# Discovery of Autonomous Semantic Services in Mobile Ad Hoc Networks

**Andronikos Nedos**

A thesis submitted to the University of Dublin, Trinity College

in fulfillment of the requirements for the degree of

Doctor of Philosophy

November 2006

## Permission to Lend and/or Copy

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

Andronikos Nedos

Dated: November 30, 2006

# Declaration

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

Andronikos Nedos

Dated: November 30, 2006

# Acknowledgements

The process of a Ph.D can best be understood in the time and space continuum [Schaeffer, 1972]. There is a strange linearity associated with that process meaning that as the time required to finish the thesis increases, the space it occupies in the mind and life of the participant, also increases, sometimes dramatically. Fundamentally though, a Ph.D is a journey of discovery, whose destination is hidden initially and is only progressively revealed with time (though it can happen that it never really discloses itself fully). As with all such journeys, it has bumps, slumps, dead-ends, lows and highs and the eventual acceptance that the journey is what matters.

This journey would never have been possible without the superb assistance of my supervisor Dr. Siobhán Clarke. She has been a constant source of encouragement and motivation, always confident in my abilities and always willing to put time and effort when needed. A great deal of help and inspiration was also given by Kulpreet Singh. It is infrequent to tread such a journey with such a fellow traveller. Many thanks go to Raymond Cunningham who put a lot of effort to help me overcome barriers in the mathematical parts of this thesis, Stefan Weber who answered all and every obscure systems question I ever had, have and will have, Anthony Harrington for the most enlightened discussions and morale support, Tim Walsh for pub related research and donating his machines so that experiments took a few weeks rather a few months, Mads Haahr and Jim Dowling for wacky conversations and sharing their inspired thoughts and finally, Dr. Cathal Walsh for last minute stochastic advice.

There are so many other great people in DSG that is hard to list them all. Many thanks go to all those who encouraged me, fed me, gave me caffeine and advised me, especially during the final months of the write up. Thanks to Prof. Vinny Cahill for establishing such

a remarkable group and giving me the opportunity to be part of it. Finally, this thesis would not have been possible without the support of my parents, my sister and my cousin.

# Abstract

Mobile Ad Hoc Networks (MANETs) are dynamic networks where mobile devices with wireless communication capabilities cooperate to provide spontaneous connectivity. In particular, a mobile ad hoc network can be considered an open distributed system, where autonomous mobile nodes with limited capabilities can join and leave the network at any time. The resulting topologies are decentralised and differ from those in stationary networks in their high degree of randomness. Although a challenging environment, MANETs find wide applicability in scenarios where spontaneous networking is required, e.g., pervasive computing. This thesis assumes that such scenarios will make use of collaborative applications with functionality that is distributed across nodes.

In such a dynamic environment, the location of required functionality cannot be hardwired. Nodes can acquire transient addresses and may fail or partition at any time. The result is that discovery becomes an important process preceding any collaborative effort. In general, discovery attempts to match required and available functionality by using an appropriate representation language and suitable network support. Service Oriented Computing (SOC) has emerged as a paradigm to design distributed applications by representing resources as modular and independent services that can be advertised, discovered and invoked. In MANETs, representation and discovery of services has several unique requirements: the opportunistic nature of the network necessitates reduced human intervention and hence requires automated discovery; node autonomy implies that service representation is difficult to standardise or to guarantee its agreement between providers and consumers; the open and dynamic nature of the network requires discovery mechanisms that scale; and resource constrained devices dictate efficient distribution of discovery load.

Service architectures that use syntactic representations and assume a priori agreement on interfaces (e.g., Sun's Jini, IETF's SLP, Web Services) cannot support automated discovery in a MANET environment. A more suitable representation is one that can describe functional service characteristics. Semantic services (e.g., OWL-S, WSDL-S, WSMO) use ontologies to represent service capabilities and properties. The formal underpinnings of ontologies means that inference can be used to automate the matching between required and available services. However, current semantic discovery protocols are designed for stationary networks and assume common ontologies that are always reachable and known to both service providers and consumers.

This thesis proposes the use of a semantic services model that provides automated discovery of distributed functionality in mobile nodes. The contribution of this thesis is twofold. First, it caters for mobile dynamic networks and semantic decentralisation. Semantic decentralisation is the idea that autonomous nodes can express services using different ontologies that are not a priori defined. Second, the thesis describes a distributed service discovery model called OntoMobil. The model relies on the decomposition of ontologies into concepts and the dissemination of these concepts through a novel gossip protocol. This randomised concept dissemination mechanism facilitates eventual semantic agreement between heterogeneous ontologies and provides a substrate for scalable discovery of services. A random walk protocol is used for the actual discovery. The random walk protocol uses a two phase approach where semantic queries are first routed and subsequently evaluated at any provider node with a compatible ontology.

