Opportunistic service composition in dynamic ad hoc environments

Christin Groba

A thesis submitted to the University of Dublin, Trinity College

in fulfillment of the requirements for the degree of

Doctor of Philosophy (Computer Science)

March 2013

Declaration

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

Christin Groba

Dated: March 26, 2013

Permission to Lend and/or Copy

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

Christin Groba

Dated: March 26, 2013

Acknowledgements

This thesis would not have been possible without the support and encouragement of many people for which they deserve thanks and recognition. First and foremost, I would like to thank my supervisor, Prof. Siobhán Clarke, for her guidance and scientific advice, patience and kind words, as well as for teaching me the beauty of precise and rigorous research. Thanks also to Lero, the Irish Software Engineering Research Centre, for giving me the opportunity to conduct research in Ireland among the first students of the Lero Graduate School. Many thanks goes to Serena Fritsch and Paula Hannon for their vivid discussions about service composition and to Stephan Weber for his insightful views on mobile ad hoc networks. I would also like to thank the members of my research group, DSG, for their support. In particular, thanks to Guoxian Yang, Luca Longo, and Mithileash Mohan for giving me their perspective on my research and for reminding me that there is a life besides the PhD. Many thanks also to Killian Levacher and Melanie Sherlock who are a constant source of inspiration always challenging me to think outside the box. Thank you to my parents who encouraged me to explore the world while grounding me in traditions and down-to-earth principles such that I always have a place to which I can return. Thank you to my sister Claudia who led by example and showed me that PhD can be done. Finally, Patrick, thank you for your constant support, love, and patience, and for being there all along.

Christin Groba

University of Dublin, Trinity College

March 2013

Abstract

Mobile and embedded devices capture, process, and exchange sensory data about their operating environment, making them suitable service providers for ubiquitous computing. In particular, composing services that are hosted on different devices creates new value-added functionality and supports context-aware applications e.g., for smart public spaces. This thesis explores service composition in dynamic ad hoc networks which represent a very dynamic form of ubiquitous computing environments. Service providers may join, roam, or leave the network and offer services only as appropriate to their own objectives and resource capacity.

Mobility and participation autonomy, however, change the network and service topology frequently. They impose a high failure probability on composites because network links are likely to break and service offers may become unavailable. Although network routes are repairable and service allocations can be re-assigned, failure recovery comes at the cost of delay and additional communication. Preventive measures such as keeping backups for service providers and network routes, on the other hand, incur additional monitoring and maintenance overhead regardless of whether failure occurs or not. Service provision through service composites, therefore, faces the challenging question of how to efficiently adapt to a continuously evolving operating environment.

Research in service-oriented computing has led to dynamic strategies for service discovery, allocation, and invocation to accommodate for changes at runtime. Common to these strategies is their reliance on a pre-established service overlay structure or the allocation of all required services at once. While in advance service registry facilitates the discovery process, its maintenance overhead increases as more dynamic provider information needs to be captured and made available. Allocating services all at once

allows for verifying whether the composite will execute to completion. However, by the time a particular service is invoked, the dynamics of the system may have rendered early allocation decisions invalid and cause failure. Techniques that finalise the allocation decision only prior to invoking the service are more flexible, but involve monitoring overhead and allocate more resources than the composite actually needs.

This thesis presents opportunistic service composition as an alternative to support the flexible implementation of complex service requests in highly dynamic environments. The novel composition model bundles and defers all interactions with a service provider until the required sub-service needs to be invoked. It interleaves service discovery and composite allocation with provider invocation to minimise the window for change to negatively impact the composite. The proposed composition protocol supports service sequences and parallel service flows. The approach lets service providers control their involvement to support participation autonomy, defines an explicit resource blocking mechanism to address provider resource-constraints, and devises a cross-layer communication solution to reduce the communication load on narrow wireless bandwidth.

Evaluation of the protocol has been achieved using model-based verification and simulation. Verification through automated model checking confirms that a formal specification of the designed composition protocol does not deadlock, terminates in a valid end state, and adequately covers all required sub-services in both sequential and parallel service composites. The verified model has also been implemented and integrated with a simulator for mobile ad hoc networks. The simulation-based evaluation assesses the performance of opportunistic service composition in comparison to more conventional baselines. It quantifies how the interplay of service discovery, allocation, and invocation affects the failure probability of composites in mobile ad hoc settings. The results of this study demonstrate that the opportunistic approach generally achieves its design objective to decrease failure in a communication-aware manner and at the same time reveal the limits of these benefits with regard to request complexity, network density, and service demand.

Publications Related to this Ph.D.

1. Groba, Christin and Clarke, Siobhán (2012). Towards in-network aggregation for people-centric sensing, *Ninth International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services (MobiQuitous)*.

2. Groba, Christin and Clarke, Siobhán (2012). Synchronising service compositions in dynamic ad hoc environments, *First International Conference on Mobile Services (MS)*, IEEE, pp. 56-63.

3. Groba, Christin and Clarke, Siobhán (2012). Towards Opportunistic Service Composition in Dynamic Ad Hoc Environments, *PhD Symposium at Ninth International Conference on Service Oriented Computing (ICSOC PhD Symposium)*, Springer, pp. 189–194.

4. Groba, Christin and Clarke, Siobhán (2011). Opportunistic composition of sequentially connected services in mobile computing environments, *Eighteenth International Conference on Web Services (ICWS)*, IEEE, pp. 17-24. (Best student paper award)

5. Groba, Christin and Clarke, Siobhán (2010). Web services on embedded systems - A performance study, *Workshop at Eighth International Conference on Pervasive Computing (PERCOM Workshops)*, IEEE, pp. 726-731.

6. White, Jules and Clarke, Siobhán and Groba, Christin and Dougherty, Brian and Thompson, Chris and Schmidt, Douglas C. (2010). R&D challenges and solutions for mobile cyber-physical applications and supporting Internet services, *Journal of Internet Services and Applications* 1(1), pp. 45-56. (invited article)

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Mobile and embedded devices have evolved from special purpose equipment to smart entities that can sense their environment, process sensory data, and communicate with nearby peers or servers on the Internet. Pervasive computing builds upon these capabilities to provide context-aware applications that, receptive to the user's environment, point to relevant information or trigger necessary actions. In addition, smart device technology appears to lower the boundary for people to track and share their experiences. Already, millions of users capture pictures, news, comments and even physical achievements with their mobile phones and post them on social networking platforms. Novel applications tap this growing potential. People-centric sensing, for example, uses the sensing capabilities of mobile phones and the movement of their carriers to collect sensory data and to improve the micro- and macroscopic view of a city (Lane et al., 2010). Aggregating and classifying data from a variety of sources produces higher level context, which allows for more robust conclusions and a deeper insight into the situation of an individual, a group, or an entire community. The opportunities that arise from mobile sensing and the positive attitude of people towards sharing are not limited to the World Wide Web, but may also take effect in a user's immediate surroundings as the following scenario shows:

> Adam uses the recording and processing capabilities of his smartphone to track his daily exposure to noise. Later, he correlates that with other health measures to better understand his stress levels. The phone samples, filters,

> classifies, and geo-tags audio data. However, being in Adam's pocket, the phone is unable to record high-quality data and would benefit from calibration with another device. In our scenario, as a field engineer, Adam is about to meet a new customer. In this unfamiliar environment, his phone's classifier software is not properly trained to categorise the noise. To make matters worse, the phone's GPS unit malfunctions and fails to determine Adam's position. As a result, the phone lacks the fundamental requirements for noise tracking and denies this service. However, Adam has heard of an innovative way of solving this issue and configured his phone to collaborate with nearby mobile phones that share their capabilities (Miluzzo et al., 2010). Adam's phone forms an ad hoc network with Joe's and Mary's phones, who are walking next to Adam, and issues the noise tracking task as a request. By incorporating these other devices Adam's phone increases the chances of covering all hardware and software requirements to complete the task.

The collective effort of multiple mobile devices is required if no single device has enough resources to handle a complex task alone. In particular, a structured approach needs to be put in place to allocate and execute subtasks. Service-oriented computing provides for such an structured approach. It is based on the notion of a service that encapsulates software-based behaviour or an application component to make it discoverable, accessible, and reusable over the communication network (Papazoglou and Heuvel, 2007). The composition of services creates new value-added functionality and involves a mechanism to specify, instantiate, and resolve complex service requests. Typically, the defined order and functionality of required sub-services guides the discovery, allocation, and invocation of suitable service providers. These service-oriented principles map to requirements in pervasive computing environments. Device capabilities like software components, locally stored data, and hardware resources can be modelled as services while service composites represent complex sensing tasks that span across multiple data sources (Kalasapur et al., 2007; Chakraborty et al., 2004; Brønsted et al., 2010). In the scenario (cp. Figure 1.1), Mary's phone encapsulates its microphone as an audio sampling service and the GPS unit as a geo-tagging service. Joe's phone provides access to different functions of

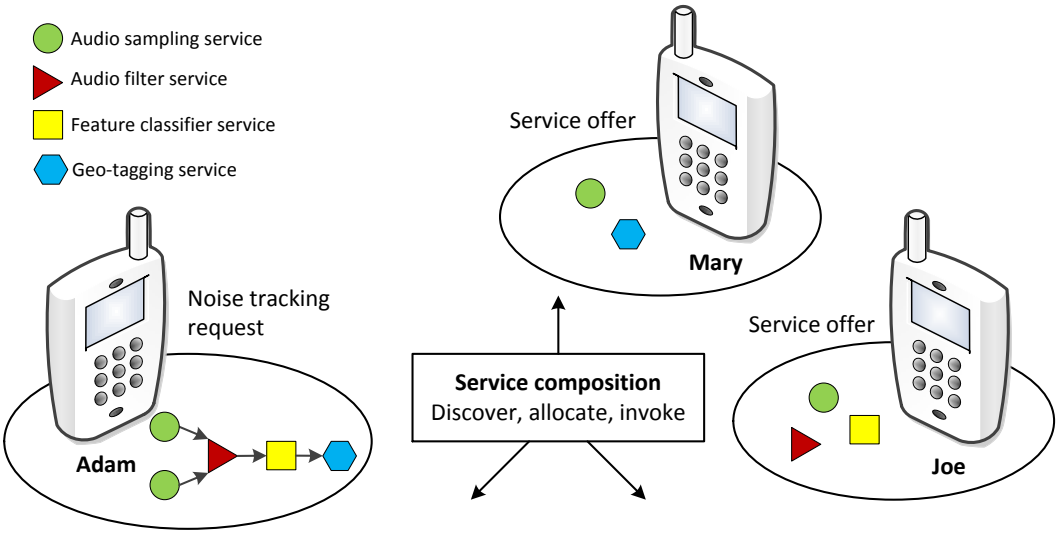


Fig. 1.1: ***Scenario*** *Adam wants to track his exposure to noise but does not have the required capabilities available on his mobile phone. Joe and Mary offer their phones' hardware and software capabilities as services. Service composition discovers, allocates, and invokes these services to satisfy Adam's complex request.*

its audio software via an audio filtering service and a feature classification service. In addition, the phone offers an audio sampling service. Adam's noise tracking task is a service composite that first requires two audio sampling services, then an audio filtering service, thereafter a feature classification service, and finally a geo-tagging service. Service composition is the mechanism that discovers Joe and Mary's phone services, allocates them to Adam's composite request, and invokes them to provide Adam with the result of that new value-added service.

The focus on service composition in dynamic ad hoc networks distinguishes this thesis from mashups based on the Web of Things (Guinard, 2010; Pintus et al., 2010) and service composition in sensor and context cloudlets (Loke, 2012), which rely on dedicated infrastructure to manage the access to smart entities. Dynamic ad hoc networks evolve spontaneously and on-demand. The absence of dedicated infrastructure requires potential composite participants to organise network and service related operations among themselves. The physical size of mobile devices limits their locally available resources and communication range. Participants thus exchange messages either directly if they are in each other's transmission range or indirectly via intermediaries that relay mes-

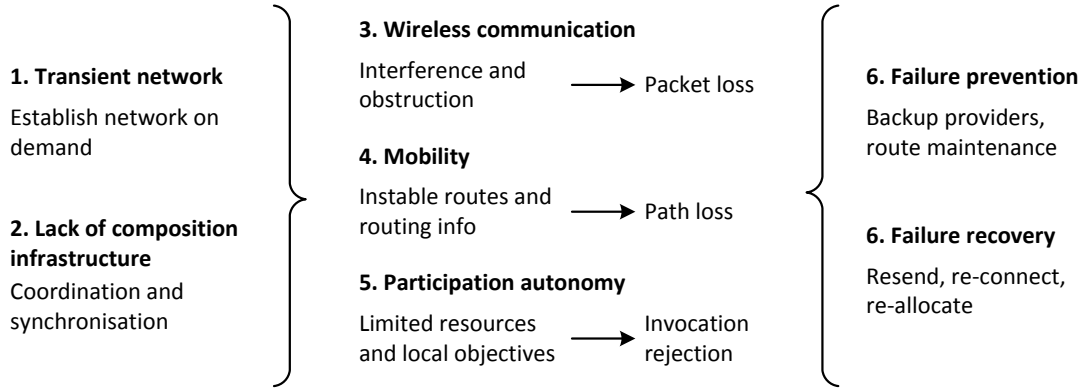


Fig. 1.2: ***Challenges*** *Service compositions in transient networks lack dedicated composition infrastructure and are likely to fail due to the unreliability of wireless communication and the instability that mobility and participation autonomy create. Failure prevention and recovery rely on further communication and expose the composition again to the dynamics of the environment.*

sages. Free of any obligations towards a unifying authority, service providers join, roam, and leave the network at any time. They engage in compositions spontaneously if this suits their residual load and objectives.

1.1 Challenges

The dynamic nature of mobile ad hoc networks presents a significant challenge for the composition of complex service requests (Brønsted et al., 2010; Kalasapur et al., 2007). In particular, service composition faces (cp. Figure 1.2):

- **Challenge 1: Transient networks**

  An ad hoc network may evolve only because a service request was raised and dissolves after the request is completed. Pre-established knowledge about the service and network topology does not exist and a network is established by interaction (Chlamtac et al., 2003). This means, service providers are initially unaware of each other's existence and find out about their neighbourhood only if co-located providers announce themselves or respond to a request.

- **Challenge 2: Lack of composition infrastructure**

  Free from any managed infrastructure, a dynamic ad hoc network lacks a single

entity with global system view. In particular, there is no dedicated entity that acts as a proxy for service discovery, allocation, and invocation and composite participants must manage complex requests between themselves in a decentralised peer-to-peer manner (Gu and Nahrstedt, 2006). Decentralisation, however, implies additional coordination and synchronisation, especially if composite requests extend beyond a sequential execution structure and contain parallel execution paths, so-called service flows[1], that must be merged prior to returning the final result (Yildiz and Godart, 2007).

- **Challenge 3: Wireless communication**

  Service composition, as well as building the service and network topology requires network and composite participants to communicate. In ad hoc networks, however, communication is wireless and inherently unreliable (Friedman et al., 2007). The exchange of messages via the radio medium is prone to interference and obstruction, which causes packet loss.

- **Challenge 4: Mobility**

  Services are hosted on mobile devices. Network paths are likely to be lost when mobile service providers move because these frequent changes of the network topology are likely to break established routes and invalidate cached routing information (Chlamtac et al., 2003).

- **Challenge 5: Participation autonomy**

  Service providers are not obliged to a higher authority. They may temporarily disable service provision and reject service invocation if this conflicts with local objectives or the availability of resources (Campbell et al., 2008). For example, although a provider has generally agreed to host a sensing service, it may only offer the service when in a certain location.

- **Challenge 6: Failure prevention versus recovery**

  In complex service requests each constituent service is prone to packet loss, path loss, or service rejection and presents a potential source of failure for the entire

[1] In the following, the terms service flow and parallel service flow will be used interchangeably.

composite. This makes it "very difficult to maintain a composed service for extended periods of time" (Sen et al., 2004). Although routes can be repaired and services re-allocated (Jiang et al., 2009; Feng et al., 2007), failure recovery comes at the cost of further communication over error-prone network links and delays the final composition result. Preventive failure recovery assigns backup providers and keeps routes updated at all times to reduce the recovery delay (Gu and Nahrstedt, 2006; Prinz et al., 2008). These techniques are based on participants issuing heart beat messages to signal changes in the topology. For composition messages this periodic network traffic, however, presents a source for interference and communication failure. Further, the resource-constraints of mobile service providers limit the number of composites they can handle simultaneously. If a provider is a backup for a service in one composite, it may not be available as a primary provider for another composite. Allocating one or more backup providers for each required service in addition to the primary provider may lead to temporary provider shortage.

Considering the dynamics of mobile ad hoc environments and the complexity of service requests the question arises: How can composites efficiently adapt to their continuously evolving operating environments to reduce the likeliness of failure to occur?

1.2 Existing solutions

In service-oriented computing, the need for more flexibility towards changes to the user requirements and operating environment has led to creating abstract composites during the design phase and finalising them by assigning service providers at runtime. This way, service providers can be dynamically selected or replaced based on their current availability and actual execution properties (Cardellini et al., 2009).

Acquiring dynamic provider information, however, is a challenge. Many composition solutions (e.g., Park and Shin, 2008; Samimi and McKinley, 2008; Wang et al., 2004) assume a service overlay structure that monitors all providers and excludes those that are currently unavailable, incapable, and unwilling to serve a request. Establishing such a structure in advance and independent from a particular composite request eases service discovery but at the same time requires effort to build and maintain. A registry,

for example, that holds both static and dynamic provider information (Schuler et al., 2004), needs repeated updating to keep up with evolving system dynamics. Alternative strategies (Gu and Nahrstedt, 2006; Samimi and McKinley, 2008) reduce this overhead and register only static service descriptions. Once a particular service request is raised, the potential providers are contacted and examined for their current properties. Based on an already existing network, these techniques study their solution in isolation from networking and service discovery aspects. In highly dynamic environments this leads to increased overhead considering that a network emerges only because of a service request and dissolves after it has been satisfied.

Decentralised composition management has been proposed for cross-organisational business processes (Martin et al., 2008; Fernández et al., 2010), peer-to-peer networks (Gu and Nahrstedt, 2006; Wang et al., 2004), hierarchical device networks (Kalasapur et al., 2007) as well as for mobile ad hoc networks (Sen et al., 2008). However, the lack of integration between the composition phases either increases the composite's failure probability or involves additional maintenance and monitoring overhead. Generally, composition solutions decouple service invocation from service discovery and allocation such that they do not invoke the first service until each required service has been assigned to a provider. In systems with moderate dynamics this ensures that a composition will not fail due to missing service providers. However, this benefit is limited if changes occur more frequently. Static allocations (Sen et al., 2008) are then nonetheless subject to failure because providers can change their availability after they have been assigned. Leaving the allocation flexible until the invocation of a service (Schuler et al., 2004; Prinz et al., 2008), is more sensitive to departing providers or the arrival of better performing candidates, however, comes at additional maintenance and monitoring cost. The group of providers that is assigned to a service has to stay updated on the dynamics within the group and must observe the primary provider to compensate for its disconnection. In addition, service providers may not have enough resources to engage in another composition simultaneously. Allocating multiple of these providers per required service over a longer period of time (i.e., until all the service's predecessors have been invoked) decreases the overall availability of services.

1.3 Thesis approach

A novel model for service composition, hereafter called opportunistic composition, is designed to reduce the failure probability of complex service requests in dynamic ad hoc environments. Based on valuable insights from existing solutions, it explores the possibility of re-organising the composition process to flexibly adapt to system dynamics.

Assumptions For the environment in which service composite requests are issued, this thesis makes the following assumptions:

1. The operating environment of service composites is highly dynamic. Mobility and participation autonomy change operating conditions more quickly than it takes to fully allocate, verify, and execute a composite request.

2. The network does not rely on dedicated communication infrastructure. Service providers establish network links in an ad hoc manner and route a message towards its destination. In line with Chakraborty et al. (2005), this work views infrastructure-based architectures as a special case of infrastructure-less environments. Stationary, resource-rich entities may relieve service providers from some of their duties but may not be accessible from all the devices at all times.

Observation One of the key drivers for this research is the observation that the delay of the traditional composition process is a source for failure. The composition process contacts a service provider at least three times: first to locate and examine the provider, then to block its resources, finally to invoke service execution. With growing complexity of the composite in terms of path length, the delay between these interactions increases. For example, the invocation of a service provider must wait until all successor services have been allocated and all predecessor services have executed. Meanwhile, the provider may move on, disappear completely, or stop offering the service. Unaware of these changes, the composite believes the provider to be in the same place and state as before and fails to reach it again. Checking with potential service providers more frequently than the minimum three times may be a solution to become aware of changes but incurs more traffic and risks clogging the network that leads to communication failure.

Hypothesis This thesis investigates how to reduce the delay between individual composition phases in a communication efficient manner. It frames its hypothesis as follows: The longer the delay between composition phases, the more likely are changes in the operating environment to occur. The later the operating environment is explored to compose a complex service request, the less time there is for the system to change and to render composition decisions invalid.

Basic idea The novel concept of opportunistic service composition bundles the discovery, allocation, and invocation for a required sub-service to decouple it from its position in the complex request. The model defers all interactions with service providers until they are indispensable for the composition to make progress. It searches and allocates a service provider when the corresponding sub-service is executed next. The approach is opportunistic because it starts executing the composite without having fully allocated all required services and thus does not know whether there is a provider for each service. Other approaches verify this up front but once they start, they have to deal with the same system dynamics in which a service provider present during verification may have moved or disappeared. It is important to notice that the objective of opportunistic service composition is to reduce failure. As it cannot prevent failure entirely, this approach is complementary to strategies that recover the composite after failure.

1.4 Thesis contribution

This thesis investigates how complex tasks can be coordinated in dynamic ad hoc environments using service composition while accommodating for changes in the operating environment that otherwise would lead to failure. This research contributes to the body of knowledge by providing:

Extensions to existing composition models Existing approaches to service composition tend to decouple service invocation from service discovery and service allocation which introduces delays and increases the likelihood of composite failure in transient networks. This thesis describes a novel opportunistic composition approach to align the complexity of a service composite to its frequently changing operating environment. In-

tegrating the composition phases and deferring allocation decisions to the latest possible moment, as proposed in this thesis, has not been previously investigated. Composing complex service requests in this manner allows for greater flexibility to adapt to available services and reduces the failure probability of composites.

Extensions to support parallel service flows With the reorganisation of the composition process, the proposed opportunistic approach requires additional means to support composites with parallel execution paths. This thesis defines algorithms on how multiple composite participants agree on a common successor. Current solutions require continuous path updates while the proposed algorithms synchronise once at the end of a parallel path. Further, the thesis provides detailed information on how to reduce a composite during its execution to avoid resource allocation for redundant or obsolete parts. These techniques ensure a consistent way of managing service composites and support the specification and resource-efficient execution of conditional composite behaviour.

Extensions to communication capabilities Currently, service composition is, as part of the top layer in the communication stack, isolated from the knowledge bases of lower layers which limits its awareness of the dynamic operating environment and risks sub-optimal composition choices. This thesis investigates how to utilize the broadcast nature of wireless radio for service composition and to enable composition actions that are aware of the surrounding network and service topology. The proposed cross-layer communication approach has not been previously examined for service composites and allows for communication-efficient service composition with less failure.

1.5 Thesis scope

The proposed composition model enables mobile entities to fulfil complex tasks by collaborating with peers in their vicinity. The model preserves participation autonomy as it considers only those service providers that actively commit to a request. However, the incentives for mobile service providers to share their capabilities are not studied. Different methods to encourage cooperation and the effectiveness of those can be found in (Li and Shen, 2012) and (Zhao et al., 2011).

Focusing on a novel way to organise service composition, this work is general in terms of how services and complex tasks are expressed. It does not assume a specific language (e.g., WSDL, BPEL, WS-CDL) to describe service offers and requests. Neither does it include particular concepts (e.g., ontologies) to classify services. The model applies syntactic matching to task descriptions that define the order and behaviour of required services but is well aware of more sophisticated approaches that allow for semantic matching (Nedos et al., 2009) and goal-oriented composition (El Falou et al., 2010).

Locality of service provision is a key criterion for service allocation in mobile ad hoc networks because the more distant a service provider, the more expensive is communication (Friedman et al., 2007). The design of the model supports locality needs by acting on demand and dynamically transferring the composition control. This way the vicinity of the current composition controller is naturally explored first. Provider allocation may consider further quality of service criteria if such provider information is available. Balancing multiple constraints, however, is not covered in this work, though has been studied, for example, by Zeng et al. (2004) and Cardellini et al. (2009).

This work analyses the discovery, allocation, and invocation of composite services as well as the interfaces between those phases with the goal to reduce the high failure probability that mobile ad hoc environments entail. Failure recovery, in particular the cost of it, is considered for building an argument for opportunistic service composition. Recovery mechanisms themselves are not integrated in this work. However, as the proposed approach does not prevent failure, different recovery methods (e.g., Philips et al. 2010; Jiang et al. 2009; Feng et al. 2007) will need to be analysed to make the approach more reliable.

1.6 Thesis structure

State of the art Chapter 2 analyses how state of the art composition solutions meet the challenges of highly dynamic pervasive computing environments. In particular, the study explores a) how dynamic provider information is discovered and kept up to date, b) how provider allocation is stabilised against frequent changes, c) how scarce provider resources are efficiently blocked, consumed, and released, d) how these concepts ex-

tend to requests with parallel execution paths, and e) how messages among composite participants can be delivered efficiently.

Design Chapter 3 returns to the challenges of service composition in highly dynamic environments outlined in Chapter 1 and describes the design objectives, system model, and design decisions of this thesis. Thereafter the chapter explains how the proposed protocol for opportunistic service composition addresses the design decisions in detail.

Design verification Chapter 4 uses model checking to verify the correctness of the designed protocol with regard to the absence of deadlocks, the termination in a valid end state, and the correct number of allocated service providers. For this the chapter first introduces the modelling tools PROMELA and SPIN, then examines basic protocol features on basis of service sequences, and thereafter turns to the extensions that support composites with parallel execution paths.

Implementation Chapter 5 describes the Java implementation of the designed and verified composition protocol. It explains how the integration with the network simulator Jist/SWANS enables mobility for composite participants. Then, the chapter highlights implementation details of the protocol and specifies the format of different composition messages. Details with regard to directed broadcasting and the cross-layered approach towards topology management complete the chapter.

Evaluation Chapter 6 evaluates how well the novel composition approach achieves its objective of reducing composite failure. It first describes the experimental setup of the simulation-based study. The second part of the chapter presents and analyses the results showing that the proposed composition protocol is a suitable alternative to existing solutions for service composition in dynamic ad hoc environments.

Conclusion Chapter 7 summarises the thesis and its achievements. It then discusses important findings with regard to the proposed composition protocol and highlights failure recovery and people-centric sensing as two potential areas for future work. A final remark provides a succinct wrap-up of this work.

1.7 Chapter summary

Mobile and embedded devices accompany people throughout the day and their processing and sensing capabilities support assisting us with information and actions that suit our current situation. Collaboration among such smart devices has the potential to achieve complex application requirements that a single device would fail to provide alone. The collective effort of multiple mobile peers can be modelled as a composition of services that mobile devices host and share. This thesis envisions service composition to occur spontaneously anytime and anywhere without the need for dedicated infrastructure to be in place. The network among potential service providers emerges on-demand and may dissolve after the composite request has been satisfied.

The transient nature of these mobile ad hoc networks, however, exposes service composites to three main failure sources: packet loss due to the narrow wireless bandwidth, path loss due to the mobility of the providers, and service rejection due to dynamic provider participation. Further, service providers need to manage complex service requests among themselves and balance the cost of failure prevention and recovery. These challenges raise the question of how composites can efficiently adapt to their evolving operating environment. Existing approaches have led to dynamic composition strategies to accommodate for change. However, as the next chapter will discuss in detail, their lack of integration between the main composition phases increases either the failure probability or the overhead for failure prevention.

This thesis builds on the hypothesis that the later the operating environment is explored to compose a complex service request, the less time there is for the system to change and to render composition decisions invalid. The following chapters describe how the novel model of opportunistic service composition reduces the delay between the phases of the composition process in a communication efficient manner to lower the composite failure in highly dynamic pervasive environments.

Chapter 2

State of the art

Service composition has been widely used to manage intra- and inter-organisational processes in enterprise systems and on the Web. With technological advances such as embedding sensors in everyday objects, the possibility to create new value-added functionality from existing services becomes attractive for pervasive computing environments (Brønsted et al., 2010). However, in contrast to reliable, wired, and resource-abundant enterprise networks, pervasive systems face frequent topology changes, error-prone wireless communication, and resource-constrained service providers. These differences introduce new challenges for service composition and limit the applicability of existing solutions. In particular, traditional centralised composition architectures with a single fixed coordinator suffer from a number of issues that make them inadequate for dynamic operating environments. As a hot spot for computation and communication (Ye, 2006) a static coordinator can run only a certain number of compositions concurrently which limits its scalability (Schuler et al., 2004; Balasooriya et al., 2008). In addition, composites that are distributed by nature and that model cross-organisational collaboration have to undergo unnecessary changes to map them to a centralised architecture (Martin et al., 2008; Zaplata et al., 2010). Further, centralised approaches lack the flexibility to cope with dynamic pervasive environments because they are designed for fixed network and service topologies (Chakraborty et al., 2004; Mostarda et al., 2010).

The drawbacks of centralised architectures have led to decentralised strategies to handle a continuously evolving network of service providers. The following review analyses

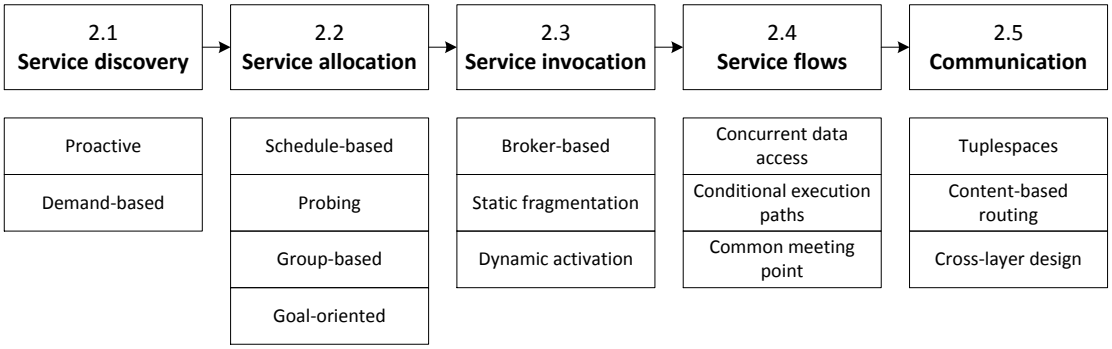


Fig. 2.1: ***Structure of state of the art review*** *The review analyses the applicability of existing service composition solutions for dynamic ad hoc networks from the perspective of service discovery, allocation, and invocation as well as the support for service flows and the awareness of wireless ad communication.*

the applicability of these strategies for dynamic ad hoc networks. Figure 2.1 illustrates the structure of the analysis, which starts with service discovery and the question of how dynamic provider information is discovered and kept up-to-date (cp. Section 2.1). Next, it explores how service allocation is stabilised against frequent changes in the operating environment (cp. Section 2.2). Then, the analysis examines service invocation approaches and how scarce provider resources are efficiently blocked, consumed, and released (cp. Section 2.3). Thereafter, it studies service flow solutions and how they synchronise parallel execution paths (cp. Section 2.4). Finally, the analysis turns to wireless communication and how messages among composite participants can be delivered efficiently (cp. Section 2.5).

2.1 Service discovery

Many composition solutions (e.g., Schuhmann et al., 2013; Park and Shin, 2008; Samimi and McKinley, 2008; Wang et al., 2004) assume the existence of some kind of service registry, which locates service providers in the network. Implementing this assumption, however, represents a challenging research field in itself. Service discovery must enable networked entities to announce local capabilities as services and enquire about services of other entities despite frequent topology changes and varying communication characteristics (Ververidis and Polyzos, 2008). While recent surveys (Mian et al., 2009;

Ververidis and Polyzos, 2008) demonstrate the maturity of service discovery in mobile ad hoc networks, this section adds a new perspective by analysing service composition solutions and by asking: How is the knowledge about available services kept up to date if provider information such as connectivity, load, and willingness to participate is dynamic? Service discovery solutions fall into two categories when it comes to publishing service offerings: proactive and demand-based service announcements.

2.1.1 Proactive

When service providers announce themselves proactively, they do so independently of a particular request and their announcements have to be cached and kept up to date.

OSIRIS (Schuler et al., 2004) is a peer-to-peer infrastructure to support reliable and scalable process management in dynamic environments. In OSIRIS, a potential composite participant maintains a local cache for information it needs to fulfil a service. As the participant's service can be part of a composite, the cache may store information about providers of a possible successor service. A centralised meta-data repository registers all dynamic changes in the environment and replicates them by pushing relevant information to particular participants. OSIRIS is not readily applicable in dynamic ad hoc environments because these environments do not have dedicated infrastructure to host a centralised entity for information replication. Alternative approaches, like CiAN (Sen et al., 2008), group-based service discovery (Chakraborty et al., 2006), and dynamic service composition Kalasapur et al. (2007), do not require additional infrastructure as they found other ways to update cached provider information.

CiAN (Sen et al., 2008) is a workflow engine to support modelling collaborative activities as structured tasks and accomplishing them in mobile ad hoc networks. In CiAN, participants synchronise locally cached service information as soon as they come into each other's communication range (Sen et al., 2004). This way, knowledge about available services propagates from participant to participant. The initiator of a composite request, however, must wait until a provider for all required services appears in the cache before it can continue to allocate the services. This delays the composition result, in particular, if the network of service providers has just started to emerge.

Group-based service discovery (Chakraborty et al., 2006) is a service discovery ar-

chitecture to allow for distributed, scalable, and adaptive service discovery in pervasive computing environments. Similarly to CiAN, it relies on local caching and participant-to-participant data dissemination. However, while CiAN needs to wait and see if a required service eventually appears in the local cache, this approach will actively search for it. Group-based service discovery selectively forwards a service request and uses the discovery route backwards to deliver a discovery response. This reduces the message overhead compared to simply flooding the network with a search request. A similar approach (Aguilera and López-de Ipiña, 2012) uses selective forwarding based on service groups and the time-to-live parameter of the communication protocol to reduce the overhead of creating a local service graph on each participant.

Dynamic service composition, as proposed by Kalasapur et al. (2007), is a composition mechanism for pervasive computing environments to provide service-related support for resource-poor devices and to assign available resources to user needs, even when an exact match does not exist. Its locality-aware hierarchical caching technique has the potential to further reduce the search overhead as it defines a parent-child relation over co-located devices. Resource-rich parent devices cache the announcements of neighbouring resource-poor devices as their children. If a required service is unavailable in the local cache, the search query travels the hierarchy upward to the parent that has greater overview. When there is a match, the parent replies with the physically closest option, otherwise it forwards the request to its own parent.

Notwithstanding the search optimisations, the general issue with proactive service announcements is that, the more dynamic information the announcements include, the more maintenance and communication they require. On the other hand, the fewer dynamic information they contain, the lower is the system's ability to sense and adapt to changes. For example, if the load or objective of a provider changes, it may temporarily disable the provision of a particular service. However, since the service has been announced, the provider may receive service invocation requests which fail because the provider refuses processing them. Revoking the announcement may not avoid failure because of the dissemination delay in the network. Announcing dynamic changes more frequently incurs higher communication overhead and strains the narrow communication bandwidth which may lead to increased network failure.

2.1.2 Demand-based

When potential composite participants announce service information on-demand (i.e., when there is a particular request), they reveal up-to-date data that do not need caching or maintenance. However, as demand-based service discovery incurs search delays, existing solutions nonetheless ask providers to register to minimise the delay.

Spidernet (Gu and Nahrstedt, 2006) is a service composition framework to provide for high quality and failure resilient service composite in peer-to-peer systems. In Spidernet, service providers announce only static service information proactively which are stored in distributed hash tables. Collecting dynamic information is left to the allocation phase that is triggered when there is a particular composite request. This avoids frequent and expensive table updates. Static service information is a prerequisite for creating a service overlay network that links available services based on whether they are connected by a network link. The overlay is later examined to find a service path that corresponds to the composite request.

Logical service groups (Prinz et al., 2008) is a concept for dynamic service composition in peer-to-peer systems to support the adaptation to peer arrivals or variations of their execution properties. It uses a decentralised publish-subscribe mechanism based on distributed hash tables to address the drawback of centralised information replication (cp. OSIRIS in Secion 2.1.1). Providers subscribe based on their service type with the corresponding administrating peer. They are notified when a request for that service type is published and may respond by publishing the static and dynamic details of their offer. The request initiator receives all published replies via the administrating peer. In contrast to Spidernet (Gu and Nahrstedt, 2006), the distributed hash tables do not register service descriptions. Rather, they support targeted multicast messaging.

Generally, demand-based service announcement is communication-efficient because service providers reveal just enough dynamic information as required by a request and target it to a particular service consumer. In addition, the service consumer learns only about providers that have enough resources and are willing to execute a required service because otherwise these providers would not respond. This avoids service invocation rejections. However, the above solutions rely on a distributed provider registry that needs

constant monitoring to replace failed parts. In networks that emerge and dissolve quickly such a registry needs to be established first and then needs to compensate instability which neutralises the communication efficiency.

2.1.3 Summary

Dynamic provider information can be announced proactively or revealed on-demand. Proactive announcements do not need additional infrastructure which makes the approach suitable for dynamic ad hoc networks. However, the approach requires high maintenance or risks service invocation rejection when service providers temporarily disable individual services due to changes of their system load or objectives. In solutions with demand-based announcements only available service providers respond to a service request which avoids service invocation rejection and repeated information updates. However, the solutions surveyed are based on an established peer structure that does not exist if a network among service providers emerges because a composite request was raised and dissolves after the request is complete. For dynamic ad hoc environments it may be more efficient to use demand-based service discovery without any pre-existing provider registry as the overhead of maintaining a peer structure may outweigh the benefit of information it contains. Instead, investigations on how to recognise service demand without explicit service requests may further improve demand-based service discovery.

2.2 Service allocation

Service allocation examines ways of finding, for each required service, a suitable provider such that an abstract request description turns into an executable composite. Composites in dynamic ad hoc environments face frequent changes in the network and service topology which may invalidate allocation decisions and cause their failure. This raises the question: How is the allocation of service providers stabilised against dynamics of the composite's operating environment? The approaches reviewed in this section base their allocation decisions on schedule information, probing, provider groups, and high level goals to counteract composite failure.

