination Function (PCF) of IEEE 802.11, a polled host is guaranteed contention-free medium-access. However, a restriction of this approach is that the interval between polls, and thus, wireless access, is non-deterministic and is not suitable to guarantee the timeliness of hard real-time transmissions. Wireless transmissions in ad hoc networks are typically broadcast using a shared contention-based physical communication medium. Competition occurs prior to each medium-access leading to non-deterministic time-bounds for successful transmission. Collisions in wireless communications can be caused by simultaneous transmissions by two or more wireless hosts sharing the same frequency band, or as a result of the hidden terminal problem (Tobagi & Kleinrock 1975). Multi-hop ad hoc communication has the potential to incur this non-deterministic medium-access latency at each hop leading to a non-deterministic latency for the communication. Attempts have been made to reduce contention for medium-access control in ad hoc networks, for example, using the Wireless Token Ring Protocol (Ergen et al. 2004), deterministic medium-access is guaranteed to a set of hosts. However, a restriction of this protocol is that the latency to join the set is non-deterministic, and thus, is not suitable for hard real-time communication guarantees. The condition for deterministic communication latency from section 1.2 is not available in dynamic wireless networks without restrictive assumption about the dynamics of the network.

The topology changes introduced by host mobility and wireless link failures must somehow be communicated to other hosts. Since wireless and computation resources, for example, bandwidth and battery power, are limited in wireless networks, any overhead must be kept to a minimum and additional communication delays due to an increase in the volume of topological updates must be avoided. In addition, due to the scarcity of resources the dynamics of the network means that the accessibility of reserved resources is time-varying, with the implication that resources may not be available to satisfy the constraints of hard real-time communication when required. The condition for guaranteed resource availability of section 1.2 is not assumed in a dynamic wireless network.

The characteristics of a dynamic wireless network make supporting the conditions for hard real-time communication, described in section 1.2, challenging. The focus of this thesis is a communication model to support hard real-time communication guarantees regardless of the dynamics of the wireless network.

1.4 Alternative Proposals

Approaches such as COSMIC for the CAN-Bus (Kaiser et al. 2005) or Time-Triggered Protocols for wireless networks (Huber & Elmenreich 2004), described in chapter 2, achieve hard real-time communication by relying on assumptions relating to the number of participants, load patterns and the scheduability and availability of resources to satisfy known communication guarantees. These static assumptions are not applicable in dynamic wireless networks where the dynamics of the network negate the applicability of the approach.

One particularly influential approach is the Timely Computing Base (TCB) (Veríssimo & Casimiro 2002), which addresses the problem of achieving and maintaining real-time guarantees in dynamic environments with uncertain timeliness. Veríssimo and Casimiro define the TCB system model to be composed of a number of participants or processes exchanging messages between several sites or hosts of the same system, with the sites interconnected by a communication network - the *payload* network. It is assumed that the payload network does not have to guarantee deterministic communication latency, i.e., no bounds on packet delivery or processing latency are assumed. The TCB architecture also encompasses a control system which comprises the TCB modules, interconnected by a medium called a *control channel*. It is assumed that the control system of the TCB guarantees deterministic communication latency. The availability of a TCB system has the potential to reduce the impact of the varying synchrony of the payload system on applications, e.g., by detecting the failure of timeliness properties in the payload system and notifying the change in the assumed synchrony to the application.

An application using the TCB has an assumption about the timeliness properties that the environment can guarantee. The relationship between the assumption of timeliness properties and the guarantees provided by the environment is the *coverage*. In a system with uncertain synchrony the assumed coverage of a timeliness property will not remain constant. If the environment starts to degrade, the probability of timing failures starts to increase, and the coverage of the timeliness assumptions starts to decrease.

The TCB model assumes that all timeliness properties are adaptable to maintain the probability of coverage irrespective of the current dynamics of the network. The time-elastic ($T\epsilon$) class was introduced for applications for which it is more beneficial to execute in a degraded mode (or with a lower QoS, e.g. increasing latency for operations), than to stop or to more critically have unexpected failures due a change in the assumed coverage (Veríssimo & Casimiro 2002).

Time-elastic adaptation uses a trade-off between time and space, where the space encompasses the distributed network of unknown synchrony. Using time-elastic adaptation, the time is adapted to achieve a probability of service coverage to the network, i.e., within a known space.

Hard real-time applications are by definition not time-elastic, i.e., timeliness properties must be guaranteed with the failure to do so potentially catastrophic. The trade-off between time and space is, however, also interesting when applied to providing guarantees for hard real-time communication and is the basis for the space-elastic model.

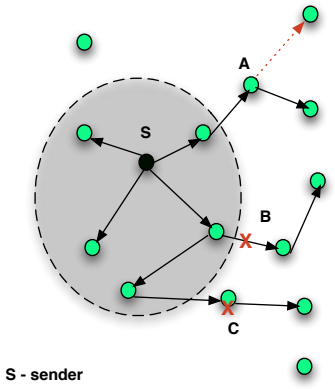


Fig. 1.2: Space where hard real-time communication is guaranteed

1.5 Space-Elastic Model

A model supporting hard real-time communication guarantees in wireless networks must overcome the impact of the characteristics of dynamic wireless networks discussed in section 1.3. In general, hard real-time communication may not always be possible within the complete network topology. For example, in Figure 1.2, the dynamics of the network, i.e., that host A has moved, link B has failed and host C has failed, implies that hard real-time communication is only available within a portion of the wireless network, the actual coverage, identified by the shaded region.

In the space-elastic model hard real-time communication is guaranteed in the current actual coverage. Given the dynamics of a mobile wireless network the actual coverage may be adapted, however, the timeliness properties guaranteed within the actual coverage are not adapted. In the space-elastic model this is the utilisation of the trade-off between time and space.

The space-elastic model supports a class of real-time applications, the space-elastic ($S\epsilon$) class, to characterise applications with space-elastic properties, i.e., the set of real-time applications that can operate correctly in an adaptable space.

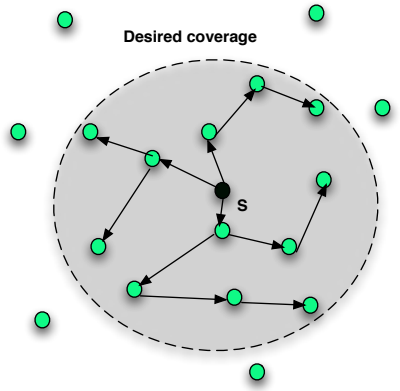


Fig. 1.3: Space-elastic application communication bounds

A space-elastic application identifies a desired space, e.g., geographical locations in a wireless network, within which timeliness properties should be guaranteed, as illustrated in Figure 1.3, where the space-elastic application is identified by S.

An example space-elastic application is a traffic light communicating a change of signal to approaching vehicles. In this scenario, the space within which timely traffic light communication is desired is the space in front of the traffic light that equates to the braking distance travelled by a

vehicle moving at the maximum speed limit for the road. This space can be calculated. The dynamics of the wireless network within the desired coverage, as illustrated in Figure 1.4, where host movement and host and link failure have lead to a network topology where real-time communication is not possible in the complete desired space but rather in the actual coverage space illustrated.

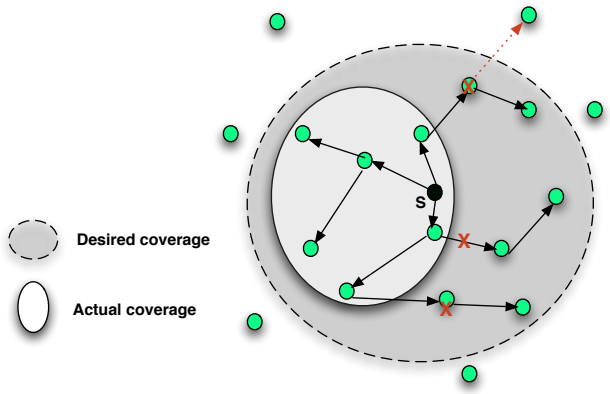


Fig. 1.4: Actual coverage for hard real-time communication

The safety and progress of a real-time application may still be guaranteed if notification of a change in network state and the corresponding change to the actual coverage of real-time communication is available to the real-time application within a known time-bound. Time-bounded notification of a change in the network state, and more explicitly the actual coverage, allows applications to change their behaviour based on the current actual coverage. For example, in the traffic light scenario, if an adaptation of the actual coverage occurs the safety of pedestrians and vehicles can still be guaranteed if the traffic light is delivered an actual coverage adaptation notification within a calculated time bound and adapts its behaviour based on the adaptation notification and actual coverage received, to guarantee safe passage past the traffic light.

Using a combination of timely communication within an adaptable space and timely adaptation notification, the space-elastic model guarantees progress for real-time space-elastic applications within a dynamic network. The dynamics of a wireless network are unrestricted and may fluctuate between high and low dynamicity. When the network is experiencing low dynamicity the space-elastic model guarantees application progress by supporting timely communication. When the network is experiencing high dynamicity, the real-time application can still make progress using the actual coverage available in the adaptation notification to form the basis for a behavioral adaptation by the application.

The design of the space-elastic model is described in detail in chapter 3.

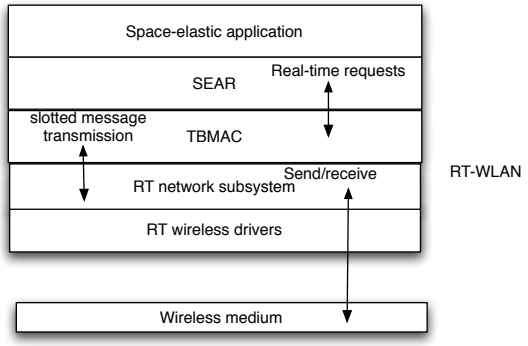


Fig. 1.5: Real-time architecture

1.6 Real-Time Architecture

A real-world implementation of the space-elastic model requires real-time guarantees at all layers of the protocol stack. To achieve this a real-time architecture was implemented which incorporated a cross-layer design approach providing real-time guarantees at each layer and combining to support the real-time guarantees of the space-elastic model. The layers and interactions between the layers, illustrated in Figure 1.5, are as follows:

Real-time network subsystem

At the lowest level of the protocol stack, real-time wireless drivers and a low-level network interface, RT-WLAN, were implemented to guarantee time-bounded message transmission and reception, (Hughes et al. 2006). A higher layer, (e.g., at the medium-access control layer), invokes the functionality of RT-WLAN for time-sensitive transmission and reception using the new real-time drivers. No medium-access control guarantees are available at this layer. The guarantees provided by RT-WLAN are the basis for the guarantees at each higher layer of the implementation.

Time-bounded medium-access control (TBMAC)

A contention-based wireless medium, such as IEEE 802.11 DCF (IEEE 2005), incurs unpredictable medium-access latency due to competition for wireless access prior to each transmission. A prerequisite for timely multi-hop communication is timely medium-access. The Time-Bounded Medium-Access Control (TBMAC) protocol (Cunningham & Cahill 2002) is based on TDMA with dynamic but predictable slot allocation. TBMAC uses an atomic broadcast protocol to

achieve distributed agreement on slot allocation and employs location information to minimise contention for slots. TBMAC is implemented as a layer above RT-WLAN.

Space-Elastic Adaptive Routing (SEAR)

SEAR is a real-time routing protocol for ad hoc networks. SEAR guarantees transmission and adaptation notification latency for space-elastic applications.

The real-world implementation of TBMAC provides timely transmission latency when all hosts are fully connected and only single-hop communication is required. SEAR, integrates with TBMAC to provide real-time communication guarantees in a multi-hop network.

Space-elastic application

The real-time application layer interfaces with SEAR to specify the geographical bounds where real-time communication is required and interpret the geographical bounds corresponding to the actual coverage where real-time properties are supported. Using the SEAR interface, the space-elastic application specifies: the real-time attributes of the communication, the desired space for real-time communication and the maximum time-bound for route discovery. Adaptation notification feedback is provided by SEAR in a format agreed with the application and interpreted by the space-elastic application for decision-making on whether a change of behaviour is required.

The implementation of SEAR is described in detail in chapter 5, and the evaluation of the space-elastic model using SEAR is described in chapter 6.

1.7 Thesis

This thesis presents a new model, the space-elastic model, to provide hard real-time communication for applications with guaranteed response-time requirements in multi-hop wireless networks in general, and ad hoc networks, in particular. In addition, a new real-time ad hoc routing protocol, the Space-Elastic Adaptive Routing protocol, is presented which is the basis for a real-world implementation and evaluation of the space-elastic model.

The challenges to supporting hard real-time communication in a wireless network were described in sections 1.1 and 1.3. To reduce the impact of a dynamic network topology, previous research has proposed a trade-off between time and space to reduce the impact of the dynamics of the network by adapting the time-bounds available for real-time communication, as discussed in section 1.4. This time-elastic approach is limited to applications that can tolerate changes to communication time-bounds without serious consequence, i.e., only soft real-time applications are supported.

A different approach is required to accommodate hard real-time communication guarantees where timeliness properties are not adaptable. The space-elastic model supports guarantees for hard real-time communication within a known and adaptable space in a wireless network and is the main contribution of this thesis.

The space-elastic model must guarantee that hard real-time communication is supported regardless of the current dynamics within the wireless network. To achieve this, and guarantee progress is made by a real-time application, all changes to the actual coverage within which real-time properties are guaranteed, are notified to the application within a known time bound. A space-elastic application may adapt its' behaviour based on the new actual coverage.

The second contribution of this thesis, is a novel real-time ad hoc routing protocol, SEAR. SEAR is used as a basis to evaluate the timeliness of real-time communication and adaptation notification in the real world.

1.8 Issues Not Covered

The space-elastic model supports timely communication in an adaptable space for both stationary and mobile real-time applications. An example of a stationary real-time application is a traffic light communicating change of traffic light signals within a bounded space for real-time communication. An example of a mobile real-time application is a vehicle platooning scenario, (Hartenstein et al. 2001), where a vehicle communicates driving instructions to vehicles in the platoon. Space-elastic support for both mobile and stationary real-time applications is fully described in chapter 3.

The implementation, and thus the evaluation of the space-elastic model presented in chapter 6, supports only stationary transmitting real-time applications. Both mobile and stationary receiving hosts for real-time communication are supported in the implementation, i.e., the implementation models a similar scenario to the traffic light scenario described previously.

To support the mobility of transmitting real-time applications necessitates a guarantee that allocated resources move with the mobile host and are available when required regardless of the current location of the host. To achieve this using the current real-world implementation requires the addition of a dynamic resource scheduling layer that includes mobility prediction and proactive resource reservation and interfaces with TBMAC to support timely allocation and re-allocation of slots depending on the movement of the transmitting host.

Integrating TBMAC with a dynamic resource scheduling layer to facilitate the inclusion of mobile transmitting applications is the focus of future work.

The wireless networks considered in this thesis, for example, in discussing related work on medium-access control and routing in chapter 2, are potentially large, local area networks. With the increased research into wireless local area networks, particularly ad hoc networks, in recent years new application domains, such as inter-vehicle communication and communication between mobile robots, have evolved. The provision of quality of service guarantees, and particularly real-time communication guarantees, to applications in these domains is an open research area, and thus, is the focus of this thesis.

1.9 Evaluation

The space-elastic model provides guaranteed transmission latency within a defined space with time-bounded space adaptation notification if the space within which the time-bounds are achievable changes. An evaluation of the model must verify that these transmission time-bounds are satisfied, within a known and allowable jitter, regardless of the dynamics of the wireless environment. Furthermore, changes to the actual coverage where real-time communication is supported must be notified to the originating host within known time-bounds.

These properties of the space-elastic model are experimentally validated using the multi-hop ad hoc real-time routing protocol, SEAR.

1.10 Road Map

The remainder of this thesis is structured as follows

Chapter 2 presents the state of the art in achieving real-time guarantees in wireless networks with particular emphasis on medium-access control and routing and resource reservation.

Chapter 3 presents the design of the space-elastic model and derives a formal specification of the properties supported by the model.

Chapter 4 presents the design of the real-time ad hoc routing protocol, SEAR.

Chapter 5 presents the implementation of SEAR.

Chapter 6 experimentally validates the properties of the space-elastic model presented in Chapter 3, for stationary transmitters.

Chapter 7 summarises and discusses future work.

1.11 Summary

This chapter outlined the goals of this thesis and also highlighted issues that will not be addressed by this work. Background information relating to wireless networks, hard real-time communication and the challenges to supporting hard real-time communication in wireless networks which are relevant to the rest of the thesis were also presented.

In addition, this chapter outlined the two main contributions of this thesis: the space-elastic model and the real-time ad hoc routing protocol SEAR.

Chapter 2

Related Work

The objective of the work described in this thesis was the design of a new communication model to provide hard real-time guarantees for communication in dynamic wireless networks supported by a new real-time routing protocol, SEAR. This chapter reviews related work with an emphasis on the applicability of existing communication models to provide timeliness guarantees in wireless networks and the extent to which the underlying assumptions and properties of these models are extensible to a dynamic environment. Existing approaches to achieving real-time communication guarantees in both wired and wireless networks emphasising the degree to which the provision of real-time guarantees is possible are also reviewed.

The networking property that is most important for hard real-time communication is timeliness. Timeliness is a property that specifies that a predicate $\mathcal{P}$ will be true at a given instant of real time, for example, that any message delivery completes within T from its transmission.

Hard real-time communication requires timeliness guarantees at both the medium-acccess control (MAC) and routing layers. The timeliness property of a MAC protocol refers to a bounded delay for a host to gain access to the medium and transmit a message. The timeliness property of a real-time routing protocol refers to a known time bound, i.e., the transmission latency, within which messages must be delivered on discovered real-time routes. To guarantee the timeliness of hard real-time communication, route discovery and rerouting must also be time-bounded, to ensure that real-time routes are available for hard real-time message transmission.

An ad hoc wireless network is characterised by a dynamic topology of potentially mobile hosts. A host in an ad hoc network is limited by the transmission range of its wireless interface. A requirement to communicate further than the transmission range needs the cooperation of one or more hosts within

transmission range to forward messages when required resulting in a multi-hop ad hoc network. If multi-hop real-time communication is required the timeliness properties for medium-access control and routing must also be extensible to a multi-hop environment.

The discussion of related work presented in this chapter begins with a review of existing communication models providing varying levels of timeliness. Following this, a discussion of current approaches to providing real-time guarantees in infrastructure-based networks of varying scale with an emphasis on the applicability and extensibility of each approach when applied in a dynamic ad hoc network is presented. In section 2.3, existing medium-access control and real-time routing protocols for wireless networks are presented and the extent to which hard real-time guarantees are provided by these protocols is discussed.

This chapter concludes with a discussion of the issues that must be resolved to achieve hard real-time communication in dynamic wireless networks.

2.1 Communication Models

A model defines the set of guarantees on which an application using the model relies. For example, a real-time communication model defines the communication guarantees, e.g., timeliness properties, on which an application with real-time requirements relies.

The terms "synchronous" and "asynchronous" have many different meanings in the context of distributed systems (Veríssimo & Rodrigues 2001). In the context of this chapter, the meaning of synchrony refers to the ability of a system to assume worst-case times for actions, for example, message delivery within a known time bound. A communication model has synchronous properties if the guarantees of the model allow decisions to be taken based on the passage of time. For example, in a synchronous routing protocol if a message is expected at a given time, failure to receive a message at this time is a fault. Using an asynchronous model means that the passage of time provides no information. For example, using an asynchronous model with no time bounds for message delivery, forwarding a message and processing a received message can be arbitrarily slow at different hosts in the network.

Synchronism is expressed in terms of timeliness properties. For example, a definition of synchrony for time-bounded message delivery is: Any message delivered is delivered **within** a known bound T_{Dmax} **from** the time the message transmission request was made.

System models can be classified according to the degree of synchrony that they provide:

1. Time bounds do not exist - *asynchronous*;

2. Time bounds exist but are not known or are transiently satisfied only - *partially synchronous*;

3. Time bounds exist and are known - *synchronous*

The communication models described in this section exhibit varying synchronism properties, ranging from synchronous to asynchronous, and correspondingly varying timeliness properties, ranging from time-bounded to unbounded.

When reviewing a model an emphasis is placed on the guarantees provided by the model and the assumptions of the model on which the guarantees rely. The communication models with the weakest and strongest synchronism properties, asynchronous and synchronous models, are described first, with the other models described relative to these extremes. This section concludes with an analysis of the models in terms of the applicability of the guarantees and assumptions of each model for providing hard real-time communication in wireless networks.

2.1.1 Asynchronous Communication Model

Asynchronous communication models do not guarantee temporal constraints, i.e., the passage of time does not provide any information about an action, e.g., in message delivery it is impossible to distinguish between a non-crashed (but slow) host and a crashed host, or, a lost message and a late message.

The properties of the asynchronous communication model are as follows:

1. Processing[1] delays are unbounded or unknown.

2. Message delivery delays are unbounded or unknown.

3. Rate of drift of local clocks is unbounded or unknown.

4. Difference between local clocks is unbounded or unknown.

Given these properties of the asynchronous model, applications using this model must be satisfied with liveness guarantees only, i.e., that a predicate $\mathcal{P}$ will eventually be true. Due to the limited guarantees available there are no restrictive assumptions made about the operational environment on which the guarantees of the model rely, i.e., communication using the model is *eventually* reliable. Liveness guarantees are not compatible with the expectations of real-time systems where time is used as an artefact to guarantee synchronisation with the environment and all participants in a communication.

[1] Processing encapsulates message processing, scheduling etc.

The Time-Free model or Time-Free Asynchronous Model (Fischer et al. 1983) is an example of an asynchronous model. The Time-Free model describes the outputs and state transitions required in response to inputs, with no time bounds being placed on when these state transitions should occur.

The impossibility of implementing critical services such as consensus and membership election using a time-free model (Fischer et al. 1983) shows that an asynchronous model has limitations that compromise its usefulness. A problem for applications using an asynchronous model is the inability to determine a slow host from a crashed host. The trustworthiness of failure detection, i.e, the correct detection of a crashed host, is constrained by the difficulty of differentiating genuine failures from unpredictable processing and message delivery delays.

2.1.2 Synchronous Communication Model

As noted above a synchronous model provides guarantees based on the passage of time. Timeliness guarantees, e.g., execution of actions at known time instants or intervals, such as, "any message is delivered within a delay T_d" or "task k is executed with a period of T_k", are provided by such models.

The properties of a synchronous communication model are as follows :

1. Processing delays have a known time bound.
2. Message delivery delays have a known time bound.
3. The rate of drift of local clocks has a known bound.
4. The difference between local clocks has a known bound.

To provide these timeliness guarantees requires significant assumptions about the environment encapsulating the network infrastructure and the participating hosts. For example, the first property assumes a homogeneity of processor power and process scheduling amongst all hosts, and the second property assumes time-bounded medium-access and routing layers to guarantee message delivery times at all hosts. The last two properties make assumptions about clock synchronisation, clock precision and bounds on clock reading errors. The assumptions on which the guarantees of the synchronous model rely limit the applicability of the model to networks and participants that can satisfy these assumptions only.

2.1.3 Partially Synchronous Communication Models

Real-time communication requires timeliness guarantees that are realisable by the synchronous communication model. However, large-scale, unpredictable and unreliable network infrastructures, of

which ad hoc networks are a good example, are not adequate environments for supporting the assumptions of the synchronous model.

The asynchronous model is appropriate in networks of uncertain timeliness as there is little assumption made about the environment. Asynchronous models do not provide the timeliness guarantees necessary for real-time communication or even to guarantee the deterministic outcome of basic distributed system problems, for example, consensus and leader election (Fischer et al. 1983).

Most practical distributed systems have a different assumption to that of the asynchronous model by observing that: a) the system is not *always* asynchronous and, b) the system is not asynchronous *everywhere*. These observations have lead to a class of systems that are called partially synchronous (Fischer et al. 1983).

In an environment of partial synchrony either parts of the system or epochs of its operational life can be reliably considered synchronous. In these cases, bounds on response time and other variables can be defined that hold on a subset of the system, or during limited periods of its operation only.

Partially synchronous communication models provide guarantees to applications with timeliness requirements that operate in environments of uncertain timeliness. The assumptions of the partially synchronous communication model are:

1. Some of the system properties, i.e., processing or message delivery delays or the rate of drift of local clocks or the difference between clocks, have a known bound.

2. If any of the system properties do not hold, a known bound may not exist or may be too large (to be practical).

The Timed Asynchronous Model (Cristian & Fetzer 1999) and the Quasi-Synchronous Model (Almeida & Veríssimo 1996), are examples of partially synchronous communication models.

Timed Asynchronous Model The timeliness properties in the Timed Asynchronous Model are conditional, i.e., only when the operational context of the system is synchronous does the system (have to) achieve correct behaviour within known time-bounds. The timed model allows practically needed distributed services, e.g., clock synchronisation, consensus, election and atomic broadcasts to be implemented (Cristian & Fetzer 1999), in an environment of uncertain timeliness.

The assumptions of the Timed Asynchronous Model are that processing and message delivery delays are variable but the rate of drift of local clocks is known and constant. Thus, the Timed Asynchronous Model assumes clock synchronisation, clock precision and bounded clock reading error exist.

Quasi-synchronous Model In the Quasi-synchronus Model, worst-case bounds on the timeliness properties of processing and message delivery delays and clock drift exist, $T_{x_{max}}$, but may be so much higher than the average case, $T_{x_{avg}}$, that $T_{x_{max}}$ becomes a useless bound, and other values closer to the average case must be used. The assumption of a shorter artificial bound, $T^{1}{}_{x_{max}}$, closer to the operational bound available, increases the expected responsiveness, but also increases the probability of timing failures, i.e., when the environment and, correspondingly, the achievable bounds change. However, if timing failures are detected when they occur a reliable system can be built that will work synchronously when the environment allows and react to preserve correctness of the system. The Quasi-synchronous model assumes that when the environment transitions from a synchronous state that timing failures can be detected and the application can change behaviour accordingly. This property of the quasi-synchronous model is particularly relevant for dynamic networks where operational contexts are available to specify real-time constraints with a path of graceful degradation through which the system progresses when guarantees cannot be maintained or faults occur.

To satisfy these timeliness properties the quasi-synchronous model assumes a known and bounded processing speed and clock drift rate exist.

A partially synchronous system provides support for building soft real-time applications that are dependable, in the sense that they offer resilience to the failure of timing assumptions. If the system is not *always* asynchronous there will be periods when, for example, message delivery and message processing is predictable. In these synchronous periods the trustworthiness of crash failure detectors is reliable. There are some systems that rely on this hypothesis, for example, Chandra & Toueg's (1996) asynchronous systems with failure detectors, to continue operating in an environment of uncertain timeliness.

2.1.4 Analysis

The asynchronous model does not provide timeliness properties, and thus, is not suitable to support hard real-time communication. The synchronous model provides guarantees based on the passage of time, and thus, provides timeliness guarantees that are suitable for hard real-time communication. However, the assumptions of the synchronous model limit the applicability of the model. Large-scale, unpredictable and unreliable networks, for example, in the ad hoc domain where dynamic connectivity, time-varying numbers of participating hosts, heterogeneous wireless devices and limited and varying resource availability are typical, are not adequate environments for enforcing the assumptions of the

synchronous model.

The partially synchronous communication model provides conditional timeliness properties based on the achievable synchrony of the environment, which may fluctuate between synchronous and asynchronous and fluctuate in different portions of the network. The conditional guarantees of the partially synchronous model imply that hard real-time application requirements are satisfied in non-deterministic periods when the environment is exhibiting synchronous behaviour, i.e., essential time-bounds for hard real-time communication are satisfiable. However, the transition of the environment to exhibit asynchronous behaviour is potentially catastrophic to hard real-time communication.

The assumptions about the environment on which a partially synchronous model relies are the most realistic in a dynamic wireless domain. However, missing from the partially synchronous models discussed here, is a specification of the time-bound to identify if a change in the current stability or operating context of the network has occurred and therefore if a corresponding change in the behaviour of the real-time application is required, i.e., to maintain dependability when timing failures occur. Furthermore, the transition time-bound, e.g., to a different operational context in the quasi-synchronous model, is dependent on the time to detect and react to a network change, and is not specified in any model.

2.2 Timeliness in Wired Networks

Advances in distributed systems for industrial automation, for example, control systems and factory automation systems, have led to increased research into providing real-time guarantees in small-scale infrastructure-based networks composed, for example, of distributed sensors and actuators connected by a field-bus architecture.

In this section, the time-triggered (TT), (Schlatterbeck & Elmenreich 2001), and the event-triggered (ET), (Bosch 2001), approaches to building small-scale infrastructure-based real-time systems are discussed. Following this an approach to achieving real-time communication in larger local area networks, using a Timely Computing Base (Veríssimo & Casimiro 2002) is presented. An analysis of the timeliness guarantees achieved by these systems and the assumptions on which they rely follows, with an emphasis on the applicability of these approaches in a dynamic wireless network.

2.2.1 Time-Triggered Systems

Time-triggered protocols (Schlatterbeck & Elmenreich 2001) are a class of protocols in which control signals are derived solely from the progression of time. Periodic clock interrupts are the only interrupts

in each host of a time-triggered system, partitioning continuous time into a sequence of equally-spaced intervals. The TT protocol has been used for safety-critical distributed real-time systems that use sensor-bus technology, i.e., bus-organised sensor-driven process control systems where high accuracy and high reliability are important (Nader & Wise 1990).

A host in a time-triggered system consists of three major subsystems: the host computer, which is capable of executing real-time computational tasks, the communication controller (CC) and the I/O subsystem, which interfaces with the sensors and actuators in the environment.

Medium-access control in TT protocols is controlled by a conflict-free TDMA (time-division-multiple-access) (Tanenbaum 1988) strategy. Every host is assigned a unique sending slot in each TDMA round. Every CC contains a dispatching table, known as message descriptor list (MEDL). The communication controller sends a message whenever the global time reaches a value that is equal to a timestamp of a message in the local MEDL. The MEDLs are constructed before run-time and are shared amongst all hosts on the network.

In a distributed time-triggered real-time system, it is assumed that the clocks of all hosts are synchronised to form a global notion of time and every message transmission includes a timestamp of this synchronised time. Using this global knowledge of time all hosts know the time at which messages will be sent, the identification of the sending hosts, and receiving hosts know a priori the expected time of arrival of each message. Using the distributed MEDL, every host has sufficient information to schedule local activities, (e.g., sampling of the environment), to minimise any impact on message transmission or reception reducing the achievable latency jitter to the order of microseconds, (Kopetz 1998).

2.2.2 Event-Triggered Systems

Event-triggered protocols derive control signals from non time-related events occurring outside or inside the computer system, for example, using an interrupt mechanism to signal the occurrence of an event to the CPU (Kopetz 1998). Event-triggered systems are suitable for sporadic events when the possible event types are known but the time and frequency of their occurrence is not known. An ET system requires a dynamic scheduling strategy to activate the appropriate software task to service the sporadic event when necessary. The CAN-Bus (Bosch 2001) is an example of an event-triggered system.

CAN is a broadcast bus that provides message identifiers to characterise the contents of a message rather than the inclusion of a source or destination address in the message. The CAN message identifier is exploited for prioritised CAN bus arbitration providing global distributed priority-based

message dispatching of all messages that are ready to be sent. The CAN standard 2.0A, defines an 11-bit message identifier. A consequence of this short identifier is that it is impossible in most application-level protocols to assign individual priorities to messages, instead only a very limited priority assignment of message classes is achieved (Kaiser et al. 2005).

In Kaiser et al.'s (2005) COSMIC, the longer, 29-bit, message identifier of CAN standard 2.0B is used, to achieve explicit priority control and identify network hosts and events. The 29-bit identifier is subdivided into 1) an 8-bit priority field; 2) a unique 7-bit host identifier and 3) a 14-bit event tag. The priority field allows 256 priority levels to be identified, which are used to identify different classes of real-time messages, for example, to allow the assignment of the highest priority to hard real-time messages. The host identifier is used to guarantee unique message identifiers in CAN, where equal message identifiers would result in an arbitration conflict that cannot be resolved. The unique host identifier is dynamically assigned when the host is attached to the network.

COSMIC uses a publisher/subscriber paradigm and an abstraction called an event channel for transmissions from each real-time class to support the timeliness properties required by that class.

Hard real-time messages require message delivery guarantees that are achieved using slot reservations in a TDMA approach. The goal of the reservation-based scheme is to avoid collisions by statically planning the hard real-time transmission schedule. The correctness of the reservations regarding timing conflicts, i.e., that transmissions never overlap in time, is checked by an admission test, which is performed off-line prior to the addition of a new reservation. Using this scheme, conflicts between hard real-time messages are avoided.

Hard real-time event channels provide a low latency jitter for periodic events under the assumption of a known and anticipated number of network failures. To guarantee that all participating hosts agree on time slots for message transmission, COSMIC assumes clock synchronisation amongst all participating hosts providing a global notion of time. COSMIC exploits the priority mechanism of CAN to guarantee that a hard real-time message is transmitted in the slot reserved for the transmission, by assigning the maximum possible priority to a hard real-time message when the reserved time slot is reached.

A message transmission on a CAN-Bus cannot be preempted. A transmission of a non hard real-time message may overlap with a slot reserved for hard real-time transmission. To accommodate this possibility, the duration of the time slot includes the maximum waiting time, which in the worst-case is the duration of the longest possible CAN message, prior to hard real-time transmission. Using this increased slot length a higher priority message will never miss a deadline due to a transmission of a lower priority message. The structure of a time slot in COSMIC incorporating this waiting time is

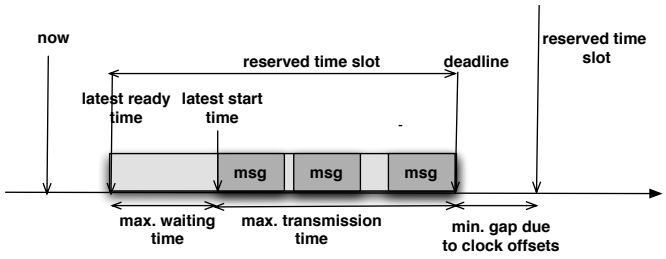


Fig. 2.1: Structure of a time slot in COSMIC

shown in Figure 2.1. In COSMIC, non hard real-time messages may be transmitted at any time, but they may not use the priority reserved for hard real-time messages, thus, hard real-time messages are guaranteed to win bus arbitration.

The use of time to trigger hard real-time transmissions in reserved slots integrates a time-triggered approach with the CAN-Bus, similar to that of TT-CAN (Fuhrer et al. 2000). Unlike TT-CAN, COSMIC augments the use of time with message prioritisation to enforce hard real-time communication guarantees. In COSMIC, time is used to separate hard real-time messages with the same priority.

The priority-based arbitration mechanism is exploited to also schedule soft real-time (SRT) and non real-time (NRT) messages. SRT messages are scheduled according to an Earliest Deadline First (EDF) scheme while NRT messages use fixed lower priorities. The relationship between the priorities of hard real-time, soft real-time and non real-time transmissions are expressed by: $P_{HRT} < P_{SRT} < P_{NRT}$, where for CAN-Bus arbitration, a lower numeric value represents a higher priority. To reflect the EDF scheme of SRT messages the priority assigned reflects the deadline order of message transmissions. The priority of SRT messages is dynamically increased as the deadline approaches, to ensure that the priority of the SRT message will reach its highest value when the transmissions deadline is reached.

2.2.3 Timely Computing Base (TCB)

The assumptions on which the time-triggered and event-triggered protocols rely, for example, global time and static scheduling, imply small-scale static networks, e.g., a field-bus architecture. Achieving timeliness properties in dynamic environments of uncertain synchrony is addressed by the Timely Computing Base (TCB) (Veríssimo & Casimiro 2002), also described as a timeliness wormhole (Martins et al. 2005), where the timeliness guarantees of applications are dependent on the availability of a wormhole or a portion of the system which is timely, e.g., the TCB.

The architecture of the TCB is illustrated in Figure 2.2. A TCB system is composed of a number of

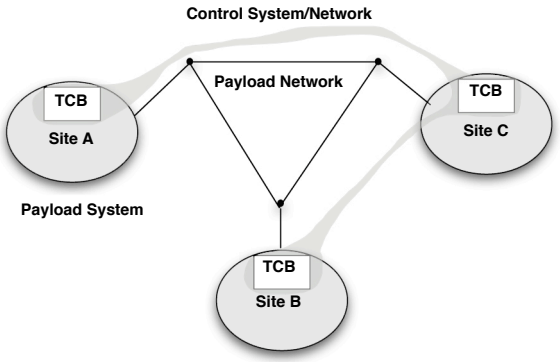


Fig. 2.2: A distributed TCB

participants or processes exchanging messages between several sites or hosts of the same system, with the sites interconnected by a communication network, the *payload* network. There are no synchronous assumptions about the payload network. For example, if bounds exist on processing or transmission latency their magnitude may be uncertain or unknown, and local clocks might not exist or might not have a bounded rate of drift towards real-time. The TCB architecture also encompasses a control system (the wormhole), which comprises the TCB modules, interconnected by a transmission medium called a *control channel*. It is assumed that the control system of the TCB displays the following synchronous properties:

1. Processing delays are known and bounded.
2. Message delivery delays are known and bounded.
3. The drift rate of local clocks is known and bounded.

The availability of a TCB system has the potential to reduce the impact of the varying synchrony of the payload system on applications, e.g., by detecting the failure of timeliness properties in the payload system and notifying changes in the assumed synchrony to applications. An application using the TCB has an assumption about the timeliness properties that the environment can guarantee. The relationship between the assumed timeliness properties and the guarantees provided by the environment is captured by the coverage. In a system with uncertain synchrony the assumed coverage of a timeliness property will not remain constant. If the environment starts to degrade, the probability of timing failures starts to increase and the coverage of the timeliness assumptions starts to decrease.

For the TCB system to be effective, i.e., to detect the violation of timeliness properties, the TCB system must be constructed such that the payload network does not impair the ability of the TCB control system to provide timeliness guarantees. In currently available TCBs for both the wired and wireless domain the synchronous control channel is based on a network that is physically different from the payload channel and is used exclusively to connect TCBs. For example, the operation of a wireless TCB (Martins et al. 2005), depends on the isolation of the control channel from the payload channel by using a dual network architecture with two non-overlapping IEEE 802.11b wireless networks.

In the TCB model it is assumed that all properties, including timeliness properties, are adaptable to maintain coverage close to the assumed values irrespective of the current dynamics of the network. The TCB model, identifies timing failures in the payload network and provides a coverage service to applications which involves adapting the timeliness properties on which the application relies, such that the application operates in a degraded context but the assumed coverage has not changed.

The assumption that timeliness properties are adaptable is not suitable to all applications. A new application class, the **time-elastic ($T\epsilon$)** class, was introduced to describe applications for which it is more beneficial to execute in a degraded mode, e.g. increasing latency for operations, than to stop or, more critically to have unexpected failures due a change in the assumed coverage (Veríssimo & Casimiro 2002). The adaptation of timeliness properties is not suitable to hard real-time applications where timeliness properties must be satisfied without variability.

2.2.4 Analysis

Time-triggered protocols assume a typically small-scale synchronous network, where the number of participants are known, share a global time base and are allocated a slot. This approach guarantees the timeliness properties required for hard real-time communication. However, the assumed synchronous properties, i.e., clocks are synchronised to a global time base and message delivery and processing latency is known and bounded, restricts the environment in which the TT protocol is applicable. The dynamics of a wireless network negate the assumptions of the TT protocols.

The event triggered approach does not assume any degree of synchrony, e.g., an asynchronous interrupt mechanism is used to signal the occurrence of an event. The CAN-Bus, is an example of an event-triggered system. By exploiting properties of the CAN-Bus, i.e., the use of message identifiers to resolve bus arbitration, this event-triggered system can achieve the same synchronous properties as the time-triggered approach (Fuhrer et al. 2000, Kaiser et al. 2005). To achieve these properties in the CAN-Bus requires restrictive assumptions about the network infrastructure. As with the time-triggered approach, all participants must have a global notion of time and remain closely

synchronised. In COSMIC static scheduling, performed once offline, is used to reserve slots for hard real-time messages. In a dynamic wireless network, typified by dynamic host mobility and connectivity, maintaining clock synchronisation is not trivial. Furthermore, the number of participants and applied load is not known and is potentially time-varying, negating the applicability of static schedules to maintain hard real-time communication guarantees. Message prioritisation in COSMIC is closely coupled to the CAN-Bus infrastructure. The availability of message prioritisation is not transferable to a wireless domain, where the assumptions about the infrastructure do not apply.

The TCB model relies on some simplifying assumptions about the environment, e.g., the control channel of the TCB is timely and that synchronous properties, such as, known bounds on processing and message delivery delays are achievable and maintained. Given the challenging characteristics of wireless mobile networks, and particularly the dynamics of the ad hoc domain, it is not clear that these synchrony properties that are essential for the operation of the TCB model can be assumed. For example, in the implementation of a wireless TCB (Martins et al. 2005), the synchronous properties of the control channel are maintained by construction using assumptions about the infrastructure of the ad hoc network, as otherwise the properties would not be applicable.

Even if the necessary synchronous properties of the control channel are achievable, the contribution of the TCB model to achieving hard real-time communication guarantees is limited. The TCB model provides time-bound adaptation to maintain the coverage of timeliness properties regardless of the dynamics of the current operational environment. However, applications that benefit from time-bound adaptation are by definition not hard real-time. In a wireless network, the synchronous assumptions of the TCB model may not be achievable, and furthermore, if the properties are achieved, the adaptation of time bounds is not suitable for hard real-time communication.

2.3 Timeliness in Wireless Networks

Wireless communication is unreliable for message transmission due to its susceptibility to a variety of transmission impediments such as path loss, interference and blockage (Murthy & Manoj 2004). These factors restrict the range, data rate and reliability of wireless transmission. Furthermore, the extent to which these factors affect transmissions depend on external conditions, such as, the dynamics of the environment or the mobility of the transmitter and receiver, and thus, cannot be determined in advance. Due to the characteristics of the wireless medium, medium-access latency, and thus, transmission latency using the medium cannot be guaranteed.

In an unreliable medium, such as wireless, providing real-time guarantees at the network layer

using routing and queuing techniques, is not sufficient (Ergen et al. 2004), e.g., the timeliness of real-time routing protocols relies on a guarantee of deterministic transmissions using the medium. Thus, the provision of timeliness guarantees must also be addressed at the medium-access control layer.

In this section, existing approaches to providing timely medium-access control and real-time routing latency to achieve real-time communication in a wireless network are discussed.

2.3.1 Medium-Access Control (MAC) Protocols

There are two main approaches to medium-access control in wired networks: *schedule-based access* and *contention-based access*. Hosts using scheduled medium-access negotiate a schedule amongst themselves to avoid colliding transmissions, e.g., the time-triggered protocol (TTP). In a contention-based approach a host with a message to transmit competes with all other hosts to gain access to the wireless medium, e.g., the CAN-bus.

The latency to transmit using a wireless medium starts with the request to transmit and finishes when the transmission has been sent on the wireless interface. In a scheduled approach, the transmission latency encompasses the time to transmit the message at the scheduled time. In a contention-based approach, the transmission latency incorporates a transmission request, a non-deterministic period during which contention to gain access to the wireless medium occurs and, finally, the transmission using the medium. The greater the contention prior to transmission the longer the latency to transmit.

2.3.2 Wireless Time-Triggered Control Protocols

The fusion of fieldbus networks with wireless technologies, for example in mobile robotics, is a recent field of research (Huber & Elmenreich 2004). The Wireless Time-Triggered Control protocol (WTT), (Huber & Elmenreich 2004), approach is to provide real-time communication amongst several, possibly mobile TTP/A (Eberle et al. 2001) clusters, as shown in Figure 2.3.

The WTT protocol is controlled by a single dedicated master host. The master is responsible for establishing the common time base between all slave hosts within the cluster and for controlling the communication among all slave hosts. TDMA is used for bus arbitration. Each slave host has a unique 16-bit identifier, with the first 8 bits used as the slave's address and the second 8 bits as a reference to a subcomponent of the addressed slave. The protocol uses round-based communication with each round consisting of the transmission of one or more frames where each frame is a sequence of bytes transmitted from a host, as shown in Figure 2.4. An inter-frame gap (IFG) occurs between any two frames and is a period without communication. The duration of the IFG is a value that

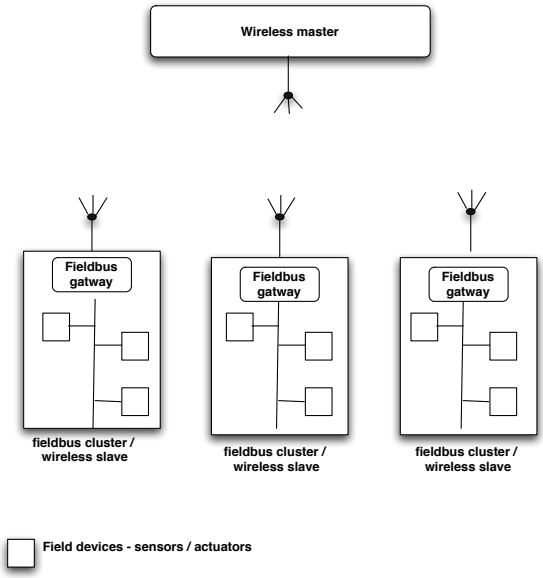


Fig. 2.3: Fieldbus clusters interconnected by wireless communication

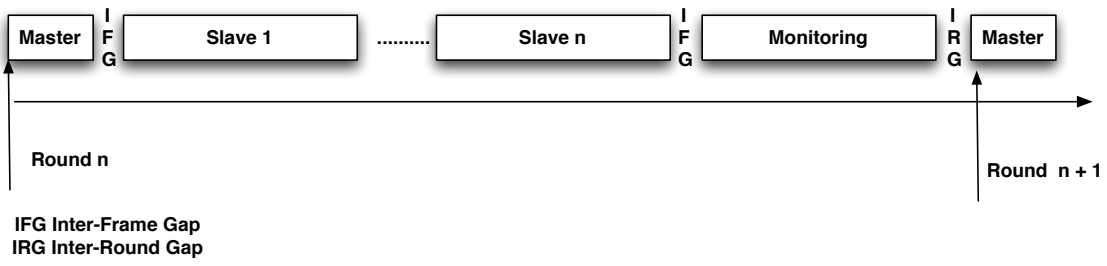


Fig. 2.4: Typical communication round in WTT protocol

depends on the clock synchronisation achieved. Rounds are separated by an inter-round gap (IRG). The structure and duration of each round is static and defined a priori. The master selects the active round by transmitting a round identifier at the start of each new round with the implication that the set of communication rounds must be known to all hosts in the system. There are three different frame types available:

Master frame This frame is transmitted by the master host. The first byte of the frame indicates the start of a new round and is the global synchronisation point for all slaves.

Slave frame Each slave may own one or more time slots for data transmissions.

Monitoring frame The monitoring frame is not statically assigned to a particular host. Depending on the monitoring request and the request type, the frame is either transmitted by the master or one of the slaves.

The master provides a synchronised global time base, represented by 8 bytes, to all slaves. A time-triggered system using TDMA-based bus arbitration means bus access is controlled by the progression of time. Timing violations may lead to collisions on the shared communication medium. In the WTT protocol each host uses a local time confidence value to estimate the maximum duration that can elapse between synchronisations by a slave with the master without violating the medium-access scheme. A slave decrements its confidence value in each round where it fails to synchronise. To ensure a host does not violate the communication scheme a host is only allowed to transmit if the confidence value is greater than zero. The initial time confidence value determines the tolerated number of rounds without synchronisation before the host must stop communication using the shared medium.

Each frame in a round contains the initial and actual confidence value of the sender. The master has the highest confidence. The reception of the first byte of the master frame is a global synchronisation point for all the slaves. Any host with a high confidence value can take on the role of a secondary master host for subsets of hosts to keep them synchronised. This feature of the protocol caters for mobile hosts where connectivity with the master may be time-varying. Transient failures of the master are tolerated if the down time of the master, i.e., the time during which no synchronisation transmissions are possible from the master, is less than the synchronisation interval which is equal to the round length of the active round.

Analysis

The real-time guarantees provided by the WTT protocol are based on the assumption that a global time base is achievable using clock synchronisation provided by prioritised transmissions from the

master. The assumption on which clock synchronisation relies is that the transmissions by the master have the highest priority and are never delayed due to contention for the medium. In a dynamic ad hoc network transmissions by hosts, including the master, may be delayed for a non-deterministic period due to contention for the wireless medium. Avoiding this inherent non-determinism is difficult and requires an agreement protocol for medium-access amongst the hosts. Thus, the dynamics of an ad hoc wireless network, impact the achievable clock synchronisation in the WTT protocol.

The WTT protocol provides real-time guarantees only when a host is synchronised with the master. However, given the dynamics of an ad hoc wireless network, the duration within which synchronisation between master and slave is achievable may be very limited. No guarantees on medium-access are provided to a slave when synchrony with the master is not within a known bound. In the non-deterministic period when a slave is not synchronised with the master the timeliness properties required by hard real-time communication are not guaranteed.

The timeliness properties of the WTT protocol assume a static network where a global time base and a static schedule of transmissions from all participants is achievable. The dynamics of an ad hoc network negate the static assumptions on which the WTT protocol relies. Thus, the real-time guarantees provided by the WTT protocol, although desirable for hard real-time communication, are not achievable in a dynamic network.

2.3.3 Medium-Access Control in Infrastructure-Based Wireless Networks

In this section, the extent to which real-time guarantees are provided in infrastructure-based networks with higher dynamicity and greater scale than assumed in the WTT protocol is discussed.

IEEE 802.11 Point Coordination Function

The Point Coordination Function (PCF) of IEEE 802.11 uses a master/slave approach to coordinating access to the wireless medium (Visser & Zarki 1995). The master is called a point coordinator (PC) or access point (AP). The functionality of the PCF relies on the AP to perform polling, enabling polled hosts to transmit without contending for the wireless medium. All hosts under the control of the same AP are called a basic service set (BSS). The method of maintaining polling tables and determining the sequence of polling is performed by the implementor of the access point (Crow et al. 1997).

The IEEE 802.11 Distributed Coordination Function (DCF), as described in section 2.3.4, and the PCF co-exist by dividing access to the medium into two distinct periods: the contention free period (CFP) controlled by the AP and the contention period (CP) where mobile hosts contend with each other using the DCF to transmit, as shown in Figure 2.5. The CFP repetition interval is used to

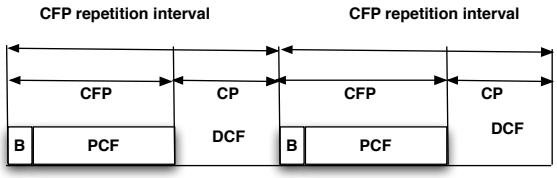


Fig. 2.5: Coexistence of the DCF and PCF

determine the frequency with which the PCF occurs. Within a repetition interval a portion of the time is allocated for contention-free traffic and the remainder is provided for contention-based traffic. The duration of the repetition interval allotted to CFP transmissions is determined by the access point and may change depending on the current network traffic. For example, if CFP traffic is very light the access point may shorten the CFP providing more time for transmissions in the DCF. The CFP repetition interval is initiated by a beacon transmission from the access point.

At the nominal start of the CFP the AP senses the medium. If the medium remains idle for a PCF Inter-Frame Space (PIFS) interval, the AP transmits a beacon frame to initiate the CFP. The AP waits a Short Inter-Frame Space (SIFS) interval following the transmission of the beacon frame to start CF transmission before sending a CF_POLL, DATA or DATA + CF_POLL.

The AP may immediately terminate the CFP by transmitting a CF_END message. A host that receives a poll waits a SIFS idle period before replying with a CF_ACK or DATA + CF_ACK. If the AP receives DATA + CF_ACK from a host the AP can transmit a DATA + CF_ACK + CF_POLL to a different host where the CF_ACK portion of the message is used to acknowledge reception of the previous data message. If the AP transmits a CF_POLL message and the destination has no data to transmit, the destination replies with a NULL message transmission. If the AP fails to receive an ACK for a transmitted data frame, the AP waits a PIFS interval and continues transmitting to the another host in the BSS.

After receiving a poll from the AP a host may choose to transmit to another host in the BSS. When the destination host receives the message a CF_ACK is returned to the source host, and the AP waits a PIFS interval following the ACK message before transmitting any further messages.