This thesis specifies the distributed model, presents a stochastic analysis of the gossip protocol and evaluates the performance of the gossip and the discovery protocols. An implementation is provided that uses RDFS to describe the semantic services and ns2 to simulate a mobile ad hoc network. A generalisation to a well-known semantic formalism, Description Logics (DL) is also described. Simulation results verify the stochastic analysis and show the scalability trade-offs of the model.

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

This thesis examines the problem of discovering semantic services in the domain of mobile ad hoc networks. The problem is studied within the context of autonomous mobile nodes that can be both providers and consumers and maintain services that are described without a priori agreement on common ontologies. The thesis proposes a solution in the form of a scalable model where nodes transmit metadata and establish associations between similar concepts. The model, *OntoMobil*, facilitates distributed matching between heterogeneous ontologies and achieves discovery of semantic services with probabilistic guarantees. This chapter provides the background and motivation to this work, introduces OntoMobil and concludes with the contributions and the layout of the thesis.

## 1.1 Background

A brief description is given here about the domain of mobile ad hoc networks and the fields of service oriented computing, semantic services and resource discovery. The focus is on issues that are relevant for this thesis, i.e., network topologies, evolution of software design for distributed applications and the semantic modelling of application functionality.

### 1.1.1 Mobile Ad Hoc Networks

A mobile ad hoc network (MANET) is a network formed on-demand, where nodes communicate without the aid of preexisting infrastructure. Nodes can join and leave an ad hoc network at any time and communication links beyond the wireless range of each node are established by utilising a cooperative forwarding strategy. The resulting topologies are dynamic with a high degree of randomness and unpredictability [Perkins, 2001].

Ad hoc networks were initially used for military purposes because of the good fault-tolerance guarantees offered by their distributed topology. However, the recent miniaturisation and robustness of wireless and processing units and the resulting rise of context aware and collaborative applications (e.g., games, m-commerce), have renewed interest in MANETs. The MANET property of spontaneous networking is a strong requirement for scenarios in pervasive computing, home networking [Kortuem et al., 2001] and emergency services [Perkins, 2001].

The challenging characteristics of MANETs have attracted a considerable amount of research in the lower networking layers. Specifically, a lot of work has been invested in protocol designs for the physical, medium access and routing layers. Similar interest has also been devoted to multicasting, unicasting and security [Perkins, 2001, Murthy and Manoj, 2004]. Although approaches range widely, an overarching theme is that of autonomous nodes that cooperatively and opportunistically engage to fulfil a task or provide a given service. Depending on the protocol layer, a service can be anything from collaborative routing [Dowling et al., 2005, Füßler et al., 2003] to security functionality [Capkun et al., 2003].

Applications in MANETs can also increase their utility using self-organising strategies. For applications to capitalise on cooperative behaviour, they need to identify and incorporate certain characteristics into their design. For example, collaborative features of MANET routing protocols include neighbour discovery and resource sharing (e.g., routing tables and packet forwarding). Such features can find a natural correspondence in the application layer. In this layer, applications must typically discover required functionality and share their own resources. Discovery and resource sharing are traditional issues in distributed systems, however, mobile autonomous devices bring the additional requirements of short-lived connectivity,

decentralised operation and agreement on resource interfaces.

These requirements are clarified when we consider an ad hoc network as an open distributed system. According to Castano et al. [2005], an open distributed system exhibits the properties of *system dynamism*, *node autonomy*, *absence of a priori agreement* and *equality of node responsibilities*. In MANETs, system dynamism maps to a network of unrestricted size, where nodes establish transient connections; autonomy characterises independent nodes that intentionally provide or require resources with an interface that may be node specific and not standardised; absence of a priori agreement corresponds to the opportunistic aspect of MANET formations that cannot assume the existence of a central coordinating authority, therefore making common resource representation unrealistic; and finally the mobile nodes are considered peers with similar physical resources.

### 1.1.2 The Service Paradigm

Service-oriented computing uses the service paradigm as the principal abstraction to build distributed applications [Huhns and Singh, 2005]. The philosophy of *advertise*, *discover* and *use* makes service-oriented computing a suitable model for on-demand sharing of application functionality. In this paradigm, services are software modules that encapsulate application logic. The advent of services represents a continuation from the domains of object orientation and component design, from where services inherit many of their properties

In object-oriented systems, an object is the main unit of problem decomposition. An object typically has a unique identity and encapsulates state and behaviour. Object-oriented design uses the properties of inheritance, encapsulation and polymorphism to build systems composed of objects. Traditionally, object-oriented systems were designed for non-distributed deployments, which meant that connecting with other systems required specialised support. The fine grain encapsulation of objects also made it harder to build distributed systems that would be loosely coupled and easier to maintain and evolve. With the advent of technologies such as CORBA and Java RMI however, the boundary between local and distributed deployment has blurred.