2.2.1 Schedule-based

One way to handle dynamic environments is to incorporate schedule information into the allocation process (Wang, Wang, Zheng, Xu and Yang, 2009; Sen et al., 2008). In service-oriented wireless sensor networks, for example, the sensor sleep schedule limits service availability and recovering from resulting disruptions consumes scarce energy. An approach to minimise re-allocation and to save energy derives the duration for which a provider is continuously available from its sleep schedule and services offerings (Wang, Wang, Zheng, Xu and Yang, 2009). From different feasible sets of providers, that steadily and collectively provide a composite, the base station chooses the set with minimum transmission cost and propagates the allocation in the network.

Similarly, in leader-team scenarios (e.g., on a remote construction site) each service provider has a schedule containing its commitments in time and space. This can be used to allocate tasks such that team members encounter each other to exchange intermediate results and trigger the next step in a workflow (Sen et al., 2008). In both approaches, failure due to service unavailability is unlikely because changes are foreseeable and thus disruptions avoidable. Further, if service providers are mobile, their obligation towards their leader (Sen et al., 2007) or organisation (Catarci et al., 2008) steers their movement towards achieving a task or staying connected.

However, the availability of schedule information and the obligation towards an authority are characteristics that apply only to a subset of dynamic ad hoc environments. Generally, service providers are autonomous and not obliged to disclose their schedules. Freely roaming the network, they rather spontaneously participate in a composite.

2.2.2 Probing

Probing-based solutions gather dynamic provider information in a service overlay network to guide their allocation toward a stable composite. A service overlay network is a representation of available services that creates logical links between services if their providers are connected by a network route (Park and Shin, 2008).

Multi-path solutions (Gu and Nahrstedt, 2006; Park and Shin, 2008; Samimi and McKinley, 2008; Wang, Xu, Qian and Lu, 2009) induce probe messages in a service

overlay network to trace the characteristics of potential service candidates and identify multiple high-quality composition paths. Probing with resource-aware routing (Park and Shin, 2008) estimates, e.g., the residual energy, contention rate, and response time to avoid exhaustion and overload in the network. Pruning strategies reduce the overhead of probing that would otherwise span the entire overlay network. Spidernet (Gu and Nahrstedt, 2006) limits (e.g., based on the priority of the request) the total number of probe messages that is allowed for a composite and the number of duplicate service providers that should be probed. Dynamis (Samimi and McKinley, 2008), an algorithm for efficient probing in service overlays, applies selective forwarding whereby probe recipients forward probe messages only if the contained path is of higher quality (e.g., in terms of end-to-end delay, load balance, or security) once they add their service.

In contrast, single-path solutions like sFlow (Wang et al., 2004) determine a single high quality path in the service overlay. sFlow is an allocation algorithm to provide resource-efficient and agile service composites. While exploring the service overlay, it allocates the most suitable provider in each hop and avoids the need for pruning strategies. sFlow applies multiple quality objectives, namely latency and bandwidth, and determines the shortest widest path from an allocated service provider to its successor.

Exploring the service overlay network and constructing the optimal composition path en route aligns the allocation to dynamic provider properties. However, probing-based allocation is decoupled from service discovery and implies additional composition delays and communication overhead.

2.2.3 Group-based

Group-based allocation solutions assign a set of providers to a required service and leave the allocation decision flexible until service invocation.

OSIRIS (Schuler et al., 2004) notifies composite participants about the arrival or departure of possible providers for the subsequent service in the composite. Once the participant receives a service invocation request, it executes the service and only then allocates the subsequent provider from the set of candidates.

Alternatively, a logical service group (Prinz et al., 2008) represents all available service providers for a particular service type. The group dynamically re-elects the

group leader, who eventually executes the required service, if the current leader departs or another member with more suitable execution properties arrives. For a composite, the requesting peer starts with the last required service, creates a logical service group and assigns the leader role to the best performing peer in that group. In the same way, the group leader appoints the leader for its preceding service. A group subscribes to changes in their successor group to ensure it stays informed about leader re-elections.

Group-based allocation flexibly adapts to dynamic service availability as it integrates better performing providers even if they arrive after the initial allocation. The proposed techniques, however, group service providers logically without imposing locality restrictions. This is at risk of increased communication cost because group members or consecutive providers can be scattered over the network.

2.2.4 Goal-oriented

Goal-oriented allocation approaches create composites automatically to cater for unpredictable and evolving circumstances (Bucchiarone et al., 2012). Instead of abstract composite descriptions, requesting entities express their needs in terms of available input and expected output. The composition handles the actual construction of the request. For this, a service aggregation algorithm (Kalasapur et al., 2007; Wang, Xu, Qian and Lu, 2009) creates a graph of available services and links those that are semantically and syntactically compatible. Traversing the graph from input to output (Thomas et al., 2009) or in reverse (Wang, Xu, Qian and Lu, 2009) creates the actual composite request with the corresponding service providers.

Other techniques employ an artificially intelligent (AI) planner (Madhusudan and Uttamsingh, 2006; Bidot et al., 2011; Pinto et al., 2012) and distribute the planning (El Falou et al., 2010) among multiple service agents to reduce the high number of possible service combinations. Agents first derive partial plans locally and a central coordinating agent merges those to a final plan.

Goal-oriented allocation is highly flexible because goals rather than specific tasks are matched against currently available services. However, it relies on an up-to-date view of available services to generate a stable execution plan and reinforces the issues related to proactive service announcements and the need for their frequent updates (cp. Section

2.1.1). Further, it is not clear whether mobile devices have sufficient resources to handle the higher complexity and coordination effort that in particular AI techniques imply.

2.2.5 Summary

Allocation solutions use schedule information, probing, provider groups, or goal-oriented composite creation to stabilise the allocation against frequent topology changes. Among these approaches, integrating schedule information in the allocation process is most effective because changes are foreseeable and their negative impact can be avoided. However, in dynamic ad hoc environments such information is unlikely to be available because service providers are autonomous and not obliged to disclose their schedules. Goal-oriented automatic composite creation is highly flexible because it matches goals rather than specific tasks to available services. Its need for an up-to-date view of available services, however, requires providers to announce themselves proactively and reinforces the associated challenges that are discussed in Section 2.1.1. Group-based allocation is also very flexible towards changes because the final provider of a required service gets allocated only prior to its execution. However, the reviewed solutions do not impose locality restrictions of service groups which increases the communication effort for managing the group. Probing-based techniques preserve the locality of allocations naturally because the service overlay network which they explore is based on physical network links. The drawback of probing is that it is decoupled from service discovery and introduces additional composition delay and communication effort. With the flexibility of group-based allocations and the locality of probing techniques the question arises whether the benefits of both can be integrated in one approach.

2.3 Service invocation

Service invocation ensures that one provider per required service is called for execution. In contrast to Internet-based service domains, mobile computing environments are at risk of exhaustion and invocation rejection sooner because mobile devices have fewer local resources available and depend on the objectives of their (human) carrier. This motivates the question: How are provider resources efficiently blocked, consumed, and

released to maximise general provider availability and to maintain stable composites? Broker-based designs, static fragmentation, and dynamic activations are ways to invoke the required services of a composite and address the question from different perspectives.

2.3.1 Broker-based

Broker-based composition approaches entrust a single entity, i.e., the broker, with the discovery, allocation, and invocation of required services. Solutions for nomadic networks place the broker on a reliable fixed node to ensure the composite does not disappear (Philips et al., 2010). Such a setup, however, implies long expensive routes or failure, if the fixed broker is not in vicinity of or disconnected from mobile service providers. In contrast, distributed dynamic brokers (Chakraborty et al., 2005) are part of the mobile ad hoc network and are dynamically selected. A requesting entity appoints its broker based on how well the broker connects to required services. The selection considers the broker's local services, services in its neighbourhood, load, and energy to increase the composite's flexibility towards topology changes.

Brokers reduce the involvement of the service provider in the composition to a minimum. Unaware of whether its invocation context is a composite or basic service request, the provider does not need to participate in a lengthy allocation process or wait for predecessors to finish. It blocks local resources only for the duration of the service execution and releases them once it sends the service result back to the broker. However, above solutions do not explicitly obtain the provider's commitment. They simply invoke the provider and assume it has sufficient resources and is willing to participate. Further, brokers are the destination for all intermediate results of the composite and prevent direct, possibly more localised, interaction among subsequent service providers. Finally, brokers may not be available in ad hoc networks that emerge and dissolve quickly because none of the participants has initially a sufficient overview of available services.

2.3.2 Static fragmentation

Static fragmentation partitions a centralised composite description and deploys the composite fragments on previously selected service providers such that the providers can

interact directly without a broker. Fragmentation techniques (Balasooriya et al., 2008; Fernández et al., 2010; Martin et al., 2008; Sen et al., 2008) are manifold but share a similar notion about fragment content which includes the service to execute and references to the input source and output destination of that service.

Bond-Flow (Balasooriya et al., 2008), for example, is a middleware to provide distributed coordination for collaborative applications. It distributes the complexity of the centralized composition logic over stateless web services and dynamically generates coordinator proxy objects to simplify the composite development. These proxies wrap a service with coordination logic as well as state and dependency information and enforce dependencies during execution. Composites that apply the chemical programming paradigm (Fernández et al., 2010; Fernández et al., 2012) use the metaphor of chemical reaction rules to coordinate complex applications. They follow the same concept as Bond-Flow but formalise decentralised composite execution in terms of a higher-order chemical language. Another alternative to formalise the control flow among fragments are executable workflow networks (Martin et al., 2008) which is a process model to partition executable BPEL processes.

CiAN (Sen et al., 2008) decouples the input and output dependencies of a fragment from the actual fragment provider to ease re-allocation. Fragments identify dependencies on other fragments via task numbers rather than provider addresses such that in case of a failure a provider can easily be replaced without notifying subsequent providers.

In contrast to fully-decentralised but hard to validate solutions, a hybrid approach (Mostarda et al., 2010) compiles centralised composite descriptions into choreographed synchronised finite state machines and a skeleton that is operated by a central leader. A consensus protocol between the leader, its backups, and the local state machines allows for consistency among distributed participants and correct execution.

Generally, static fragmentation has three drawbacks: First, all fragments are allocated at once and before the composite starts executing. This makes the composite insensitive to changes in the operating environment that occur during the allocation or invocation. Second, fragments block resources for all required services even those that do not get executed due to conditional paths or premature termination. Third, with the possibility of dead execution paths, a provider faces the challenge of determining when

it is safe to uninstall a fragment and free up resources. Regarding the first drawback, group-based allocation approaches (Schuler et al., 2004; Prinz et al., 2008) are more receptive to the system dynamics as they postpone the final allocation decision to when the required service needs to be invoked. However, until then they block a number of service providers and minimise provider availability for other composite requests.

2.3.3 Dynamic activation

The dynamic activation of service providers is an alternative approach to execute a composite in a decentralised manner and to block resources that are actually needed.

One way to achieve this is to store task files in a centralised repository rather than on the allocated service provider (Ye, 2006). Service invocation signals activation and the invoked provider first fetches its task file according to which it then executes the assigned service and invokes the subsequent service provider. A technique based on dependency tables (Fdhila et al., 2009) may be used to partition the centralised composite description and generate task files instead of code fragments.

Continuation-passing (Yu, 2009b) is the fully decentralised alternative that transfers the remainder of a composite along with the composition control from provider to provider. Continuation objects originate from programming languages and are a viable solution for composite execution as they reduce the accumulation of control context (Manolescu, 2002). Self-describing workflows (Atluri et al., 2007) detail how a service provider derives its execution part from the continuation and how it afterwards creates a new self-describing workflow for the remainder.

Dynamic activation addresses the shortcomings of static fragmentation and activates service providers as the composite execution enfolds. However, the way existing approaches (Ye, 2006; Atluri et al., 2007; Yu, 2009b) are organised, implies that service providers have been identified in an allocation process but have not yet been notified to reserve resources. The approaches assume that a service provider is always ready to serve and ignore the fact that changing load and objectives may negatively affect the availability of a provider. In addition, these solutions have not evaluated the suitability of dynamic activation in mobile ad hoc environments.

2.3.4 Summary

Solutions to service invocation range from broker-based designs over static composite fragmentation to dynamic provider activation. Broker-based designs reduce the involvement of a service provider in a composite to a minimum because the broker handles all composition-related tasks and the provider only executes a required service. However, brokers prevent direct provider interaction and are unlikely to be available in transient networks where all participants initially have a limited view of available services. Static fragmentation provides for direct provider interaction and formalises the control flow among consecutive service providers. However, deployed prior to composite execution, static fragments are insensitive to system dynamics and allocate resources for conditional parts of the composite that do not get executed. Dynamic provider activation is an alternative for decentralised composite execution and blocks resources that are actually needed. However, none of the reviewed solutions explicitly confirms the availability a provider to execute a service. One the one hand, they may simply assume the provider is available for service invocation which increases composite failure because the load and objectives of a provider changes dynamically. On the other hand, the solutions may assume providers have been blocked during service discovery which would imply a long blocking period that minimises the general availability of service providers. Either way, these assumptions highlight a gap and the need to address it.

2.4 Service flows

Service flows are service composites with parallel execution paths. This section refers to composites whose parallel execution paths merge in a single thread of control. For example, in the introductory scenario (cp. Figure 1.1 on page 3) Adam's noise tracking composite contains an audio filter service that merges the results of two independent audio sampling services and then continues with a single feature classifier service. The filter service is a merge service and represents the common meeting point of the two parallel execution paths. Service flows may require the synchronisation of concurrent data access and may contain conditional execution paths. For example, Adam could have modelled his composite request such that a global variable holds the noise readings

and needs to be accessed by the two sampling services concurrently. In addition, he could have included a conditional path that integrates a third sampling service if it is available. As the complexity of composites increases, the challenges of managing concurrent data access, handling conditional execution paths, and identifying common meeting points arise. This motivates the question: How can service composites efficiently synchronise parallel execution paths despite frequent topology changes?

2.4.1 Concurrent data access

Similar to multiple threads in a program, parallel paths of a service composite may access and modify the same global data object. Executing parallel paths without synchronisation leads to inconsistency (Russell et al., 2005) and could produce wrong or undesired results (Zaplata et al., 2010). One solution is to reduce concurrent access to shared data during the design of the composite and otherwise define data classes from which an appropriate synchronisation strategy can be derived at runtime (Zaplata et al., 2010). Such a strategy may be based on a unique token for data access that only one path at a time can hold. The execution paths exchange the token by including it within control messages to reduce the synchronisation effort (Yu, 2009a). Alternatively, a centralised leader forwards the token to interleave parallel tasks and to synchronise the composite state immediately after the execution of each service (Mostarda et al., 2010).

Among these solutions, avoiding concurrent data access seems most desirable as it does not incur any effort at runtime. Otherwise, passing data access tokens with control flow messages is more suitable for dynamic ad hoc networks than a centralised synchronisation solution because a central entity prevents from direct provider interaction.

2.4.2 Conditional execution paths

Conditional execution paths allow for flexibility in modelling a composite request as intermediary results may change how the composite proceeds. However, a composite reveals only during its execution which parallel paths actually need to be merged into a single thread of control (Russell et al., 2006). A merge service must be prepared to receive a message from any possible path (Yu, 2009a).

Static fragmentation approaches (cp. Section 2.3.2) pre-allocate all services in a composite and need to send an invalidation message along obsolete paths to release redundant service providers during composite execution. The concept of pattern replication (Fdhila and Godart, 2009) reduces this effort by creating composite partitions that, aligned to the control logic, disable entire successor partitions rather than individual services. For this solution to become effective, providers for a partition cannot be allocated until the partition is about to be invoked. However, the approach, as published, does not provide details on how the allocation and invocation of composite partitions is organised.

Dynamic activation solutions (cp. Section 2.3.3) remove obsolete conditional paths in the remainder of the composite as the composite enfolds. Continuation passing, as proposed by Yu (2009a), initialises service providers with potential input dependencies from preceding services and needs to notify them if their path becomes obsolete. Self-describing workflows (Atluri et al., 2007) do not require additional notifications. However, as this solution does not explicitly block resources on allocated service providers, which potentially leads to service invocation rejections, it is unclear how it would otherwise release them. Similarly, broker-based designs (cp. Section 2.3.1) also do not obtain the provider's commitment prior to its invocation and thus do not specify how to release them when an execution path becomes obsolete.

Generally, handling conditional execution paths depends on how service providers are allocated and when they block resources for service invocation. Service invocation solutions that do not block provider resources, do not need to release them if execution paths become obsolete but risk failure due to service invocation rejection. On the other hand, solutions that allocate providers prior to the execution of the composite require additional communication effort to release redundant service providers. Dynamic provider activation can potentially reduce this effort but needs to integrate an efficient way to block provider resources prior the invocation of a particular service to be applicable for dynamic ad hoc environments.

2.4.3 Common meeting point

Parallel execution paths, that merge in a single service, need a common meeting point. Recent work on the dynamic execution of business process (Zaplata et al., 2010) acknowl-

edges the need for a defined synchronisation point that can be chosen at runtime. The following analysis first revisits allocation and invocation techniques and then reviews election and consensus protocols to investigate how a common provider for a merge service can be identified dynamically.

2.4.3.1 Service allocation and invocation

In broker-based designs (cp. Section 2.3.1), a broker handles composition logic locally and represents the common meeting point. It merges parallel service flows naturally without additional synchronisation effort. However, the availability of brokers in transient networks is unlikely because none of the participants have initially sufficient overview of available services that would qualify them as brokers.

In static fragmentation solutions (cp. Section 2.3.2), the composite initiator finalises provider allocation. In particular, it selects the provider for the merge service and includes the decision in the fragments for the predecessors of the merge service. While this avoids negotiation during execution, topology changes may render such early decisions less optimal by the time the merge service is invoked. Existing solutions for dynamic provider activation (Yu, 2009b; Atluri et al., 2007) allocate a merge service prior to composite execution and are similarly prone to changes. Group-based allocation is more flexible towards system changes as it finalises provider allocation during composite execution. However, among the existing solutions, only OSIRIS (Schuler et al., 2004) addresses service flows and requires a reliable and globally available synchronisation node, which may not exist in dynamic ad hoc networks.

Among probing techniques, multi-path solutions (cp. Section 2.2.2) find, based on quality criteria, the optimal allocation for the composite which includes the optimal path between the merge service and its successors. However, confirming the final allocation decision with the selected service providers (Gu and Nahrstedt, 2006), introduces delays that may invalidate the optimum. A single-path solution (Wang et al., 2004) may confirm the allocation in each hop, except from the merge service which is selected by the composite initiator when it assembles the allocated parallel paths. While this reduces the allocation delay, it compromises the allocation optimum because the initiator, due to its limited system view, does not know how well a merge candidate connects to all

predecessors. Solutions in which the provider of a split service allocates the provider of the corresponding merge service (Yildiz and Godart, 2007) face the same issue. The split provider cannot foresee which yet-to-be-allocated service providers will actually interact with the merge service and how well they connect to their successor.

The composition solutions reviewed up to this point are limited in their applicability for dynamic ad hoc networks. They either assume that a reliable or well-informed composite participant exists to represent the common meeting point. Or, they compromise the flexibility or locality of the merge allocation. A way to increase the allocation flexibility and locality is to enable the predecessors of the merge service to allocate a common provider during execution. This, however, involves two challenges (Yildiz and Godart, 2007). First, multiple service providers have to agree on the same merge provider at runtime. Second, these synchronisation partners are mutually unknown as they, too, are bound only prior to their invocation. The synchronisation algorithm, proposed by Yildiz and Godart (2007), addresses these challenges with continuous path updates. Parallel paths update each other about their next allocation decision such that final synchronisation partners are mutually known to run an agreement protocol. However, as the algorithm has not been evaluated in dynamic ad hoc networks, its communication overhead and suitability for mobile service providers is unclear.

2.4.3.2 Election and consensus protocols

Election and consensus protocols for distributed systems relate to some extent to the problem of finding a common merge provider. Election protocols have to ensure that all participants in the network refer to the same new coordinator after the current one has disconnected (Park et al., 2009; Singh and Sharma, 2011). An election is led by a single decision maker who needs a global view of the system to consider the logical distance between each participant and the new coordinator (Higashi et al., 2011). Maintaining such a view is expensive and infeasible in true peer-to-peer networks (Gu and Nahrstedt, 2006). Further, identifying a common provider for a merge service involves only a subset of participants which can communicate in a more targeted way than through controlled network flooding (Park et al., 2009; Singh and Sharma, 2011). Consensus protocols achieve an agreement among agents if the opinion of all agents stabilises. Stochastic

consensus models (Roy et al., 2006) assume that agents are connected to each other and aware of all possible opinions. This thesis explores a communication-efficient way to establish such knowledge by mutually introducing synchronisation partners and revealing possible merge candidates.

2.4.4 Summary

Service flows introduce the challenge of managing concurrent data access, conditional execution paths, and common meeting points. Avoiding concurrent data access in the design is desirable because it does not require synchronisation during execution. If this is not possible, exchanging data access token (Yu, 2009a) is a viable solution for dynamic ad hoc environments because these tokens can be integrated with composition control flow messages.

For handling conditional execution paths, dynamic provider activation is a promising approach. It removes obsolete parts from the composite as it enfolds during execution and activates only required providers. However, existing solutions (Yu, 2009b; Atluri et al., 2007) do not specify when they confirm the commitment of the provider, block its resources for service invocation, and release provider resources in case a path becomes obsolete. Without such a strategy, the composite is prone to failure as the load and objectives change the provider availability dynamically.

Solutions for finding a common meeting point tend to be of limited applicability for dynamic ad hoc networks. The reviewed techniques either assume reliable and well-informed entities or compromise on the flexibility and locality when allocating it dynamically. Among the existing solutions, the synchronisation algorithm, proposed by Yildiz and Godart (2007), is promising because it increases flexibility and locality. Parallel execution paths continuously exchange updates about their allocation decisions such that the final providers can run an agreement protocol about a common successor. However, as this algorithm has not been evaluated for dynamic ad hoc networks, its impact on mobile service providers and service composites is unclear.

2.5 Communication

In dynamic ad hoc environments composite participants cannot rely on managed communication infrastructure and have to self-organise a network to collectively achieve a complex task. Their mobility and the nature of the wireless medium, however, make service composition challenging as network routes may change frequently and bandwidth is limited. This raises the question: How can service composition and ad hoc communication be aligned to deliver messages among composite participants efficiently? The reviewed approaches address this question with tuplespaces, content-based routing, and cross-layer designs.

2.5.1 Tuplespaces

Service invocation via tuplespaces (Martin et al., 2008; Fernández et al., 2010) enables composite participants to exchange messages while they may not be present in the network at the same time. They use a shared remotely-accessible container (i.e., a tuplespace) to hold service allocations and control flow messages. If a control flow message arrives in a tuplespace and matches a service allocation, execution of the corresponding service provider is triggered. While allowing for timely decoupling, tuplespaces imply a strong spatial dependency because a service provider and a service consumer must refer to the same tuplespace to communicate. This is challenging in networks that lack dedicated infrastructure. LIME (Murphy et al., 2001), is a middleware to coordinate applications that are distributed among mobile hosts. It breaks a tuplespace in multiple tuplespaces and deploys them on mobile hosts. When two hosts come in each other's communication range, their tuplespaces merge to one virtual space and increases the possibility for service allocations to meet and match control flow messages. However, the source and destination of a control flow message may never meet. CRUST (Artail et al., 2009), is an extension to LIME and offers clustering and routing capabilities for tuplespaces. It forwards tuples across tuplespaces to ensure that service allocations and control messages eventually meet. From the communication perspective, this approach converges with routing protocols for direct messaging in mobile ad hoc networks.

2.5.2 Content-based routing

CiAN (Sen et al., 2008) presents a content-based publish-subscribe routing scheme to provide for interaction among composite participants despite a dynamic and fragmented network. It assigns a unique totally-ordered service id to each required service which service providers use to deliver their service result to their successor or to subscribe for service data from their predecessor. These messages are then relayed by intermediaries based on their service id. For example, an intermediary forwards a subscription if its own service id is between the id of the source and destination of the message and smaller than the last intermediary's id. A message containing a service result is routed the same way, only in increasing manner. Eventually a subscription and a service result meet on one intermediary who dispatches the service result directly to its destination. The protocol is robust to topology changes because intermediaries store messages until they encounter a suitable router to forward the message. However, as service ids do not reflect location, a message may take a detour to reach its destination and requires more time and communication effort to be delivered.

2.5.3 Cross-layer design

Cross-layer approaches combine service discovery with underlying routing mechanisms to exploit routing traffic (Ververidis and Polyzos, 2008). For example, a modified route request carries a service discovery request while periodic messages for route maintenance contain service announcements (Varshavsky et al., 2005). This is to reduce the communication overhead of otherwise separately operating layers. Group-based service discovery (Chakraborty et al., 2006), on the other hand, uses the backward route of a service request to deliver a service discovery response. This way, a new route and the associated overhead for route discovery is only necessary if the original route breaks. While these approaches show a benefit in terms of communication overhead they are limited to service discovery and do not investigate how the need for interaction during service allocation and invocation could be integrated with routing protocols.

2.5.4 Summary

Communication approaches based on tuplespaces, content-based routing, and cross-layer designs all provide means to adapt to the dynamics and bandwidth constraints of dynamic ad hoc networks. They differ, however, in their awareness of a message being sent as part of a service composition process. Tuplespaces support the interaction between composite participants that are not present at the same time. Their extensions for dynamic networks, however, dispatch messages similarly to conventional routing protocols and regardless of their composition context. Content-based routing, as proposed by Sen et al. (2008), uses the order of required services in a composite to relay messages. While robust to topology changes, this approach may incur more overhead than address-based routing schemes as the content of the message may not correlate with the location of its recipient. Cross-layer designs send routing and composition content in one message and use routes established during service discovery to reduce the network traffic. Current cross-layer designs are, however, limited to service discovery and unaware of the potential of service allocation and invocation to further reduce the communication overhead of a composite. This thesis addresses this gap and examines how participants may derive actions (e.g., announcing a service or releasing resources) by following the progress of composites in their communication range rather than by expecting an explicit message that triggers them. This line of thought ties in with the need for "new search techniques that exploit [radio] broadcast to become aware of network services by simply listening (eavesdropping) to the traffic other nodes generate" (Mian et al., 2009).

2.6 Chapter summary

This chapter reviewed service composition solutions for their applicability in dynamic ad hoc networks and based on the main building blocks service discovery, allocation, and invocation as well as service flows and communication.

Service discovery investigated how dynamic provider information is maintained and kept up-to-date. The analysis showed the trade-off between proactively announcing services and revealing service information on demand. While proactive solutions do not

rely on additional infrastructure, they require high maintenance and periodic communication. Demand-based solutions do not need frequent updates but rely on a pre-established peer-to-peer structure for information replication. Periodic communication and pre-established structures do not map well to the characteristics of dynamic ad hoc networks. This motivates research on optimising demand-based service discovery that builds on the co-location of composite participants rather than pre-established structures and recognises demand without explicit service discovery requests.

Service allocation studied how the provider choice for composites is stabilised against the dynamics of the operating environment. Among the reviewed solutions, group-based and probing-based approaches are most applicable for dynamic ad hoc networks. Group-based solutions are flexible towards changes because they defer the final allocation decision to when a service needs to be invoked. However, they require additional effort for group management. Probing-based solutions explore the physical network of potential service providers and preserve the locality of the allocation. However, isolated from the service discovery phase, probing incurs additional communication and delays. A solution that efficiently preserves the locality and the flexibility of the provider allocation is desirable for dynamic ad hoc environments, however, does currently not exist.

Service invocation examined how provider resources can be efficiently blocked, consumed, and released. Distributed invocation approaches allow for direct interaction between composite participants and cater for the fact that in ad hoc environments the view of available services is limited for all participants. Among the decentralised approaches, dynamic provider activation is more suitable than static composite fragmentation because it consumes resources as the composite enfolds. However, existing activation solutions do not confirm the provider availability after allocation. This can lead to composite failure as a service provider may reject the service invocation due to limited resources or conflicting objectives. It remains to be investigated, how dynamic activation can block and release provider resources such that it stabilises the execution of a composite and at the same time maximises the general availability of service providers.

Service flows analysed how parallel execution paths can efficiently synchronise their access to concurrent data, conditional execution paths, and the allocation of a common meeting point. Avoiding concurrent data access during the composite design is most efficient because it does not require synchronisation at runtime. Otherwise, passing a data access token is a viable option for dynamic ad hoc environments because it can be integrated with control flow message during composite execution. For conditional execution paths, dynamic provider activation is desirable because it removes obsolete parts as the composite is executed and activates providers that are actually needed. However, without a strategy of when to confirm the provider availability, existing solutions are not readily applicable. For a common meeting point that merges parallel execution paths into a single thread of control, existing solutions tend to compromise the locality or the flexibility of the allocation. Allocating the meeting point immediately prior to its invocation allows for high flexibility and locality. In an existing solution, parallel execution paths synchronise their allocation decision for each service toward the common meeting point. The implications of such an approach in dynamic ad hoc networks have not been evaluated and motivate further investigations.

Communication examined how composite messages can be efficiently delivered in ad hoc networks. Cross-layer designs show high potential to reduce network traffic because they integrate networking and composition aspects in their techniques for message delivery. However, current solutions are limited to service discovery. This raises the question of how service allocation and invocation can be integrated with lower communication layers to reduce the message overhead for service composition.

Wrap-up Few approaches integrate all aspects of service composition and the KIVIAT diagram in Figure 2.2 illustrates those solutions that address most of them. The diagram has seven dimensions representing the criteria that were used throughout the chapter to compare different solutions. Each dimension has three levels which, outgoing from the centre, increase in their applicability for dynamic ad hoc networks. The coloured lines depict for the composition solutions how they address each dimension. The diagram highlights three important findings of the state of the art review:

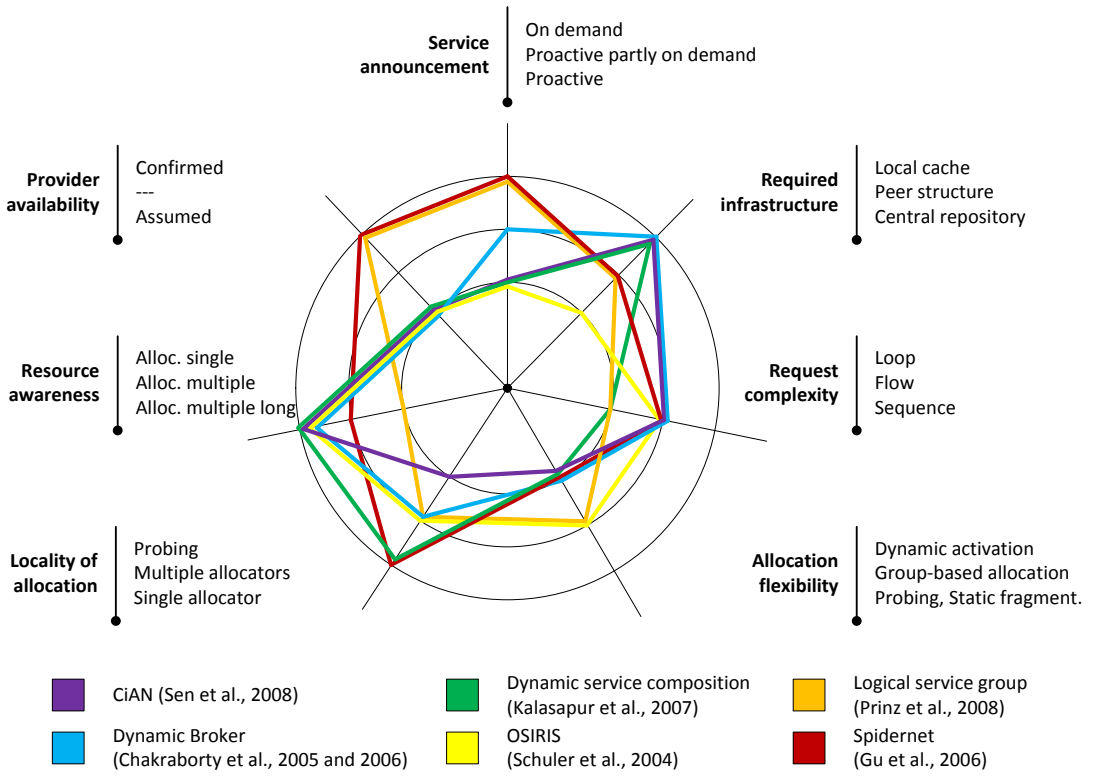


Fig. 2.2: *The KIVIAT diagram shows how solutions, that cover most aspects of service composition, address seven reference criteria. Each criterion has three levels which increase in applicability for dynamic ad hoc networks, the further they are away from the centre. Among other things, the diagram highlights that allocation flexibility based on dynamic allocation has not been adequately addressed.*

- The availability of a provider for service invocation tends to be assumed rather than confirmed (cp. provider availability), in particular, by solutions that apply local caching (cp. required infrastructure).

- Solutions that allocate a single provider per required service are resource-aware (cp. resource awareness), however, tend to compromise allocation flexibility (cp. allocation flexibility).

- Dynamic activation (cp. allocation flexibility), i.e., blocking, consuming, and releasing provider resources that are actually required by a composite, has not been adequately addressed.

As an aside, the diagram shows that loops of repetitive execution paths (cp. request complexity) have not been covered, however, this was outside the scope of the analysis and is not further addressed. This thesis draws inspiration from deferring the final provider allocation to when a required service needs to be invoked and activating providers that are actually needed. These concepts are proposed by group-based allocation and dynamic activation techniques. In contrast to existing solutions, the approach, described in this thesis, lets a service provider search for its successor after it executed its allocated service. This way, the approach aims to obtain the locality, resource-awareness, and provider availability of the composite without pre-existing structures for service discovery and information replication.

Chapter 3

Design

This chapter returns to the characteristics of mobile ad hoc networks and corresponding composition challenges to frame the problem and solution that are addressed in this thesis. Figure 3.1 outlines the structure of this chapter: First, the chapter introduces the design objectives for a novel service composition protocol (cp. Section 3.1). Then the system model defines service composites and service provision to formalise service composition as a graph-mapping problem and to specify the assumptions that scope the contribution of this thesis (cp. Section 3.2). The subsequent section on design decisions discusses alternatives related to service discovery, allocation, and invocation as well as service flows and communication aspects to motivate the choices for the novel protocol (cp. Section 3.3). The section thereafter explains how the proposed solution integrates the design decisions (cp. Section 3.4). It is subdivided in three parts: The first part shows the basic protocol and how it handles sequential service requests. The second part describes the protocol extensions to support parallel service flows. The third part provides details on the cross-layered communication approach. The chapter concludes with a summary of the key design aspects (cp. Section 3.5).

3.1 Design objectives

Chapter 1 introduces opportunities and challenges for service composition in mobile ad hoc networks. In the scenario, Adam's noise tracking task incorporates other mobile devices to produce an audio snapshot of his surroundings. In open environments, com-

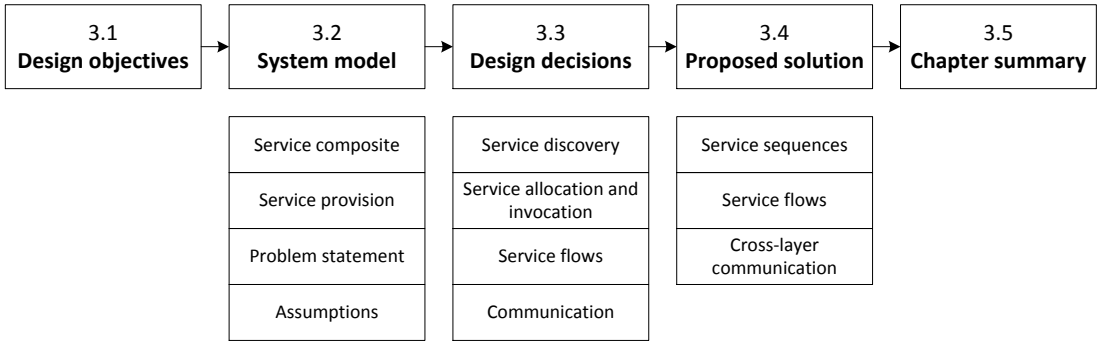


Fig. 3.1: ***Design overview*** *The sections design objectives and system model define the problem space that is addressed in this thesis. The sections design decisions and proposed solution describe the novel opportunistic composition protocol. The summary highlights the key design aspects.*

plex tasks are likely to rely on multiple data sources to compensate uncertainty and to improve the quality of the final result. The data sources first work in parallel, each sampling and pre-processing data, and then merge their parallel execution paths into one for further processing. In Adam's case, three or four audio sampling services may merge their results to increase the quality of the final audio snapshot. The collaborating devices, however, are autonomous, communicate wirelessly, and are part of a transient network which lacks dedicated composition infrastructure and is likely to fail completing a complex request. These system characteristics correspond to the challenges identified in chapter 1 on page 4 and introduce new design objectives for service composition in highly dynamic environments. To address the challenges a service composition must:

- **Design objective 1: Proactively gather information**

  In transient networks that evolve and dissolve based on demand (challenge 1), composite participants must be prepared to gather service information on their own initiative because a decentralised management of cached service provider data may not be established.

- **Design objective 2: Self-organise service composition**

  The lack of dedicated composition infrastructure (challenge 2) requires composite participants to organise the discovery, allocation, and invocation of services among themselves because an entity that manages these tasks and has global view of available services does not exist.

- **Design objective 3: Support parallel service flows**

  The absence of composition infrastructure (challenge 2) also implies that composite participants must coordinate the synchronisation of parallel execution paths themselves because there is no dedicated entity that can reliably handle this task.

- **Design objective 4: Reduce communication**

  Because of the narrow bandwidth of wireless communication links (challenge 3), composite participants must limit the exchange of messages to lower the potential for radio interference.

- **Design objective 5: Enable short and localised interaction**

  Service composites should reduce the impact of mobility (challenge 4) by completing service discovery and invocation before the service provider moves and by allocating providers that are close-by.

- **Design objective 6: Obtain provider commitment**

  Due to the participation autonomy of service providers (challenge 5), service composites require the providers' explicit commitment to engage in a particular composite. Each commitment indicates that the provider blocks enough resources to handle part of the composite request.

Collectively, these design objectives target the failure potential of service composites (challenge 6). Strategies for additional failure prevention and recovery are outside the scope of this thesis which examines the composition procedure itself to reduce failure. Further, while the first three design objectives address the implementation of the service composition in mobile ad hoc environments, the last three aim at reducing the composite's exposure to the inherent unreliability of these environments. Combined the objectives highlight the tension between resource-intensive composition tasks on the one hand and the resource-constrained operating environment on the other hand. The system model in the next section formalises this tension and introduces the terminology and assumptions to design a possible solution.