The IEEE 802.11 PCF provides guaranteed wireless access to the set of hosts known to the AP, i.e., on the current polling list, within a CFP repetition interval. The AP drops a host from the polling list if the host does not transmit or receive any data for k consecutive polls in the CFP interval. The value of k has an effect on the timeliness properties available to hosts. For example, with a large value

of k, more hosts are polled in the CFP increasing the length of the CFP and the interactions with the AP, thus, reducing the data transmissions in the CFP. The variability in the length of the CFP means that a host with data to transmit may not transmit at the same offset from the start of the CFP repetition interval with the implication that some transmissions, e.g., periodic transmissions, are not suitable to this polling schedule, as the host may not be polled when the data is available. The variability in the CFP interval also means there is a non-deterministic interval between polls from the AP. The IEEE 802.11 PCF provides guaranteed wireless access to a set of hosts, but, the timeliness of the access is not guaranteed.

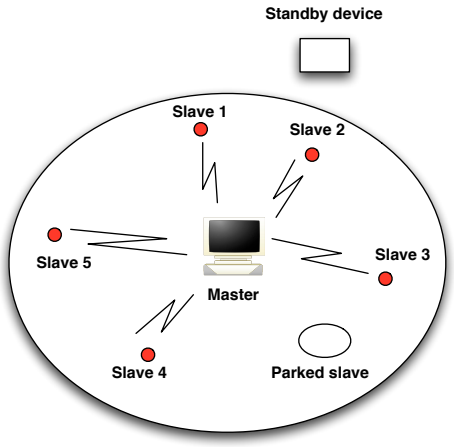


Fig. 2.6: A sample piconet

Bluetooth

Bluetooth devices operate in the ISM frequency band between 2.4GHz and 2.480GHz. The frequency band is divided into 79 1 MHz wide channels, with frequency hopping used to avoid interference using these channels. Bluetooth communication takes place by ad hoc creation of a network called a piconet, where the initiator of the piconet is the master and all other hosts are slaves of the piconet, as shown in Figure 2.6. A piconet can have up to seven active slaves at any instant. For the purpose of identification each active slave of the piconet is assigned a locally unique active member address.

Initially all devices are in standby mode until paged by the master node, moving the device to a connected state. A connected device can participate in data transmission. In the connected state, the clock (of the device) and the address of the master determine the frequency hopping sequence.

The wireless communication channel is divided into time slots of 625μs in length. A time division

duplex (TDD) scheme is used where the master and slaves alternate transmissions, with the master transmitting in even-numbered time slots only and the slaves in odd-numbered slots. A slave is only allowed to transmit if it has previously received a poll from the master. A single message can be transmitted in a slot. A typical message consists of an access code, a header and payload. All messages exchanged on the wireless medium are identified by the master's identity. A message is accepted by a recipient only if the access code matches the access code corresponding to the piconet master. This also assists in conflict resolution when there are two piconets operating on the same frequency.

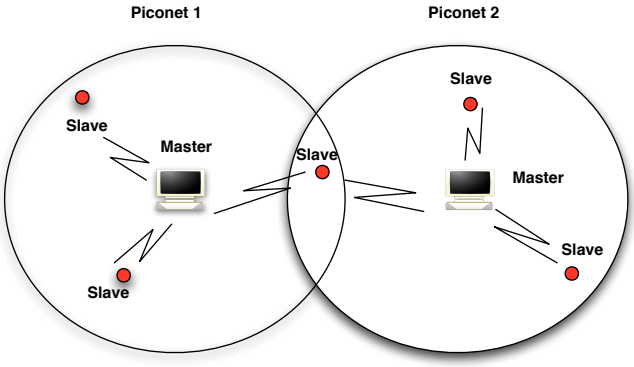


Fig. 2.7: A sample scatternet

Piconets may overlap both spatially and temporally. However, each piconet is characterised by a known master and hence each piconet hops independently with a frequency hopping sequence determined by its respective master. A Bluetooth device may participate in two or more overlapping piconets using time-sharing. To participate on the proper channel, the device must use the appropriate master device address and clock offset. A Bluetooth unit can act as a slave in many piconets but may be the master in at most one piconet. A collection of two or more overlapping piconets is called a scatternet, as shown in Figure 2.7.

Using a master/slave approach, Bluetooth provides guaranteed wireless access to the set of slaves currently known to the master. The duration of wireless access per host is fixed and the polling sequence known, thus, deterministic wireless access is available to each slave. The upper bound on the number of slaves per piconet is limiting especially in a dense network where a host with a requirement for hard real-time communication guarantees may not be a member of any piconets. The ad hoc creation of a piconet, increases the probability of all hosts being a member of at least one

piconet at the cost of increasing the control overhead to coordinate each piconet. There is a trade-off between the number of piconets required to provide guaranteed wireless access to hosts and the resource utilisation to administer the piconets.

Analysis

Infrastructure-based approaches to medium-access control rely on the coordination between a central coordinator, i.e., an access point or master in IEEE 802.11 PCF or Bluetooth respectively, and polled hosts. Wireless access is guaranteed to the polled host. The assumption under which the guarantee of wireless access relies is that hosts are contactable by, and remain within contact of, the central coordinator. Hosts are guaranteed wireless access when in range of the central coordinator only. This assumption of accessibility is not applicable in a wireless ad hoc network where the dynamics of the network topology and mobility of the hosts means that connectivity with a central coordinator is time-varying.

The infrastructure-based approaches to medium-access control provide guaranteed wireless access to a set of hosts. This set is limited to those hosts known to the central controller. Hosts that are not within the set known to the central controller are not guaranteed access to the medium, for example, competing in the contention-period of IEEE 802.11 DCF to gain wireless access, with no guarantees of wireless access provided.

A feature of the IEEE 802.11 PCF is that the length of the contention-free period may change depending on such factors as the data traffic in the network or the number of participating hosts. A change in the length of the CFP has an impact on the polling interval of the access point and the interval between guaranteed access to the wireless medium for a host. The impact of a change in the polling interval may mean that a host with data to transmit is waiting an undetermined duration prior to gaining access to the medium to transmit. Hard real-time communication requires timely medium-access. The IEEE 802.11 PCF guarantees medium-access to a set of wireless hosts, but the timeliness of the access to these hosts is not guaranteed.

2.3.4 Medium-Access Control in Ad Hoc Wireless Networks

The high dynamicity characteristic of ad hoc networks brings unique problems to medium-access control in this domain. For example, the assumption of an accessible master host co-ordinating medium-access is no longer applicable and competition and contention for the wireless medium is time-varying and unpredictable.

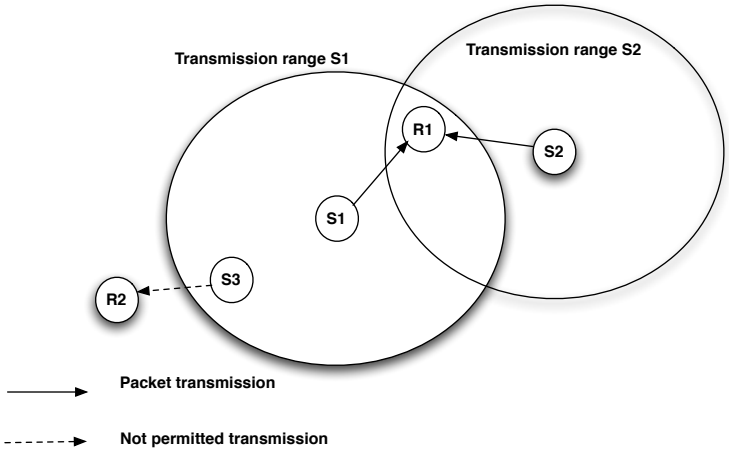


Fig. 2.8: Hidden and exposed terminal problems

Unique to ad hoc wireless networks but with a high impact on MAC layer design are the hidden terminal (Tobagi & Kleinrock 1975) and the exposed terminal (Shukla et al. 2003) problems.

The hidden terminal problem refers to the collision of messages at a receiver due to the simultaneous transmissions of hosts that are not within the transmission range of each other. Message collision occurs when both hosts transmit at the same time. For example in, Figure 2.8, transmitters S1 and S2 although having an overlapping transmission range, are not within direct transmission range of each other and do not know about transmissions from each other. If both S1 and S2 transmit at the same time to R1, a message collision occurs at R1.

The exposed terminal problem occurs when a host is unable to transmit due to the ongoing transmission of a host in the same transmission range. For example in Figure 2.8, the transmitter S3 cannot transmit to the receiver R2 and remains contending for the medium due to the ongoing transmission from S1 to R1.

Attempts have been made to reduce the impact of both the hidden terminal and exposed terminal problems, for example, using MACA (Karn 1992), and MACAW (Bharghavan et al. 1994).

MACA (Multiple Access Collision Avoidance), (Karn 1992), uses additional control messages known as request-to-send (RTS) and clear-to-send (CTS) to gain channel access. Prior to transmitting a data message a host must transmit a RTS message and receive a CTS message from the receiver. Only if the transmitter receives the CTS can the transmitter start a data transmission, otherwise a binary exponential back-off algorithm is started to wait a random interval before retrying. Both the RTS and CTS messages carry the expected duration of the data transmission. All hosts

within range of the sender that hear the RTS defer transmissions for the duration specified in the RTS. All hosts within range of the receiver that hear the CTS defer transmissions for the indicated interval so the receiver can receive the data transmission, eliminating the exposed terminal problem also. However, if a host near the transmitter does not hear the CTS reply, the host waits a short interval and is then free to transmit. Thus, a host that only hears the RTS message may contend with the transmitter of the RTS for wireless access.

In MACA, there is no error recovery performed in the context of lost messages. The responsibility to recover from lost messages, e.g., to retransmit, lies with the transport layer. To push this responsibility to the MAC layer, the MACAW protocol (Bharghavan et al. 1994), uses an additional control message, ACK, sent from the receiver to acknowledge the reception of a data transmission. If the transmitter does not receive the ACK the data transmission is rescheduled for transmission after a calculated timeout. Thus, error recovery in MACAW is much faster than in MACA.

Removing both the hidden terminal and exposed terminal problems does not provide real-time guarantees at the MAC layer, but does reduce the occurrence of one potential source of non-determinism for medium-access and therefore message transmission latency.

IEEE 802.11 Distributed Coordination Function

The IEEE 802.11 protocol for medium-access control in ad hoc networks, the Distributed Coordination Function (DCF) (IEEE 2005), relies on Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA), to determine whether or not a mobile host may gain access to the wireless medium.

To use collision avoidance, a host with a message to transmit senses the medium and, if it is busy, waits a random amount of time (the back-off interval) prior to attempting the transmission. Each host uses a contention window (CW) and generates a random number between zero and CW as the backoff interval. A host freezes its countdown if a transmission from a different host is initiated and restarts the countdown when this transmission has completed. Only when the backoff counter reaches zero and the medium remains free for a DCF Inter Frame Space (DIFS) interval, may a transmission take place. However, if the medium is still busy at this time, CW is increased, and the countdown commences again.

Using this collision avoidance technique, concurrent transmissions, and thus, message collisions, are reduced. However, collisions may still occur amongst all hosts that finish the backoff countdown at the same time, perceive the medium to be free for the DIFS interval, and proceed to transmit. In addition, the backoff countdown may be deferred a number of times, with each deferral reflecting the length of an ongoing transmission. It is not therefore possible to predict the *actual* maximum backoff

interval that will precede a message transmission.

Using IEEE 802.11 DCF, a bounded number of retransmissions are attempted for each colliding message with each retransmission subject to the collision avoidance mechanism outlined previously. In non real-time communication, retransmissions are beneficial as the probability of *eventual* transmission is increased. However, in hard real-time communication, where transmission at a known deadline or within a known period, is critical, a retransmission, i.e., a delayed transmission, may have adverse implications, e.g., out of date information may be worse than no information at all. Retransmissions compete for wireless medium-access, increase the probability of message collisions and the potential for unpredictable backoff intervals.

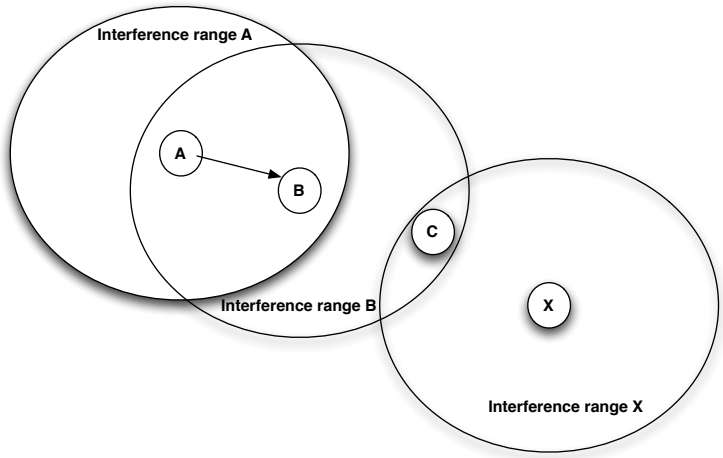


Fig. 2.9: Hidden terminal in RTS-CTS exchange

The IEEE 802.11 DCF uses the RTS-CTS exchange to reduce the occurrence of the hidden-terminal problem. The interference range of a host is the area within which a host may cause interference to message transmissions. The interference range of a host may exceed the transmission range (Yoo & Kim 2005). For example, in Figure 2.9, host C is within the transmission range of B, receives a CTS message transmitted from B and defers transmission. However, host X is not within the transmission range of B, it will not receive the CTS message from B, and thus, may transmit if the carrier is not busy. Host X is in the interference range of B and will interfere with messages transmitted from B, e.g., with the potential for collisions at C. The unpredictability of medium-access in the presence of hidden terminals still exists when using the RTS-CTS approach.

The IEEE 802.11 DCF does not guarantee timely medium-access latency. Transmission latency

using IEEE 802.11 DCF is non-deterministic and subject to contention for the wireless medium and unpredictable backoff intervals. Hard real-time communication guarantees are therefore not possible using this approach to medium-access control.

Medium-Access Control with Reservation Mechanisms

The objective of the Medium-Access Control with Piggyback Reservation (MACA/PR) protocol, (Lin & Gerla 1999), is to establish real-time connections over a single hop only.

The key components of the MACA/PR architecture are a MAC protocol for the transmission of best-effort data messages and a reservation protocol to accommodate transmissions of real-time data messages.

In MACA/PR the available bandwidth is divided into slots. The slot size depends on the bandwidth requirements of transmissions and may be of varying length. Each host in the network maintains a reservation table (RT) that records all the reserved transmit and receive slots for all hosts within its transmission range.

To transmit a non real-time message a host with a message to send waits for a free slot, and when found waits for an additional random amount of time, prior to sensing the wireless channel. If the channel is free the host transmits a RTS message, for which the receiver responds with a CTS message. On reception of the CTS message the transmitter sends a DATA message to which the receiver replies with an ACK. The RTS and CTS control messages contain the time duration in which the DATA message is to be sent. Hosts that receive the control messages avoid transmitting during the indicated time. If after the random interval the channel is sensed as busy, the host waits for the channel to be idle again and repeats the same process.

To transmit real-time messages the reservation mechanism of MACA/PR is used. A transmitter is assumed to transmit real-time messages at regular intervals, e.g., periodically. The procedure to send the first data message is the same as in the non real-time case. Reservation information for the next real-time message transmission, (e.g., in the next period), is piggy-backed on the current message. On receiving the DATA message the receiver updates its RT with the piggy-backed reservation information and sends an ACK to the transmitter which is confirmation of the reservation included in the previous DATA message. Hosts that hear either the DATA or ACK messages update their RT with the reservation information included and refrain from transmitting when the next real-time message is to be transmitted.

Unlike MACAW, the RTS-CTS exchange is not performed prior to the transmission of subsequent DATA messages. After receiving the ACK the source transmits data at the next scheduled transmission

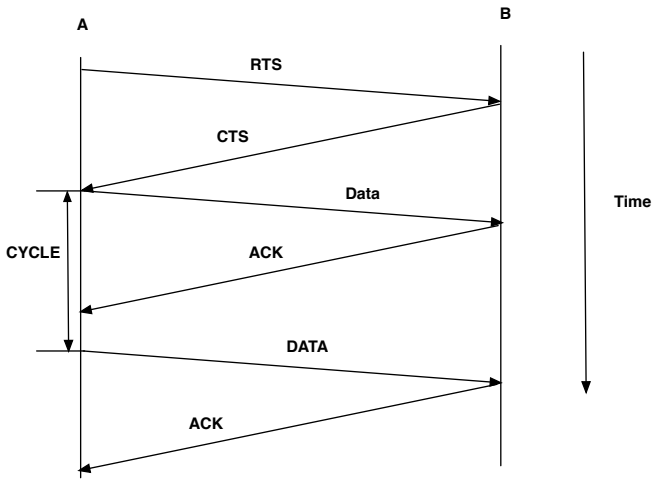


Fig. 2.10: Exchange for real-time transmissions

time. Each DATA message contains reservation information for the next DATA message, thus, the scope of the reservation is for the next DATA message. The exchange of real-time messages (from host A to host B) is illustrated in Figure 2.10.

Real-time messages are transmitted once only. If an ACK message is not received for a DATA message, the source drops the data message. If no ACK messages are received for a number of consecutive DATA messages the reservation is assumed to be lost, and the RTS-CTS exchange is started again to set up reservations if possible. In this case, the source attempts to find a free slot at both the source and receiver to synchronise the transmission of the RTS by the source with the reception of the RTS by the receiver.

To maintain consistent information about free slots at each host, the MACA/PR protocol periodically exchanges reservation tables. The periodic reservation table exchange removes the hidden terminal problem, because a host who would have been a hidden terminal refrains from transmitting in the reserved slots indicated in the received RT. If a reservation is not refreshed for a number of intervals it is dropped. There is no discussion in the MACA/PR protocol of how the mobility and speed of hosts effects the exchange of reservation tables, and thus, the ability to adhere to reservations to support real-time transmissions. In addition, it is assumed that the reservation protocol must be restarted if a transmitter moves, and there is no guarantee that the real-time requirements of transmissions will be satisfied following the move.

A new host joining the MACA/PR protocol remains in a listening mode during which the reser-

vation tables of hosts within transmission range are received and learns about the reservations in the network. Following this learning phase the joining host transitions to being an active host in the protocol. Again, there is no discussion of how the mobility of hosts effects the learning phase of a joining host.

MACA/PR provides timeliness properties for real-time transmissions only when it is guaranteed that all hosts in transmission range of the DATA message share a common schedule of transmissions from each source host. No timeliness guarantees are provided for the first DATA message transmission from a source host, or, for the first DATA message after a reservation has been assumed to be lost. The timeliness properties of hard real-time communication are not satisfied due to these non-deterministic transmission latencies possible in the protocol.

Real-Time Medium-Access Control

The Real-Time Medium-Access Control protocol (RTMAC) (Manoj & Murthy August 2002) provides real-time extensions to the IEEE 802.11 DCF with a medium-access control protocol for best-effort traffic and a reservation protocol for real-time traffic, over a single hop only.[2]

An RTS-CTS-ACK exchange is used prior to best-effort message transmission. Separate control messages, consisting of *ResvRTS*, *ResvCTS* and *ResvACK* messages are used for bandwidth reservation for real-time transmissions. The wait time prior to message transmission for best-effort messages is the IEEE 802.11 DIFS interval. To give higher priority to real-time message transmissions the wait time is reduced to half the DIFS interval.

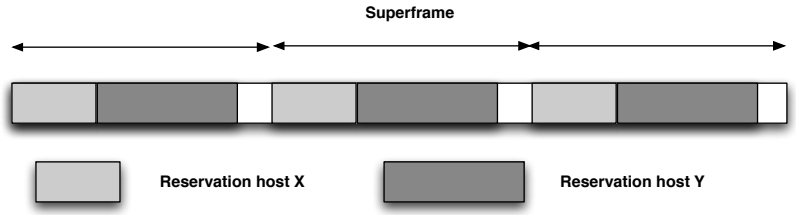


Fig. 2.11: Time slot reservation in RTMAC

Time is divided into superframes. Bandwidth reservations for real-time transmissions are made by a host reserving variable length time slots in superframes within which the transmissions will take place, as illustrated in Figure 2.11. In RTMAC no time synchronisation is assumed. Relative time

[2] An end-to-end route is found by extending RTMAC to use the Dynamic Source Routing (DSR) (Johnson & Maltz 1996).

is used for all reservation purposes. When a host receives this relative time-based information, it converts the relative time to absolute time by adding its current time maintained by its clock. The reservations are entered in to a reservation table that maintains the time intervals during which this host and all hosts within its transmission range require medium-access.

A three-way handshake is performed to make a reservation. For example, assume that a source, host A, wants to send real-time transmissions to a destination, host B. Host A sends a *ResvRTS* message containing the relative start time and end time of the slots to be reserved. Host B checks its reservation table to see if reception in these slots is possible. If these slots are available, host B replies with a *ResvCTS* containing the relative time of the slots to be reserved. All hosts in the transmission range of host B update their reservation tables accordingly. Upon receiving the *ResvCTS*, host A responds by sending a *ResvACK* message, also carrying relative time information regarding the reserved slots. All hosts that receive the *ResvACK* update their reservation tables upon reception of this message, if not already performed following the reception of the *ResvCTS*. Transmission of the *ResvACK* message completes the reservation process. Real-time transmission using these reserved slots follows with the exchange of DATA and ACK messages from host A and host B respectively.

Slot reservations in the current superframe use the same relative slots for transmission in subsequent superframes.

If host B receives a *ResvRTS* message for a slot from host A that has already been reserved for a host within the transmissions range of host B, no *ResvCTS* message is sent and the *ResvRTS* message is discarded. The reason for this is that a reply of either a negative or positive *ResvACK* could cause collisions with reservations already made by host B. Host A times out and starts the same procedure to retry slot reservation again at a later time.

It is not clear how the mobility of hosts effects the protocol. For example, it would appear that a host that moves into the transmission range of host A following the exchange of control messages to establish a reservation, could interfere with data transmissions from host A until reservation tables are available to the new host. There is no indication of how the new host learns of existing reservations apart from timing out while waiting for *ResvCTS* replies to its previously transmitted *ResvRTS* messages. The duration within which a new host learns of existing reservations is non-deterministic.

If host A no longer requires real-time transmissions it releases the resources reserved by broadcasting a *ResvRelRTS* message. All hosts in the transmission range of host A that receive this message update their reservation tables to remove the slots allocated to A. If host B receives this message, it responds by broadcasting a *ResvRelCTS* message. All hosts within the transmission range of host B free the corresponding slots in their reservation tables.

The basis of RTMAC is an extension to the RTS-CTS exchange of IEEE 802.11 DCF to include timing intervals within which real-time transmissions are required. To guarantee the medium is free during the reserved time intervals requires that all hosts in the transmission range of the senders and receivers of real-time transmissions maintain consistent reservation tables. It is unclear how a new host joining the protocol learns of the reservations made, or, of the interval between joining the protocol and requesting reservations. In addition, the effect of host mobility on maintaining consistent reservation tables is not discussed.

The exchange the RTS-CTS messages does not completely remove the hidden terminal problem, as discussed in section 2.3.4. In RTMAC a host in the interference range of either the sender or receiver may not receive the *ResvRTS* or *ResvCTS* messages respectively, and thus, not update its reservation tables. There is no guarantee that a future transmission by this host will not collide with a scheduled transmission, thus, removing the real-time guarantees for the transmitter. The timeliness guarantees for hard real-time communication are not supported in this case.

RTMAC gives priority to real-time messages by using a reduced DIFS interval. More than one real-time message may be ready to transmit to start the reservation process at the same time, wait for the same reduced DIFS interval and attempt to transmit. In this case, the *ResvRTS* messages collide and are subject to the non-deterministic backoff interval of IEEE 802.11 DCF, i.e., the medium-access latency to start the reservation process is non-deterministic.

Wireless Token Ring Protocol (WTRP)

The Wireless Token Ring Protocol (WTRP) (Ergen et al. 2004) is a MAC protocol that provides delay and bandwidth guarantees to applications in ad hoc wireless networks by controlling the timing of transmissions by different hosts over a single hop.

In WTRP the network topology is organised as a ring with transmissions in one direction around the ring only, determined by a special message called the token. A host receives the token frame from its predecessor, transmits data if required, and passes the token to its successor. The token holding time (THT) bounds the duration within which a host may transmit after receiving the token and prior to passing the token to its successor.

Each ring has a unique ring address to distinguish between messages from different rings. The ring address is the MAC address of the host who becomes the ring owner. To ensure a ring owner is always in the ring, the successor of the ring owner claims the ring address and becomes the new ring owner when the old ring owner leaves the ring. The ring owner updates a sequence number in the token every time a valid token is received. If a host receives a token without an updated sequence

number, the host assumes the ring owner is unreachable, and elects itself as the new ring owner.

Successful token transmission relies on implicit acknowledgment, i.e., hearing a transmission from a successive host in the ring. Each host maintains an idle timer that is reset upon the detection of an implicit acknowledgment to a previous token transmitted by them. If the token is lost in the ring, the idle timer expires and a new token is generated by the detecting host who becomes the ring owner.

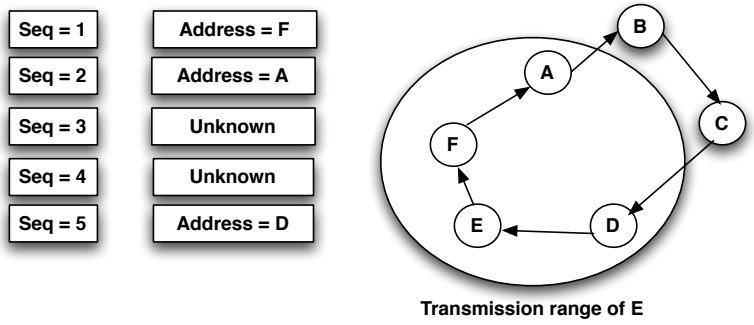


Fig. 2.12: Connectivity maintenance

A connectivity manager resides on each host and monitors transmissions from its ring and from other rings within transmission range to maintain a connectivity table of hosts within transmission range, as shown in Figure 2.12. The connectivity table is used to determine predecessor and successor hosts for token transmission and to start ring recovery. For example, if a successor is unreachable the predecessor uses the connectivity table to determine the next connected host in transmission order and sends a *set_predecessor* message to the next connected host in the ring.

To minimise interference by nearby rings, each ring is allocated a separate wireless channel for transmissions within the ring. A channel allocator, local to each host, uses network topology information to determine the channel for the ring the host is in. The token circulating the ring includes the number of nodes (NoN) in the ring. If the NoN reaches the maximum number of hosts that may transmit on the ring, hosts outside the ring learn that the ring is full. In this case the host switches to the next channel to search for another ring to join. A mobility manager, located on each host, determines when a host should join or leave the ring, i.e., if the host is drifting away from one ring and entering the vicinity of another, thus, solving a mobile hand-off problem. The criteria on which the mobility manager basis its decision are not further elaborated.

The admission controller resident in each ring, limits the number of hosts that can transmit on the medium in a ring. The admission control must maintain the following inequalities in a ring:

1. RESV_MTRT < MAX_MTRT, where MTRT is the maximum token rotation time and RESV_MTRT is the sum of token holding times of each host. Thus, the latency to transmit is guaranteed;

2. NoN < MAX_NoN, where MAX_NoN the maximum number of nodes allowed on the ring. Thus, the number of participants in the ring is bounded.

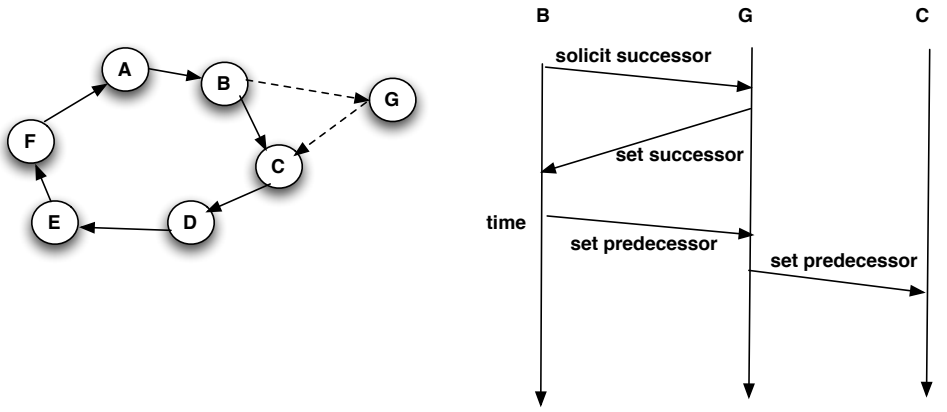


Fig. 2.13: Joining

The interactions to join the ring are illustrated in Figure 2.13. In this scenario, the admission controller, host B, solicits host G to join the ring, and includes the address of successor C in a *solicit_successor* message, and waits a known interval, the response window, divided into equal length slots, within which hosts interested in joining the ring respond. When host G hears the solicitation, it picks a random slot in the response window and transmits a *set_successor* token. At the end of the response window, host B decides on the responses that were received. If host G wins contention, host B passes the *set_predecessor* token to G, and G sends the *set_predecessor* to host C. The joining process concludes.

A host, for example, E, leaves the ring by sending a *set_successor* message to it's predecessor, D. Host D then tries to connect with F by sending a *set_predecessor* token to host F. An implicit acknowledgment will signal if host F is contactable, if not, host D signals the ring is broken.

Using WTRP, admitted hosts in a ring are provided with bounded latency and a guaranteed share of available bandwidth. Thus, real-time communication is possible for admitted hosts. The time to join the ring is non-deterministic and is related to the number of hosts interested in joining the ring at the same time and the outcome of the selection of a joining host by the admission controller. The greater the number of hosts that replied to join the ring within the response window, the lower the

probability of selection for an individual host. The scenario may arise where a host has not joined any ring, and thus, no timeliness properties are guaranteed.

Host mobility is managed by changing the channel to the channel allocated to a specific ring. There is no discussion as to the duration of the change channel procedure which might be of importance as within this period the host is isolated from all channels and is not guaranteed any timeliness properties. Furthermore, a mobile host incurs a non-deterministic join latency when moving between rings which is not suitable for real-time communication.

Time-Bounded Medium-Access Control (TBMAC)

The Time-Bounded Medium-Access Control (TBMAC) protocol (Cunningham & Cahill 2002) is based on TDMA with dynamic but predictable slot allocation, for use in multi-hop ad hoc networks, whose goal is to provide, with high probability, time-bounded access to the wireless medium. The TBMAC protocol uses an atomic broadcast protocol to achieve distributed agreement on slot allocation and employs location information to minimise contention for slots. The TBMAC protocol provides the participants of the protocol with a set of time bounds (with associated probabilities) on gaining access to the wireless medium. To provide these time bounds TBMAC must reduce the probability of message collisions between two or more mobile hosts, detect collisions in bounded time and prevent them from recurring.

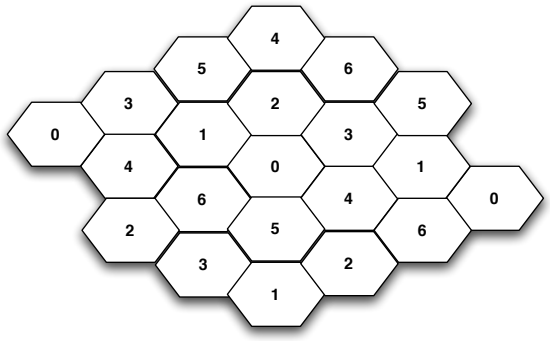


Fig. 2.14: Possible cell and channel allocation

To reduce the probability of collisions in the transmission of hosts, the geographical area occupied by the hosts is statically divided into a number of geographical cells, for example, as hexagons in Figure 2.14. Each cell is numbered and allocated a unique radio channel, thus, reducing the probability of interference from other cells and the possibility of the hidden terminal problem (Tobagi & Kleinrock

1975). Each host must know this layout and the boundaries of the cells. To satisfy this requirement, each host must be able to access location information, for example using GPS (Dana 1997). If a mobile host knows the cell in which it is located it can infer the radio channel to use.

To further reduce the possibility of collisions, access to the wireless medium within a cell is divided into two distinct time periods: the Contention Free Period (CFP) and the Contention Period (CP). Both the CFP and the CP are divided into slots with each period lasting a well-known interval of time. A CFP followed by a CP constitutes a round of the TBMAC protocol. Once a mobile host has been allocated a CFP slot, it has predictable access to the wireless medium. The mobile host can then transmit data in its slot until it either leaves the cell or fails. Mobile hosts that do not have CFP slots allocated to them use dynamic slot allocation to contend with each other for CFP slots in the CP. Dividing access to the wireless medium into these two well-known time periods requires synchronisation amongst all the clocks of the mobile hosts in a cell and furthermore, in the ad hoc network.

To enable communication between adjoining cells, one or more slots in the CFP, called inter-cell slots, are statically allocated. In a similar way to the configuration of geographical cells, the mobile hosts in the ad hoc network are required to reach agreement on these static inter-cell slots before the TBMAC protocol starts executing. Mobile hosts must then dynamically agree on which mobile hosts are allocated each of these inter-cell slots to transmit in.

To reach distributed agreement on the allocation and deallocation of CFP slots, a synchronous atomic broadcast protocol (Cristian 1990) is used.

To explain how the synchronous atomic broadcast protocol works, consider a mobile host with a CFP slot that wishes another CFP slot to be allocated to it. The host creates a request message for a slot, including a sequence number and the current time, and submits the message to the atomic broadcast protocol by broadcasting the message a number of times using its CFP slot. A receiving host checks the sequence number to determine if the message has been received before and if not stores it and also rebroadcasts the message in its allocated slot until the delivery time, i.e., the completion of the atomic broadcast, of the message arrives. When the delivery time of the message arrives, all hosts update their information consistently and allocate the same slot.

There are two important data structures maintained by each mobile host running the TBMAC protocol: the Slot Owners and Slot Bitmap structures. Slot Owners is an array of addresses, CFP in size, which stores the address of the host to which each CFP slot is allocated. The Slot Bitmap contains two bits for each slot in the CFP to represent the four possible states of a slot: Owner, Other, Collision and Available. When a mobile hosts sets a position in Slot Bitmap to Owner it is indicating

to other hosts that it is using the slot, and the identity of the Owner is updated in the Slot Owners array. The consistency of these arrays across all participating hosts is maintained using the atomic broadcast protocol described previously.

To provide time bounds to participating hosts requires guarantees for the management of slots for allocation, deallocation and inter-cell communication. To allocate a slot when a mobile host joins a cell, e.g., powers on or moves to this cell, the host must first learn which CFP slots have been allocated and which are still available. By receiving one message in the CFP correctly, a listening host obtains the number and position of allocated CFP slots. The mobile host then requests a CFP slot to be allocated to it by sending a message in the CP. For example, if a host enters a non-empty cell as the CFP begins, the host must wait until the end of this CFP and the following CP before listening at the beginning of the next CFP. The host then broadcasts a request resulting in an atomic broadcast being transmitted.

To deallocate a slot a mobile host atomically broadcasts a request to deallocate a specific slot. However, a more complex scenario is to deallocate slots from hosts that have failed. Each mobile host, that has been allocated a CFP slot, monitors each of the other allocated CFP slots for correct reception. If a mobile host does not correctly receive a number of messages from a failed mobile host, then it includes this information in its CFP transmissions indicating the failure of the mobile host to use its slot. When other mobile hosts receive this transmission, they incorporate the received information into their CFP transmissions if they also have not correctly received a number of messages from the failed mobile host. After receiving a majority of such indications from hosts in the cell, a host atomically broadcasts a request for the CFP slot to be deallocated. After the delivery of the atomic broadcast message, the CFP slot of the failed host is deallocated by each mobile host in the cell.

The TBMAC protocol does not provide guaranteed medium-access but does provide to a host a time bound and probability of gaining wireless access.

Analysis

Hard real-time communication requires guaranteed medium-access latency for all real-time transmissions. MACA/PR, RTMAC and WTRP do not provide deterministic latency between requesting a real-time transmission and first transmission of a real-time message. In MACA/PR the first data transmission is best-effort and the latency non-deterministic. In RTMAC the time-bound between requesting a transmission and transmitting data is wholly dependent on the existing reservations in the transmission range of the destination host. This latency is non-deterministic and influenced by the

dynamics and density of the hosts in the environment and is sensitive to the traffic generated by these hosts. In WTRP, real-time guarantees are available to a ring member only, however, joining a ring is non-deterministic. The TBMAC protocol does not provide a guarantee of wireless medium-access but does provide a time-bound with an associated probability of gaining wireless access.

Both MACA/PR and RTMAC rely on the exchange of, at a minimum, RTS-CTS-DATA messages to make bandwidth reservations to satisfy the requirements of real-time transmissions. The exchange of RTS-CTS-DATA messages does not eliminate the hidden terminal problem, thus, contention for medium-access may occur during the required reserved times for real-time transmissions, causing unexpected collisions and non-deterministic transmission latency. Both WTRP and TBMAC use different wireless channels or frequencies to remove the hidden terminal problem by design. Thus, both WTRP and TBMAC provide access to the wireless medium for admitted hosts, i.e., hosts that have joined the ring or have been allocated a slot respectively, within known time bounds of MAX_MTRT in WTRP and one round in TBMAC.

2.3.5 Real-Time Routing in Wireless Ad Hoc Networks

The goals of real-time routing in ad hoc networks are two-fold: firstly to discover and select feasible routes with sufficient resources to guarantee the timeliness constraints of the real-time communication and secondly to maintain these routes regardless of the underlying dynamics of the network topology.

Routing to satisfy timeliness constraints has been the subject of much research, e.g., in the multimedia domain (Setton et al. 2005). The routing protocols discussed in this section are exemplars of real-time routing protocols designed to satisfy the timeliness constraints of real-time communication. The protocols included in this section were selected based on two criteria. Firstly, the protocol must exemplify a new approach to achieving timeliness constraints in ad hoc wireless networks compared to previous work and secondly, be representative of widely deployed protocols in its class.

To provide background to the discussion, the provision of real-time communication guarantees in an ad hoc wireless sensor network where support for real-time communication does not have to accommodate the impact of a changing network topology is discussed first. The remainder of this section discusses exemplars of real-time routing in dynamic mobile ad hoc environments.

Throughout this section, the term *delay-constrained routing*, often used with respect to routing, means message transmission with known latency bounds.

Real-Time Routing in Ad Hoc Wireless Sensor Networks

Ad hoc sensor networks are composed of sensor hosts that communicate with each other using wireless communication.

Wireless sensor networks are typically larger than ad hoc networks and their wireless hosts have more severe limitations in memory, power and processing capabilities than their ad hoc network counterparts. Wireless sensor networks, although prone to route failure due to energy constraints, do not suffer from route failure due to host mobility related network dynamics.

SPEED (He et al. 2003) provides a uniform delivery latency across the sensor network so that the end-to-end delay is proportional to the distance between the source and destination. SPEED is a stateless routing protocol, that maintains immediate neighbour information only and thus has no requirement for routing tables. SPEED assumes that all hosts are location-aware. All distributed operations in SPEED are localised, i.e., affect the local area of a host and not the network as a whole.

SPEED uses periodic beaconing between hosts to exchange location information with neighbouring hosts. Each host maintains a neighbour table with each entry for each neighbour containing <NeighbourID, Position, SendToDelay, ExpireTime>, where ExpireTime is used to timeout the entry if it is not refreshed by reception of another beacon and SendToDelay is an estimation of the transmissions delay to the neighbour host. Single hop delay is used as the metric to approximate the load of a host. Delay is measured at the sender who timestamps each message when sending and estimates the round trip single-hop delay for this message when receiving an ACK from the destination.

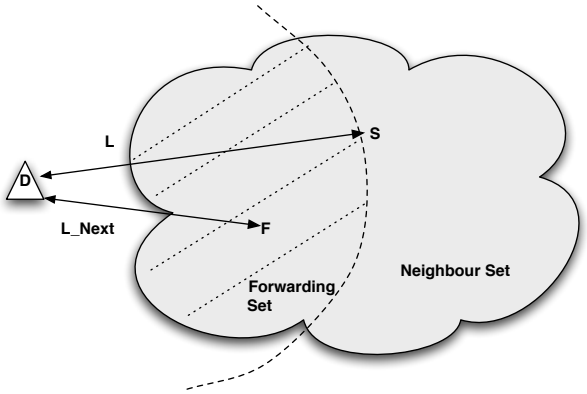


Fig. 2.15: Route discovery in SPEED

The route discovery protocol employed by SPEED is Stateless Non-deterministic Geographic Forwarding (SNGF). SNGF uses geographic information to forward messages only to those hosts that are

closer to the destination. These hosts are the Forwarding Candidate Set. The hosts in the forwarding set (FS) are divided into groups depending on their estimated forwarding delay. Those with the least delay are candidates to be chosen as the forwarding hosts for the communication, as illustrated in Figure 2.15, where L is the distance from the source S to the destination, D, and L_{next} is the distance from a forwarding candidate, F, to the destination. If there are no hosts in the forwarding set or no hosts that satisfy the delay criteria, the message is dropped.

SPEED uses a Neighbourhood Feedback Loop (NFL) to maintain the single-hop estimated delay so that frequent updates of delay estimation are not required. When a message has to be dropped, i.e., there is no path to meet the delay constraints of the communication, the sending rate to the downstream hosts, i.e., hosts closer to the destination, is reduced to avoid congestion, and maintain the single-hop delay. If there are no hosts closer to the destination available to forward a message, SPEED transmits a back-pressure beacon from a downstream host in a congested area of the network to upstream hosts with the estimated delay to downstream hosts set as infinite, which triggers a search for alternative routes that satisfy the delay constraints of the communication. However, the discovery of alternative routes is non-deterministic. In addition, there is no discussion of the latency to detect that a back-pressure beacon should be transmitted.

SPEED provides soft real-time end-to-end delivery latency with a theoretical delay bound reflecting the distance between the source to the destination. The use of average single-hop delay estimates to calculate the required end-to-end delay for a communication of a known distance in the network means that deterministic latency, using worst-case delay bounds as required by hard real-time communication is not possible.

The domain for which SPEED is designed is more static than a wireless ad hoc network where the dynamics of host mobility must be considered in the real-time routing approach. The remainder of this section discusses approaches to real-time routing in the more challenging environment of ad hoc wireless networks.

QoS-Enabled Ad Hoc On-Demand Distance Vector Routing Protocol (E-AODV)

The ad-hoc on demand distance vector (AODV) routing protocol (Perkins et al. 2000*a*) is a widely deployed routing algorithm for ad hoc networks.

In AODV, the source host floods a *RouteRequest* message in the network when a route to some destination is not known. A *RouteRequest* includes the source identifier, the destination identifier, the source sequence number, the destination sequence number, the broadcast identifier and the time to live. The destination sequence number is used to determine the freshness of the route that is accepted

by the source. An intermediate host either forwards the *RouteRequest*, or prepares a *RouteReply* if a valid route to the destination is available. The validity of the route at the intermediate host is determined by comparing the sequence number at the intermediate host with the destination sequence number in the *RouteRequest* message. Intermediate hosts with valid routes to the destination, and the destination itself, reply to the source. When a host receives a *RouteReply* message information about the downstream hosts from which the message was received is stored to route data transmissions to these downstream hosts if required. All intermediate hosts that receive a *RouteReply* update their route tables with the latest destination sequence number.

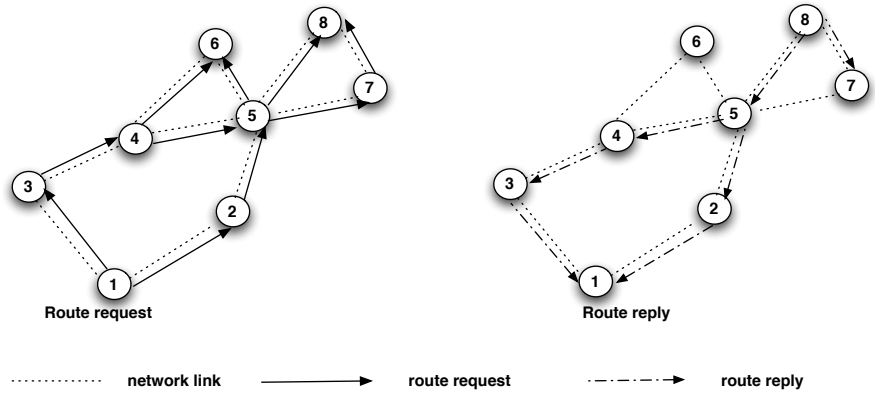


Fig. 2.16: Routing in AODV

Figure 2.16 illustrates an example of routing using AODV from the source, 1, to the destination, 8. The *RouteRequest* propagates to all hosts on the forward route to the destination host. The *RouteReply* is only back-propagated by those hosts who have a valid route to the destination. For example, the intermediate host 6, does not have a route to the destination and does not reply. The lack of a reply from host 7, is interpreted as a failure by host 5, who does not include 7 on a route to the destination. All the intermediate hosts that receive a *RouteReply* update their local routing tables with the latest destination sequence number.

To apply real-time constraints to the AODV protocol, extended as E-AODV (Perkins et al. 2000*b*), extensions are added to the *RouteRequest* messages during the route discovery phase to allow a source to specify the maximum time delay experienced by any route from the source to the destination. Thus, using E-AODV an attempt is made to provide guaranteed transmission delay on routes to a destination. A host that receives a *RouteRequest* with real-time extensions attempts to satisfy the delay-constraints of the *RouteRequest* to either rebroadcast the *RouteRequest* if this host does not

have a route to the destination or forward the *RouteRequest* to the destination.

To discover a route, each host that receives a *RouteRequest* message compares the maximum delay field with the hosts traversal time, i.e., the time for the host to process a message[3]. If the delay value in the *RouteRequest* is less than the host's traversal time, the host discards the *RouteRequest* message. Otherwise the host subtracts the host traversal time from the maximum delay in the *RouteRequest* and either forwards the *RouteRequest* or prepares a *RouteReply* message if this is the destination host, as per AODV.

The delay field in the *RouteReply* originated by the destination host is initially zero. Each intermediate host forwarding the *RouteReply* adds its own host traversal time to the delay field and updates the stored maximum delay field in a routing table with this cumulative value. Thus, the maximum delay field of the *RouteReply* message indicates the current estimate of cumulative delay from the intermediate host forwarding the *RouteReply* to the destination host. The route discovery phase finishes with the reception of a *RouteReply* message reception at the source host.

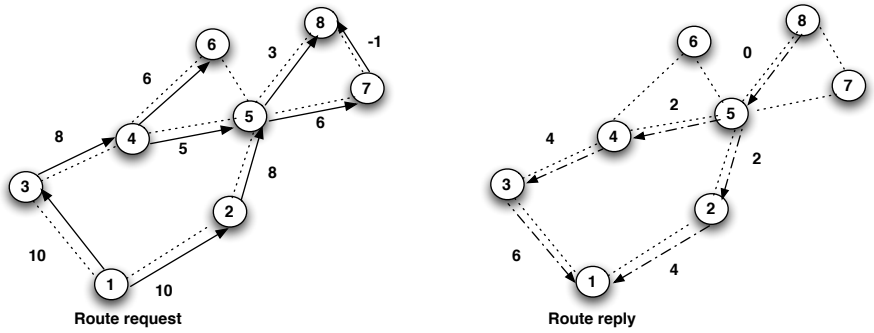


Fig. 2.17: Real-time routing in E-AODV

For example, in Figure 2.17, the transmission delay requested by the source is 10 seconds. Route discovery finds routes to the source that satisfy this delay bound, for example, the host traversal time at host 7 means that no route is available from host 7 to the destination. The back-propagation of replies from the destination increases the delay by the host traversal time with a decision on the route to use for transmissions being made at the source host depending on the received delay values.

Route failures in AODV, and similarly in E-AODV, are not repaired locally, i.e., at the intermediate host where the break occurs. In E-AODV an intermediate host generates an ICMP_QOS_LOST message when there is an increase in the host's traversal time or an increase in the volume of traffic to

[3]It is unclear whether the traversal time is relative to the processing of a RouteRequest or a data message.

this host. The QOS_LOST message is forwarded to all hosts potentially affected by the change in the delay provided on the route. These hosts are all sources to whom a *RouteReply* has been forwarded from this intermediary before.

E-AODV extends a widely used ad hoc routing protocol (AODV) to include the specification of transmission delay constraints for communication within an ad hoc network. No resources are explicitly reserved to maintain the real-time constraints specified in the *RouteRequest*. Thus, the timeliness of the route is affected by the capacity of the network and the current traffic on the routes. There is no guarantee that the delay constraints of a communication will be guaranteed when transmissions using the route are required. Only, soft real-time communication is possible and hard real-time guarantees are not provided.

Ticket-Based QoS Routing (TBR)

The Ticket-based QoS Routing Protocol (TBR)[4](Chen & Nahrstedt 1999) also attempts to find a route that satisfies a specified transmission delay.

The TBR protocol assumes that local state information about all outgoing links from this host, including the delay of a link between two neighbours encompassing the radio propagation delay, the queueing delay and the protocol processing time, obtained using periodic beaconing, is maintained. TBR removes flooding from route discovery and limits the search for a route that satisfies delay constraint to a known and bounded number of possible routes.

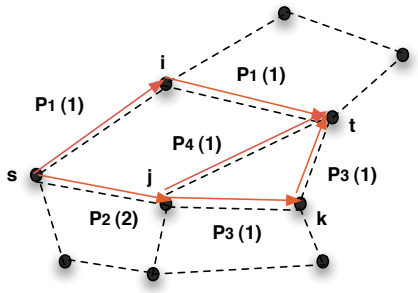


Fig. 2.18: Route discovery in the Ticket-based routing protocol

Discovery of a delay-constrained route in TBR commences by issuing a number of *tickets* at the source. A ticket gives permission to search for one route. The number of tickets issued by the

[4]Sometimes referred to as ticket-based probing.

source relates to the delay state information maintained at the source, and the timeliness required by the delay constraints for a route. For example, a route with strict delay constraints will be harder to discover, thus, more tickets are issued to allow more routes to be searched. The tickets are transmitted in a routing message called a *probe*, as shown in Figure 2.18. Each probe must contain at least one ticket. If a probe contains numerous tickets, it may be split to explore more routes, thus, increasing the probability that a route satisfying the specified delay constraint will be discovered. Local state information at the intermediate host is used to determine if the ticket should be split. For example, in Figure 2.18, the source, s, sends two probes P_1 and P_2, with 1 ticket in P_1 and 2 tickets in P_2, respectively. At host j the probe P_2 is split into two further probes P_3 and P_4, with one ticket each. There are at most three probes at any time and three routes being explored, they are $s \rightarrow i \rightarrow t$, $s \rightarrow j \rightarrow t$ and $s \rightarrow j \rightarrow k \rightarrow t$.

Each probe accumulates the delay of the route traversed so far. A probe is forwarded only when the accumulated delay does not violate the overall delay required for the route from the source to the destination. The fields of a probe routing message include:

- *id* - system wide unique identifier for the delay constrained route;
- s - source host;
- t - destination host;
- D - delay requirement;
- k - sender of the probe p;
- *route* - the route p has traversed so far;
- *delay* - accumulated delay of the route traversed so far;

The fields k, *path* and *delay* are modified as the probe traverses the network. For example, initially *delay* is 0. Following traversal through the link(i.j), *delay(p)* := *delay(p)* + *delay(i,j)*. A ticket is invalidated if an intermediate host is unable to forward the probe to a neighbour that satisfies the delay constraints of the probe.

Route discovery is terminated when all probes with valid or invalidated tickets reach the destination. If only invalidated tickets are received at the destination, a rejection of the route request is transmitted from the destination to the source. If at least one valid ticket is received, the destination accepts the route request. If multiple probes with valid tickets are received at the destination the route with the lowest delay is selected as the primary route. A confirmation message is sent from the

destination to the source and is used to reserve resources along the way. To accommodate route failures both the source and destination use timeouts to prevent waiting indefinitely, e.g., for confirmation of the route or transmissions on the route, respectively.

TBR uses a soft state approach to resource reservation, i.e., the resources are reserved for known time intervals only and are refreshed periodically. In TBR each host in the network maintains a connection table with an entry for every connection passing through the host and for the incoming and outgoing links used by the connection. For example, when a data message is received on an incoming link, the connection table is searched to determine the outgoing link on which the data message should be forwarded. A *refreshing* message is sent from the destination to the source periodically. When the refresh message is received at an intermediate host the timer of the corresponding soft state entry is updated and the message is forwarded to the upstream host.