On the other hand, a component-based system is built as a collection of components. A

component has a strict contract, independent deployment and can be developed or purchased outside organisational boundaries. Components are frequently described as black boxes that expose only an interface with no externally observable state. These properties make components more appropriate candidates for distributed systems. Indeed, technologies like the OMG's CORBA Component Model (CCM), Sun's JavaBeans and Microsoft's COM/DCOM enable the composition of systems from interacting distributed components. Such systems are typically deployed in an intra-organisational scale. Facilities such as the Object Query Language (OQL) [O2, 1998] and the CORBA Query Service [OMG, 2000] make basic component discovery possible. There are inevitable similarities and subtle differences between objects and components. A more detailed discussion on the differences can be found in Szyperski [2002, Section 4.1.3].

Two properties differentiate service-oriented computing from object and component-based systems; standardised access methods and loose interface coupling. The first is apparent in the architecture of web services [Booth et al., 2004], a diverse suite of standards that have a major influence in service-oriented computing. Web services pioneered standardised access methods by adopting XML as the interface specification language and HTTP as the transport language [Stal, 2002]. The use of standards has enabled the progressive layered specification of more complex interaction patterns, ranging from XML-RPC [Winer, 2001] and SOAP [Gudgin et al., 2003] to WS-Coordination [Microsoft et al., 2005] and WS-Transaction [Cox et al., 2004]. To enable the rapid evolution and the adaptation to the dynamic Web environment, service-oriented computing features loose interface coupling, which keeps the service interface between clients and providers deliberately open and not specified. The combination of standardisation and loose interface coupling have enabled web services to scale across administrative domains and the Internet.

The evolution in the different paradigms can be summarised according to *granularity of abstraction*, *automated interaction* and *deployment scale*. Granularity of abstraction defines the degree of encapsulation in each paradigm. Automated interaction captures the suitability of each paradigm for complex interactions, e.g., discovery, binding and composition, without manual intervention. It describes the level of machine-processable representation that is

| | abstraction granularity | automated interaction | deployment scale |
|---|---|---|---|
| object paradigm | fine-grained | none | small |
| component paradigm | coarse-grained | basic | intranet |
| service paradigm | coarse-grained | basic | Internet |

**Table 1.1**: Evolution in abstraction paradigms for distributed computing.

necessary for automation. Although automated interaction is a complex issue, this discussion considers only the availability of query languages and interface standardisation. Finally, deployment scale presents a qualitative metric for the expected number of interacting entities in each paradigm.

Objects have fine-grained granularity since their interface exposes a lot of implementation details. Typically, objects do not provide facilities to discover their functionality, though object introspection and reflection can provide the basic infrastructure. Because of these characteristics, scale is limited to small networks or a single processing environment.

Component systems are designed to encapsulate coarser granularity and in many cases a set of objects compose a component. Since it is treated as a black box implementation and can be independently deployed, some standardisation on component interfaces is possible. As such, they can typically scale within an organisation's intranet.

Finally, services are designed to encapsulate enterprise scale functionality, with a very high level interface. Services are essentially stateless and can be deployed on an Internet scale. Although facilities like the Universal Description, Discovery and Integration (UDDI) [McKee et al., 2001] offer attribute-based service discovery, the deployment scale and lack of standardised, expressive interfaces cannot support automation. Table 1.1 illustrates the comparison. Other properties, such as security or reliability guarantees, are omitted as they are not directly relevant to this thesis.

Apart from the Web, the service paradigm is also a suitable candidate for application interoperability in ad hoc networks. Its distinguishing features of high encapsulation, focus on modularity, the asynchronous communication paradigm and the principles of advertise, discover and use have correspondence in the design of lightweight collaborative applications

for small mobile devices.

### 1.1.3 Semantic Services

To invoke a service, knowledge of its interface is required. An interface can be represented in a variety of ways, e.g., as a numerical identifier, as a method name or as a set of attribute value pairs. It constitutes part of the contract or the implicit knowledge between the client and the provider of the service. Web services do not specify a coordinating entity to map interfaces to the functionality of services or to guarantee consistency in interface names so that contracts between clients and providers remain valid. One of the consequences of loose interface coupling is that service discovery remains a manual task. Through an unspecified process, one has to first identify a suitable service that matches specific user requirements and subsequently invoke the service based on its interface.