3.2 System model

The applications under consideration in this thesis reside in dynamic ad hoc environments. They build on components that provide and require services. Complex application requirements are modelled by a composite description that defines the behaviour and order of required services. At runtime a composition algorithm processes the description. It incrementally allocates and invokes service providers to return in the end either a valid composition result or a failure notification. The system model first defines service composite (cp. Section 3.2.1) which represents a composite description. It then defines service provision which represents available services in an ad hoc network (cp. Section 3.2.2). These are the prerequisites to formally describe the problem (cp. Section 3.2.3) and to define the assumptions (cp. Section 3.2.4) for the design of the novel composition protocol.

3.2.1 Service composite

A service composite specifies required services and the data and control flow between them. A service composite can be represented as a directed graph in which the vertices correspond to the required services and the directed edges to the control and data flow. This thesis focuses on composites that do not contain repetitive parts and models a service composite as a directed acyclic graph G_c:

Definition 1 *A service composite is a directed acyclic graph* $G_c = (V_c, E_c, sid, pid, src, dst, eval)$ *with a set of vertices* V_c *for required services and a set of labelled edges* E_c *for the conditional data and control flow between the services. The functions* $sid : V_c \rightarrow \mathcal{N}$ *and* $pid : V_c \rightarrow \mathcal{N}$ *return the unique service and provider id of a required service. The functions* $src : E_c \rightarrow V_c$ *and* $dst : E_c \rightarrow V_c$ *return the unique source and destination of an edge. The function* $eval : E_c \rightarrow \{0, 1\}$ *returns whether the condition to enable the edge is true.*

An example for a service composite graph is depicted in Figure 3.2. The graph handles the complex task of classifying audio data. It builds on services that sample raw audio data, extract features, aggregate output from different sources, and classify features.

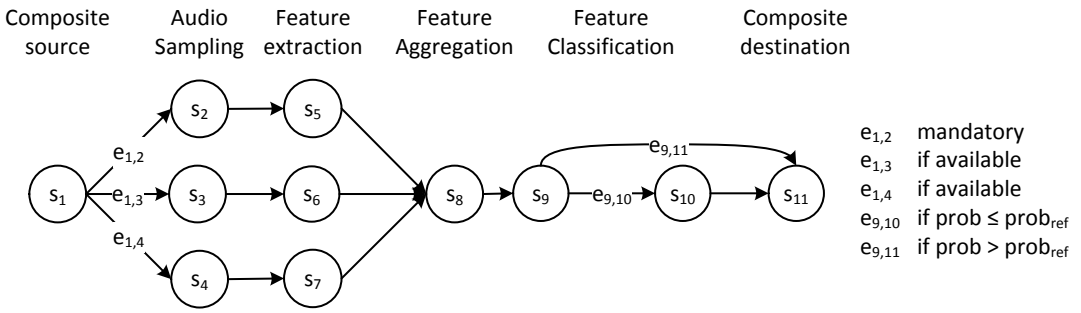


Fig. 3.2: ***Composite graph for audio classification*** *The composite graph extracts and aggregates features of raw audio data to classify them with a certain probability e.g., as traffic or music. The composite retains the services of multiple audio sampling services if available and commissions a second classifier if the probability of the first classification is insufficient.*

The data and control flow splits in parallel execution paths if multiple independent audio sources are available and continues in two possible ways depending on the output quality of the first classifier. A service s in the composite graph G_c has a defined set of predecessors, successors, and synchronisation partners:

Definition 2 *The predecessors of a service s is the set of services, $pred(s) = \{v \in V_c | \exists e \in E_c, src(e) = v \wedge dst(e) = s\}$, for which s must wait to finish until it can execute.*

Definition 3 *The successors of a service s is the set of services, $succ(s) = \{v \in V_c | \exists e \in E_c, src(e) = s \wedge dst(e) = v\}$, which s needs to notify to trigger their execution.*

Definition 4 *The synchronisation partners of a service s is the set of services, $partner(s) = \{v \in V_c | \exists m \in V_c, v \in pred(m) \wedge s \in pred(m) \wedge (v \neq s)\}$, which s must synchronise with to find a common provider for a merge service m.*

For example in Figure 3.2, the predecessors, successors, and synchronisation partners of service s_8 are $pred(s_8) = \{s_5, s_6, s_7\}$, $succ(s_8) = \{s_9\}$, and $partner(s_8) = \emptyset$. A composite graph G_c has a unique start service $s_{start} \in V_c$ which does not have any predecessors $pred(s_{start}) = \emptyset$ and a unique end service $s_{end} \in V_c$ without any successors $succ(s_{end}) = \emptyset$. In Figure 3.2, the composite graph specifies $s_{start} = s_1$ and $s_{end} = s_{11}$.

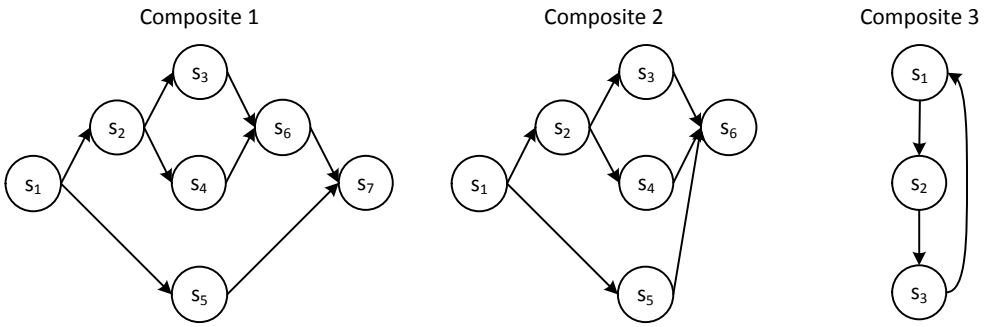


Fig. 3.3: ***Valid and invalid composites*** *This thesis focuses on acyclic well-structured composites (composite 1). It does not address composites that are unstructured (composite 2) or contain repetitive parts (composite 3).*

The structure of a composite graph that is studied in this thesis complies with the following Extended Backus-Naur Form in which each service $s_i \in V_c$ occurs only once:

$$\begin{aligned} < c > \quad ::= \quad & s_1|...|s_n| \\ & < c > seq < c > | \\ & < c > split(< c >, [< c >])merge < c > \end{aligned}$$

According to this definition, a composite graph may be a sequence of services or a service flow that splits and merges multiple execution paths. In case of a service flow, the composite graph must be well-structured i.e., all parallel paths that originate from one split service must terminate in one corresponding merge service and all parallel paths that terminate in one merge service must originate from one corresponding split service (Polyvyanyy et al., 2010). For example in Figure 3.3, composite 1 is well-structured, while composite 2 is not because the paths that terminate in the merge service s_6 originate from two different split services s_2 and s_1. Inspired by the work of Atluri et al. (2007), the control logic for split and merge services is modelled implicitly by the number of edges in E_c that evaluate to true during the execution of the composite. Initially, these edges have OR or XOR semantics depending on whether their conditions are mutually exclusive. For example in Figure 3.2, edge $e_{1,3}$ or $e_{1,4}$ or both may evaluate to true whereas either $e_{9,10}$ or $e_{9,11}$ will be true. Service s_1 represents an AND-splitting service, if edge $e_{1,2}, e_{1,3}$ and $e_{1,4}$ are true. Service s_8, on the other hand, is an OR-merging service and must be prepared to handle any subset of predecessors until it is actually invoked.

3.2.2 Service provision

A service provider grants access to locally hosted services and executes the control logic that combines basic services to complex composites. The provider communicates with other composite participants wirelessly, either directly if they are in its direct communication range, or indirectly via intermediaries that relay messages. Service providers are mobile and autonomous which implies a frequently changing network and service topology. Service provision depends on whether a service is currently offered and on whether there is a network link to its provider. This thesis uses the concept of time-varying graphs (Casteigts et al., 2011; Santoro et al., 2011) to model the dynamics of service provision. A time-varying graph defines, in addition to the collection of vertices and edges, a presence function that indicates whether a given edge is available at a given time. In such a graph, a vertex represents a service that is offered by a particular service provider. An edge between two vertices represents the network link between the two service providers. At times this link may be disabled indicating that the two providers are disconnected. Service provision can be formally defined as follows:

Definition 5 *Service provision is a network of provided services and represented as an undirected time-varying graph* $G_p = (V_p, E_p, T_p, \rho)$ *over a graph lifetime* T_p*. The set of vertices* V_p *corresponds to services offered by particular providers. The set of edges* E_p *represents the network links between service providers. The presence function* $\rho : E_p \times T_p \rightarrow \{0, 1\}$ *returns for a given edge and given date whether the edge is enabled and two service providers are connected.*

The evolution of service provision can be modelled as a sequence of static sub-graphs. Each sub-graph represents the interval in which all enabled edges remain enabled. Changes occur from one sub-graph to the next. The edges in E_p evolve over time and get disabled in two cases: First, the service provider temporarily disables the provision of a service. Second, the network link between two service providers breaks because one of them or an intermediary has moved, failed, or powered-off. Figure 3.4 illustrates an example for the evolution of the service provision graph G_p.

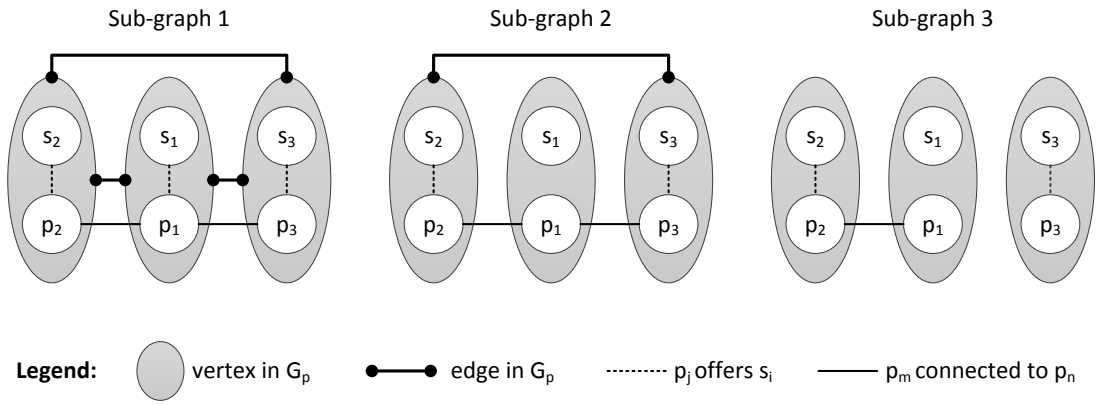


Fig. 3.4: ***Dynamic service provision*** *In sub-graph 1 the services s_1, s_2, and s_3 are connected. In sub-graph 2 provider p_1 disables its support for s_1 but remains as an intermediary for the network link between p_2 and p_3 such that s_2 and s_3 stay connected. In sub-graph 3 provider p_3 disconnects from p_1 and disables the edge between s_2 and s_3. Hence, there is only limited time to compose a service sequence of s_1, s_2, and s_3.*

3.2.3 Problem statement

Service composition in dynamic ad hoc networks can be modelled as the problem of finding a mapping between the sequence of service provision graphs G_p and the composite graph G_c such that the edges between two provided services are enabled for the time they interact.

3.2.4 Assumptions

The system model makes the following assumptions to scope the design of the novel composition protocol:

- Network links between service providers are bi-directional and allow for symmetric connectivity. If provider i is connected to provider j, then provider j is connected to provider i.
- The network among service providers evolves over time as the providers appear and disappear in each other's communication range. These dynamics are modelled by enabling and disabling network links. When a provider disconnects from the network it is considered to have left.

- From the communication perspective, all service providers are cooperative and attend to their routing responsibilities. They share their resources if they can and are not malicious.

- Service providers are resource-constrained and engage in a limited number of compositions simultaneously. If this number is exceeded the service provider ignores any further composite requests.

- Once a service provider agrees to participate in a particular composite, it does not revoke its commitment. Unanticipated failure is not considered as this would require additional recovery strategies that are outside the scope of this thesis.

- A composite request is a well-structured acyclic directed graph that represents a template of required services. Graphs with unstructured parallel service flows or repetitive parts are not considered.

- Composite participants use a common language and rely on a global ontology to specify offered and required services.

The assumptions complete the description of the system model which together with the design objectives (cp. Section 3.1) cover the problem space of this thesis. The next section turns to the solution space and examines different alternatives to motivate the design decisions for an opportunistic composition protocol.

3.3 Design decisions

For service composition in mobile ad hoc environments there are different ways to organise the service discovery, allocation, and invocation, handle parallel service flows, and manage the communication between service providers. The following discussion takes these activities to structure the design decisions and to show how these decisions address the design objectives that were identified in Section 3.1.

3.3.1 Service discovery

For the discovery of available services, service providers may announce their capabilities through proactive unsolicited advertisements or by responding to service discovery requests. The former approach eases service discovery once the network is established but needs to keep cached provider information up to date. The latter solution does not incur such maintenance overhead but delays the composition because service providers first need to be identified. With regard to service discovery the first two design decisions are as follows:

Design decision 1: Demand-based service announcement
The proposed protocol combines both discovery approaches. Service providers respond to service discovery requests and proactively announce a service if they observe its necessity in a particular composite. This demand-based strategy limits the information exchange to provider data that a composite actually needs and reduces communication. This addresses design objective 4. Further, the strategy targets design objective 1 as it is independent from a distributed service registry that may not exist in transient networks.

Design decision 2: Resource-blocking service announcement
In conjunction with design decision 1, the issuer of a service announcement already blocks sufficient resources because the announcement is targeted at the next required service of an actual composite and the allocation decision is anticipated soon. This avoids additional communication to confirm the participation commitment and contributes to design objective 6. At the same time, however, this strategy requires the release of redundant service providers after the allocation is complete. For reducing this overhead the approach followed in this work is to limit proactive service announcements and efficiently disseminate multicast messages.

The next section discusses different ways of organising the allocation and invocation of required services.

3.3.2 Service allocation and invocation

Service allocation and invocation distinguishes three approaches: First, broker-based designs, as in dynamic brokers (Chakraborty et al., 2005), appoint a single entity that handles both tasks in a centralised manner. Second, semi-centralised solutions, as in CiAN (Sen et al., 2008), rely on a single entity to allocate all service providers and thereafter distribute service invocation among the allocated providers. Third, decentralised approaches, as in logical service groups (Prinz et al., 2008), distribute the responsibility for service allocation and invocation within the network of service providers.

The existence of composition brokers is unlikely in transient environments because a network is established based on a request and none of the participants has initially sufficient overview of available service providers. In addition, centralised service invocation prevents direct interaction between consecutive service providers and increases communication. Semi-centralised and decentralised solutions manage provider interaction without a mediator, however, come with their own trade-offs.

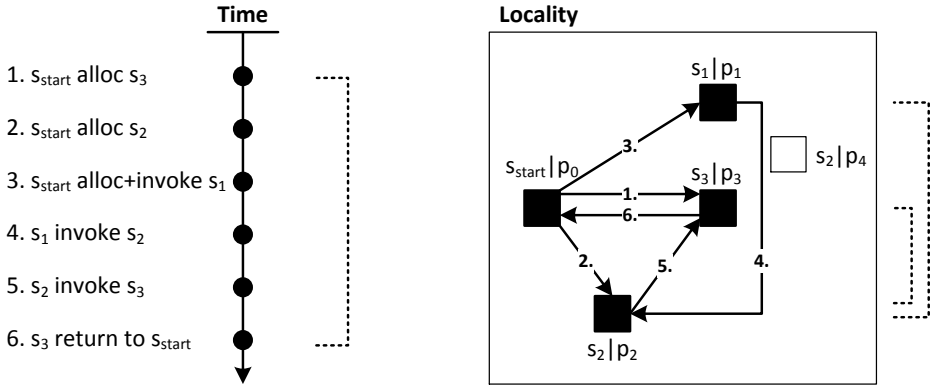
In semi-centralised solutions the composite initiator allocates all required services. Its limited view of how consecutive service providers actually connect, increases the allocation's potential for long routes and network failure as each intermediary may move out of transmission range. For example, in Figure 3.5a step 2, s_{start} allocates the close-by provider p_2 for service s_2. However, from the perspective of provider p_1 who has to invoke s_2 in step 4, provider p_4 would have been closer.

In decentralised designs one provider allocates the next by examining its surroundings first and keeping the routes short. However, the interaction between consecutive service providers stretches over an extend period of time in which even short routes may break. For example, in Figure 3.5b, the route between provider p_3 and p_4 may be short but must remain intact from when s_3 allocates s_2 (step 2) to when s_2 invokes s_3 (step 5).

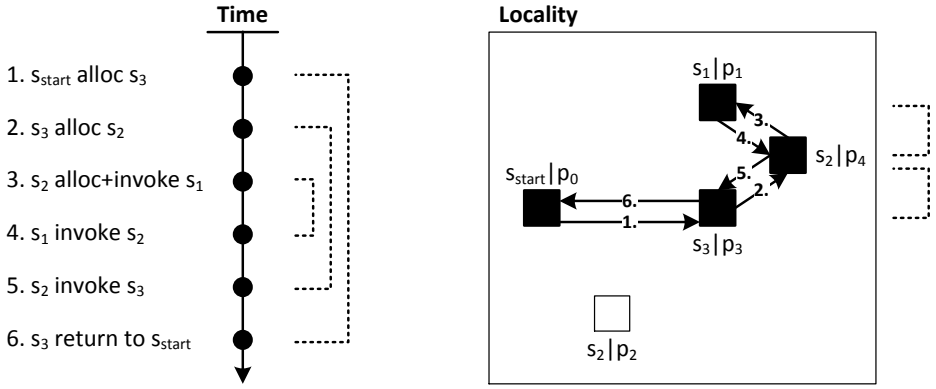
From the analysis of the semi-centralised and decentralised approach emerges another alternative, namely decentralised interleaved composition (cp. Figure 3.5c), which is the approach followed in this work and represents the third design decision:

Design decision 3: Decentralised interleaved composition

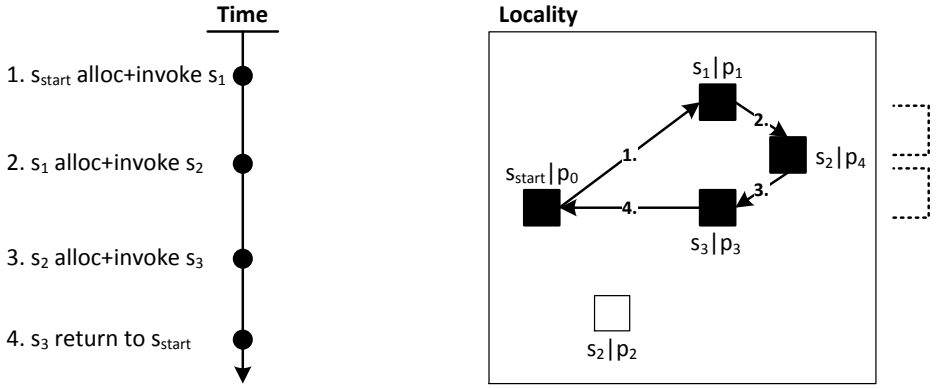
The proposed protocol distributes and interleaves the allocation and invoca-



(a) Semi-centralised



(b) Decentralised



(c) Decentralised interleaved

Fig. 3.5: ***Service allocation and invocation alternatives*** *for a sequential request s_1 to s_3. Perpendicular braces show for Time how many routes are used twice and must stay intact for an extended period of time (the fewer and shorter, the better). For Locality the braces show the distance between consecutive service providers (the shorter, the better).*

tion of required services. The composite travels hop-by-hop from one provider to the next to assign and execute a required service. Once a provider has been allocated, it is immediately invoked. This addresses design objective 5 and allows for short and localised interactions between consecutive providers.

Collectively, the first three design decisions aim at realising design objective 2 because they do not require dedicated composition infrastructure and allow for self-organisation among the composite participants. Further, they build the basis for design decisions related to service flows which are discussed next.

3.3.3 Service flows

For the support of parallel flows the design needs to decide who identifies a common provider for a merge service: a single entity (the composite initiator or corresponding split provider) or the immediate predecessors of the merge service. While a single entity is limited in its system view and may not choose the candidate that is close-by to all synchronisation partners, it requires no additional synchronisation effort. On the other hand, immediate predecessors of a merge service better assess their connectivity to a merge candidate but need to be aware of each other's identity to run an agreement protocol. Once a service provider is ready to synchronise, it may only know the partner service but not the partner's provider id. The discovery of that provider would incur some form of controlled network flooding. Alternatively, a parallel path may stepwise update its partner paths about each allocation decision towards the synchronisation partners. Both approaches increase the communication overhead since the partner identification is separate from running an agreement protocol afterwards. For the novel composition protocol to support service flows the design decisions are as follows:

Design decision 4: Late merge allocation

The predecessors of a merge service are in the best position to allocate a provider that is close-by for all synchronisation partners and to support parallel service flows as outlined by design objective 3. In line with the decision for decentralised hop-by-hop service composition, a merge service is treated as any other service and allocated by its predecessors. The protocol allocates

parallel paths in isolation to avoid repeated allocation updates until synchronisation partners are reached and to reduce communication as required by design objective 4. The protocol uses the composite structure to deliver synchronisation messages to initially unknown synchronisation partners. This way the identification process already carries the possible merge candidates as basis for an agreement.

Design decision 5: Composite reduction
The decision to interleave service allocation and invocation comes with the benefit of immediately evaluating the conditions that decide whether a request splits into parallel execution paths or not. If the condition of an edge in the composite description evaluates to false, the edge gets disabled and the composite is reduced accordingly. This avoids allocating resources for obsolete paths and reduces communication as outlined in design objective 4.

Having covered design decisions with regard to the basic composition procedure and parallel service flows, the next part attends to design decisions that address the communication aspect of a service composition.

3.3.4 Communication

In highly dynamic networks there is the notion of end-to-end and opportunistic message delivery. End-to-end communication assumes the source and destination of a message are available at the same time and connected via a network path. Opportunistic communication relaxes this requirement. This means if a composite allocated a service provider that is currently not available, the networks stores and carries any messages for that provider until it is available again. The two alternatives lead to the next design decision:

Design decision 6: End-to-end communication
The protocol uses end-to-end message delivery because its design is based on the idea of tapping the potential of those service providers that are available when the next allocation decision is to be made. In other words, the composite is not aware of providers that may become available at a later stage.

This design decision corresponds to design objective 5 and allows for short and localised interaction.

Service composition can be regarded as part of the application layer which typically has an interface to the networking layer to send and receive messages but lacks access to cached topology information. Responsible for communication and routing related tasks, the network layer shields the application layer from any messages that are not destined for it and drops them silently. In wireless communication, a participant's physical layer receives all messages that are issued in its transmission range. However, the rigorous distinction between network and application layer keeps potential service providers from following the progress of composites around them. The design of the proposed protocol changes this as follows:

Design decision 7: Observable composition messages
The protocol declares composition messages to be observable such that the network layer delivers them to the application layer regardless of whether a participant is a primary recipient or not. This strategy risks flooding the application layer with unnecessary messages and is avoided by standard communication stacks. However, in this case it is useful because service providers can announce services proactively or release blocked resources by observing a composition instead of exchanging dedicated messages which reduces communication as outlined by design objective 4.

Design decision 8: Multicast messaging
The release of redundant service providers requires the same message to be delivered to multiple destinations. Instead of handling each recipient separately, all destinations are included in one message. Based on the broadcast nature of the radio medium, this message is posted only once per hop for the part of the route that is the same for all destinations and reduces communication as outlined by design objective 4.

Figure 3.6 summarises the design decisions that constitute the novel opportunistic composition protocol and maps them to the design objectives mentioned earlier in this

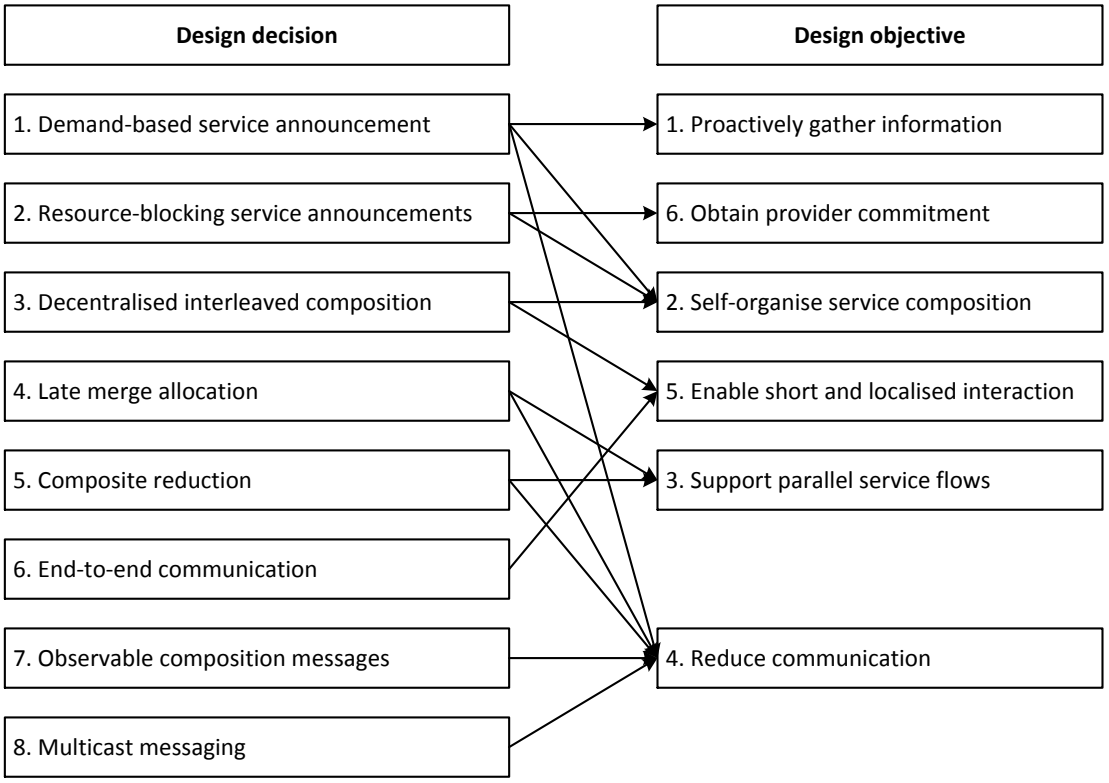


Fig. 3.6: ***Design decisions*** *Eight design decisions constitute the novel opportunistic composition protocol and address the six design objectives of this thesis.*

chapter. The next section continues to provide details about the proposed solution and explains how the design decisions materialize in the protocol.

3.4 Proposed solution

This thesis describes a protocol for opportunistic service composition that runs on mobile service providers. A finite state machine models the protocol and describes the decentralised interaction among composite participants who receive composition messages and derive state transitions accordingly. For ease of presentation, the description of the protocol distinguishes between a composite initiator that issues a complex service request but does not provide any services and service providers that have only enough resources to cover one service at a time. Note, that the protocol itself is free of such restrictions. The protocol description involves three parts. The first two parts cover

the details of the finite state machine by means of scenarios, first with focus on service discovery, allocation, and invocation for service sequences (cp. Section 3.4.1), then with focus on extensions for parallel service flows (cp. Section 3.4.2). The third part explains the cross-layered communication approach (cp. Section 3.4.3).

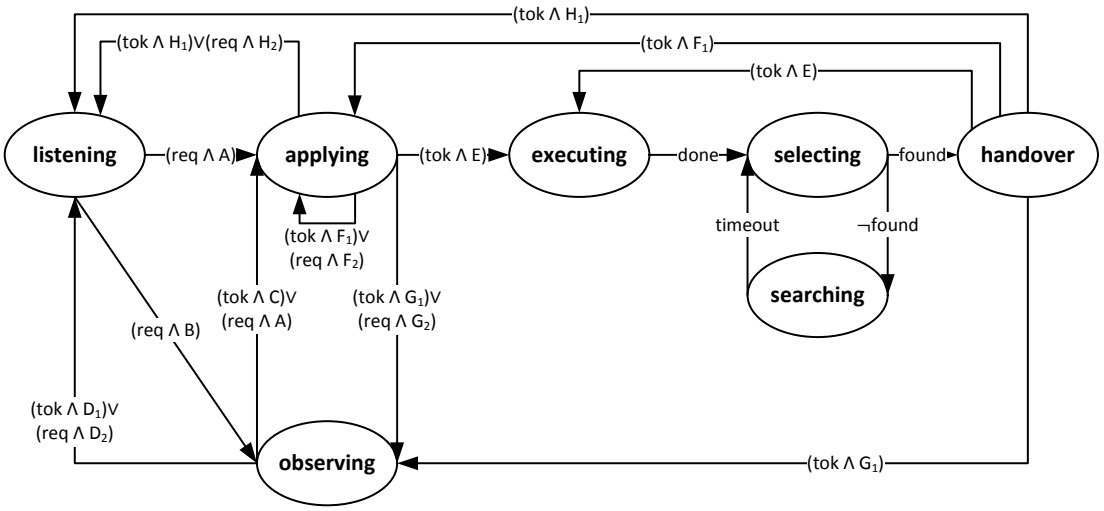
3.4.1 Service sequences

In the proposed protocol for service sequences (cp. Figure 3.7) service providers start and end in the listening state while a composite initiator starts in the searching state and ends in the listening state. The arrival of a composition message triggers the transition between the protocol states. A composition message contains a unique composite identifier to distinguish messages that concern different composites. Three types of composition messages are relevant for the composition of service sequences:

- **Composite request (req)** represents a complex service request. It contains the composite graph and a pointer to the service that is required next.
- **Token (tok)** represents an allocation decision. It holds the service to execute, its corresponding provider, and service input data.
- **Service announcement (ad)** represents the participation commitment of a service provider. It encloses the service for which the commitment is made.

Generally, a potential composite participant updates its local version of the composite graph and its knowledge about the network topology when it receives a composition message, regardless of whether the message is destined for it or just passes by. In addition to the above message types, the protocol uses the following notation:

cid(msg)	Composite identifier of message msg
cid(self)	Composite identifier to which the participant commits
pid(s)	Provider identifier for provider of service s
offerNxt	Participant offers next required service
offerRemaining	Participant offers a remaining service
adObsolete	Participant's service announcement is obsolete



(a)

A	offerNxt
B	¬offerNxt ∧ offerRemaining
C	cid(self)==cid(tok) ∧ offerNxt ∧ inRange
D_1	cid(self)==cid(tok) ∧ ¬offerNxt ∧ ¬offerRemaining
D_2	cid(self)==cid(req) ∧ ¬offerNxt ∧ ¬offerRemaining
E	cid(self)==cid(tok) ∧ allocated
F_1	cid(self)==cid(tok) ∧ ¬allocated ∧ offerNxt ∧ inRange
G_1	cid(self)==cid(tok) ∧ ¬allocated ∧ ¬[offerNxt ∧ inRange] ∧ offerRemaining
H_1	cid(self)==cid(tok) ∧ ¬allocated ∧ ¬[offerNxt ∧ inRange] ∧ ¬offerRemaining
F_2	cid(self)==cid(req) ∧ adObsolete ∧ offerNxt
G_2	cid(self)==cid(req) ∧ adObsolete ∧ ¬offerNxt ∧ offerRemaining
H_2	cid(self)==cid(req) ∧ adObsolete ∧ ¬offerNxt ∧ ¬offerRemaining

(b)

Fig. 3.7: ***Protocol for service sequences****. The state diagram (a) shows how composite participants change their system state based on the labelled transitions (b). Service providers start and end in the listening state. A composite initiator starts in the searching state and ends in the listening state.*

allocated	Participant is allocated as the next composite controller
inRange	Participant is in transmission range of the next controller
done	Service execution finished
timeout	Search for a service provider timed out
found	Provider for next required service was found

Next, three scenarios explain how the finite state machine integrates the first three design decisions, namely demand-based service announcement (design decision 1), resource-blocking service announcements (design decision 2), and decentralised interleaved composition (design decision 3). Each scenario refers to the same composite request which starts and ends with the composite initiator and contains a sequence of two services, s_1 and s_2. Further, each scenario depicts a network graph to show how the corresponding composite participants connect, a composite graph to show how the request evolves, and a sequence diagram to show how the participants interact.

3.4.1.1 Scenario 1: Basic protocol

In the basic protocol (cp. Figure 3.8) the initiator starts in the searching state and issues its composite request. Once service provider p_1 responds and the search times out, the initiator allocates p_1 as the provider for s_1 and hands over the composition control by sending a token. Based on this token, the initiator infers that it is not allocated for the next service and does not offer any required services such that it transitions to listening and waits for the composite result:

$$H_1\text{: } \text{cid(self)==cid(tok)} \wedge \neg\text{allocated} \wedge \neg[\text{offerNxt} \wedge \text{inRange}] \wedge \neg\text{offerRemaining}$$

Provider p_1 demonstrates a typical sequence of state changes for service providers. First, p_1 listens for a composite request and applies if it offers the next required service:

$$A\text{: } \text{offerNxt}$$

then executes the services if it receives a token that indicates that p_1 is the allocated provider and thus the next composite controller:

$$E\text{: } \text{cid(self)==cid(tok)} \wedge \text{allocated}$$

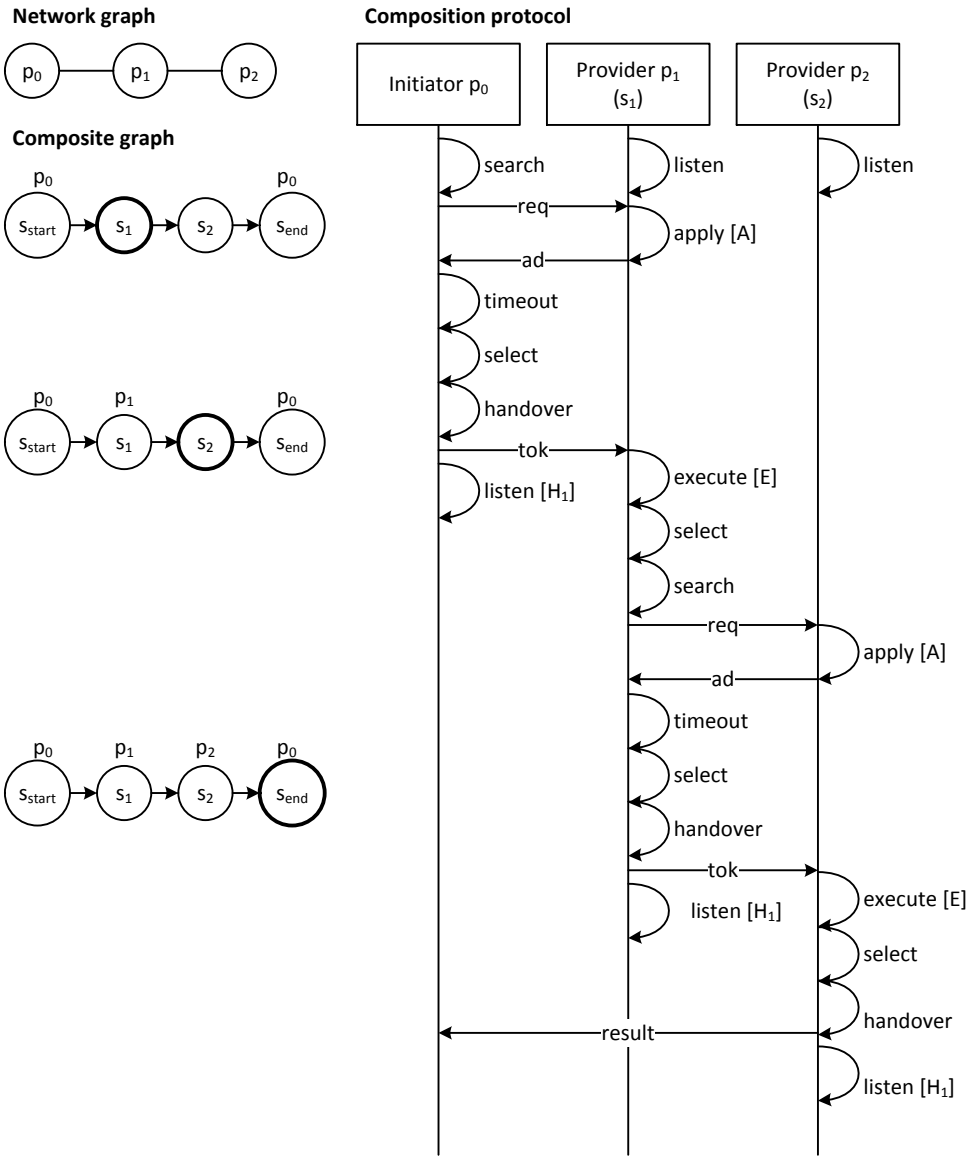


Fig. 3.8: ***Scenario 1 Basic protocol*** *The network graph and initial composite graph define the scenario setting. The composition protocol illustrates the interaction between composite participants. Its annotations in square brackets refer to the conditions stated in Figure 3.7b. The initiator allocates provider p_1 to the required service s_1. After executing service s_1, provider p_1 assigns provider p_2 to cover service s_2. Provider p_2 returns the final result to the initiator.*

Thereafter, provider p_1 alternates between selecting and searching until it finds a provider for the next required service. Finally, provider p_1 hands over the composition control by sending a token to its successor and goes back to the listening state if it is not allocated for the next service and cannot further contribute to the composite. Provider p_2 continues with executing s_2 and hands the result over to the initiator without searching because the initiator's provider id was part of the composite graph that p_1 send in its request.

The scenario demonstrates decentralised interleaved service composition (design decision 3) as each service provider assigns its successor (decentralised) after it executed its own service (interleaved). Further, with the announcement of a service, the provider blocks local resources (design decision 2). The announcement is directed to the controller of a particular composite and only valid for the enclosed composite id. It cannot be reused for other composites. This way the protocol supports explicit resource allocation and avoids composite failure due to provider overload.