If a route break is detected by a host a look up of the connection table is performed to find the source host of the connection and a *route-breaking* message is sent upstream from the detecting host to the source. A *resource-releasing* message is sent upstream on the original route to release the resources on the route. When the route-breaking message is received by the source, the route discovery process is restarted to find an alternative route. Using the soft-state approach to resource reservation all downstream hosts from the route break will timeout and release resources for routes that do not receive any refresh updates within a known time bound.

The Ticket-Based Routing protocol discovers routes satisfying delay constraints for soft real-time communication only. The non-determinism of route discovery, e.g., that routes will be discovered or resources will be available, coupled with the unpredictability of rerouting, e.g., that all data transmissions are sent as best-effort messages when rerouting occurs, emphasises that this protocol satisfies soft real-time constraints only. Applications with hard real-time communication constraints are not satisfiable using this protocol.

Trigger-Based Distributed Routing (TBDR)

The Trigger-Based Distributed Routing (TBDR) protocol (De et al. 2002), combines local state information with location information, obtained for example using GPS, to provide routes satisfying specified delay constraints in an ad hoc network.

All hosts in the network maintain local neighbour information by periodic beaconing of location and mobility information to all hosts within their transmission range. A host listens for beacons and maintains a local neighbourhood table with each entry consisting of: received power level, current x and y coordinates of the sender, and the velocity and direction of motion of the sender. In addition to

the neighbourhood information, a host maintains additional information if it participates in a route as the source, the destination or an intermediate host, with a corresponding table called the source table ST_n, the destination table DT_n or an intermediate table IN_n, respectively. These tables are called the *activity-based tables*. At any time a host may maintain some or all of these activity tables for discovered routes with ongoing transmissions. The activity-based tables are soft-state and require refreshing by data transmissions.

Route discovery is started when a source host requires a route to be established to a destination host. The source host first checks whether the location of the destination is known using its local neighbour information. If the location of the host is known, the source uses a selective forwarding approach, i.e., using stored locations to direct the route discovery, to transmit to the destination. If no information about the location of the destination is maintained the source initiates a flooding-based initial route discovery, but limits the forwarding of route discovery messages to only those neighbours with a receive power greater than some threshold power level *(P_{th1})*. The source temporarily reserves sufficient bandwidth for the data transmissions with a lifetime within which an acknowledgment is expected from the destination. A route discovery message includes the following fields:

- *session_id* - system wide unique identifier for the delay constrained route;
- *s_id* - source host;
- *d_id* - destination host;
- *s_loc* - location of the source;
- *dist* - the hop count from the source;
- *Max_BW* - the maximum bandwidth demanded for the route;
- *Max_delay* - maximum acceptable delay for the route;

An intermediate host, upon reception of a discovery message, increases the *dist* field by one and checks whether the bandwidth available at this host satisfies the bandwidth demand for the route *Max_BW*. If the host can reserve sufficient bandwidth and the *dist* field is less than the *Max_delay*, the host temporarily reserves the bandwidth and adds an entry to the IN activity table. If either or both of the *Max_BW* or *Max_delay* constraints cannot be satisfied, the route request is dropped. Otherwise, the updated route request message is forwarded to a known downstream neighbour.

Resource reservations on the discovery route are temporary. The lifetime of the resource reservation depends on the hosts' location. For example, a host close to the destination, determined by comparing

the current *dist* with the *Max_delay* of the route request, expects to receive an acknowledgment for the route from the destination within a short period of time. Failure to receive this acknowledgment within the lifetime of the temporary reservation means the resources are no longer reserved for the route.

If the route discovery message is received at the destination and the destination satisfies the *Max_delay* constraint, after incrementing the *dist* field, the discovery message and the route the message traversed are accepted. Upon accepting a route the destination host builds the DT table and starts a route acknowledgment (ACK) towards the source along the selected route. Upon reception of the ACK all intermediate hosts and the source host update the fields in their respective IN and ST tables, i.e., making the route active and refreshing the reservation information.

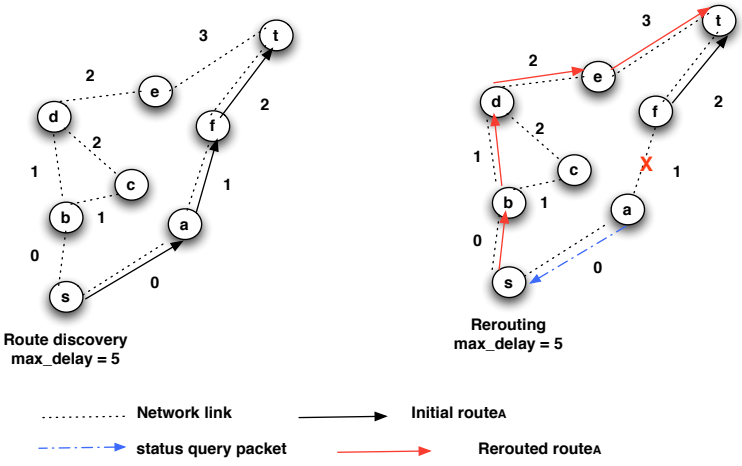


Fig. 2.19: Rerouting in TBDR

If route breaks occur or are perceived as imminent, i.e., due to changes in the receive power levels of a neighbour, a rerouting process is started. Rerouting may be initiated by either the source or an intermediate host. If the receive power level at an intermediate host reduces to a threshold value, (P_{th2}), the intermediate sends a *status_query* to the source who restarts the route discovery process. If new routes are discovered that do not include some of the intermediates on the old route, the resources reserved on the old routes are timed out using the soft state approach. For example, in Figure 2.19, the route break detected at intermediate host, *a*, on a route to the destination, *t*, starts a new route discovery process from the source, *s*, upon reception of the status query message from host *a*. A new route discovery is started from the source which follows the minimum delay route to the destination.

Hosts on the old route to the destination, i.e., a and f, will release resources reserved for the route upon the expiration of a timer for route refresh.

To terminate a route the source purges its corresponding ST table and sends a route deactivation message to the destination using the old route. Upon reception of a route deactivation message each host releases the resources reserved for the route and purges the entries in the IN or DT tables for the route.

TBDR supports communication using routes satisfying specified delay constraints. TBDR does not guarantee that routes or resources will be available when required for a communication. Thus, only soft real-time communication is supported. The trigger-based routing approach uses location information to both limit the search area and the state information maintained at each host, thus, reducing the control overhead maintained and transmitted between hosts. Reducing control overhead is a contribution to routing in general, but particularly real-time routing where the transmission delay of a real-time route is affected by all other (control and data) transmissions.

Predictive Location-Based QoS Routing (PLBQR)

In Predictive Location-Based QoS Routing (PLBQR) (Shah & Nahrstedt 2002) location and movement information is maintained at each participating host with the assumption that future geographic locations can be predicted based on previous locations.

The PLBQR protocol assumes that each host can obtain its location, for example, using GPS. A host in the network propagates its current geographic location and resource information, using a broadcast flooding protocol, to all other hosts in the network. There are two types of updates used by hosts:

Type 1 updates are generated periodically and signal small changes in the location and resource availability of the host;

Type 2 updates are generated when there is a significant change in a host's velocity or direction of movement. Each host calculates an expected location based on its recent history. The host then periodically checks if it has deviated greater than δ from this expected location. The value of δ is large enough to prevent reporting minor perturbations in direction. If the deviation is greater than δ a Type 2 update is generated.

In PLBQR, to establish a route between a source, a, and a destination, b, the source must first predict the geographic location of the destination and all the intermediate hops *at the instant when the first message will reach the respective hosts.* Both location and delay prediction are used to satisfy

this routing requirement. Location prediction is used to determine the geographic location of a host (either an intermediate or a destination) at a particular instant of time t_p in the future when the message will reach the host. Delay prediction is used to estimate the value of t_p, i.e., what time will a message be received at a host given the delay to transmit to that host. The delay used in delay prediction is the end-to-end delay experienced by an update message from host b to a.

As a result of location and resource updates each host has information about the complete topology of the network. Each host maintains two tables containing the state information about the network - the *update table* and *route table*. The update table contains information pertaining to every host in the network, and includes: identifier of the sender of the update, the transmission time of the update message and the geographic co-ordinates, direction of motion and speed of the sender. The update table also maintains a *proximity list* of all hosts lying within some proximity of the host. The proximity is slightly larger than the transmission range of the host to accommodate the movement of hosts into their transmission range, and becoming eligible as next hop neighbours of the host during route computation.

The route table at a host, a, contains information about all active routes with this host as the source. When an update message is received at a source it checks if any routes in the route table are broken or are about to be broken, e.g., by host movement or changes in the resource availability for a route, by the update.

A host starts route discovery to a destination by specifying a route request as a tuple: <estimated route duration, maximum delay, maximum delay jitter>. The estimated route duration is mapped to the minimum requirement for battery power for the intermediate hosts. The maximum delay is mapped to the end-to-end delay observed for updates from the destination to the source and the maximum delay jitter is mapped to the mobility of the intermediate hosts. In response to the request, location prediction is performed on each host in the proximity list to obtain a list of neighbours with the required resources to satisfy the constraints of the request. The source starts a depth-first search for the destination starting at each candidate neighbour to find the candidate routes to the source. At each stage of the depth-first search only neighbours with sufficient resources are considered further. From the resulting candidate routes the geographically shortest route is selected and data messages are forwarded on this route until no more data is available for the route or the route is recomputed in anticipation of a breakage.

Anticipated rerouting, i.e., where either small *type 1* or large *type 2* updates have been received, is initiated from the source when the update messages reach the source. Unanticipated rerouting, e.g., where a route break occurs due to the sudden movement of a host out of range, may also be required

and is detected by the failure to receive update messages from these hosts. In this case the source initiates a route discovery for an alternative route to the destination using the same process as for the initial route discovery.

Resource reservation is not included in PLQBR. The omission of resource reservation from PLBQR means that there is no guarantee that resources will be available for data transmission on a discovered route when required. Due to the unpredictability of resource availability, hard real-time communication guarantees are not supported by this approach.

Analysis

Hard real-time guarantees are not provided by any of the routing protocols discussed in this section. To achieve hard real-time communication relies on the fact that routes are discovered and resources are reserved to satisfy a timeliness constraint and both routes and resources remain available, i.e., able to satisfy the timeliness constraint, for future real-time transmissions using the route when required. The dynamics of an ad hoc network mean that route and resource availability is time-varying. The extent to which the protocols discussed in this section accommodate the changing dynamics of the network reflect the protocols suitability to support any communication in a dynamic network.

Ad hoc networks are typified by host mobility and dynamic connectivity, both of which impact the availability of routes over time and must be reflected in the route discovery and route maintenance processes. E-AODV and TBR either search by flooding or by searching a bounded number of routes in parallel. The only criteria on which a route is selected is the validity of the route in terms of a delay constraint, the stability of the route, e.g., depending on the mobility of the hosts encompassing the route, is not considered. TBDR provides some discrimination in route selection based on received power levels with neighbouring hosts with the routing decision only to select hosts with power levels above a threshold. PLBQR enhances the selection process further by using location prediction and the mobility of the host to decide on hosts that should be included on a real-time route. In a dynamic network where host mobility directly impacts the stability of routes including factors such as location and mobility in the route discovery criteria are advantageous.

The dynamics of an ad hoc environment may mean that route failure occurs at any time. None of the routing protocols discussed provide a deterministic latency for rerouting due to route failures. For example, in TBR the source is notified that a route break has occurred and rerouting is required. Both the latency to back-propagate the route break notification to the source and the rerouting initiated by the source are non-deterministic. There are no guarantees for data transmissions during the rerouting interval and no guarantee that an alternative route will be found. The rerouting process

is similar in TBDR and differs only in the ability to preempt a route break by monitoring changes in the received power levels from neighbours. Rerouting in PLBQR may be anticipated, i.e., predicted using the location information maintained, or unanticipated, i.e., sudden changes in network topology. The detection of the route break and the initiation of the rerouting process is non-deterministic in each case. The dynamics of an ad hoc network mean that the potential for route failure is very high. Given the dynamics of the network it is difficult to guarantee that alternative routes to satisfy timeliness criteria will be discovered at all. However, what is missing from the protocols discussed is a deterministic latency to notify the source of the change in the guarantees provided by the network.

Ad hoc wireless networks have limited resources available. The impact of this is two fold. Firstly, routes must be found with sufficient resources to satisfy a timeliness constraint, and secondly, discovering and maintaining the route must not introduce further competition for the scarce resources. Thus, the transmission of control information, e.g., beacons to maintain neighbour awareness, and the maintenance of state information, e.g., host information for the complete network, must be kept to a minimum. SPEED is stateless but does introduce control overhead by the use of beacon exchange to discover neighbours closer to the destination of the route. TBR, TDR and PLBQR use beacon exchange to discover neighbour connectivity and to provide additional location information or mobility information about the neighbours. The trade-off here is the reduction in bandwidth resources for real-time transmissions against the increased information available for the judicious selection of a route to satisfy real-time routing criteria.

Furthermore, flooding is used in the route discovery process of SPEED and E-AODV. Flooding is resource intensive in terms of the overhead in routing messages and the processing of these messages at each host, potentially hosts that are a great distance from the destination. An attempt to reduce the overhead of flooding is performed by the remaining three protocols discussed. TBR associates the number of parallel routes to search with the real-time constraints of the communication. TBDR and PLBQR resort to flooding if the location of the destination is not available via beaconing. Reducing the overhead of route discovery is important when the competition for resources is high, i.e., there are scarce resources and a potentially high density of hosts. More importantly, reducing the control overhead of the routing protocols increases the availability of resources to satisfy the constraints of real-time communication.

The dynamics of the ad hoc domain introduce non-determinism and unpredictability that must be handled by real-time routing. There is no guarantee that a route will be discovered to satisfy the constraints of the real-time communication and furthermore remain available when required for real-time communication. None of the protocols discussed in this section provide the timeliness guarantees

required for hard real-time communication in an ad hoc wireless network as none of the protocols accommodate the unpredictability introduced by the dynamics of the domain.

2.4 Summary of Related Work

This related work chapter started with a discussion of the timeliness properties provided by various communication models. Following this the provision of timeliness properties in wired networks of different scales and the applicability of the assumptions on which each approach relies when applied to an ad hoc wireless network were discussed. Next medium-access control protocols for wireless networks, both infrastructure-based and ad hoc, and the extent to which the protocols provide the timeliness guarantees needed for hard real-time communication guarantees were presented. This chapter concluded with a discussion of real-time ad hoc routing protocols and the extent to which hard real-time guarantees are supported by the protocols presented.

Providing hard real-time communication guarantees in ad hoc wireless networks is challenging due to the dynamics of the ad hoc environment. The properties of the models and functionality of the protocols contribute to providing hard real-time communication in wireless ad hoc networks to varying extents. For example, the properties of the synchronous model, although desirable, are not applicable in a dynamic network, the guarantees of TBMAC, although time-bounded are probabilistic, and the delay-constrained routes of any of the routing protocols discussed, although satisfying a delay constraint are not guaranteed to be discovered within a deterministic time bound or remain available when required for real-time communication.

This chapter concludes with a wish list of the properties of real-time protocols to guarantee hard real-time communication in ad hoc wireless networks:

- *the clocks of all hosts are closely synchronised and remain synchronised*
- *medium-access latency is guaranteed with sufficient bandwidth to satisfy real-time constraints*
- *route discovery is time-bounded*
- *the impact of network dynamics on discovered routes is minimal*
- *a change in the real-time guarantees of the route is notified to the source within a time bound*
- *prediction is used to reduce the impact of the dynamics of the network*

Chapter 3

The Space-Elastic Model

Any model providing hard real-time communication guarantees in wireless networks must overcome the impact of the characteristics of dynamic wireless networks. As discussed in chapter 2, a dynamic network topology impacts the predictability of medium-access control, the determinism of route discovery and maintenance and the resulting extent to which the timeliness of real-time communication can be guaranteed. In particular, it must be realised that hard real-time communication may not always be possible within the complete network topology. For example, in Figure 3.1, the dynamics of the network, i.e., that host A has moved, link B failed and host C failed, imply that hard real-time communication is, at best, only available within a portion of the wireless network, the actual coverage, identified by the shaded region.

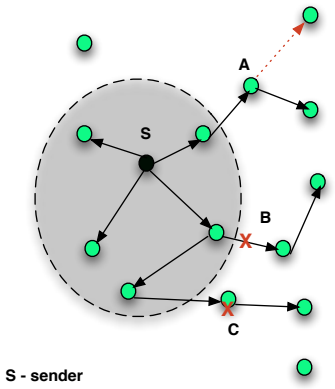


Fig. 3.1: Space where hard real-time communication is guaranteed

In the space-elastic model, hard real-time communication is only guaranteed in the actual coverage. The dynamics of the network may mean that the actual coverage changes but the timeliness guarantees in the remaining actual coverage are not adapted. Furthermore, time-bounded adaptation notification is guaranteed when a change in the actual coverage occurs. The space-elastic model utilises a trade-off between time and space that contrasts with models such as the TCB model (Veríssimo & Casimiro 2002), which uses time-bound adaptation to maintain a probability of communication regardless of the current dynamics of the environment.

The space-elastic model supports a class of real-time applications, i.e., the space-elastic application class, that can operate correctly in an adaptable space (actual coverage). In the remainder of this chapter, the use of the term real-time application implies a (real-time) space-elastic application. We describe a number of applications in the space-elastic class and how they use the properties of the space-elastic model to guarantee hard real-time communication later in this chapter.

A space-elastic application identifies a desired coverage, e.g., a set of geographical location(s) in a wireless network, within which timeliness properties should be guaranteed, as illustrated in Figure 3.2, where messages with real-time constraints are transmitted by host *S*.

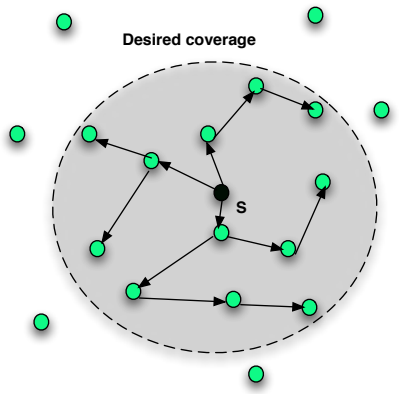


Fig. 3.2: Space-elastic application communication bounds

This desired coverage is application-specific. The dynamics of the wireless network within the desired coverage may mean that real-time communication is not possible within the complete desired coverage. For example, in Figure 3.3, host movement and host and link failure have lead to a network topology where real-time communication is possible only in the actual coverage space illustrated.

The safety and progress of a space-elastic application can still be guaranteed following an adaptation of the actual coverage if notification of a change in network state and the corresponding change to

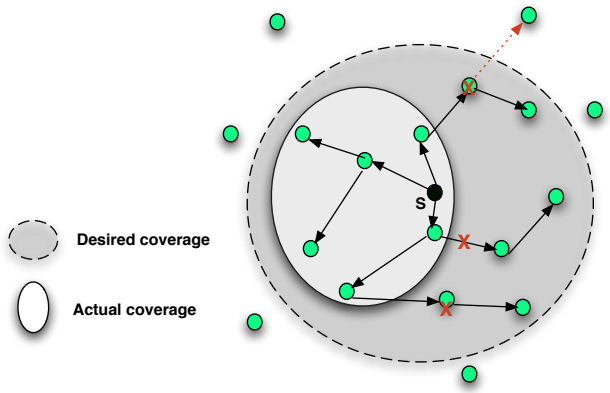


Fig. 3.3: Actual coverage for hard real-time communication

the actual coverage is available to the real-time application within a known time bound. A guarantee of time-bounded adaptation notification allows space-elastic applications to change their behaviour based on the current actual coverage, within a known time bound.

Using a combination of timely communication within an adaptable space and timely adaptation notification, the space-elastic model guarantees progress for space-elastic applications within a dynamic network. The dynamics of the wireless network are unrestricted and may fluctuate between high and low dynamicity. When the network is experiencing low dynamicity the space-elastic model guarantees application progress by supporting timely communication. When the network is experiencing high dynamicity, a space-elastic application can still make progress using adaptation notification of the actual coverage available to form the basis for behavioural adaptation by the application.

3.1 Space-Elastic Model API

The space-elastic model provides a real-time channel abstraction for communication by real-time space-elastic applications.

3.1.1 Real-Time Channel

The properties of a real-time channel are specified by its sender, who will transmit messages over the real-time channel that require the guarantees provided by the channel in a specified desired coverage area.

A real-time channel is created as a member of a real-time channel group, which is a set of real-time channels with some common relationship, for example, the channels are used to transmit the same type of message. A real-time channel group is created when the first member channel is created.

The interface to create a real-time channel is shown in Figure 3.4.

```
channel_id_t create_channel(in channel_id_t channel_group_id,
                            in double start_time,
                            in long period,
                            in coverage_t desired_coverage,
                            in short absolute_relative;
                            in coordinates_t reference_point;
                            in coordinates_t navel;
                            in long msg_latency,
                            in long max_discovery_latency,
                            in AdaptationHandler adaptation_notification) ;
```

Fig. 3.4: Interface to create a real-time channel

At a minimum a space-elastic application must specify both the time and space properties of the channel, i.e., the required delivery latency for the channel, `msg_latency`, and the desired coverage within which the timeliness properties should apply, `desired_coverage`. The delivery latency is specified upon creation and applied to each transmission on the channel. The `start_time` and `period` state from what time and with what frequency transmissions on the channel will occur. Channel creation is time-bounded as specified in the `max_discovery_latency` parameter. The sender is delivered notification of the initial actual coverage at creation and subsequently any adaptations to the actual coverage using the supplied handler `AdaptationHandler`. A sender may create multiple channels over time. The maximum bound on the number of channels created is an artefact of the implementation and is known a priori. A unique channel identifier, based on the input channel attributes, including the `channel_group_id`, is returned to the channel sender from the `create_channel` interface.

A real-time channel is created and used by a single sender to send messages to all receivers listening on the channel to deliver message i at its delivery deadline, $t_{deliver_i}$. Thus, there is a 1:N relationship between a sender and receivers on a real-time channel and a transmission by a sender on a channel may be received by many receivers on the channel. The set of receivers may vary over time, for example, due to the potential mobility of receivers.

Each channel has an associated desired and actual coverage. The *desired coverage* (DC) is specified at channel creation and bounds the area of the wireless network within which the guarantees provided

by the real-time channel are desirable. The sender may be mobile or stationary and the desired coverage may be absolute or relative to the sender, identified in `absolute_relative`, `reference_point` and `navel` parameters in the interface to create the channel.

If the sender is stationary the desired coverage is fixed regardless of whether the desired coverage is absolute or relative to the sender. If the sender is mobile and the desired coverage is relative to the sender, the desired coverage will move with the sender. If the sender is mobile and the desired coverage is absolute, the desired coverage will be anchored to a fixed position. (An application scenario discussed later in the chapter describes the latter case in more detail.) Regardless of the mobility of the sender, a location beyond the bounds of the desired coverage of a real-time channel, which in the mobile scenario must be further qualified by a point in time, will never be covered by the real-time channel.

The *actual coverage* (AC) of a real-time channel is a subset of the desired coverage that includes the sender. If the sender is outside the desired coverage the actual coverage is empty. To be more precise, the actual coverage of a real-time channel is the space that includes the sender and within which there is a guarantee to deliver transmissions on the channel at their delivery deadline.

Given the dynamics of the wireless network within the actual coverage, the actual coverage may change over time, AC(t). A change in the actual coverage is an actual coverage adaptation. To guarantee both the progress of the channel sender and the safety of the real-time application[1], the adaptation of the actual coverage is notified to the sender within a known time bound, the *adaptation notification time*. Adaptation notification is performed on a per-channel basis.

The worst-case adaptation notification time (WC_ADAPT) depends on the extent of the desired coverage and is an artefact of the implementation that can be calculated at channel creation, as shown in chapter 6. The first adaptation notification to the sender following channel creation includes both WC_ADAPT and the actual adaptation notification time based on the current actual coverage for the channel. All subsequent adaptation notifications delivered to the sender include only the current adaptation notification time based on the current actual coverage.

In the case of a reduction of the actual coverage, the current adaptation notification time is guaranteed to be delivered within the last adaptation notification time received by the channel sender, which will be at worst WC_ADAPT. For example, in the scenario depicted in Figure 3.5, the adaptation notification traverses a path in the reduced actual coverage to the sender incurring a reduced adaptation notification time.

An expansion of the actual coverage is somewhat different. In this case, the actual coverage is

[1]The real-time application is composed of the sender and all listening hosts on the channel.

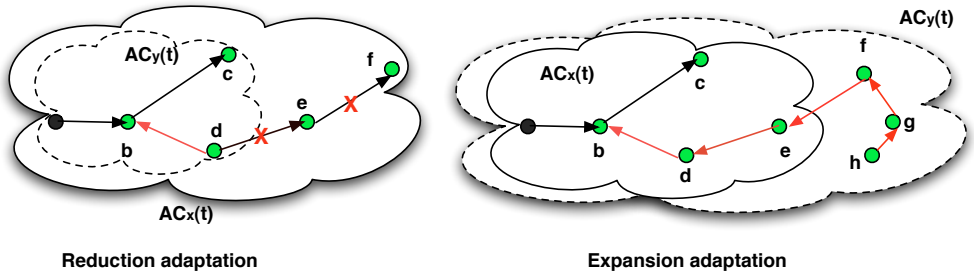


Fig. 3.5: Adaptation notification following actual coverage adaptation

expanding, but the extent of the expansion, i.e., the new space to be included in the actual coverage, is not known until the expansion completes, e.g., until the bounds of the desired coverage are reached. When the expansion completes, a notification of the expansion is back-propagated to the sender on an expanded back-propagation path. Figure 3.5, shows the extended actual coverage, AC_y(t), with the implication that the adaptation notification must traverse a path in the increased actual coverage to the sender.

The notification of a reduction is started when it is detected. The notification of an expansion is started when resource-constrained routes are created which is dependent on the extent of the expansion possible within the desired coverage of the channel.

A notification of an expansion is considered to be of lower priority than the notification of a coverage reduction. The reason for this prioritisation is that the current behaviour of the sender is by definition compatible with maintaining the safety of the application in the coverage prior to the expansion. Therefore, the sender's current behaviour is still compatible with maintaining the safety of the application in the expanded actual coverage following the expansion.[2] In contrast, in the reduction case, the behaviour of the sender may no longer be compatible with maintaining the safety of the application in the reduced coverage. In this case, the sender should be able to adapt its behaviour to maintain the safety of the application as soon as possible.

The detection of an adaptation in the actual coverage is also time bounded and is included in the adaptation notification time calculated, as shown in Figure 3.6, where the timeline starts from when the actual coverage has adapted, which in the expansion case is when the extent of the new actual coverage is known. The latency to detect the adaptation is an artefact of the implementation and can be calculated, as shown in chapter 4, and is the same for both reduction and expansion adaptations.

[2] An alternative model might provide an intermediary notification of the potential to expand when the adaptation is detected and provide a notification of the expansion when resource-constrained routes are available.

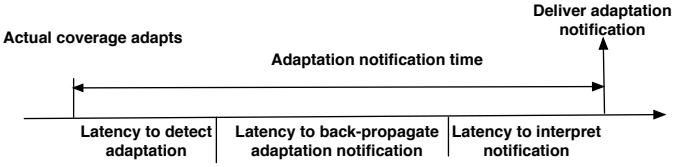


Fig. 3.6: Adaptation notification time bounds

3.1.2 API for Channel Usage

Prior to creating the real-time channel the senders must join the system, using the routine whose interface is shown in Figure 3.7, and provide a callback handler which is invoked to notify the host of the status of the join. A successful join means that this host may create channels and transmit real-time messages using these channels, i.e., the host becomes a channel sender. An unsuccessful join, e.g., there are insufficient resources available to join the system, means that this host may not create channels, and thus, not transmit. The latency to join the system (WC_JOIN) is an artefact of the implementation and can be calculated, as shown in chapter 4. Joining the system and receiving messages on channels of interest are orthogonal, i.e., a host does not have to join the system to receive messages transmitted on a channel of interest.

```
short join(in JoinHandler status) ;
```

Fig. 3.7: Interface to join the group of senders

A property of a created channel is the guaranteed delivery of transmissions on the channel at a calculated delivery deadline, $t_{deliver}$ depending on the specified channel latency. The interface to transmit on a real-time channel is shown in Figure 3.8, where the channel on which to transmit is identified by `channel_id`, which is returned from the initial call to `create_channel`. A transmission on a specific channel is started with the invocation of `transmit`. The measurement of the delivery latency for the transmission starts with the call to `transmit` and finishes when the transmission is delivered at each receiver, i.e., at $t_{deliver}$. Transmissions occur at the periodicity specified in the *period* parameter to `create_channel`.[3]

The real-time channel persists until the sender decides to remove the channel. The `remove_channel` interface, Figure 3.9, provides the voluntary removal of channels. If agreement is reached by other

[3]A channel can be created for transmission once at a specified time by setting the `period` to 0.

```
short transmit(in channel_id_t  channel_id , in string channel_data) ;
```

Fig. 3.8: Interface to transmit on a real-time channel

hosts that due to a period of inactivity on the channel the sender is deemed to have failed, the resources reserved for transmissions on channels will be removed, and thus, the channel is implicitly removed. The details of the agreement protocol for failure suspicion are described in chapter 4. A receiver of a channel that has been removed is not explicitly notified of the removal of the channel but can determine the removal implicitly due to the failure to receive transmissions when expected, using the `start_time` and `period`, on the channel, and is also discussed further in chapter 4.

```
short remove_channel(in channel_id_t channel_id[]) ;
```

Fig. 3.9: Interface to remove real-time channels

3.1.3 Listen on a Channel

So far in this section, the role of channel sender has been discussed. In the remainder of this section the role of channel listeners, the recipients of transmissions on a channel, will be described.

Listening on a channel is how a host expresses an interest in the transmissions on the channel. Using the interface in Figure 3.10, the host specifies the channel group to listen to, `channel_group_id`, and a handler to process transmissions received on that channel, `data_handler`.

```
short listen_on_channel(in channel_id_t channel_group_id, in DataHandler data_handler ) ;
```

Fig. 3.10: Interface to listen on a real-time channel

When a listener listens on a real-time channel group it implicitly listens on all the channels of the group. Thus, a transmission may be received on any of the channels in the real-time channel group and the associated `data_handler` invoked to process transmissions received. The existence of the channel group is largely transparent to the listener. It is the responsibility of the higher layer to include a channel identifier in the metadata of a transmission if the listener should know on which channel in the channel group the transmission was received. There is theoretically no maximum bound

on the number of listeners on a channel. A listener listens on a channel group until the listener either requests to stop listening on the channel, the listener fails, or the channel is removed.

3.1.4 Receive Transmissions on a Channel

To receive transmissions on a channel the host must become *present* in the actual coverage of the real-time channel. To be present in the actual coverage in time to receive a transmission, the host must be listening on the channel and be within the actual coverage a known minimum time, $T_{present}$ before the delivery deadline, $t_{deliver}$, for that transmission on the channel. Only listening hosts that have been present in the actual coverage of a real-time channel for at least $T_{present}$ are guaranteed the properties of the real-time channel. These hosts are the set of receivers for a real-time channel. The possible state transitions for a host wishing to receive on a real-time channel are shown in Figure 3.11.

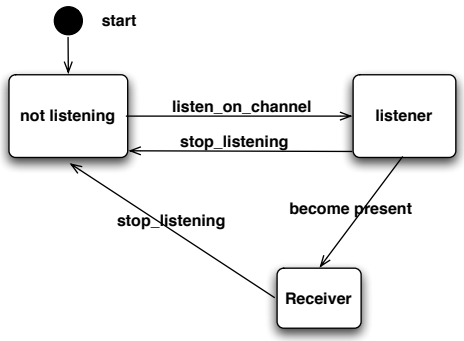


Fig. 3.11: State transitions for a host interested in a channel

A host may be present in the actual coverage of many real-time channels at the same time. As shown in Figure 3.12, the set of receivers associated with the real-time channel a, are all receivers in AC_a, some of which are also in AC_b and/or AC_c. Real-time channel guarantees are satisfied for a receiver regardless of the number of concurrent channels within whose actual coverage the receiver is present at a point in time.

3.1.5 Calculating $T_{present}$ for a Channel

The duration of $T_{present}$ represents the minimum time a receiver must be within the actual coverage prior to the delivery deadline for a transmission on a real-time channel. A host may or may not be

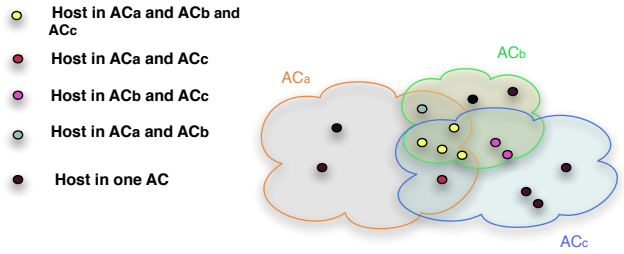


Fig. 3.12: Hosts present in more than one actual coverage at time t

already listening on a channel when entering the actual coverage and the calculation of $T_{present}$ must include both cases. For example, in Figure 3.13*(a)*, the host is already listening on a channel when entering the actual coverage, however, in Figure 3.13*(b)*, the host must start listening, which incurs a known latency, prior to becoming present in the actual coverage. The value of $T_{present}$ must include both the latency for a host to start listening and the time to be present within the actual coverage of a channel.

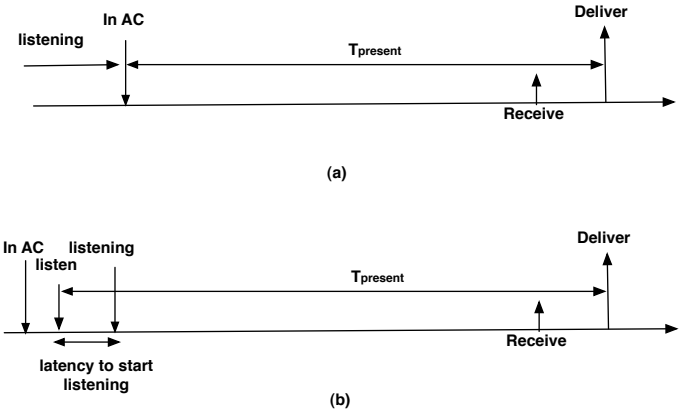


Fig. 3.13: Minimum time to be present

The latency of message transmission on a channel is measured from the call to `transmit` until the delivery deadline for the message is reached and corresponds to $t_{deliver}$. This duration is the delivery latency, `msg_latency`, of the channel as specified when the channel is created.

A message transmitted by some sender is propagated within the delivery latency specified for the transmission in the current actual coverage. The time at which the message is received by listeners

in the actual coverage varies according to distance from the sender, i.e., the closer the listener is to the sender the earlier the message will be received. Regardless of distance, message reception is guaranteed to be within the delivery latency for the transmission. To guarantee a listener receives a message regardless of the listeners location in the actual coverage, the listener must be present in the actual coverage for the delivery latency of the message. Thus, the value of $T_{present}$ includes the delivery latency of the message.

To guarantee a host that is not listening prior to entering the actual coverage will receive a message regardless of the hosts location in the actual coverage, the value of $T_{present}$ must include both the latency to start listening and the delivery latency of the message. This is the worst-case value of $T_{present}$, and thus, is the minimum time a host must become present within the actual coverage before the delivery deadline for a message. For example, in Figure 3.14, the reception of the message (Receive), may occur at any point within $T_{present}$, but will be received by all listeners present in the actual coverage for $T_{present}$. All hosts that receive the message, deliver the message at $t_{deliver}$.

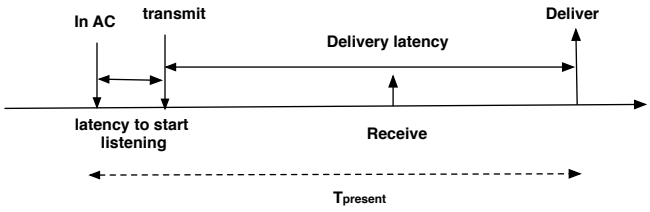


Fig. 3.14: Calculation of $T_{present}$

An adaptation of the actual coverage or the mobility of a receiver may mean that the receiver is not present in the actual coverage for $T_{present}$ prior to $t_{deliver}$. Transmission delivery at $t_{deliver}$ is not guaranteed in this case.

Figure 3.15, presents two scenarios. In scenario *(a)*, host A, is listening on the channel and is present for $T_{present}$, and thus, delivers the message. The movement of host B, into the actual coverage means that host B has not been present for $T_{present}$, and thus, is not guaranteed to deliver the message. In scenario *(b)*, the adaptation of the actual coverage means that host A, previously within the actual coverage, is not within the adapted actual coverage, and thus, not within the actual coverage for $T_{present}$, and not guaranteed to deliver the message.

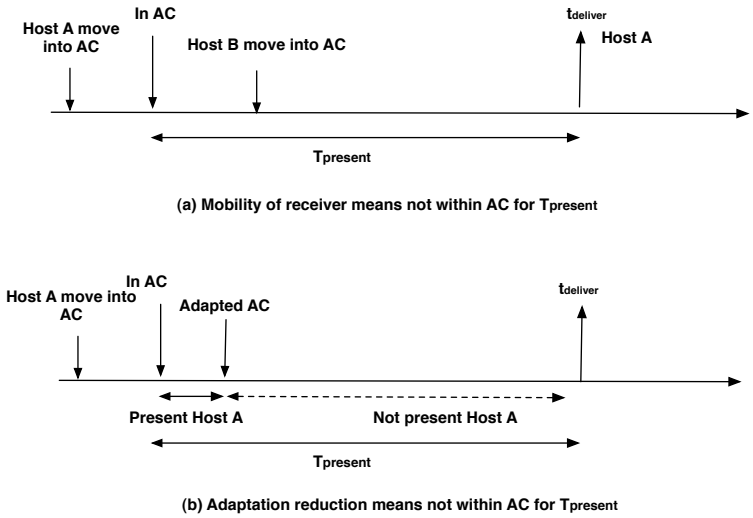


Fig. 3.15: Impact of $T_{present}$ on delivery

3.2 Properties of the Space-Elastic Model

The space-elastic model provides guarantees for both the sender of a channel and the receiving hosts present in the actual coverage. In this section, the properties of the space-elastic model are discussed.

Property 1: Hosts listening on a real-time channel group that are present in the actual coverage of one of its channels for at least $T_{present}$ before $t_{deliver}$ of a message sent on that channel, will deliver the message at the calculated $t_{deliver}$ for the transmission

The mobility of a host or the adaptation of the actual coverage may mean that the host is not within the actual coverage for $T_{present}$ prior to $t_{deliver}$. There is no guarantee that the host will deliver the message at $t_{deliver}$ in this case.

Property 2: Notification of an adaptation of the actual coverage to a channel sender is guaranteed within a calculated time bound

The space-elastic model guarantees that notification of an adaptation to the actual coverage is delivered to the channel sender within a calculated adaptation notification time bound, WC_ADAPT.

The adaptation notification time does not include the latency to react to the notification by the channel sender, i.e., to perform a behavioural adaptation based on the current actual coverage available if required. It is assumed that transitions between behaviours are application specific and

can be performed within deterministic time bounds if required.

Given that the adaptation notification time is known and transition to a new behaviour is performed within a deterministic time bound, the safety of the real-time application and progress of the channel sender is guaranteed when an adaptation occurs.

Property 3: The actual coverage of a real-time channel is always $\leq$ the desired coverage specified for the channel

A real-time channel does not extend beyond the desired coverage of the real-time channel specified when the real-time channel is created. No attempt is made to guarantee the properties of the real-time channel beyond the specified desired coverage. The largest actual coverage of a real-time channel is at most the desired coverage of the channel. An adaptation of the actual coverage to a space within the desired coverage is possible, however, an adaptation of the actual coverage to a space beyond the bounds of the desired coverage is not possible.

3.3 Application Scenarios

The usability of the space-elastic model, e.g., the behavioural adaptation performed by a higher level, is outside the scope of this thesis. However, to show the models usefulness in building real-time applications, two scenarios involving autonomous vehicles are discussed in this section: the *pedestrian crossing at a traffic light* scenario (Bouroche et al. 2006*a*), where the channel sender is stationary, and the *unsignalised junction* scenario (Bouroche et al. 2006*b*) where channel senders are mobile.

Pedestrian Crossing at a Traffic Light

The goal of the scenario is to the guarantee the safe crossing of pedestrians at a traffic light following a request to cross, i.e., so that no pedestrian is killed when crossing at a red light[4]. To guarantee the safety of the pedestrian no vehicles should pass through the traffic light when red.

In this scenario there are two types of entities: traffic lights and vehicles. This scenario encompasses a stationary channel sender, the traffic light, and mobile receivers, which are the vehicles approaching the traffic light within a known proximity. It is assumed that the maximum speed a vehicle may travel is determined by the maximum speed limit of the road. Without loss of generality, only the states red and green for the traffic light are considered. Periodically the traffic light must reassess its state;

[4]Throughout this scenario, the colour of the traffic light is intended for vehicles and not for pedestrians.

it can either choose to stay in the current state, or transition to the other state. Therefore, there are four possible actions for the traffic light: *(i)* switch from red to green; *(ii)* stay green; *(iii)* switch from green to red, and *(iv)* stay red. The actions for vehicles are assumed to be: *(i)* moving at maximum speed; *(ii)* braking; *(iii)* remaining stopped, or, *(iv)* accelerating. The state of a vehicle is described as the action it is undertaking, its position, speed and direction.

The safety constraint in this scenario is that no vehicle should pass through a red light. If the traffic light is red, i.e., either remaining red or switching to green, the safety constraint may be violated if the vehicle is too close to the traffic light and is not stopped. The actions, remaining red for the traffic light and moving at maximum speed for the vehicle, are not compatible. The states of the vehicle moving at maximum speed and a red light at the traffic light may be compatible, however, if they are far enough apart, i.e., the vehicle is not within the proximity where the traffic light could influence the behaviour of the vehicle.

The responsibility to ensure the safety constraints of the application are not violated is discussed in detail in (Bouroche et al. 2006*a*). One example of responsibility is for an entity, e.g., the traffic light, to adapt its behaviour, i.e., perform an action other than the one planned, to ensure that at all times the incompatibility for which it is responsible will not occur, for example, for the traffic light to remain green, if the safety of the application would be compromised otherwise.

In this scenario, every traffic light will send messages to vehicles early enough for vehicles to have sufficient time to stop before the traffic light when it is red. The traffic light will also warn vehicles early enough so that they have time to pass through the traffic light before it is red, or, stop. This is the contract between the traffic light and the vehicles and can be translated into geographical zones around entities, e.g., the constraint that a traffic light will warn approaching vehicles that it is red early enough, can be translated into a zone within which communication must be guaranteed. More specifically, the traffic light must be able to communicate with vehicles in a zone that is sufficiently large such that vehicles will have time to receive a message and stop before the traffic light, i.e., the desired coverage. Thus, the traffic light must send a message at least over a desired coverage of diameter

$$T_{present} + T_{period} + (T_{v\_reaction} \times v_{max}) \qquad (1)$$

which represents the distance travelled by a vehicle in the time for the vehicle to become present in the actual coverage, $T_{present}$, the periodicity of transmissions on the real-time channel, T_{period} and the reaction time of the vehicle, traveling at v_{max}, upon receiving a transmission, $T_{v\_reaction}$.

If the message is not delivered in the complete desired coverage the traffic light needs to prevent incompatibilities by adapting its behaviour, i.e., the traffic light should switch to (or remain) green.

Thus, the traffic light should also transmit messages over a zone sufficiently large so that it will have time to be notified if the message was not delivered and react to it by adapting its behaviour to switch to green if it was red, or cancel its change to red if it was green. Hence, the traffic light should communicate over a zone of length:

$$(T_{adapt\_notification} + T_{tl\_reaction}) \times v_{max} \qquad (2)$$

which represents the distance travelled by a vehicle in the time required for the traffic light to receive an adaptation notification and adapt its behaviour accordingly.

To amalgamate both considerations for the size of the desired coverage and to guarantee the safety of pedestrians crossing regardless of whether an adaptation occurs, the size of the desired coverage must be at least of length:

$$T_{present} + T_{period} + max(T_{v\_reaction} \times v_{max}, (T_{adapt\_notification} + T_{tl\_reaction}) \times v_{max}) \qquad (3)$$

The delivery latency within which the traffic light should warn vehicles that it will switch to red can be deduced similarly, and is described in detail in (Bouroche et al. 2006*a*).

Unsignalised Junction

The goal of the autonomous junction crossing scenario is to coordinate the crossing of autonomous vehicles at a junction and ensure that the vehicles cross the junction safely. The safety of vehicles at the junction is guaranteed if one vehicle crosses the junction at a time. If a vehicle cannot communicate within a zone of the required size, the vehicle will cancel its crossing and wait until the covered zone is sufficient to communicate with all other vehicles waiting to cross at the junction. In this scenario, both channel sender and receivers are mobile.

Each vehicle that wishes to cross the junction creates a real-time channel with a desired coverage sufficiently large to transmit to all other vehicles approaching the junction, with sufficient time for them to stop at the junction. The desired coverage specified in the create channel interface is absolute in terms of the junction to provide the same level of coverage to all vehicles wishing to cross at the junction from any intersection, as shown in Figure 3.16.

This scenario encompasses a single type of entity, autonomous vehicles. The actions of vehicles are: *(i)* waiting at the junction; *(ii)* crossing; *(iii)* not interested, e.g., when the vehicle is leaving the junction. The state of a vehicle can be described by its action, position and the duration for which it has been waiting to cross the junction (if relevant). The safety constraint is that at any time there should be only a single vehicle in the junction. To ensure that the safety constraint is not violated, any

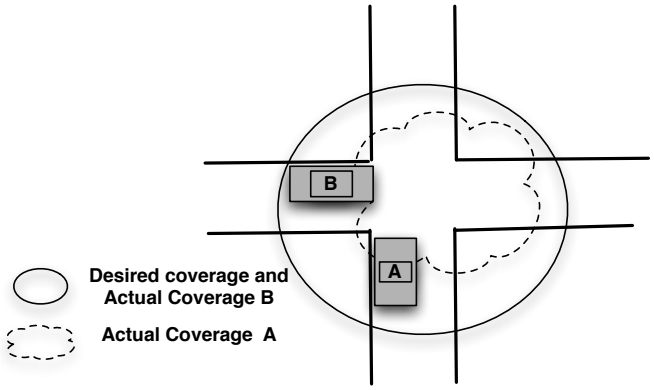


Fig. 3.16: Junction crossing

vehicle entering the junction is responsible for preventing incompatibilities. Thus, when entering the junction each vehicle must maintain that the states of all vehicles will remain compatible. A vehicle sends a message to all other vehicles prior to entering the junction to warn them that this vehicle is intending to cross and all other vehicles should adapt their behaviour to defer crossing until the other vehicle is finished. This is the contract between all vehicles intending to cross at the junction.

When a vehicle intends to cross the junction it should warn other vehicles early enough for them to stop before entering the junction. The calculation of desired coverage size in this scenario, shown in Equation (4), must be at least large enough for a message to be received by all other vehicles at the junction and to adapt behaviour and stop, i.e., the distance travelled in the time for a vehicle traveling at the maximum speed for the road to stop, $T_{stop} \times v_{max}$, after delivering a transmission by the vehicle intending to cross. If it is not guaranteed that all other vehicles delivered the transmission, the vehicle intending to cross the junction must be notified in sufficient time to stop, in the worst-case when traveling at maximum speed, and adapt its behaviour, i.e., to stop before the junction. For example in Figure 3.16, the actual coverage of car A does not cover the desired coverage for the channel and the car may not cross the junction, whereas the actual coverage of car B does cover the desired coverage, and the car may cross the junction. The desired coverage is calculated to encapsulate the maximum distance travelled by a vehicle traveling at maximum speed if either an adaptation occurs or not.

$$T_{present} + T_{period} + max(T_{stop} \times v_{max}, (T_{adapt\_notification} + T_{v\_reaction}) \times v_{max}) \quad (4)$$

The calculation of the delivery latency within which each vehicle intending to cross at the junction must warn the other vehicles is derived similarly, and described in (Bouroche et al. 2006*b*).

Using the properties of the space-elastic model, coupled with the coordination model described in (Bouroche et al. 2006*b*), it has been shown that constraints on the behaviour of autonomous vehicles can be derived to ensure that they will cross the junction safely.

3.4 Summary

This chapter described the design of the space-elastic model to support hard real-time communication in wireless networks. The properties of the space-elastic model are timely communication in an adaptable actual coverage and timely adaptation notification if a change in the actual coverage occurs. This chapter introduced the real-time channel abstraction which is used for communication by space-elastic applications and described the interface to create, transmit and receive using real-time channels in the API of the space-elastic model. Finally this chapter illustrated using two application scenarios the usefulness of the space-elastic model in building real-time applications with both mobile and stationary senders.

Chapter 4

Space-Elastic Adaptive Routing (SEAR) Protocol

This chapter describes the design of the Space-Elastic Adaptive Routing (SEAR) protocol for real-time multi-hop ad hoc routing, which provides the basis for the implementation of the space-elastic model described in chapter 3.

The goal of SEAR is to support the properties of the space-elastic model, i.e., to guarantee time-bounded delivery of messages to hosts listening on a real-time channel and within the actual coverage for a known minimum time, $T_{present}$, before the delivery deadline and to provide time-bounded adaptation notification when a change in the actual coverage occurs.

Section 4.1 introduces the SEAR protocol and the SEAR API. Section 4.2, describes the algorithms used to achieve multi-hop communication in an ad hoc network and section 4.3 describes the algorithms used to provide the properties of the space-elastic model. This chapter finishes with a detailed description of the control information, i.e., the data structures and routing messages, used by the SEAR protocol.

4.1 SEAR Protocol Basics

Before describing the SEAR protocol in detail, some background relating to the factors which influenced the design of the protocol is provided.

4.1.1 Background

SEAR is a multi-hop real-time routing protocol for ad hoc networks. SEAR overlays the ad hoc network with a virtual cellular structure, where each cell is of a known size and shape, and contains potentially mobile hosts. The ad hoc network is completely covered by the union of all cells. All hosts in a cell are within transmission range of each other, and communication within a cell is called *intra-cell communication*.

A cell has the potential to be connected to all other cells adjacent to it. For example, in Figure 4.1, a host in the cell identified by *0* may communicate with the adjacent cells, *1*, *2*, *3*, *4*, *5*, and *6*.

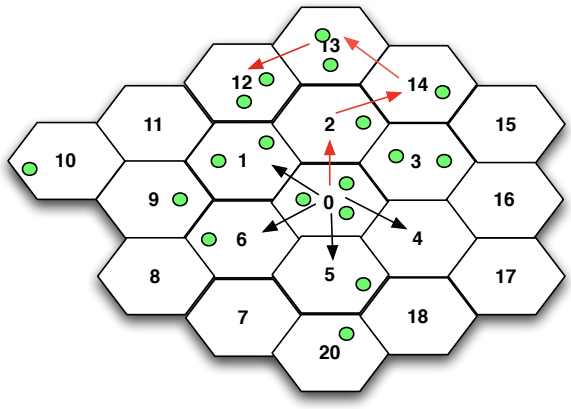


Fig. 4.1: Virtual cellular topology used by SEAR

Communication between adjacent cells is called *inter-cell communication* and is possible if the participating adjacent cells are not empty. Using this cellular structure, communication between adjacent cells is analogous to communication over one hop in a traditional routing protocol. Thus, an inter-cell route is analogous to a multi-hop route, as shown in Figure 4.1, by the arrows connecting cells: $0 \rightarrow 2 \rightarrow 14 \rightarrow 13 \rightarrow 12$.

To guarantee the delivery constraints specified in a request to create a real-time channel requires the distributed reservation of resources on routes in the network that satisfy these constraints. The SEAR protocol performs bandwidth reservation by interfacing with the underlying medium-access control protocol to reserve bandwidth to satisfy the delivery latency of transmissions on the channel. For example, using a slotted TDMA style medium-access layer, as supported by the TBMAC protocol (Cunningham & Cahill 2002), slot reservations to satisfy future real-time transmissions are performed in each cell. To integrate this slot reservation protocol with the cell-based routing paradigm of SEAR

requires the reservation of intra-cell slots, to support real-time transmissions within a cell, and inter-cell slots, to support real-time transmissions between cells.