When minimum user intervention is required in mobile networks with no stable and permanent set of nodes, manually identifying relevant services requires user involvement and constitutes additional discovery latency. Representing services in terms of capabilities rather than in terms of names can address the problem of complex service interactions. A more expressive service description can encourage automated discovery based on queries about explicit functionality [Martin et al., 2004], rather than discovery based on the implicit association between syntactic interfaces and functionality. Semantic services advocate the use of ontologies for the description of capabilities and as the *semantic interface* for discovery.

Semantic web services represent the largest effort to describe services with ontologies and have gained wide industrial and academic acceptance. The motivation behind semantic web services is the automated access and discovery of all available web information. It is a vision that combines semantic representation with the service-oriented computing model.

Semantic representation addresses both the formal description of content and also the automated and goal-oriented knowledge extraction using inference mechanisms. We briefly explore these two parts here.

The need for a formal characterisation of content has arisen because of the growing need to integrate data from different domains. In an open environment, one way to achieve that

is by sharing metadata [Shadbolt et al., 2006]. However, a top down agreement on data schemas cannot work in large scale systems like the Web. Instead, a layered approach to integration can be more effective. This is currently undertaken by the semantic web. By layering specifications that increase in formal expressiveness, bridges for shared semantics can be built. Two such efforts are the Ontology Web Language (OWL) [McGuinness and van Harmelen, 2004] and the Web Service Modeling Ontology (WSMO) [de Bruijn et al., 2005]. In particular, OWL defines a set of languages layered on top of standards such as XML and the Resource Description Framework and Schema (RDF/RDFS) [Beckett, 2004, Brickley and Guha, 2000], and make use of Universal Resource Identifiers (URI) [Berners-Lee, 1994] for distributed identification of ontologies and concepts. The layered approach has led to different OWL flavours that have increased expressiveness, as most are based on the formalism of Description Logics (DL), while their syntax is compatible with accepted standards. Additionally, the use of URIs and the graph model provided by RDF enables ontologies described in OWL to cross reference terms and thus improve their extensibility. Note however, that while knowledge representation in OWL is more formal, it is also more complex to describe and represent than in RDF/RDFS.

The canonical layered model for the semantic web was first illustrated in Berners-Lee [2003] and is reprinted in Fig. 1.1. This progressive agreement from mere symbols (XML) to ontology (OWL) and logic (SWRL) is a significant building block for the Web Services Architecture (WSA) [Booth et al., 2004]. In both the semantic web and WSA this layering is a crucial step because it captures knowledge as "meaning" of increasing semantic complexity and represents that in a form suitable for a network of unprecedented scale.

Although a formal representation of content is necessary to automate access to information, it is not sufficient. Inference is also required to extract hidden knowledge, which is accomplished with the use of reasoners. Fast and efficient reasoners are an important requirement when scale is an issue. Fortunately, extensive theoretical work on formal ontology reasoning, mainly in the field of Description Logics (DL), was done in parallel with the development and implementation of reasoners [Baader et al., 2003, Chapter 8]. Formalisation and a wide choice of reasoners led to the establishment of Description Logics as the main

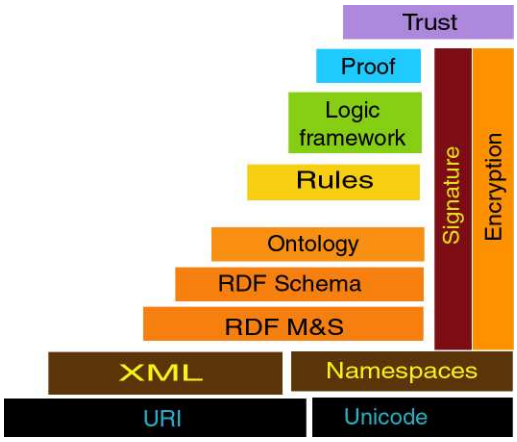


**Fig. 1.1**: The semantic web stack [Berners-Lee, 2003, slide 30].

theory for semantic representation. This is depicted in the fact that two versions of OWL, OWL-Lite and OWL-DL, and a version of the Web Service Modeling Language (WSML), WSML-DL [Lausen et al., 2005], are based on DL.

Both the formal representation of service functionality and the associated capacity for inference reduce human intervention and automate the tasks of service discovery, invocation and composition. Since this thesis is concerned with the discovery of services, we further elaborate on the issues of discovery.