3.4.1.2 Scenario 2: Proactive service announcement

The setup in Figure 3.9 differs from the previous scenario in that p_2 is now in range of the initiator and receives the initial service request. While the initiator and p_1 act as before, p_2 starts observing the composite because it does not provide the currently required service but the one thereafter:

$$B\text{:} \quad \neg\text{offerNxt} \wedge \text{offerRemaining}$$

When the initiator sends the token to p_1, p_2 overhears that because composition messages are observable by anyone in the sender's transmission range (design decision 7). From the token p_2 derives that s_2 is required next and proactively announces itself (design decision 1) since it is in the range of the new controller p_1:

$$C\text{:} \quad \text{cid(self)==cid(tok)} \wedge \text{offerNxt} \wedge \text{inRange}$$

After p_1 is done executing s_1 it does not have to search for a successor due to the proactive announcement of p_2 and hands over to p_2. The protocol restricts the number of unsolicited service announcements by the *inRange* condition. Otherwise, if a request must be sent multiple times each time with a bigger search radius to find a service provider,

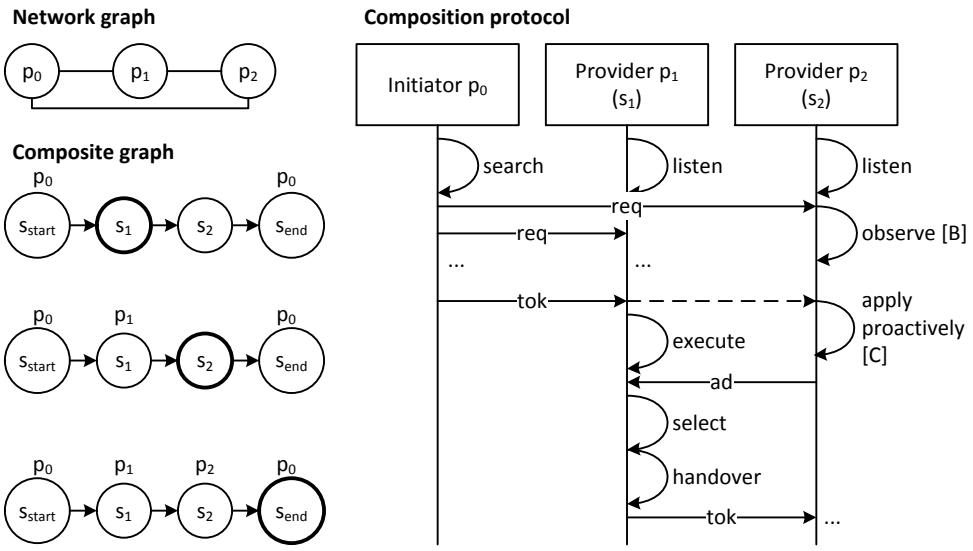


Fig. 3.9: ***Scenario 2 Proactive service announcement*** *Provider p_2 is in transmission range of the composite initiator, receives the initial request and observes until it infers from the overheard token (perpendicular arrow) that its service s_2 is required. Then p_2 proactively announces itself and spares p_1 the search for a suitable provider for s_2.*

the subsequent token must travel multiple hops to the new controller. All participants en route to the new controller could potentially announce themselves proactively. The new controller, however, only needs one successor and must release redundant advertisers. This overhead would outweigh the benefit of announcing services proactively.

Generally, potential applicants have the choice to ignore a composition message. The protocol models the check of local objectives and available resources implicitly and defines state transitions on the basis of a positive attitude towards participating in the composite. This means, when a participant is an observer or in the position to (re-)apply, it may also transition to the initial listening state and not participate at all.

3.4.1.3 Scenario 3: Lost release

The scenario in Figure 3.10 has the same setup as the previous one, except that provider p_2 offers two services s_1 and s_2 and along with p_1 responds to the initiator's request. The initiator allocates p_1 as its successor. The particular selection algorithm is out of scope of this thesis. For simplicity, the provider with the lowest provider id is chosen.

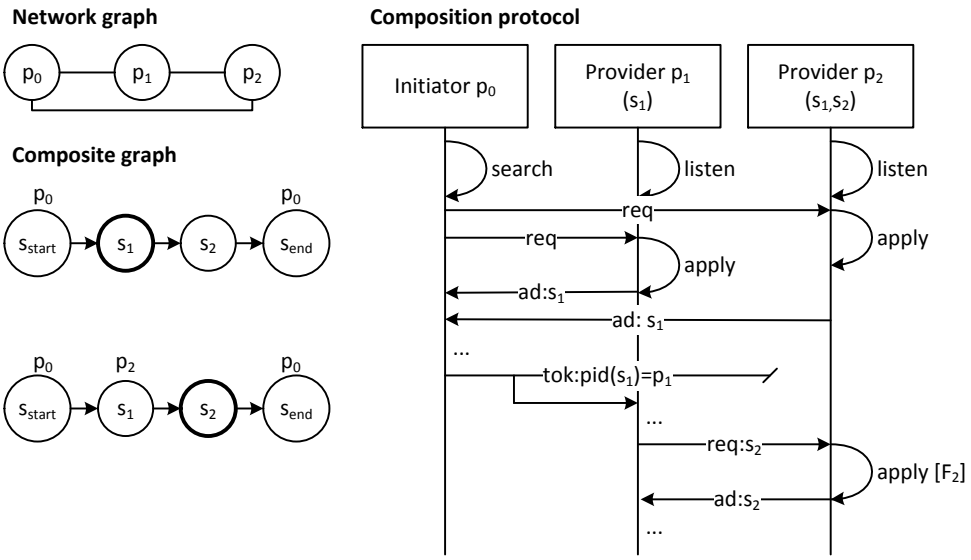


Fig. 3.10: ***Scenario 3: Lost release*** *Provider p_2 applies for covering service s_2, however, does not get the releasing token from the initiator. It stays blocked until it receives a request from p_1 indicating that the composition has made progress and announcements for s_1 are obsolete. This unblocks p_2 and enables it to apply for s_2.*

The initiator sends a token to p_1 to transfer the composition control and the same token to p_2 to release its resources. However, p_2 does not receive the token as it has moved out of the initiator's transmission range and stays blocked. In such a situation a timeout is the last resort to unblock redundant resources. The protocol allows for a complementary solution by analysing by-passing traffic and unblocking resources if service announcements have become obsolete. In case of p_2, it is still in range of p_1 and receives a request for s_2. The request indicates that an allocation decision for s_1 has been made and that all announcements for s_1 are obsolete. Generally, a service announcement is obsolete if the participant receives a request for a service that follows at some stage in the path of the advertised service. Provider p_2 unblocks its resources and applies for executing the next required service s_2:

$$F_2\text{:}\quad \text{cid(self)==cid(req)} \wedge \text{adObsolete} \wedge \text{offerNxt}$$

Other participants whose announcements are obsolete may transfer to observing if they can still contribute to the composite:

$$G_2\text{:}\quad \text{cid(self)==cid(req)} \wedge \text{adObsolete} \wedge \neg\text{offerNxt} \wedge \text{offerRemaining}$$

Otherwise they transfer to listening:

$$H_2: \quad \text{cid(self)==cid(req)} \wedge \text{adObsolete} \wedge \neg\text{offerNxt} \wedge \neg\text{offerRemaining}$$

The previous three scenarios illustrate the main aspects of the composition protocol for service sequences. They build the basis for extending the protocol to support the composition of parallel service flows which are discussed next.

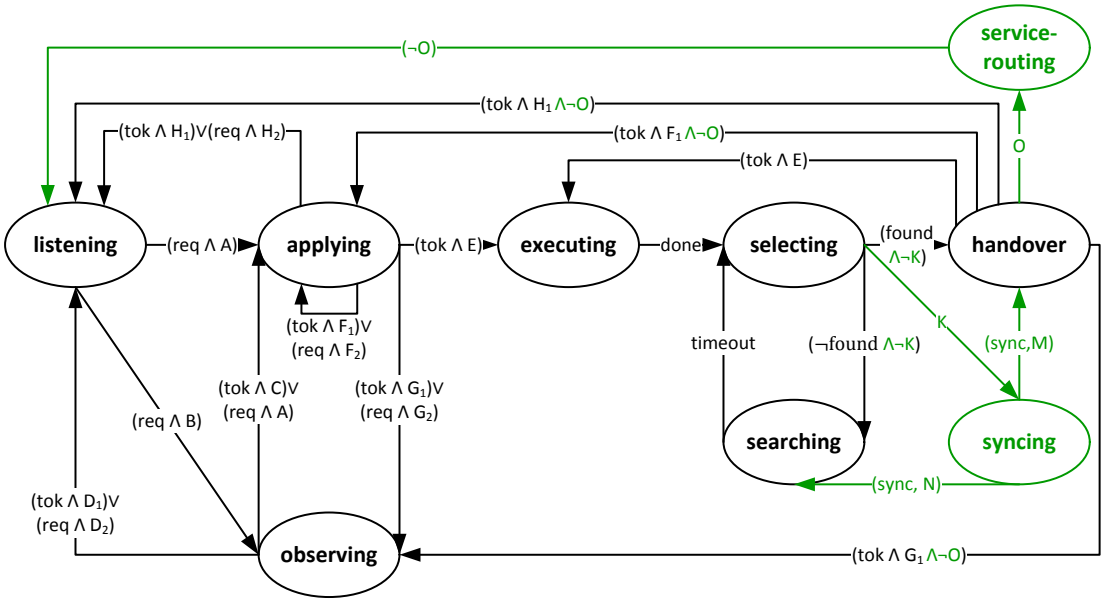
3.4.2 Service flows

Figure 3.11 illustrates the extensions of the protocol to support the composition of parallel service flows. Synchronisation messages are an important part of these extensions and another type of composition messages. Synchronisation partners exchange synchronisation messages to agree on a provider for their common successor, the so-called merge service. With service flows, a composite participant may receive token and synchronisation messages from multiple sources and relies on additional information to distinguish them. In particular, token and synchronisation messages include the last service which the source of a message executed along with the source's provider id. For composite reduction, these two message types further include a placeholder for edges in the composite graph that have become obsolete during the composite execution:

- **Synchronisation message (sync)** represents a suggestion for a merge service provider from a synchronisation partner. The synchronisation partner includes its own provider id and last executed service along with the provider id of its suggested merge provider and the synchronisation partner service to which the message is addressed. In addition, a placeholder contains those edges that got disabled in the composite graph.

- **Token (tok)** represents an allocation decision. In addition to the service to execute, its corresponding provider, and service input data, it contains the provider id and the last executed service of the message source and a placeholder for disabled edges in the composite graph.

The following syntax extends the notation introduced for service sequences to specify the protocol for service flows and its corresponding algorithms:



(a)

K	partner(self) $\neq \emptyset$
M	cid(self)==cid(sync) $\wedge$ allSyncRec $\wedge$ found
N	cid(self)==cid(sync) $\wedge$ allSyncRec $\wedge$ $\neg$found
O	SRR(self) $\neq \emptyset$

(b)

Fig. 3.11: ***Protocol for service flows*** *The green states and transitions in the state diagram (a) illustrate the protocol extensions to support the opportunistic composition of parallel service flows. The conditions (b) extend those specified in Figure 3.7.*

partner(self)	Participant's synchronisation partner services
allSyncRec	Participant received a synchronisation message from all its partners
SRR(self)	Participant's service routing responsibilities
SRR(s)	Service routing responsibilities of a service s
pred(s)	Predecessor services of service s
succ(s)	Successor services of service s
src(msg)	Service of the message's source
dst(msg)	Service of the message's destination
lastSR(msg)	Service of the message's last service router
eval(e)	Evaluate the conditional edge e
disable(e)	Reduce the composite graph by disabling edge e

The next two scenarios describe how the protocol supports the late allocation of a common merge service provider without updating the parallel paths along the way (design decision 4). Thereafter, three scenarios discuss how the protocol reduces a composite at runtime (design decision 5) to avoid the resource allocation for services that have become obsolete.

3.4.2.1 Scenario 4: Synchronisation

In the scenario of Figure 3.12 all service providers are in transmission range of provider p_0 and get activated by its request for the two parallel services s_1 and s_2. While p_1 and p_2 apply for covering them, p_3 and p_4 start observing the composite. When p_0 issues its allocation decision, it sends one token to both of its successors p_1 and p_2 who start executing. As p_3 and p_4 receive the token, too, they apply proactively for service s_3 with the successor that is in their transmission range. That is, p_4 sends its service announcement to p_1 while p_3 sends its announcement to p_2. These protocol steps have been covered in previous scenarios. This scenarios shows that p_1 and p_2 need to synchronise their allocation as they would otherwise hand over the composition control to different successors, p_2 to p_3 and p_1 to p_4. Provider p_1 and p_2, therefore, determine their set of synchronisation partner services (cp. Definition 4 in Section 3.2.1) and transition to synchronising as this set is not empty:

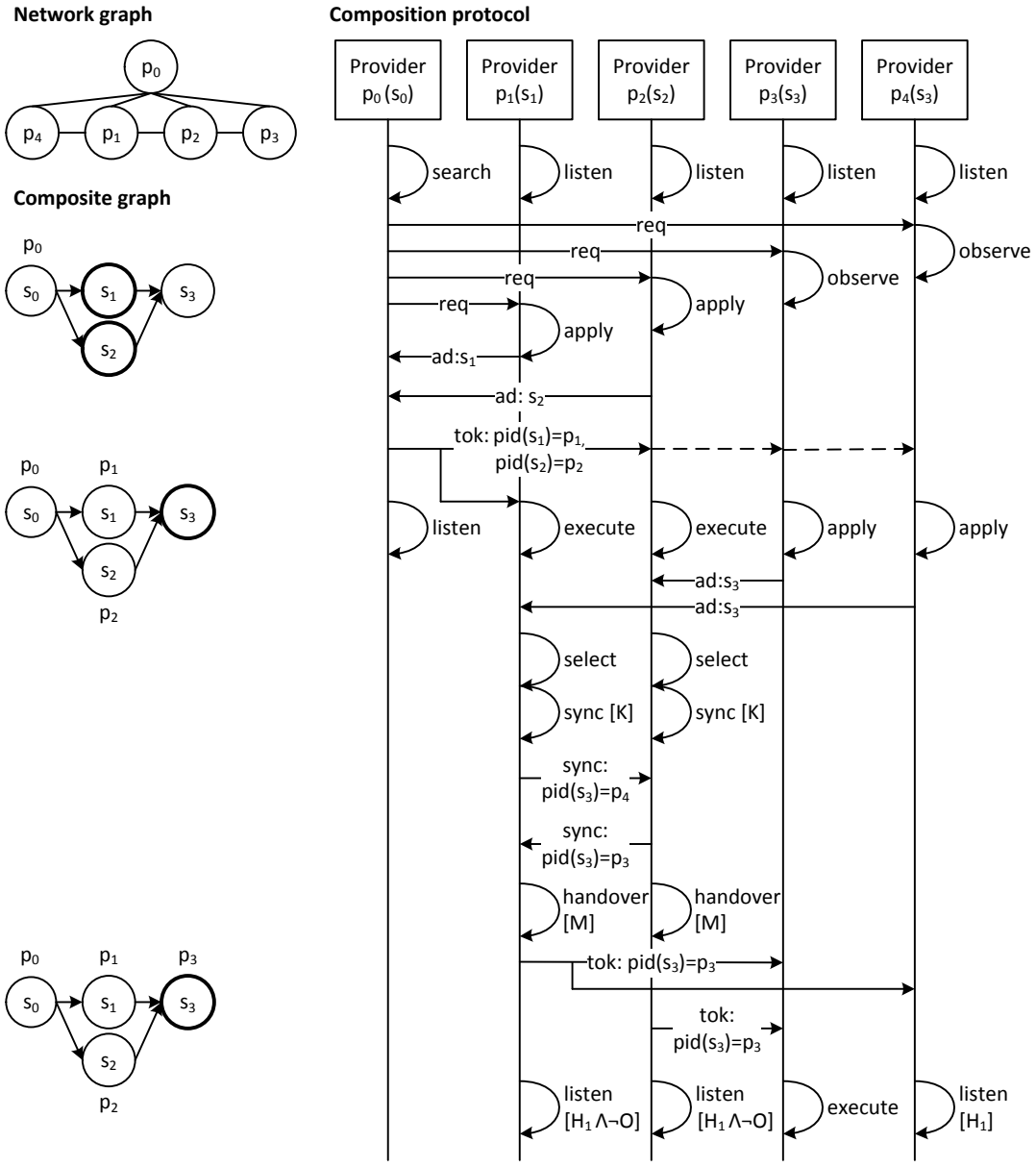


Fig. 3.12: ***Scenario 4: Synchronisation*** *Provider p_1 and p_2 have to synchronise their allocation for a common successor as they would otherwise transfer the composition control to different providers, i.e., p_1 to p_4 and p_2 to p_3. Instead, p_1 and p_2 exchange synchronisation messages with their candidate and run the same selection algorithm on the same set of candidates to determine p_3 as the common provider for s_3.*

K: partner(self) $\neq \emptyset$

Entering the synchronising state, p_1 and p_2 exchange sync messages that contain their candidate for the merge service s_3. Once each provider received the synchronisation message from its partner, it runs the same selection algorithm on the same set of candidates and individually determines the same merge provider. As mentioned before, this thesis does not focus on advanced selection criteria and rather lets synchronisation partners agree on the candidate that has the most proponents. As p_3 and p_4 each have one proponent, the one with the lowest provider id is chosen, which is p_3. Then, p_1 and p_2 hand over composition control as they agreed and found a common successor:

M: cid(self)==cid(sync)$\wedge$ allSyncRec $\wedge$ found

Provider p_1 sends its token also to p_4 which has become redundant and can release its resources. Thereafter, p_1 and p_2 return to the listening state as they do not provide any services that contribute to the composite (H_1) and do not have any service routing responsibilities:

$\neg O$: SRR(self)==$\emptyset$

Service routing responsibilities $SRR(s)$ represent the set of services for which the provider of service s must be prepared to forward synchronisation messages. Algorithm 1 takes service s as input and traverses the composition graph from s onwards. As soon as it finds a merge service for which it has not encountered a split service previously, it checks whether s is among the predecessors of that open merge service and thus a synchronisation partner. If not, it adds those predecessors of the open merge service to $SRR(s)$ that do not reside in the same path as s. For example in Figure 3.12, p_1 runs the algorithm with $s = s_1$ and discovers that s_3 is an open merge. However, since s_1 is among the predecessors of s_3, no service routing responsibilities are added. The algorithm terminates the graph traversal early (Line 20) if service s is not part of the service route, i.e., the shortest path between the open merge service and its corresponding split service. This is to limit the number of service routers to one path in case multiple paths between two partner services exist. Consider, for example, the complex composite in Figure 3.13. For $s = s_4$ the algorithm detects s_{12} as the first open merge service and

Algorithm 1 Determine set of service routing responsibilities $SRR(s)$

Input: Service s

Output: Set $SRR(s)$

```
1: Queue queue, visited
2: split ← merge ← openmerge ← 0
3: if |succ(s)| > 1 then // count first split
4:     split ← 1
5: end if
6: visited.add(s), visited.add(succ(s)), queue.add(succ(s))
7: while queue not empty do // traverse composite graph breadth-first
8:     n ← queue.dequeue()
9:     if |pred(n)| > 1 then // count merge
10:        merge++
11:    end if
12:    if merge > split then // no corresponding split
13:        openmerge++
14:        merge ← split ← 0
15:        if s not in pred(n) then
16:            for all i in pred(n) do // add predecessors of open merge that do not reside in s path
17:                if i not in visited then SRR(s).add(i) endif
18:            end for
19:        end if
20:        if openmerge == 1 and s not in shortestPath(getSplitFor(n), n) then // stop if s not in
           shortest path between open merge and corresponding split
21:            break
22:        end if
23:    end if
24:    if |succ(n)| > 1 then // count split
25:        split++
26:    end if
27:    for all o in succ(n) do
28:        if o not in visited then visited.add(o), queue.add(o) endif
29:    end for
30: end while
31: return SRR(s)
```

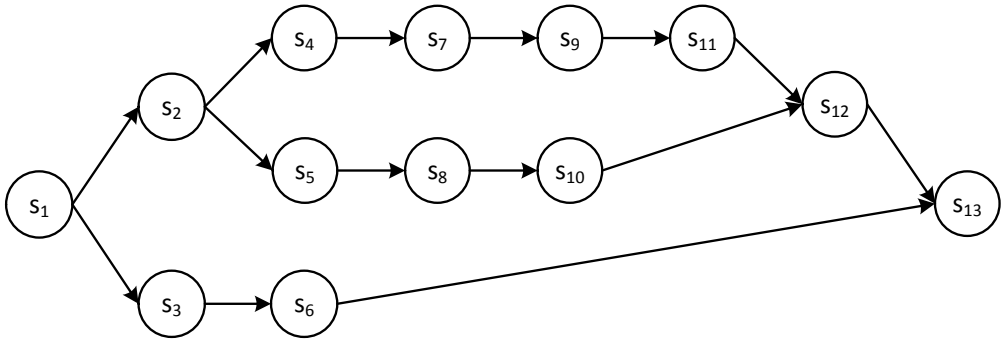


Fig. 3.13: ***Example complex service flow*** *Synchronisation messages between the synchronisation partners s_6 and s_{12} are routed along the shortest path which includes the section s_5, s_8, and s_{10} and releases the parallel path starting with s_4 from their routing responsibilities for s_6.*

adds s_{10} to $SRR(s_4)$. Then it checks whether s_4 is part of the shortest path between the open merge service s_{12} and its corresponding split service s_2. This is not the case because the path via s_5, s_8, and s_{10} is shorter. This means, s_4 is not responsible for any service after s_{12} and thus can stop traversing the graph. In contrast, $SRR(s_5)$ contains s_{11} and s_6 as s_5 is part of the shortest path between s_2 and s_{12} and responsible for the second open merge service s_{13}.

3.4.2.2 Scenario 5: Service routing

Initially, a service provider may not be aware of the provider id that corresponds to a synchronisation partner service and relies on service routers to deliver its synchronisation message. This is the case for provider p_3 and p_4 in Figure 3.14. The scenario differs from the previous one in that the required composite extends its parallel execution paths by two services such that p_1 and p_2 assign their successors p_4 and p_3 independently. However, when p_4 and p_3 have to synchronise, they do not know each other's provider id as their local view on the composite illustrates. They send their synchronisation message to the provider, they know in the partner path, i.e., p_3 to p_1 and p_4 to p_2. The providers p_1 and p_2 have determined their service routing responsibilities earlier:

O: $\text{SRR(self)} \neq \emptyset$

and are prepared to forward the synchronisation messages toward the final destinations using service layer routing. After p_3 and p_4 received each other's initial synchronisation

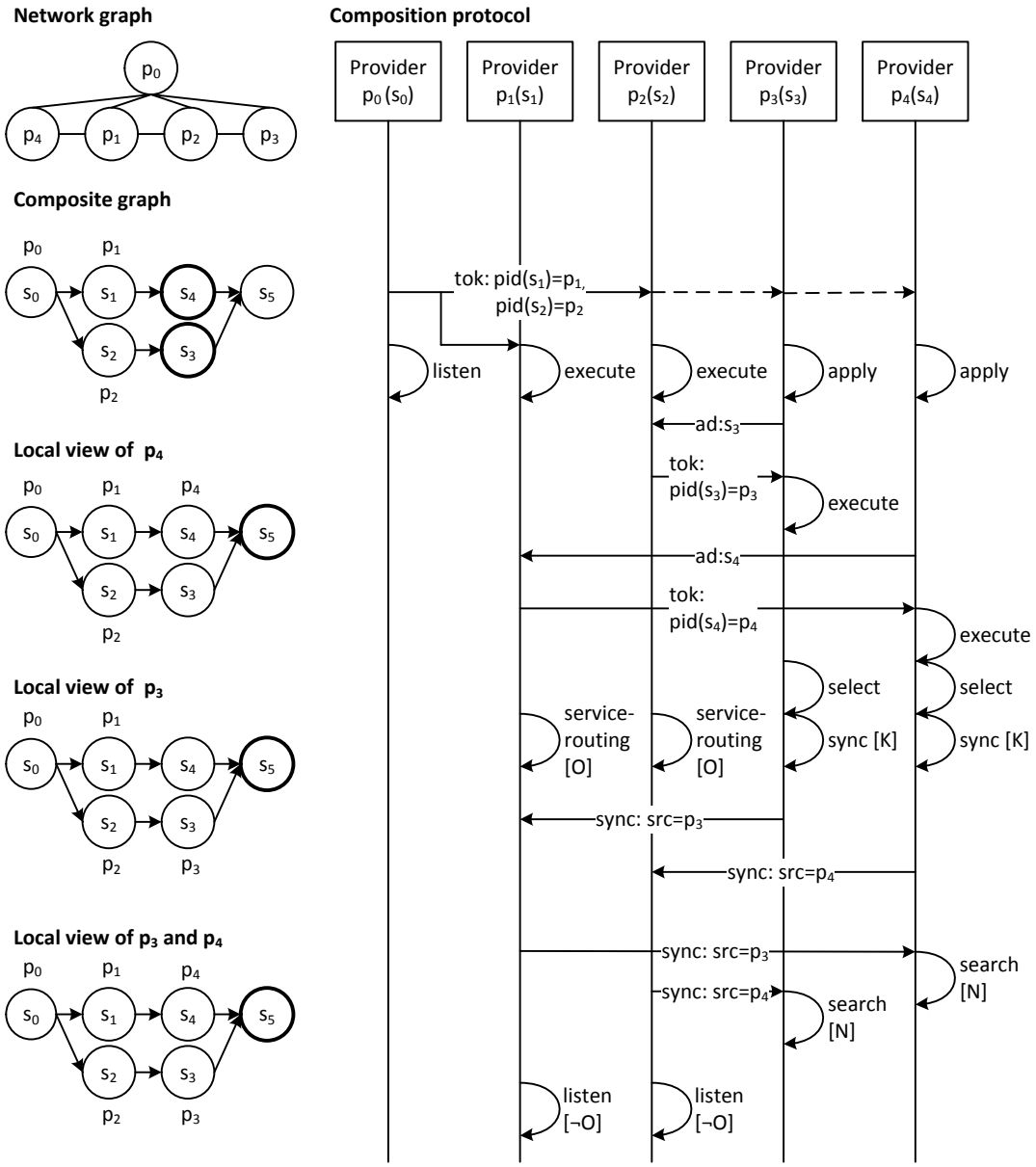


Fig. 3.14: ***Scenario 5 Service routing*** *The providers p_3 and p_4 have to synchronise their allocation decision for the merge service s_5. However, initially they are not aware of each others identity as their local view of the composite graph illustrates. They rely on service routing from p_1 and p_2 to dispatch their synchronisation messages. Once they have received their partner's initial synchronisation message they update their local view and from there on interact directly without any service routers.*

message, they interact directly and do not rely on the service routers p_1 and p_2 because the synchronisation message includes the provider id of the message source. Before the providers p_3 and p_4 synchronise again, they have to search because their previous synchronisation messages did not contain a candidate for the merge provider:

N: cid(self)==cid(sync)$\wedge$ allSyncRec $\wedge$ $\neg$found

Service layer routing determines the next recipient of a synchronisation message. This is used by synchronisation partners and participants with service routing responsibilities, so-called service routers. Algorithm 2 implements service layer routing and returns a list of recipients based on the service s that was executed last by the synchronisation partner or service router. The algorithm distinguishes three cases based on the position of service s in the composite:

1. The **source of the synchronisation message** (Line 1-13) traverses the composite graph from the destination, i.e., the partner service, backwards until it finds a known provider id. It stops at latest at the first service of the destination path because as seen in the scenario, the split provider includes all successors in one token. Subsequent service requests carry this information in their composite graph.

2. The **first or any router thereafter in the destination path** (Line 15-21) forwards the synchronisation message towards its destination. If a service router receives a synchronisation message before it allocates the next service, it stores the message and dispatches it once the provider for the next service is found.

3. The **first or any previous router in the destination path** (Line 22-28) relays the synchronisation message backwards in the destination path to release those service routers that have been skipped.

After a service router has forwarded a synchronisation message, it removes the message's source from its SRR such that it eventually becomes empty and the service router can return to listening.

The scenarios described up to this point assume that all edges in the composite graph are valid. The next three scenarios show how the composite graph can be reduced at

Algorithm 2 Service layer routing of synchronisation messages

Input: Service s locally last executed, Synchronisation message $sync$

Output: Set $recipients$

```
1: if s == src(sync) then // source of the sync message
2:    queue.add(dst(sync))
3:    while queue not empty do // traverse composite graph backwards
4:       n ← queue.dequeue
5:       if pid(n) known then // stop if provider id in partner path is known
6:          recipients.add(pid(n)), break
7:       else // check previous service routers
8:          for all i in pred(n) do
9:             if src(sync) in SRR(i) then queue.add(i), break endif
10:         end for
11:      end if
12:   end while
13: else
14:   if s ≠ dst(sync) then
15:      if s in succ(lastSR(sync)) or lastSR(sync) == src(sync) then // 1st or any subsequent
         router in dest path
16:         for all o in succ(s) do // route toward destination
17:            if src(sync) in SRR(o) or o == dst(sync) then
18:               recipients.add(pid(o)), break
19:            end if
20:         end for
21:      end if
22:      if s in pred(lastSR(sync)) or lastSR(sync) == src(sync) then // 1st or any previous router
         in dest path
23:         for all i in pred(s) do // route backward to release skipped routers
24:            if src(sync) in SRR(i) then
25:               recipients.add(pid(i)), break
26:            end if
27:         end for
28:      end if
29:   end if
30: end if
31: return recipients
```

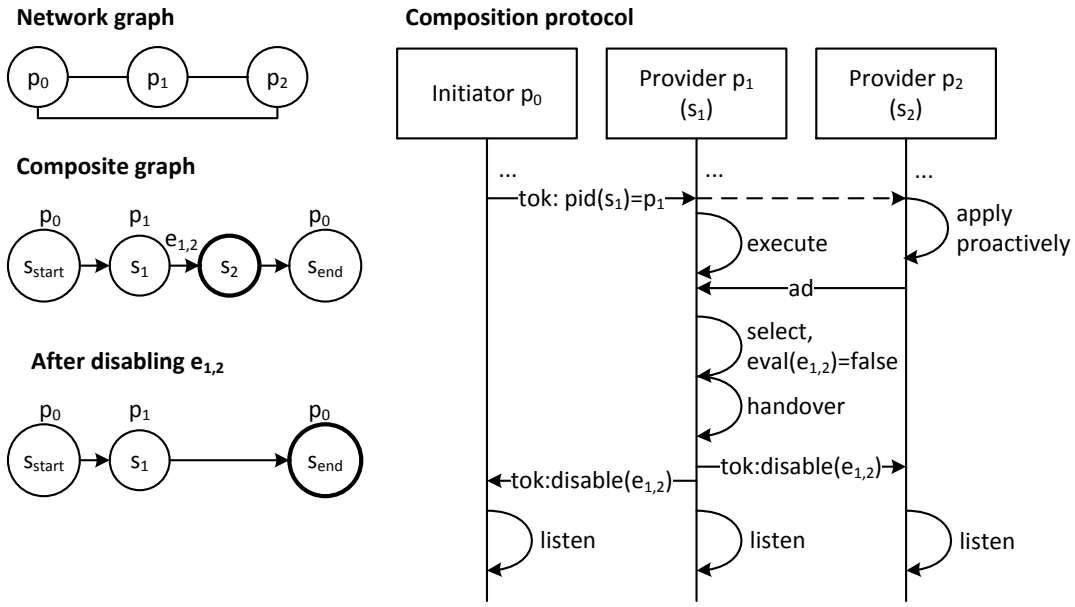


Fig. 3.15: ***Scenario 6 Reducing a sequence*** *Provider p_1 discovers that the edge $e_{1,2}$ evaluates to false and releases proactive service announcements for the obsolete service s_2 before it returns the composition result to the composite initiator.*

runtime (design decision 5) if a controller discovers that an edge has become obsolete, i.e., its edge condition evaluates to false.

3.4.2.3 Scenario 6: Reducing a sequence

In the scenario illustrated in Figure 3.15 provider p_1 and p_2 receive the allocation decision of the initiator. While p_1 starts executing s_1, p_2 proactively applies for covering s_2. However, in the selecting state p_1 discovers that the control edge $e_{1,2}$ is invalid. Provider p_1 updates its composite graph by disabling $e_{1,2}$ and sends a a token to the initiator to deliver the composite result as well as to p_2 to release it. The token contains the disabled edge such that each token recipient can update their local view of the composite graph.

3.4.2.4 Scenario 7: Reducing after a split

The scenario in Figure 3.16 differs from the previous one in that p_1 executes a split service and the edge $e_{1,3}$ evaluates to false while the edge $e_{1,2}$ remains true. Provider p_1 allocates the valid successor s_2 to p_2 and releases the obsolete provider p_3. Both token

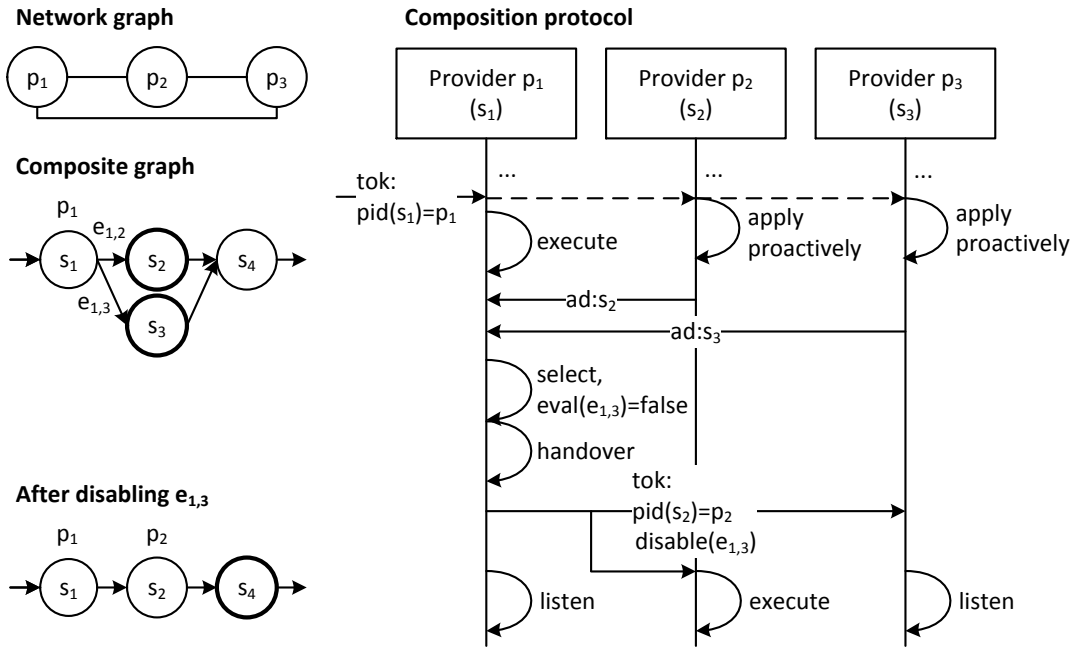


Fig. 3.16: ***Scenario 7 Reducing after a split*** *When provider p_1 discovers that the edge $e_{1,3}$ evaluates to false, it releases the proactive advertiser p_3 as service s_3 is obsolete and hands over the composition control to the p_2 to execute the valid successor service s_2.*

recipients receive the allocation decision and the disabled edge to update their composite graph and determine their next system state.

3.4.2.5 Scenario 8: Reducing a service route

The scenario in Figure 3.17 focuses on s_4 in the complex service flow. When its assigned provider p_4 discovers that the edge $e_{4,6}$ evaluates to false, it takes over all responsibility of the last service in this parallel path, which is s_6. This is necessary because the composite view will not be updated in all participants. Provider p_4 synchronises on behalf of s_6 with the synchronisation partner s_8 to find a common merge provider for s_9 and adds the service routing responsibilities of s_6 to its own. With the token from p_4, the merge service s_9 updates its composite graph and integrates the disabled edge as well as the intermediate result from s_4 in its service logic. Provider p_4 transitions to the service-routing state because s_4 and s_6 have routing responsibility for s_3. Generally, recipients of a message with a disabled edge, include this edge in their subsequent tokens to spread

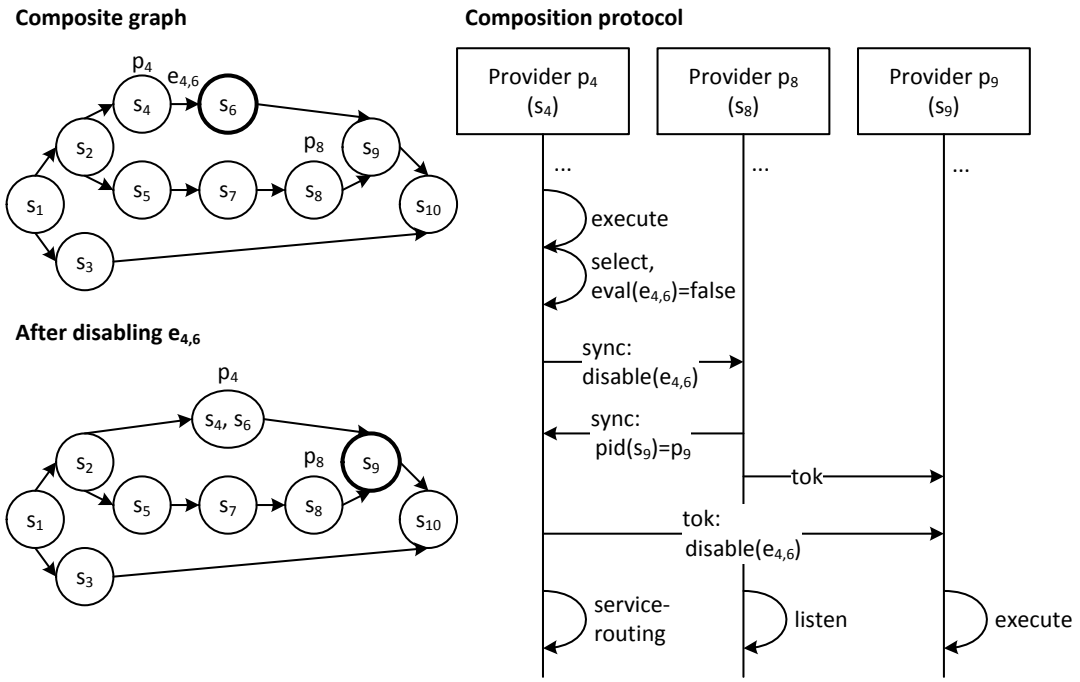


Fig. 3.17: ***Scenario 8 Reducing a service route*** *Provider p_4 takes on the synchronisation and service routing responsibilities of the last service in its path, namely s_6, when it discovers that edge $e_{4,6}$ evaluates to false.*

the update among proactive applicants and silent observers. Once they issue a composite request, this is no longer necessary because the request contains the up-to-date version of the composite graph.

The previous eight scenarios for service sequences and parallel service flows build on the idea that a service provider can observe network traffic related to a service composition that is issued in its transmission range. The next part explains how the protocol realises observable composition messages (design decision 7) and multicast messaging (design decision 8) in a cross-layer communication approach.

3.4.3 Cross-layer communication

A composition message, regardless of its final destination contains valuable information for a potential composite participant. For example, the participant that forwarded the message last is in direct communication range and also the first hop towards the message's original sender. The service provider uses this information to update its local

network graph and to extend its topology knowledge. Similarly, composite requests, synchronisation messages, and token messages reveal current and previous allocation decisions which the provider uses to update its local view of the composite graph. Together the network and composite graph enable a service provider to better understand its neighbourhood and to derive composition-related actions such as when to apply proactively, observe silently, or release blocked resources. In the protocol, the service provider has access to both graphs when it engages in a composition.