4.1.2 Overview of SEAR

This section gives a high-level overview of the main features of the SEAR protocol and how it supports the space-elastic communication model.

In SEAR, the desired coverage is a set of cells specified by a channel sender in a channel creation request. The actual coverage is the subset of cells within the desired coverage where the delivery latency of the channel is guaranteed. The SEAR protocol is designed to discover and maintain (potentially) multi-hop routes to satisfy the time and space bounds of a request. In the terminology of the SEAR protocol, such routes are multi-cell routes with intra-cell and inter-cell slots being reserved to support the specified delivery constraints of the real-time channel within the actual coverage.

Route discovery is time-bounded as specified in the request and follows a request-reply paradigm, i.e., a channel creation request propagates from the channel sender to cells within the desired coverage with replies back-propagated to the channel sender within a known time bound.

The SEAR protocol differentiates data messages from control messages. Data messages are associated with transmissions from a channel sender. Control messages are messages required by the protocol, such as route discovery or reply messages. To limit the impact of control messages on data messages, a specific number of slots are allocated exclusively for the exchange of control information both between cells, i.e., inter-cell slots, and within a cell, i.e., intra-cell slots. These control slots are not available for data messages.

Route discovery starts with distributed agreement by hosts in the cell of the channel sender on the allocation of intra-cell data slots to satisfy the delivery latency of the channel. If intra-cell data slots can be scheduled the request is admitted and the cell is included in the actual coverage, otherwise the channel sender is notified of failure to create the channel.

To extend the actual coverage to include the adjacent cells specified as part of the desired coverage requires the scheduling of inter-cell data slots satisfying the constraints of the channel. If no inter-cell data slots are available for adjacent cells the actual coverage includes the cell of the sender only. Otherwise gateways, i.e., hosts elected for communication to/from adjacent cells, transmit updated requests, i.e., with adjusted delivery and discovery latency to reflect the transmission and discovery time to their cell, on the inter-cell control slots to adjacent cells. The request may be forwarded to many adjacent cells and route discovery continues in parallel.

In any cell that receives such a request on an inter-cell control slot, a gateway initiates distributed

agreement on admitting the request in the cell. If intra-cell data slots are not available to satisfy the delivery latency in the adjacent cell, or the request message has been previously admitted in the cell, for example, on a different route[1], a reply is returned immediately to the cell forwarding the request. Time-bounded route discovery continues in each cell by scheduling intra- and inter-cell data slots and forwarding the request on inter-cell control slots to adjacent cells in the desired coverage.

When the time-bound for route discovery in a cell expires the actual coverage of a channel is back-propagated in a reply by the gateway using an inter-cell control slot to the cell that forwarded the request. A reply is expected from all cells to whom a request was forwarded. Failure to receive a reply from an adjacent cell means that the adjacent cell is not included in the actual coverage. At the expiration of the discovery timer in the cell of the channel sender the actual coverage is delivered to the higher layer.

An adaptation occurs when a change in the actual coverage occurs and is detected by a change in the connectivity between adjacent cells. A route break is the loss of connectivity between adjacent cells and is notified as a reduction of the actual coverage to the channel sender within a known time bound. Route recovery attempts to reestablish and extend routes, and thus, is an adaptation that expands the actual coverage. Route discovery during actual coverage expansion is time-bounded and follows the same request-reply paradigm as a channel creation request. The identifiers of new cells in the actual coverage are back-propagated to the channel sender within a known time bound at the expiration of expansion route discovery.

To summarise, the actual coverage of a real-time channel is a set of routes, encompassing a set of cells, extending over the desired coverage that satisfies the delivery latency constraints of the channel. Hosts present within cells covered by the routes for $T_{present}$ are guaranteed to deliver a message at its delivery deadline, which corresponds to $t_{deliver}$. The routes encompassing the actual coverage may change over time, e.g., due to route breaks. Any change to the actual coverage is identified as an adaptation. An adaptation notification is then propagated to the channel sender, within a known time bound, who may then adapt its behaviour accordingly.

The current design of the SEAR protocol accommodates mobile and stationary receivers and stationary senders. As discussed in chapter 1, extending both the SEAR and TBMAC protocols to include mobile senders is the focus of future work.

[1]The first request for a channel that can be satisfied in a cell is admitted.

4.1.3 SEAR API

The functionality provided by the SEAR protocol depends on whether the higher-level application is a channel sender, receiver or both. A channel sender interfaces with SEAR to join the system, create real-time channels, transmit messages on channels and receive adaptation notifications when the actual coverage of a channel changes. A host wishing to receive messages from a channel must listen on the channel specifying a delivery handler for received messages.

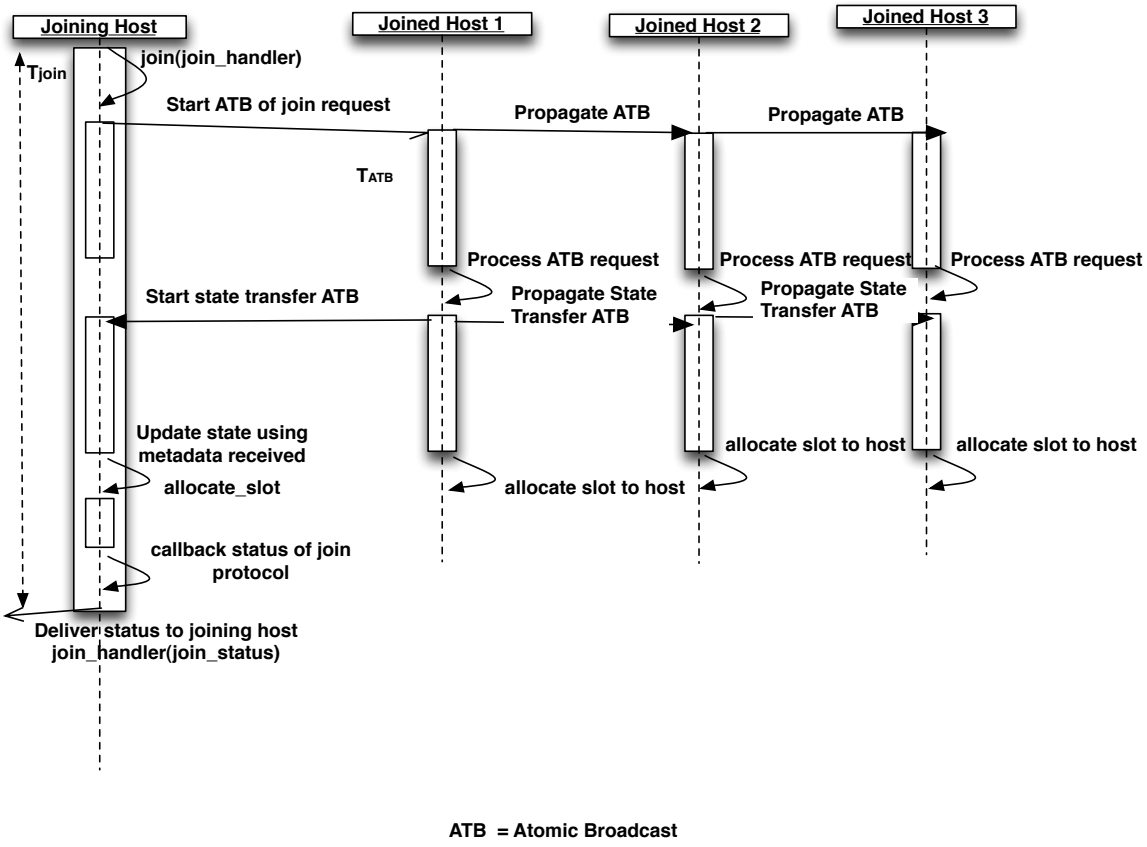


Fig. 4.2: Joining the SEAR protocol

Join

A host that requires to become a channel sender must first successfully join the system. All joined hosts in a cell maintain replicated state information about the cell, including the schedule of reserved intra-cell and inter-cell slots and channels admitted in the cell. To reach distributed agreement and

maintain consistent metadata between all joined hosts in a cell, any message that updates cell metadata is atomically broadcast (Cunningham & Cahill 2002), using the TBMAC protocol.

To join SEAR a host must be allocated an intra-cell slot (to satisfy the join protocol of TBMAC) and maintain consistent metadata for its cell. The process to join the SEAR protocol is illustrated in Figure 4.2.

The join protocol starts with an asynchronous call from the joining host which includes a handler to be used to notify it of the outcome, i.e., success or failure, of the join protocol. SEAR translates the `join` request into a request for an initial slot allocation which is atomically broadcast in the cell. On receiving the request, some joined host replies with a message that includes the current metadata for the cell also using the atomic broadcast protocol. Using this transfer of metadata the joining host will obtain all the information required for decision making in the cell, e.g., to allocate or deallocate slots, at the delivery deadline of the atomic broadcast.

On the delivery of this message, the joining host updates local data structures representing the metadata received about the cell. All hosts in the cell use the current metadata to decide on the outcome of the join protocol for the host, i.e., to allocate an intra-cell slot, or, to deliver a failure notification. The join protocol finishes when the status of the join is delivered to the higher layer.

The duration of the join protocol is known (T_{join} in Figure 4.2,) and depends on the underlying mechanism used for state transfer in the protocol, e.g. in the SEAR implementation in chapter 5, the atomic broadcast protocol of TBMAC is used. The worst-case latency to join SEAR (WC_JOIN) depends on the worst-case time-bounds to join the TBMAC protocol, i.e., be allocated an intra-cell slot.

An inconsistency may arise if there are changes to the cell metadata, for example, a request to remove a channel, during the transfer of state information to the joining host. To prevent inconsistencies during the join protocol, all control messages are stored at the joining host and applied when the transfer of metadata has completed at the joining host. The host applies the control messages in temporal order. The upper bound on the number of control messages received by the joining host during the join protocol is bound by the number of control slots that occur within WC_JOIN, and can be calculated.

When the cell metadata is available and all control messages applied, the joining host performs the same slot allocation procedure as other joined hosts in the cell and allocates itself the same intra-cell slot as allocated by all other hosts.

Following the notification from SEAR that the join phase has completed successfully, the host has joined the system and henceforth maintains all data structures required by SEAR, as described

in section 4.4.2, to discover and maintain real-time channels. A joined host is a participant in all resource reservations and routing decisions and participates in actual coverage adaptation detection and notification. A joined host may create real-time channels, and thus, become a channel sender. A joined host may be elected as a gateway, described in section 4.2.1 for the cell.

The host remains in the joined context until a leave request is signaled to SEAR or the host fails.

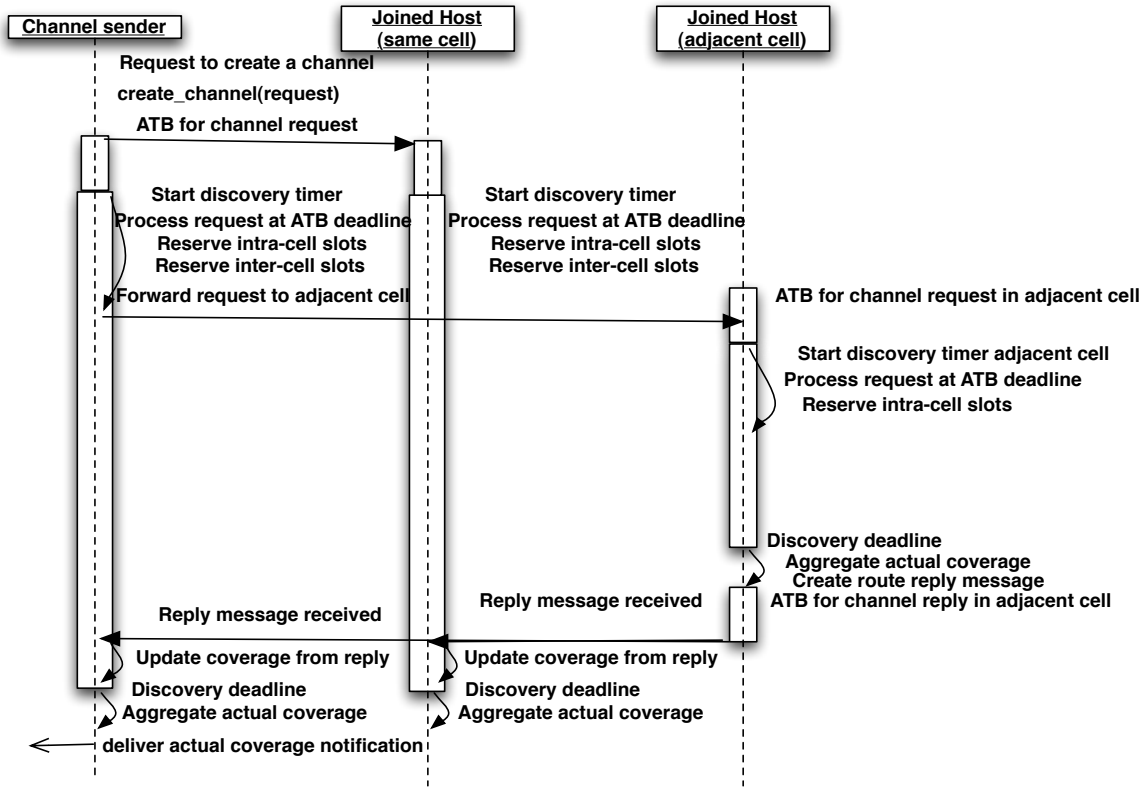


Fig. 4.3: Request the creation of a real-time channel

Create a Real-Time Channel

The properties of a real-time channel are specified by the sender who will transmit messages over the channel.

A channel sender requests the creation of a real-time channel by invoking the `create_channel` function shown in Figure 4.3. Two callback handlers are included in the `create_channel` interface to deliver the outcome of the channel creation to the channel sender within the time bound

specified for route discovery. One handler, `reserve_status`, is invoked for the delivery of the first notification of the actual coverage following the request to create a channel if the actual coverage for the channel is the same as the desired coverage specified. In all other scenarios, i.e., when the initial actual coverage is not the same as the desired coverage, or, following adaptation notification, the handler `adapt_notification` is called. The interpretation of a notification delivered using the `adapt_notification` handler is that the actual coverage for a channel has been adapted and an adaptation of the behaviour of the application may be required.

The SEAR protocol supports periodic message transmissions on a created real-time channel. Therefore, the interface to create the channel includes sufficient information to determine the periodicity of the transmissions on the channel, `period`, and the intended time from which the periodic transmissions start, `start_time`.

The processing of route request messages for route discovery and back-propagated reply messages for actual coverage notification are described in section 4.2.2. The scheduling of intra-cell slots is also described that section.

The created channel persists and the channel sender may transmit on the channel, until the channel sender requests to remove the channel, as described in section 4.1.3, or, until the failure of the channel sender as described in section 4.2.4.

Transmit on a Real-Time Channel

A sender transmits a message on a channel to be delivered at the delivery deadline of the transmission which corresponds to $t_{deliver}$. Sporadic transmissions on real-time channels[2] are not precluded by the current design of SEAR, it is possible to state when the sporadic transmission is required, i.e., the `start_time`, with the `period` set to 0 when creating the channel. In this case, resources will be reserved for the sporadic transmission and removed using failure detection, discussed in section 4.2.4.

To transmit, the sender invokes `transmit` and passes the unique channel identifier returned from `create_channel`, and the real-time data to be transmitted, as shown in Figure 4.4.

The transmission by the sender is started in the slot reserved for transmissions on the channel in the sender's cell. If the actual coverage of a channel extends beyond the sender's cell, the message is transmitted on a reserved inter-cell slot to propagate to the adjacent cell, where it is again transmitted on a reserved intra-cell slot in that cell. The message traverses the set of routes composed of reserved intra-cell and inter-cell slots which constitute the current actual coverage for the channel.

[2]Sporadic in this context means sparse.

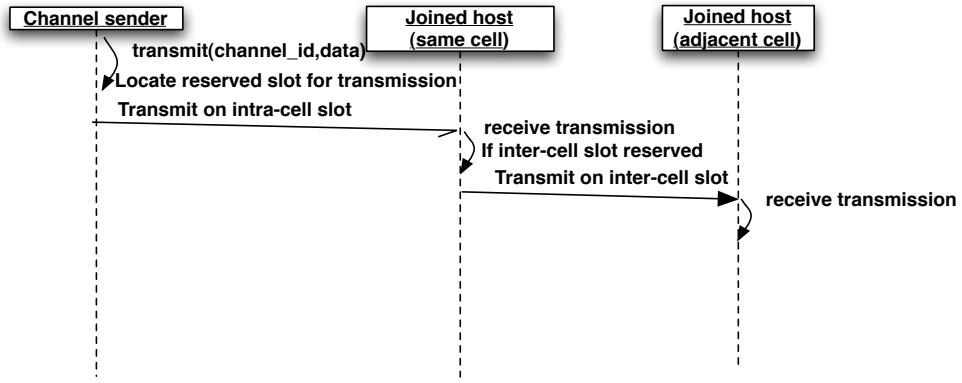


Fig. 4.4: Real-time transmission using a real-time channel

Remove a Real-Time Channel

A real-time channel persists until the sender decides to remove the channel. Removing a channel translates to the deallocation of intra-cell and inter-cell slots reserved for the channel in the actual coverage. The remove channel request traverses the actual coverage of the real-time channel, by forward propagation using intra-cell and inter-cell control slots in the actual coverage. An atomic broadcast protocol is used to guarantee that all hosts in all cells in the actual coverage receive the removal request and decide on the same slots to deallocate.

Listen

Listening on a channel is how a host expresses an interest in transmissions on a channel. A sender may also be a listener, or, a host may be a listener only. In the latter case, the host need not join the SEAR protocol, and therefore is not able to create real-time channels or participate in decision making for real-time channel discovery or maintenance.

To `listen` on a channel a host specifies the channel identifier on which to listen and the callback handler to be invoked to deliver messages arriving on this channel. The latency to listen encompasses the time to store the association between the channel identifier and the callback handler locally at this host. Listening on a channel group persists until the listener either requests to stop listening, or, fails.

Receive a Transmission on a Real-Time Channel

To receive a transmission on a real-time channel the host must become present in the actual coverage of the real-time channel, i.e., be listening on the channel and be within the actual coverage a known minimum time, $T_{present}$ before $t_{deliver}$ for the transmission.

In the SEAR protocol, a transmission on a channel traverses the actual coverage using the reserved intra-cell and inter-cell slots. Transmission in a reserved intra-cell slot has the potential to reach all hosts in a cell. However, a message is only delivered to those hosts that have expressed an interest in transmissions on the channel, i.e., the set of listening hosts.

Any message received on a channel in which this host is not interested is not delivered.

Leave

A channel sender remains a participant in the SEAR protocol until it explicitly requests to `leave` the protocol. Leaving the SEAR protocol, means that any slot allocated to this host is deallocated, and thus, the host may no longer participate in decisions for the SEAR protocol. A request to leave the SEAR protocol does not impact the listening state of a host, i.e., a host that has left the SEAR protocol may still be a listener on a (number of) channel(s).

A channel sender may request to leave the SEAR protocol at any time, i.e., when its channels are still active, or, when all channels have been removed by the sender. In the former case, the request to leave must start the removal of all the channels created by this sender prior to the removal of the allocated slot to the sender. The removal process for each channel is as described in section 4.1.3. Messages transmitted by the sender that have not yet reached their delivery deadline are delivered at their deadline with the rationale of compatibility with the behaviour of the space-elastic application.

Following a leave notification a host must resubmit a join request to send further real-time messages.

4.2 SEAR Protocol Detail

The rest of this chapter describes the design of the core algorithms in the SEAR protocol to support the properties of the space-elastic model.

4.2.1 Role of the Gateway

Multi-hop routing in the SEAR protocol requires a combination of intra-cell and inter-cell communication. The limited transmission range of wireless devices bounds the range within which a message

may be received, with the consequence that it may not be possible for an inter-cell transmission to reach all hosts located in the adjacent destination cell.

In SEAR the impact of the limitation of wireless transmission range is removed by design. Firstly, the size of a cell reflects the transmission range of the wireless medium. Thus, a transmission on an intra-cell slot can reach all hosts in the cell. Secondly, the range of a transmission on an inter-cell slot can be extended by the retransmission of the message on an intra-cell slot in the destination adjacent cell, as shown in Figure 4.5, where a transmission on a real-time channel is started using an intra-cell slot in cell *0* and forwarded using an inter-cell slot to the adjacent cell *3* where the message is rebroadcast on an intra-cell slot. Forwarding in this way continues for each cell on the route. Extending the range of an inter-cell transmission increases the number of hosts that receive the transmission.

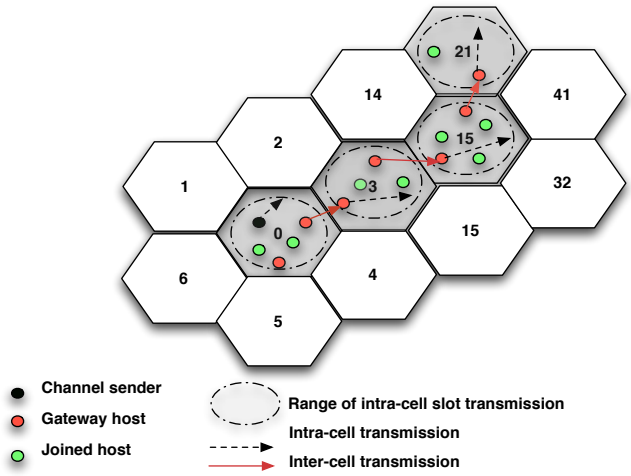


Fig. 4.5: Multi-cell intra-cell and inter-cell communication

The host in a cell responsible for extending the range of a transmission from an adjacent cell is the *gateway*. The gateway host is elected by all joined hosts in a cell and is the interface with all adjacent cells. The gateway is responsible for forwarding transmissions received on intra-cell slots in its cell by retransmitting the message on inter-cell slots to the destination adjacent cell, as shown in Figure 4.5 between cells *3* and *15*. The gateway also monitors adjacent cells in the desired coverage of any channel admitted in its cell to determine changes in connectivity between the cells, and thus, adaptations of the actual coverage.

The ability to discover routes in the desired coverage and propagate transmissions in the actual

coverage depends on the availability of at least one gateway for a cell and is critical for the functionality of SEAR. There may be a number of gateways per cell, e.g., a gateway per adjacent cell. Decisions on the number of gateways to elect per cell may be related, for example, to the number of adjacent cells, the expected traffic between cells, the mobility of hosts and potential dynamics in the network. There is no requirement to have the same number of gateway hosts in each cell. The failure of an elected gateway, or the failure to elect a gateway to interface with an adjacent cell in the actual coverage of any channel, is translated to an actual coverage adaptation of the relevant channel(s).

Gateway Eligibility

To achieve inter-cell communication a gateway must be located close to its cell boundary with an adjacent cell. The criteria used to elect a gateway are based, firstly, on the location of the host in the cell, or more specifically, the location of the host in relation to the cell boundaries with adjacent cells, and, secondly, on the mobility of the host, i.e., the estimated duration that this host will be near the cell boundaries. The implementation of the division of the cell to determine regions of the cell of interest for gateway election is discussed in detail in chapter 5.

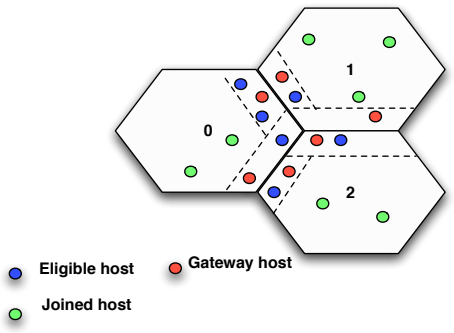


Fig. 4.6: Thresholds within a cell

A host is elected as gateway to an adjacent cell if the host has joined the system and is within a threshold region from the cell boundary, i.e., the host can communicate with an adjacent cell to perform inter-cell communication. For example, in Figure 4.6, the threshold region is shown by dashed lines at the boundary of each cell. The elected host should also be the "best" host located in this threshold region. For example, the least mobile host or the host in the centre of the region with the greatest distance to cross before leaving the threshold area. It is assumed that no two hosts will be at the same location at the same time, therefore, it is always possible to select a host based on its

location within the threshold regions.

To determine the location of a host in a cell, the host must piggyback its current location[3] on all transmissions it originates. At a minimum, the slot exclusively allocated when the host joined the system is used for the propagation of location information. However, to increase the accuracy of location information, additional slots reserved for transmissions on real-time channels by a sender, are also used to piggyback location information.

A host is no longer eligible to be a gateway when it requests to `leave` the system, and thus, is no longer maintaining metadata for the cell. Alternatively, a joined host may no longer be eligible to be elected as gateway due to movement, i.e., to a location beyond the threshold region, or, due to speed, i.e., the host is moving too fast to remain in the threshold region for a duration long enough to communicate with an adjacent cell. A host that is no longer eligible to be a gateway may still participate in gateway election.

Joining the Election

A host may join the system at any time. The scenario may arise where a host joins the system with insufficient time prior to the next gateway election to receive all location information from transmissions in the cell on which to base an election decision. To prevent a host with insufficient information participating in an election, a joined host enters a learning phase prior to its first participation in an election. The duration of the learning phase is an artefact of the implementation and can be calculated. When the learning phase for the joined hosts ends the host may participate in the next, and all subsequent, elections while in the cell and remaining a joined member of the system. The eligibility of the host to be elected as gateway is time-varying based on the mobility of the host within the cell.

Gateway Election

Gateway election is performed periodically. The first phase of the election is to elect the required number of eligible hosts as gateways for a cell. Each host participating in the election determines if new gateways should be elected. If the current gateways are still eligible and are the "best" for the role, there is no requirement to elect new gateways. If there is a requirement to elect a new gateway, i.e., because the previous gateway is, or will within a duration shorter than the election period, no longer be eligible, the location information maintained for each host is evaluated to determine the host to elect as gateway. Using the location information received with transmissions in a cell all hosts make the same decision on a host to elect as gateway for the cell. Failure to receive a transmission

[3] Available using GPS for example.

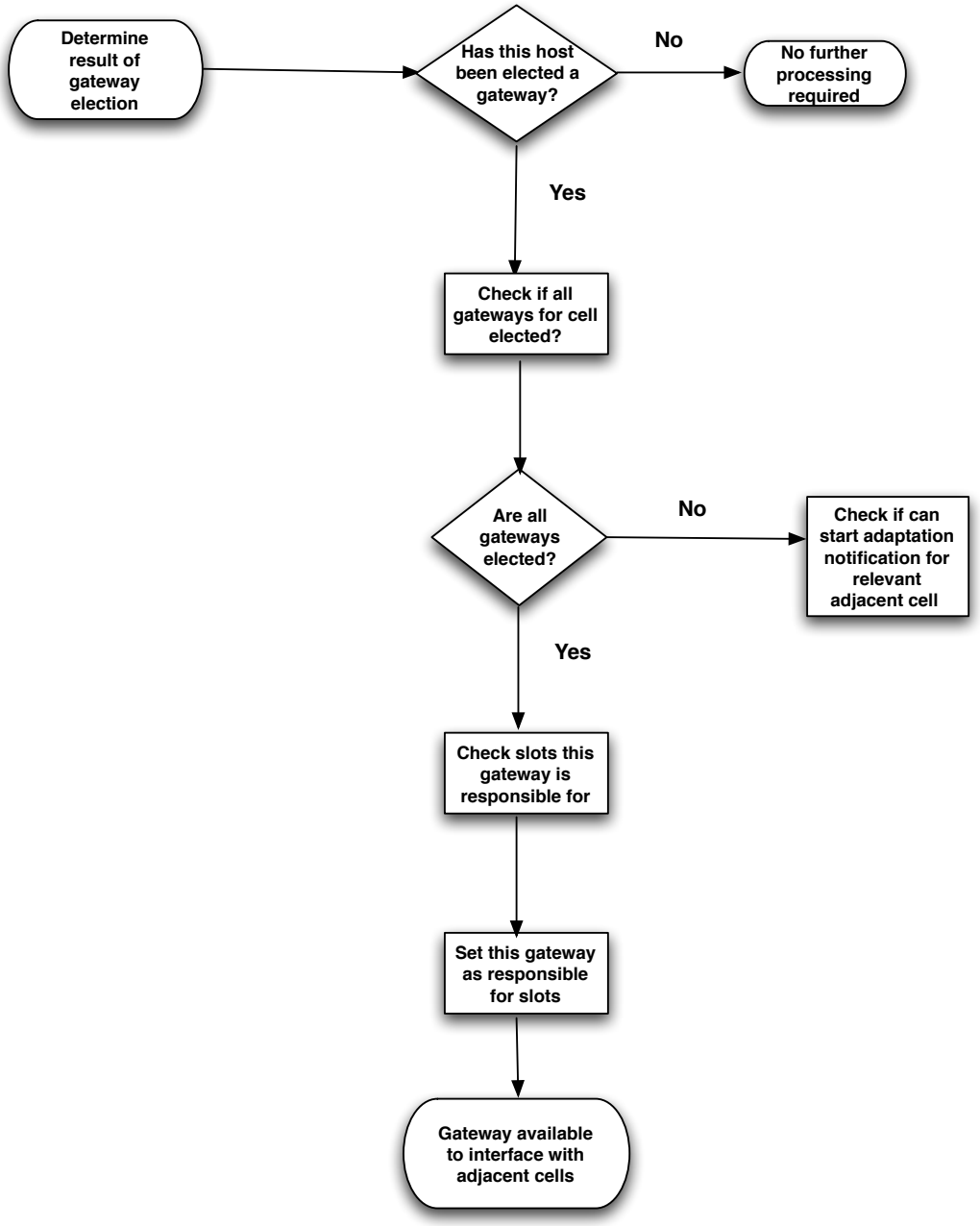


Fig. 4.7: Functionality of an elected gateway to become available for adjacent cell communication

by any joined host in a cell means that inconsistent location information on which to base an election decision exists. The failure suspicion protocol, described in section 4.2.4, caters for this scenario.

The second phase of the election requires all elected gateways to perform the duties of the gateway role which are illustrated in Figure 4.7 and are enumerated as follows:

1. Check all gateways are elected - if a gateway could not be elected, the current cell cannot communicate with at least one adjacent cell. The implication is that the actual coverage of all channels that include routes between these cells has been adapted. Adaptation notification starting from the cell that detected the adaptation is transmitted to the channel sender.

2. Check intra-cell slots for which this gateway is responsible - the failure detection protocol, discussed in section 4.2.4, uses the absence of transmissions in an intra-cell slot as the implicit indication that the slot is no longer in use. In a remote cell, i.e., a cell in the actual coverage in which the channel sender does not reside, an intra-cell slot allocated for a transmission on a real-time channel, may be wrongly identified as unused if transmissions on the channel are infrequent and no other transmissions are made using the slot. To avoid erroneous failure detection a gateway in a remote cell for a transmission on a real-time channel transmits a NULL message in each reserved intra-cell slot when no transmission on the channel is expected, as calculated using the `start_time`, `period` and latency to reach the cell.

An elected gateway also has a role in route discovery and adaptation detection and notification, and is highlighted when appropriate in the following sections.

4.2.2 Route Discovery and Resource Reservation

The SEAR protocol uses the desired coverage specified in the call to `create_channel` to scope the extent of the network in which the discovery and maintenance of routes and resources are necessary to satisfy the specified delivery latency of a channel.

The requirement to reserve both intra-cell and inter-cell slots is determined on a cell-by-cell basis depending on the location of the cell in the desired coverage, e.g., a cell with no adjacent cells in the desired coverage does not perform inter-cell slot reservation. Transmissions in slots reserved for intra-cell and inter-cell communication in the actual coverage of a real-time channel should guarantee the delivery of the message in the actual coverage at the delivery deadline of the transmission.

To start route discovery, the parameters to `create_channel` are translated into a `request_msg_t` routing message, (see section 4.4.3), and are transmitted in the intra-cell slot allocated to the channel sender in the join protocol. The `request_msg_t` propagates within the desired coverage using control

intra-cell and inter-cell slots to discover the actual coverage for the corresponding channel. The `max_discovery_latency` parameter to the `create_channel` function, bounds the duration of route discovery.

Route Discovery within Specified Time Bounds

To support time-bounded route discovery each joined host, regardless of its location in the desired coverage, starts a timer when the `request_msg_t` for a new real-time channel is received. The route discovery timers set by hosts in the actual coverage reduces with distance from the sender to accommodate the timeliness of a reply from anywhere within the actual coverage.

The calculation of the discovery deadline, $t_{discovery\_deadline}$ for the channel sender is

$$discovery\_deadline = maximum\_discovery\_latency - upcall\_discovery\_latency \quad (1)$$

where *upcall_discovery_latency* includes time for the reception of the notified actual coverage and the delivery of the notification to the higher layer.

The latency to propagate a `request_msg_t` to a cell and the latency to back-propagate a reply to the cell from which the request was received (and on a route to the channel sender) are deducted from the discovery latency when the request is received in a cell. The route discovery timer is set as the remaining discovery latency calculated as shown in (2), and includes the latency for a gateway to retransmit the `request_msg_t` using an intra-cell slot in the cell.

$$discovery\_latency\_remaining = discovery\_latency - (latency\_to\_here + latency\_from\_here) \quad (2)$$

The calculation of *latency_to_here* is the elapsed time for the request to reach this cell from the discovery start, t_{disc_start}, and includes the intra-cell retransmission by a gateway, as shown in (3).

$$latency\_to\_here = t_{now} - \ t_{disc\_start} \quad (3)$$

latency_from_here is the calculated back-propagation latency from this cell to the cell of the channel sender and is calculated as :

$$latency\_from\_here = (actual\_num\_cells \times back\_propagate\_reply\_latency) \quad (4)$$

where each *back_propagate_reply_latency* includes the population of a reply message, `reply_msg_t`, and the latency to transmit the `reply_msg_t` to the adjacent cell from which the `request_msg_t` was received. The durations of *latency_to_here*, *latency_from_here* and *back_propagate_reply_latency* are guaranteed using a timely medium access control protocol.

A gateway forwards a `request_msg_t` received if the calculated *discovery_latency_remaining* is sufficient to forward the request to the adjacent cell and back-propagate the composite view of actual coverage from the adjacent cell on a route to the channel sender.

The expiration of a discovery timer in a cell signals the start of the aggregation of the actual coverage for a real-time channel using `reply_msg_t` messages received in this cell and the back-propagation of the aggregated actual coverage to the adjacent cell from which a `request_msg_t` was received. The back-propagation of a reply continues on the route the create request traversed to the cell of the channel sender. The retransmission of the `reply_msg_t` using an atomic broadcast started by a gateway in the cell of the channel sender[4], is received by the channel sender.

Admitting a Request for a Real-Time Channel

All joined hosts in a cell perform an admission control procedure upon reception of a `request_msg_t` message. Admission control in a cell first decides whether there are data intra-cell slots available for admission in the cell, i.e., to satisfy the delivery latency of the received request, and following this if the request should be forwarded to an adjacent cell, i.e., whether the adjacent cell is in the desired coverage and whether inter-cell slots are available for forward propagation to the adjacent cell.

The latency of transmissions on channels can be known by the intra-cell and inter-cell slots reserved for the transmissions and the latency to transmit using these slots in cells on routes in the actual coverage. Thus, the latency of a transmission to reach a cell in the actual coverage using reserved slots can also be known. Using this observation, the scheduling policy in SEAR is to reserve the intra-cell slot that will be current (with access to the wireless medium) at the time when a real-time transmission reaches a cell. If this slot is already in use, the next available slot is reserved. The rationale for this policy is to minimise the latency for transmission in each cell (where the latency encompasses the delay to transmit in a slot following reception in a cell and the duration of the transmission), and thus, maximise the remaining transmission latency available for admission control decisions in other cells in the desired coverage with the goal of maximising the actual coverage of the channel.

The decisions made by each joined host upon reception of a `request_msg_t` message are shown in Figure 4.8.

A `request_msg_t` message is admitted in the current cell if there are sufficient resources available to satisfy the constraints of the input delivery latency (Line 1). If a local resource, i.e., an intra-cell slot, is available for transmissions on the real-time channel, a further decision on the inclusion of an

[4]The channel sender may also be a gateway for the cell.

```
1:    if( localResourceAvailable( deliveryLatency, latencyToHere ) )
2:             if( nextHop in desiredCoverage)
3:                     if( nextHopResourceAvailable( deliveryLatencyRem) )
4:                             reserveResource( nextHop )
5:                             forwardToNextHop( updatedRequest )
6:                     else
7:                             ResourceNotAvailable( nextHop )
8:             else
9:                     nextHopNotInDesiredCoverage( nextHop )
10:      else
11:           ResourceNotAvailable( thisHop )
```

Fig. 4.8: Pseudo code for route and resource discovery at each hop

adjacent cell in the actual coverage route is made (Line 2), and whether a resource, i.e., an inter-cell slot, is available to forward messages to the adjacent cell subject to the remaining delivery latency available following adjustment to reflect the accumulated latency to transmit using reserved resources on the route to this cell (Line 3). Lines 4 and 5 reserve resources for forwarding the transmission and forward an adjusted `request_msg_t` message. Lines 7 through 11 cater for the scenarios where there are insufficient resources to include the current and/or adjacent cells in the actual coverage or an adjacent cell is not within the desired coverage specified in the `request_msg_t` message.

Omitted for clarity from Figure 4.8 is a decision as to whether forward propagation to an adjacent cell is possible given the remaining route discovery latency. This decision is considered in the context of Line 5, and must determine the feasibility of forwarding a real-time request while guaranteeing the back-propagation of a reply on a route to the channel sender within the specified discovery latency. An observation is that intra-cell and inter-cell slot scheduling are not performed in parallel. Only if an intra-cell slot is available is the channel admitted and inter-cell slot scheduling performed depending on the constraints of the desired coverage. The incremental transmission latency using reserved intra-cell and inter-cell slots on a route in the actual coverage is used to schedule slots to satisfy the delivery latency of the channel and is only known in each cell following intra-cell slot scheduling.

Slots are reserved for admitted channels in a cell using a soft-state approach. The `start_time`, `period` and propagation delay of a transmission on a channel are used to calculate the expected time of transmission in a reserved intra-cell slot in each cell on a route in the actual coverage. The failure to transmit in an intra-cell slot when expected means the resource reservation is not refreshed for the channel, and the failure suspicion protocol, described in section 4.2.4, is started.

Creating a Route Reply

The coverage included in a `reply_msg_t` from a cell in the actual coverage is the aggregate coverage based on `reply_msg_t` messages received from adjacent cells. A `reply_msg_t` message propagated from a cell is uniquely associated with a previous `request_msg_t` message propagated to the cell.

Actual coverage aggregation is performed as follows. A negative (NACK[5]) reply message received from an adjacent cell means resources, i.e., intra-cell slots, are not available to satisfy the delivery latency specified in an associated channel creation request. However, a NACK reply for one route to a cell may be negated by any ACK[6] reply on a different route to the same cell, i.e., if the cell is adjacent to more than one cell in the desired coverage. The delivery latency to reach a cell varies depending on the route traversed and resources reserved and thus, data intra-cell slots may satisfy the delivery latency on one route which would not satisfy the delivery latency on an alternative route.

If a cell in the desired coverage can schedule resources that satisfy the delivery latency of a channel creation request, the cell is included in the actual coverage of the channel. Actual coverage aggregation is illustrated in Figure 4.9, where the channel sender is in cell 0 and 1, 2, 3, 4 and 5 denote cells within the actual coverage. Thus, the cell with the least discovery latency remaining, in this case, cell 5 in Figure 4.9, starts the back-propagation of the actual coverage in a reply message to the channel sender, and all other cells aggregate the actual coverage from replies received.

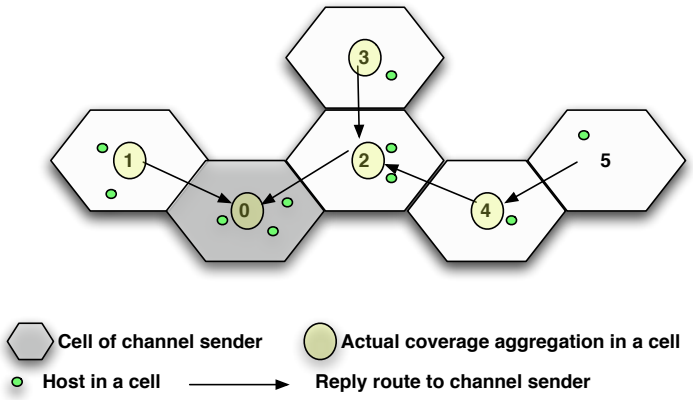


Fig. 4.9: Aggregating the actual coverage of a real-time channel

[5]Negative route reply
[6]Positive route reply

Back Propagation of a Route Reply Message

At the expiration of the route discovery timer in a cell, a gateway starts the back-propagation of the reply message on a route to the channel sender. Hosts in a cell expect to receive a reply from each adjacent cell to whom a request to create a channel was forwarded. The absence of a reply from an adjacent cell is interpreted as a NACK status for the cell and the cell is not included in the actual coverage. Using the calculation of remaining discovery latency in a cell, `discovery_latency_remaining`, a reply from an adjacent cell should be received prior to the expiration of the timer in the forwarding cell. For example, in Figure 4.9, the aggregate view of the received actual coverage from cell 5 is back-propagated to cell 4 where the aggregate view of the received actual coverage from 5 and 4 are back-propagated as a reply to cell 2. In cell 2 the replies received from cells 3 and 4 are aggregated and the actual coverage is back-propagated to cell 0, the cell of the channel sender. In cell 0, the replies received from cells 1 and 2 are used to aggregate the actual coverage for the real-time channel.

At the expiration of the discovery timer in the cell of the channel sender the aggregate actual coverage is delivered to the higher layer.

4.2.3 Actual Coverage Adaptation

The actual coverage of a real-time channel may change over time. To guarantee both the progress of the channel sender and the safety of the real-time application, the adaptation of the actual coverage is notified to the channel sender within a known time bound, the adaptation notification time. Adaptation notification is performed on a per-channel basis.

The first notification of actual coverage occurs upon the expiration of the route discovery timer of the channel sender and identifies the actual coverage available for the real-time channel. Included in the notification delivered is the worst-case adaptation notification time (WC_ADAPT), and the current adaptation notification time based on the current actual coverage available for the channel. All future adaptation notifications delivered to the sender include only the current adaptation notification time based on the current actual coverage available for the channel.

The adaptation notification time encapsulates the time to detect the actual coverage adaptation and to back-propagate and interpret the notification at the channel sender.

Detecting Adaptation of the Actual Coverage

The detection of an adaptation of the actual coverage is time bounded. Adjacent cells in an actual coverage are connected if both cells are populated and there is an inter-cell slot reserved for the

```
1:    if( ConnReceived() )
2:             if( previouslyInContact() )
3:                     // no change to connectivity status
4:             else
5:                     // expansion of actual coverage
6:                     if (reserveResource( nextHop) )
7:                             forwardToNextHop( updatedRequest )
8:     else
9:             if( previouslyInContact() )
10:                  // reduction of actual coverage
11:                  backPropagateReduction( nextHop )
12:          else
13:                  // do nothing no change in actual coverage
```

Fig. 4.10: Pseudo code for interpreting connectivity status

propagation of transmissions on a real-time channel between the cells. An adaptation of the actual coverage is detected as any change to the connectivity of adjacent cells.

The status of the connectivity between adjacent cells in the actual coverage is checked periodically using the exchange of beacons from a forwarding cell to the successor adjacent cells on routes in the actual coverage. Only gateway hosts in a cell perform the beacon exchange, but all hosts in the cell participate in the adaptation decision and notification. The beacon exchange period, $T_{connectivity}$, is the calculated minimum duration to transmit one beacon on an inter-cell slot to a destination adjacent cell, say cell X, and to receive an amended beacon as a reply on an inter-cell slot from cell X. A change in connectivity is identified as shown in Figure 4.10.

The first check at $T_{connectivity}$ is to determine whether a beacon has been received in this period (Line 1). The reception of a beacon may imply a change in the connectivity between adjacent cells, and thus, a change in the actual coverage (Line 2). If the adjacent cell was connected before the current $T_{connectivity}$ period, and has replied in this $T_{connectivity}$ period, there is no change to the status of the replying adjacent cell. Lines 4-7, cater for the scenario where the replying adjacent cell was not previously connected, but a beacon has been received within this $T_{connectivity}$ period. This is an example of *actual coverage expansion*. To facilitate an expansion attempt an inter-cell slot (for communication of real-time transmissions) must be reserved in the beaconing cell, i.e., the forwarding cell on a route, to satisfy the delivery latency of a stored request for the creation of a real-time channel[7] whose actual coverage includes the newly contactable adjacent cell. If an inter-cell slot is available a

[7]There may be a number of create channel requests that include the newly contactable adjacent cell in the desired coverage. The selection of the `request_msg_t` is on a First-Come-First-Served basis from the time of reception of the create channel request.

stored `request_msg_t` is forwarded to the newly contactable cell and time-bounded actual coverage expansion is initiated (Line 7).

Lines 8-13 cater for the alternative scenario where no beacon has been received within the $T_{connectivity}$ period. If the adjacent cell was previously contactable, the lack of a beacon is interpreted as an *actual coverage reduction*, and the back-propagation of the adaptation notification is started (Line 11). If the adjacent cell was not within the actual coverage in the previous period and is still not contactable, there is no change in the actual coverage.

To detect the expansion of the actual coverage, cells at the boundary of the current actual coverage of any real-time channel periodically beacon adjacent cells within the desired coverage but not currently within the actual coverage.

$T_{connectivity}$ bounds the latency to detect a change in the actual coverage, i.e., this is the maximum duration that may elapse prior to a change in the actual coverage being detected, and is included in the adaptation notification time delivered to the channel sender.

Back Propagation of Adaptation Notification

The detected actual coverage adaptation is delivered to the channel sender. The latency to back-propagate the adaptation notification to the sender is included in the current adaptation notification time delivered to the sender.

An actual coverage adaptation may occur at any time and at any location in the actual coverage. To deliver a notification of an adaptation to the channel sender, the adaptation notification must traverse a route to the sender. In each cell on the back-propagation route, a gateway interfacing with the cell from whom the adaptation notification was received retransmits the notification using an atomic broadcast for all hosts, including gateways, in the cell to receive. Each gateway in a cell checks whether it is responsible for interfacing with an adjacent cell on a back-propagation route to the sender and if so, transmits the notification on the inter-cell slot for communication with the adjacent cell. This process continues until the back-propagation reaches the cell of the channel sender. The gateway in this cell, retransmits the notification using an atomic broadcast which is received by the channel sender.

4.2.4 Host Failure Management

The failure of a host who is a channel sender means the slot allocated during the join protocol and all slots reserved for message transmission on channels created by this host are unused at their expected transmission times. For the remainder of this section, the slot allocated during the join protocol and

any slots reserved for transmissions by the sender on channels are collectively called reserved slots.

The SEAR protocol assumes a fail-safe model of host failure, i.e., a joined host exhibits correct behaviour, i.e., transmits in reserved slots, or, stops, i.e., the host is no longer a member of the system and does not participate further in the SEAR protocol until rejoining the system. The failure of a host to transmit in its reserved slots is interpreted as failure of the host.

Recovering unused slots increases both the number of slots available to allocate to hosts joining the system, which in turn increases the number of hosts that can join the system, and the number of slots available to satisfy requests for real-time channel creation. The SEAR protocol uses failure detection and resource recovery to detect and recover reserved, but unused, slots.

Failure Detection

Failure detection in the SEAR protocol is started with a notification by the medium-access control (MAC) layer using the `slot_not_used` callback, as shown in Figure 4.11, of the absence of a transmission in a slot. The MAC layer monitors reserved intra-cell slots to determine if at least one transmission has been performed in each slot within a time bound calculated by the MAC layer, e.g., in TBMAC the time bound is one round of the TBMAC protocol (Cunningham & Cahill 2002).

```
void slot_not_used(in int slot_num) ;
```

Fig. 4.11: Callback to SEAR to notify a failure to transmit in a reserved slot

Given the characteristics of the wireless medium it is possible that some hosts, e.g., in the same region of a cell, do not receive a transmission in a reserved slot. The MAC layer of each of these hosts makes an autonomous decision to invoke the `slot_not_used` callback. The notification from the MAC layer raises a *suspicion* of host failure to the SEAR protocol. The SEAR protocol starts a failure suspicion procedure to reach distributed agreement with all joined hosts in a cell to determine if: a) the host has failed and resource recovery for reserved slots should be started, or, b) the host has not failed and no change is necessary to the slots reserved for the host. Inter-cell slots do not have a designated owner and are not monitored by the MAC layer. However, inter-cell slots reserved by a failed channel sender are recovered using a resource recovery procedure in the SEAR protocol.

Failure Suspicion

For the duration of the failure suspicion procedure a host may be either in the state *under suspicion*, i.e., if it is the host responsible for transmissions in the slot identified by `slot_num`, or, *suspicious*, i.e., a host that received a callback to notify it of the failure to receive in the slot identified by `slot_num`.

A slot is reserved for exclusive use by a host at a point in time, i.e., based on the `start_time` and `period` specified in the request to create a channel. Thus, there is at most one host in the state *under suspicion* for a point in time. There may, however, be a set of *suspicious* hosts.

The failure suspicion procedure, as shown in Figure 4.12, starts by using metadata maintained by the SEAR protocol to map `slot_num` to the host responsible for transmitting in the identified slot in the TBMAC round in which the suspicion was raised. The host responsible for transmitting in an intra-cell slot may have any of the roles of joined host, channel sender or gateway, which are not mutually exclusive. Using metadata maintained in each cell, the role of the responsible host can be determined, for example, it is possible to determine if a sender is under suspicion and resides in the cell where the suspicion is raised.

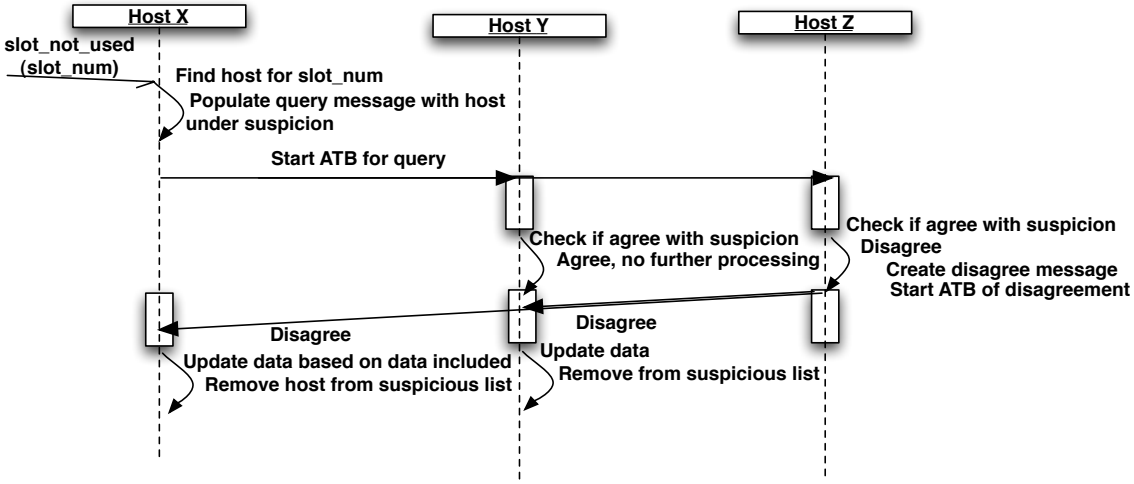


Fig. 4.12: Failure suspicion protocol

The details of the host under suspicion are stored locally in a table, `sus_hosts_log_t` by the suspicious host, where each entry includes the time at which the host became under suspicion and whether the host under suspicion resides in the cell where the suspicion is raised. The suspicious host queries all other joined hosts in the cell by sending an atomic broadcast in its allocated slot, to solicit

a disagreement on the suspicion from any other host. For example, in Figure 4.12, host X is suspicious of host A and propagates, using atomic broadcast, a query about host A to other hosts in the cell. Host Y is also suspicious of host A, however host Z disagrees with the suspicion. Host Z propagates a disagreement message using atomic broadcast which both hosts X and Y receive and remove host A from suspicion. If host X does not receive notification of a disagreement to its suspicion within the time bound to receive a disagreement message, all other hosts in the cell implicitly agree with the failure suspicion and decide that host A has failed. Depending on the role of the failed host reserved resources are recovered.