### 1.1.4 Service Discovery

Service discovery faces two fundamental problems. First, there is the problem of *what* to discover, i.e., what is an appropriate interface that can represent service functionality and can also support the evaluation of complex queries against such an interface. Complex queries are necessary to increase the flexibility of enquiry, especially when knowledge is distributed and not represented in a standardised form. Choices can range from type-based interfaces to capability-based languages. The second problem is *how* to discover a service, i.e., how to design a discovery mechanism so that given a service representation, providers with matching services can respond. The two issues of representation and discovery are elaborated below.

### Service Representation

Through efforts to formalise the definition of a service [Dumas et al., 2003, Baida et al., 2004, Haas and Brown, 2004, Booth et al., 2004], the issue of representation has received a lot of attention. Representations have progressed from simple hierarchical naming schemes, such as X.500 [Chadwick, 1996], to complex semantic descriptions, e.g., OWL-S [Martin et al., 2004]. The reason for this progression is the need for more complex service interactions (e.g., automated discovery, composition, orchestration), which in turn is caused by the the large scale deployment of services. A similar need has also appeared in decentralised environments where opportunistic communication is the norm [Heflin, 2004]. Examples of these environments can be found in ubiquitous computing scenarios where tasks require distributed execution and collaboration amongst transiently connected devices. Since syntax-based interfaces could not adequately capture all possible interactions and required a lot of administrative support for standardisation and coordination, the development of more sophisticated service representations was necessary.

Early systems placed services very close to the operating system [Beitz et al., 1995]. By trying to bridge systems based on fundamental operating system services (e.g., remote file systems, monitoring services), the focus was more on service invocation and there was an implicit assumption that interfaces were known. As such, type safety issues were important, so representation was partially consisted of types.

With the move from centrally managed distributed systems to open distributed systems like the Internet, many assumptions about service representations required reexamination. The W3C's Web Service Architecture (WSA) [Booth et al., 2004] captures these assumptions as an agreement in three layers. At the bottom there is agreement in terms of form, i.e., agreement on syntax. Higher up there is agreement in terms of meaning, i.e., agreement on semantics. The top layer is specific to the message-oriented design of web services and is agreement on message flow.

As more complex interactions depend on the ability to query functional service characteristics, the need for more expressive representations arises. Agreement on various syntactic descriptions, for examples typed interfaces, attribute value pairs, URI's or tuples, cannot

represent service capabilities adequately. Instead, ontologies offer a more general and comprehensive formal description that can capture meaning and semantics. However, agreement in meaning requires common ontologies. In an environment with autonomous participants, this is equally challenging to achieve as agreement on syntactic descriptions.

### Discovery Mechanism

A discovery mechanism gives the ability to defer lookup of functionality until it is needed. This thesis covers discovery mechanisms that are distributed and as such they usually require specialised discovery protocols. The function of discovery protocols can be decomposed into the following processes, *request*, *matchmaking* and *response*.

A request routes[1] a service query to nodes where it can be evaluated. The process of evaluating a service query against available services is called matchmaking. After matchmaking, a response may be transmitted back to the service client, with the response incorporating a service handler if successful or just indicating failure.

Practically every discovery protocol, from Bluetooth's Service Discovery Protocol (SDP) [Blu, 1999] to UDDI, specifies all or some of these processes. Since service discovery generally crosses processing boundaries, many distributed computing issues [Mullender, 1989, Chapter 1], such as timeliness, reliability, security and scale are pertinent to discovery. Moreover, the choice of representation language has a significant effect and cannot be separated from the design of the discovery mechanism. The current proliferation of discovery protocols, e.g., SLP, Jini, SSDP, WS-Discovery [Beatty et al., 2005], could be partly explained by the many possible combinations that arise between service representations and protocol properties.

In fixed networks of small scale, a service representation that employs standardised syntactic interfaces can provide a simple solution. In terms of the discovery protocol, a broker architecture or the use of multicast can achieve efficiency with low complexity. Sometimes both brokers and multicast is used. The IETF's SLP [Guttman et al., 1999], Sun's Jini [Arnold et al., 1999], Bluetooth's SDP and Microsoft's Simple Service Discovery Protocol (SSDP) [Corporation, 1999] specifications follow this design. The scale of targeted networks and the

[1] We use the non technical definition for the word route here. It can imply multicast, unicast or any other communication paradigm.

assumption of a central administrative authority makes this model practical. Interestingly, since networks are managed and the interfaces are static and well-known, some basic service automation, i.e., service interaction without manual intervention, can also be achieved.