3.4.3.1 Observable composition messages

For the service provider to receive any composition message that is issued in its transmission range, the underlying network layer must pass it on to the composition layer. Generally, a message can be marked as a broadcast or a unicast. Broadcast messages are put through all the way up to the composition layer but are not acknowledged and do not get recovered if they are lost. Unicast messages, on the other hand, get acknowledged but enter the composition layer only if the participant is a designated addressee. The protocol combines broadcast and unicast and sends a message as an directed broadcast which is observable by anyone in transmission range and gets acknowledged by its designated recipients. Sent as an ordinary broadcast, an intermediate hop acknowledges the receipt of a directed broadcast such that the previous hop can detect and recover a possible loss.

3.4.3.2 Multicast messaging

Token and synchronisation messages have multiple recipients as they, amongst other things, signal redundant providers to release blocked resources. For these messages all primary recipients are included in one directed broadcast message. The directed broadcast gets multicasted because it contains a list of final destinations. The sender of the message assigns for each final destination, the next hop based on its local topology knowledge. If the next hop is unknown or has expired, the sender initiates a route discovery process. Once all next hops are identified, the sender forwards the message.

3.5 Chapter summary

The design objectives for service composition in mobile ad hoc networks highlight the tension between resource-intensive composition tasks and the resource-constrained operating environment. On the one hand, a composite must support parallel service flows, self-organise the composition process, proactively gather service information, and obtain the provider commitment. On the other hand, it needs to reduce communication and enable short and localised interaction. Formally, service composition in dynamic networks corresponds to the problem of finding a mapping between the composite graph of required services and the time-varying graph of provided services such that the communication link between two services is active whenever these two services interact. The proposed protocol for opportunistic service composition addresses this problem with demand-based resource-blocking service announcements, decentralised interleaved composition, composite reduction at runtime, and observable end-to-end multicasting. These design decisions come with a number of implications: Service announcements already confirm the provider commitment without further communication but require the release of redundant provider resources once a service is allocated. Interleaving service allocation with service invocation reduces the composite at runtime to those parts that will actually be executed but with regard to parallel service flows requires the agreement of multiple initially unknown service providers. Allocating parallel execution paths in isolation avoids the frequent communication between parallel paths but introduces the need for service layer routers which prolongs resource blocking.

Chapter 4

Design verification

This chapter verifies the correctness of the designed protocol and checks whether the following properties hold:

1. The protocol does not deadlock and at least one composite participant moves forward in the composition.

2. Once the composition is complete, each composite participant has returned to listening which is the end state of the protocol.

3. At the end of a successful composition all required services executed once.

This thesis uses model checking to verify the properties because this method automatically decides if a communication protocol modelled as a finite-state program satisfies its specification (Clarke et al., 1994). A model checker requires the formal description of the protocol and the properties. Then, it resolves the non-determinism that stems from random choices and concurrent actions in the protocol and creates a finite state automaton. Afterwards, the model checker negates the property that is being checked and searches the automaton exhaustively for a path where the negated property is true. If such a path exists, the model checker found a counter example that falsifies the correctness of the specification. Otherwise, the property is correct. This chapter contains three parts: The first part introduces the specification language PROMELA and its interpreter SPIN that are used for the design verification (cp. Section 4.1). The second part focuses on the verification of service sequences and the basic protocol features (cp.

Section 4.2). The third part verifies the protocol extensions for parallel service flows (cp. Section 4.3). The second and the third part follow the same structure and describe protocol abstractions and modelling details first before they state the verification result.

4.1 PROMELA and SPIN

This thesis uses PROMELA and SPIN (Holzmann, 2003) for checking the correctness of the protocol properties. Both tools are designed for the analysis of communication protocols and are used in academia and industry. PROMELA is a process-meta language to model distributed and concurrent software systems. It focuses on process synchronisation and message passing rather than process-internal computation. Processes represent protocol entities and are identifiable via their process id. They implement protocol-specific behaviour using statements and can communicate by sending and receiving messages via channels. The PROMELA interpreter executes a protocol by randomly interleaving the statements of different processes. These statements may be guarded and a process blocks until one of its guarded options becomes true. If more than one guard is enabled, the PROMELA interpreter randomly selects one possible option. Among the different means of specifying properties, this thesis uses PROMELA's built-in support for assertions and meta labels. The formal protocol specification adopts the following PROMELA syntax:

`proctype`	Protocol entity modelled as a process
`_pid`	Id of a process
`chan c`	Message channel c
`c ! m`	Sending a message m via a channel c
`c ? m`	Receiving a message m via a channel c
`d_step` or `atomic`	Indivisible statement
`:: guard -> statement`	Guarded option
`end`	Meta label for valid end state
`assert()`	Assertion

SPIN is a PROMELA interpreter. It generates a verifier from the PROMELA code that checks the model with regard to the specified property. When a counter example exists, SPIN simulates and examines the failed run.

4.2 Service sequences

Service sequences represent the basic behaviour of opportunistic composite participants. The following description first details how the sequential PROMELA model abstracts from the original protocol. Then, it explains how service sequences, communication, and participants are modelled. Thereafter, it shows how the correctness properties mentioned at the beginning of the chapter are specified and verified.

4.2.1 Protocol abstractions

The sequential PROMELA model abstracts from the original protocol (cp. Figure 3.7 on page 58) and focuses on the communication part because the properties to check refer to the interaction of service providers rather than their internal computations. This means, the executing and selecting state are not explicitly modelled as they do not involve communication. In the model all required services are available and can be allocated to a service provider. Otherwise, the current composite controller would reduce the composite and return the final result early not testing the full service sequence (cp. Scenario 6 on page 74). For simplicity but without loss of generality, the model assumes a service provider offers all required services but requires that consecutive services are allocated to different providers. For the transition conditions this implies that *offerNxt* is always true. Further, the model focuses on the composition layer. It masks routing layer details and fully connects the service providers such that *inRange* is always true. Under these conditions the observing state is redundant. Finally, the model examines a single composite such that cid(self)==cid(msg) is always true. Figure 4.1 depicts the protocol after it applied the abstractions.

4.2.2 Modelling service sequences

The variable `local_sid` and the constant `MAX_SERVICES` model a sequential composite. Each participant has a `local_sid` that represents what the participant beliefs is the current position in the sequence. A service provider that receives the composition control increments this value and includes it in all its messages. Recipients of these messages update their `local_sid` and derive their further actions. For example, a new controller

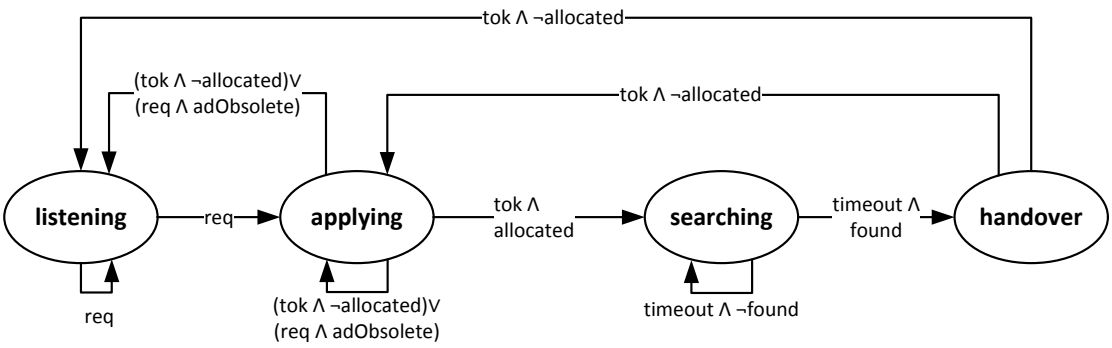


Fig. 4.1: ***Abstractions for the sequential PROMELA model*** *focus on the communication among composite participants. They eliminate the executing, selecting, and observing state and simplify the remaining transition conditions without loss of generality. The listening, applying, and handover state have two outgoing transitions with the same condition but different destination states to explicitly model participation autonomy.*

returns to listening if it detects that all services have been covered and the end of the service sequence has been reached. Otherwise, it starts searching for a successor:

```
local_sid = local_sid+1;
if
:: local_sid==MAX_SERVICES -> state=listening;  // all services covered
:: else -> state=searching;
fi;
```

4.2.3 Modelling communication

The global array `channels` stores the communication channel to each composite participant and allows for the retrieval of a particular channel based on the recipient's process id. A message sent to all channels in the array models broadcasting. For example, broadcasting a request is modelled as:

```
for(i in channels){
  if
  :: i==(_pid-1) -> skip;          // skip my channel
  :: else -> send(req, channels[i], local_sid, _pid, 0);
  fi;
}
```

The `send` macro requires a message type, a destination channel, a service id, a sender id, and an id for the next controller (zero means unspecified). Apart from requests `req` other message types are service announcements `ad`, allocation decisions that are modelled separately as tokens `tok` and release `rel`, and acknowledgements `ack`. The send macro models unreliable communication as it randomly chooses whether to actually dispatch or drop a message:

```
inline send(msg_type, dst_channel, service_id, sender_id, nxt_id){
   if
   :: dst_channel ! msg_type, service_id, sender_id, nxt_id;     // dispatch
   :: true;                                                      // drop
   fi;
}
```

Both options in the if statement are true and the PROMELA interpreter randomly decides which to execute. The model recovers from message loss by resending a message after a timeout which reflects the basic recovery strategy of the medium access control (MAC) layer in wireless networks. The model uses PROMELA's built-in `timeout` which becomes executable if no other statement can execute. This is sufficient because the protocol does not specify detailed real-time behaviour and rather resends a message after some time of idleness.

4.2.4 Modelling participants

The `participant` process models the behaviour of controllers, listeners, and applicants who repeatedly check their state and act accordingly.

```
proctype participant(mtype state){
   do
   :: state == controlling -> ... // ancillary for initialisation
   :: state == searching -> ...
   :: state == handover -> ...
   :: state == listening -> ...
   :: state == applying -> ...
   od;
}
```

The next paragraphs explain how the model reflects the main aspects of each such state.

Searching Initially, a controller switches to searching, blocks until it receives a service announcement and broadcasts a request when it times out. Once the controller receives the first service announcement for the service that corresponds to its `local_sid`, it selects the sender as the new controller and hands off control by sending a token:

```
:: msg_type==ad && msg_sid==local_sid ->
   d_step{
      local_nxtctrl = msg_src;          // select successor
      state = handover;                 // wait for ack in handover
   }
   send(tok, channels[(local_nxtctrl-1)], local_sid, _pid, local_nxtctrl);
```

Handover Then, in the handover state the controller listens for an acknowledgement from the token recipient or requests for subsequent services that confirm that the successor has picked up the composition control. When the controller receives a service announcement, it releases the sender as a redundant advertiser and includes the new controller:

```
do
:: channels[(_pid-1)] ? msg_type, msg_sid, msg_src, msg_nxt ->
   if
   :: (msg_sid==local_sid && msg_type == ack) ||
      (msg_sid > local_sid && msg_type == req) ->
      local_sid = local_sid+1;  // successor has picked up control
      ...                       // choose to listen or apply
   :: msg_sid == local_sid && msg_type == ad && msg_src!=local_nxtctrl ->
      send(rel, channels[(msg_src-1)], local_sid, _pid, local_nxtctrl);
   :: else -> skip;
   fi;
:: timeout -> ... // resend token
od;
```

Listening In the listening state when a service provider receives a request, it randomly chooses whether to apply. This participation autonomy is modelled using PROMELA's ability to randomly choose between multiple enabled guards and other model segments will refer to it as '// choose to listen or apply':

```
...                                          // receive message from mgs_src
if
:: skip -> state=listening;                  // enabled guard 1
:: skip ->                                   // enabled guard 2
   if
   :: local_sid==MAX_SERVICES ->
      state = listening;                     // end of request, do not apply
   :: else ->
      d_step{
         state=applying;                     // apply with source of message
         local_nxtctrl=msg_src;              // that triggered this decision
      }
      send(ad, channels[(local_nxtctrl-1)], local_sid, _pid, 0);
   fi;
fi;
```

A listener compensates potential message loss and responds to announcements from redundant applicants or confirms tokens after it has dismissed its controller role. Further, the listener specifies two references, the labels `end` and `L0`, that the verifier will use to check the correctness properties. The model represents a listener as follows:

```
end:
L0:
   do
   :: channels[(_pid-1)] ? msg_type, msg_sid, msg_src, _ ->
      if
      :: msg_type==req && msg_sid>=local_sid ->
         local_sid = msg_sid;    // update which service is required next
         ...                     // choose to listen or apply
      :: msg_type==ad && msg_sid<local_sid -> send(rel,...);
      :: msg_type==tok && msg_sid<local_sid -> send(ack,...);
      :: else -> skip;
      fi;
   od;
```

Applying In the applying state, when an applicant receives a token, it picks up the composition control and marks that for the verifier in the global `controllerPerService`

array which counts the number of controllers per required service:

```
:: msg_sid==local_sid && msg_type==tok ->
   send(ack,..);                                    // I am the new controller
   d_step{
    state=controlling;
    controllerPerService[local_sid] =               // set for verification
        controllerPerService[local_sid] + 1;
   }
   break;
```

When an applicant receives a release, it increments its `local_sid` to indicate that the next service in the sequence is required. Then it chooses whether to listen or apply. If the applicant applies again, it does so proactively without having received an explicit request for the next service. It uses the new controller id that was enclosed in the release message to dispatch its proactive service announcement:

```
:: msg_sid==local_sid && msg_type==rel ->
   local_sid=msg_sid+1; // move to next required service
   ...                  // choose to listen or apply (proactively)
```

With a request for a subsequent service the applicant knows its service announcement is obsolete and can choose whether to apply for the next service:

```
:: msg_sid>local_sid && msg_type==req ->
   local_sid = msg_sid; // ad obsolete, move to next required service
   ...                  // choose to listen or apply
```

4.2.5 Verification

For the verification SPIN activates three instances of the participant process (one as the controller and two as listeners) and a monitor process. The participants run the composition protocol for a sequence of four services:

```
MAX_SERVICES 4
init{
  atomic{
    run participant(controlling);       // handle first service
    run participant(listening);         // wait for request
    run participant(listening);         // wait for request
```

```
    run monitor();                          // watch participants
  }
}
```

The monitor asserts that when all participants have reached label `L0` in the listening state each required service has been covered once in the `controllerPerService` array:

```
do
:: (participant[1]@L0 && participant[2]@L0 && participant[3]@L0) ->
    assert(
    controllerPerService[0]==1 && controllerPerService[1]==1 &&
    controllerPerService[2]==1 && controllerPerService[3]==1);
    break;
od;
```

Using SPIN the correctness of the properties are verified as follows:

- Property 1 is correct if the verifier checks for deadlocks and completes without any errors.
- Property 2 is correct if the verifier completes without any participant process being in an invalid end state. The participant processes are non-terminating and require the insertion of the meta label `end` at the beginning of the listening state to inform SPIN that the listening state is actually a valid end state.
- Property 3 is correct if the verifier completes and the monitor process has reached the valid end state. This is only possible if the assertion is true and the loop terminates.

Running SPIN with this PROMELA model does not produce any errors and shows the correctness of all three properties for the sequential model. Figure A.2 in Appendix A shows the corresponding output of the model checker. In comparison, note Figure A.1 which illustrates the output if the verification of the model fails.

4.3 Service flows

The parallel PROMELA model extends and modifies the previous model to support the verification of parallel service flows. The following description centres on the changes that

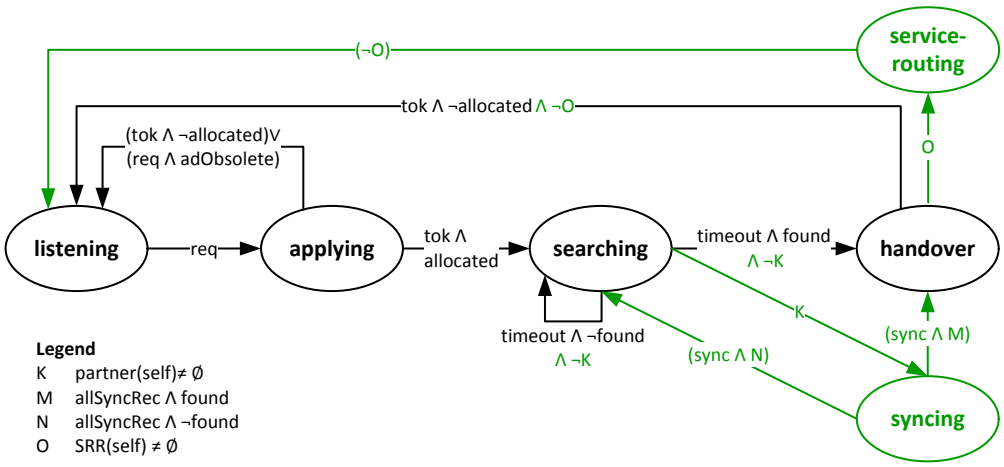


Fig. 4.2: ***Abstractions for the parallel PROMELA model*** *focus on service routing and synchronisation (depicted in green). Proactive service announcement and participation autonomy are omitted because these features have been tested in the sequential model.*

have been made and keeps the same structure as the previous section. It first explains the protocol abstractions and modelling details before it turns to the verification results.

4.3.1 Protocol abstractions

Figure 4.2 illustrates the abstractions from the original protocol for service flows (cp. Figure 3.11 on page 65). Focusing on synchronisation and service routing, the model omits proactive service announcements and participation autonomy as this has already been verified in the sequential model. This means, while listening service providers always apply when receiving a request, providers in other states return to listening when they get released and start a new application from there.

4.3.2 Modelling service flows

As in the sequential model, the variable `local_sid` indicates which service is required next. In addition, the Boolean variables `s_split`, `s_sync`, and `s_merge` determine the type of the required service, i.e., whether it is a split service, a service with synchronising partners, or a merge service. The controllers set these variables during the course of interaction. In the model the service flow starts with a split service, continues with two

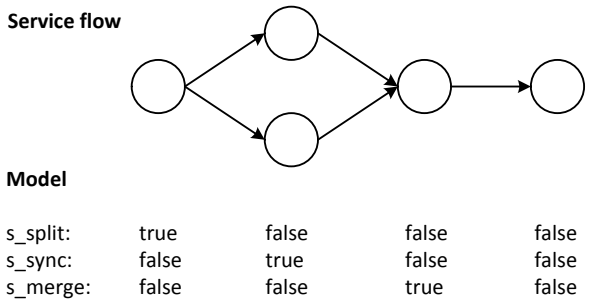


Fig. 4.3: ***Model of service flow*** *The variables s_split, s_sync, and s_merge model the service flow that is used for verification by changing their Boolean value as the composition progresses.*

synchronisation partners, then joins the parallel execution paths in a merge service, and finishes with a final service (cp. Figure 4.3). This is sufficient to verify the properties for the split, synchronisation, and merge behaviour of the protocol.

Further each participant has two placeholders `cp1` and `cp2` to store information about its counterparts. Depending on the type of service the participant controls, the counterparts represent successors (of the split service), predecessors (of a merge service), or the successor candidate and the partner (of a synchronisation partner).

4.3.3 Modelling communication

For service synchronisation, the model introduces two additional message types: synchronisation messages `sync` and their acknowledgments `syncAck`. The model changes the `send` macro to include a partner id, candidate id, synchronisation id in a message:

```
inline send(msg_type, dst_channel, service_id, sender_id,
    partner_id, candidate_id, sync_id)
```

The loss of messages in the parallel model centres on the newly introduced `sync` and `syncAck` because unreliable communication for the other message types has already been tested in the sequential model.

4.3.4 Modelling participants

In addition to the existing states, the participant process integrates the states synching and service routing to verify the procedure of finding a common merge service provider:

```
proctype participant(mtype state){
   do
   :: state == controlling -> ... // ancillary for initialisation
   :: state == searching -> ...
   :: state == handover -> ...
   :: state == listening -> ...
   :: state == applying -> ...
   :: state == synching -> ...
   :: state == service_routing -> ...
   od;
}
```

The details of this procedure will be explained based on the modelled service flow, namely in terms of a split service, a service with synchronisation partners, and a merge service.

4.3.4.1 Split service

Searching and handover A controller of the split service waits in the searching state for two service announcements. It selects and stores the first two applicants in `cp1` and `cp2` as its successors, advances the service flow to indicate that the subsequent service providers are synchronisation partners, and sends them its allocation decision in a token:

```
:: msg_type==ad && msg_sid==local_sid && s_split==true ->
   if
   :: cp1.id==0 -> cp1.id=msg_src;       // first successor
   :: cp1.id>0 && msg_src!=cp1.id ->
      d_step{
         state = handover;
         cp2.id = msg_src;               // second successor
         s_split = false;                // split service has been covered
         s_sync = true;                  // successors are sync partners
      }
      send(tok, channels[(cp1.id-1)], local_sid, _pid, 0, 0, -1);
      send(tok, channels[(cp2.id-1)], local_sid, _pid, 0, 0, -1);
      break;
   :: else -> skip;
   fi;
```

Thereafter the controller waits in the handover state for acknowledgements that its successors have picked up the composition control. In the original protocol, the split controller includes all successor ids in the token. In the model this is omitted to create the need for service routing while keeping the service flow simple. The split controller moves to the service routing state after the handover state and the meaning of its successors in `cp1` and `cp2` changes to service responsibilities. In contrast to the original protocol, the model masks how a participant determines its service responsibilities as this process-internal computation cannot be expressed with PROMELA. Instead, it focuses on the required interactions that result from having service responsibilities.

Service routing In the service routing state, when the participant receives a `sync` message from one of its service responsibilities, it first sends a `syncAck` back to the message source and then forwards the `sync` to its destination, if the destination has not yet acknowledged the receipt of that message:

```
:: msg_type==sync ->
   send(syncAck, channels[(msg_src-1)], ..., msg_syncid);
   if
   :: msg_src==cp1.id ->                    // cp1 sent sync
      if
      :: cp2.syncAckRecv == false ->        // cp2 has not received cp1's sync
         send(sync, channels[(cp2.id-1)], msg_sid, _pid, cp1.id,
             cp1.candidate_id, cp1.sync_id)
      :: else -> skip;
      fi;
   ...                                      // vice versa if cp2 sent sync
   fi;
```

Similarly to a `syncAck`, a request for a subsequent service signals the service router that the sender of the message has received all required synchronisation messages and that the sender interacts with its partner directly. If all its routing responsibilities have been covered this way, the service router returns to listening:

```
:: msg_type==syncAck || msg_type==req ->
   d_step{
    if
```

```
    :: msg_src==cp1.id -> cp1.syncAckRecv = true; // cp1 got sync from cp2
    :: msg_src==cp2.id -> cp2.syncAckRecv = true; // cp2 got sync from cp1
    :: local_sid < msg_sid ->   // req for next service
       cp1.syncAckRecv = true;  // cp1 and cp2 interact directly
       cp2.syncAckRecv = true;
    fi;
  }
  if
  :: cp1.syncAckRecv==true && cp2.syncAckRecv==true ->
     state = listening;        // cp1 and cp2 interact directly
     break;                    // service routing no longer required
  :: else -> skip;
  fi;
```

4.3.4.2 Service with synchronisation partner

Applying and searching A service provider applies for a service with synchronisation partner and stores its predecessor in `cp2` as any other applicant. Once it receives the token it switches to searching to issue a request for the next required service. In contrast to applicants without a synchronisation partner, the provider switches thereafter immediately to the synching state and waits there for service announcements. The searching state models this as follows:

```
:: timeout ->
   ... // broadcast request
   if
   :: s_sync == true -> state = synching; break; //await ads in synching
   :: else -> skip;
   fi;
```

Synchronising In the syncing state, the participant selects the first applicant as its candidate and stores it in `cp1`. When the discovery time is up, the participant sends a synchronisation message with its merge candidate id (`cp1.id`) to the participant stored in `cp2`:

```
:: msg_type==ad && msg_sid==local_sid && cp1.id==0 && meSyncSend==false ->
   cp1.id = msg_src;                    // my merge candidate
```

```
:: timeout ->                                   // discovery time is up
   if
   :: cp2.syncAckRecv == false ->               // my sync has not been received
      send(sync, channels[(cp2.id-1)], ..., _pid, cp1.id, local_syncid);
   :: else -> skip;
   fi;
```

In the model, `cp2` is the participant's predecessor who changed its role from being a split service to a service router. The original protocol uses a service routing algorithm to determine the next hop for a synchronisation message. The PROMELA model masks such process-internal computation and focuses on the message exchange between synchronisation partner regardless of how their service routers are determined. As long as service routing itself is part of the property verification for service flows, it is sufficient that both synchronisation partners use the same service router (which is their predecessor and former split service provider).

When the participant receives a `sync` from its partner, it overrides `cp2` with the details of the partner. This indicates that the participant starts interacting with its partner and no longer relies on further service routing. Once the participant received the `sync` from its partner and `syncAck` from the service router that indicates its own `sync` is en route to the partner, the participant selects the common provider. Depending on the selection outcome it either needs to search and synchronise again, or hands over the composition control:

```
if
:: (cp2.candidate_id==0 && cp1.id==0) ->
   state = searching;                           // no partner found a candidate
:: else ->
   if
   :: cp2.candidate_id>0 && cp1.id>0 && _pid>cp2.id ->
      send(rel, channels[(cp1.id-1)], ...);     // release my candidate
      cp1.id = cp2.candidate_id;                // select partner's candidate
   :: cp2.candidate_id > 0 && cp1.id==0 ->
      cp1.id=cp2.candidate_id;                  // select partner's candidate
   :: else -> skip;                             // select my candidate
   fi;
   s_merge = true;                              // successor is a merge service
```

```
    state = handover;                          // wait for ack in handover
    send(tok, channels[(cp1.id-1)], local_sid, _pid, 0, 0, local_syncid);
fi;
```

4.3.4.3 Merge service

Applying For a merge service the behaviour in the applying state changes because it needs to wait for tokens from two predecessors. The model represents this as follows:

```
:: msg_sid == local_sid && msg_type == tok ->
    send(ack, ...);                                  // ack token received
    if
    :: s_merge == true ->
       if
       :: msg_src == cp1.id -> cp1.tokRecv = true; // first predecessor
       :: msg_src == cp2.id -> cp2.tokRecv = true; // second predecessor
       fi;
       if
       :: cp1.tokRecv && cp2.tokRecv ->
          s_sync = false;                          // sync covered
          s_merge = false;                         // merge covered
          state=controlling;
          controllerPerService[local_sid] =        // set for verification
                 controllerPerService[local_sid] + 1;
          break;
       :: else -> skip;
       fi;
    :: else -> ... //no merge service, same as in sequential model
    fi;
```

Apart from these changes, a merge service works the same way as a service in a service sequence.

4.3.5 Verification

The verification uses the modelled service flow that starts with a split service, continues with two services who are each other's synchronisation partners, merges in a merge service, and finishes with a final sequential service. SPIN activates four participant

processes: One acts initially as the controller of a split service and the other three as listeners. The split provider then becomes a service router and two of the listeners become synchronisation partners. Depending on how SPIN interleaves the processes, the service router may return to listening before the synchronisation partners found a merge candidate. In this case, two listeners apply for covering the merge service.

As in the sequential model, SPIN also activates a process monitor to assert the correct number of controllers per service. Given the way the service flow is modelled and that the initial controller is already set, the two services with synchronisation partners both increment the first field in the `controllerPerService` array and the field's value must be two. The two fields thereafter should contain one provider as they represent the merge service and the final service. Property 3 is correct if the following loop in the monitor process terminates:

```
do
:: (participant[1]@L0 && participant[2]@L0 && participant[3]@L0 &&
    participant[4]@L0) ->
   assert(controllerPerService[0]==2 && controllerPerService[1]==1 &&
       controllerPerService[2]==1);
   break;
od;
```

The verification setting for Property 1 and Property 2 remains unchanged. Running SPIN with the parallel PROMELA model does not produce any errors and confirms the correctness of the three properties for synchronising parallel service flows. Figure A.3 in Appendix A shows the corresponding output of the model checker.

4.4 Chapter summary

This chapter applied model checking based on PROMELA and SPIN to verify the correctness of the composition protocol design. The first part of the verification focused on the composition of service sequences under unreliable communication and featured proactive service announcements, and participation autonomy. The second part verified that the extensions for parallel service flows, namely service synchronisation and service routing, are correct. For both parts, the verification results confirm that the protocol

does not deadlock, terminates in a valid end state, and allocates one service provider per required sub-service. The model created to formally specify the protocol builds the basis for the prototype implementation that is described in the next chapter.

Chapter 5

Implementation

This chapter describes the implementation of the opportunistic service composition protocol. The implementation is written in Java, integrates the network simulator Jist/SWANS, and builds on the PROMELA specification that was introduced in the previous chapter. Figure 5.1 illustrates the main classes of the protocol implementation and overlays them with the chapter outline for further reference. Focusing on the key concepts of the protocol, the chapter contains five sections: Section 5.1 explains how composite participants become mobile entities by linking the protocol implementation to the simulator. Section 5.2 presents the implementation of those process-internal details that the PROMELA model does not adequately cover e.g., the selection and execution of services. Section 5.3 motivates and specifies the format of different composition messages that composite participants exchange during their interaction. Section 5.4 describes how directed broadcasting is realised with the network simulator. Section 5.5 shows the implementation of the cross-layered approach that integrates the service composition layer with the networking layer.

5.1 Mobile composite participants

For the development of mobile composite participants this thesis uses Jist/SWANS (Barr and Kargl, 2005), a discrete event simulator in Java that manages the movement and wireless communication of participants in a mobile ad hoc network (cp. Figure 5.2). Service composition for these participants is enabled by extending the simulator class

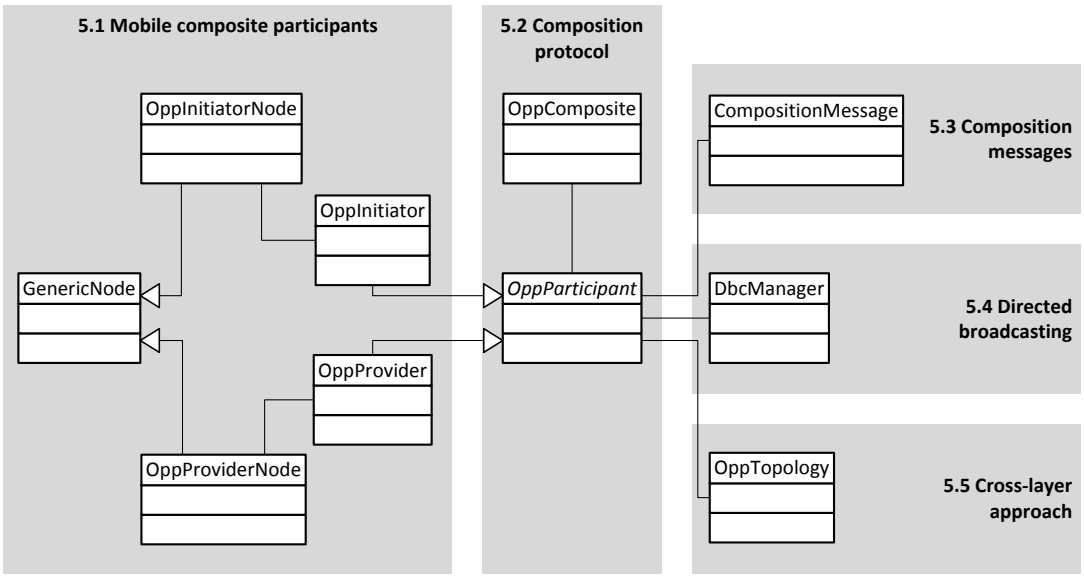


Fig. 5.1: ***Implementation overview*** *The class diagram shows the main components of the protocol implementation for opportunistic service composition and refers to the particular section for further explanations.*

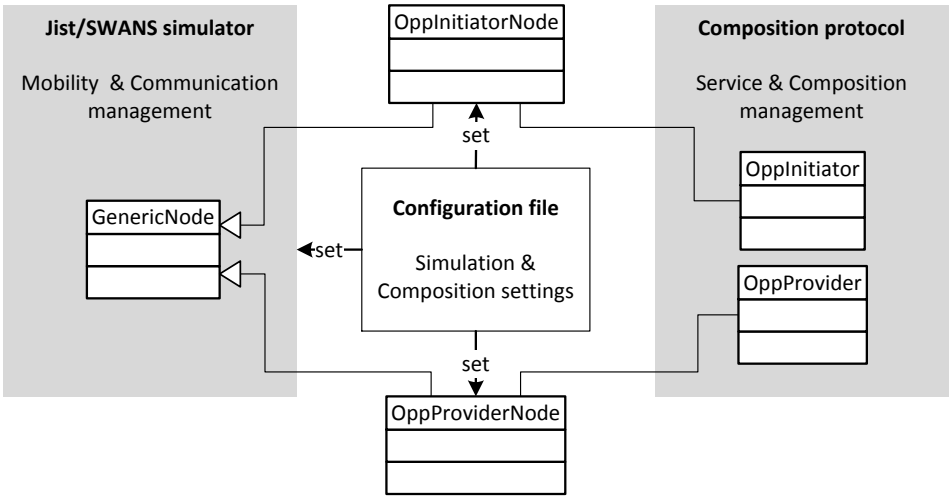


Fig. 5.2: ***Mobile composite participants*** *rest upon the integration of the proposed composition protocol with the Jist/SWANS simulator that manages the mobility and communication of participants in an mobile ad hoc network. OppInitiatorNode and OppProviderNode link the simulator, the composition protocol, and the configuration management.*

GenericNode which assembles the network stack for a mobile participant, links protocol handlers, and places the participants on the simulation field. Classes that extend GenericNode implement addApplication() to specify an application layer protocol, in this case, the proposed service composition protocol. In addition, the simulator supports external configuration files to set simulation and composition-related parameters.

The classes OppInitiatorNode and OppProviderNode link the simulator, the composition protocol, and the configuration management. They extend GenericNode to define the participant's composition-specific behaviour, apply configuration settings, and activate logging for later performance studies. OppInitiatorNode instantiates a participant with a complex service request and as a composite initiator (OppInitiator). OppProviderNode instantiates a participant with a repository of hosted services and the ability to act as a service provider (OppProvider). This way the protocol entities OppInitiator and OppProvider represent two types of mobile composite participants and are set to run the proposed composition protocol.

5.2 Composition protocol

The composition protocol defines how composite participants interact to achieve a complex service request. On behalf of the OppInitiator and the OppProvider, the class OppParticipant implements the state transitions and corresponding actions that an incoming composition message triggers (cp. Figure 5.1). For this, OppParticipant contains the attribute state, the methods send() and receive(), and methods to handle a particular type of composition message. In addition, OppParticipant implements:

- **Bounded search**: The search for a required service is confined to a configurable search radius to provide a stop criterion for otherwise impractical search results. A request message reflects the radius in its TTL (time to live) attribute that indicates whether the recipient should spread the message further.

- **Contact-aware selection**: A participant selects the provider for the subsequent service among a set of applicants based on when it was last in contact with the applicant to increase the probability that the link between them is still intact. If

the same contact time holds for multiple candidates, the applicant with the lowest provider id is selected. This allows for a deterministic choice that is required to find a common merge service provider among multiple synchronisation partners. Optimisations may implement more advanced selection algorithms with multiple selection criteria including the number of the candidate's proponents or the candidate's proximity to all partners.

- **Service execution** A complex service request requires a set of different subservices. For simplicity, services differ in their name and execution time which is configured in the service repository of the OppProvider. The service implementation is the same across all services and returns the increment of a given number. An assigned service provider executes the service, waits for the duration of the execution time, and thereafter proceeds with the next protocol state.

- **Service layer routing**: The participant routes synchronisation messages as defined in Algorithm 2 on page 73. It relies on the class OppComposite to determine its service routing responsibilities.

The class OppComposite represents a locally observed composite and the progress it makes. It stores the composite's structure, a reference to the current position in the composite, and service input data. This is the basis for retrieve methods that return the next required service, a list of all remaining services, or a list of partner services. Further, OppComposite determines service routing responsibilities as defined in Algorithm 1 on page 69 and updates the composite after a composition message was received. The next section describes the format of different composition messages whose content is relevant for OppComposite and other components.

5.3 Composition messages

The protocol relies on the exchange of composite requests, service announcements, tokens, and synchronisation messages to compose a complex service. These messages are embedded in directed broadcasts to allow for their observability and multicast delivery. Figure 5.3 illustrates the different message types and their fields. All fields are manda-

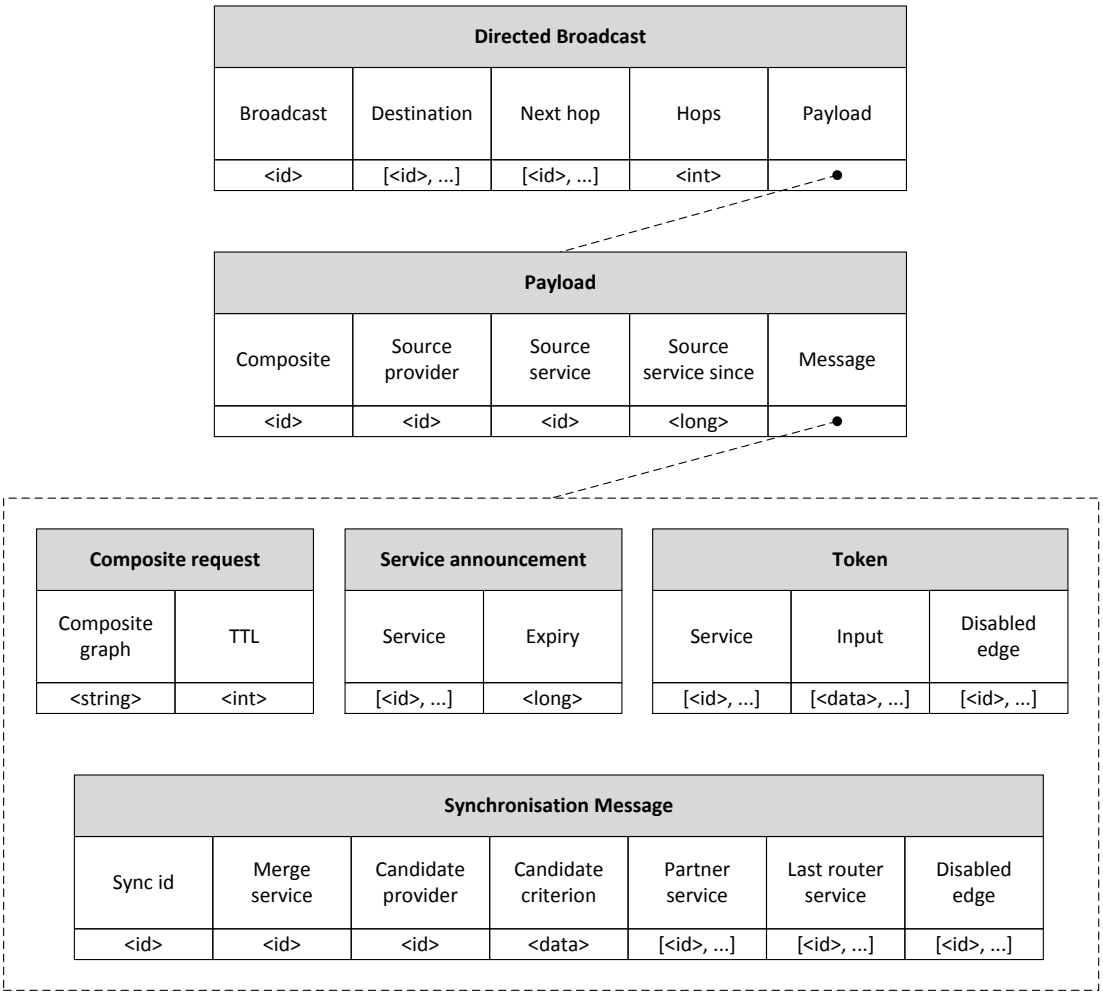


Fig. 5.3: ***Composition messages*** *The proposed protocol defines four types of composition messages: requests, service announcements, tokens, and synchronisation messages. Each message is part of the payload of a directed broadcast message. The description of a message field contains a field name and below its data type which may be a list of items [< type >, ...].*

tory, however, optimisations may consider making some fields optional that are not used in every transmission (e.g., 'Disabled edge' in token and synchronisation messages).