Resource Recovery

The role of the host under suspicion determines the context for resource recovery. Using metadata maintained at each joined host the context of the failed host is determined. For example, an intra-cell slot allocated to a joined host during the join protocol should be recovered, whereas the failure of a channel sender requires the recovery of intra-cell and inter-cell slots in its cell and the cells that constitute the actual coverage of each of its channels. The functionality to determine the failure of a host and recover any resources reserved for the host is illustrated in Figure 4.13.

Joined Host: A joined host is responsible for transmissions in the slot allocated during the join protocol only. To recover this slot requires distributed agreement by all hosts in a cell to deallocate the slot, which is possible following the atomic broadcast of a suspicion to which no disagreement was received.

Channel Sender: A channel sender is responsible for transmissions in the slot allocated during the join protocol and transmissions in intra-cell slots reserved for its channels. If a channel sender is under suspicion in a cell, the slot allocated during the join protocol and all slots reserved for its channels should be recovered. Following the atomic broadcast to solicit a disagreement to a suspicion in a cell, all hosts have sufficient information to reach a decision on the reserved slots to be recovered in the cell of the channel sender.

If the actual coverage of such a channel includes adjacent cells any intra-cell and inter-cell slots in these cells should also be recovered. Resource reservations in cells in the actual coverage are refreshed using a soft-state approach. Failure to refresh the reservation by a transmission on a channel and following the completion of the failure suspicion protocol, all slots reserved for the channel in the cell are recovered.

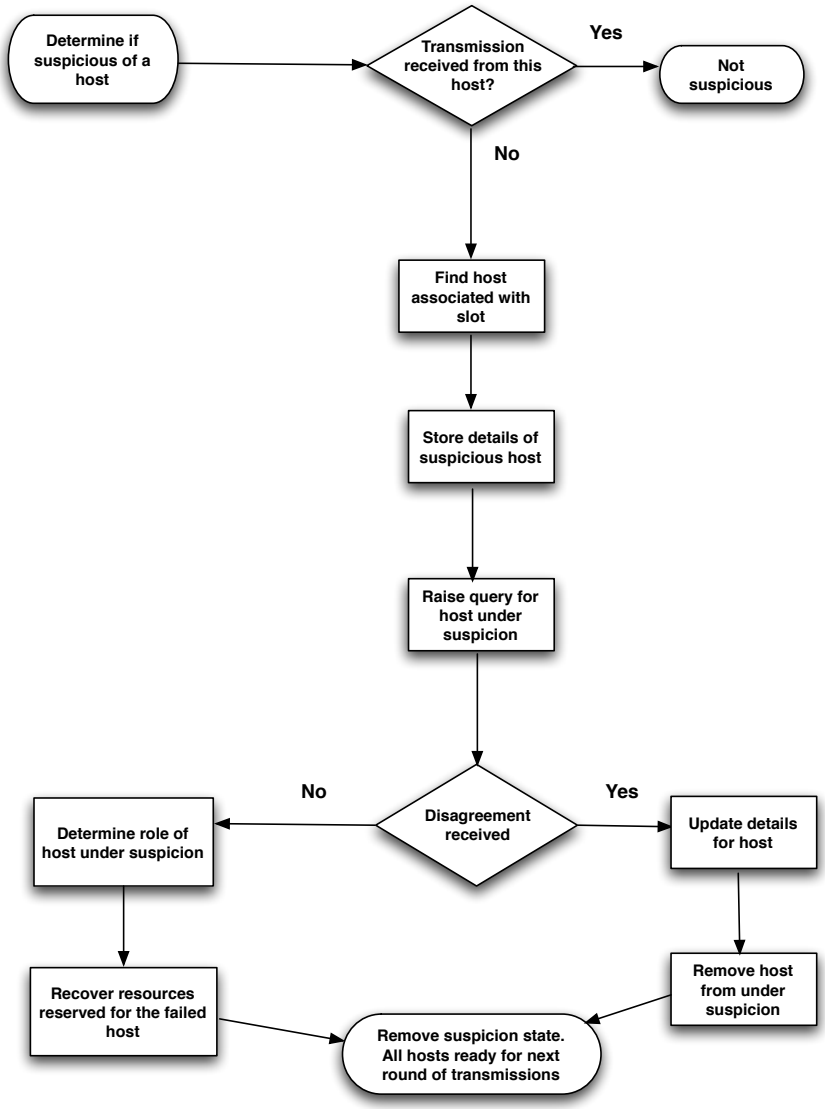


Fig. 4.13: Failure suspicion protocol

Gateway Host: A gateway in a cell within the actual coverage of a real-time channel is responsible for retransmitting messages received on reserved intra-cell slots within the cell. The failure of the gateway to transmit in an intra-cell slot for which it is responsible may be attributed to: a) the gateway has failed; b) a gateway in a previous cell has failed; or c) the channel sender has failed. In the latter two scenarios, the failure to transmit on a route to a cell means a gateway in the cell has no message to retransmit at the expected transmission time and resource recovery is by the soft-state approach explained previously.

The failure of a gateway, on a route in the actual coverage[8] is detected within a time bound ($T_{connectivity}$) by adjacent cells and notified to the channel sender as an adaptation. The failure to transmit in reserved slots in the remaining cells in the actual coverage is detected and resources are recovered using a soft-state approach.

To summarise, the failure of the channel sender is detected in its cell where resources are recovered following distributed agreement. In all other cells, it is not possible to differentiate between the failure of the channel sender or a gateway on a route. The failure of a gateway on a route in the actual coverage means that real-time messages are not available in its cell, and subsequent cells on the route, thus, the actual coverage has adapted. The adaptation is identified either during beaconing for connectivity or at gateway election, both of which are executed at the same known periodicity. Resource recovery is performed in each cell using a soft-state approach.

4.3 Maintaining Properties of the Space-Elastic Model

The design of the SEAR protocol supports the properties of the space-elastic model, as described in chapter 3.

Property 1: Hosts present in the actual coverage of a real-time channel for $T_{present}$ deliver the message at the calculated delivery deadline, $t_{deliver}$ for the transmission

The channel sender specifies the delivery latency for a transmission on a real-time channel when the channel is created. The delivery latency represents the maximum duration a transmission on a channel will traverse the actual coverage prior to the delivery deadline, $t_{deliver}$, when the message is delivered by all hosts present, for $T_{present}$, in the actual coverage. Thus, $t_{deliver}$ is calculated as an instant in time that represents the offset of the delivery latency, `msg_latency`, from the transmission

[8]The failure to elect a gateway in a cell in the actual coverage of a real-time channel is accommodated in the same way.

start time, i.e., the call to `transmit` by the channel sender.

$$t_{deliver} = t_{transmission\_start} + delivery\_latency \quad (5)$$

The SEAR protocol guarantees that all hosts that become present for at least $T_{present}$ before $t_{deliver}$ deliver the message at the delivery deadline, $t_{deliver}$. The time at which a message transmitted on a channel is received by hosts present in the actual coverage varies depending on distance from the sender. The SEAR protocol guarantees that regardless of host location within the actual coverage, a message received is delivered by all hosts that become present at least $T_{present}$ before $t_{deliver}$ for the transmission. To achieve this, the SEAR protocol, calculates an interval which is the delay to deliver, $T_{delivery\_delay}$, which must elapse prior to message delivery at a receiver. The $T_{delivery\_delay}$ varies with distance from the channel sender.

$T_{delivery\_delay}$ is calculated on a per cell basis using equation (6).

$$T_{delivery\_delay} = (t_{transmission\_start} + delivery\_latency) - latency\_to\_here \quad (6)$$

where `latency_to_here` represents the propagation delay of a message transmitted on a channel to reach the current cell and subsequently all hosts present in the actual coverage.

Transmission within a specified delivery latency is started by a channel sender who initiates the real-time transmission using the `transmit` interface. A message is propagated within the actual coverage where all hosts present in the actual coverage for $T_{present}$ prior to $t_{deliver}$ defer delivery of the message for $T_{delivery\_delay}$, and deliver the message at $t_{deliver}$ for the transmission. The mobility of a receiver or an adaptation of the actual coverage may mean that the host is not present in the actual coverage for $T_{present}$. Transmission delivery at $t_{deliver}$ is not guaranteed in this case.

Property 2: Notification of an adaptation of the actual coverage to a channel sender is guaranteed within a calculated time bound.

The space-elastic model guarantees both the progress of a channel sender and a real-time application by guaranteeing that notification of an adaptation of the actual coverage is delivered to the channel sender within the worst-case adaptation notification time for the channel, WC_ADAPT.

The SEAR protocol calculates WC_ADAPT for transmissions on a channel based on the desired coverage specified, e.g., using the route with the most cells to reach the bound of the desired coverage.

Notification of a Reduction The notification of a reduced actual coverage is guaranteed to be delivered to the channel sender within the last adaptation notification time which is bound by WC_ADAPT. The adaptation notification time includes the latency to detect the adaptation,

bound by $T_{connectivity}$ as described in section 4.2.3, and the latency to back-propagate the notification in the reduced actual coverage. The adaptation notification latency to back propagate the notification to the channel sender is calculated as follows:

$$T_{connectivity} + ((actual\_num\_cells \times latency\_process\_notification) + (actual\_num\_cells - 1 \times back\_propagation\_latency)) \quad (7)$$

where *actual_num_cells* represents the number of cells the notification must traverse in the remaining actual coverage to reach the channel sender. Both *latency_process_notification*, i.e., the latency to process a received adaptation notification message, and *back_propagation_latency*, are calculated based on the guarantees provided by the timely medium-access control layer, i.e., the latency to transmit the notification using intra-cell and inter-cell slots reserved for control messages in the remaining actual coverage.

In Equation (7), $T_{connectivity}$ is fixed but the remaining values in the equation are variable depending on the extent of the remaining actual coverage. A reduction in the actual coverage represents a reduction in the back-propagation route to the channel sender, and thus, a reduction in the adaptation notification time.

Notification of an Expansion An expansion of the actual coverage means additional cells have been included in the actual coverage. The extent of the expansion of a channel is only known when the expansion is complete. To determine the extent of the expansion, a route discovery lasting for *discovery_latency_remaining* as calculated in section 4.2.2, is started in the cell that detects the expansion. At the end of route discovery a notification of the expanded actual coverage is notified to the channel sender. The adaptation notification time for the expanded actual coverage, is also calculated by Equation (7).

Property 3: The actual coverage of a real-time channel is always $\leq$ the desired coverage specified for the channel

In the SEAR protocol the desired coverage is identified by a set of cells, where each cell is uniquely identifiable. Control slots are allocated for intra-cell and inter-cell transmissions with specific adjacent cells. Route discovery and resource reservation in the SEAR protocol, involves the mapping of cells in a specified desired coverage to associated intra-cell and inter-cell slots.

4.4 Control Information in SEAR

Control data in SEAR includes all data that is not transmitted on a real-time channel. Timeliness guarantees are required for the exchange of control messages. Control messages that require a change to the metadata maintained at each joined host in a cell, e.g., requests to create a channel, replies about resource availability, are transmit using an atomic broadcast protocol within a cell.

4.4.1 Co-Existence of Control and Data Transmissions

In the SEAR protocol, two different classes of both intra-cell and inter-cell slots are available: data slots and control slots, as shown in Figure 4.14. Control slots are used exclusively for the exchange of control messages. Control information is of a known size and is wholly encapsulated in a control slot. The calculation of the slot size is discussed in chapter 6. Data slots are exclusively reserved for messages transmitted on real-time channels. Control slots are not exclusively owned but are used by an elected gateway in the cell.

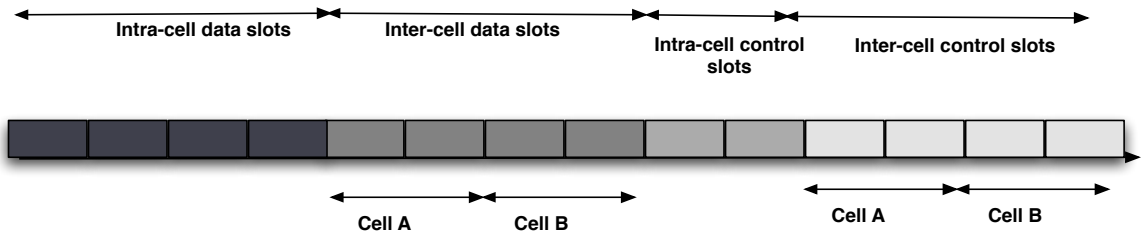


Fig. 4.14: Control slots and data slots in a round

The number of control slots available is known by all joined hosts in a cell a priori. The upper bound on the number of control slots limits the amount of control information it is possible to transmit which may lead to competition to transmit in a control slot. For example, a requirement to transmit at the time bound for route discovery may coincide with the requirement to transmit a beacon in a control slot.

To reduce competition, the control slots are subdivided into slots per adjacent cell, i.e., each adjacent cell has a set of control slots available for transmissions to/from the cell. Thus, competition to transmit in a control slot only occurs if there is a requirement to transmit more than one control message to the same adjacent cell. In addition, control messages are prioritised as follows starting with the highest precedence:

1. Connectivity beacon
2. Route reply
3. Route request
4. Route removal

A connectivity beacon has the highest priority to detect adaptation of the actual coverage. A route reply, which provides the initial actual coverage for a channel, is transmitted in preference to a route request as the deferral of a create channel request reduces the remaining route discovery latency for the channel, which is less critical than delaying the back-propagation of reply within a known discovery latency.

4.4.2 Control Metadata

All hosts that have joined the SEAR protocol maintain the metadata initalised during the join protocol. The following data structures are maintained by each joined host in a cell and are described in the rest of this section.

1. `schedule_per_slot` - maintains the intra-cell and inter-cell reservations.
2. `channels_admitted` - maintains the admitted channels in a cell.
3. `channel_coverage` - maintains the coverage of channels during route discovery.
4. `adjacent_channel_coverage` - maintains the connectivity of adjacent cells.

Schedule Per Slot

The `schedule_per_slot_t` array maintains the metadata relating to the usage of reserved intra-cell and inter-cell slots in a cell in the actual coverage, as shown in Figure 4.15. Using this data structure it is possible to determine allocated and unallocated slots and determine the expected usage pattern of a slot, and thus, detect failure to use a slot when expected. The size of the schedule per slot array is bounded by the number of slots available for intra-cell and inter-cell transmissions within a cell and between cells respectively, for example, the number of CFP slots in the TBMAC protocol.

SEAR attempts to maximise slot usage by allowing sharing of slots for transmissions on channels that are guaranteed to occur at distinct times, i.e., transmissions are never required in the same slots at the same time. Since the periodicity and start time of transmissions on channels are known the

```
typedef struct {
         unsigned long long period ;
         short admitted_idx  : 4;
         RTIME next_usage;
         unsigned long long msg_latency_to_here ;
} slot_sharing_t;

typedef struct {
         unsigned slot_num :4 ;
         share_slot_t share_flag :1;
         slot_sharing_t sharing_slot[MAX_SHARE];
} schedule_per_slot_t ;
```

Fig. 4.15: Schedule per slot data structure

intervals when the slots are not required for transmissions are calculable. The SEAR protocol supports the scheduling of these idle intervals for transmissions by other real-time channels, and are stored in the `slot_sharing_t` structure, shown in Figure 4.15.

To guarantee all joined hosts in a cell reach the same decision on the reservation of intra-cell and inter-cell slots, the schedule per slot array must be synchronised across them. The schedule per slot array is included in the transfer of metadata during the join protocol. Updates to the schedule per slot array are subject to the atomic broadcast protocol to guarantee consistency with all hosts performing the same decisions on the same data at the same time.

```
typedef struct {

         unsigned long long channel_id : 48;
         unsigned short orig_proximity[MAX_COVERAGE];
         long long msg_latency_rem;
         long long msg_latency_to_here;
         long long discovery_time_rem;
         short cell_channel_sender ;
         short forwarding_cell;
         ........
         RTIME start_time ;
         long long delivery_delay ;

} channels_admitted_t ;
```

Fig. 4.16: Channels admitted data strucutre

Channels Admitted

Each channel admitted in a cell has an entry in the channels admitted array, where each entry in the array is described by the structure in Figure 4.16. Using this data structure it is possible to recreate a channel request (with updated delivery and discovery constraints relevant to the current cell) and is used to identify duplicate requests, metadata relating to the channel, and for expansion discovery. The size of the array is equal to the maximum number of concurrent real-time transmissions possible in a cell, which is bound by the slots available and the maximum number of transmissions allowed to share a slot.

A new entry is made in the channels admitted array if an intra-cell slot is available that satisfies the delivery latency of transmissions on a channel. All joined hosts in a cell must maintain an identical channels admitted array. The join protocol, as described in section 4.1.3, incorporates the transfer of the current channels admitted array for this cell to a new host. Subsequently, updates to the channels admitted array are synchronised by all joined hosts in a cell using an atomic broadcast to guarantee the same updates are performed on the same data at the same time.

```
typedef enum {
         ACK,
         NACK,
         DACK,
         UNKNOWN,
         FORWARD
}coverage_status ;

typedef struct {
          unsigned short covered_cells [MAX_COVERAGE]  ;
          unsigned short not_covered_cells[MAX_COVERAGE];
}channel_coverage_t ;


typedef struct {
         unsigned long long channel_id :48 ;
         unsigned long long channel_sender_id :48 ;
         unsigned short forwarding_cell_id :3 ;
         coverage_status this_cell  ;
         channel_coverage_t coverage_per_adj_cell[MAX_ADJ] ;
         coverage_status this_adj_cell[MAX_ADJ] ;
} channel_coverage_per_cell_t ;
```

Fig. 4.17: Channel coverage data structure

Channel Coverage

The channel coverage data structure is an array of `channel_coverage_per_cell_t` entries, shown in Figure 4.17. The channel coverage array is maintained by all hosts in a cell. An entry in the array is made for each channel for which a request for a route discovery is ongoing in a cell, either at initial discovery or during an expansion.

Each `channel_coverage_per_cell_t` entry, contains the coverage accumulated from route replies received from adjacent cells.

An entry in the channel coverage array is maintained in a cell for the duration of the remaining discovery time (`discovery_time_rem`) for the channel. At the discovery deadline, all entries relating to the channel are aggregated to create an aggregate view of actual coverage that is back-propagated to the channel sender. The entries are subsequently removed.

The channel coverage array is not included as metadata transfered in the join protocol. A race condition between the reception of reply messages during the join protocol and decisions made by the joining host can be avoided if the joining host stores all replies received after the request to join. If the discovery deadline for the cell is reached prior to the end of the join protocol, the joining host plays no part in the back-propagation of a reply from this cell, but has all replies received to update its metadata when the join protocol completes. If the discovery deadline occurs after the completion of the join protocol, the host has joined and participates in the back-propagation from the cell.

```
typedef struct {
         coverage_status adj_cell_status ;
         short admitted_idx :4 ;
} status_per_channel_t ;

typedef struct {
         unsigned short adj_cell_id :3;
         unsigned prev_num_slot_free :3;
         RTIME conn_start ;
         RTIME conn_period ;
         status_per_channel_t channel_status[MAX_CHANNELS_ADJ_CELL];
} adj_cell_coverage_t ;
```

Fig. 4.18: Adjacent channel coverage data structure

Adjacent Channel Coverage

The adjacent channel coverage array, shown in Figure 4.18, maintains metadata relating to the status of each adjacent cell in the actual or desired coverage of a real-time channel. The size of the adjacent channel coverage array is limited by the maximum number of adjacent cells to whom it is possible to forward a route request, `MAX_ADJ`. An entry is made in the adjacent channel coverage array, `adj_cell_coverage_t`, when the status of connectivity between cells is known.

All joined hosts in a cell maintain the adjacent channel coverage array which is transferred during the join protocol. All updates to the array are synchronised amongst all hosts in the cell to ensure any host elected as gateway has current information on the connectivity available between cells.

4.4.3 Routing Messages

The two routing messages of most interest in the SEAR protocol are: the request to create a real-time channel, `request_msg_t`, and the back-propagation of a reply containing the actual coverage of a channel at the deadline for route discovery, or, when an adaptation of the actual coverage has been detected, using `reply_msg_t`.

Structure of a Request for Channel Creation

```
typedef struct {
         route_msg_reason_t request_reason :3;
         unsigned short  orig_proximity[MAX_COVERAGE];
         unsigned long long period ;
         RTIME start_time ;
         long long msg_latency_rem ;
         long long msg_latency_to_here;
         unsigned short hop_count_to_here ;
         long long max_msg_latency;
         unsigned long long channel_id :48 ;
         unsigned long long channel_sender_id : 48 ;
         unsigned short cell_channel_sender ;
         unsigned long long discovery_start_time ;
         long long discovery_duration ;
         ........
} request_msg_t ;
```

Fig. 4.19: Route request message

A `request_msg_t`, shown in Figure 4.19, is populated based on the parameters input to the

create_channel function and is propagated where possible in the cells specified in desired coverage. The combination of channel_id and channel_sender_id are used to uniquely identify a request for a channel and are stored as the key for each entry in the channels_admitted array.

All joined hosts in a cell process a request_msg_t message received. The request_msg_t message must maintain sufficient detail to decide:

1. If the delivery latency required for transmissions on the channel, using the combination of msg_latency_rem , start_time, period and hop_count_to_here, is supported using the (potentially shared) slots available in the cell;

2. If adjacent cells are in the desired coverage of the channel, based on the orig_proximity field;

3. If a time-bounded reply of the actual coverage is possible from this cell within the combination of the specified discovery_start_time and discovery_duration for the channel.

The fields of a received request_msg_t message are updated to forward the message to an adjacent cell within the desired coverage for the channel, e.g., msg_latency_to_here is modified to reflect the propagation latency of the route to the forwarding cell, and msg_latency_rem is updated to reflect the delivery latency remaining for messages traversing this route.

```
typedef struct {
         route_msg_reason_t reply_reason :3  ;
         unsigned long long channel_id :48 ;
         unsigned long long channel_sender_id :48 ;
         coverage_status this_cell  ;
         channel_coverage_t coverage_for_cell;
         int num_covered : 4;
         int num_not_covered :4;
         unsigned short adapted_cell_id  :3 ;
         short actual_num_cell_on_route;
         .......
} reply_msg_t ;
```

Fig. 4.20: Route reply message

Structure of a Reply to a Channel Sender

A reply_msg_t structure, as shown in Figure 4.20, is returned with the actual coverage for a cell on a route of a channel and is identified by the combination of channel_id and channel_sender_id. The actual coverage included in the message is maintained in two arrays identifying those cells that are

covered and those cells that are not covered by the actual coverage. A count of the number of cells in each of these arrays is maintained as `num_covered` and `num_not_covered` respectively, and are used in the aggregation of actual coverage.

The reception of `reply_msg_t` messages at the channel sender terminates the back-propagation of a route reply. Upon reception and processing of received `reply_msg_t` messages by the channel sender, either from initial route discovery or following an adaptation, the current view of actual coverage of a real-time channel is available.

4.5 Summary

This chapter described the design of the SEAR protocol to provide the properties of the space-elastic model over a multi-hop ad hoc network. In chapter 6, the SEAR protocol is used to evaluate the space-elastic model in the real world.

Chapter 5

Implementation of the SEAR Protocol

This chapter describes the implementation of the SEAR protocol to provide real-time multi-hop ad hoc routing in a dynamic ad hoc wireless network. The space-elastic model requires real-time guarantees at all layers of the protocol stack. This chapter starts with a discussion of the implementation of a cross-layer architecture on which the SEAR real-time routing protocol relies. The rest of this chapter describes the implementation of the core algorithms of the SEAR protocol supporting the properties of the space-elastic model.

5.1 Real-Time Architecture

The implementation of the SEAR protocol incorporates a cross-layer design approach to provide real-time guarantees at each layer and to combine to support the real-time guarantees of the space-elastic model. The layers and interactions between the layers are illustrated in Figure 5.1.

The SEAR protocol relies on the Time-Bounded Medium-Access Control Protocol (TBMAC), discussed in chapter 2, to provide time-bounded allocation of slots, which SEAR reserves to satisfy the delivery constraints of real-time channels by achieving time-bounded access to the wireless medium.

The TBMAC protocol is implemented as a TDMA layer above the IEEE 802.11 wireless ad hoc protocol and serves to reduce the influence of the characteristics of the IEEE 802.11 contention-based approach to medium-access. The implementation and evaluation of a single-cell approach, i.e., to provide deterministic medium-access to hosts within a cell, is described in (Hughes et al. 2006).

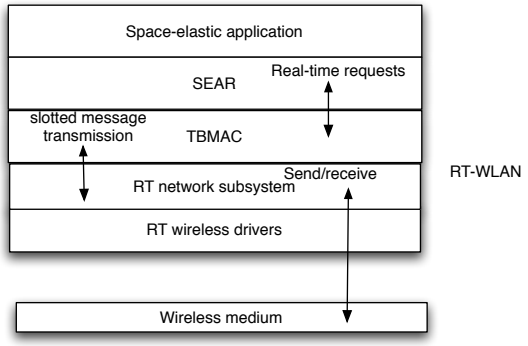


Fig. 5.1: Real-time architecture

The implementation of the SEAR protocol necessitated an extension to the TBMAC implementation to support timeliness guarantees for multi-hop (multi-cell) transmissions. The design of the extensions to the implementation of TBMAC was performed by myself and implemented by Mark Gleeson (Gleeson et al. 2006).

5.1.1 Extensions to the TBMAC Protocol

The SEAR protocol requires distributed agreement for any decision that updates the metadata for a cell to maintain consistent metadata across all joined hosts in the cell. TBMAC uses an atomic broadcast protocol to reach distributed agreement on slot management decisions. To support the functionality required by SEAR, the atomic broadcast interface of the TBMAC implementation was decoupled from use in slot management decisions exclusively and exposed as a callable interface with an additional callback from TBMAC being introduced to notify SEAR of the deadline of an atomic broadcast.

The new functions are shown in Figure 5.2, where the `AdditionalInfo` parameter (of TBMAC) is used to carry the shared metadata required for the SEAR implementation.

```
int atomic_broadcast(struct  AdditionalInfo*) ;
int notify_make_decision(struct AdditionalInfo*) ;
```

Fig. 5.2: Atomic broadcast interface and callback

Slot scheduling by SEAR attempts to reserve specific intra-cell slots that satisfy the delivery latency

of channels. To support this, the slot allocation and deallocation procedures of TBMAC were modified to allow (de)allocation of specific slots, as shown in Figure 5.3.

```
int allocate_specific_slot(int slot_num, long long channel_sender) ;
int deallocate_specific_slot(int slot_num) ;
```

Fig. 5.3: Extended slot management functionality

To increase the availability of slots, the SEAR protocol also allows slot sharing for transmissions in slots that are guaranteed to occur at a distinct time. The impact of this extension on the TBMAC protocol was the inclusion of a slot locking mechanism to ensure that a slot with multiple users is locked and only deallocated when there are no users of the slot remaining. In addition, the concept of ownership of a slot in TBMAC was adapted for the SEAR implementation to provide the ability to change the owner when required, e.g., if the current owner no longer wishes to transmit in the slot. The functions to lock slots and change ownership are shown in Figure 5.4.

```
int lock_allocated_slot(int slot_num) ;
int change_slot_owner(int slot_num, long long new_owner) ;
```

Fig. 5.4: Extended slot management functionality

The most important extension to the implementation of the TBMAC protocol was the inclusion of support for inter-cell slots allowing communication between adjacent cells. Inter-cell slots are exclusively associated with transmissions to/from specific adjacent cells, for example, inter-cell slot 5 is associated with transmissions to cell 3 whereas inter-cell slot 6 is associated with transmissions from cell 3.

```
int change_channel(unsigned short cell_id) ;
int send_pkt(int slot_num, message_hdr * full_msg) ;
```

Fig. 5.5: Extensions for inter-cell communication

To address the hidden terminal problem the TBMAC protocol assigns a specific wireless channel to a cell. Thus, inter-cell communication encompasses a change to the channel of the destination cell (`change_channel`) and a transmission on an inter-cell slot allocated for transmissions to the adjacent

cell (`send_pkt`), as shown in Figure 5.5.

To implement timely medium-access requires predictable behaviour at all phases of packet send and receive, thus, a further prerequisite is to implement a deterministic Linux network subsystem to achieve predictable asynchronous packet transmission and reception, and thus, a basis upon which the TBMAC protocol could be implemented and is described in the next section.

5.1.2 Real-Time Network Subsystem

A general-purpose Linux operating system has three main sources of unpredictability specifically effecting message transmission and reception: *(1)* a dependence on the dynamic allocation of `socket_buffers` for message transmission and reception; *(2)* interruptible interrupt dispatching and interrupt service routine (ISR) execution, and *(3)* queuing received messages for future notification to higher layers. With RTAI (Mantegazza et al. 2000) providing the real-time Linux environment, it was still necessary to implement a real-time network subsystem[1], RT-WLAN (Hughes & Cahill 2006), the modules of which are shown in Figure 5.6, to address these three sources of unpredictability.

1. **Predictable socket buffer allocation**

   Real-time memory management design must guarantee that memory is available and maintained in physical RAM. A `MemoryManager` module creates a pool of `RTsocket_buff` structures (a real-time `socket_buffer` abstraction), implemented as a fixed-size doubly-linked list that is guaranteed to remain in scope until explicitly removed. Using the `MemoryManager` interface, `RTsocket_buff` structures are available to all other modules in RT-WLAN.

2. **Timely interrupt handling**

   RTAI provides a framework for a timely interrupt dispatcher by diverting interrupt handling to RTAI and then Linux. A new real-time interrupt handler was implemented in `RTorinoco` and registered with RTAI to service interrupts by diverting them to RTAI and then PCMCIA. With timely interrupt dispatching available, the next step is to guarantee that interrupt service routines are timely. Thus, all interrupt service routines were designed to execute with interrupts disabled, guaranteeing that all interrupts are serviced to completion, with a predictable latency.

3. **Timely notification of packet reception**

   Eliminating unpredictable interrupt handling removes one source of uncertainty in packet reception. The implementation of a real-time queuing strategy removes the other. The `RTDeviceWrapper`

[1] RT-WLAN is referred to as a network subsystem as all modules apart from RTorinoco and RTorinoco_cs are network independent and are available for real-time network drivers to be plugged in.

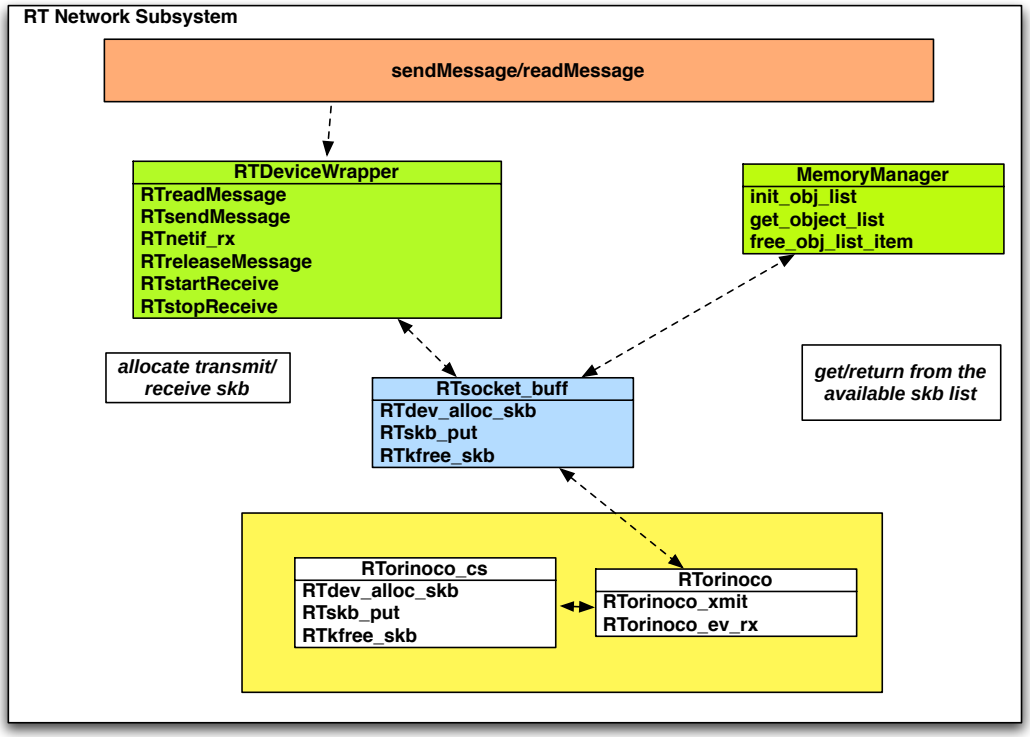


Fig. 5.6: Real-time network subsystem

module provides an interface to `RTnetif_rx`, the real-time version of the standard Linux network interface `netif_rx`. Using this interface, a real-time memory pool, representing the pending packets, is maintained. This real-time memory pool is updated during the non-interruptible packet reception interrupt service routine, `RTorinoco_ev_rx`, within which immediate notification of packet availability is made to the higher layers. Combining real-time interrupt handling with predictable packet queuing provides real-time packet reception notification to higher layers.

Three sources of transmission latency are incurred prior to a message leaving a wireless card: *software latency*, incurred in the interval from the initial transmission request to the transfer of the message to the device driver queue; *firmware latency*, incurred in the transfer of the message to the physical wireless card, and finally, *communication latency*, incurred awaiting wireless medium access (Hughes & Cahill 2006).

Using RT-WLAN, the objective is to obtain a predictable software latency regardless of the current offered load. As shown in Table 5.1, this objective was achieved and maintained regardless of the

Transmission period	Software	Firmware
20ms	384μs	120μs
5ms	387μs	122μs
3ms	389μs	124μs
1.5ms	388μs	159μs

Table 5.1: Average software and firmware latency

increase in offered load, illustrated by the decreasing period between packet transmissions. These experiments were repeated in environments with high levels of wireless interference with similar results obtained with a maximum variance of $\pm 5\mu$ from the illustrated cases. The firmware latency is not under software control, but as illustrated a maximum variance of 40μs under high offered load is possible. Thus, a known and tolerable upper bound is available. With both predictable software and firmware latency achieved, the communication latency, i.e., the phase of packet transmission under the influence of the IEEE 802.11 MAC protocol, is the main source of unpredictability for wireless message transmission latency that remains and for which the TBMAC protocol was designed.

The TBMAC protocol invokes the `sendMessage` and `readMessage` interfaces to invoke the functionality of RT-WLAN.

5.2 Background to the SEAR Implementation

The implementation of the SEAR protocol is encapsulated in the `sear` kernel module, which consists of a number of kernel submodules as shown in Figure 5.7. The interactions between the submodules to support the functionality of the protocol will be described in the rest of this chapter.

The TBMAC protocol overlays a virtual cellular structure on an ad hoc network. The SEAR protocol uses this cellular structure to achieve multi-hop, (multi-cell), communication. To support multi-cell communication, the SEAR protocol uses a cell map that uniquely identifies each cell in the map and the relationship between cells. The cell map is populated at module initialisation and is available to all hosts with a loaded `sear` kernel module.

A host can check what cell they are in and set the cell when moving using the interface provided in Figure 5.8.

Each host maintains the metadata structure, `adj_cell_coverage_t` (see chapter 4, Figure 4.18) which contains details about adjacent cells of a cell and includes an entry for each adjacent cell. The field, `adj_cell_id` identifies the unique cell identifier from the cell map and is used to or-

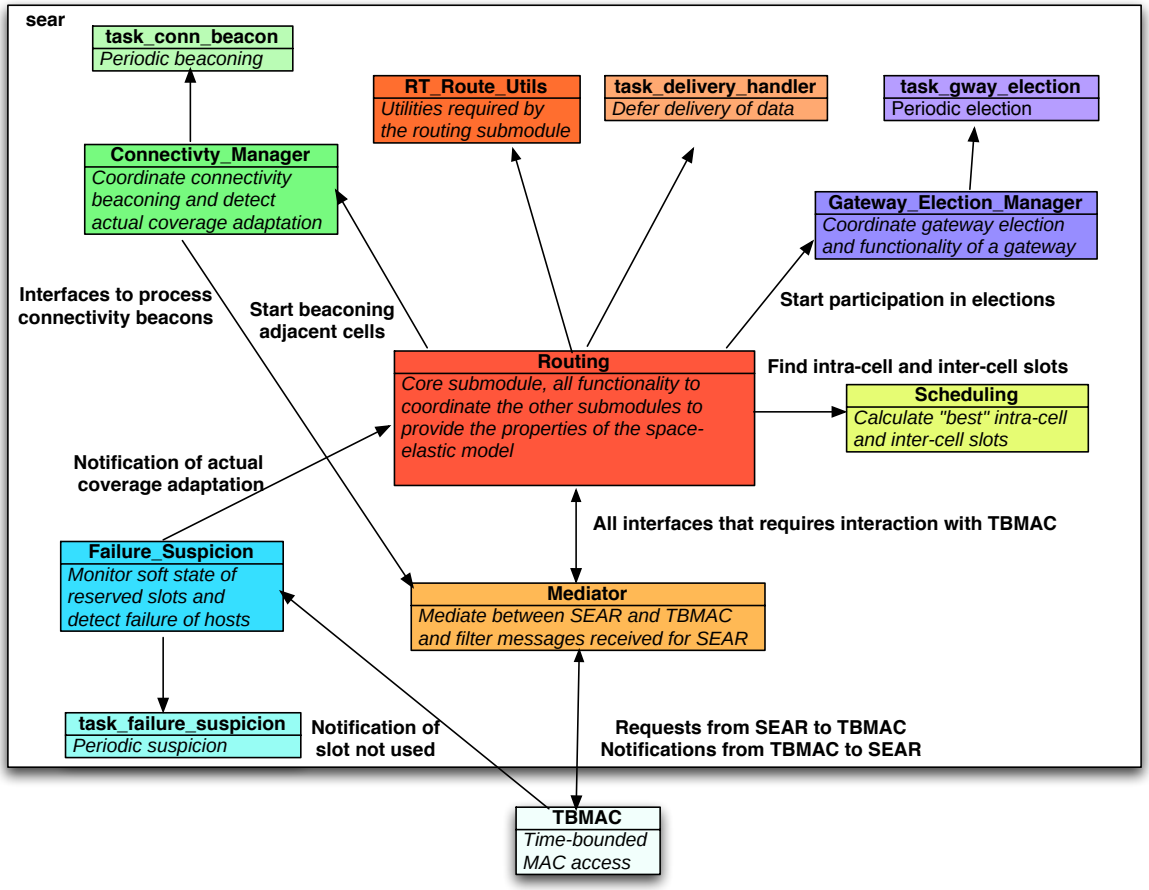


Fig. 5.7: Components of the SEAR module

der the adjacent cell entries by ascending cell identifier. The offset at which an adjacent cell is located in `adj_cell_coverage_t` is used to calculate the starting offset of inter-cell slots, in the `schedule_per_slot_t` array (see chapter 4, Figure 4.15), associated with communication to/from the adjacent cell. Utility functions are provided to map a cell to its adjacent cells and cells to associated inter-cell slots, as shown in Figure 5.9.

5.3 Communication between SEAR and TBMAC

The SEAR protocol interfaces with TBMAC to support deterministic message transmission and reception and to receive callback notifications, such as, the delivery of an atomic broadcast, or, the

```
unsigned short get_current_cell(void) ;
void set_current_cell(unsigned short cell_id) ;
```

Fig. 5.8: Interface to determine current cell of a host

```
int find_cell_in_map(unsigned short adj_cell_id);
int find_adjacent_cells_of_cell(unsigned short cell_id, unsigned short adj_cells[MAX_ADJ]);
int map_adjacent_cell_to_offset(unsigned short adj_cell_id) ;
unsigned short map_inter_slot_to_adjacent_cell(int inter_slot_num) ;
int map_adjacent_data_inter_slot(int inter_slots[], unsigned short adj_cell_id);
int map_adjacent_control_inter_slot(int inter_slots[], unsigned short adj_cell_id);
```

Fig. 5.9: Mapping inter-cell slots to adjacent cells

absence of a transmission in a reserved slot. Both the interface to call to TBMAC and to receive notifications from TBMAC, are encapsulated in the `mediator` submodule. For example, the request to transmit on an intra-cell slot or to request TBMAC to start an atomic broadcast are invoked in SEAR using the functions shown in Figure 5.10, and are translated to the TBMAC functions, `send_pkt` and `atomic_broadcast` respectively.

```
int transmit_on_intra_cell_slot(int slot_num) ;
int start_atomic_broadcast(AdditionalInfo * additInfo) ;
```

Fig. 5.10: Interface from SEAR to the `mediator` submodule

To receive notifications from TBMAC the `mediator` registers callbacks with TBMAC when the `sear` module is initialised. Notifications from TBMAC to SEAR are mediated by the `mediator` to determine what functionality within SEAR to invoke.

In addition, the `mediator` performs initial parsing of messages received from TBMAC, firstly to identify SEAR messages and secondly to determine the type of message, i.e., a data or a control message, and perform any initial processing required prior to propagating the message for servicing within `sear`. For example, a data message received may require forwarding to adjacent cells.

The role of the `mediator` in the functionality presented in the rest of this chapter will be described where appropriate.

5.4 Joining the SEAR Protocol

A request to join the system is passed to the function `notify_join_sear_request` of the `mediator`, which starts the atomic broadcast of the request marking the start of the join protocol in SEAR.

At the delivery of the atomic broadcast request the `notify_make_decision` callback, of the `mediator`, is invoked by each host and the status of the join request encapsulated in the atomic broadcast message is checked. If the status is `STATE_UNKNOWN`, the current shared metadata must be transferred to the joining host, also using an atomic broadcast. All joined hosts share the same metadata, thus, all joined hosts populate an atomic broadcast message with the shared metadata for the cell, update the status to `STATE_KNOWN` and request an atomic broadcast of this message. The rationale for this approach is to avoid additional rounds of the TBMAC protocol to reach agreement on a host to reply when all hosts share consistent metadata and can perform the metadata transfer.

The joining host stores all atomic broadcast messages received during the join protocol that update the metadata of the cell, e.g., route requests and replies, using `store_additional_info`. On the delivery of the atomic broadcast message containing shared metadata for the cell the joining host calls the function `update_shared_data_per_cell` to populate the local data structures with the received shared metadata for the cell, and `apply_additional_info` to update the metadata with the stored atomic broadcast messages received during the join protocol. Thus, the joining host and all joined hosts in the cell use the same metadata to decide on the outcome of the join protocol for the host, i.e., to allocate an intra-cell slot (using `notify_allocate_initial_slot`) or to deliver a failure notification. The join protocol finishes when the status of the join is delivered to the higher layer.

Following the join request and transfer of metadata to the joined host (which accumulate to the duration of two atomic broadcasts), the worst-case latency to join SEAR (WC_JOIN) is bound by the worst-case time-bounds to join TBMAC, i.e., to be allocated an intra-cell slot.

A joined host is eligible to participate in gateway election and to be elected as gateway. To participate in gateway election the newly joined host must have the same information as all other joined hosts on which to base an election decision. In SEAR, gateway election is based on location information piggybacked with intra-cell transmissions. The newly joined host enters a learning phase prior to its first election the duration of which is bounded by the time to receive the same location information on which to base an election decision as all other hosts in its cell. Before completing the join protocol, the newly joined host starts periodic beacons to detect changes in connectivity with adjacent cells in the desired coverage of any real-time channel admitted in its cell, i.e., to detect adaptation reductions and expansions. All joined hosts execute periodic beaconing tasks which are used to transmit and process received beacons if elected as gateway.

Both of these operations are described in more detail in the remainder of this section.

5.4.1 Start Beaconing an Adjacent Cell

To determine the time of the next beacon to each adjacent cell (and the first beacon for a newly joined host), requires access to the time at which beaconing started with the adjacent cell (available as `conn_start` in the `adj_cell_coverage_t` data structure) and the beaconing period (`BEACON_INTERVAL`). The number of beacons sent in the interval since beaconing started is calculated and used to determine the time of the next beacon, as shown in Figure 5.11.

```
RTIME interval_since_conn_start = rt_get_time_ns()  - conn_start ;
int num_beacons_in_interval =
              ulldiv( interval_since_conn_start, BEACON_INTERVAL, &remainder);
RTIME next_conn = conn_start + ((num_beacons_in_interval + 1)* BEACON_INTERVAL) ;
```

Fig. 5.11: Determine the next beacon to an adjacent cell

When the time of the next beacon with an adjacent cell is known, a periodic beaconing task is started using the `rt_make_task_periodic` call from `RTAI`, as shown in Figure 5.12, where `adj_cell_idx` identifies the offset of the adjacent cell (in `adj_cell_coverage_t`) with whom beaconing is required and the start time and periodicity of beaconing is specified.

```
status = rt_task_make_periodic( &conn_task[adj_cell_idx],
               nano2count(conn_start_times[adj_cell_idx] ) ,
               nano2count(BEACON_INTERVAL) ) ;
```

Fig. 5.12: Start a periodic beaconing task at a time in the future

Following the start of periodic beaconing with adjacent cells any changes in connectivity is detactable within a maximum bound of `BEACON_INTERVAL`.

5.4.2 Calculate First Participation in Election

A joining host must defer participation in gateway election until it has the same location information available on which to base an election decision as all other hosts in its cell. Thus, the new host enters a learning phase prior to its first election. The duration of the learning phase is bounded by the maximum time to guarantee that all hosts have received the same transmissions in a round of

the TBMAC protocol and therefore have the same location information, and is described in detail in section 5.10.4.

```
next_round_start = calculate_start_of_next_round(cycle_start_time_this_cell) ;
time_to_election = (next_round_start +  LEARNING_PERIOD) - now  ;
```

Fig. 5.13: Calculation of the first election for a joining host

From the calculated start time, an election task executes with a periodicity of `ELECTION_PERIOD`, and performs all functionality to elect gateways in a cell and is described further in section 5.8.

5.5 Request to Create a Real-Time Channel

The `routing` submodule supports the `create_channel` function. In this function, the parameters specified by the channel sender are augmented by additional parameters required for route discovery and resource reservation within the desired coverage of a channel and are mapped to a `request_msg_t` message. For example, `discovery_start_time` is set to the current time returned from `rt_get_time_ns()` from `RTAI` and `cell_channel_sender` is set to the return from `get_current_cell()` to identify the current cell of the channel sender. Finally the request is atomically broadcast by TBMAC using `request_slot_allocation()` to reach distributed agreement on the slots to reserve for the channel sender (using `allocate_specific_slot()`) and to perform all processing of the request in the sender's cell, e.g., to forward to adjacent cells based on the specified desired coverage. The processing of a request message, i.e., reserving intra-cell and inter-cell slots and forwarding to adjacent cells if necessary, is identical in all cells on a route in the desired coverage.

5.5.1 Processing a Request to Create a Channel

At the deadline of the `request_slot_allocation` atomic broadcast, the message (encapsulating the create channel request) is delivered to the `mediator` where the message is parsed and the function `process_route_request` of the `routing` submodule is called to process it. All joined hosts in a cell perform the same processing of a received request message.

Start Time-Bounded Route Discovery

Using the parameters of the received request message the allowable discovery latency remaining is calculated, as discussed in chapter 4, section 4.2.2. The `end_discovery` time is calculated using

the fields `discovery_start_time` and `discovery_duration`. The `latency_to_here` is calculated as the elapsed time since `discovery_start`, and the `latency_from_here` is calculated as the latency to back-propagate a reply from the current cell to the channel sender using the route to this cell.

Using the `end_discovery` time, `latency_to_here` and `latency_from_here` the remaining discovery time for a channel request (from the current cell) can be calculated and is stored in the `channels_admitted` array as `discovery_time_rem`.

To support concurrent channel creation requests in cells in overlapping desired coverage areas, a `route_discovery` array containing metadata relating to concurrent route discoveries, is maintained by each joined host in a cell. The size of the array is limited by the maximum concurrent number of channel creation requests admitted in a cell. When a new request is received in a cell, the bounded `route_discovery` array is searched to find a free position to store the discovery metadata relating to the request.[2] If a free entry is found, a time-bounded route discovery for the request is started, using a `route_discovery_task` real-time task, with the calculated `discovery_time_rem` as the discovery time bound.

Each discovery task waits on a `RTAI` condition variable for the bounded remaining discovery time as shown in Figure 5.14. The notification from the route discovery task at the discovery deadline is the signal for actual coverage aggregation to be performed prior to the back-propagation of a route reply to the channel sender.

```
rt_cond_timedwait(&cond, &mtx, start_time + nano2count(timer_attributes[idx].disc_time));
```

Fig. 5.14: Time-bounded wait for a route discovery task

Schedule Intra-cell and Inter-cell Slots

To include a cell, say cell 2, in the actual coverage of a real-time channel an intra-cell slot should be reserved for transmissions on the channel in cell 2. The latency of transmissions on channels can be calculated based on the intra-cell and inter-cell slots reserved and the latency to transmit using these slots in cells on routes in the actual coverage. The scheduling policy has been discussed in chapter 4, section 4.2.2, and in summary the scheduled slot is the slot that minimises the transmission latency, (where the latency encompasses the delay to transmit and the transmission in a slot), when the transmission reaches a cell, e.g., from the channel sender to cell 2.

A calculation of the future time at which a transmission on the channel will reach cell 2 is available

[2] An error is raised to notify the channel sender if a free position cannot be found.

using the supplied start_time, period and msg_latency_to_here fields in a request message. The intra-cell slot with access to the wireless medium at this future time will minimise the latency for transmissions in the cell, and is calculated as shown in Figure 5.15, where now is the local time returned from rt_get_time_ns() of RTAI, and first_slot is the slot with access to the wireless medium when the transmission reaches cell 2.

```
RTIME first_execution = start_time + period  + msg_latency_to_here;
RTIME interval_to_first_execution = first_execution - now ;
num_slots = ulldiv(interval_to_first_execution, CFP_SLOT_TIME, &remainder) ;
first_slot = num_slots % SCHEDUABLE_SLOTS ;
```

Fig. 5.15: Scheduling intra-cell slots

If the first_slot is free for transmission in cell 2, the slot_num is returned to the routing submodule at this stage. Otherwise each intra-cell slot starting from this slot is checked to locate an intra-cell slot to reserve for transmissions on the channel. The greater the distance from the first_slot, the greater the delay to retransmit messages received on intra-cell slots in the cell, and thus, the greater the transmission latency incurred to reach the cell.

If an intra-cell slot is not free it may be possible to share transmissions on a new channel with transmissions on an existing channel. The check_if_slot_eligible function in the scheduling submodule first checks if an additional user of the slot is possible based on the bounds for slot sharing, i.e., the value of MAX_SHARE. The checks for slot availability and eligibility are performed sequentially for each slot. If sharing of a slot is possible, the next step is to determine if sharing is allowed given the semantics of the required slot usage. Using the metadata stored for the existing channel in the schedule_per_slot_t array and the start_time, period and msg_latency_to_here included with the new request message, slot sharing is allowed if periodic transmissions starting at the specified start_time can never overlap with existing transmissions in the slot.[3] If the intra-cell slot can be shared, the slot_num is returned to the routing module, otherwise, the next intra-cell slot is checked.

If all intra-cell slots have been checked and none are eligible, the routing submodule is notified of the failure to reserve a slot for transmissions on the channel in the cell. Thus, the channel creation request is not satisfiable in this cell (cell 2), and the status of coverage for the cell is set to NACK (not covered). The coverage status is used in actual coverage aggregation for the cell and included in the back-propagation of a reply containing the actual coverage from cell 2, which commences with

[3]To simplify the scheduling decision, a slot may only be shared amongst transmissions with the same periodicity.

an atomic broadcast in cell 2, and continues with the transmission of the reply to the cell from which the route request was received.[4]

The reservation of an intra-cell slot for a channel means the cell (cell 2) is in the actual coverage of the channel and all joined hosts in the cell have updated the metadata for the cell accordingly, e.g., by updates to the `current_channel_coverage_t`, `channels_admitted_t` and `schedule_per_slot_t` data structures.