In open networks that have the scale of the Internet, service standardisation is impractical. Web services address this problem by *avoiding* standardisation. Service clients either have to locate the needed interfaces in an "out of bound" fashion, i.e., a different channel than the one used for discovery such as a google search, or service brokers that aggregate available services emerge to fill the need. For example, UDDI provides an Internet-wide brokering service, but its centralised nature raises questions about its ability to cope as the number of available services increases. Furthermore, retrieval in UDDI is syntax-based and so in the absence of a web services naming standard, UDDI is not appropriate for automated discovery. A current trend is the utilisation of P2P networks to distribute the load of service discovery [Schmidt and Parashar, 2004]. The intersection between representations that allow complex service interactions and scalable discovery protocols is the central theme of this thesis and is an emerging new field that promises to integrate decentralised network topologies with automated knowledge extraction.

## 1.2 Motivation

The proliferation of mobile devices is giving everyone access to computing power that although limited, is enough to run basic applications. The incorporation of multiple short-range wireless interfaces (Bluetooth, 802.11, Zigbee [Kinney, 2003]) has also equipped these devices with the ability to spontaneously form networks when other devices are in proximity. Communication in these uncoordinated networks happens through a network stack composed of standardised protocols. Expecting all devices to use a common set of protocols is a reasonable assumption, since they are usually part of the operating system and perform a very specialised function that most users do not replace.

The same is not the case for applications. They are high-level programs with functionality that varies widely. They can be installed and uninstalled on demand and high level languages have made it easier than before to develop new or extend existing applications. This presents

a challenge when applications, typically from categories such as mobile games, m-commerce, context-aware or pervasive computing, wish to augment their utility and collaborate by sharing functionality. The plethora of mobile applications coming from different users and from different administrative domains means that even similar functionality will be encoded differently. This makes application interoperability based on interface standardisation unrealistic.

The problem of standardisation is further compounded when one considers the self-contained and unpredictable ad hoc environment where devices can only discover functionality that is available at connection time. In such an environment, automated discovery, flexibility in query formation and even the ability to provide remote functionality that only partially matches what is requested are desirable properties. One solution is for applications to discover functionality based on required capabilities, rather than hard-wired interfaces. This removes the need for standardisation and enables flexible requests and automated discovery. However, this solution presents new challenges in that a level of agreement on capability representation is still required to achieve meaningful discovery, while the open and dynamic nature of MANETs dictates a discovery approach that is scalable and efficient.

The principle motivation behind this thesis is the formation of a mechanism so that independently developed applications running in small devices can locate required functionality in a distributed and mobile network. Using semantic services as the main abstraction for encapsulating shared functionality, the automated discovery of services remains unresolved, when projected in an open, decentralised environment, where semantic agreement cannot be guaranteed. This *semantic decentralisation* can find applicability in situations where uncoordinated interaction and ad hoc configuration of applications from different administrative domains is required.

For example, a similar class of applications is considered by the `SpatialViews` programming model [Ni et al., 2005]. Amongst the supported applications are sensor data aggregation, dynamic service installation and an augmented reality (AR) pacman game. To enable the programming of this class of distributed mobile applications the `SpatialViews` language adopts the use of services that have an interface and semantics that must be agreed upon by all nodes in the ad hoc network. It can be envisioned that a potential integration between

the proposed model in this thesis and a programming model similar to `SpatialViews` would enable applications to define services independently and without agreement on basic data types such as location or time.

In networks where knowledge representation is difficult to standardise, it is expected that communicating entities will introduce independent conventions. A comprehensive specification of this problem is found in Aberer et al. [2004], where the authors use the term *emergent semantics* to describe semantic consensus based not on standardisation but on emergent behaviour. The general problem of different representations is mapped to the problem of heterogeneous domain ontologies. The authors argue that an ontology is another instance of a common vocabulary and requires agreement and standardisation to be useful. In environments that are open and where unanticipated interactions are likely, common ontologies are insufficient and impose unnecessary constraints. Instead, what is advocated is an acceptance that similar information may have different semantic representations. It is also suggested that the formulation of semantic resolution protocols can address the mismatch problem in a best-effort manner.

In MANETs, the model of emergent semantics is appropriate and useful. Mobile ad hoc networks, unlike other open distributed systems, are self-contained as they can form in places where no Internet connectivity is guaranteed. It follows that nodes cannot avail of common ontologies on the Internet, but can only use the semantic knowledge of connected nodes. This is an appropriate setting for semantic resolution protocols to assist service interaction and application interoperability.