Directed broadcasts contain a unique broadcast id to detect message duplicates, a list of destinations, a list for next hops, the number of hops the message has already travelled, and the actual payload. These fields support message routing in the network. Each destination has a corresponding next hop. The list index maps a destination and its next hop.

Payload subsumes the fields that are part of every composition message, namely the unique composite id to distinguish different composites, the provider and service id of the message source that correlate messages with service dependencies, the date when the source was allocated the service, and the actual message. These fields contain service-related data about the message sender and help the receiver to update its service topology and locally observed composite. With the field 'Source service since' the receiver decides whether to update its composite graph. The receiver refrains from doing so if it received a message from the source with a more recent allocation.

Composite requests contain the string representation of the composite graph and a time to live (TTL) that indicates whether the request should be spread further. With a request, a receiver determines the next required service which is the successor of the message source service in the composition graph.

Service announcements contain a list of services for which the message source applies and an expiry date of the announcement to indicate when the message source will, at the latest, release blocked resources. This is a timeout strategy in case dedicated release messages cannot be dispatched due to a broken network link.

Tokens carry a list of services, their corresponding inputs, and a list of disabled edges that inform the receiver where to reduce its local composite graph. A service and its input are correlated via the list index. Similarly, the list index identifies the allocated service provider from the list of destinations (cp. field in directed broadcast). Destinations that

do not have a corresponding entry in the list of services are redundant and the token signals them to release blocked resources.

Synchronisation messages enclose a synchronisation id to distinguish subsequent synchronisation iterations, the merge service id, and the candidate and its selection criterion. The candidate criterion is used during the allocation of a common merge service provider to find an agreement among multiple synchronisation partners and their candidates. In addition, the message contains a list of partner services, for each partner the last router service, and a list of edges that got disabled during the composition. The list of partner services maps to list of destination ids (cp. field in directed broadcast) via the list index. The destination/provider of a partner service may first contain the provider id of a service router and en route change to the actual partner provider id. The field 'Last router service' is essential for service layer routing as a recipient determines with that and the local composite graph whether to relay a synchronisation message forward or backward in the service path (cp. Algorithm 2 on page 73). The final destination of a synchronisation message extracts partner information from the payload fields 'Source provider' and 'Source service'.

5.4 Directed broadcasting

Directed broadcasts are part of ordinary broadcast messages. They are sent from the application layer directly to the IP network layer, surpassing the transport layer and omitting any details regarding TCP or UDP. The class DbcManager wraps composition messages into a directed broadcast and determines, for each destination address, its next hop on the network route. If the next hop is unknown, the DbcManager performs a route discovery. In addition, it relays messages toward their destination, acknowledges the receipt of messages, and resends unacknowledged messages after a timeout. In this respect the DbcManager is similar to an AODV (Perkins et al., 2003) routing entity that represents a common routing protocol for mobile ad hoc networks. It differs from an AODV entity in that it handles messages that contain a set of destinations. The DbcManager organises its knowledge about the network topology in the class OppTopology

which is central to the cross-layer approach and described in the next section.

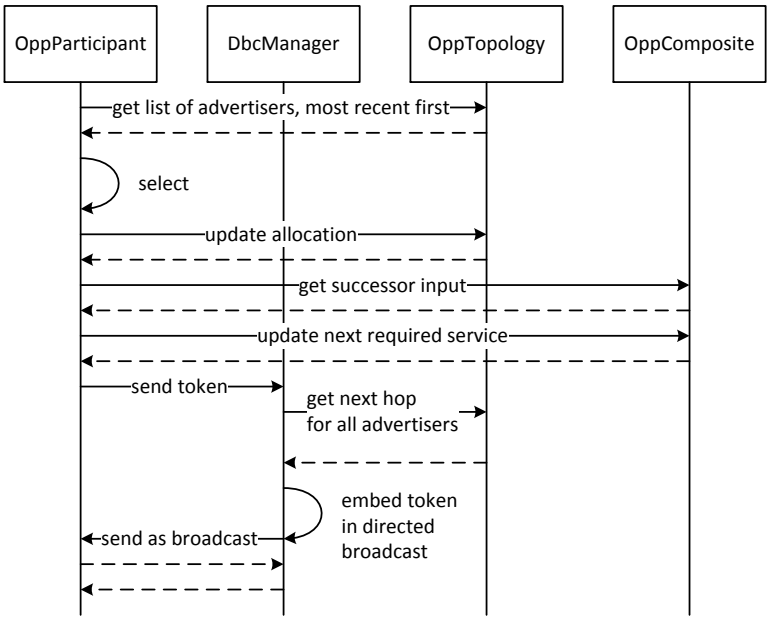
5.5 Cross-layer approach

The class OppTopology represents a cross-layer data repository that provides access to both the OppParticipant as a composition layer entity and the DbcManager as a network layer entity. It holds information about the service and network topology which get updated with each incoming and outgoing message. The stored data refers to the fields of a directed broadcast and include, for example, the sender's allocated service and provider id, the last direct contact with a participant, or the next hop toward a destination. OppTopology extends a standard routing table with service-related data and provides methods for service composition that are aware of the mobile operating environment. Such methods decide whether a controller is communication range to support proactive service announcements, retrieve a list of advertisers sorted by the most recent neighbour to support contact-aware selection, and determine whether the provider id of a synchronisation partner is known to support service layer routing.

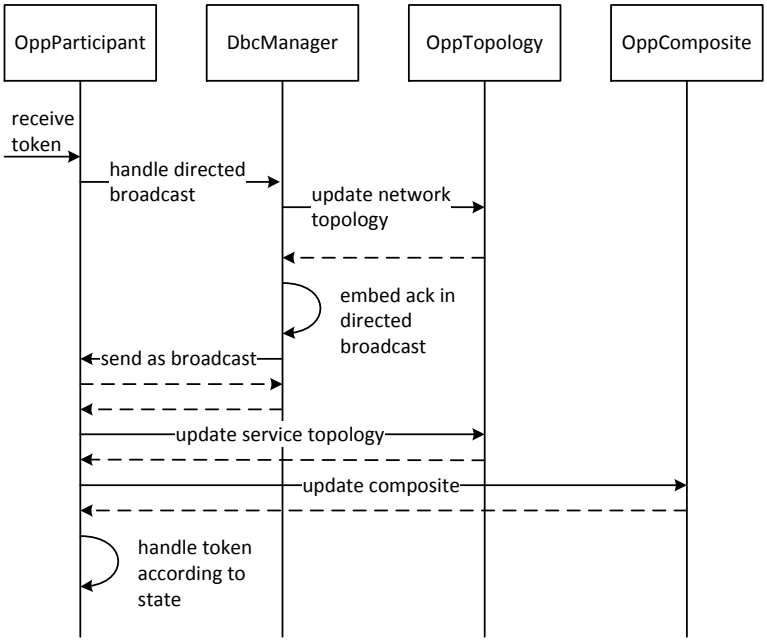
The sequence diagrams in Figure 5.4 show the interplay of the OppTopology with other classes and illustrate the cross-layer approach as well as directed broadcasting for sending and receiving a token message.

5.6 Chapter summary

The description of the implementation focused on the key aspects of opportunistic service composition, namely mobile composite participants, the composition protocol, types of composition messages, directed broadcasting, and cross-layer management of topology data. The integration with the Jist/SWANS network simulator allows for the mobility and configuration of composite participants and builds the basis for evaluating the protocol's performance. The main concern of this implementation is to provide an initial prototype for the experimentation and comparative study with other approaches that is described in the next chapter.



(a) Send token message



(b) Receive token message

Fig. 5.4: ***Cross-layer approach*** *The composition layer entity OppParticipant and the network layer entity DbcManager manage their topology information in one repository, the OppTopology. The sequence diagrams illustrate the interplay of the main components of the protocol implementation to send (a) and receive (b) a token message.*

Chapter 6

Evaluation

This chapter evaluates how well the protocol for opportunistic service composition addresses the overall objective of reducing the failure probability of service composites in dynamic ad hoc environments. The evaluation tests the hypothesis (cp. Section 1.3 on page 9) that the later the operating environment is explored to compose a complex service request, the less time there is for the system to change and to render composition decisions invalid. The proposed protocol has been designed to address the design objectives that are associated with that hypothesis and has been verified with regard to correctness properties. The benefit of the protocol over existing solutions remains to be assessed. This chapter evaluates the protocol in comparison to other solutions and for a variety of settings. In particular, it examines the impact of the composite structure and operating environment on the protocol's performance. For this, the chapter is organised in two parts: In the first part the experimental setup (cp. Section 6.1) describes the general settings, evaluation scenarios, failure types, metrics, and baselines used in the experiments as well as threats to the study's validity. The second part presents and analyses the results of the experiments (cp. Section 6.2).

6.1 Experimental setup

The evaluation of the proposed protocol is based on simulations because they allow for controllable, repeatable, and scalable experiments. Despite being established and widely used in research on mobile ad hoc networks (Kiess and Mauve, 2007), simulations limit

Table 6.1: *General simulator and composition settings*

Parameter	**Configuration**
Simulator	Jist/SWANS Ulm edition
Simulation time (sec)	1300
Initial placement	Uniformly distributed
Mobility model	Random Waypoint
Pause (s)	0
Initial network and service cache	Empty
Communication protocol	IEEE 802.11
Communication range (m)	100
Services hosted per provider	all required
Service execution time (ms)	10-100
Composite source and destination	Composite initiator
Composite allocation policy	Unique providers
Composite per provider	1
Maximum search radius	6-hop neighbourhood
Participants in total	150
Sample size	100

the validity of the results as Section 6.1.6 will outline in more detail. The discrete event simulator Jist/SWANS is used to simulate node movement, wireless communication, and different composition protocols. The following description explains the general settings and the specifics of the evaluation scenarios. It then outlines potential failure types, performance metrics, and baseline implementations to run the experiments.

6.1.1 General settings

The general simulator and composition settings are summarised in Table 6.1 and can be described as follows:

Mobility and communication In the simulation, the potential composite participants are initially distributed uniformly over the simulation field. When they start moving they do so according to the Random Waypoint mobility model which is commonly used in simulations for mobile ad hoc networks (Kurkowski et al., 2005). The

simulator uses a warm-up time of 1000 seconds because in many simulations the mobility model converges towards a steady state during this time (Navidi and Camp, 2004). In addition, participants move at minimum speed greater than zero because Yoon et al. (2003) showed that this achieves steady state even for small minimum speeds. Varying participant speeds are set in the scenarios. Initially, the participant's cache holding information about the network and service topology is empty. This reflects environments without dedicated infrastructure and pre-existing network links. The network evolves when a service request is raised. Participants then communicate wirelessly using IEEE 802.11. Their transmission range is 100 metres, which is similar to the maximum range of Bluetooth technology that is present in today's smartphones (West, n.d.).

Service provision Each composite participant that is initialised as a service provider hosts all required services to shift the focus from service discovery to the overall process of service composition. In turn, a complex service request requires the allocation of unique providers for its sub-services to test the most dynamic behaviour of the composition. The execution time of a particular service in a composite is uniformly distributed between 10 and 100 milliseconds and is the same on each provider. A service provider commits only to one composite at a time and does not respond to any other composites until it has been released. This simulates local resource-constraints and creates sufficient dynamics in the service provision such that a random variation of the provider's willingness to participate is omitted.

Operating environment The simulator places in total 150 potential composite participants on the simulation field. The ratio between service providers and composite initiators is scenario-specific. If multiple initiators exist, they issue the same composite request and stress test the protocol in terms of service demand. The size of the simulation field also depends on the scenario. Different sizes at a constant number of participants control the network density and the number of one hop neighbours. Generally, the simulator configuration ensures that participants are on average within a three hop reach and that the average degree of network partitioning is zero. This is to reduce the impact of the underlying routing protocol and gives messages the opportunity to get

delivered. The settings for the simulation field in combination with the communication range and number of participants base on the suggestions of Kurkowski et al. (2006).

Simulation The simulation starts all composite initiators at the same time and runs until each initiator has received the final result or a failure notification. After the warm-up time of 1000 seconds, the simulator runs for additional 300 seconds to allow for each initiator to issue a request and receive the response. The total simulation time amounts to 1300 seconds. The simulation of one experimental setup is repeated 100 times (unless stated otherwise), each time with a different initialisation of the pseudo-random generator to vary the simulated mobility and communication model. This sample size ensures high confidence in the outcome of the experiment because the resulting 95 percent confidence interval is for all metrics sufficiently narrow.

6.1.2 Evaluation scenarios

The following four scenarios present, based on their evaluation objective, the details of the simulation setup. They differ in terms of the service demand which represents the percentage of composite initiators in the network, the network density which is controlled by the field size, the participant speed, and the structure of the tested composite. Table 6.2 shows which parameters apply to a particular scenario in addition to the general settings in Table 6.1.

Scenario 1: Impact of service sequence length The sequential order of required services is the basic structure of a composite. The objective of this scenario is to explore how the length of a service sequence affects the performance of the composition protocol. For this, a single composite initiator varies, in different experiments, the number of required sub-services and issues the request in a medium-dense network of slow moving service providers. If the proposed protocol does not show any improvement over its baselines under such ideal operating conditions it is not worth evaluating the protocol any further because the other scenarios will challenge the protocols even more.

Table 6.2: *Scenario-specific settings*

Parameter	**Configuration**		**Scenario**			
			1	2	3	4
Service demand/	single initiator		•	•		
Percentage of initiators (%)	1	low			•	•
	5	medium			•	•
	10	high			•	•
Network density/	300x300	dense			•	•
Field size(m x m)	500x500	medium	•	•	•	•
	820x820	sparse			•	•
Participant speed (m/s)	1-2	slow	•	•	•	•
	2-8	medium			•	•
	8-13	fast			•	•
Service sequence length	4		•			
	5		•		•	
	6		•			
	7		•			
Service flow type	2			•		•
	3			•		
	4			•		
	5			•		
	6			•		
	7			•		
Sample size	100		•		•	•
	1000			•		

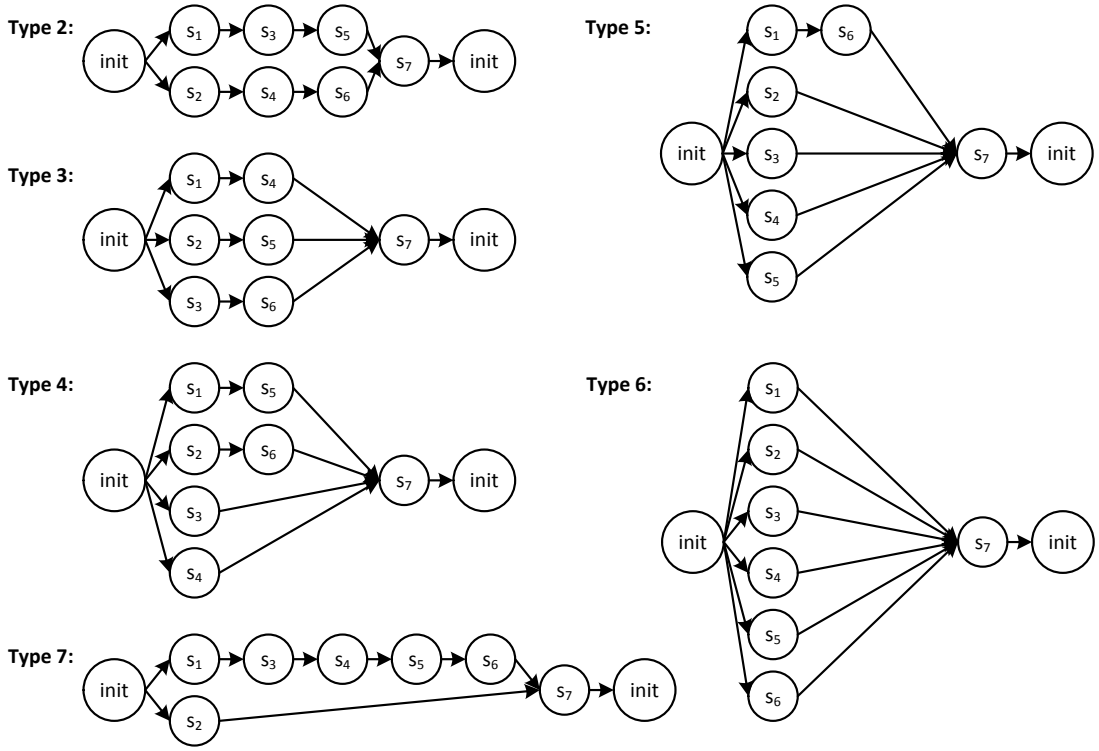


Fig. 6.1: ***Service flow structures*** *vary in number and length of parallel execution paths. Scenario 2 uses them to assess the performance of the proposed service layer routing algorithm. The type names are for ease of reference and start with 2 as type 1 refers to the service sequence.*

Scenario 2: Impact of service flow structure Composites contain parallel execution paths e.g., to improve the quality of the final result. The objective of this scenario is to investigate how the performance of the composition protocol changes if service sequences get replaced with service flows that merge into a common service before reaching the composite's final destination. Adopting the methodology described by Atluri et al. (2007), the scenario tests, in multiple experiments, different types of flow structures that contain seven sub-services and vary in number and length of parallel paths (cp. Figure 6.1). During the composition, all directed edges between services remain enabled, representing composites with AND split and AND merge services. This is to test the performance of the service layer routing algorithm for different numbers of synchronisation partners. Further, the sample size increases to 1000 data points per experiment because pilot studies prior to this evaluation showed that then a sufficiently narrow confidence interval of 95 percent can be maintained.

Scenario 3: Impact of environment using a service sequence The operating conditions for composite requests may not always be as ideal as in single-initiator and medium-dense environments. The objective of this scenario is to explore how the composition performance for a service sequence changes under different settings in the operating environment. This includes the variation of three impact factors: First, the variation of the network density between sparse, medium-dense, and dense. Second, the variation of the participant speed between slow (walking), medium (cruising), and fast (driving). Third, the variation of the service demand between low, medium, and high. The network density depends on the field size which controls the number of neighbours in direct transmission range. The different field sizes are chosen to keep the impact of the routing protocol low and still challenge the composition protocol with sparse and dense networks. The service demand depends on the percentage of initiators in the network. Already a small number of initiators stress tests the protocol because the initiators issue the exact same request at the exact same time. Combining the three impact factors each with three levels of variation supports a detailed analysis of the protocol.

Scenario 4: Impact of environment using a service flow This scenario runs the same set of experiments as in scenario 3, except that it uses the service flow type 2 as the test composite. The objective of this scenario is to assess the performance of the protocol's synchronisation features in different operating environments.

6.1.3 Failure types

In the simulation, a composite may fail due to network failure, service discovery failure, or provider overload. The associated failure types are defined as follows:

Network failure Wireless communication is unreliable because the composite participants move in and out of each other's range losing messages due to signal collision or broken network links. This is recovered by resending undelivered messages. However, once the maximum number of retries is exceeded, the recovery does not continue (i.e., route recovery is omitted) and the composite fails.

Discovery failure Bound search is common practise in the design of network protocols. In the simulation it constrains the allocation of service providers. If a composite request has reached the 6-hop neighbourhood without finding a service provider, the composition fails.

Overload failure The provider's commitment to one composite at a time implies that if it receives service allocations from two different composites, it handles the first and silently drops the second. The second composite fails because the service layer recovery is out of scope of this thesis and omitted.

6.1.4 Metrics

Three metrics, namely the failure ratio, communication overhead, and response time, assess the performance of a composition protocol in the different scenarios. The failure ratio is measured to assess whether the overall objective of reducing the failure probability of a composite is achieved. Measuring the communication effort follows from design objective 4 (cp. Section 3.1 on page 43), which reasons that less communication is a way to achieve the overall objective. Recording the response time allows for studying possible trade-offs in the protocol. The three metrics are defined as follows:

Failure ratio A composite request has failed if the initiator does not receive a valid composition result from the last required service in the composite. The failure ratio is the number of failed composites relative to the total number of composite requests that have been issued. For a single initiator, the total number of requests is equal to the sample size as the initiator issues one request per run of the experiment. In experiments with multiple initiators, one run involves multiple requests, one for each initiator. The total number of requests over all runs is then equal to the sample size multiplied by the number of initiators. A high failure ratio means poor protocol performance.

Communication effort The communication effort in this evaluation comprises the messages sent from the composition and network layer and is recorded on the MAC layer. This includes composition messages as well as messages for route discovery and

Table 6.3: *Comparison of baselines CiAN* and Yil with proposed protocol Opp*

	CiAN*	**Yil**	**Opp**
Service announcement	Proactive	On demand	
Service selection	Most recent neighbour first		
Service allocation by	Initiator	Predecessor	
Synchronisation	inapplicable	Continuous path updates	Service layer routing
Service invocation	Decentralised		
Communication	Non-observable AODV unicast	Observable directed broadcast	
Message format	Non-standard		

acknowledgements. It excludes any other network layer messages e.g., those for route maintenance. Counting only the sent messages and not the received messages reduces the impact of the network density that is not specific to the tested composition protocol. The communication effort counts all sent messages regardless of whether the composition was successful or has failed to reflect the strain of service composition on the network. A high communication effort means poor protocol performance.

Response time The response time is the delay from the composite initiator sending the composite request to its receiving the final result. The response time is measured for composite requests that complete successfully without failure. A high response time indicates poor protocol performance.

6.1.5 Baselines

The evaluation uses two baselines, in the following referred to as CiAN* and Yil, to compare the performance of the proposed protocol. Similarly to the proposed protocol, the baselines are implemented as part of the application layer in the simulator and run the same set of evaluation scenarios. The next two paragraphs motivate the choice of the baselines and present their implementation details. Table 6.3 summarises the comparison of the baselines with the proposed approach, hereafter referred to as Opp.

CiAN* Although the research community has invested substantial effort into decentralising service composition, solutions for mobile ad hoc networks are only emerging. CiAN (Sen et al., 2008) is a composition engine for such networks, which allows for a broad range of comparison because it differs in many aspects from the proposed protocol. In particular, CiAN is a semi-centralised approach in which service providers announce themselves proactively and the initiator allocates all required services at once before decentralised service invocation starts. The comparison with this state-of-the-art composition solution allows for valuable insights on the different approaches to service composition in dynamic ad hoc networks. CiAN offers also a distributed planning mode (Sen et al., 2007), however, this mode is less suitable because it assumes that service providers, once they complete a service, move towards an priori chosen service allocator to collect the next service request. In other words, the distributed planning mode assumes a leader-team relationship that does not reflect autonomous composite participation. The evaluation uses our adaptation of CiAN, in the following referred to as CiAN*, which is modified in two ways: First, CiAN* delivers allocation decisions over multiple hops because original CiAN fails if a service provider does not come into the initiator's direct transmission range to collect its allocation decision. The failure ratio in Figure 6.2 highlights the impact of this change as CiAN* fails 30 percent points less than CiAN. Second, despite our best effort to preserve the original content-based publish-subscribe mechanism to deliver messages, we found that AODV (Perkins et al., 2003), a common routing protocol in mobile ad hoc networks, incurred less communication overhead and used it instead. The communication effort in Figure 6.2 shows the difference between dispatching messages directly with AODV in CiAN* and gossiping messages based on their content to multiple intermediaries in original CiAN. CiAN*'s improvement with regard to the failure ratio and communication effort, outweighs its slightly increased response time over original CiAN (cp. Figure 6.2) and supports its choice as baseline.

Yil The late allocation of a merge service by its predecessors, as proposed in this thesis, is not widely used. Most composition solutions, like CiAN, assign a provider in advance and avoid the need of multiple parties to synchronise their allocation decision.

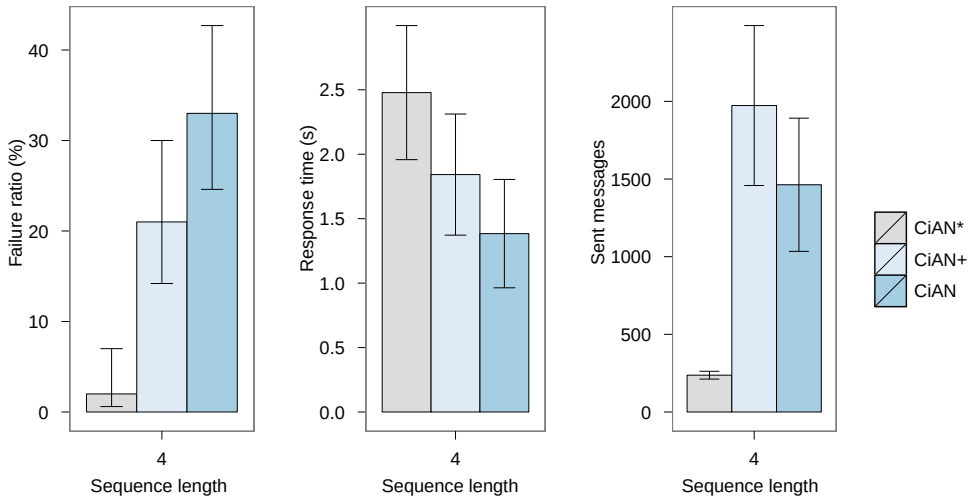


Fig. 6.2: ***Pilot study for CiAN*** *The study runs scenario 1 with a sequence of 4 for services (cp. Table 6.2) to assess the performance of the original CiAN, CiAN with multihop allocation delivery (CiAN+), and CiAN with multihop allocation delivery and AODV routing (CiAN*). CiAN* achieves a significantly lower failure ratio at lower communication effort than the other two versions and motivates its use as a baseline.*

Yildiz and Godart (2007), however, approached this topic and suggest a synchronisation algorithm that relies on continuous path updates to introduce synchronisation partners. This algorithm is used to compare the performance of the proposed service layer routing algorithm. The corresponding baseline Yil implements the same underlying composition protocol as the proposed opportunistic approach, exchanges messages via directed broadcasts, and composes services in a decentralised interleaved manner. As Yil differs only in the way partners synchronise their allocation decision, it is used in experiments with parallel service flows and omitted for service sequences.

6.1.6 Threats to validity

The following aspects of the experimental setup potentially limit the conclusions that can be drawn from the simulation study in comparison to a realistic test environment.

Simulation Although simulation is an established and widely used evaluation method for protocols in mobile ad hoc networks, its simplification of the real-world may lead to results that do not reflect the actual protocol behaviour in the real world (Kiess and

Mauve, 2007). In particular, using a model for mobility and radio propagation may produce different results in the simulation than in a realistic deployment. However, as the simulation setup is the same for all tested composition protocols, the simulation allows for comparing their performance trends.

Provider participation Under the premise that the service provider has sufficient resources, the study assumes the provider's willingness to participate in a composite is always positive. In reality, mobile device users may carefully weigh the cost and benefit of sharing the resources of their devices. They may not participate at all if the right incentives are missing. From this perspective, the study is rather idealistic as it limits the number of available service providers only by the network density and the restriction that each provider handles one composite at a time.

Network partitioning A low degree of network partitioning, as used in the study, allows for the delivery of messages to their destination and is desired when evaluating the performance of a protocol in mobile ad hoc networks (Kurkowski et al., 2007). In realistic test environments, however, the degree of network partitioning may vary depending on the network density and the willingness of mobile devices to relay messages. In addition to signal collision and stale routing information, network partitioning presents an additional source for composite failure on the network layer.

6.2 Results and analysis

This section presents the results and analysis of the simulation study. Its structure follows the evaluation scenarios: The first two subsections examine the impact of the composition structure, namely the length of service sequences (scenario 1) and the structure of service flows (scenario 2). The two subsections thereafter study the impact of the operating environment using a service sequence (scenario 3) and a service flow (scenario 4). Each subsection first presents and analyses the outcome of the evaluation metrics and then summarises how the results address the evaluation objective. The error bars in the figures represent the 95 percent confidence interval for the results.

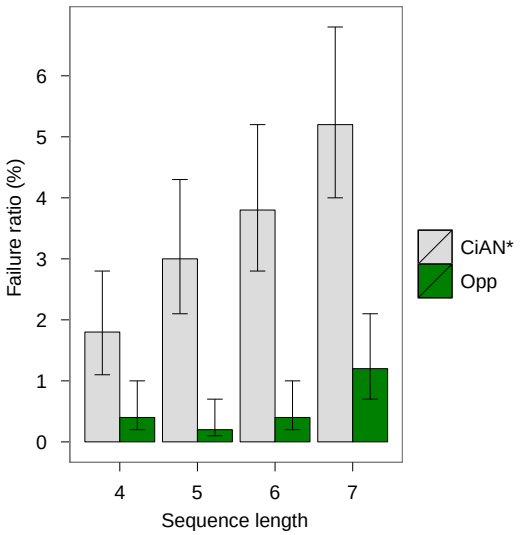


Fig. 6.3: ***Failure ratio in scenario 1*** *For different lengths of a service sequence, the opportunistic protocol fails less than CiAN* because it is less prone to the unreliable network.*

6.2.1 Impact of service sequence length

The first set of results corresponds to evaluation scenario 1 (cp. Table 6.2) and studies how the length of a service sequence affects the performance of the composition protocol.

6.2.1.1 Failure ratio

The failure ratio in Figure 6.3 shows that the proposed opportunistic protocol fails less composite requests than CiAN*. The improvement of 6 percent points is highest for a sequence with 7 sub-services. Generally, the sequence length affects the proposed approach less than CiAN* for which the failure ratio increases by almost 1 percent point per additional service. The difference between both composition protocols is statistically significant as their confidence intervals do not overlap. With regard to the source of failure, the opportunistic protocol fails mostly due to signal collisions. CiAN* is more prone to stale routes, in particular when the last service provider returns the final result to the initiator.

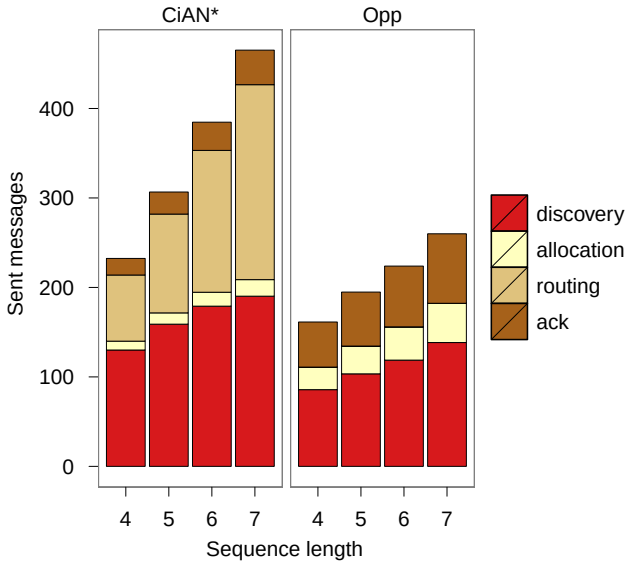


Fig. 6.4: ***Communication effort in scenario 1*** *For different lengths of a service sequence, the opportunistic protocol sends fewer messages than CiAN*. It benefits from a localised demand-based service discovery which establishes network links without additional routing layer messages.*

6.2.1.2 Communication effort

The graphs in Figure 6.4 show the communication effort for the proposed opportunistic protocol and CiAN* itemised by different message types. The proposed approach exchanges fewer messages and its communication effort grows less with increasing composition length than for CiAN*. The opportunistic protocol sends almost no routing requests and responses because its on-demand service discovery is localised and establishes routing information which service providers later use to hand over the composition control. In CiAN*, on the other hand, the number of routing related messages is substantial because first the initiator establishes routes to deliver its allocation decisions and then each allocated service provider needs to establish a route to its successor.

6.2.1.3 Response time

The response time in Figure 6.5 shows that across all lengths, the proposed protocol responds more quickly than CiAN*. While CiAN*'s response time increases roughly by 1 second per additional service, the proposed approach's response time remains with about 1 second almost constant. Only the service execution time of each additional

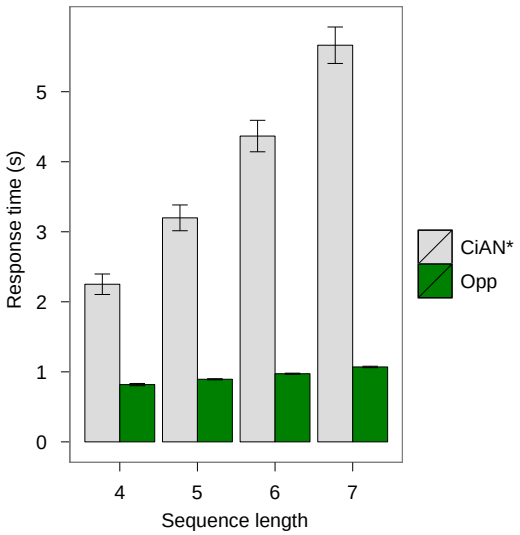


Fig. 6.5: ***Response time in scenario 1*** *For different lengths of a service sequence, the opportunistic protocol responds more quickly than CiAN* because it does not have to acquire additional routing information.*

service causes a small increase. In this and all following experiments, CiAN*'s response time excludes the discovery phase and starts when the composite initiator sends its allocation decisions. Nonetheless, CiAN* experiences longer delays for two possible reasons: First, routing requests cannot be satisfied by the neighbourhood and must wait for a timeout before they can be resent with a bigger search radius. Second, resending lost messages also needs to wait for a timeout as this distinguishes lost from delayed messages. With regard to the communication effort, the primary reason why the proposed protocol responds more quickly is because it does not have to acquire routing information separately.

6.2.1.4 Summary

For service sequences, the novel opportunistic composition protocol outperforms CiAN* in all three metrics: Failing less composite requests, it responds more quickly, and it requires less communication. The length of a service sequence has a smaller effect on the proposed protocol than on CiAN*. The results further show that the protocol's demand-based service discovery in combination with interleaved service invocation establishes knowledge about the network topology without additional routing layer messages, which

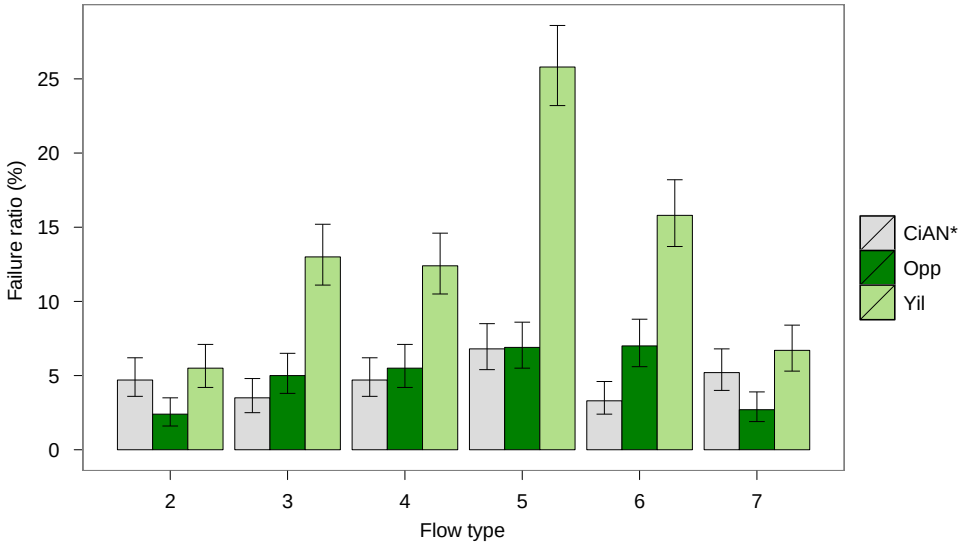


Fig. 6.6: ***Failure ratio in scenario 2*** *For different flow types, the opportunistic protocol maintains its advantage over CiAN* if requests contain few and long parallel paths (flow type 2 and 7). Compared to Yil, the protocol fails less across all flow types.*

keeps the communication effort low and the response time short. These benefits of the novel protocol motivate further investigations of more challenging scenarios.

6.2.2 Impact of service flow structure

For evaluation scenario 2 (cp. Table 6.2) and the next set of results, the request changes from service sequences to parallel service flows to explore the effect of different flow types (cp. Figure 6.1) on the protocol's performance. The analysis focuses on two questions: First, does the proposed protocol maintain its advantage over CiAN*? Second, does the proposed service layer routing outperform the synchronisation algorithm with continuous path updates, which is represented by Yil?

6.2.2.1 Failure ratio

The failure ratio in Figure 6.6 shows that the opportunistic protocol is more affected by the parallel than the sequential composite structure. The protocol's failure ratio increases with the increase of parallelism from flow type 2 to flow type 6. Compared to CiAN*, the proposed protocol maintains its advantage if requests contain few and long

parallel paths as in flow type 2 and flow type 7. If parallelism is high, as in flow type 6, the protocol performs worse than CiAN*. Compared to Yil, the proposed protocol achieves a lower failure ratio across all flow types. Its improvement ranges from 3 percent points for flow type 2 to 19 percent points for flow type 5.