The next phase of request message processing is to schedule inter-cell slots that satisfy the remaining delivery latency, `msg_latency_rem`, for transmissions to each adjacent cell in the `desired_coverage`. Scheduling inter-cell slots is performed using the `find_inter_cell_schedule` of the `scheduling` submodule, as shown in Figure 5.16, where the same functionality to find an `eligible` inter-cell slot is performed.

```
find_inter_cell_schedule(adj_cells_in_dc,num_adj,rt_request->period,
    rt_request->start_time, rt_request->msg_latency_to_here + latency_for_slot,
    rt_request->msg_latency_rem) ;
```

Fig. 5.16: Scheduling the inter-cell slots for an adjacent cell

where `latency_for_slot` is the latency to retransmit in an intra-cell slot in the cell and `msg_latency_rem`, is the remaining `delivery_latency` calculated from this cell.

Intra-cell and inter-cell slot scheduling is not performed in parallel as the incremental latency (which includes the latency to retransmit in a cell) is used for scheduling (both intra-cell and inter-cell slots) to satisfy the delivery latency of a channel creation request. Thus, the latency incurred to retransmit using an intra-cell slot in a cell reduces the remaining delivery latency for transmissions from the cell to adjacent cells on routes in the actual coverage. A possible consequence of this is that the number of possible cells in the actual coverage of a channel may be reduced.

Forward a Channel Creation Request

Following the reservation of inter-cell slots, the channel creation request is updated to reflect the latencies incurred in a forwarding cell to transmit on intra-cell and inter-cell slots to adjacent cells in the desired coverage of the request. The function `update_request_for_forwarding` is called within the `routing` submodule to update the `request_msg_t` parameters relating to the delivery latency, as shown in Figure 5.17.

[4] If cell 2 is the channel sender's cell, the delivery of the atomic broadcast to the channel sender includes the notification of the actual coverage.

```
rt_request->msg_latency_rem = rt_request->msg_latency_rem
          - slot_to_schedule.latency_for_slot;
rt_request->msg_latency_to_here = rt_request->msg_latency_to_here
         + slot_to_schedule.latency_for_slot ;
rt_request->forwarding_cell_id = get_current_cell() ;
```

Fig. 5.17: Updated delivery latency for forwarding to adjacent cells

The updated request message is forwarded to an adjacent cell by an elected gateway who maps the adjacent cell identifier to the control inter-cell slot for transmissions to the adjacent cell and invokes `transmit_on_inter_cell` to forward the updated request message to that cell, using the interfaces shown in Figure 5.18, where `full_msg` is a reference to a TBMAC message encapsulating an updated channel creation request.

```
inter_cell_slot_count = map_adjacent_to_control_inter_cell(adjacent_cells_in_dc[i],
                                          inter_cell_slots) ;
transmit_on_inter_cell_slot(control_slot , &full_msg) ;
```

Fig. 5.18: Forward a request on a control inter-cell slot

5.6 Back Propagation of Actual Coverage

At the expiration of the time bound for route discovery, a gateway back-propagates the actual coverage of a channel by transmitting a reply message on a control slot to the cell from which the associated channel creation request was received. A gateway in the receiving cell starts an atomic broadcast for the reply message which is delivered to all hosts in the cell to synchronise their updates to the data structure `channel_coverage_per_cell_t` based on the reply. This data structure is used as the basis for actual coverage aggregation for the cell.

The discovery latency in a cell is based on the latency of a request to reach the cell and back-propagate a reply from the cell on a route to the channel sender. Thus, the furthest cell on a route has the least time bound for discovery, and starts the back propagation of the actual coverage to the channel sender. Implicitly the discovery time bound of a successor cell on a route will expire prior to its predecessor's, thus the predecessor includes replies from its successor in its actual coverage aggregation, as shown in chapter 4, section 4.2.2.

5.6.1 Actual Coverage Aggregation

The first phase in the back propagation of the actual coverage of a channel is to create a `reply_msg_t` message that is populated with the metadata maintained about the channel in the `channels_admitted_t` data structure. The next phase is to determine the actual coverage of the channel based on the contents of the `channel_coverage_per_cell_t` array, populated by replies from adjacent cells.

The `this_cell` field of the `channel_coverage_per_cell_t` array attributes a coverage status to the cell, let's say, cell 2, in which aggregation is being performed. The status identified by `this_cell` determines the extent of further actual coverage aggregation necessary for cell 2. If `this_cell` is `DACK`, a request to create the channel had already been processed in cell 2 from an alternative route. If `this_cell` is `NACK`, no intra-cell slot is available in cell 2. In both cases the `reply_msg_t` is assigned the relevant coverage status for `this_cell` and the back propagation of the `reply_msg_t` is started to the cell from which the original request was received.

Completely different processing is required if `this_cell` is `ACK`, which means that an intra-cell slot has been reserved for transmissions on the channel in cell 2, and the actual coverage of the channel may include adjacent cells if inter-cell slots have also been reserved for the channel. In this case the actual coverage is aggregated from the status of cell 2 and reply messages received from these adjacent cells using `process_this_cell_adjacent_coverage()`. The aggregation of actual coverage, to compare the coverage status of adjacent cells on routes in the actual coverage of a channel, has been described in chapter 4, section 4.2.2.

The outcome of the aggregation process is a reply message with the actual coverage of cells on routes from this cell (cell 2) and cells in the desired coverage but not in the actual coverage. The population of the reply message is shown in Figure 5.19, where `rt_reply` is a pointer to a `reply_msg_t` message, and `num_covered` and `num_not_covered` represent the current count of covered cells in the actual coverage and cells in the desired coverage but not in the actual coverage, respectively. All details are populated from the stored `channel_coverage_per_cell_t` data structure based on replies received from each adjacent cell.

```
rt_reply->coverage_for_cell.covered_cells[rt_reply->num_covered++] =
current_channel_coverage[idx].coverage_per_adj_cell[ack_idx[i]].covered_cells[j];
rt_reply->coverage_for_cell.not_covered_cells[rt_reply->num_not_covered++] =
current_channel_coverage[idx].coverage_per_adj_cell[ack_idx[i]].not_covered_cells[j];
```

Fig. 5.19: Populating the `reply_msg_t` message with aggregate actual coverage

If an adjacent cell replied with a `DACK` or `NACK` status it cannot guarantee the delivery deadlines

of transmissions on the channel on this route to it. In this case, the inter-cell slots reserved for transmissions to the adjacent cell are removed, using the `remove_current_slot` function shown in Figure 5.20, and are available for subsequent channel creation requests received.

```
inter_cell_data_slot =
  find_data_inter_cell_for_channel(adj_cells_of_cell[i],channel_id,channel_sender_id);
if(inter_cell_data_slot > -1 )
  remove_current_slot_schedule(channel_id,channel_sender_id,inter_cell_data_slot);
```

Fig. 5.20: Removing inter-cell slot resources

5.6.2 Back Propagation to Channel Sender

The back propagation of the aggregate actual coverage terminates at the channel sender. The first step is to determine whether the cell of the channel sender is the current cell, let's say cell 2. If so, the channel sender, (and all other joined hosts in cell 2), have been notified of the expiration of route discovery by the time-bounded route discovery task, and have performed actual coverage aggregation. The actual coverage is delivered to the channel sender using one of the callbacks specified by the channel sender in the call to `create_channel`, i.e., `reserve_status` or `adapt_notification`, depending on the whether the actual coverage encompasses the complete desired coverage or not.

If the channel sender is not in cell 2 the route reply message is propagated on an inter-cell control slot to the adjacent cell, let's say, cell 1, from which the channel creation request was received, i.e., the predecessor cell on the route. The hosts in cell 1 perform the same checks to determine the location of the cell of the channel sender. The back propagation continues until the cell of the channel sender is reached and a reply message is available to the channel sender, to deliver to the higher layer.

5.6.3 Beacon Depending on Status of Adjacent Cells

An adaptation of the actual coverage implies that a change in the connectivity status with an adjacent cell has occurred. To detect changes in connectivity, adjacent cells with either `ACK` or `NACK` status are included in the `adj_coverage_per_cell_t` data structure where the `channel_id` and related `coverage_status` are updated to identify the connectivity available with the adjacent cells. An adaptation reduction occurs if an adjacent cell with a stored `ACK` status is no longer within contact. An adaptation expansion occurs if an adjacent cell with a stored `NACK` status becomes contactable. `start_conn_beaconing` of the `connectivity_manager` submodule, is invoked to start periodic beacon to adjacent cells and to detect a change in the connectivity with the adjacent cell.

5.7 Transmit on a Real-time Channel

To transmit on a real-time channel a channel sender invokes the `transmit` function and specifies the channel identifier of the channel and the data to transmit. The transmission starts in the cell of the channel sender using `find_intra_slot_allocated_for_channel()`, to locate the reserved intra-cell slot for the channel, and specifying this slot for the transmission, `transmit_on_intra_cell_slot()`, as shown in Figure 5.21.

```
slot_num = find_intra_slot_allocated_for_channel(channel_id, getMacAddr() ) ;
status = transmit_on_intra_cell_slot( slot_num , &full_msg ) ;
```

Fig. 5.21: Transmission on a reserved intra-cell slot

where `slot_num` is the identifier of the intra-cell slot reserved for the specified `channel_id` for the channel sender retrieved using `getMacAddr()`. Transmissions using reserved intra-cell slots in all further cells in the actual coverage are performed in the same way.

Other hosts in the cell of the channel sender receive the real-time transmission which is notified to the `mediator` submodule using the `packet_arrival` callback from TBMAC. The `mediator` identifies the message received as a data message and decides if the actual coverage of the channel on which the message was transmitted includes any adjacent cells to whom the data message should be forwarded. If there are, and this host is a gateway, the inter-cell slots reserved for transmissions to adjacent cells must be identified and the message forwarded as described in the next section.

5.7.1 Propagation within the Actual Coverage

Using the `channel_id` supplied, the reserved inter-cell data slots for the channel are located, as shown in Figure 5.22, where `rt_data` is a pointer to a data structure that encapsulates the message, `inter_slot_nums` is a reference to an array to be populated with the identifiers of inter-cell slots reserved for the specified channel and `inter_cell_slot_count` is the number of inter-cell slots retrieved. Multiple adjacent cells may be included in the actual coverage and there must be an inter-cell slot reserved for each adjacent cell.

```
inter_cell_slot_count = find_all_data_inter_cells_for_channel(rt_data->channel_id,
                rt_data->channel_sender_id, inter_slot_nums) ;
```

Fig. 5.22: Locating all inter-cell data slots reserved for a channel

The received message is transmitted to each adjacent cell (by the elected gateway for the adjacent cell) using the reserved inter-cell slots for the cell. Transmission on an inter-cell slot is shown in Figure 5.23, where `full_msg` is a TBMAC message that encapsulates a `rt_data_t` message for the transmission of data messages. This function is called for each inter-cell slot (identified by `i`) which is reserved for adjacent cells in the actual coverage.

```
transmit_on_inter_cell_slot(inter_slot_nums[i], full_msg) ;
```

Fig. 5.23: Transmission on an inter-cell slot

The transmission is received by a gateway in an adjacent cell who extends the transmission by retransmitting the message on an intra-cell slot reserved for the channel within the receiving cell. The `mediator` module of a receiving gateway host invokes `check_and_rebroadcast()` to determine the intra-cell slot reserved for transmissions on the channel and invokes `transmit_on_intra_cell_slot` to perform the retransmission. All hosts in the cell are notified by TBMAC, using `packet_arrival` of the existence of a message following the retransmission, but only interested joined hosts will process the message, as described in the next section.

If any further adjacent cells are included in the actual coverage the propagation of the message continues throughout the actual coverage using transmissions on reserved intra-cell and inter-cell slots.

5.7.2 Listen on a Real-Time Channel

Hosts listening on a real-time channel group that are present in the actual coverage of one of its channels for at least $T_{present}$, will deliver the message at the delivery deadline which corresponds to $t_{deliver}$ for the transmission. To listen on a channel a host invokes the `listen_on_channel` interface as shown in Figure 5.24, where the data handler to process the data received is identified by the pointer `process_data`. The data handler is added to an array of data handlers that are maintained to identify the channels on which the host requests to listen. The data handler for a channel is invoked to deliver a data message at the delivery deadline for a transmission on the channel.

```
int listen_on_channel(unsigned long long channel_id,
      unsigned long long channel_sender_id,  int (*process_data) ( rt_data_t * ))
```

Fig. 5.24: Start listening on a channel

5.7.3 Transmission Delivery at a Delivery Deadline

The `mediator` invokes `process_rt_data` of the `routing` submodule when a data message is received. Using the `channel_id` included in the `rt_data` message, the metadata relating to the channel is retrieved from the `channels_admitted` data structure and the data handler to deliver the message at the delivery deadline is located, as shown in Figure 5.25.

```
channel_offset = find_channels_admitted_using_channel_id(channel_id,channel_sender_id);
callback_idx = find_callback_handler_idx_for_channel(channel_id, channel_sender_id);
```

Fig. 5.25: Locating the metadata relating to a data message received

Using the function `store_data_for_delivery` the data message is stored for delivery at the delivery deadline and a real-time task is started which notifies the `routing` submodule at the expiration of the `delivery_delay` (see chapter 4, section 4.3), corresponding to the deadline for the transmission when the data handler for the channel is invoked to deliver the data to the higher layer (of each listening host), as shown in Figure 5.26, where `curr_callback_handler` is the data handler for the channel and the `contents` field of the `data_for_delivery` array encapsulates the data to deliver.

```
curr_callback_handler = callback_channel_handlers[task_idx].rt_data_handler ;
curr_callback_handler( (rt_data_t * ) data_for_delivery[task_idx].contents) ;
```

Fig. 5.26: Deliver the data received

5.8 Role of the Gateway

Multi-hop routing in the SEAR protocol requires a combination of intra-cell and inter-cell communication. The host responsible for transmission to and reception from adjacent cells is the gateway (see chapter 4 section 4.2.1). All joined hosts in a cell participate in a periodic election to elect the gateway(s) for the cell.

Each cell is subdivided into sectors which are threshold regions from the cell boundary with adjacent cells (see chapter 4, Figure 4.6). The criteria used for gateway election is firstly, the location of the host in the cell, or more specifically, the location of the host in relation to the threshold regions with other cells, and secondly, the mobility of the host, i.e., the estimated duration that this host will be within the threshold regions.

A gateway election is performed periodically as initiated by the periodic gateway election real-time task. To reduce the impact of frequently changing gateways, for example, changing the hosts responsible for rebroadcasting messages received on inter-cell slots, the first check during the election is to determine if the current gateways remain eligible, using `current_gateway_in_sector()`. If the previous gateway elected for a sector is still eligible, no further decisions are required for that sector. If the previous gateway is no longer eligible, a distributed decision by all hosts on the new gateway to elect is made.

To elect a host, the first step is to compare the locations of all joined hosts (available from piggy-backed location information on transmissions on allocated slots), using `compare_location_to_sector`, to the threshold regions from the cell boundaries. If a host's location is within a sector, the host is eligible to be elected as a gateway for an adjacent cell (of its cell boundary). If more than one host is eligible, the function `location_is_better` selects the host with a "better" location, i.e., closest to the midpoint of the sector with a greater distance to cross before leaving the sector.

The data structure, `elected_gateways_t`, maintains the identifiers of hosts elected as gateways, the adjacent cells each gateway is responsible for interfacing with and a `gateway_metric_t` which is the composition of the details relating to why this host was elected as gateway, i.e., the calculated distance of this host from the boundaries of the sector. The function `set_gateway_for_sector`, updates the `elected_gateways_t` array with the elected host for the relevant sector. The responsibilities of an elected gateway are described in subsequent sections.

A host must be able to determine if it is an elected gateway. The interfaces provided to a host to query if it is an elected gateway are shown in Figure 5.27, where the context of the query determines the interface to invoke. For example, to retransmit a data transmission received in the cell, a host queries whether it is the gateway to interface with the forwarding cell using `im_gateway_from_forwarding_cell`, to forward a transmission to an adjacent cell a host invokes `im_gateway_host`, and, to query if a host can transmit using an inter-cell slot, `im_gateway_for_slot`, is invoked.

```
int im_gateway_from_forwarding_cell(unsigned short forward_cell)
int im_gateway_host(unsigned short cell_id)
int im_gateway_for_slot(int inter_slot)
```

Fig. 5.27: Interfaces to query the gateway status of a host

5.8.1 Failure to Elect Gateways

At the end of the election, each joined host checks if a gateway has been elected for each sector. The failure to elect a gateway for a sector in the cell effects transmissions on channels to and from the cell. If no gateway is elected to receive transmissions on inter-cell slots and rebroadcast them in the cell, transmissions are not received in the cell. This scenario is an actual coverage reduction and detected within `BEACON_INTERVAL` by the failure to reply to beacons from forwarding adjacent cells. The actual coverage adaptation is back-propagated to the channel sender as described in section 5.9.

The failure to elect gateways to forward transmissions on channels to adjacent cells in the actual coverage is also a reduction and must be notified to the channel sender. Using the `gateways_elected_t` data structure the adjacent cell(s) under the responsibility of each gateway are known. Thus, if a gateway was not elected, the impacted adjacent cells are known and `map_adjacent_to_data_inter_slot` is used to locate the inter-cell slots reserved to forward transmissions to these adjacent cells. To start the adaptation notification, `find_channels_effected_by_conn_reduction` is called. This function determines, using the next usage of the reserved slot for a transmission, if a transmission is due before the next election (when a gateway may be available), and returns the channels effected by an actual coverage reduction caused by gateway election failure. The metadata relating to the effected channels is used to populate a reply message which includes notification of the adapted cell(s), i.e., the cell(s) for which a gateway could not be elected, and it is back-propagated to the relevant channel senders. In this scenario, the actual coverage reduction is detected within `ELECTION_PERIOD` which is the same periodicity as `BEACON_INTERVAL`.

5.8.2 Perform Intra-Cell Slot Transmissions

The absence of a transmission in an allocated slot is interpreted by TBMAC as the failure of the slot and the slot is deallocated. To avoid the erroneous deallocation of slots, the slot management module of TBMAC (residing on a slot owner) transmits NULL messages in rounds where there are no data transmissions by the slot owner.

In the SEAR implementation, intra-cell slots are reserved for transmissions by channel senders over multi-cell routes. Thus, intra-cell slots are reserved for the channel sender (slot owner) in cells in which it does not reside (remote cells). If transmissions on channels are infrequent, the slot management functionality of TBMAC in remote cells would erroneously interpret the absence of transmissions on these reserved slots as the failure of the channel sender and deallocate the slots.

To guarantee that this does not happen, gateways assume responsibility for reserved intra-cell slots in their cell if not the same cell as the channel sender. A gateway locates the channels on

which transmissions are forwarded from adjacent cells it interfaces with (predecessors cells on an actual coverage route). Following this, `find_intra_slot_allocated_for_channel` is used to locate the intra-cell slots allocated for these channels. If a transmission on the channel is expected in the next TBMAC round, there is no further action required by the gateway. If not, the gateway invokes `notify_set_send_pkt_behaviour` of TBMAC's slot management to transmit a NULL message in the next round. In the implementation the `ELECTION_PERIOD` is set to one TBMAC round. Thus, following each election, gateways check the usage of reserved slots in the next round and NULL transmissions (if required) are performed in that round only.

5.9 Actual Coverage Adaptation

The known connectivity between adjacent cells changes if an adjacent cell is in the actual coverage and connectivity is lost, or, is not in the actual coverage and connectivity is established.

A periodic beaconing task is created per adjacent cell during the initialisation of the `sear` module. However, a beacon task is only started when the status of connectivity with an adjacent cell is known, i.e., when the adjacent cell is identified as in the actual coverage (to detect future actual coverage reductions) or the desired coverage (to detect future actual coverage expansions) of some real-time channel.

A periodic beacon is transmitted as a `conn_beacon_t` message which includes the source of the beacon, i.e., the forwarding cell on a route, using `pred_cell_id`, the destination, `succ_cell_id`, and `time_sent` and `time_replied`, from source and destination respectively. To transmit a beacon to an adjacent cell the `pred_cell_id` and `time_sent` are populated by the forwarding cell, the message is wrapped in a TBMAC message and is transmitted (by the elected gateway) on an inter-cell control slot to the adjacent cell, using the interface shown in Figure 5.28, where `conn_beacon` is the populated beacon to forward and `full_msg` is the beacon encapsulated in a TBMAC message structure. The inter-cell control slots to use per adjacent cell are determined relative to the `succ_cell_id` in the beacon, and use the mapping functionality included in Figure 5.9 of section 5.2.

```
encapsulate_conn_beacon_in_full_msg(conn_beacon , &full_msg) ;
transmit_conn_beacon_to(conn_beacon->succ_cell_id, &full_msg) ;
```

Fig. 5.28: Encapsulate and transmit a beacon on an inter-cell control slot

5.9.1 Interpreting Changes in Connectivity

Connectivity beacons are transmitted from and received by gateways in adjacent cells. Upon reception of a beacon in the destination cell the `mediator` notifies the `connectivity_manager` using the `notify_process_conn_beacon` callback and passes a reference to the received beacon.

The first action upon reception of the beacon by the `connectivity_manager` is to determine the beacon context to decide if either the beacon was received from an adjacent cell and requires a reply, or, the beacon is a reply to a previous beacon transmitted from this cell. The beacon context is determined by comparing the `pred_cell_id` of the received beacon (which identifies the originating cell of the beacon) with the return from `get_current_cell()` (which identifies the current cell where the beacon was received). If the cell identifiers are not the same, the beacon was originated by a different cell and a time-bounded reply to the beaconing cell is required. If the cell identifiers are the same, the beacon was originated from this cell and this message is the reply received from an adjacent cell. The processing in the former case is to populate the `succ_cell_id` and `time_replied` and transmit a reply to the beacon on an inter-cell control slot to the adjacent cell identified in `pred_cell_id`. The processing in the latter case may detect a change in the connectivity status between adjacent cells and is detailed further.

To determine a change in the connectivity between adjacent cells, the bounded `adj_cell_coverage_t` array (populated initially with the connectivity status available from the channel creation request and updated subsequently when changes in connectivity have occurred) is searched to locate the current connectivity status of the adjacent cell, using `find_succ_cell_status`. If the status returned is `ACK`, there is no change to the connectivity between the cells. However, if the connectivity status was `NACK` the reception of the beacon reply identifies an actual coverage expansion. To process the expansion the `conn_expansion` callback of the `routing` submodule is called, as described in section 5.9.2.

Similarly, the absence of a beacon reply from an adjacent cell may mean either the actual coverage has adapted, i.e., a reduction has occurred, or, there is no change to the actual coverage as connectivity was never possible between the adjacent cells. If no reply beacon has been received and the connectivity status of the adjacent cell was previously `ACK` the absence of a reply is interpreted as a reduction of the actual coverage and the `conn_reduction` callback in the `routing` submodule is called to start the processing for an actual coverage reduction, described in section 5.9.3.

In both the expansion and reduction cases the `adj_cell_coverage_t` array is updated with the change in connectivity for the relevant adjacent cell.

5.9.2 Adaptation Expansion

The detection of newly established connectivity between adjacent cells has the potential to expand the actual coverage of channels to include further adjacent cells on a route. If the connectivity status for the adjacent cell in the `adj_cell_coverage_t` array is `NACK` for any channels the newly connected adjacent cell is in the desired coverage but not in the actual coverage of these channels. All such channels are located as shown in Figure 5.29, where `adjacent_cell_id` is the identifier of the newly connected adjacent cell, and the expansion of each channel is attempted in temporal order of channel creation.

```
int num = find_NACK_channels_with_successor(adjacent_cell_id,
             channel_ids, channel_sender_ids, MAX_CHANNELS_ADMITTED) ;
```

Fig. 5.29: Locate all channels effected by the connectivity change

The gateway in the cell where the change in connectivity was detected, let's say cell 2, populates a `request_msg_t` message, using metadata retrieved from the `channels_admitted` data structure, to request the expansion of each channel potentially effected by the change in connectivity. The `request_msg_t` message includes `EXP` in the `request_reason` field of the message to differentiate this request from a `DISC` request propagated during initial channel creation. The created request message is atomically broadcast by the gateway in cell 2.

At the delivery of the atomic broadcast, each joined host uses the `request_reason` field to determine the processing required, which in this case is the invocation of `process_expansion_request()`. Using the metadata maintained for the channel, including the `msg_latency_rem`, `msg_latency_to_here` and `discovery_time_rem` (which is the expansion discovery time bound), a new `request_msg_t` is created.

The actual coverage of a channel is expanded if the delivery constraint of the original channel creation request (`msg_latency_rem` and `msg_latency_to_here` updated to reflect the transmission latency on a route to the current cell) is satisfiable, i.e., both an inter-cell slot to forward to the newly connected adjacent cell, say cell 3, and an intra-cell slot in that cell satisfy the delivery constraints of the request.

Similarly to route discovery during channel creation a route discovery task is started by each joined host to notify its `routing` submodule when the time bound for route expansion (`discovery_time_rem`) has completed. The `start_time` of the created request is set to the time of the next transmission using the reserved intra-cell slot for transmissions on the channel in cell 2.

The reservation of an inter-cell slot for transmissions on the channel to the adjacent cell (cell 3) is the same as for the initial create channel request. If an inter-cell slot is reserved, each joined host invokes `update_coverage_expansion_list()`, to store the channel identifier of the channel that may expand. The outcome of the expansion is only known when a reply is received from the adjacent (expanding) cell. The `request_reason` is set to `DISC` for transmission to cell 3 where the processing of the received message is as per the initial request to create a channel.

The back propagation process for a reply message for an expanding channel is the same as for the initial create channel request, with actual coverage aggregation performed in each new cell, populated in a reply message which is back-propagated to the cell (cell 2) in which the expansion adaptation was detected. In this cell, the `channel_id` of the reply message received is compared to the channel identifiers stored in the expansion list. If found, the `reply_reason` is updated to `EXP` to differentiate a reply from an adaptation from a reply from initial route discovery. The back propagation of the modified reply continues to the channel sender, where the `EXP` value means that the `adapt_notification` callback is invoked to deliver the notification.

5.9.3 Adaptation Reduction

The failure to receive a beacon from a previously connected adjacent cell is notified to the `routing` submodule of the gateway, using the callback `conn_reduction` with the cell identifier of the adapted cell supplied. The gateway populates a reply message with the identification of the cell `adapted_cell_id` and the specification of `RED` in the `reply_reason` to identify a reduction. This reply message is atomically broadcast by the gateway in its cell.

At the delivery of the atomic broadcast, all joined hosts invoke the `process_route_reply` function to process the message. The message is parsed using the `reply_reason` field and the function `process_adjacent_cell_reduction` called. If the adapted adjacent cell was included in the actual coverage of any channel, the relevant channel sender must be notified (adaptation notification) of the reduction of the actual coverage. To locate the channels effected by the change in actual coverage, the function `find_channels_effected_by_reduction` is called, where all channels with reservations on the inter-cell slots with the adjacent cell are retrieved, as shown in Figure 5.30, where `rt_reply` is a pointer to the populated reply message, `adapted_cell_id` is the cell that is the subject of the adaptation and `effected_slot_nums` are the relevant inter-cell slots to communicate with the adjacent cell.

A reply message is created for each effected channel and includes an adapted actual coverage reflecting the change in connectivity. The `reply_reason` is set to `RED` and the `adapted_cell_id` is populated.

```
int effected_slot_count =
find_data_inter_slots_for_adjacent_cell(rt_reply->adapted_cell_id, effected_slot_nums);
```

Fig. 5.30: Locate all inter cell slots for the adapted cell

Additional fields, e.g., `channel_id`, `cell_channel_sender`, that assist in the back propagation of the reply message to a channel sender are included from details stored in the `channels_admitted` array and the back propagation of the message to the channel sender is started by the gateway interfacing with adjacent cells on a route to the channel sender. Reserved inter-cell slots for each channel are removed throughout the old actual coverage using a soft state approach and the adjacent cell coverage status in `adj_coverage_t` is updated to `NACK` for the adapted adjacent cell.

5.10 Failure Management

Failure suspicion in the SEAR protocol is started with a notification from the TBMAC protocol using the `slot_not_used` callback, as shown in Figure 5.31, of the absence of a transmission in a slot (`slot_num`). Given the characteristics of the wireless medium it is possible that some hosts, e.g., in the same region of a cell, do not receive a transmission in a reserved slot. The MAC layer of each of these hosts makes an autonomous decision to invoke the `slot_not_used` callback. The notification from the MAC layer raises a *suspicion* of host failure to the SEAR protocol.

```
void slot_not_used(int slot_num) ;
```

Fig. 5.31: Callback to notify SEAR of failure to transmit in a slot

5.10.1 Raising a Failure Suspicion

A host, say host A, that receives a notification from TBMAC of `slot_not_used` determines the context of the notification to decide whether a suspicion of failure should be raised or not. The first step is to determine the host responsible for transmissions in the slot, i.e., the channel sender or gateway, using `check_user_responsible_for_slot()`, and determine the next scheduled transmission using the slot, using `check_scheduled_transmission_for_slot()`. If a transmission was expected but not received by host A the host responsible for transmissions in the identified slot, say host B, is under suspicion

by host A. Host A invokes `query_suspicious_host()` to locate the metadata relating to host B which is used to populate an atomically broadcast query by the suspicious host (host A).

A host who received a transmission by the host under suspicion (host B) in the round in which the suspicion is raised, disagrees with the suspicion, described in section 5.10.2. There is a known time bound within which a disagreement to a suspicion would be received by the suspicious host. The suspicious host starts a real-time task, `task_failure_suspicion`, to notify it of the expiration of this time bound. If a disagreement is received within the time bound, the task is preempted, otherwise the failure to receive a disagreement is interpreted as an implicit agreement with the suspicion and a host failure has been detected.

5.10.2 Disagreeing with a Failure Suspicion

At the delivery of the atomically broadcast query message, each host autonomously processes the query to determine the context of the query for itself, e.g., is this host also suspicious of the queried host or did this host receive a transmission from the queried host. If the query is received by any host, say host C, who has not logged a previous suspicion, host C disagrees with the suspicion and invokes `disagree_suspicious_host()` to update the `current_type` of the message to `DISAGREE`, include metadata about the host under suspicion (using `update_sus_node_with_location()`) and atomically broadcast the disagreement message, which is delivered to the suspicious host at its deadline.

The location information included in the disagreement message must relate to the TBMAC round in which the suspicion was raised. Given the query is atomically broadcast and an atomic broadcast is a multiple of a round (in the implemented TBMAC) it is possible to calculate the number of elapsed rounds since the query was raised. Using the calculated round, say round i, the location information received for the specific host under suspicion for round i is obtained from a `location_log_t` array which maintains a known number of entries, N, (calculated in the next section) relating to the location information of hosts received in the previous N rounds. This location information for round i is included in the disagreement message.

It is possible that more that one host raises a disagreement for the same query. To reach agreement amongst hosts on who should transmit the disagreement requires additional rounds of the atomic broadcast protocol incurring additional overhead in terms of processing and time. However, the reception of one disagreement message is sufficient for a suspicious host to remove a suspicion. Thus, any host that disagrees may reply but the reception of one disagreement message is sufficient and all other disagreement messages are ignored reducing processing overhead.

Thus, the time bound for a suspicious host to wait for a disagreement to a raised suspicion (and

the time bound set for notification from the failure suspicion real-time task) is two atomic broadcasts, one to deliver the query to all hosts in a cell and one to deliver the disagreement to suspicious hosts.

5.10.3 Failure Detection

At the delivery of a disagreement message the suspicious host (host A), locates the metadata relating to the host previously under suspicion (host B), and updates it with the information included with the atomic broadcast, such as, the location information of the host under suspicion and the expected next usage of the slot. Following this the failure suspicion real-time task is signaled to stop.

If no disagreement message is received within a known time bound after raising a suspicion, the failure suspicion task notifies the suspicious host (using `process_suspicion()`) at the expiration of the time bound. The failure to receive a disagreement message is implicit agreement by other hosts that the host should be under suspicion. Processing the confirmed suspicion depends on the role of the host under suspicion and is described in chapter 4.

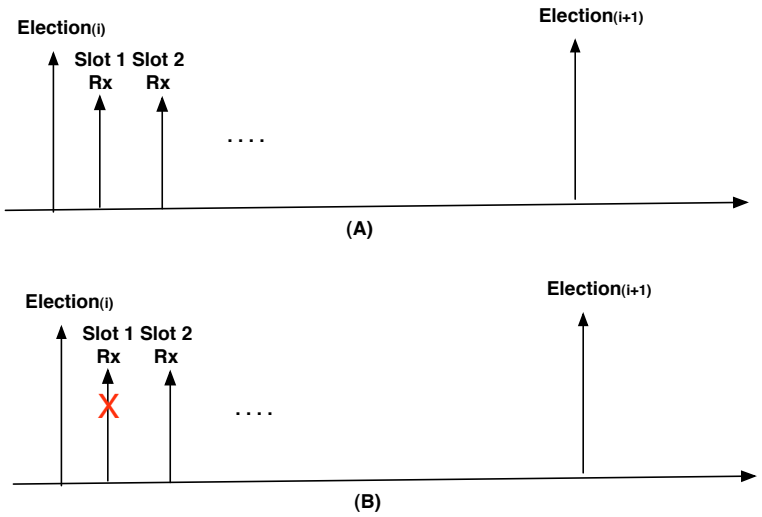


Fig. 5.32: Transient failures effect agreement in gateway election

5.10.4 Influence of Failure Suspicion on Gateway Election

To reach distributed agreement on the election of gateways requires the same information on which to base the election decision is available at each host participating in the election. Due to transient failures to receive messages in a cell a host may not receive all transmissions in the cell, and thus, may

not have the same metadata on which to base an election decision. The periodicity of gateway election relates to one round of the TBMAC protocol which is the shortest duration within which at least one transmission, with location information on which to base an election, from each joined host in a cell could be received. However, due to transient failures to receive messages there is no guarantee that all hosts have the same location information at the end of a round. For example in Figure 5.32, host A receives transmissions on all intra-cell slots. Host B does not receive a transmission in slot 1, and thus, does not have the location information piggybacked with transmissions in slot 1 and therefore does not have the same information on which to base an election decision.

The failure suspicion protocol executes within a known time bound of two atomic broadcasts. Using this time bound it is possible to calculate the round in which the failure suspicion relates and locate the metadata related to this round, which (when received by suspicious hosts) will be used as the basis for an election decision by all hosts. Thus, it is possible to calculate the earliest time when all hosts in a cell *could* have the same location information, i.e., at the completion of the failure suspicion protocol for a suspicion raised in a previous round.

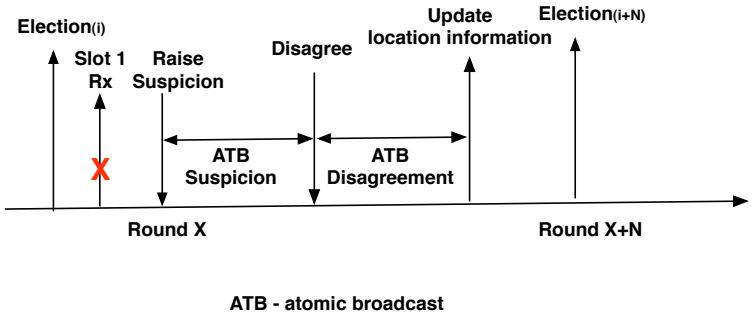


Fig. 5.33: Using location history in gateway election

For example in Figure 5.33, a failure suspicion is raised in round X. Given the known time bounds of two atomic broadcasts for suspicion and disagreement in the failure detection protocol (which are two rounds each in the implementation) the round in which the suspicion was raised (round X) can be known from the round in which the suspicion was received by other hosts in the cell. Thus, at round X + N, all hosts have the same location information (received in, and relevant for round X) on which to base an election decision. In the implementation, N encompasses two atomic broadcasts, which is four rounds of the TBMAC protocol.

The learning phase of a host prior to the first election, *LEARNING_PERIOD*, is also N to ensure

that a newly joined host has the relevant information on which to make an election decision.

The disadvantage of basing an election decision on location history is that the information could be invalidated prior to the election, i.e., due to host movement. However, an election is performed every round and any trend in the information, for example, a direction of movement, can be used in an election decision. In addition, the duration of the failure suspicion protocol limits the amount of movement that could be performed in this time interval.

5.11 Summary

This chapter described the implementation of the SEAR protocol to provide the properties of the space-elastic model over a multi-hop ad hoc network. This implementation is the basis for an evaluation of the SEAR protocol in the real world which is described in chapter 6.

Chapter 6

Evaluation

The space-elastic model provides hard real-time communication in the actual coverage of a real-time channel. The dynamics of the network may mean that the actual coverage changes but the timeliness guarantees in the remaining actual coverage are not adapted. Furthermore, time-bounded adaptation notification is guaranteed when a change in the actual coverage occurs. The objective of the evaluation of the space-elastic model described in this chapter was to determine if these properties of the model are realisable in the real world.

The space-elastic model was evaluated using the SEAR real-time ad hoc routing protocol, described in chapter 4. All the experiments described here were executed in the real world and no simulation results are included in the evaluation. The rationale for pursuing a real-world evaluation of the space-elastic model, and of the SEAR protocol, was to obtain results from a typical, uncontrolled ad hoc environment where external factors, such as wireless interference, influence the results obtained. Evaluations performed in MANETs typically rely on simulations, with ns2 (NS2 2003) being a popular choice of network simulator. However studies show that results obtained from real-world evaluation have a closer accuracy to actual usage scenarios than the execution of experiments in a controlled ns2 environment (Gray et al. 2004, Gaertner et al. 2004). The execution of a real-world evaluation does have a drawback as the range of possible scenarios addressed is necessarily more limited. However, the purpose of this evaluation was to establish the feasibility of the model and the real-world scenarios considered in this chapter are sufficient for that purpose.

The goal of the evaluation was to establish if the properties of the space-elastic model are feasible in a real-world, ad hoc scenario. The first suite of experiments evaluate the achievability of time-bounded transmissions in a multi-hop ad hoc network, i.e., the timeliness of transmissions on real-time channels.

The second suite of experiments evaluate the ability to provide time-bounded adaptation notification within a multi-hop ad hoc network, i.e., the timeliness of notifications of adaptations to the actual coverage of some channel.

Throughout this chapter, a channel sender is stationary, thus, the desired coverage is absolute. Receivers may be either stationary or mobile. This evaluation does not include the timeliness of the actions taken by a channel sender based on feedback from the SEAR protocol, e.g., behaviour adaptations based on adaptation notifications received.

This chapter starts with a discussion of the experimental environment and configuration of the parameters of the TBMAC protocol. Following this the two suites of experiments are described and the results obtained are analysed. The chapter finishes with some remarks and a summary of the results obtained.

6.1 Experimental Setup

The evaluation of the space-elastic model, using the SEAR protocol, was performed using four notebooks equipped with Lucent ORiNOCO 11Mbit/s Gold PCMCIA cards executing with real-time RTorinoco network drivers that interface with the RT-WLAN kernel modules (as described in chapter 5, section 5.1.2). The TBMAC protocol is a loadable kernel module, which interfaces with RT-WLAN, and the SEAR protocol, also a loadable kernel module, interfaces with TBMAC. The computers used to execute the experiments were four Pentium M notebooks at 2 Ghz and with 512MB RAM. Each notebook ran an identical version of the Red Hat Linux 7.3 operating system with the kernel 2.4.20 patch and the RTAI release version 3.0 applied.

6.1.1 Cellular Structure

A virtual cellular structure, required by TBMAC, was achieved by assigning notebooks different default IEEE 802.11 channels. There are eleven IEEE 802.11 wireless channels available, with each channel spaced 5MHz from its neighbour, e.g., channel 1 is 2412Mhz, channel 2 is 2417Mhz etc. The IEEE 802.11 signal is 22Mhz in width (IEEE 2005). Thus, to guarantee no interference between wireless transmissions of adjacent cells a gap of more than 22Mhz between channels is required. Thus, the three channels: 1 (2412Mhz), 6 (2437Mhz) and 11 (2463Mhz) were used.

Intra-cell transmissions are performed using the default channel assigned to the notebook. Inter-cell transmissions are performed by changing to the channel of the destination cell prior to transmitting.

The notebooks shared a view of a cell map in which the relationships between cells were defined.

6.1.2 TBMAC Parameters

The execution of the TBMAC protocol relies on the setting of two essential parameters: the slot size and the number of CFP and CP slots in a round.

Slot size In the TBMAC protocol a slot (or the slot time) is a logical representation of a known and bounded period of time, the duration of which represents the maximum interval within which a specific host, i.e., the host allocated use of the slot, may transmit.

In the single cell implementation of TBMAC, the dominant factor in the selection of the slot size is the propagation delay for a transmission of a message of a specific size. All slots are the same size and there are no inter-cell slots. The inclusion of inter-cell slots requires an additional component in the calculation of the slot size - the time to change wireless channel prior to transmitting using the slot.

The latency to change to a specific channel is measured from the call to the real-time driver function to change the wireless channel (`change_channel()` as discussed in chapter 5, section 5.1.1), to the reception of a signal from the real-time kernel module `RTorinoco` (discussed in chapter 5, section 5.1.2) that the channel has been changed. The change channel latency was tested over 100 iterations for each wireless card. The maximum latency incurred by each wireless card is shown in Table 6.1.

Wireless Card	Max. Latency (μs)	Std. Dev (μs)
1	47173	15
2	44624	14
3	45767	8
4	45929	25

Table 6.1: Maximum latency and standard deviation of changing wireless channel

The maximum latency to change channel was 47173μs and was factored into the calculation of the size of inter-cell slots. The maximum size of a message in SEAR is 1576 bytes which equates to a 8086μs propagation delay and is also a factor in the selection of the slot size.

The slot size for all CFP and CP slots was 60ms to encapsulate the latency to change channel, the propagation delay of a message, and other components of the slot size, such as guard space, as described in (Hughes et al. 2006). The slot size is large to accommodate the significant latency to change IEEE 802.11 channel and has an impact on the length of a round of the TBMAC protocol and subsequently the latency of real-time transmissions. However, the focus of this evaluation was to achieve predictable transmission latencies in the real world and not to minimise the latency incurred.

Number of CFP Slots and CP Slots The SEAR protocol interfaces with TBMAC to reserve intra-cell and inter-cell CFP slots for transmissions on real-time channels. Thus, the number of CFP slots available bounds the number of possible real-time channels available. A large number of CFP (and CP) slots increases the duration of a TBMAC round, which may be quite significant given a 60ms slot size.

To synchronise transmission to an adjacent cell with reception in that cell, the hosts in the adjacent cell must listen on the default channel assigned to the cell. Two inter-cell slots are allocated for transmission and reception for each adjacent cell, to achieve this synchronisation. Using the channel separation prescribed by the IEEE 802.11 standard, there are at most two adjacent cells, thus, the minimum number of inter-cell slots needed is 8, with one inter-cell slot for data transmission, one inter-cell for data reception, one inter-cell slot for control transmission and one inter-cell slot for control reception, for each adjacent cell.

Throughout this evaluation there are 12 CFP slots, of which 8 are inter-cell slots and 4 are intra-cell slots consisting of 2 slots for control transmissions and 2 slots for data transmissions. In addition, there are 2 CP slots. All slots are the same size of 60ms. First impressions may be that intra-cell slots do not change channel prior to transmission, and thus, could be significantly smaller (reflecting the propagation delay of 8086μs and the guard space only). However, to transmit on an intra-cell slot following transmission on an inter-cell slot necessitates a change to the default wireless channel assigned to the cell prior to the intra-cell transmission. The dispersal of intra- and inter-cell slots in a round may mean that a request to change channel is required in each slot (if intra-cell and inter-cell slots alternate). Thus, to facilitate this allocation of intra-cell and inter-cell slots the sizes of all slots include the change channel latency.

A round of the TBMAC protocol therefore consists of 14 slots each of 60ms, which is an interval of 840ms. The duration of an atomic broadcast in TBMAC is two rounds, i.e., 1680ms. Given the large slot size, the round length and atomic broadcast length are also large and will impact the duration of all transmissions. However, the objective of this evaluation was to show that predictable transmission latency is supported in the real-world and not to minimise latency.

6.1.3 Clock Synchronisation

The single-cell implementation of TBMAC uses a simple master-slave approach to distributed clock synchronisation. Essentially, a master broadcasts its current clock time periodically[1] using its allocated

[1] Piggybacked on other transmissions.

intra-cell slots. Upon reception of a packet with such a timestamp included, each host (slave) compares its current time with the timestamp received augmented by the propagation delay to reach this host (known as the expected time)[2]. The difference, Δ, between the expected time and the current time represents the difference between the clock of the master and other hosts in the same cell. If Δ is negative the clock of the receiving host is ahead of the clock of the master, otherwise the receiving host's clock is behind the master's clock.

To reconcile the clocks of receivers with the master requires applying Δ to the receiver's clocks. If Δ is large, a large adjustment to the clock of the receiver is required. However, applying a large adjustment impacts the execution of RTAI, particularly the schedule of execution of periodic tasks. TBMAC therefore bounds the maximum adjustment to apply, δ, at 80μs. Thus, if Δ is larger than δ, δ is applied, otherwise, Δ is applied.

Distributed clock synchronisation over multiple cells extends the single-cell approach by propagating synchronisation information between adjacent cells using inter-cell slots. A gateway in each cell receives the synchronisation packet on an inter-cell slot, calculates Δ[3] and applies either Δ or δ to its clock. The gateway then piggybacks the adjusted clock time for its cell on intra-cell transmissions in its cell where Δ is calculated and either Δ or δ is applied to the clock of each receiving host in the cell. The gateway for a subsequent adjacent cell transmits the adjusted clock time on an inter-cell slot to the adjacent cell, where the same procedure to adjust the clocks of receiving hosts is performed.

6.1.4 Implementation of Multi-Cell Clock Synchronisation

The bootstrapping protocol of the single-cell implementation of TBMAC was extended for multi-cell clock synchronisation. In the multi-cell case, each host changes to its designated wireless channel, known a priori, following the first application of Δ or δ based on the clock synchronisation information from the master. At the completion of the change of the wireless channel the host is in its cell (which may be the same cell as the master). If the host is in the same cell as the master, clock synchronisation is as per the single-cell approach. If the host is in a different cell, synchronisation information reaches the cell of the host using inter-cell slots. The adjacent cell from which a host receives synchronisation information is preassigned and known to the host (initialised in the cell map discussed in chapter 5 section 5.2), and is the adjacent cell that is the shortest distance, in terms of hops, from the cell of the master. If more than one adjacent cell is the same distance from the master, the cell with the

[2]Propagation delay was calculated using experimental evaluation of the one-way latency of message transmission in a benign environment.

[3]The propagation delay in this case is the same as in the intra-cell case as the timestamp used for synchronisation is returned after the channel is changed.

lowest identifier is used.

TBMAC includes clock synchronisation information with all transmissions on inter-cell slots (while both cells remain populated and contactable). However, the first transmission by TBMAC on an inter-cell slot is when a channel creation request is received and propagation to an adjacent cell is required.[4] Thus, a significant duration, increased by distance, may elapse between bootstrapping in the cell of the master and receiving the next synchronisation message on an inter-cell slot, resulting in a large Δ (also increasing with distance). However, regardless of Δ between the source of synchronisation information[5] and the receivers in a cell, the maximum clock adjustment applied is δ, which may mean the clocks of hosts in adjacent cells are never reconciled with the master, and clock synchronisation fails.

In SEAR, the scheduling of intra-cell slots and the corresponding calculation of the remaining delivery latency (the delivery delay) in a cell during channel creation, as discussed in chapter 4 section 4.3, is relative to the current clock time in a cell. Thus, $\Delta_{request}$ (the difference in clocks between the synchronisation source and hosts in the cell when a channel creation request is received) is implicitly included in the delay to deliver for the cell which is propagated with the channel creation request to adjacent cells in the desired coverage, where scheduling decisions are made based on the remaining delivery latency.

The delay to deliver encompasses $\Delta_{request}$, thus subsequent changes to Δ during an experiment, which may be large if clock synchronisation fails between cells, have an impact on the results for jitter observed. For example, Figure 6.1A, shows the expected delivery delay from reception for delivery at a deadline. In Figure 6.1B, Δ (when the transmission is received) is greater than $\Delta_{request}$[6], thus, the delivery delay is too long and delivery is after the expected delivery deadline for the channel. In Figure 6.1C, Δ is less than $\Delta_{request}$[7], thus, the delivery delay is too short and delivery is before the expected delivery deadline for the channel.

The difference between $\Delta_{request}$ and $\Delta_{receive}$, denoted by Φ, is the difference in the actual clock adjustment when a transmission is received in a cell (from the original clock adjustment included in scheduling) and contributes to jitter in the delivery deadline in the cell from the expected delivery deadline of the channel sender (see chapter 4, section 4.3). The example in Figure 6.2 highlights this. In this scenario the desired coverage is 3 cells, cells 1,2 and 3. Clock synchronisation has failed in cell 3, i.e., Δ is much larger than the applied adjustment δ, when the channel creation request is received, and thus, the Φ values increase over the course of the run.

[4]Following the first transmission on an inter-cell slot TBMAC transmits on these slots in each round.
[5]Either the master or a gateway in an adjacent cell.
[6]The clock difference is greater.
[7]The clock difference is less.

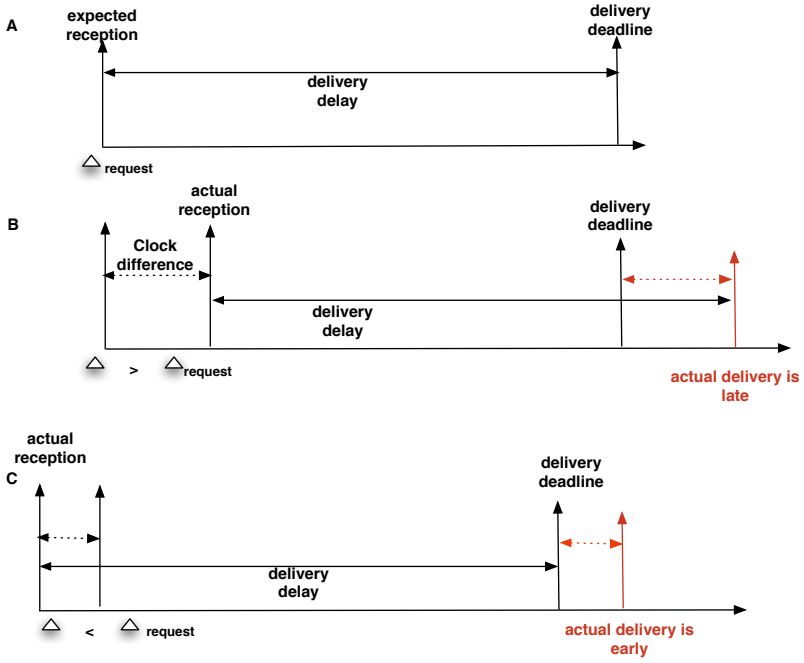


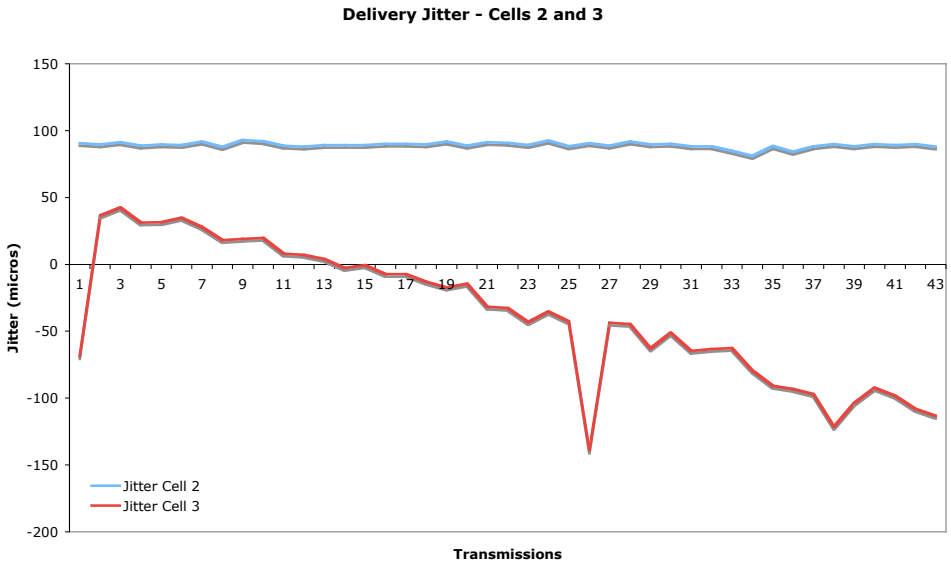
Fig. 6.1: Impact of difference in $\Delta_{request}$ and $\Delta_{receive}$

The jitter in cells 2 and 3 are shown in Figure 6.2(a) and the corresponding Φ values are shown in Figure 6.2(b).