In the distributed context envisioned by the emergent semantics model, maintaining semantic interoperability remains a strong requirement. Even when resources and services are described by multiple and independently developed ontologies, service discovery, content retrieval and semantic inference should be guaranteed as if operating in a single ontology. To this end, appropriate mechanisms are required to map or translate metadata [Aberer et al., 2003, Doan et al., 2002]. The field of ontology matching [Shvaiko and Euzenat, 2005] has extended earlier work in database schemas [Rahm and Bernstein, 2001] to the area of ontologies. Projects like H-MATCH [Castano et al., 2003a], LARKS [Sycara et al., 2002] and

GLUE [Doan et al., 2002] accept semantic heterogeneity and have devised techniques and algorithms for ontology matching, mapping and evolution.

The additional challenges that MANETs pose to semantically heterogeneous mobile services relate to scalability and mobility. In particular, scalability is a concern that is caused by the open and distributed nature of MANETs. Networks can expand and shrink in size when new nodes join and existing nodes disconnect. Furthermore, node autonomy means that nodes will be providers as well as consumers of services. This changes the assumption in traditional discovery architectures that service clients outnumber the providers and requires a novel model to address the challenges of large numbers of mobile providers.

Similar scalability concerns in semantic interoperability have also been investigated in the domain of P2P networks. MANETs share a lot of characteristics with P2P networks. In both environments, nodes are autonomous, have the capacity to contribute local resources and require access to remote ones. A number of efforts [Nejdl et al., 2002, Castano et al., 2003b, Schlosser et al., 2002] are looking into decentralised topologies that can better support data and discovery queries described in more expressive terms. Such topologies typically assume data described in RDF or other similar formalisms and are designed to address the lack of expressive querying in contemporary P2P networks, such as Chord [Stoica et al., 2001] or Pastry [Rowstron and Druschel, 2001]. This problem becomes particularly acute as P2P networks grow in scale and their shared data is stored and accessed in representations that are unstructured and keyword-based. Instead, a semantic representation would relieve users from having to query data based on syntactical naming conventions that are "guessed", thereby improving recall and precision of user queries.

There are also differences. In the P2P case, the network and the nodes will generally be of higher capacity (e.g., bandwidth, processing power) and the churning rate will be lower than what is assumed for mobile ad hoc scenarios. Additionally, modern P2P networks can scale to thousands of nodes [Jelasity and Babaoglu, 2005], while in mobile ad hoc networks even large scale simulations do not involve more than a few hundred nodes [Kurkowski et al., 2005]. Even at this scale, problems with scalability are serious however, affecting low layer protocols as much as service interaction. For example, many discovery protocols used in

ad hoc scenarios [Helal et al., 2002, Chakraborty et al., 2002, Authority, 2003, Nidd, 2001], depend on the existence of a broadcast (flooding) or multicast protocol. Although single hop broadcast is readily available, the lack of standardised and efficient network-wide broadcast functionality in MANETs, results in the use of flooding, which impacts network performance. This can be a limiting factor when scalable discovery is required. Finally, node mobility requires specialised protocols that deal with topology changes and node failures. In general, application interoperability in MANETs faces issues that require careful consideration of common assumptions as well as the utilisation of techniques that scale and adapt to mobility.

We can now summarise the list of requirements for supporting interoperability between independently developed applications in mobile ad hoc networks:

- The automated discovery of remote services is required by applications that need to interact in an uncoordinated fashion and because of transient network connectivity. These factors are incompatible with the time consuming manual selection of available services. Automated discovery without standardisation is not possible using syntax-based service interfaces. Rather, it necessitates the use of a more expressive language that captures the capabilities and functionality of services. Using ontologies to describe services can provide the required flexibility and expressiveness.

- When service-based applications from different administrative domains interact in a MANET, it is inappropriate to assume that service interoperability will be facilitated by means of a common ontology. Instead, autonomy requires each node to maintain a different ontology that will be used to represent semantic services and may not necessarily be known a priori by other nodes.

- Discovery of semantic services when described by heterogeneous ontologies requires semantic integration. The mobile ad hoc environment dictates that this integration should not be derived from standardisation but it should be dynamic and emerge through the interaction of the participating nodes. This requires a method to establish eventual semantic agreement between the different ontologies.

- Open networks with nodes that can be both service providers and consumers require

discovery protocols that scale. Distributed techniques can address scalability issues for both discovery and semantic integration. Furthermore, mobility and limited device resources dictate that selected techniques must be adaptable, configurable and efficient.

## 1.3 A Distributed Model for Service Discovery in MANETs

This thesis presents OntoMobil, a distributed model to support the automated discovery of services. Discovery is addressed within the domain of mobile ad hoc networks, where the issues of transient connectivity and failure prone nodes need consideration. OntoMobil permits the independent description of services, facilitates eventual semantic agreement and provides scalable service discovery with probabilistic guarantees.

The challenging MANET environment and the envisioned application requirements have led to the design of a new discovery mechanism over the adaptation of an existing one. There is currently no discovery protocol in MANETs that could support semantic decentralisation in a scalable way.