There are two reasons of why the proposed protocol fails more often for service flows than for service sequences: First, after synchronisation partners are agreed, they hand over the composition control at about the same time. The higher the parallelism, the more partners, and the more likely is the collision of signals at the common successor. Second, the protocol fails due to stale routes. The protocol should reduce the risk of stale routes as its interleaved approach establishes connections immediately before they are used. However, finding an agreement among multiple parties introduces delays during which already established routes may expire. Combining the opportunistic protocol with continuous path updates, as in the Yil baseline, increases this effect: Routes that have been established during demand-based service discovery grow stale because the topology changes while a service provider waits for updates from other paths before it hands over the composition control to its successor.

6.2.2.2 Communication effort

The communication effort in Figure 6.7 shows that the opportunistic protocol sends fewer messages across all flow types than CiAN*, despite additional synchronisation messages. Synchronisation messages increase the number of acknowledgements and introduce the need for routing layer messages. The benefit of on-demand service discovery already establishing routing information applies only to consecutive service providers and not to synchronisation partners. For partners the network links have to be established separately. However, the protocol has another advantage: With increasing parallelism, the number of discovery messages decreases because a request contains all required services and a provider can apply for multiple services at once. Compared to Yil, the proposed protocol exchanges slightly less messages because Yil uses the same communication means as the proposed protocol but interacts more often between parallel paths, which requires more routing effort.

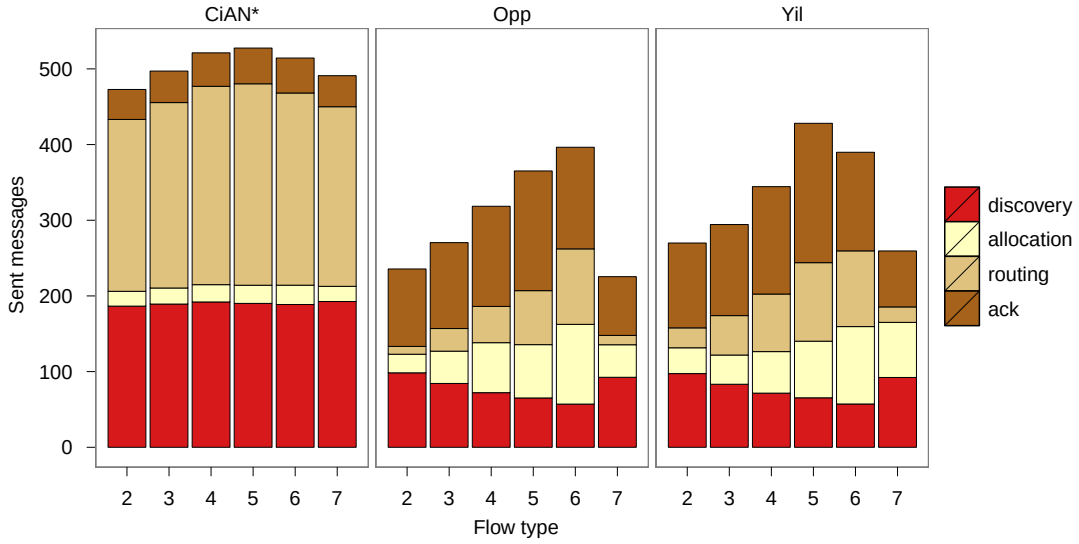


Fig. 6.7: ***Communication effort in scenario 2*** *For different flow types, the opportunistic protocol sends fewer massages than CiAN* because it requires less routing and discovery effort. Compared to Yil, the proposed protocol sends fewer messages because it establishes fewer routes between parallel paths.*

6.2.2.3 Response time

The response time in Figure 6.8 shows that the proposed protocol outperforms CiAN* clearly and Yil slightly. CiAN* returns the final composition result after 5 seconds. Similarly to service sequences, several attempts to discover routing information produce such a delay. The response time for the proposed approach increases with increasing parallelism from 1 to 2 seconds because finding an agreement among synchronisation partners introduces delays. The protocol's low response time despite the need for dedicated routing messages indicates that routing information is already available in the immediate neighbourhood. This demonstrates the benefit of observable composition messages and cross-layer management of topology information.

6.2.2.4 Summary

For service flows, the opportunistic protocol maintains a better failure ratio than CiAN* if parallel paths are few and long. Otherwise, the high number of synchronisation partners delays agreement and causes more signal collision and stale routes. Using service

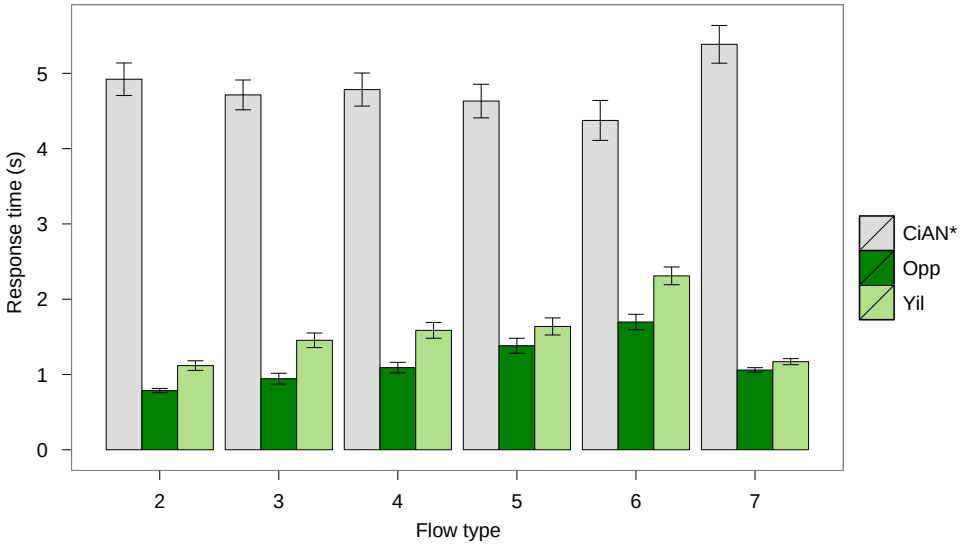


Fig. 6.8: ***Response time in scenario 2*** *For different flow types, the opportunistic protocol maintains a shorter response time than CiAN* because required routing information is already available in its immediate neighbourhood.*

layer routing for synchronisation, the proposed protocol fails significantly less than with continuous path updates as proposed by Yil because fewer interactions between parallel paths reduce the risk of stale routes. In terms of communication, the proposed protocol maintains its advantage over CiAN* although synchronisation introduces additional message overhead. In particular, the benefit of demand-based service discovery establishing routes applies only to consecutive service providers and not to synchronisation partners for which the routes have to be established separately. However, the required routing information is available in the immediate neighbourhood and enables the protocol to respond significantly more quickly than CiAN*. Overall, the proposed protocol maintains its benefits over the baselines if service flows are of moderate parallelism.

6.2.3 Impact of environment using a service sequence

Having evaluated the impact of the composite structure, the next results demonstrate the impact of the operating environment. In particular, this subsection corresponds to scenario 3 (cp. Table 6.2) and investigates how the performance of the proposed protocol changes if it composes a sequence of 5 sub-services in an environment that varies

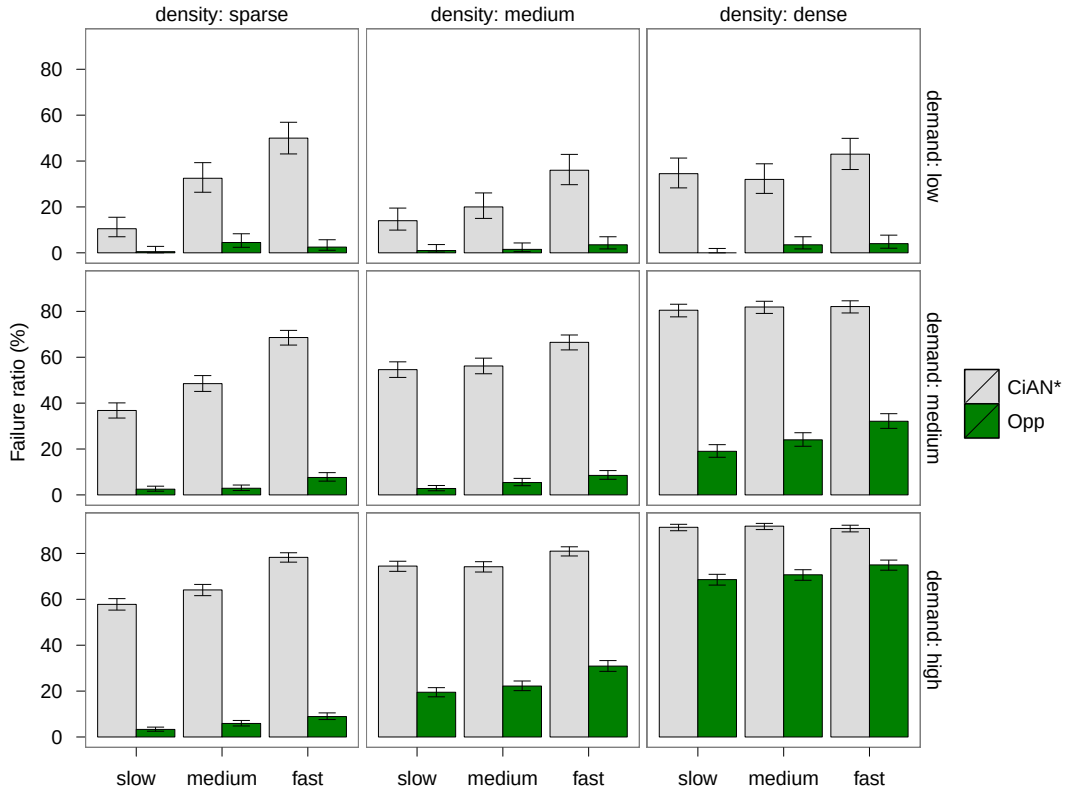
in service demand, network density, and participant speed. For ease of comparison, a 3-by-3 result matrix presents how a metric evolves for these impact factors.

6.2.3.1 Failure ratio

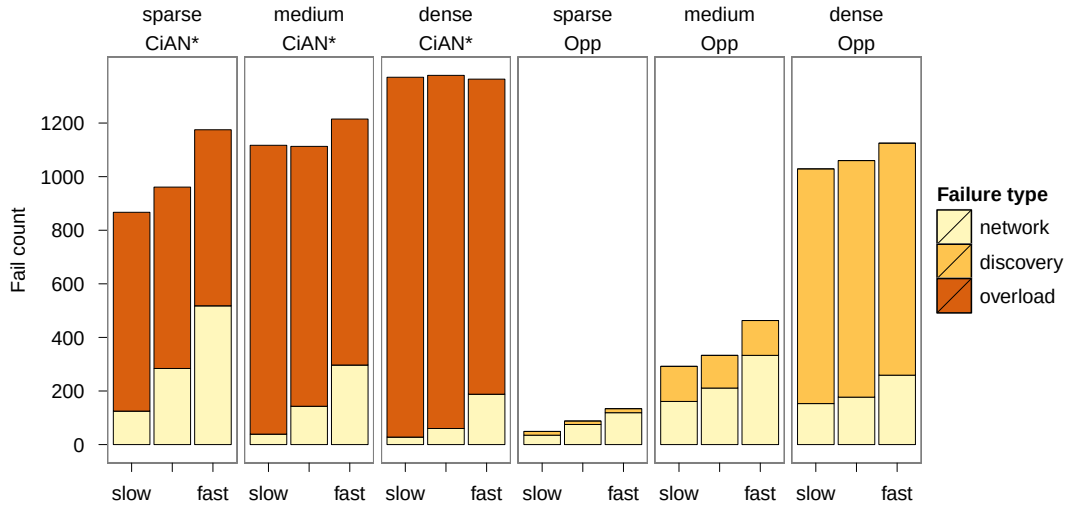
The failure ratio illustrated in Figure 6.9a shows that across all fields of the matrix the proposed protocol fails less than CiAN*. Compared to the results in scenario 1 (cp. Figure 6.3), the benefit of the proposed protocol over CiAN* has increased: In scenario 1 the difference between the protocols amounted to about 2.5 percent points for 5 sub-services. Now in this scenario, it varies between 70 percent points in sparse high-demand environments (bottom left graph) and 20 percent points in dense high-demand environments (bottom right graph).

The proposed protocol performs very well with a maximum of 5 percent failure if the service demand is low (top row). It remains under 10 percent failure in sparse networks (left column) and in medium-dense networks with medium service demand (centre graph). Among the impact factors, service demand and network density dominate the failure ratio while the participant speed has the least effect.

In terms of failure sources, both protocols are prone to network failure. In addition, CiAN* is prone to provider overload whereas the proposed protocol suffers from discovery failure. Figure 6.9b illustrates the number of failures per failure type for high service demand. It shows that discovery failure for the proposed protocol is substantial in dense networks while overload failure for CiAN* occurs even in less dense environments. The proposed protocol avoids overload failure because it allocates only free service providers. In dense environments, however, multiple composite requests interfere with each other and compete for the same set of service providers up to the point where all service providers in the search radius are blocked and a provider for the next required service cannot be found. CiAN* does not incur discovery failure because it waits on proactive service announcements which are eventually disseminated in the network. However, unaware of who else uses the announcement, CiAN* risks overloading its service providers. The increasing network density promotes the dissemination of proactive service announcements and increases the number of composites that have access to the same set of providers.



(a) Failure ratio



(b) Failure types for high service demand

Fig. 6.9: ***Failure ratio in scenario 3*** *For a service sequence and varying operating environment, the opportunistic protocol fails less than CiAN* (a). Its exposure to discovery failure is substantial for dense networks while CiAN* overloads providers already in sparse networks (b).*

6.2.3.2 Communication effort

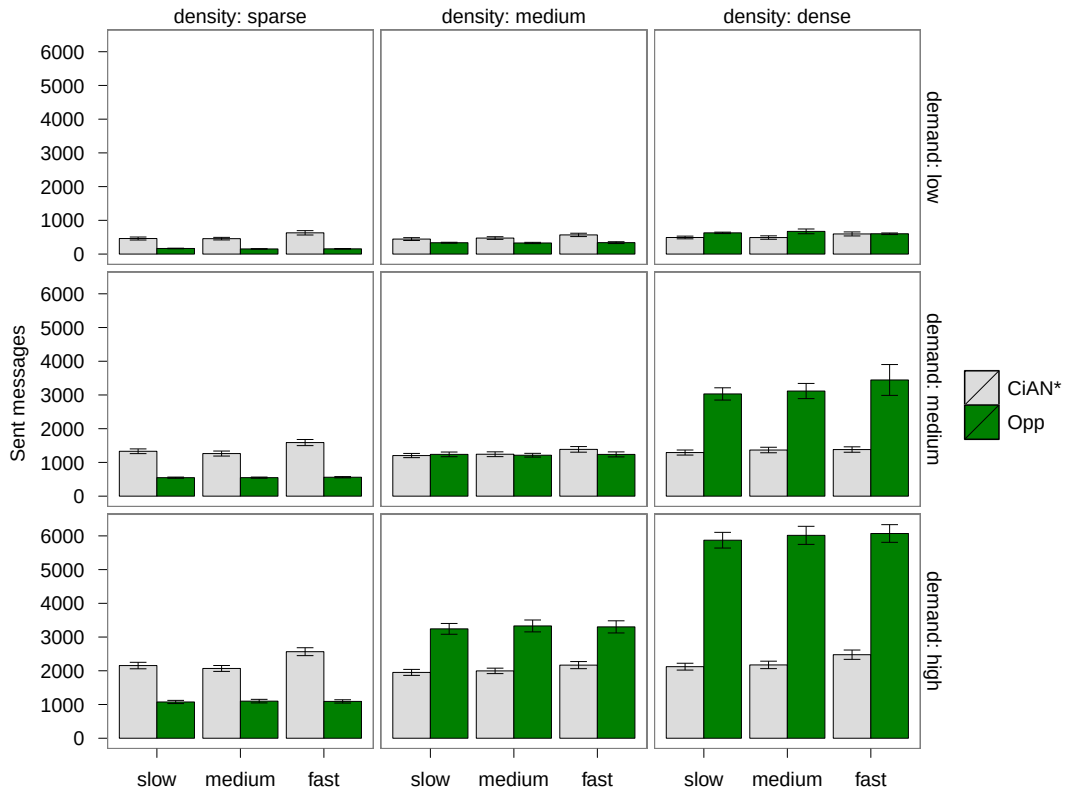
The communication effort illustrated in Figure 6.10a shows that in sparse environments (left column), the proposed protocol sends fewer messages than CiAN*. With increasing network density and service demand (towards bottom right graph) its communication effort, however, increases until it exceeds CiAN*. There are two reasons for this: First, the proposed protocol is still operating to complete a composite, while CiAN* has already failed and has stopped sending messages. Second, the shortage of free providers in the immediate neighbourhood forces the proposed protocol to broaden its search and send more composite requests. Figure 6.10b illustrates that by breaking service discovery messages down into composite requests and service announcements (service ad), CiAN* exchanges mainly routing messages that are unaffected by the network density. The participant speed has no effect on the communication overhead.

6.2.3.3 Response time

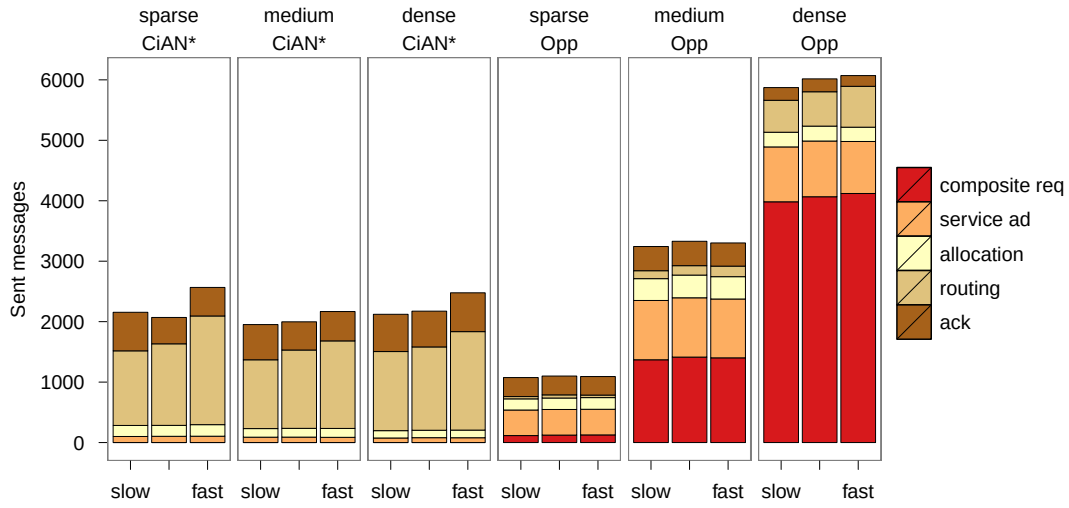
The response time in Figure 6.11 shows that the proposed protocol responds on average 3.3 seconds faster than CiAN* if the network is sparse (left column) or the service demand low (top row). In these cases, it finds free service providers and corresponding routing information in its immediate neighbourhood. As demand and density increase, the response times of both protocols converge until the proposed protocol exceeds CiAN* by 2.2 seconds in dense high demand environments (bottom right graph). However, completing on average 20 percent more compositions in such a setting justifies the delay. With regard to the impact factors, service demand and network density are equally dominant while the participant speed has no effect on the response time. In addition, the lower the network density, the more quickly the proposed protocol responds. For CiAN* the opposite is true.

6.2.3.4 Summary

The proposed protocol continues to show reduced composite failure compared to CiAN*. The improvements regarding the failure ratio reach up to 70 percent points in sparse high-demand networks. This fulfils the protocol's overall objective of reducing the composites'



(a) Total communication effort



(b) Communication effort for high service demand

Fig. 6.10: ***Communication effort in scenario 3*** *For a service sequence and varying operating environment, the opportunistic protocol maintains its advantage over CiAN* in sparse networks (a). With increasing network density it issues more composite requests to find free service providers (b).*

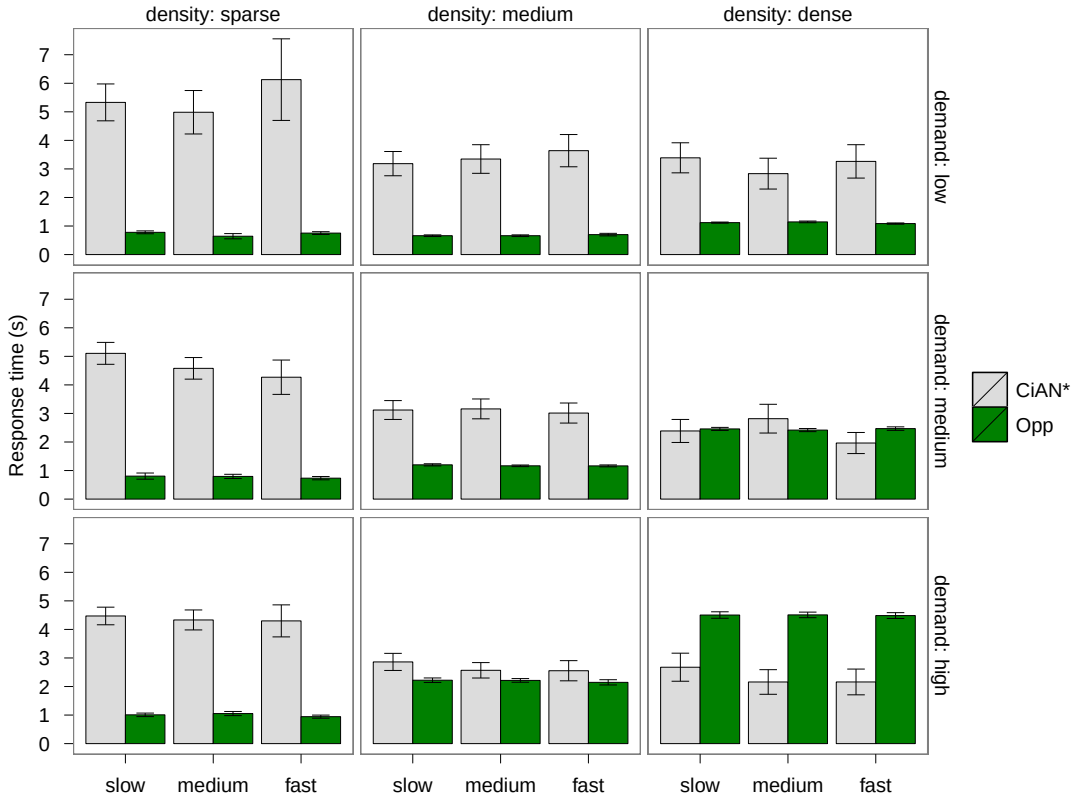


Fig. 6.11: ***Response time in scenario 3*** *For a service sequence and varying operating environment, the opportunistic protocol responds in dense or low-demand settings more quickly than CiAN* as free service providers and routing information are available in the neighbourhood.*

failure probability. In most cases, the proposed protocol achieves a lower failure ratio at lower or similar communication effort and shorter response times. The proposed protocol suffers from discovery failure if network density and service demand increase at the same time. Then, multiple composite requests interfere with each other and compete for the same set of service providers, which gradually leads to more communication, longer delays, and finally to more failure. Among the impact factors, network density and service demand dominate the outcome of all three metrics while the participant speed has only an effect on the failure ratio. The faster the participants move, the sooner routes become stale, and the more likely is the composite to fail. While slow providers communicate to complete a composite request, fast providers incur the same communication effort but to recover lost messages. The response time is measured only for successful composites and its independence of the participant speed is expected.

6.2.4 Impact of environment using a service flow

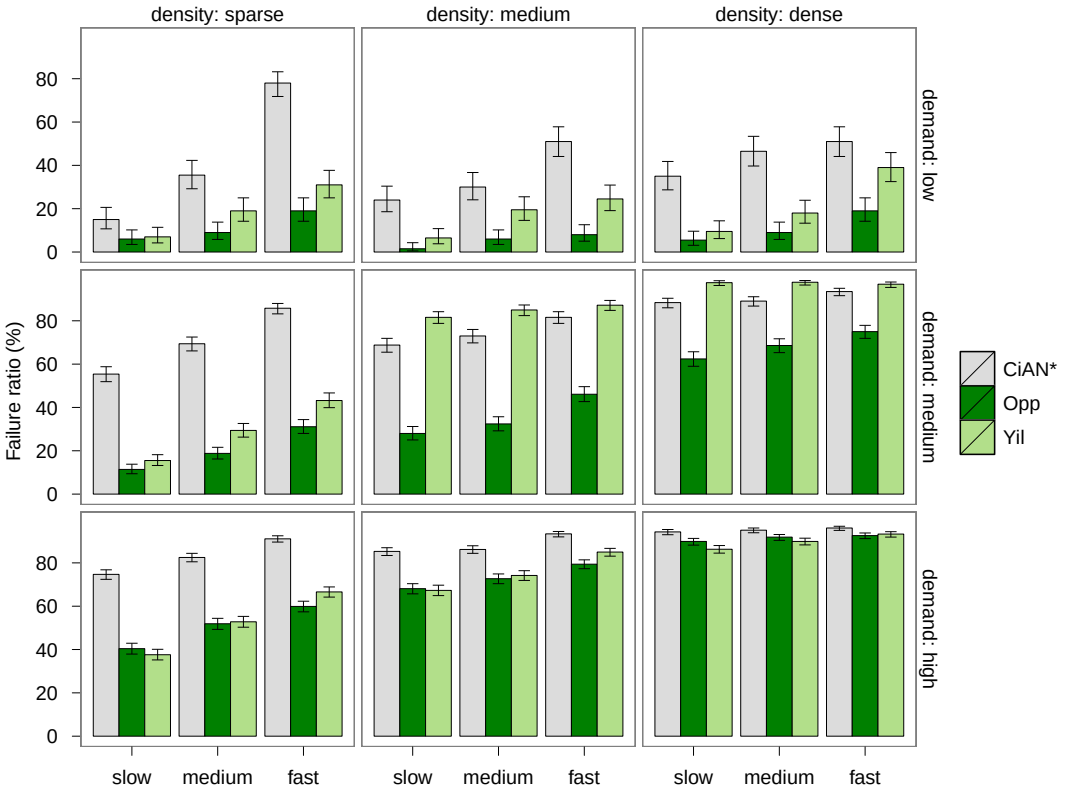
The analysis of previous scenarios showed among other things that the proposed protocol outperforms its baselines for composites with moderate parallelism and that the variations of the operating environment reinforce the protocol's benefits for service sequences. This subsection corresponds to scenario 4 (cp. Table 6.2) and examines whether the protocol maintains its benefit for a moderate flow type 2 and different levels of service demand, network density, and participant speed.

6.2.4.1 Failure ratio

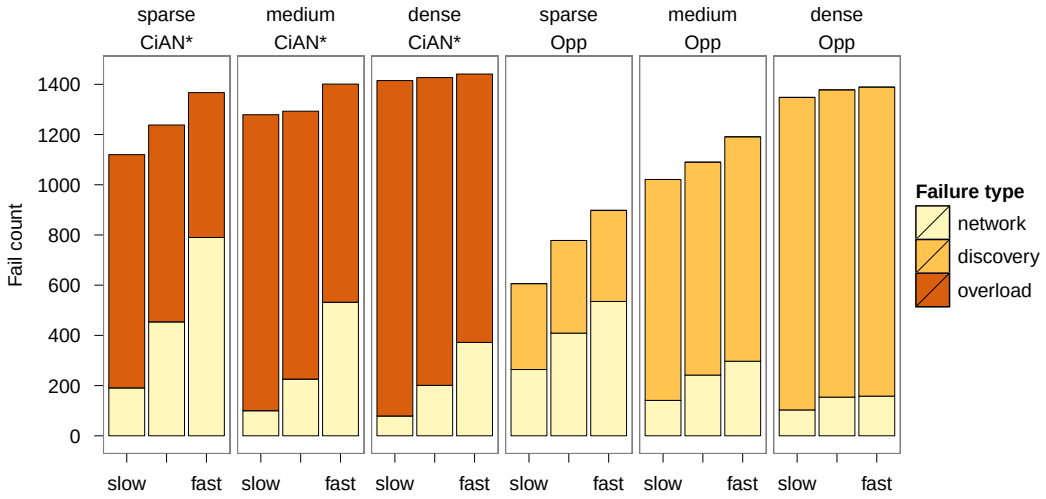
The graphs in Figure 6.12a show that the proposed protocol fails less often than CiAN* in all settings except for high-demand dense networks, where both protocols incur a similar failure ratio. The improvement of the proposed protocol ranges from a minimum of 9 percent points to a maximum of 59 percent points (both in top left graph). With regard to the synchronisation algorithms, the proposed protocol outperforms Yil in some cases, most notably in medium-dense medium-demand networks (centre graph) with an improvement of 53 percent points. In other cases, both algorithms perform similarly and the proposed protocol does not fail significantly more often than Yil.

Compared to the failure ratio in scenario 3 (cp. Figure 6.9a), network density and service demand have a higher impact as the protocol's failure ratio is higher across all graphs in the matrix for service flows.

The source of failure has not changed for the protocols; only its impact has increased. Compared to scenario 3 (cp. Figure 6.9b), the main difference for service flows is that the proposed protocol is prone to discovery failure in less dense networks (cp. Figure 6.12b). The reason is that the two parallel paths in flow type 2 originate from the same composite and compete for service providers in the same search area. Figure 6.12b omits Yil because its behaviour for high service demand is similar to that of the proposed protocol. If the service demand and network density is medium (not depicted), Yil is more prone to network failure than the proposed approach because it requires more interaction between parallel paths.



(a) Failure ratio



(b) Failure type for high demand

Fig. 6.12: ***Failure ratio in scenario 4*** *For a service flow and varying operating environment, the proposed protocol fails less often than CiAN*, except for dense high-demand networks. Its improvement over Yil is most notably in medium-demand medium-dense networks (a). The proposed protocol is exposed to discovery failure in sparse networks because the parallel paths of a composite compete for the same set of providers (b).*

6.2.4.2 Communication effort

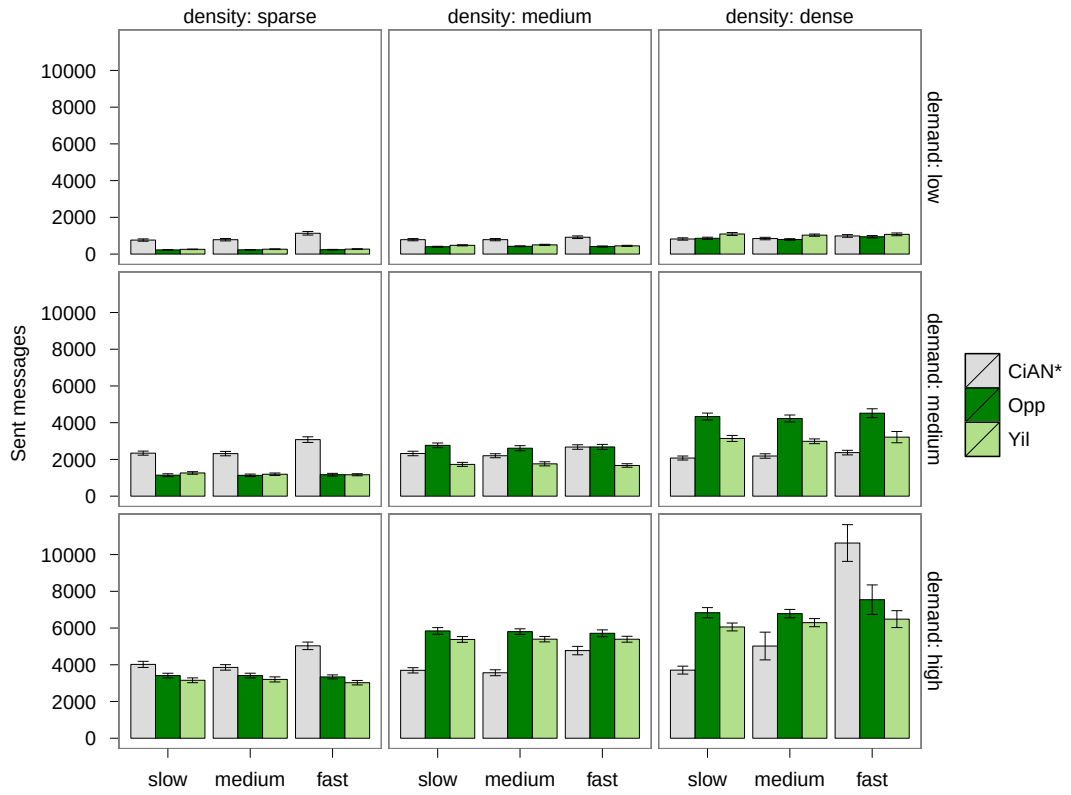
The communication effort illustrated in Figure 6.13a shows that in most cases, the proposed protocol's communication effort is similar or higher than that of CiAN* or Yil. However, the protocol's lower failure ratio means it completes more composites successfully, which justifies this overhead. With increasing service demand and network density, its communication effort increases substantially. Figure 6.13b shows for high service demand that the proposed protocol issues a high number of composite requests to find free service providers. CiAN*, on the other hand, spends most of its messages to establish routes between consecutive service providers. Figure 6.13b omits Yil as it similarly to the proposed protocol encounters service discovery issues.

6.2.4.3 Response time

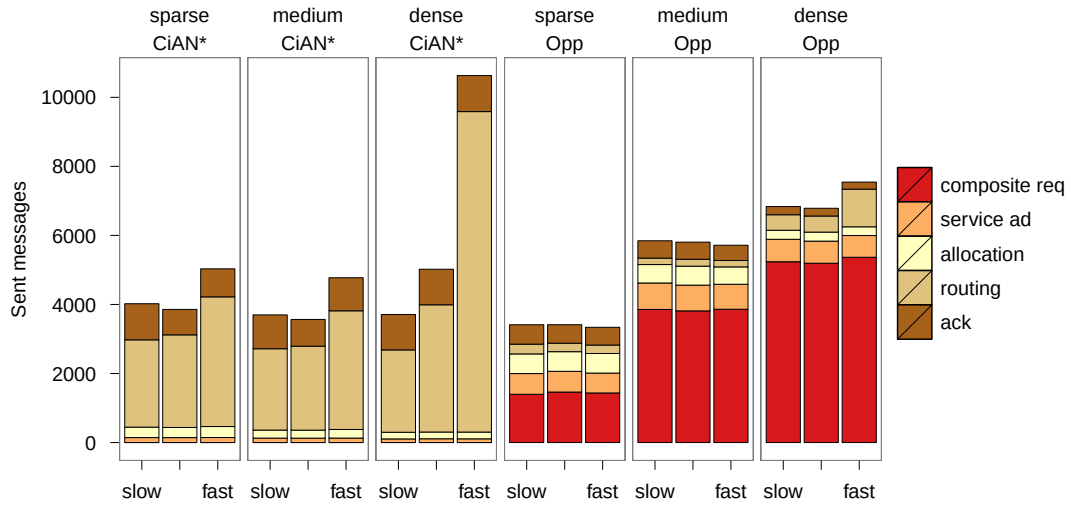
The response time illustrated in Figure 6.14 shows that the proposed protocol responds on average 4 seconds faster than CiAN* if the network is sparse (left column), service demand is low (top row), and in medium-dense medium demand networks (centre graph). Compared to scenario 3 (cp. Figure 6.11) the response time for these cases increases by a maximum of 2.5 seconds due to the need for synchronisation and the increased challenge of finding free service providers. The difference to Yil is negligible.

6.2.4.4 Summary

Generally, the proposed protocol reduces the composition failure for a moderate service flow in varying operating environments. Except for high-demand dense networks, it fails less often than CiAN* and reaches an improvement of up to 59 percent points. Using service layer routing shows the greatest improvement of 53 percent points over Yil with continuous path updates, for medium service demand and network density. In some cases, the proposed protocol achieves a lower failure ratio with fewer messages overhead. The protocol responds more quickly than CiAN* if the network is sparse or service demand low. Compared to service sequences, the impact of service demand and network density has increased for all composition protocols because the parallel paths of the service flow operate in the same area and compete for the same set of providers



(a) Communication effort



(b) Communication effort for high service demand

Fig. 6.13: ***Communication effort in scenario 4*** *For a service flow and varying operating environment, the opportunistic protocol sends in most cases similar or more messages than CiAN* and Yil (a). With increasing service demand and network density the proposed protocol requires a high number of composite requests to find free service providers (b).*

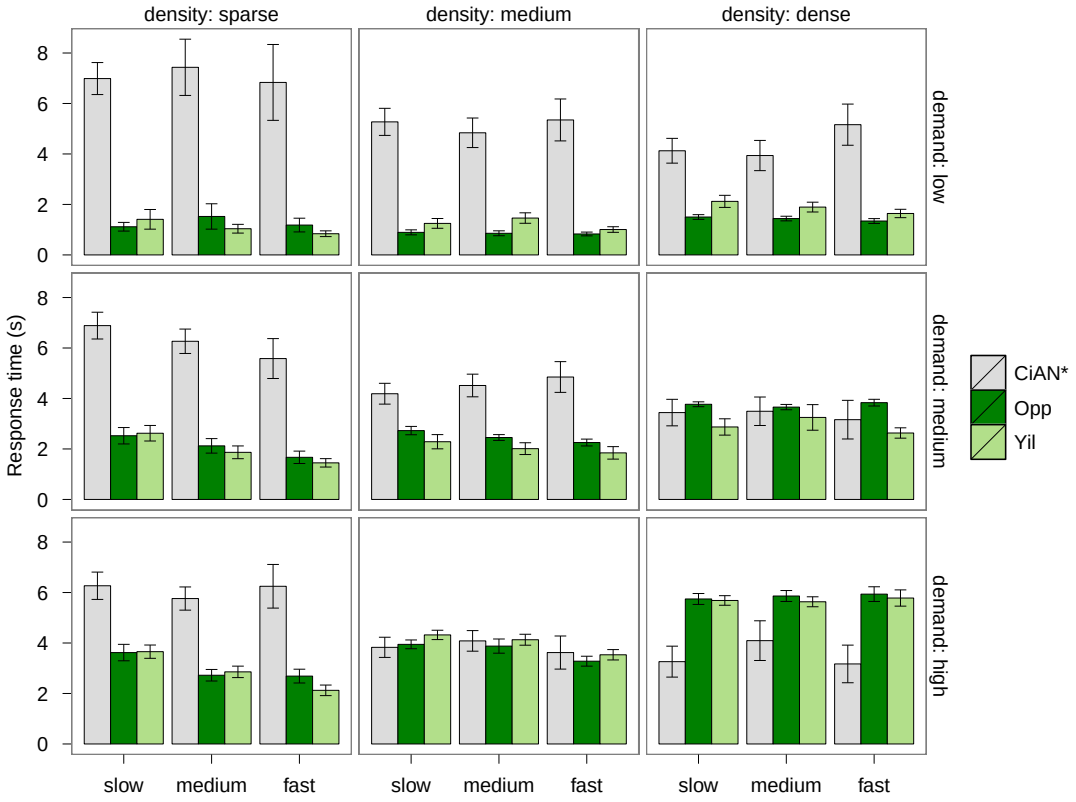


Fig. 6.14: ***Response time in scenario 4*** *For a service flow and varying operating environment, the opportunistic protocol responds on average 4 seconds faster than CiAN* in sparse networks, if service demand is low, and if demand and density is at a medium level. The difference to Yil is negligible.*

Table 6.4: ***Result summary*** *For the tested scenarios and compared to the baselines CiAN* and Yil, the performance of the proposed protocol is similar ○, sometimes better sometimes worse ◑, better or similar never worse ◕, always better ● as follows:*

	CiAN*			Yil		
	Failure ratio	Communication	Response time	Failure ratio	Communication	Response time
Impact of service sequence length	●	●	●	inapplicable		
Impact of service flow structure	◑	●	●	●	●	○
Impact of environment using a service sequence	●	◑	◑	inapplicable		
Impact of environment using a service flow	◕	◑	◑	◕	◑	○

which leads to more discovery or overload failure. The impact of the participant speed on the failure ratio has also slightly increased because with multiple composite controllers in service flows, more routes are created that are likely to break if participants move fast. Altogether, the proposed protocol maintains its benefit over the baselines for the moderate flow type 2 under various operating conditions. It is intuitive that a lower failure ratio may come at the expense of higher communication effort and longer delays.