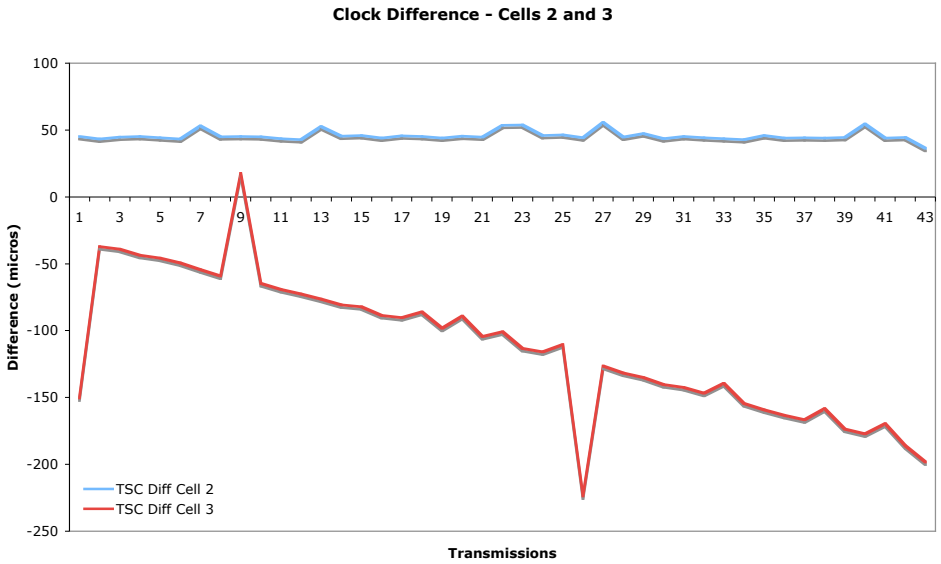
The observed jitter in cell 3 highlights the impact of Φ on the jitter incurred. It is important to note that a Φ value of 0 means $\Delta_{request}$ and the current $\Delta_{receive}$ are the same. The jitter observed up to and including transmission 25 is small and less than $\pm$ 50μs, and is related to the small values of Φ, less than 100μs in this interval. The observation here is that $\Delta_{receive}$ values may move towards or away from $\Delta_{request}$ during an experimental run.

As the experiment continues, Φ values increase, to a maximum of -224μs ($\Delta_{receive}$ moving further away from $\Delta_{request}$) and the jitter increases to a maximum of -139μs. The difference in jitter between the first and second transmissions is 68μs and is related to the difference in the Φ values, which is -113μs.

The influence of the Φ values on the observed jitter is evident in this example. One important observation is that the failure of clock synchronisation may mean that the clocks of receivers in a cell are running before or after the clock of the channel sender, with an impact on early or late transmission delivery. In this case, the clocks of receivers are running after the clock of the channel sender (negative



(a) Observed jitter cells 2 and 3



(b) Observed Φ values cells 2 and 3

Fig. 6.2: Delivery jitter and Φ values in a desired coverage of 3 cells

Φ values), and delivery, although initially before the deadline (early) happens after the deadline (late) throughout the remainder of the run.

To achieve a better understanding of the predictability of transmissions, regardless of the influence of clock synchronisation failure and the skews in jitter attributed to it, the average over a sample of 90 iterations is used in the experiments in the next section.

6.2 Evaluation of Timeliness of Transmission Delivery

Before discussing the results obtained some background to the terminology used in the analysis is provided. The configuration of cells used in all experiments is shown in Figure 6.3. The channel sender (and master) is located in cell 1. In these experiments the desired and actual coverage were always the same.

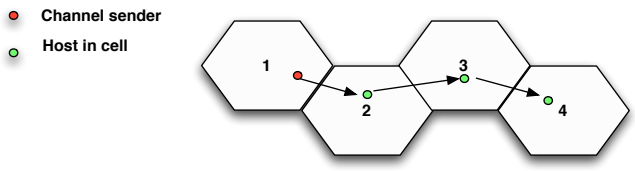


Fig. 6.3: Cell configuration used in the evaluation

The metric of interest for evaluating the timeliness of transmission delivery is jitter which is measured as the difference in the observed delivery deadline at receiving hosts from the expected delivery deadline of the channel sender (see chapter 4, section 4.3). A positive value of jitter means the observed delivery deadline is before the expected deadline, whereas a negative value means the value is after the delivery deadline. A jitter value of 0 means the observed and expected delivery deadlines are the same.

Variances in the firmware and software latencies of RT-WLAN (as shown in chapter 5, section 5.1.2) can impact the delivery deadlines for receivers and are evident in the evaluation as jitter. A greater source of observed jitter, however, is attributed to the implementation of the multi-cell version of clock synchronisation used by TBMAC as described in the previous section.

6.2.1 Measurement Process

The evaluation of the timeliness of transmission delivery was performed using a channel with a varying sized desired coverage, from 1 to 4 cells, depending on the required scope of transmissions for an experiment. For example, the desired coverage was specified as all four cells if the timeliness of transmission delivery in all four cells (i.e., a 3-hop scenario) was under evaluation.

The measurement of a transmission starts with the invocation of `transmit` by the channel sender at the specified start time and subsequently at the specified periodicity (included in the channel creation request) for transmissions on the channel. The start of a transmission is logged as a timestamp at the channel sender only. In addition the channel sender logs the expected delivery time of each message transmitted. Each receiver that delivers a message logs its delivery time. For both channel sender and receiver a unique identifier for the message is also logged. The jitter is the difference between the expected delivery time of the channel sender and the actual delivery time of receivers for each message.

In all experiments the average, worst-case, standard deviation and 99% confidence intervals are shown. The evaluation of transmission delivery for a desired coverage of varying size using one channel is performed using the average results from 3 runs of 30 transmissions. Due to the rapid failure of clock synchronisation (as discussed previously), it was not feasible to perform any more iterations of an experiment in a single run. However, using the average, worst-case and 99% confidence intervals for 90 iterations it can be illustrated that predictable transmission delivery is achievable for all receivers regardless of the desired coverage size. The worst-case is presented as an absolute value.

6.2.2 Transmission Delivery with a Desired Coverage of 1 Cell

Transmission delivery with a desired coverage of 1 cell, i.e., the single cell scenario, illustrates the jitter experienced by receivers in the cell of the channel sender (cell 1) and is shown in Figure 6.4. The worst-case jitter is low at 14μs. The maximum difference between the average and worst-case is small at 11μs. The average and standard deviation are both 3ms. The 99% confidence interval yields small bounds of (-4.74, 10.74)μs, which means that 99% of all transmissions with a desired coverage of 1 cell fall within these tight bounds, which highlights the predictability of transmission delivery in the single cell scenario.

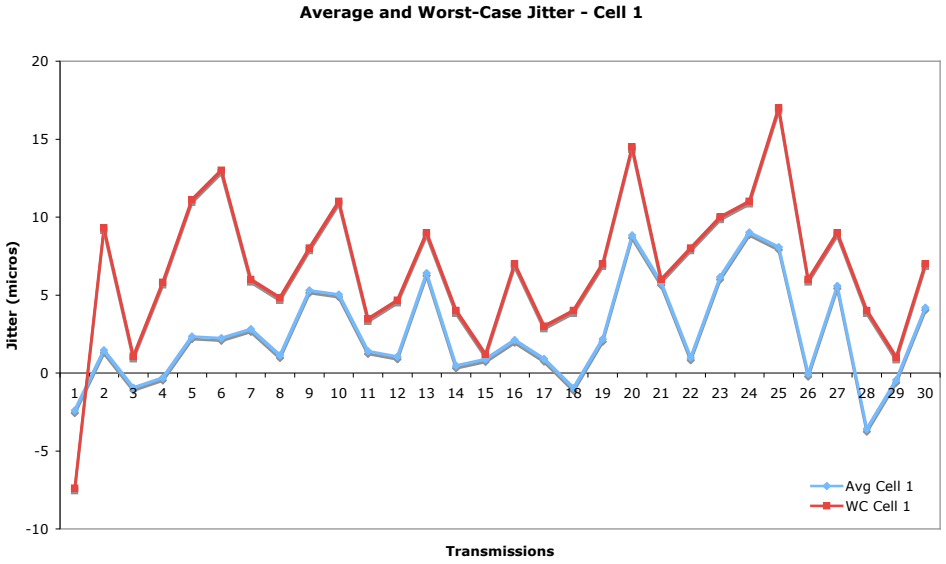


Fig. 6.4: Average and worst-case jitter with a desired coverage of 1 cell

6.2.3 Transmission Delivery with a Desired Coverage of 2 Cells

Transmission using a real-time channel with a desired coverage encompassing two cells (cell 1 and cell 2) is the minimum scenario requiring transmissions using inter-cell slots. The master is also a gateway and transmits synchronisation messages on inter-cell slots to cell 2.

The observed results for jitter in cell 2 adjacent to the cell of the channel sender are shown in Figure 6.5. The worst-case jitter is 93μs. The maximum difference between the average and worst-case jitter is 4μs. The mean and standard deviation are 88μs and 1.12μs respectively. The 99% confidence interval is between 85.11μs and 90.89μs, thus, 99% of all transmissions over a desired coverage of two cells fall within these bounds, which given the small range of the bounds shows that deterministic transmission delivery is achieved in this scenario.

6.2.4 Transmission Delivery with a Desired Coverage of 3 Cells

In the three-cell scenario, cell 2 is adjacent to the master (in cell 1) and cell 3 is adjacent to cell 2, but not adjacent to the master. This scenario requires two sources of synchronisation information, i.e., the master in cell 1, and a gateway, who transmits the adjusted clock time in cell 2 to cell 3. The results for jitter in cell 2, are shown in 6.6(a). The worst-case jitter is 93μs. The maximum difference

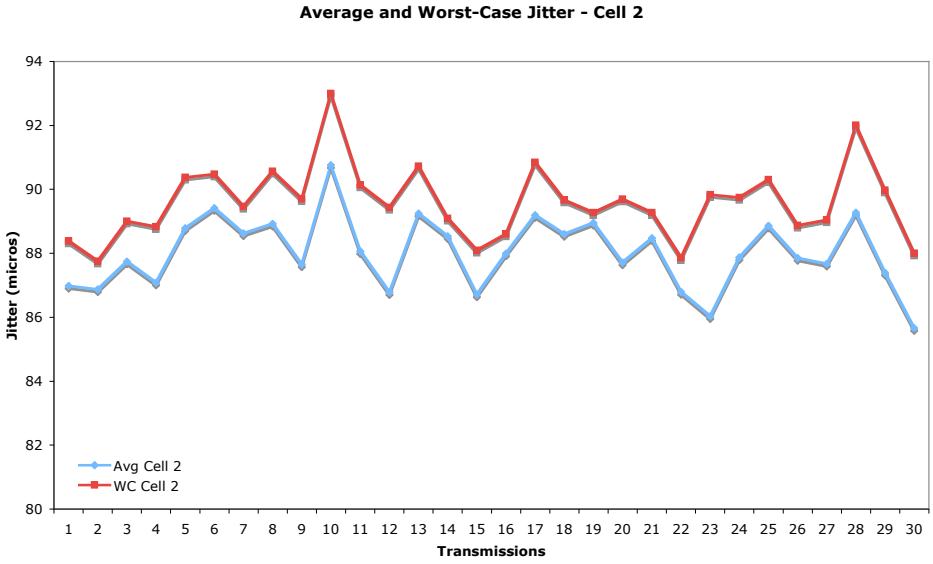


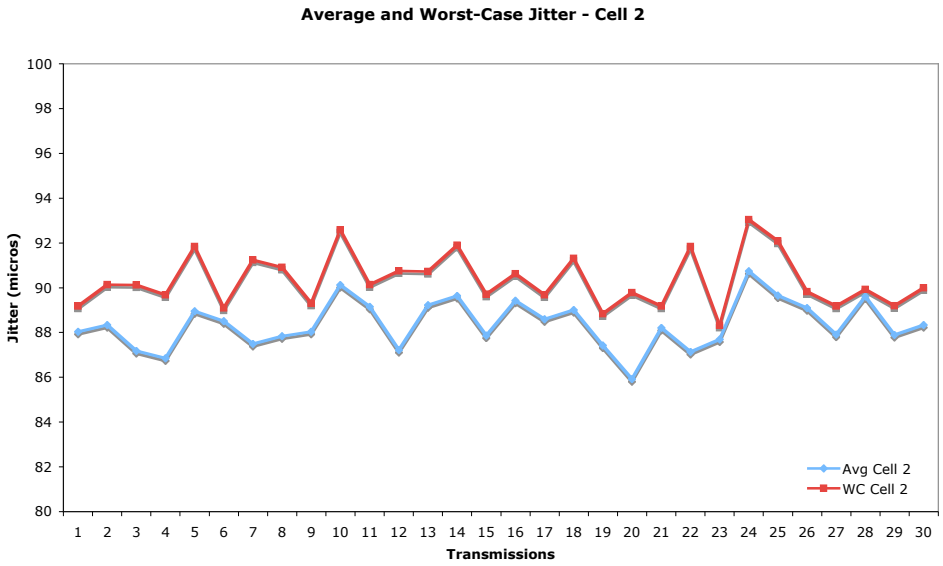
Fig. 6.5: Average and worst-case jitter in cell 2 in a desired coverage of 2 cells

between the average and worst-case jitter is 4μs. The mean and standard deviation are 88μs and 1.07μs respectively. The 99% confidence interval yields small bounds of (85.24, 90.76)μs, with 99% of all transmissions in cell 2 in a desired coverage of 3 cells falling in this range. These bounds are within 0.5μs of the 99% confidence intervals for transmissions in cell 2 in a desired coverage of two cells, and show the timeliness of transmission delivery in cell 2 regardless of the size of the desired coverage.

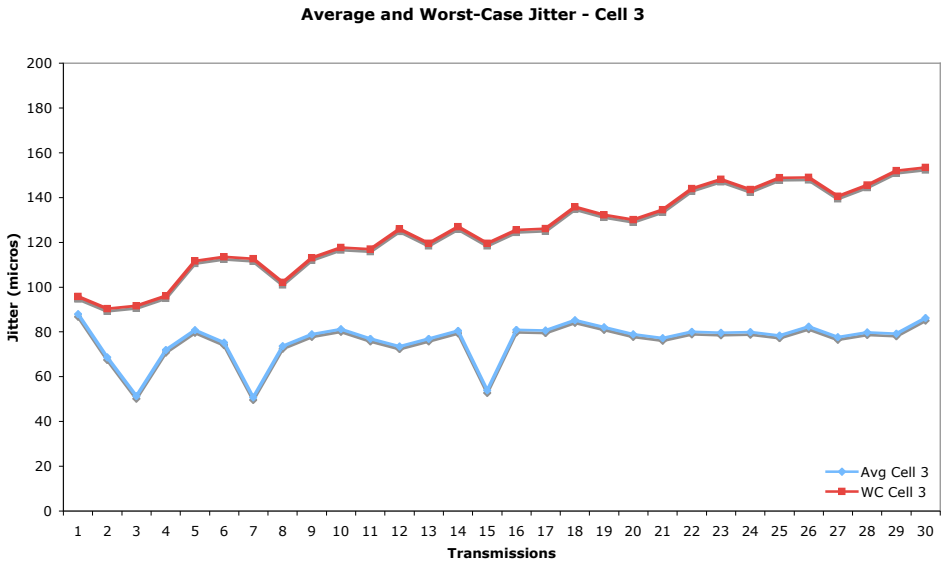
The results for jitter in cell 3 are shown in 6.6(b). The worst-case jitter is 153μs. The maximum difference between the average and worst-case bounds is 73μs. This difference is attributed to the failure of clock synchronisation in cell 3 and its influence on the observed jitter in all runs. The mean and standard deviation are 76μs and 9.15μs respectively, yielding 99% confidence bounds of 52.39μs to 99.61μs. Thus, regardless of the failure of clock synchronisation, 99% of all transmissions will be delivered in cell 3 have a small jitter that is within these bounds, highlighting that predictable transmission delivery is achieved.

6.2.5 Transmission Delivery with a Desired Coverage of 4 Cells

In the four-cell scenario, cell 2 is adjacent to the master (in cell 1), cell 3 is adjacent to cell 2 and cell 4 is adjacent to cell 3, thus, cell 4 is 3 hops away from the master.

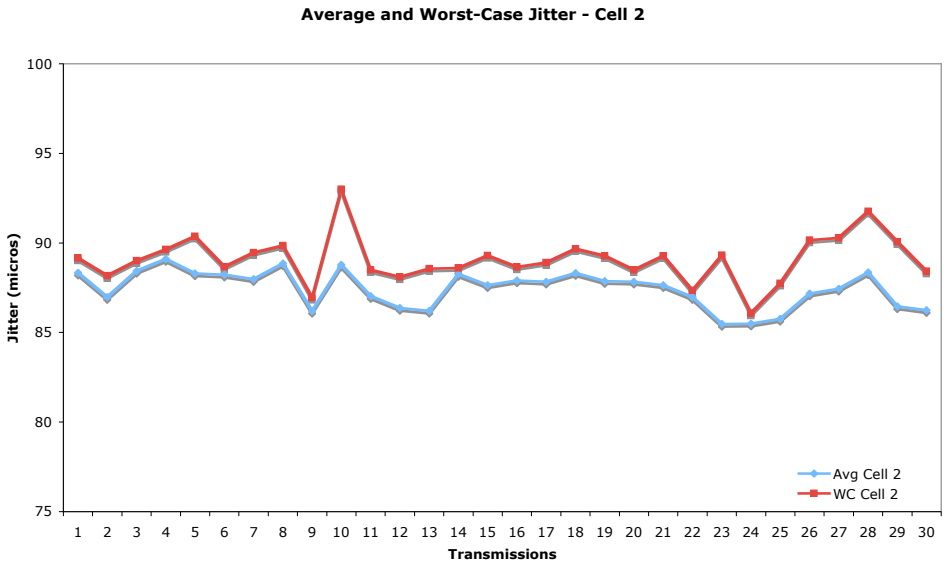


(a) Average and worst-case jitter in cell 2

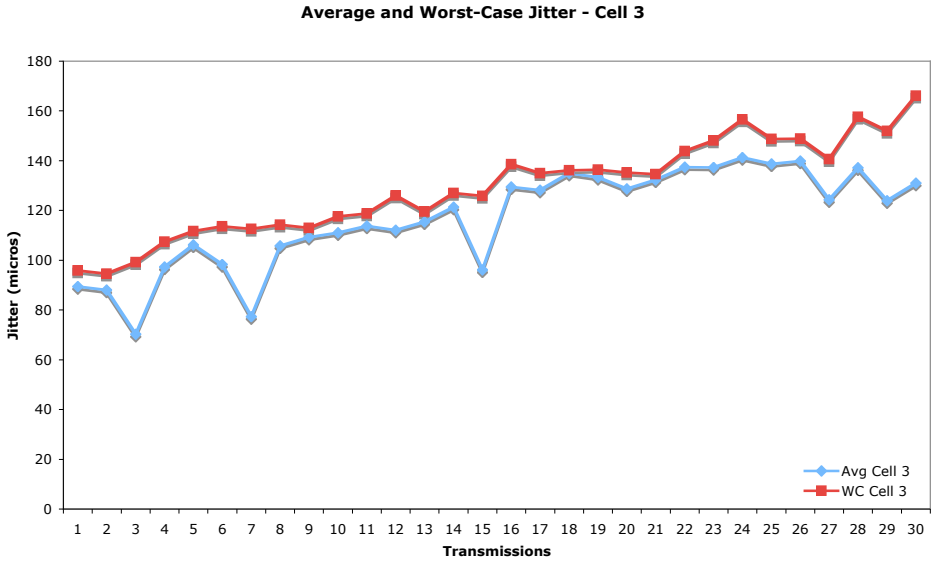


(b) Average and worst-case jitter in cell 3

Fig. 6.6: Average and worst-case jitter in cells 2 and 3 with a desired coverage of 3 cells



(a) Average and worst-case jitter in cell 2



(b) Average and worst-case jitter in cell 3

Fig. 6.7: Average and worst-case jitter in cells 2 and 3 with a desired coverage of 4 cells

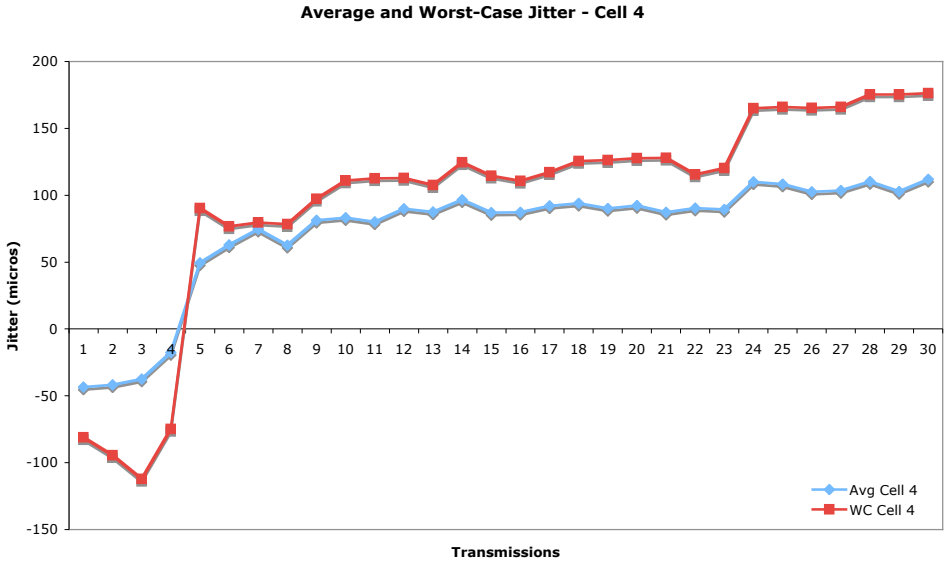


Fig. 6.8: Average and worst-case jitter in cell 4 with a desired coverage of 4 cells

The average and worst-case results observed in cells 2 and 3 are shown in Figure 6.7(a) and 6.7(b) respectively.

The worst-case jitter in cell 2 is 93μs. The maximum difference between the average and worst-case jitter is 4μs. The mean and standard deviation are 87μs and 1.03μs respectively. The 99% confidence interval yields bounds of 84.34μs and 89.66μs. All 99% upper and lower confidence interval bounds for cell 2 in desired coverages ranging from 2 to 4 cells have been within 1μs of each other, highlighting the determinism of transmission delivery in cell 2 (1-hop from the channel sender).

The worst-case jitter in cell 3 is 166μs. The maximum difference between the average and worst-case jitter is 35μs. The mean and standard deviation are 117μs and 19.73μs respectively. The 99% confidence interval gives bounds of 66.98μs to 167.9μs, which highlights the predictability of transmission delivery in cell 3 (2 hops from the channel sender). The slightly wider bounds in this scenario, (the upper bound has increased by 68μs) is again attributed to clock synchronisation failure in the cell. However, the 99% confidence bound illustrates that just 0.01% of all transmissions delivered in cell 3 in a desired coverage of 4 cells will incur jitter greater than 167.9μs, illustrating that predictable delivery is achieved in the cell.

The worst-case jitter in cell 4 is 176μs and is shown in Figure 6.8. The maximum difference between the average and worst-case jitter is 73μs. The mean and the standard deviation are 73μs and

46μs respectively. The 99% confidence intervals for transmission delivery in cell 4 (3 hops from the channel sender) are -55.7μs and 181.7μs respectively. The 99% confidence bounds are wider in this scenario due to worst-case jitter related to clock synchronisation failure in the cell. However, only 0.01% of all transmissions over 4 cells are delivered with a jitter of less than -56μs (after the expected delivery deadline) or greater than 182μs (before the delivery deadline), illustrating the timeliness of transmission delivery in all 4 cells in the desired coverage.

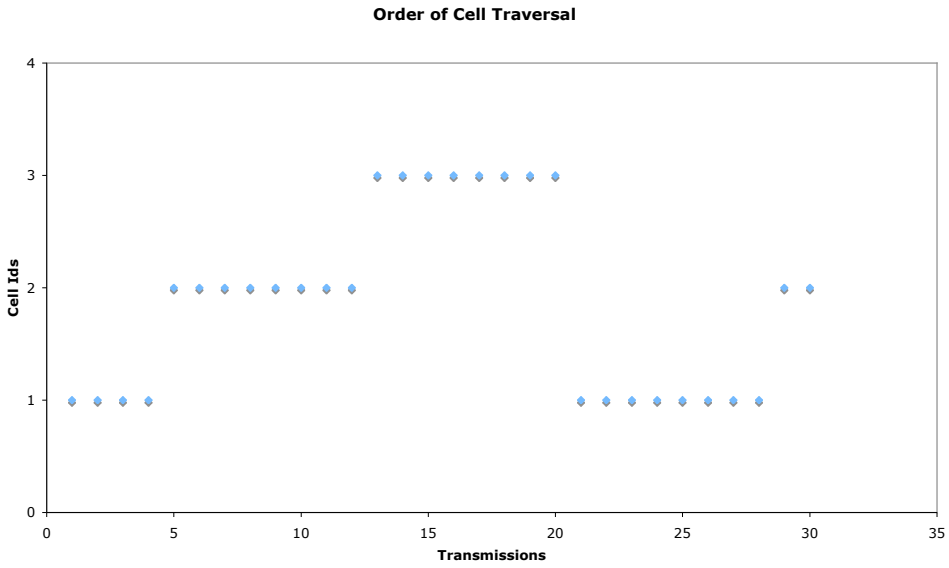


Fig. 6.9: Cell traversal by a mobile receiver

6.2.6 Transmission Delivery by a Mobile Receiver

In all scenarios presented thus far the receivers have been stationary which translates in the implementation to changing to and remaining listening on a preassigned wireless channel. The next experiment evaluates transmission delivery for a mobile receiver. A mobile receiver is a host listening on a real-time channel that moves to a different cell within the actual coverage implemented by changing wireless channel to the appropriate wireless channel of the destination cell.

The cell traversal when a transmission is received is shown in Figure 6.9. The desired coverage is 3 cells. The mobile receiver is listening on the channel prior to the first transmission on the wireless channel, and remains within the actual coverage for the duration of the experiment. The mobile

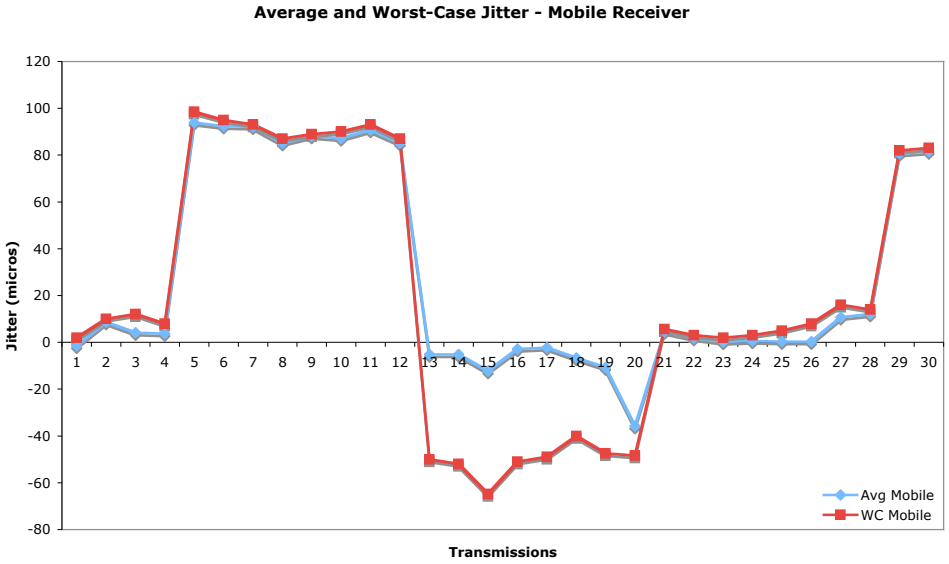


Fig. 6.10: Average and worst-case jitter - mobile receiver

receiver is a listener only, (see chapter 3, section 3.1.3).

Using the Lucent ORiNOCO cards described in section 6.1.2, the upper bound on the latency to change channel is 47173μs, with the implication that a receiver is effectively not within any cell for this interval. To avoid lost messages whilst changing wireless channel, and using the observation that messages can only be received in intra-cell slots when in a cell, a request to change channel, which may occur at any time, is deferred to, and executed during, the first inter-cell slot following the request. Thus, the mobile receiver is always listening on a wireless channel when a transmission occurs on an intra-cell slot in a cell.

The average and worst-case jitter for a mobile receiver are shown in Figure 6.10. The worst-case jitter is 98μs. The maximum difference between the average and worst-case is small at -45μs, and occurs when the mobile receiver is in cell 3. The worst-case jitter occurs in cell 2 and is 98μs, which is only 5μs greater than the worst-case jitter for stationary receivers in cell 2. The mean and standard deviation are 28μs and 43μs respectively. A 99% confidence interval yields bounds of (-82.94, 138.94)μs. Thus, 99% of transmissions delivered by a mobile receiver will be within these bounds showing the timeliness of delivery regardless of mobility.

The average and worst-case jitter for cell 2 are shown in Figure 6.11. The worst-case jitter is 93μs. The maximum difference between the average and worst-case jitter is 3μs. The mean and standard

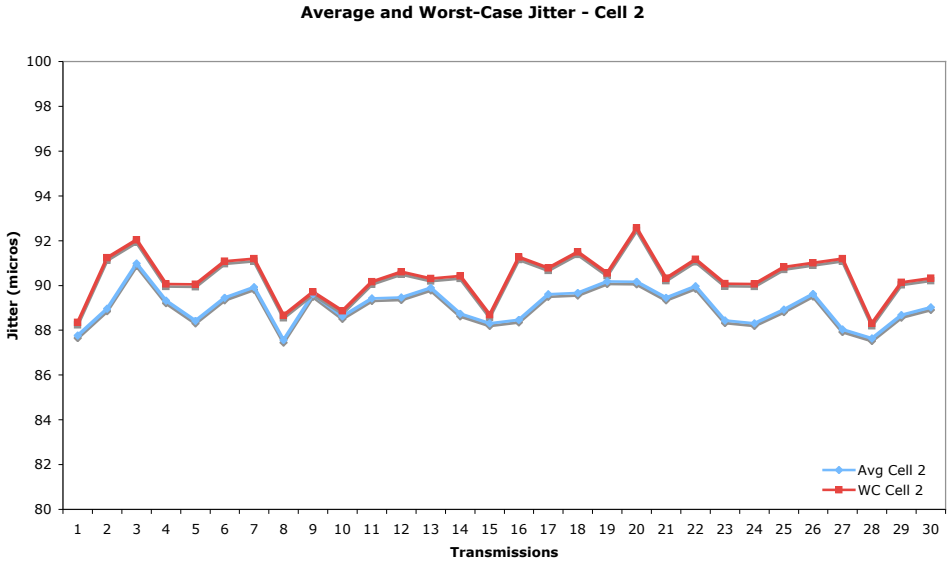


Fig. 6.11: Average and worst-case jitter in cell 2 with a desired coverage of 3 cells

deviation are 89μs and 1μs respectively. The 99% confidence interval bounds are 86.42μs and 91.58μs.

The average and worst-case jitter for cell 3 are shown in Figure 6.12. The worst-case jitter is 153μs. The maximum difference between the average and worst-case jitter in cell 3 is 73μs. The mean and standard deviation in cell 3 are 79μs and 10μs respectively, yielding a 99% confidence interval bounded by 53.2μs and 104.8μs respectively.

The jitter for both cells 2 and 3 are similar to previous results and illustrates the predictability of transmission delivery for a desired coverage of 3 cells.

6.2.7 Transmission Delivery using Simultaneous Channels

This section concludes with experiments performed to determine the timeliness of transmissions over two simultaneous channels. In this case, both channels have the same specification of desired coverage and delivery latency. Thus, slots are shared by transmissions over the channels. Each experimental run starts with a transmission on channel 1 and alternates transmissions on each channel for the remainder of the run. The requests to create the channels occur in the same slot in separate rounds.

The goal of these experiments was to show that timely transmission delivery is feasible using simultaneous channels in the same actual coverage, i.e., the observed jitter for each channel per cell

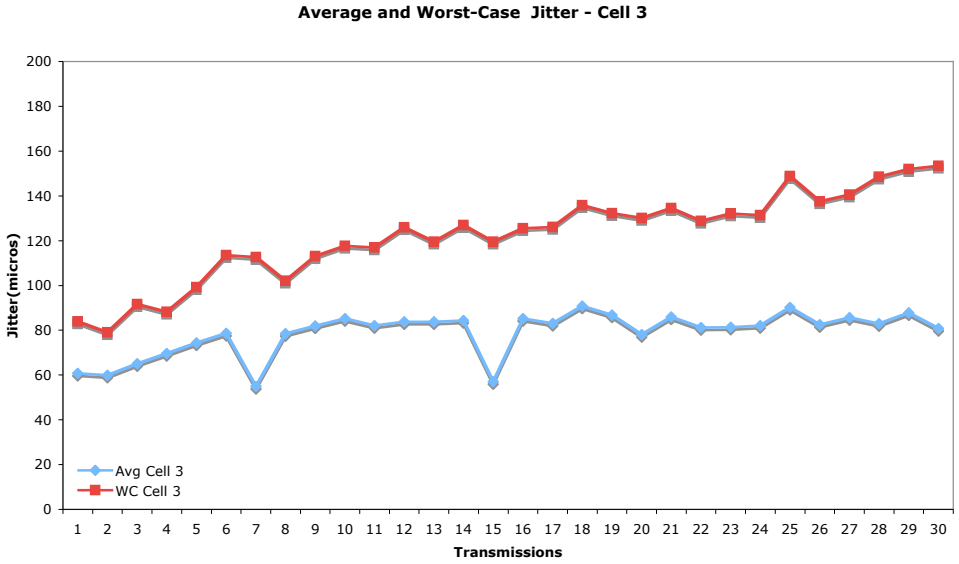


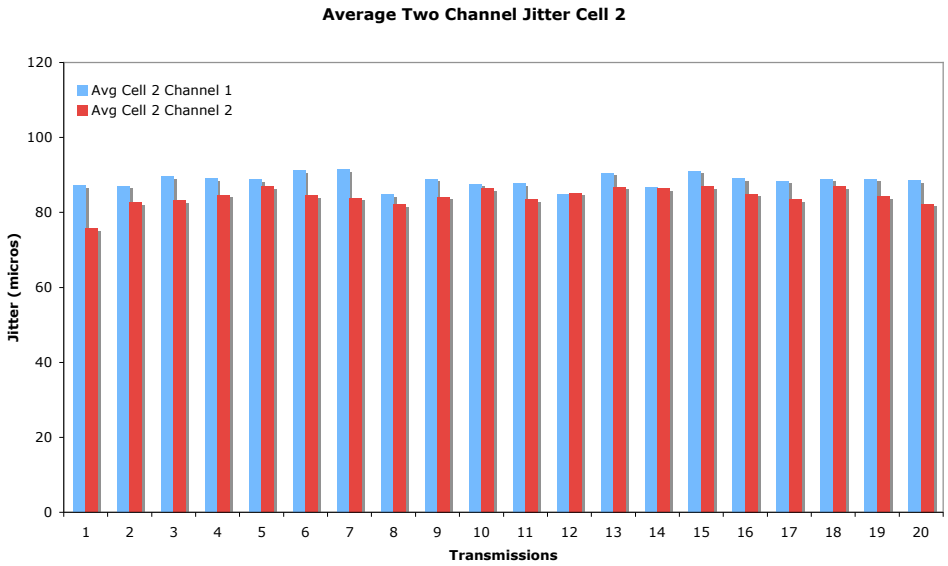
Fig. 6.12: Average and worst-case jitter in cell 3 with a desired coverage of 3 cells

should be similar to previous results and due to the similar specification of the channels, the difference in observed jitter per channel per cell should be small.

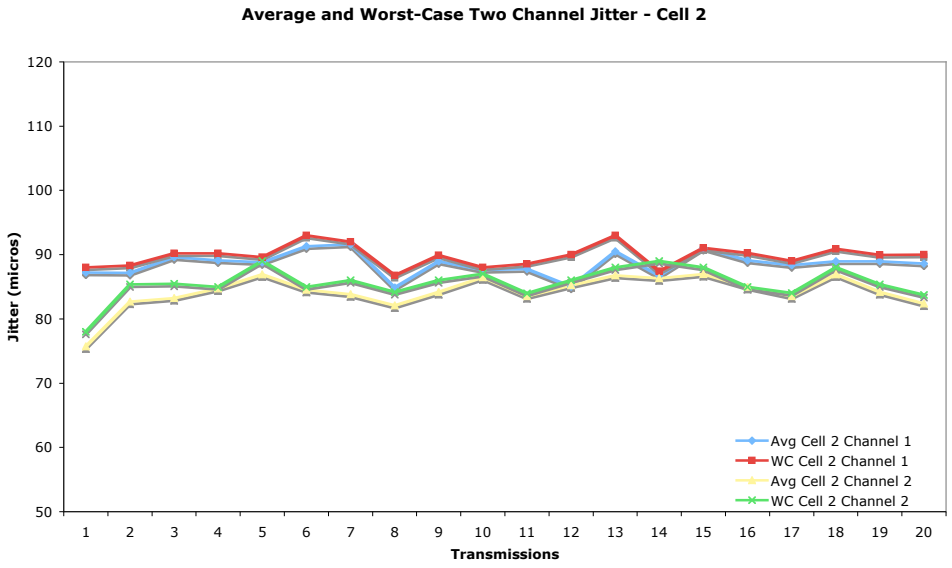
Alternating transmissions on each channel to yield the same number of transmissions on each channel extends the duration of a run. Extending the duration of a run leads to a greater influence of clock synchronisation failure on transmission delivery. Thus, in the evaluation of simultaneous channels using a desired coverage of 2 and 3 cells, the number of iterations in a run was reduced to 20. Each run was performed twice. Extending the evaluation to include simultaneous transmissions in a desired coverage of 4 cells again increases the duration of a single iteration. In this case, the evaluation of simultaneous transmissions over a desired coverage of 4 cells is performed on 10 iterations in a run, where each run is performed twice. Comparing the 99% confidence interval bounds obtained with previous results it is clear that predictable transmission delivery is achieved regardless of the presence of simultaneous channels and size of the desired coverage.

The observed average jitter for two channel transmissions in a desired coverage of two cells are shown in Figure 6.13(a). The maximum difference in the jitter between the channels is 11μs and occurs in the first transmission.

The average and worst-case jitter in the two channels are shown in Figure 6.13(b). The worst-case jitter in channel 1 is 89μs and the worst-case jitter in channel 2 is 84μs. The mean and standard



(a) Average 2 channel jitter in cell 2



(b) Average and worst-case jitter in cell 2

Fig. 6.13: Average and worst-case 2 channel jitter in cell 2 with a desired coverage of 2 cells

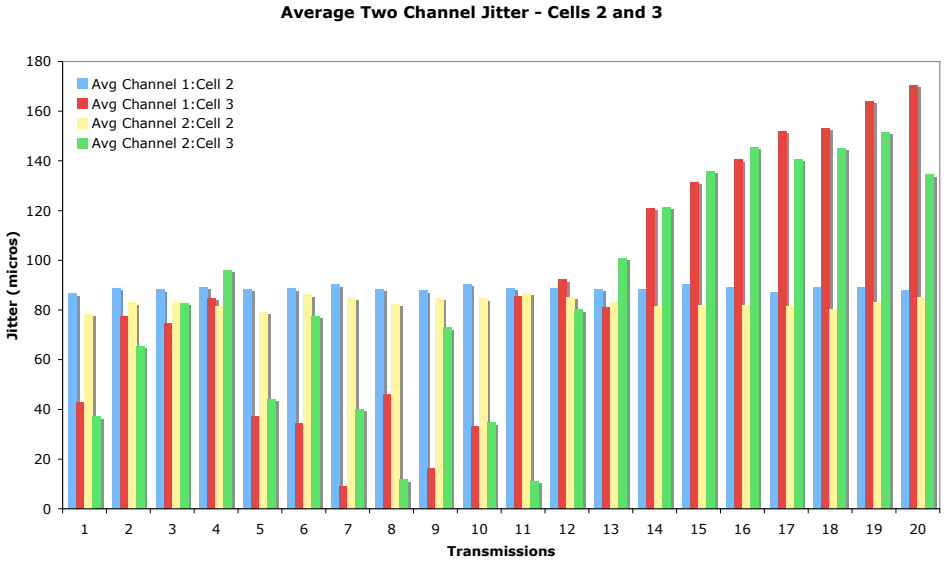


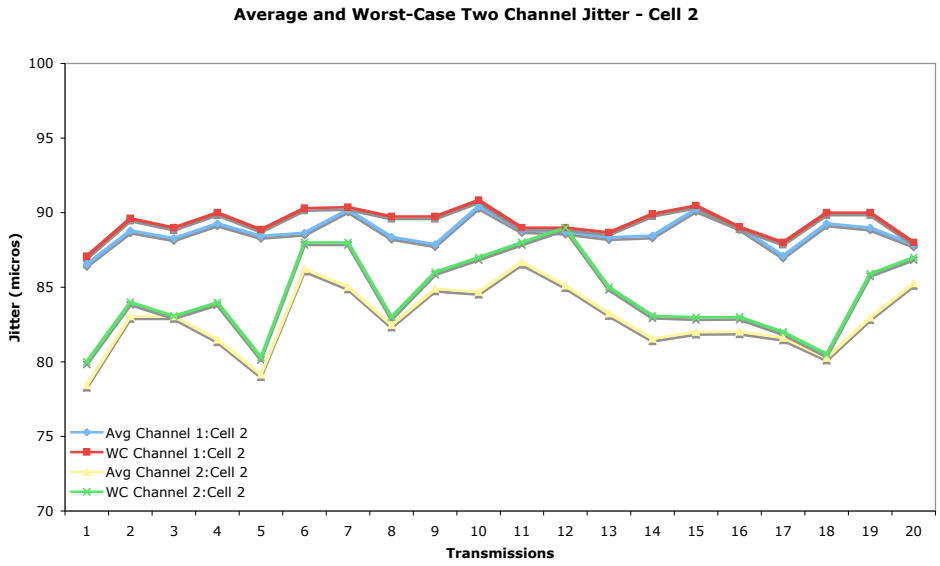
Fig. 6.14: Average jitter for 2 channels in cells 2 and 3 with a desired coverage of 3 cells

deviation for channel 1 are 89μs and 1μs respectively. The mean and standard deviation for channel 2 are 84μs and 2.6μs respectively. The 99% confidence interval bounds for channel 1 are (87.87, 90.18)μs and for channel 2 are (82.425, 85.57)μs. The 99% bounds for both channels are similar to previous results for transmission delivery in cell 2, thus, the predictability of transmissions over 2 cells regardless of the number of simultaneous transmissions is illustrated.

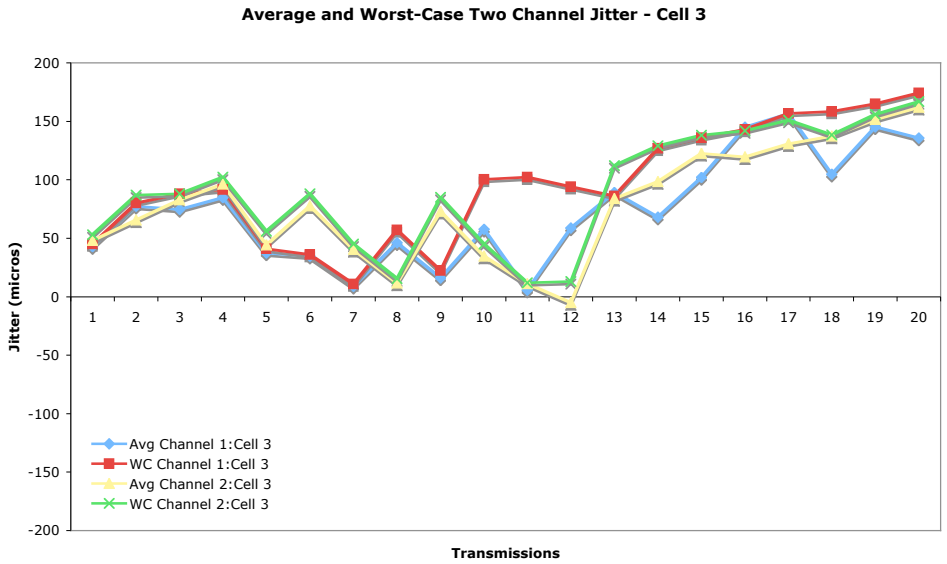
The average jitter observed for two channel transmissions with a desired coverage of three cells is shown in Figure 6.14. The observed jitter results for both channels 1 and 2, denoted by blue and yellow bars respectively, in cell 2 (adjacent to the master) are similar to all previous results where the worst-case jitter for channel 1 is 89μs and for channel 2 is 83μs. The observed jitter for both channels 1 and 2 in cell 3 are denoted by red and green bars respectively. The worst-case jitter for channel 1 is 174μs and for channel 2 is 154μs.

The average and worst-case jitter for both channels in cell 2 are shown in Figure 6.15(a). The 99% confidence interval yields bounds between 85.91μs and 92.09μs for channel 1 and 81.69μs and 84.31μs for channel 2, and are both similar to previous results for 99% of transmissions delivered in cell 2 with a desired coverage of 3 cells.

The average and worst-case jitter for both channels in cell 3 are shown in Figure 6.15(b). The 99% confidence interval bounds are (54.18, 119.82)μs for channel 1 and (56.151, 117.85)μs for channel 2.



(a) Average and worst-case jitter in cell 2



(b) Average and worst-case jitter in cell 3

Fig. 6.15: Average and worst-case 2 channel jitter in cells 2 and 3 with a desired coverage of 3 cells

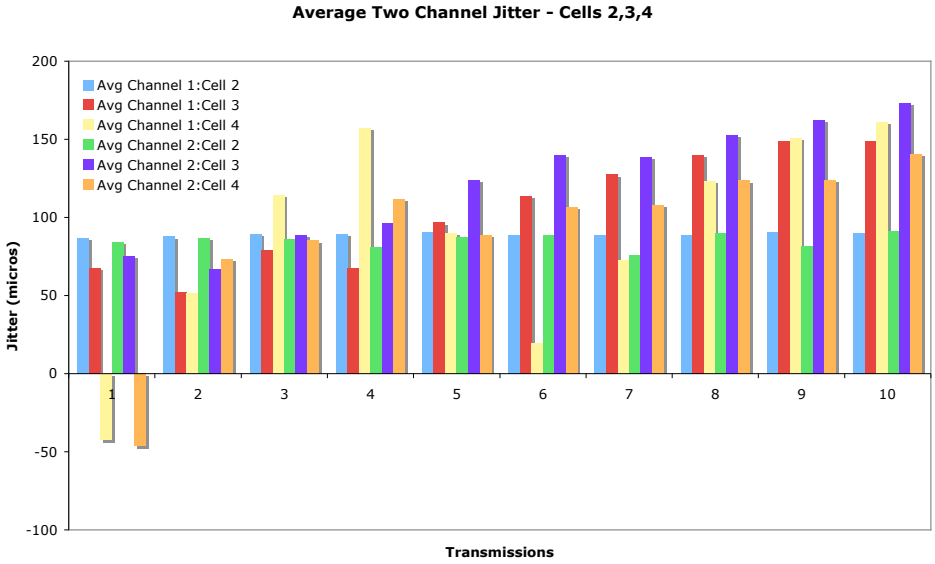


Fig. 6.16: Average jitter for 2 channels in cells 2,3 and 4 with a desired coverage of 4 cells

Both bounds are similar to previous results for 99% delivery of transmissions in cell 3.

The average jitter observed for two channel transmissions with a desired coverage of four cells is shown in Figure 6.16. The worst-case observed jitter results for channels 1 and 2, denoted by blue and green bars, for cell 2, are 92μs and 94μs respectively. The worst-case observed jitter for cell 3, denoted by red and purple bars, are 152μs and 176μs respectively. The observed worst-case jitter in cell 4, denoted by yellow and orange bars, are 167μs and 146μs for channels 1 and 2 respectively.

The average and worst-case jitter in cell 2 is shown in Figure 6.17. The mean and standard deviation in cell 2 are 89μs and 1μs for channel 1 and 85μs and 5μs for channel 2. The 99% confidence interval yields bounds between 87.92μs and 90.1μs for channel 1 and 79.58μs and 90.4μs for channel 2, which are again similar to all previous results for transmissions in cell 2.

The average and worst-case jitter in cell 3 is shown in Figure 6.18. The mean and standard deviation in cell 3 are 104μs and 36μs for channel 1 and 122μs and 38μs respectively for channel 2. The 99% confidence interval bounds for transmissions over two channels in cell 3 with a desired coverage of 4 cells are (65μs, 143μs) for channel 1 and (80.33μs, 163.17μs) for channel 2. The range of the bounds are very similar to the 99% bounds for 1 channel transmission delivery in cell 3 with a desired coverage of 4 cells.

The average and worst-case jitter in cell 4 is shown in Figure 6.19. The mean and standard

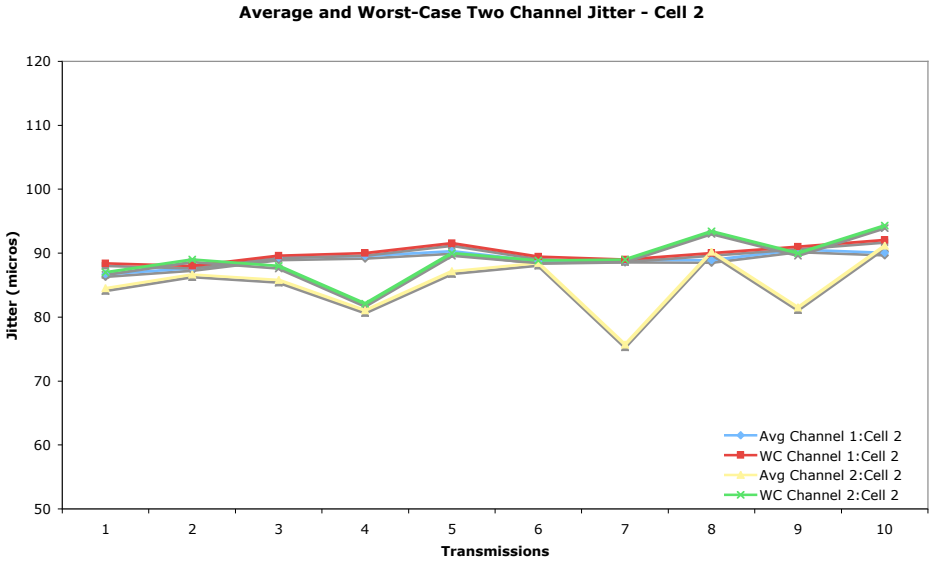


Fig. 6.17: Average and worst-case 2 channel jitter in cell 2 with a desired coverage of 4 cells

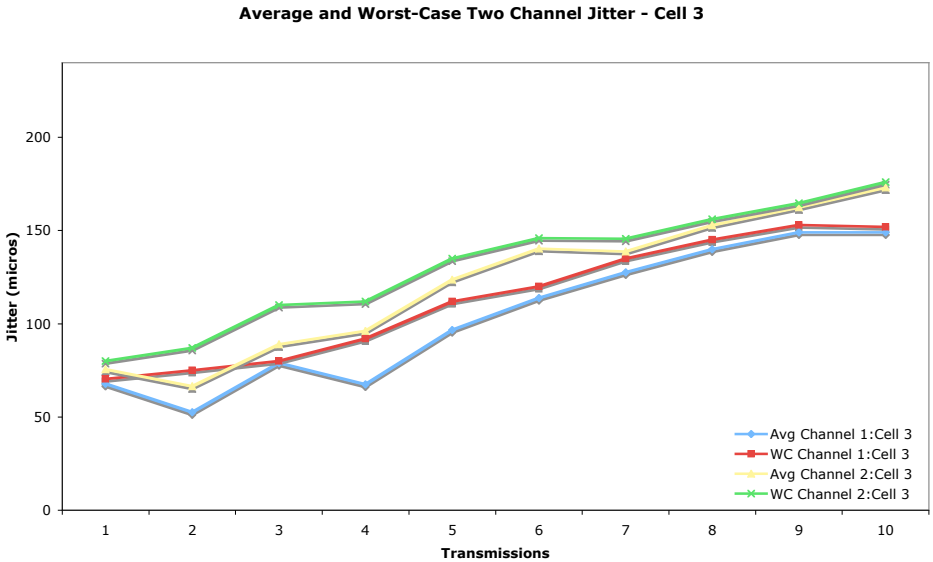


Fig. 6.18: Average and worst-case 2 channel jitter in cell 3 with a desired coverage of 4 cells

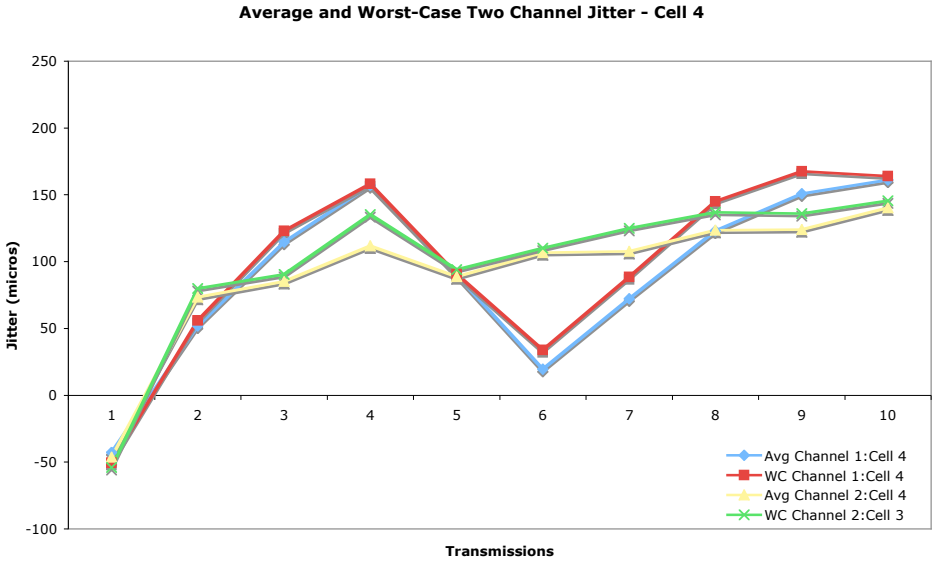


Fig. 6.19: Average and worst-case 2 channel jitter in cell 4 with a desired coverage of 4 cells

deviation in cell 4 are 90μs and 66μs for channel 1 and 92μs and 52μs for channel 2 respectively. The 99% confidence interval bounds for transmissions in cell 4 with a desired coverage of 4 cells are between 18.5μs and 161.5μs for channel 1 and between 35.67μs and 148.33μs for channel 2. Comparing these confidence interval bounds with the results from the one-channel scenario over a desired coverage of 4 cells, it can be seen that the lower bound is larger in the two-channel case. The reason for this is that in the worst-case in the one-channel scenario the Δ values are large, whereas in the two-channel scenario, although clock synchronisation has failed, the Δ values are not as large and thus, do not have such an influence on the jitter incurred.