6.3 Chapter summary

This chapter presented a simulation-based study to evaluate how well the novel opportunistic composition protocol achieves its objective to reduce the failure probability of service composites in dynamic ad hoc environments. The study uses four evaluation scenarios and two baselines to assess the protocol's performance with regard to its failure ratio, communication effort, and response time. Table 6.4 gives an overview of the results that can be summarised as follows:

- For service sequences, the proposed protocol lowers the failure ratio by 20 to 70 percent points compared to CiAN* and depending on the particular setting of the operating environment. There are two reasons for this: First, the length of a service

sequence has less impact on the protocol's performance than on CiAN* as it is less prone to stale routing information. Second, the protocol's observable demand-based interaction reduces the communication effort and composition delays.

- For different service flows types, the protocol's performance is more diverse. Compared to CiAN*, the proposed protocol maintains a lower failure ratio for composites with moderate parallelism. If parallelism is high, synchronisation for an allocation decision delays the composition process and increases the probability of network failure. With regard to the synchronisation algorithm, the proposed service layer routing algorithm lowers the failure ratio across all tested flow types. Its improvement ranges from 3 and 19 percent points compared to Yil, which uses continuous path updates. The proposed algorithm synchronises once at the end of parallel paths and exposes the composite less often to the unreliable network.

- For a moderate service flow in different operating environments, the proposed protocol improves the failure ratio between 9 and 59 percent points compared to CiAN*. When service demand is high and the network dense, both protocols fail similarly often. Compared to Yil, the proposed synchronisation algorithm fails similarly often in some cases and in other cases achieves an improvement of a failure ratio of up to 53 percent points. In particular, for medium service demand the proposed algorithm can leverage the benefit of less communication between parallel paths before it suffers similarly to Yil from service discovery issues when service demand is high.

- With regard to the operating environment, service demand and network density dominate the impact on the protocol's performance. If both increase, multiple composites interfere with each other and compete for the same set of providers, which gradually leads to more communication, longer delays, and eventually more failure. The participant speed affects the failure ratio because the faster the service providers move, the more likely is routing information to grow stale.

The results show that the proposed protocol generally achieves its objective and reduces the failure probability of composites in dynamic ad hoc environments. The protocol suits

service sequences well because it reduces the composition delay. The improvements for service flows are smaller because synchronisation further delays the composition process and increases the probability for network failure. This supports the hypothesis that shorter delays imply lower composite failure.

Chapter 7

Conclusion

This chapter summarises the thesis and its achievements. It discusses the trade-offs of the designed protocol and highlights potential areas for future work.

7.1 Thesis summary

This thesis described a novel model for service composition that addresses a number of issues that occur if components in a complex application are distributed in dynamic ad hoc environments.

Introduction Chapter 1 motivated this work and argued that pervasive computing environments rely on the composition of services from several mobile devices to leverage the opportunities of mobile sensing in the immediate surroundings of a user. However, service composition in such environments faces transient networks, the lack of composition infrastructure, unreliable communication, participation autonomy, and mobility. These dynamic operating conditions together with the delay introduced by the composition process itself lead to potential failure of service composites. The chapter hypothesised that a novel composition protocol could lower the failure probability of composites by reorganising the composition process and thus reducing the delay between the composition phases in a communication efficient manner. The hypothesis was that: The later the operating environment is explored to compose a complex service request, the less time there is for the system to change and to render composition decisions invalid.

State of the art Chapter 2 reviewed the service-oriented computing domain in more detail. It examined how different approaches establish and maintain their service topology, stabilise the composite against changes, block and release resources, and apply these principles to parallel service flows. The analysis highlighted the gap for an opportunistic composition protocol as existing solutions do not study the possibility of integrating the composition phases to reduce composite failure.

Design Chapter 3 first derived a set of design objectives and argued that the novel composition protocol must proactively gather information, self-organise service composition, and support parallel service flows. In addition, the protocol must reduce communication, enable short and localised interaction, and obtain the providers' commitment. Thereafter, the system model framed service composition in dynamic ad hoc environments as a mapping problem between the composite graph and the sequence of time-varying graphs of provided services. In addition, the system model formally defined the scope of this thesis. The discussion of design alternatives then led to the design decisions that constitute the main contribution of this thesis, the proposed opportunistic composition protocol. The protocol relies on demand-based and resource-blocking service announcements to exchange only provider data that is relevant for a composite and to confirm the commitment of a service provider. The protocol decentralises and interleaves the composition phases such that a service provider first executes its assigned service and then allocates its successor. This allows for short localised interactions between consecutive service providers and the opportunity to reduce a composite during execution, i.e., by removing parts of the composite that have become obsolete. The concept applies to all required services, including those that synchronise parallel execution paths. For the end-to-end communication between composite participants the protocol employs observable multicasting to allow for proactive yet demand-based service announcements and an efficient way to release redundant resources. Collectively, these design decisions support self-organisation of service composition in an ad hoc network of mobile devices.

Design verification Chapter 4 applied model checking as a formal method to verify the correctness of protocol properties that target the interaction of multiple composite

participants. For service sequences and parallel service flows, the verification results confirmed that the proposed composition protocol does not deadlock and terminates in a valid end state after having allocated the correct number of service providers for all required sub-services.

Implementation Chapter 5 presented details on the prototype implementation that allows for simulating mobile composite participants and their collective effort to achieve a complex task. The prototype serves a basis for evaluating the protocol's performance.

Evaluation Chapter 6 evaluated how well the proposed protocol achieves the overall objective of reducing the failure probability of service composites in dynamic ad hoc networks. It simulated various scenarios to expose the protocol to different composite structures, levels of service demand, network densities, and speeds of composite participants. The results showed that the protocol presents a suitable composition model for dynamic ad hoc environments because it generally fails less than the baselines CiAN* and Yil. For sequential composites the protocol reduces the failure ratio by 20 to 70 percent points, compared to the baselines and depending on the level of the service demand and network density. For service flows with few and long parallel execution paths, the improvement of the failure ratio ranges between 9 and 59 percent points, depending on the different operating conditions. However, the results also showed that increasing parallelism introduces synchronisation delays and increases the probability of network failure. The protocol performs best when multiple composite requests are least likely to interfere with each other, which is when service demand or network density stay at a low or medium level.

7.2 Discussion

The proposed protocol generally reduces the failure probability of service composites in dynamic ad hoc environments, which suggests that the novel opportunistic composition model is a suitable alternative to existing techniques. However, an important finding of this thesis is that an optimisation in one part of the composition process often implied a cutback in another part, as the following design details show:

- Service announcements are designed to block provider resources immediately to avoid additional confirmation about the provider's commitment and to prevent failure due to provider overload. However, these resource-blocking announcements block more providers than needed and require extra communication to release redundant providers. In addition, service discovery failure may increase due to temporary provider shortage: A composite may fail to allocate a free service provider because another composite has blocked all relevant providers and not yet released them. A way to address this issue, is to introduce cooperation among composites such that they release redundant providers as soon as they become aware of co-located composites and their service discovery requests. The observability of the composition process raises this awareness without additional communication cost.

- Service announcements are issued if there is demand for that service. This reduces the use of bandwidth for provider data that is immediately applicable. The simulation study showed that this also creates network links between consecutive providers without additional route discovery. However, this benefit does not apply to synchronisation partners and the protocol relies on route discovery nonetheless. The cross-layered communication approach may provide for a way to increase the efficiency of route discovery. With observable composition processes, route requests may be sent more directed towards the synchronisation partner, rather than flooded concentrically from the message source into the network.

- Interleaving the composition phases localises the provider allocation and reduces the composition delay such that the protocol improves the failure probability of service sequences. Service flows, however, require the synchronisation of allocation decisions which introduces delays and makes locality an issue: Multiple synchronisation partners try to address the same successor at about the same time which increases the demand for bandwidth and network failure. The wireless communication standard IEEE 802.11 supports RTS/CTS (Request to Send / Clear to Send), a mechanism to avoid signal collision and typically used for unicast messages that exceed a certain packet size. Applying RTS/CTS to directed broadcasting may improve the protocol's performance for parallel service flows.

- Decentralised composition allows for direct and localised interaction between consecutive service providers. This, however, implies that the composite initiator loses control over the composite until it receives the final result or a failure notification. It cannot intervene in intermediate service results. The protocol eases this limitation by supporting composites with conditional execution paths. At runtime, the protocol reduces the composite and allocates resources to paths that are still valid. Alternatively, the composite may specify checkpoints which require service providers to transfer intermediate results and the composition control back to the initiator who then can adapt the composite to its needs.

The protocol builds on two underlying assumptions: First, users are generally willing to share the capabilities of their mobile devices with their immediate surroundings. Second, the disclosure of composite-related data is not an issue. The concept of exchanging messages as directed broadcasts benefits from these assumptions and allows for observing composition traffic, tracking the evolution of the service and network topology, and initiating proactive yet demand-based actions at lower communication effort. For applications like mobile sensing, where physical phenomena can be experienced by anyone in the same area, consuming and providing services does not disclose sensitive data. However, if applications require a higher level of privacy the proposed protocol would require additional means to protect the content of messages.

7.3 Future work

This thesis examined a new research direction for how service composition can be organised to adapt to dynamic operating environments. Notwithstanding its contribution to knowledge, the thesis may serve as a starting point for further investigations in areas such as failure recovery and people-centric sensing.

Failure recovery Failure recovery is an important challenge for service composition in transient networks but while the proposed protocol reduces composition failure, it cannot prevent it. Recovery strategies may be most effective if they start in lower layers (Jiang et al., 2009) e.g., by avoiding signal collision or repairing broken network links. However,

composition-related failure sources require suitable means of recovery at the service composition layer. For example, the simulation study showed that dense networks with high demand for services are a challenge for any of the tested composition protocols due to the temporary shortage of unblocked service providers. A possible solution may build on the observability of composition messages and introduce cooperation among composite controllers such that they release redundant providers once other composite controllers signal service discovery issues. The challenge of this undertaking is to strike the right balance between cooperation and communication effort as each additional message is a potential source for network failure.

People-centric sensing Complex sensing tasks can be initiated by mobile users as well as distant cloud services that require sensor data to provide higher level context information. People-centric sensing, for example, relies on citizens to collect and upload sensor data to improve the micro- and macroscopic view of a city (Lane et al., 2010). However, increasing the number of contributors and the demand for context information strains the device-to-cloud connections and challenges the system's scalability. Opportunistic service composition may be a way to balance the load for data processing. Co-located mobile devices could collaborate to aggregate their sensor readings and upload one big data packet from one device instead of transmitting multiple small data packets via individual connections. Future work will have to investigate how the composition effort in an ad hoc network compares to the overhead of uploading sensor data individually. Each device may experience delays, losses, and energy costs when communicating either with the distant cloud or consecutive service providers.

7.4 Final remark

This thesis investigated how to reduce the failure probability of service composites in dynamic ad hoc environments. Based on the observation that the composition delay is a major failure source, a novel opportunistic composition protocol was designed to allow for short, localised, and demand-based interactions between mobile and initially unknown service providers. As the first of its kind, the protocol examined the impact of

reorganising the composition phases and making the composition process observable to potential service providers. The thesis showed that the proposed protocol is a suitable alternative to conventional composition solutions as it reduces the composite's exposure to its unreliable operating environment and generally achieves a lower failure probability. The protocol is designed for applications that run in highly dynamic environments, require multiple participants, and allocate components in an ad hoc manner. These characteristics apply to data collection and in-network processing tasks for which pervasive computing and the collection of context information is a prominent example. Context-aware applications require insight into the current situation of their users to assist them with relevant information and targeted services. In contrast, the protocol does not suit applications that run in stable environments in which task allocation can be planned and verified. For example, business process and e-commerce applications require transactional behaviour which the protocol does not provide. It is hoped that the findings of this thesis offer a new perspective on the use of ad hoc networks and encourage further research on composition-based collaboration in emerging areas such as people-centric sensing.

Appendix A

Verification with SPIN

The following screenshots illustrate the output of the model checker SPIN (version 6.2.2) for the verification of the proposed opportunistic service composition protocol. Figure A.1 demonstrates the output of SPIN if the assertion of a correctness property fails. Figure A.2 shows the output for the sequential PROMELA model and that it terminates without any errors. Figure A.3 shows the correctness of the parallel PROMELA model.

```
1  spin -a  opportunistic-seq-safety-fail.pml
2  gcc-3 -DMEMLIM=1024 -O2 -DXUSAFE -DSAFETY -DNOCLAIM -DNOREDUCE -w -o pan pan.c
3  ./pan -m10000  -n -c1
4  Pid: 45832
5  pan:1: assertion violated ((((controllerPerService[0]==1)&&(controllerPerService[1]==1))&&(controller
   PerService[2]==1))&&(controllerPerService[3]==1)) (at depth 316)
6  pan: wrote opportunistic-seq-safety-fail.pml.trail

7  (Spin Version 6.2.2 -- 6 June 2012)
8  Warning: Search not completed

9  Full statespace search for:
10             never claim         - (not selected)
11             assertion violations +
12             cycle checks        - (disabled by -DSAFETY)
13             invalid end states +

14 State-vector 132 byte, depth reached 316, errors: 1
15     760 states, stored
16     571 states, matched
17    1331 transitions (= stored+matched)
18       3 atomic steps
19 hash conflicts:         0 (resolved)

20 Stats on memory usage (in Megabytes):
21    0.101   equivalent memory usage for states (stored*(State-vector + overhead))
22    0.266   actual memory usage for states
23   64.000   memory used for hash table (-w24)
24    0.305   memory used for DFS stack (-m10000)
25   64.501   total actual memory usage




26 pan: elapsed time 0.01 seconds
27 To replay the error-trail, goto Simulate/Replay and select "Run"
```

Fig. A.1: ***SPIN output for failed verification*** *The model checker runs with a modified sequential PROMELA model (Line 1) in which a controller assigns multiple providers to a required service. This leads to a violation of the correctness property (Line 5). The model checker stops and does not continue searching the finite state machine (Line 8). It found a counter example after 316 steps that produced an error (Line 14).*

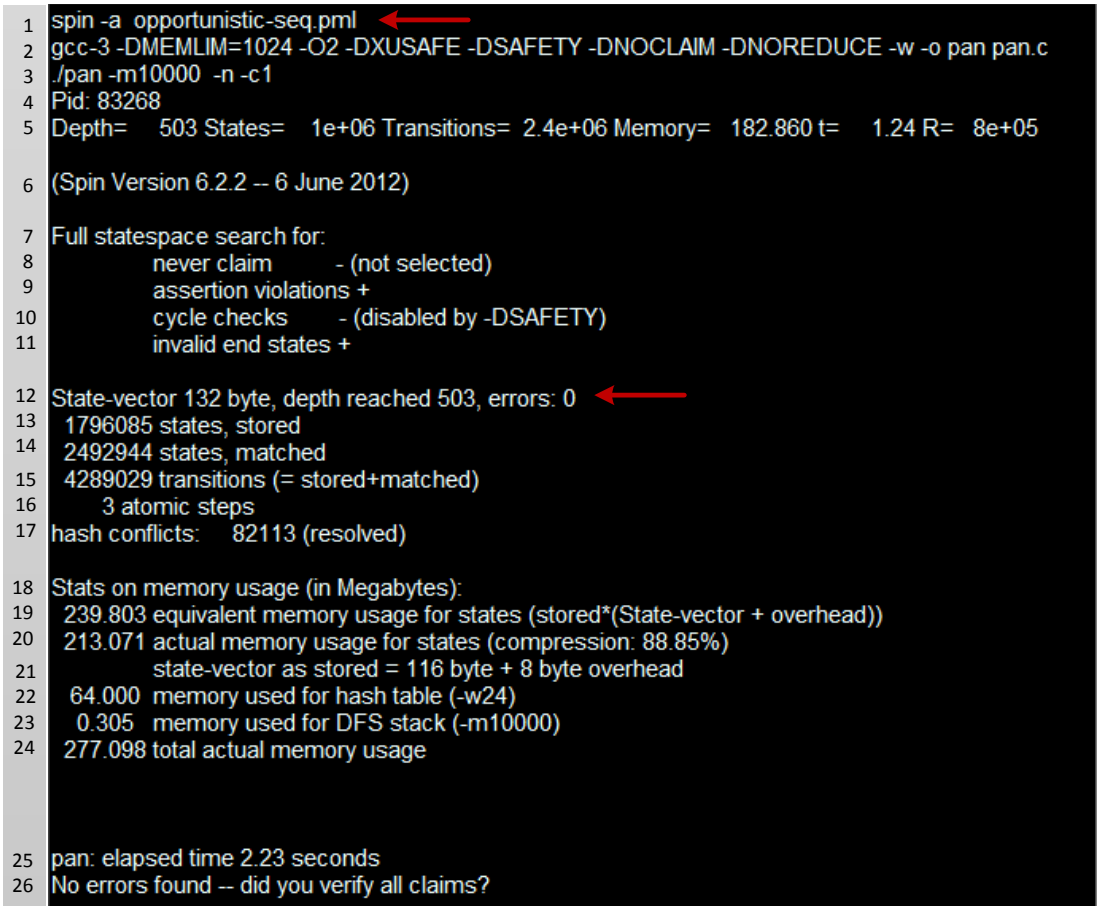


Fig. A.2: ***SPIN output for sequential PROMELA model*** *The model checker runs the sequential PROMELA model for the proposed opportunistic composition protocol (Line 1) and does not produce any errors (Line 12). This means no assertion violations (Line 9) or invalid end states (Line 11) have occurred and the protocol is correct for its defined correctness properties.*

```
1  spin -a  opportunistic-par.pml
2  gcc-3 -DMEMLIM=1024 -O2 -DXUSAFE -DSAFETY -DNOCLAIM -DNOREDUCE -w -o pan pan.c
3  ./pan -m10000  -n -c1
4  Pid: 40444
5  Depth=     463 States=    1e+06 Transitions= 2.57e+06 Memory=   316.844 t=    2.09 R=   5e+05
6  Depth=     463 States=    2e+06 Transitions= 5.09e+06 Memory=   569.090 t=    4.19 R=   5e+05

7  (Spin Version 6.2.2 -- 6 June 2012)

8  Full statespace search for:
9              never claim         - (not selected)
10             assertion violations +
11             cycle checks        - (disabled by -DSAFETY)
12             invalid end states +

13 State-vector 272 byte, depth reached 463, errors: 0
14   2911037 states, stored
15   4464699 states, matched
16   7375736 transitions (= stored+matched)
17        4 atomic steps
18 hash conflicts:    220853 (resolved)

19 Stats on memory usage (in Megabytes):
20   777.331 equivalent memory usage for states (stored*(State-vector + overhead))
21   736.445 actual memory usage for states (compression: 94.74%)
22             state-vector as stored = 257 byte + 8 byte overhead
23    64.000  memory used for hash table (-w24)
24     0.305  memory used for DFS stack (-m10000)
25     1.688  memory lost to fragmentation
26   799.071 total actual memory usage




27 pan: elapsed time 6.13 seconds
28 No errors found -- did you verify all claims?
```

Fig. A.3: ***SPIN output for parallel PROMELA model*** *The model checker runs the parallel PROMELA model for the proposed opportunistic composition protocol (Line 1) and does not produce any errors (Line 13). This means no assertion violations (Line 10) or invalid end states (Line 12) have occurred and the protocol is correct for its defined correctness properties.*

Bibliography

Aguilera, U. and López-de Ipiña, D. (2012). Service composition for mobile ad hoc networks using distributed matching, *Ubiquitous Computing and Ambient Intelligence*, Vol. 7656 of *LNCS*, Springer, pp. 290–297.

Artail, H., Antoun, R. and Fawaz, K. (2009). CRUST: Implementation of clustering and routing functions for mobile ad hoc networks using reactive tuple-spaces, *Ad Hoc Networks* **7**(6): 1064 – 1081.

Atluri, V., Chun, S. A., Mukkamala, R. and Mazzoleni, P. (2007). A decentralized execution model for inter-organizational workflows, *Distributed and Parallel Databases* **22**(1): 55–83.

Balasooriya, J., Prasad, S. and Navathe, S. (2008). A middleware architecture for enhancing web services infrastructure for distributed coordination of workflows, *International Conference on Services Computing (SCC)*, Vol. 1, IEEE, pp. 370 –377.

Barr, R. and Kargl, F. (2005). JiST/SWANS Edition of Ulm University.
URL: *http://vanet.info/jist-swans/download.html*

Bidot, J., Goumopoulos, C. and Calemis, I. (2011). Using ai planning and late binding for managing service workflows in intelligent environments, *International Conference on Pervasive Computing and Communications (PerCom)*, pp. 156 –163.

Brønsted, J., Hansen, K. and Ingstrup, M. (2010). Service composition issues in pervasive computing, *Pervasive Computing* **9**(1): 62–70.

Bucchiarone, A., Marconi, A., Pistore, M. and Raik, H. (2012). Dynamic Adaptation of

Fragment-Based and Context-Aware Business Processes, *International Conference on Web Services (ICWS)*, IEEE, pp. 33–41.

Campbell, A. T., Eisenman, S. B., Lane, N. D., Miluzzo, E., Peterson, R. a., Lu, H., Zheng, X., Musolesi, M., Fodor, K. and Ahn, G.-S. (2008). The Rise of People-Centric Sensing, *IEEE Internet Computing* **12**(4): 12–21.

Cardellini, V., Casalicchio, E., Grassi, V., Lo Presti, F. and Mirandola, R. (2009). QoS-driven runtime adaptation of service oriented architectures, *European Software Engineering Conference and The Foundations of Software Engineering (ESEC/FSE)*, ACM, pp. 131–140.

Casteigts, A., Flocchini, P., Quattrociocchi, W. and Santoro, N. (2011). Time-varying graphs and dynamic networks, *International conference on Ad-hoc, mobile, and wireless networks (ADHOC-NOW)*, Springer-Verlag, pp. 346–359.

Catarci, T., de Leoni, M., Marrella, A., Mecella, M., Salvatore, B., Vetere, G., Dustdar, S., Juszczyk, L., Manzoor, A. and Truong, H.-L. (2008). Pervasive software environments for supporting disaster responses, *Internet Computing, IEEE* **12**(1): 26 –37.

Chakraborty, D., Joshi, A., Finin, T. and Yesha, Y. (2005). Service composition for mobile environments, *Mobile Networks and Applications* **10**: 435–451.

Chakraborty, D., Joshi, A., Yesha, Y. and Finin, T. (2006). Toward distributed service discovery in pervasive computing environments, *IEEE Transactions on Mobile Computing* **5**(2): 97 – 112.

Chakraborty, D., Yesha, Y. and Joshi, A. (2004). A distributed service composition protocol for pervasive environments, *Wireless Communications and Networking Conference (WCNC)*, Vol. 4, IEEE, pp. 2575–2580.

Chlamtac, I., Conti, M. and Liu, J. J.-N. (2003). Mobile ad hoc networking: imperatives and challenges, *Ad Hoc Networks* **1**(1): 13 – 64.

Clarke, E. M., Grumberg, O. and Long, D. E. (1994). Model checking and abstraction, *ACM Transactions on Programming Languages and Systems* **16**(5): 1512–1542.

El Falou, M., Bouzid, M., Mouaddib, A.-I. and Vidal, T. (2010). A distributed planning approach for web services composition, *International Conference on Web Services (ICWS)*, IEEE, pp. 337 –344.

Fdhila, W. and Godart, C. (2009). Toward synchronization between decentralized orchestrations of composite web services, *International Conference on Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom)*, IEEE, pp. 1 –10.

Fdhila, W., Yildiz, U. and Godart, C. (2009). A flexible approach for automatic process decentralization using dependency tables, *International Conference on Web Services (ICWS)*, IEEE, pp. 847 –855.

Feng, X., Wang, H., Wu, Q. and Zhou, B. (2007). An adaptive algorithm for failure recovery during dynamic service composition, *Pattern Recognition and Machine Intelligence*, Vol. 4815 of *Lecture Notes in Computer Science*, Springer, pp. 41–48.

Fernández, H., Priol, T. and Tedeschi, C. (2010). Decentralized approach for execution of composite web services using the chemical paradigm, *International Conference on Web Services (ICWS)*, IEEE, pp. 139 –146.

Fernández, H., Tedeschi, C. and Priol, T. (2012). Decentralized workflow coordination through molecular composition, *Service-Oriented Computing - ICSOC 2011 Workshops*, Vol. 7221 of *LNCS*, Springer Berlin Heidelberg, pp. 22–32.

Friedman, R., Gavidia, D., Rodrigues, L., Viana, A. C. and Voulgaris, S. (2007). Gossiping on MANETs: The beauty and the beast, *ACM SIGOPS Operating Systems Review* **41**(5): 67–74.

Gu, X. and Nahrstedt, K. (2006). Distributed multimedia service composition with statistical qos assurances, *IEEE Transactions on Multimedia* **8**(1): 141–151.

Guinard, D. (2010). Mashing up your web-enabled home, *in* F. Daniel and F. Facca (eds), *Current Trends in Web Engineering*, Vol. 6385 of *LNCS*, Springer Berlin / Heidelberg, pp. 442–446.

Higashi, S., Ata, S., Nakao, A. and Oka, I. (2011). Server selection for equalizing of performance in distributed cooperative system, *International Conference on Information Networking (ICOIN)*, IEEE, pp. 519 –524.

Holzmann, G. J. (2003). *The Spin Model Checker, Primer and Reference Manual*, Addison-Wesley.

Jiang, S., Xue, Y. and Schmidt, D. C. (2009). Minimum disruption service composition and recovery in mobile ad hoc networks, *Computer Networks* **53**: 1649–1665.

Kalasapur, S., Kumar, M. and Shirazi, B. (2007). Dynamic service composition in pervasive computing, *Transactions on Parallel and Distributed Systems* **18**(7): 907–918.

Kiess, W. and Mauve, M. (2007). A survey on real-world implementations of mobile ad-hoc networks, *Ad Hoc Networks* **5**(3): 324–339.

Kurkowski, S., Camp, T. and Colagrosso, M. (2005). MANET simulation studies: the incredibles, *Mobile Computing and Communications Review* **9**(4): 50–61.

Kurkowski, S., Camp, T. and Navidi, W. (2006). Two standards for rigorous manet routing protocol evaluation, *International Conference on Mobile Adhoc and Sensor Systems (MASS)*, IEEE, pp. 256–266.

Kurkowski, S., Navidi, W. and Camp, T. (2007). Constructing manet simulation scenarios that meet standards, *IEEE International Conference on Mobile Adhoc and Sensor Systems Conference* **0**: 1–9.

Lane, N., Miluzzo, E., Lu, H., Peebles, D., Choudhury, T. and Campbell, A. (2010). A survey of mobile phone sensing, *Communications Magazine, IEEE* **48**(9): 140 –150.

Li, Z. and Shen, H. (2012). Game-theoretic analysis of cooperation incentive strategies in mobile ad hoc networks, *IEEE Transactions on Mobile Computing* **11**(8): 1287 –1303.

Loke, S. W. (2012). Supporting ubiquitous sensor-cloudlets and context-cloudlets: Programming compositions of context-aware systems for mobile users, *Future Generation Computer Systems* **28**(4): 619–632.

Madhusudan, T. and Uttamsingh, N. (2006). A declarative approach to composing web services in dynamic environments, *Decis. Support Syst.* **41**: 325–357.

Manolescu, D. A. (2002). Workflow enactment with continuation and future objects, *ACM SIGPLAN conference on object-oriented programming, systems, languages, and applications (OOPSLA)*, ACM, pp. 40–51.

Martin, D., Wutke, D. and Leymann, F. (2008). A novel approach to decentralized workflow enactment, *International Enterprise Distributed Object Computing Conference (EDOC)*, IEEE, pp. 127–136.

Mian, A., Baldoni, R. and Beraldi, R. (2009). A survey of service discovery protocols in multihop mobile ad hoc networks, *IEEE Pervasive Computing* **8**(1): 66 –74.

Miluzzo, E., Cornelius, C. T., Ramaswamy, A., Choudhury, T., Liu, Z. and Campbell, A. T. (2010). Darwin phones: The evolution of sensing and inference on mobile phones, *International Conference on Mobile systems, applications, and services (MobiSys)*, ACM, pp. 5–20.

Mostarda, L., Marinovic, S. and Dulay, N. (2010). Distributed orchestration of pervasive services, *International Conference on Advanced Information Networking and Applications (AINA)*, IEEE, pp. 166 –173.

Murphy, A. L., Box, P. O., Picco, G. P., Milano, P., Leonard, P. and Roman, G.-c. (2001). LIME : A Middleware for Physical and Logical Mobility, *International Conference on Distributed Computing Systems*, IEEE, pp. 524–533.

Navidi, W. and Camp, T. (2004). Stationary distributions for the random waypoint mobility model, *IEEE Transactions on Mobile Computing* **3**(1): 99 – 108.

Nedos, A., Singh, K., Cunningham, R. and Clarke, S. (2009). Probabilistic discovery of semantically diverse content in manets, *IEEE Transactions on Mobile Computing* **8**(4): 544 –557.

Papazoglou, M. P. and Heuvel, W.-J. (2007). Service oriented architectures: Approaches, technologies and research issues, *The VLDB Journal* **16**(3): 389–415.

Park, E. and Shin, H. (2008). Reconfigurable service composition and categorization for power-aware mobile computing, *Transactions on Parallel and Distributed Systems* **19**(11): 1553–1564.

Park, S.-H., Lee, T.-G., Seo, H.-S., Kwon, S.-J. and Han, J.-H. (2009). An election protocol in mobile ad hoc distributed systems, *International Conference on Information Technology: New Generations (ITNG)*, IEEE, pp. 628 –633.

Perkins, C. E., Belding-Royer, E. M. and Das, S. (2003). Ad hoc on-demand distance vector (aodv) routing.
URL: *http://www.ietf.org/rfc/rfc3561.txt*

Philips, E., Van Der Straeten, R. and Jonckers, V. (2010). NOW: A workflow language for orchestration in nomadic networks, *in* D. Clarke and G. Agha (eds), *Coordination Models and Languages*, Vol. 6116 of *LNCS*, Springer, pp. 31–45.

Pinto, L. S., Cugola, G. and Ghezzi, C. (2012). Dealing with changes in service orchestrations, *Symposium on Applied Computing (SAC)*, ACM, pp. 1961–1967.

Pintus, A., Carboni, D., Piras, A. and Giordano, A. (2010). Connecting smart things through web services orchestrations, *in* F. Daniel and F. Facca (eds), *Current Trends in Web Engineering*, Vol. 6385 of *LNCS*, Springer Berlin / Heidelberg, pp. 431–441.

Polyvyanyy, A., Garca-Bauelos, L. and Dumas, M. (2010). Structuring acyclic process models, *in* R. Hull, J. Mendling and S. Tai (eds), *Business Process Management (BPM)*, Vol. 6336 of *Lecture Notes in Computer Science*, Springer Berlin / Heidelberg, pp. 276–293.

Prinz, V., Fuchs, F., Ruppel, P., Gerdes, C. and Southall, A. (2008). Adaptive and fault-tolerant service composition in peer-to-peer systems, *International Conference on Distributed Applications and Interoperable Systems (DAIS)*, Springer, pp. 30–43.

Roy, S., Herlugson, K. and Saberi, A. (2006). A control-theoretic approach to distributed discrete-valued decision-making in networks of sensing agents, *IEEE Transactions on Mobile Computing* **5**(8): 945–957.

Russell, N., Arthur, van der Aalst, W. M. P. and Mulyar, N. (2006). Workflow control-flow patterns: A revised view, *Technical Report BPM-06-22*, BPM Center.

Russell, N., ter Hofstede, A., Edmond, D. and van der Aalst, W. (2005). Workflow data patterns: Identification, representation and tool support, *in* L. Delcambre, C. Kop, H. Mayr, J. Mylopoulos and O. Pastor (eds), *Conceptual Modeling ER 2005*, Vol. 3716 of *Lecture Notes in Computer Science*, Springer Berlin / Heidelberg, pp. 353–368.

Samimi, F. A. and McKinley, P. K. (2008). Dynamis: Dynamic overlay service composition for distributed stream processing, *International Conference on Software Engineering and Knowledge Engineering (SEKE)*, pp. 881–886.

Santoro, N., Quattrociocchi, W., Flocchini, P., Casteigts, A. and Amblard, F. (2011). Time-varying graphs and social network analysis: Temporal indicators and metrics, *Convention on Artificial Intelligence and Simulation of Behaviour (AISB)*, pp. 33–38.

Schuhmann, S., Herrmann, K. and Rothermel, K. (2013). Adaptive composition of distributed pervasive applications in heterogeneous environments, *ACM Transactions on Autonomous and Adaptive Systems* . (to appear).

Schuler, C., Weber, R., Schuldt, H. and Schek, H.-J. (2004). Scalable peer-to-peer process management - The OSIRIS approach, *International Conference on Web Services (ICWS)*, IEEE, pp. 26 – 34.

Sen, R., Handorean, R., Roman, G.-C. and Hackmann, G. (2004). Knowledge-driven interactions with services across ad hoc networks, *International conference on Service oriented computing (ICSOC)*, ACM Press, p. 222.

Sen, R., Roman, G.-C. and Gill, C. (2007). Distributed allocation of workflow tasks in manets, *Technical Report 2007-41*, Department of Computer Science & Engineering - Washington University in St. Louis.

Sen, R., Roman, G.-C. and Gill, C. (2008). CiAN: A workflow engine for manets, *in* D. Lea and G. Zavattaro (eds), *Coordination Models and Languages*, Vol. 5052 of *LNCS*, Springer, pp. 280–295.

Singh, A. and Sharma, S. (2011). Message efficient leader finding algorithm for mobile ad hoc networks, *Conference on Communication Systems and Networks (COMSNETS)*, IEEE, pp. 1–6.

Thomas, L., Wilson, J., Roman, G.-C. and Gill, C. (2009). Achieving coordination through dynamic construction of open workflows, *in* J. Bacon and B. Cooper (eds), *Middleware 2009*, Vol. 5896 of *Lecture Notes in Computer Science*, Springer Berlin / Heidelberg, pp. 268–287.

Varshavsky, A., Reid, B. and de Lara, E. (2005). A cross-layer approach to service discovery and selection in manets, *IEEE International Conference on Mobile Adhoc and Sensor Systems Conference*, IEEE, pp. 459–466.

Ververidis, C. and Polyzos, G. (2008). Service discovery for mobile ad hoc networks: A survey of issues and techniques, *Communications Surveys Tutorials, IEEE* **10**(3): 30 –45.

Wang, M., Li, B. and Li, Z. (2004). sFlow: Towards resource-efficient and agile service federation in service overlay networks, *International Conference on Distributed Computing Systems (ICDCS)*, IEEE, pp. 628–635.

Wang, X., Wang, J., Zheng, Z., Xu, Y. and Yang, M. (2009). Service composition in service-oriented wireless sensor networks with persistent queries, *Consumer Communications and Networking Conference (CCNC)*, IEEE, pp. 1–5.

Wang, Z., Xu, T., Qian, Z. and Lu, S. (2009). A parameter-based scheme for service composition in pervasive computing environment, *International Conference on Complex, Intelligent and Software Intensive Systems (CISIS)*, IEEE, pp. 543–548.

West, A. (n.d.). Smartphone, the key for bluetooth low energy technology.
URL: *http://www.bluetooth.com/Pages/Smartphones.aspx*

Ye, X. (2006). Towards a reliable distributed web service execution engine, *International Conference on Web Services (ICWS)*, IEEE, pp. 595–602.

Yildiz, U. and Godart, C. (2007). Synchronization solutions for decentralized service orchestrations, *International Conference on Internet and Web Applications and Services (ICIW)*, IEEE, pp. 39 –39.

Yoon, J., Liu, M. and Noble, B. (2003). Random waypoint considered harmful, *Annual Joint Conference of the IEEE Computer and Communications. IEEE Societies (INFOCOM)*, Vol. 2, IEEE, pp. 1312–1321.

Yu, W. (2009a). Decentralized orchestration of BPEL processes with execution consistency, *Joint International Conferences on Advances in Data and Web Management APWeb/WAIM*, Vol. 5446 of *LNCS*, Springer, pp. 665–670.

Yu, W. (2009b). Scalable services orchestration with continuation-passing messaging, *International Conference on Intensive Applications and Services (INTENSIVE)*, IEEE, pp. 59–64.

Zaplata, S., Hamann, K., Kottke, K. and Lamersdorf, W. (2010). Flexible execution of distributed business processes based on process instance migration, *Journal of Systems Integration* **1**(3): 3–16.

Zeng, L., Benatallah, B., Ngu, A., Dumas, M., Kalagnanam, J. and Chang, H. (2004). Qos-aware middleware for web services composition, *IEEE Transactions on Software Engineering* **30**(5): 311 – 327.

Zhao, D., Qu, Z., Yang, Y. and Feng, X. (2011). A service negotiation mechanism in mobile ad hoc network, *International Conference on Wireless Communications, Networking and Mobile Computing (WiCOM)*, pp. 1 –4.