To summarise, the results observed show that predictable transmission delivery at a delivery deadline is possible in variable sized desired coverages with a number of simultaneous channels. The jitter does increase slightly with distance from the channel sender. In this evaluation, transmission delivery in cell 4 (3 hops from the channel sender) incurred the largest jitter where the range for the jitter of 99% of all transmissions in cell 4 is 236.7μs. An influential factor on these results is the failure of clock synchronisation in the cells, the likelihood of which increases with distance, as discussed in section 6.1.4. Even with the failure of clock synchronisation it has been shown that predictable transmission delivery was achieved. The remaining experiments performed in this chapter, evaluate the timeliness of adaptation notification over varying distances.

6.3 Evaluation of the Timeliness of Adaptation Notification

A property of the space-elastic model is to guarantee timely adaptation notification to a channel sender if an adaptation of the actual coverage occurs. The goal of the experiments in this section was to determine if timely adaptation notification is feasible in the real world.

The adaptation notification time includes the latency to detect an adaptation, to back propagate the adaptation notification to the channel sender and for the channel sender to deliver the notification received to the higher layer.

6.3.1 Measurement Process

The timeliness of adaptation notification was evaluated with a varying sized desired coverage ranging from 1 to 4 cells depending on the goal of the experiment.

The measurement of an adaptation starts when the adaptation is detected which is signified by a change in connectivity between adjacent cells. The change of connectivity may have occurred at any time within the last connectivity beaconing period. Thus, the worst-case latency to detect the adaptation, identified by the periodicity of beacon exchange in chapter 4, section 4.2.3, is subtracted from the time at which the connectivity change was detected, and is logged by a gateway in the cell where the adaptation is detected. The adaptation notification is back propagated to the channel sender. The channel sender also logs a timestamp when the adaptation notification is delivered to the higher layer. The difference between the delivery time of the adaptation notification and the logged detection time of the adaptation, is the adaptation notification time.

The worst-case adaptation notification time for a reduction is delivered to the higher layer following initial channel creation. The worst-case adaptation notification time for an expansion is also available to the higher layer but is of lesser significance to the behaviour of the application, as discussed in chapter 3, section 3.1.1. The value of of the worst-case adaptation for both reduction and expansion adaptations is based on the longest route in the desired coverage and is specified in terms of rounds of the TBMAC protocol. The actual adaptation notification time varies depending on the size of the actual coverage and whether the adaptation is a reduction or an expansion. The size of the desired coverage and the type of adaptation will be specified in each experiment. To remove ambiguity the worst-case adaptation notification time (WC_ADAPT) is referred to as either WC_RED_ADAPT or WC_EXP_ADAPT depending on the context of the adaptation, i.e., either a reduction or an expansion adaptation respectively.

The duration of one run of an adaptation experiment consists of a reduction of the actual coverage

followed by an expansion of the actual coverage within the specified desired coverage, with back-propagation of each adaptation to the channel sender. Thus, the duration of one run is increased by distance as specified in the desired coverage. Due to the rapid failure of clock synchronisation (as discussed previously), the number of iterations of reduction and expansion adaptations within a run was limited. The results presented in this section are for 30 reduction and expansion adaptations for varying desired coverage specifications. All results are rounded up to the nearest millisecond. The results in nanoseconds are presented in corresponding tables in appendix A.

In all experiments the worst-case adaptation notification time returned from initial channel creation for variable sized desired coverages for both reduction and expansion adaptations is shown. This is the bound within which all observed actual coverage adaptation notifications must lie. In each experiment the mean, standard deviation and 99% confidence intervals are also shown.

6.3.2 Adaptation Reduction

The adaptation notification time was evaluated for the notification of a reduction adaptation in a desired coverage ranging from 2 to 4 cells. In the first set of reduction adaptation experiments included in this section the outermost cell in the actual coverage is no longer included in the actual coverage. In the last set of experiments, cells on the actual coverage route (other than the outermost cell) were disconnected, for example, in a desired coverage consisting of cells 1 - 4, cell 2 or 3 was disconnected.

Adaptation Reduction in a Desired Coverage of 2 Cells

In this scenario, a reduction of the actual coverage implies the cell of the channel sender, cell 1, detects the disconnection of cell 2 and becomes the only cell included in the actual coverage. The WC_RED_ADAPT returned from channel creation is 5880ms.[8] The actual adaptation notification times observed for 30 iterations of a reduction in a desired coverage of 2 cells are shown in column 2 of Table 6.2. All observed results are within WC_RED_ADAPT for a desired coverage of 2 cells. The observed mean is 5204ms, the observed worst-case is 5319ms and the standard deviation is 33ms. The 99% confidence interval bounds for adaptation notification of a reduction in a desired coverage of 2 cells are between 5119ms and 5289ms.

Adaptation Reduction in a Desired Coverage of 3 Cells

The desired coverage in this scenario consists of cells 1, 2 and 3. The change of connectivity occurs between cells 2 and 3, with cells 1 and 2 remaining in the actual coverage only. A desired coverage of

[8] 840 (Latency to start atomic broadcast) + 1680 (atomic broadcast) + 3360 (latency to detect)

Adaptation Reduction Notification			
	WC_RED_ADAPT = 5880 (DC = 2 cells)	WC_RED_ADAPT = 9240 (DC = 3 cells)	WC_RED_ADAPT = 12600 (DC= 4 cells)
No.	Adaptation notification time (DC = 2 cells)	Adaptation notification time (DC = 3 cells)	Adaptation notification time (DC = 4 cells)
1	5159	7335	10670
2	5279	7647	10329
3	5159	7447	10569
4	5279	7875	10610
5	5159	8148	10393
6	5199	7567	10490
7	5199	8088	10809
8	5199	8148	10209
9	5199	7548	10370
10	5199	8028	10509
11	5199	7548	10809
12	5199	7548	10490
13	5199	7927	10490
14	5319	7447	10569
15	5199	7567	10449
16	5199	7934	10569
17	5199	7447	10569
18	5199	7548	10370
19	5199	7927	10670
20	5199	7447	10569
21	5199	7867	10610
22	5199	7867	10509
23	5199	7488	10393
24	5199	8028	10393
25	5199	8088	10610
26	5199	7747	10610
27	5199	7687	10209
28	5199	7807	10490
29	5199	7488	10509
30	5199	7934	10369

Table 6.2: Reduction adaptation notification time in a desired coverage ranging from 2 to 4 cells

3 cells is the minimum scenario requiring the back propagation of the adapted actual coverage using inter-cell slots to the cell of the channel sender (cell 1).

The WC_RED_ADAPT returned from channel creation is 9240ms. The actual adaptation notification times observed for 30 iterations of a reduction in a desired coverage of 3 cells are shown in column 3 of Table 6.2. All observed results are within WC_RED_ADAPT for a desired coverage of 3 cells. The observed mean is 7739ms, the observed worst-case is 8148ms and the standard deviation is 249ms. The 99% confidence interval bounds for adaptation notifications in a desired coverage of 3 cells are (7097, 8381)ms. There is a 1501ms difference between WC_RED_ADAPT and the observed mean over the 30 iterations. The reason for this difference is that the worst-case adaptation notification latency is calculated in terms of rounds of the TBMAC protocol where the latencies to start inter-cell transmissions or atomic broadcasts following inter-cell reception, incur the maximum latency in each cell, which is a TBMAC round. The actual adaptation notification time will be less than the worst-case if the latency to start inter-cell transmission or atomic broadcasts is less than a round of the TBMAC protocol, which was the case in all iterations.

Adaptation Reduction in a Desired Coverage of 4 Cells

The desired coverage in this scenario consists of cells 1 2 3 and 4. The change of connectivity occurs between cells 3 and 4, leaving cells 1 - 3 in the actual coverage following the reduction.

The WC_RED_ADAPT returned from channel creation is 12600ms. The actual adaptation notification times observed for 30 iterations are shown in column 4 of Table 6.2. All observed results are within WC_RED_ADAPT for a desired coverage of 4 cells. The observed mean is 10507ms, the observed worst-case is 10809ms and standard deviation is 145ms. The 99% confidence interval bounds for adaptation notifications in a desired coverage of 4 cells are (10133, 10881)ms.

Adaptation Reduction of an Interior Cell in the Desired Coverage

In the following experiments the desired coverage was 4 cells. The objective of these experiments was to show that timely adaptation notification is achievable regardless of where the adaptation occurs within the desired coverage. The actual adaptation notification time for the adaptation of cell 2 and cell 3 in a desired coverage of 4 cells are shown in Table 6.3, columns 2 and 3 respectively. The actual adaptation notification time for the adaptation of cell 4 have already been shown in column 4 of Table 6.2. The returned WC_RED_ADAPT for the experiment is 12600ms and reflects a desired coverage of 4 cells.

All observed results for the adaptation of both cells are within WC_RED_ADAPT. The observed

Adaptation Reduction Notification		
	WC_RED_ADAPT = 12600 (DC = 4 cells)	WC_RED_ADAPT = 12600 (DC = 4 cells)
No.	Adaptation notification time (DC = 4 cells)	Adaptation notification time (DC = 4 cells)
1	5319	7488
2	5199	7387
3	5319	8088
4	5199	7327
5	5199	8028
6	5199	8208
7	5199	7728
8	5199	7867
9	5199	7548
10	5199	8148
11	5199	8047
12	5199	8047
13	5199	8148
14	5199	7548
15	5199	7867
16	5199	7728
17	5199	8208
18	5199	8028
19	5199	7327
20	5199	8088
21	5199	7387
22	5199	7488
23	5199	7728
24	5199	7927
25	5199	7548
26	5199	8028
27	5199	7934
28	5199	8148
29	5199	7387
30	5319	7807

Table 6.3: Reduction adaptation of cells 2 and 3 in a desired coverage of 4 cells

average for cell 2 is 5211ms, the worst case is 5319ms and the standard deviation is 37ms. The 99% confidence interval bounds for adaptation notifications detected by cell 1 in a desired coverage of 4 cells are within (5116, 5306)ms. The observed average for cell 3 is 7808ms, the worst-case is 8208ms and the standard deviation is 296ms. The 99% confidence interval bounds are between (7044, 8572)ms. The 99% confidence interval bounds for both cells in this experiment are very similar to previous results for notification of an actual coverage reduction. Thus, predictable adaptation notification was achieved from any cell in the desired coverage regardless of the size of the desired coverage.

6.3.3 Adaptation Expansion

The adaptation notification time was evaluated for the notification of an expansion adaptation in a desired coverage ranging from 2 to 4 cells. In the first set of experiments in this section the expansion covers the outermost cell in the desired coverage only. For example, in a desired coverage consisting of cells 1, 2 and 3, only cells 1 and 2 are in the actual coverage. The expansion is detected in cell 2 to include cell 3 in the actual coverage. In the last set of experiments the expansion starts from cells on an actual coverage route and expands to include all cells in the desired coverage. For example, cells 1 - 4 are within the desired coverage, but only cell 1 is within the actual coverage. In this case an expansion adaptation includes cells 2 - 4 in the actual coverage.

The maximum discovery latency for the desired coverage is specified in the channel creation request in each experiment. The maximum discovery latency is the maximum time bound for a request to propagate to all cells in the desired coverage and for a reply to be received by the channel sender. For example, the maximum discovery latency for a desired coverage of all 4 cells is as follows:

$$Request = \underbrace{840 + 1680 + 840}_{latency\ ATB\ +\ ATB\ +\ latency\ forward} \times 3 + \overbrace{840 + 1680}^{latency\ ATB\ +\ ATB} = 12600ms$$

which is the latency to start an atomic broadcast for the channel request, the atomic broadcast duration and the latency to propagate to an adjacent cell, in cells 1 - 3, and the latency to start and execute the atomic broadcast in cell 4.

The maximum time bound for a reply to reach the channel sender (in cell 1) is:

Desired Coverage Size (cells)	Discovery Latency (ms)
2	9240
3	15960
4	22680

Table 6.4: Discovery latency for varying desired coverage specifications

$$Reply = \underbrace{840}_{to\ cell\ 3} + \overbrace{840 + 1680 + 840}^{latency\ ATB\ +\ ATB\ +\ latency\ forward} \times 2 + \underbrace{840 + 1680}_{latency\ ATB\ +\ ATB} = 10080ms$$

where the back-propagation of a reply starts in cell 4 following the atomic broadcast for the request and traverses cells 3 and 2, where the latency is again the latency to start and execute an atomic broadcast and back-propagate the reply, and cell 1 where the latency to start and execute an atomic broadcast are the only latencies incurred.

Thus, the worst-case discovery latency over 4 cells is 22680ms. The discovery latency and desired coverage used in the evaluation is shown in Table 6.4. A desired coverage of two cells is the smallest desired coverage that can detect a change in connectivity between adjacent cells.

The adaptation-notification time in an actual coverage expansion augments the adaptation notification time of an actual coverage reduction with a time-bounded expansion of the channel within the specified desired coverage. WC_EXP_ADAPT for an expansion of the actual coverage is also available from initial channel creation. In this case, WC_EXP_ADAPT includes the maximum expansion latency from a cell in the actual coverage to cover the most cells in the desired coverage (see Property 2 of chapter 4, section 4.3).

Adaptation Expansion in a Desired Coverage of 2 Cells

In this scenario, the desired coverage is 2 cells (cells 1 and 2). An expansion of the actual coverage implies the cell of the channel sender, cell 1, detects a change in connectivity with cell 2. The maximum expansion latency from cell 1 is 7560ms.[9] The WC_EXP_ADAPT for an expansion over a desired coverage of 2 cells is 13440ms.[10] The adaptation notification time observed for 30 iterations of an expansion in a desired coverage of 2 cells are shown in column 2 of Table 6.5. All observed results are

[9] 9240 (discovery latency) - 1680 (atomic broadcast), i.e., 0 latency to start the discovery

[10] 7560 (expansion latency) + 840 (latency to start atomic broadcast) + 1680 (atomic broadcast) + 3360 (detection)

Adaptation Expansion Notification			
	WC_EXP_ADAPT = 13440 (DC = 2 cells)	WC_EXP_ADAPT = 20160 (DC = 3 cells)	WC_EXP_ADAPT = 26880 (DC = 4 cells)
No.	Adaptation notification time (DC = 2 cells)	Adaptation notification time (DC = 3 cells)	Adaptation notification time (DC = 4 cells)
1	12867	15305	19587
2	12867	15305	19587
3	12867	16145	19754
4	12867	16145	19587
5	12867	16145	19587
6	12867	15305	19587
7	12867	15305	19754
8	12867	15305	19587
9	12867	15305	19754
10	12867	15305	19754
11	12867	15305	19754
12	12867	15305	19625
13	12867	16145	19625
14	12867	15305	19625
15	12867	15305	19625
16	12867	15305	19625
17	12867	15305	19625
18	12867	15305	19625
19	12867	16145	19625
20	12867	16145	19625
21	12867	15305	19625
22	12867	15305	19625
23	12867	15305	19625
24	12867	15305	19625
25	12867	15305	20585
26	12867	16145	19625
27	12867	15305	19625
28	12867	15305	19625
29	12867	15305	19754
30	12867	15305	19587

Table 6.5: Expansion adaptation notification time in a desired coverage from 2 to 4 cells

within WC_EXP_ADAPT for an expansion over 2 cells. The observed mean is 12867ms, the observed worst-case is 12867ms and the standard deviation is 0. Thus, the 99% confidence bounds are also both 12867ms.

Adaptation Expansion in a Desired Coverage of 3 Cells

The desired coverage in this scenario consists of cells 1, 2 and 3, with cells 1 and 2 in the actual coverage. The change of connectivity between cells 2 and 3 causes an expansion to include cell 3 also in the actual coverage.

The expansion latency from cell 2 is 10920ms.[11] The WC_EXP_ADAPT for an expansion over a desired coverage of 3 cells is 20160ms. The observed adaptation notification time for 30 iterations of an expansion in a desired coverage of 3 cells are shown in column 3 of Table 6.5, where all results are within WC_EXP_ADAPT for an expansion over 3 cells. The mean is 15501ms, the worst-case is 16145ms, and the standard deviation is 361ms. The 99% confidence interval bounds for adaptation notification of an expansion in a desired coverage of 3 cells are 14570ms and 16432ms. The increase in the bounds for the notification over 3 cells is to include the increased discovery latency and back propagation over an extended actual coverage route to the channel sender.

Adaptation Expansion in a Desired Coverage of 4 Cells

The desired coverage in this scenario consists of cells 1 - 4, with cells 1, 2 and 3 in the actual coverage. The change of connectivity between cells 3 and 4 causes an expansion to include cell 4 in the actual coverage also.

The expansion latency is 14280ms. The WC_EXP_ADAPT for an expansion over a desired coverage of 4 cells is 26880ms. The results for 30 iterations of an expansion adaptation notification in a desired coverage of 4 cells are shown in column 4 of Table 6.5. All observed results are within WC_EXP_ADAPT for the adaptation. The mean is 19674ms, worst-case is 20585ms and the standard deviation is 182ms. The 99% confidence interval bounds are 19204ms and 20144ms.

Adaptation Expansion of an Interior cell in the Desired Coverage

In the following experiments the desired coverage was 4 cells. The objective of these experiments was to show that timely adaptation notification was achieved regardless of where the adaptation occurs within the desired coverage. For example, an actual coverage expansion starting in cell 2, expands to include all cells in the desired coverage (cells 3 and 4), with the adaptation notification back

[11] The difference in the expansion latency is attributed to the larger discovery latency for channel creation.

Adaptation Expansion Notification		
	WC_EXP_ADAPT = 26880 (DC = 4 cells)	WC_EXP_ADAPT = 26880 (DC = 4 cells)
No.	Adaptation notification time (From cell 1)	Adaptation notification time (From cell 2)
1	26307	22985
2	26307	22145
3	26307	22985
4	26307	22025
5	26307	22145
6	26307	22145
7	26307	22985
8	26307	22145
9	26307	22985
10	26307	22145
11	26307	22145
12	26307	22145
13	26307	22985
14	26307	22985
15	26307	22145
16	26307	22145
17	26307	22145
18	26307	22144
19	26307	22145
20	26307	22145
21	26307	22145
22	26307	22985
23	26307	22145
24	26307	22145
25	26307	22985
26	26307	22145
27	26307	22145
28	26307	22145
29	26307	22145
30	26307	22145

Table 6.6: Expansion adaptation notification in a desired coverage of 4 cells

propagated to the channel sender (in cell 1). The WC_EXP_ADAPT in all experiments in the section was 26880ms. The timeliness of an expansion from cell 3 to cover all cells in a desired coverage of 4 cells was shown previously in column 4 of Table 6.5.

In cell 1 the expansion latency is 21000ms.[12]. The expansion adaptation notification time from cell 1 for 30 iterations are shown in column 2 of Table 6.6. The mean is 26307ms, the worst-case is 26307ms and the standard deviation is 0ms. The 99% confidence interval bounds for an expansion adaptation starting in cell 1 for a desired coverage of 4 cells are both 26307ms.

In cell 2 the expansion latency is 17640ms.[13] The expansion adaptation notification time from cell 2 for 30 iterations are shown in column 3 of Table 6.6. The mean is 22365ms, the worst-case is 22985ms and the standard deviation is 381ms. Thus, the 99% confidence interval bounds for an expansion adaptation starting in cell 2 for a desired coverage of 4 cells are 21382ms and 23348ms.

These results show that regardless of the location of the adaptation the notification of an expansion is within the bounds for WC_EXP_ADAPT for the desired coverage.

6.4 Remarks

The evaluation highlighted a number of artefacts of the implementation of both the SEAR and TBMAC protocols. Firstly, the timeliness of transmission delivery is influenced by the clock synchronisation algorithm implemented in the TBMAC protocol. The extension of the clock synchronisation protocol of the single-cell implementation of TBMAC is a very simple approach and may fail throughout an experimental run. A new clock synchronisation protocol for TBMAC, for example, to include earlier notification over inter-cell slots, may reduce Δ in cells, increase the duration within which clock synchronisation is occurring and improve the overall results.

Secondly, the assignment of inter-cell slots to adjacent cells effects the adaptation notification time. For example, the latency to start an atomic broadcast and latency to transmit following an atomic broadcast are directly related to slot assignment. A different assignment of slots could reduce the maximum adaptation notification time over varying actual coverage sizes.

6.5 Summary

The goal of the experiments performed was to evaluate the feasibility of the space-elastic model in the real world. From the evaluation it has been shown that timely transmissions are supported within the

[12] 22680 (discovery latency) - 1680 (atomic broadcast)

[13] 22680 (discovery latency) - 1680 (atomic broadcast cell 2) - 1680 (atomic broadcast reply to cell 1) - 1680 (atomic broadcast cell 1)

actual coverage regardless of distance, receiver mobility and the presence of simultaneous channels. In addition, timely adaptation notification of both reductions and expansions of the actual coverage was achieved regardless of the size of the desired coverage or the cell on a route in the actual coverage where the adaptation was detected.

Chapter 7

Conclusions and Future Work

This thesis presented the space-elastic communication model designed to address the problem of providing hard real-time communication guarantees in mobile wireless networks, and the SEAR real-time ad hoc routing protocol, which provides the basis for a real-world implementation and evaluation of the space-elastic model.

This chapter summaries the most significant contributions of the work described in this thesis and outlines its contribution to the state of the art. The chapter concludes with a discussion of related research issues that remain open for future work.

7.1 Contribution

The motivation for the work presented in this thesis arose from the observation that state-of-the-art research in supporting hard real-time communication guarantees has failed to address the challenges of dynamic wireless environments where route and resource availability is necessarily time-varying.

As described in chapter 2, timeliness has been achieved in wired networks where the typical assumptions are that offline scheduling for a known number of participants and applied load is possible and coupled with architecture-specific medium-access arbitration is used to satisfy hard real-time guarantees, for example in COSMIC. However, approaches used in wired networks do not translate to the wireless domain where the dynamics of the environment, for example, the dynamic topology and variable link quality, negate the assumptions on which these approaches rely. Furthermore, previous approaches to real-time routing in wireless networks do not accommodate the non-determinism and unpredictability characteristic of the dynamics in this domain. For example, unpredictable latency for route discovery and route maintenance coupled with a failure to provide time-bounded feedback

on changes in route availability, results in non-deterministic intervals within which hard real-time communication is not possible. Soft real-time communication is only provided by previous real-time routing approaches in the wireless domain.

The main research challenge was to design a communication model that supports hard real-time communication in wireless networks that overcomes the impact of the characteristics of this dynamic domain. A further challenge was to provide a real-time routing protocol that provides the properties of the model in the ad hoc domain. Both of these challenges have been overcome. The resulting communication model is known as the space-elastic model and the real-time ad hoc routing protocol is known as SEAR. Both have been presented in this thesis.

The space-elastic communication model supports a class of real-time applications, i.e., the space-elastic class that can operate correctly in an adaptable actual coverage. A space-elastic application identifies a desired space e.g., a set of geographical locations in a wireless network, within which timeliness properties should be guaranteed. The space-elastic model provides hard real-time communication in wireless networks by utilising a trade-off between time and space to provide timeliness guarantees in an adaptable actual coverage space. The safety and progress of a space-elastic application can still be guaranteed following an adaptation of the actual coverage if notification of a change in network state and the corresponding change to the actual coverage is available to the real-time application within a known time bound. A guarantee of time-bounded adaptation notification allows space-elastic applications to change their behaviour based on the current actual coverage within a known time bound. Using a combination of timely communication within an adaptable space and timely adaptation notification, the space-elastic model guarantees progress for space-elastic applications within a dynamic wireless network.

The second contribution of this thesis is a novel real-time ad hoc routing protocol, SEAR. The SEAR protocol supports the properties of the space-elastic model in a dynamic multi-hop ad hoc environment. SEAR was used as the basis for the evaluation of the feasibility of the space-elastic model in the real world. Using SEAR, it was shown that timely transmission latency is achievable with little jitter (in the worst-case 237μs) over a range of desired coverage sizes and in the presence of simultaneous channels. Furthermore, timely adaptation notification for both reduction and expansion adaptations can be provided. Thus, the objective of this thesis, to provide hard real-time communication guarantees in a dynamic wireless environment, was realised.

7.2 Future Work

As is always the case with research, there are some issues that remain open for possible future work, both in the short term and for longer term investigation.

7.2.1 Short Term

The space-elastic model supports both mobile and stationary real-time applications. The SEAR implementation presented in chapter 5, supports both mobile and stationary receivers but only stationary senders. To support the mobility of transmitting real-time applications necessitates a guarantee that allocated resources move with the mobile sender and are available when required regardless of the current location of the sender. This extension of the SEAR protocol requires the addition of resource scheduling to support timely allocation and re-allocation of slots depending on the movement of the transmitting host.

A challenge in the mobile sender scenario is to guarantee timely delivery of transmissions in a mobile (and dynamic) actual coverage. The integration of the SEAR protocol with mobility prediction (Tang et al. 2004, Sathyaraj & Doss 2005) to forecast the future movement and location of the sender at its next transmission on a channel and proactive routing and resource reservation to discover routes and schedule resources in a desired coverage relative to this predicted location, is another future research goal. Proactive routing and resource reservation is resource intensive in an already resource limited environment. Therefore, the accuracy of mobility prediction and the speed of detecting unnecessary route and resource reservations is paramount. Interesting questions arise, such as, how far in advance accurate predictions can be made with routes discovered and resources reserved, how fast "wrong" routes and resource reservations can be detected and how does the speed of movement and constraints of the channel, i.e., the specified periodicity and latency of transmissions, influence each of these decisions.

The approach to multi-cell clock synchronisation in the current implementation of SEAR is a simple master/slave approach, which, as shown in chapter 6, is prone to failure, the likelihood of which increases with distance from the synchronisation master. A new clock synchronisation algorithm, for example, to include earlier synchronisation information over inter-cell slots, may reduce the clock adjustments necessary in each cell, and thus increase the duration within which distributed multi-cell clock synchronisation is achievable. In addition, investigation into the provision of a new callback from TBMAC to SEAR signaling that required local clock adjustments are in excess of a known maximum adjustment bound and that clock synchronisation has failed, could be worthwhile. Using

the relationship between clock adjustments and observed jitter in delivery times, an upper bound representing the maximum allowable jitter could be set, with adjustments in excess of this value implying intolerable jitter which are translated to an adaptation of the actual coverage and notified by SEAR to channel senders.

The current SEAR implementation uses RT-WLAN (and real-time versions of the Lucent ORiNOCO drivers) as a basis for predictable message transmission and reception. Implementing real-time drivers for other widely deployed IEEE 802.11b wireless cards, e.g., Cisco Aironet (Cisco-Aironet 2006), D-Link (D-Link 2006), is a focus of future work. A comparative analysis of the latency to change wireless channel and the subsequent influence on slot size, round length and observed results for the execution of the same experiments as presented in chapter 6, would be of interest.

7.2.2 Long Term

The real-time channel abstraction of the space-elastic model does not preclude the inclusion of additional quality-of-service constraints, such as, bandwidth and power availability. In this case all constraints specified in a channel creation request would be used to perform route discovery and resource reservation. The inclusion of additional quality-of-service support in the real-time channel interface to SEAR is trivial. However, discovering routes and reserving resources to satisfy all specified constraints is challenging particularly in a dynamic ad hoc environment. A prerequisite to broadening the constraints supported would be an investigation of the quality-of-service guarantees that are of interest to real-time applications in different scenarios and includes useful insight into any prioritisation or trade-offs that are possible between the constraints.

A property of the space-elastic model is to provide time-bounded notification of an adaptation of the actual coverage. Behavioural changes by a real-time application are based on the notification of a detected adaptation that has occurred within a known time-bound. A possible alternative approach, with a potential for future research, is the notification of a predicted adaptation of the actual coverage. For example, using link quality prediction (Gaertner et al. 2004) and mobility prediction the availability of established routes in the actual coverage within a time bound in the future could be predicted, and if a change is forecast a notification transmitted to the channel sender. Thus, in this scenario behavioural adaptations by the channel sender are based on predicted actual coverage adaptations. The accuracy of prediction is again paramount as erroneous notifications would cause unnecessary changes to application behaviour. The applicability of this approach for real-time applications in different scenarios and the impact of it on maintaining safety would require further research.

Increasing context-based decision-making both within the SEAR protocol and provided by SEAR

for real-time applications, is of interest for future research. For example, the basis for gateway election in SEAR is location information. Augmenting the election decision with, for example, processing and battery power levels available to each host could enhance the election decision and provide routes with greater stability, thus, reducing adaptations resulting from gateway failure.

At present the only context information provided with an adaptation notification is the type of the notification (reduction or expansion), and the current actual coverage following the adaptation. Augmenting these details with additional contextual information relating to the reason for the adaptation may provide greater capability to the real-time application to reason about the behavioural change to make or an action to take. For example, the context of an adaptation reduction, such as, due to the evacuation of cells on a route in the actual coverage or due to the failure to elect a gateway in a cell in the actual coverage, may be valuable information to a real-time application. In this case, it may be possible to reason about the permanence (due to empty cells), or transience (due to the possibility of electing a gateway at the next election) of the adaptation and use this in subsequent actions taken by the real-time application. Using reasoning about the permanence of the notified adaptation a real-time application could initiate alternate route discovery in the desired coverage, i.e., initiating a route rediscovery based on the adaptation received. The feasibility and applicability of combining increased contextual content with notifications of predicted actual coverage adaptations for real-time applications may also be a basis for future research.

The availability of the space-elastic model opens up new approaches to building real-time applications. For example, the fields of Intelligent Transport Systems (ITS) and inter-vehicle communication both rely on timely context-based adaptations by applications to coordinate their behaviour to guarantee safety. Extending such real-time applications to coordinate their behaviour based on timely notifications provided by the space-elastic model may be an interesting topic of future research. In addition, the inclusion of the space-elastic model in the coordination of real-time sentient objects participating in stigmergic systems (Barron & Cahill 2004) that apply timely behavioural adaptation to interact and modify their environment may also be a potential area for future work. Using the space-elastic model to build such real-time applications will illustrate the applicability and feasibility of the model and present its limitations.

Bibliography

Almeida, C. & Veríssimo, P. (1996), Timing Failure Detection and Real-Time Group Communication in Quasi-Synchronous Systems, *in* '8th Euromicro Workshop on Real-Time Systems', L'Aquila, Italy.

Barron, P. & Cahill, V. (2004), Using Stigmergy to Co-Ordinate Pervasive Computing Environments, *in* '6th IEEE Workshop on Mobile Computing Systems and Applications', Lake District National Park, UK.

Bharghavan, V., Demers, A., Shenker, S. & Zhang, L. (1994), MACAW: A Media Access Protocol For Wireless LANS, *in* 'ACM SIGCOMM 1994', pp. 212–225.

Bosch, R. (2001), 'CAN Specification v 2.0'.

Bouroche, M., Hughes, B. & Cahill, V. (2006*a*), Building Reliable Mobile Applications with Space-Elastic Adaptation, *in* 'Mobile Distributed Computing Workshop (MDC 2006)'.

Bouroche, M., Hughes, B. & Cahill, V. (2006*b*), Real-Time Coordination of Autonomous Vehicles, *in* 'IEEE Conference on Intelligent Transportation Systems (ITSC 06)'.

Chandra, T. & Toueg, S. (1996), 'Unreliable Failure Detectors for Reliable Distributed Systems', *Journals of the ACM* **43**(2), 225–267.

Chen, S. & Nahrstedt, K. (1999), 'Distributed Quality-of-Service Routing in Ad Hoc Networks', *IEEE Journal in Selected Areas in Communications (JSAC)* **17**(8), 1–18.

Cisco-Aironet (2006), 'Cisco Aironet, http://www.cisco.com'.

Cristian, F. (1990), 'Synchronous Atomic Broadcast for Redundant Broadcast Channels', *Journal of Real-Time Systems* **2**, 195–212.

Cristian, F. & Fetzer, C. (1999), 'The Timed Asynchronous Distributed System Model', *IEEE Transactions on Parallel and Distributed Systems* **10**(6), 642–657.

Crow, B., Widjaja, I., Kim, J. G. & Sakai, P. (1997), 'IEEE 802.11 Wireless Local Area Networks', *IEEE Communications Magazine* **35**(9), 116–126.

Cunningham, R. & Cahill, V. (2002), Time Bounded Medium Access Control for Ad Hoc Networks, *in* 'Principles of Mobile Computing (POMC2002)', Toulouse, France.

D-Link (2006), 'D-Link Networking, http://www.dlink.com'.

Dana, P. (1997), 'The Global Positioning System (GPS Time Dissemination for Real-Time Applications', *Journal of Real-Time Systems* **12**, 9–40.

De, S., Das, S., Wu, H. & Qiao, C. (2002), 'Trigger-based Distributed QoS Routing in Mobile Ad Hoc Networks', *ACM SIGMOBILE Mobile Computing and Communications Review* **6**(3), 22–35.

Eberle, S., Ebner, C., Elmenreich, W., Farber, G., Gohner, P., Haidinger, W., Holzman, M., Huber, R., Schlatterbeck, R., Kopetz, H. & Stothert, A. (2001), Specification of the TTP/A Protocol, Technical report, Technische Universitat Wien, Viennna, Austria.

Ergen, M., Sengupta, R., Lee, D. & Varaiya, P. (2004), 'WTRP - Wireless Token Ring Protocol', *IEEE Transactions on Vehicular Technology* **53**(6), 1863–1881.

Fischer, M. J., Lynch, N. A. & Paterson, M. S. (1983), Impossibility of Distributed Consensus with One Faulty Process, *in* '2nd ACM SIGACT-SIGMOD Symposium on Principles of Database Systems', Atlanta, Georgia.

Fuhrer, T., Muller, B., Dieterle, W., Hartwich, F., Hugel, R. & Walther, M. (2000), Time Triggered Communication on CAN, Technical report, Robert Bosch GmbH.

Gaertner, G., ONuallain, E., Butterly, A., Singh, K. & Cahill, V. (2004), 802.11 Link Quality and Its Prediction - an Experimental Study, *in* 'Personal Wireless Communications', Vol. 3260 of *Lecture Notes in Computer Science*, Springer, pp. 147–163.

G.Cugola, Nitto, E. D. & Fuggetta, A. (2001), 'The JEDI Event-based Infrastructure and Its Application to the Development of the OPSS WFMS', *IEEE Transactions on Software Engineering* **27**(9), 827–858.

Gleeson, M., Hughes, B., Cunningham, R., Weber, S. & Cahill, V. (2006), Implementation and Evaluation of Time-Bounded Medium Access Control for Ad Hoc Networks, In Submission WCNC 2007.

Gomez, J., Campbell, A. T., Naghshineh, M. & Bisdikian, C. (1999), Power-aware Routing in Wireless Packet Networks, *in* '6th IEEE International Workshop on Mobile Multimedia Communications (MOMUC'99)', San Diego.

Gomez, J., Campbell, A. T., Naghshineh, M. & Bisdikian, C. (2001), Conserving Transmission Power in Wireless Ad Hoc Networks, *in* '9th International Conference on Network Protocols (ICNP 2001)', Riverside, California.

Gray, R. S., Kotz, D., Newport, C., Dubrovsky, N., Fiske, A., Liu, J., Masone, C., McGrath, S. & Yuan, Y. (2004), Outdoor Experimental Comparison of Four Ad Hoc Routing Algorithms, *in* 'International Workshop on Modeling Analysis and Simulation of Wireless and Mobile Systems', Venice, Italy, pp. 220–229.

Hartenstein, H., Bochow, B., Lott, M., Ebner, A., Radimirsch, M. & Vollmer, D. (2001), Position-aware Ad Hoc Wireless Networks for Inter-Vehicle Communications: the FleetNet Project, *in* '2nd ACM International Symposium on Mobile Ad Hoc Networking & computing (MobiHoc2001)', ACM Press, New York, NY, USA, pp. 24–35.

He, T., Stankovic, J. A., Lu, C. & Abdelzaher, T. (2003), SPEED : A Stateless Protocol for Real-Time Communication in Sensor Networks, *in* '23rd International Conference on Distributed Computing Systems (ICDCS03)', Providence, Rhode Island, U.S.A.

Huber, B. & Elmenreich, W. (2004), Wireless Time-Triggered Real-Time Communication, *in* '2nd IEEE Workshop on Intelligent Solutions in Embedded Systems', Graz, Osterreich, pp. 169–182.

Hughes, B. & Cahill, V. (2006), Wireless Communication Using Real-Time Extensions to the Linux Network Subsystem, Technical report, "Computer Science, Trinity College, Dublin".

Hughes, B., Gleeson, M., Karpinski, M., Cunningham, R., Weber, S. & Cahill, V. (2006), Real-Time Communication in IEEE 802.11 Mobile Ad Hoc Networks - A Feasibility Study, Technical report, "Computer Science, Trinity College, Dublin".

IEEE (2005), *IEEE Std. 802.11 2nd Edition - Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*, IEEE.

Johnson, D. B. & Maltz, D. A. (1996), Dynamic Source Routing in Ad Hoc Wireless Networks, *in* Imielinski & Korth, eds, 'Mobile Computing', Vol. 353, Kluwer Academic Publishers.

Kaiser, J. & Mock, M. (1999), Implementing the Real-Time Publisher/Subscriber Model on the Controller Area Network (CAN), *in* '2nd IEEE International Symposium on Object-oriented Real-Time Distributed Computing', Saint-Malo, France.

Kaiser, J., Brudna, C. & Mitidieri, C. (2005), 'COSMIC: A Real-time Event-based Middleware for the CAN-bus', *IEEE Journal of Systems and Software* **77**(1), 27–36.

Karimi., H. A. & Krishnamurthy, P. (2001), Real-Time Routing in Mobile Networks Using GPS and GIS Techniques, *in* '34th Hawaii International Conference on System Sciences', Maui, Hawii.

Karn, P. (1992), MACA - A New Channel Access Method for Packet Radio, *in* '9th Amateur Radio Computer Networking Conference (ARRL/CRRL)', pp. 134–140.

Kopetz, H. (1998), A Comparison of CAN and TTP, *in* 'Proceedings of the IFAC Distributed Systems Workshop', Como, Italy.

Lin, C. R. & Gerla, M. (1999), 'Real-Time Support in Multihop Wireless Networks', *ACM/Baltzer Wireless Networks* **5**(2), 125–135.

Manoj, B. S. & Murthy, C. S. R. (August 2002), Real-Time Traffic Support for Ad Hoc Wireless Networks, *in* '10th International Conference on Networks (ICON02)', Singapore, pp. 335–340.

Mantegazza, P., Bianchi, E., Angelo, M., Beal, D. & Yaghmour, K. (2000), DIAPM-RTAI Programming Guide 1.0, DIAPM.

Martins, P., Sousa, P., Casimiro, A. & Veríssimo, P. (2005), A New Programming Model for Dependable Adaptive Real-Time Applications, *in* 'IEEE Distributed Systems Online - Real-time Systems', Vol. 6.

McDonald, A. B. & Znati, T. (1999), 'A Mobility Based Framework for Adaptive Clustering in Wireless Ad Hoc Networks', *IEEE Journal in Selected Areas in Communications (JSAC)* **17**(8), 1466–1487.

Murthy, C. S. R. & Manoj, B. S. (2004), *Ad Hoc Wireless Networks Architecture and Protocols*, Prentice Hall Professional Technical Reference.

Nader, N. & Wise, D. (1990), 'An Organization and Interface For Sensor-Driven Semiconductor Process Control Systems', *IEEE Transactions on Semiconductor Manufacturing.*

NS2 (2003), NS2 Reference Manual, Technical report.

OMG-CORBA (1995), The Common Object Request Broker: Architecture and Specification Revision2.0, Technical report, Object Management Group.

Perkins, C. E., Belding-Royer, E. & Das, S. R. (2000*a*), 'Ad Hoc On-Demand Distance Vector (AODV) Routing'.

Perkins, C. E., Belding-Royer, E. & Das, S. R. (2000*b*), 'Quality of Service for Ad Hoc On-Demand Distance Vector (AODV) Routing'.

Sathyaraj, B. H. & Doss, R. C. (2005), Route Maintenance using Mobility Prediction for Mobile Ad Hoc Networks, *in* 'IEEE International Workshop on Heterogeneous Multi-Hop Wireless and Mobile Networks', Washington DC,USA.

Schlatterbeck, R. & Elmenreich, W. (2001), TTP/A: A Low-Cost, Highly Efficient, Time-Triggered Fieldbus Architecture, *in* 'Society of Automotive Engineers (SAE) World Congress', Detroit, Michigan, USA.

Schmidt, D., Levine, D. & Mungee, S. (1998), 'The Design of the TAO Real-Time Object Request Broker', *Computer Communications Special Issue on Building Quality of Service into Distributed Systems (Elsevier Science).*

Setton, E., Yoo, T., Zhu, X., Goldsmith, A. & Gorid, B. (2005), 'Cross-Layer Design of Ad Hoc Networks For Real-Time Video Streaming', *IEEE Wireless Communications, Invited Paper.*

Shah, S. & Nahrstedt, K. (2002), Predictive Location-Based QoS Routing in Mobile Ad Hoc Networks, *in* 'IEEE International Conference on Communications', New York City, U.S.A.

Shukla, D., Chandran-Wadia, L. & Iyer, S. (2003), Mitigating the Exposed Node Problem in IEEE 802.11 Ad Hoc Networks, *in* '12th International Conference on Computer Communications and Networks, (IC3N'03)', Dallas, Texas, USA.

Tanenbaum, A. S. (1988), *Computer Networks: 2nd Edition*, Prentice-Hall, Inc., Upper Saddle River, NJ, USA.

Tang, J., Xhe, G. & Zhang, W. (2004), Reliable Routing in Mobile Ad Hoc Networks Based on Mobility Prediction, *in* '1st IEEE International Conference on Mobile Ad Hoc and Sensor Systems', Fort Lauderdale, USA.

Tobagi, F. & Kleinrock, L. (1975), 'Packet Scheduling in Radio Channels: Part ii-The Hidden Terminal Problem in Carrier Sense Multiple Access and the Busy-Tone Solution', *IEEE Transactions on Communications* **23**(12), 1417–1433.

Veríssimo, P. & Casimiro, A. (2002), 'The Timely Computing Base Model and Architecture', *IEEE Transactions on Computers* **51**(8), 916–930.

Veríssimo, P. & Rodrigues, L. (2001), *Distributed Systems for System Architects*, Kluwer Academic Publishers, Norwell, MA, USA.

Visser, M. & Zarki, M. (1995), Voice and Data Transmission Over an 802.11 Wireless Network, *in* '6th International Symposium on Personal Indoor and Mobile Radio Communcations (IEEE PIMRC)', Toronto, Canada, pp. 648–652.

Wang, K. & Li, B. (2002), Efficient and Guaranteed Service Coverage in Partitionable Mobile Ad Hoc Networks, *in* 'IEEE Joint Conference of Computer and Communication Societies (INFOCOM'02)', New York City, New York.

Yoo, J. & Kim, C.-K. (2005), 'On the Hidden Terminal Problem in Multi-Rate Ad Hoc Wireless Networks', *Lecture Notes on Computer Science (LNCS)* pp. 479–488.

Appendix A

Adaptation Notification Times

The evaluation of the timeliness of adaptation notification for both reduction and expansion adaptations presented in chapter 6, section 6.3, shows the observed adaptation notification times rounded up to the nearest milliseconds in each experiment.

The actual adaptation notification times returned for both reduction and expansion adaptations are presented in this appendix in nanoseconds. The tables presented in this appendix have a one-to-one correspondence with the relevant table presented in chapter 6, and are presented in the same order.

Adaptation Reduction Notification			
	WC_RED_ADAPT = 5880 (DC = 2 cells)	WC_RED_ADAPT = 9240 (DC = 3 cells)	WC_RED_ADAPT = 12600 (DC= 4 cells)
No.	Adaptation notification time (DC = 2 cells)	Adaptation notification time (DC = 3 cells)	Adaptation notification time (DC = 4 cells)
1	5158650382	7335311138	10670003376
2	5278518076	7647000110	10329128389
3	5158657054	7446679355	10569115117
4	5278654797	7875272426	10609979476
5	5158619281	8147679541	10393000081
6	5198658327	7566783347	10490011649
7	5198589099	8087708329	10809117103
8	5198649039	8147670919	10209119461
9	5198655791	7547583819	10369982945
10	5198589297	8027715196	10509097795
11	5198594913	7547716103	10809126343
12	5198591115	7547714652	10489989393
13	5198576079	7926783548	10489986089
14	5318654159	7446729704	10569128196
15	5198649810	7566802458	10449103013
16	5198649333	7934207364	10569130476
17	5198650131	7446800665	10569125308
18	5198660095	7547622005	10369990921
19	5198653646	7926724749	10669983460
20	5198650351	7446746601	10569125121
21	5198649010	7866746865	10609986350
22	5198613152	7866637672	10509044029
23	5198611301	7487563718	10393005107
24	5198603233	8027598937	10392997237
25	5198614564	8087605955	10609992589
26	5198609582	7746704704	10609990514
27	5198647607	7686724627	10209115868
28	5198579437	7806727492	10489984003
29	5198614536	7487564913	10509123376
30	5198571182	7933987297	10369122718

Table A.1: Reduction adaptation notification time in a desired coverage ranging from 2 to 4 cells

Adaptation Reduction Notification		
	WC_RED_ADAPT = 12600 (DC = 4 cells)	WC_RED_ADAPT = 12600 (DC = 4 cells)
No.	Adaptation notification time (DC = 4 cells)	Adaptation notification time (DC = 4 cells)
1	5318643456	7487705675
2	5198646891	7386801735
3	5318644437	8087707364
4	5198646662	7326846364
5	5198635705	8027717987
6	5198638430	8207667733
7	5198640034	7727659488
8	5198628769	7866806598
9	5198624609	7547716906
10	5198632564	8147675195
11	5198622878	8046803282
12	5198567373	8046803282
13	5198625346	8147675195
14	5198630337	7547716906
15	5198623898	7866806598
16	5198623898	7727659488
17	5198630337	8207667733
18	5198625346	8027717987
19	5198567373	7326846364
20	5198622878	8087707364
21	5198632564	7386801735
22	5198624609	7487705675
23	5198628769	7727657750
24	5198640034	7926805448
25	5198638430	7547721873
26	5198635705	8027709554
27	5198646891	7934204225
28	5198627351	8147666707
29	5198624235	7386850266
30	5318652867	7806937164

Table A.2: Reduction adaptation of cells 2 and 3 in a desired coverage of 4 cells

Adaptation Expansion Notification			
	WC_EXP_ADAPT = 13440 (DC = 2 cells)	WC_EXP_ADAPT = 20160 (DC = 3 cells)	WC_EXP_ADAPT = 26880 (DC = 4 cells)
No.	Adaptation notification time (DC = 2 cells)	Adaptation notification time (DC = 3 cells)	Adaptation notification time (DC = 4 cells)
1	12867022112	15304733463	19587192755
2	12866902852	15304868024	19587399107
3	12866905274	16144883466	19754461745
4	12866923724	16144890151	19587386634
5	12866903408	16144747839	19587195000
6	12866928822	15304872260	19587399408
7	12866889389	15304877129	19754472044
8	12866924607	15304877090	19587397280
9	12866930290	15304899106	19754482780
10	12866924823	15304893771	19754480794
11	12866933897	15304883691	19754484165
12	12866912065	15304875577	19625301281
13	12866918516	16144889299	19625313940
14	12866923429	15304881233	19625289740
15	12866967984	15304866006	19625314845
16	12866921724	15304778912	19625464934
17	12866908208	15304870123	19625370735
18	12866919477	15304885360	19625362691
19	12866782190	16144904228	19625296714
20	12866728670	16144751427	19625166329
21	12866794943	15304878824	19625330893
22	12866823739	15304879656	19625400867
23	12866804445	15305283740	19625341441
24	12866808611	15305245772	19625302994
25	12866809171	15305366285	20585278205
26	12866818966	16145288549	19625232113
27	12866802989	15305260511	19625373410
28	12866768878	15305306818	19625244780
29	12866766673	15305383840	19754009669
30	12866879030	15305433261	19586712051

Table A.3: Expansion adaptation notification time in a desired coverage from 2 to 4 cells

Adaptation Expansion Notification		
	WC_EXP_ADAPT = 26880 (DC = 4 cells)	WC_EXP_ADAPT = 26880 (DC = 4 cells)
No.	Adaptation notification time (From cell 1)	Adaptation notification time (From cell 2)
1	26306903920	22984847440
2	26306900630	22144818962
3	26306925540	22984815211
4	26306922755	22024831613
5	26306924971	22144824150
6	26306623176	22144707356
7	26307260108	22984826254
8	26307269194	22144831896
9	26307245861	22984702685
10	26307264591	22144825710
11	26306799427	22144733406
12	26306911134	22144818464
13	26306920615	22984824708
14	26306907106	22984833410
15	26306912605	22144799586
16	26306925924	22144798930
17	26306933049	22144771067
18	26306913545	22144432514
19	26306904455	22144861005
20	26307266109	22144831308
21	26307269204	22144819713
22	26307496185	22984902770
23	26307252144	22144839425
24	26306739350	22145290496
25	26306740905	22985279374
26	26306714720	22145276416
27	26306807015	22145306849
28	26306903211	22145277734
29	26306843830	22145305933
30	26306895995	22144864693

Table A.4: Expansion adaptation notification in a desired coverage of 4 cells