### Scope

The model described in this thesis supports the discovery of semantic services. The process of matchmaking between required and provided services is outside the scope of this thesis, though one of the many algorithms [Li and Horrocks, 2003b, Trastour et al., 2001, Schade et al., 2004] that have been proposed in the literature could be leveraged without difficulty. The model does not specify service composition or invocation as they are considered orthogonal to discovery. Especially for invocation, it is envisioned that using a specification similar to OWL-S, a high level semantic query can be grounded to a set of service calls that can be invoked by the client and served by the provider.

### Assumptions

The model assumes that nodes are peers of equal responsibility and processing capacity. At any time, each node can be either a service provider or a client or both. In reality, the model

is generic enough to allow only a subset of nodes to share services within a larger ad hoc network. It does not constraint all nodes to run the OntoMobil protocol stack, allowing non-participating nodes to be oblivious to the exchange of services, forwarding broadcast packets from OntoMobil nodes only during the initial stages of a bootstrap process. An additional assumption is that nodes and applications can be trusted and security and privacy is not a concern.

Independent service description is realised by having different nodes maintain different ontologies to describe their services. The model makes no assumptions about whether ontologies describe the service's capabilities, properties, policies, context or security aspects. There is also no assumption as to whether each node uses one or many ontologies to describe its services. There is an assumption, however, of an agreed upper ontology or a common schema for the actual service specification. An upper ontology dictates the organisational form within which different services can be described, and is different from the specification of the service's functional or non-functional characteristics.

The model considers an ontology to be composed of concepts, which makes any ontology specification (e.g., OWL, WSMO-DL) based on the formalism of Description Logics suitable. The model also assumes that each provider will have an ontology reasoner that can decompose an ontology into concepts and provide basic service matchmaking facilities[2].

Finally, it is assumed that discovery is not a "one-off" process. Rather, discovery of services or content in general, is integrated into the application logic and is an important part of the application functionality that is repeatedly used.

### Approach

In a network with many providers and heterogeneous service ontologies, in order for a service query to me meaningful, it must be evaluated against ontologies compatible with the ontology the query originated from. The query must also be matched against services defined in compatible ontologies in order to yield succesful results. Depending on the network scale, the two processes of ontology matching and service matchmaking can be resource intensive

[2]This usually happens with subsumption-based inference.

when centralised. This is especially true in an ad hoc environment, where relying on a centralised brokering architecture to identify similar ontologies and provide a repository for available services is not a practical solution. Transient network connectivity makes a brokering architecture prone to failures, while device resources cannot adequately support centralised ontology matching and service matchmaking.

The approach described in this thesis is distributed and relies on the formation of a randomised semantic overlay through the exchange of metadata. Ontology concepts are randomly replicated across participating nodes, while a semantic similarity module in each node is responsible for establishing relatedness between concepts from different ontologies. Semantic agreement is thus uniformly spread between all nodes, providing a substrate on top of which concept-based queries are routed to compatible ontologies. Service matchmaking can then take place at provider nodes with any services that have matched the query aggregated at the client node.

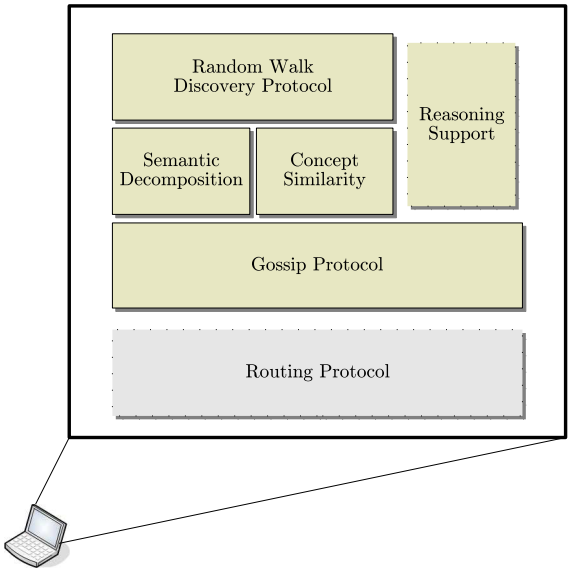


**Fig. 1.2**: Node architecture in OntoMobil. It is a layered architecturea, where lower modules provide their services to modules higher up in the stack. Reasoning is not provided by OntoMobil but is required for the processes of semantic decomposition and service matchmaking.

The basis of the model is the decomposition of ontologies and semantic service queries into concepts (Section 1.3.1), the use of a gossip protocol to disseminate random concepts

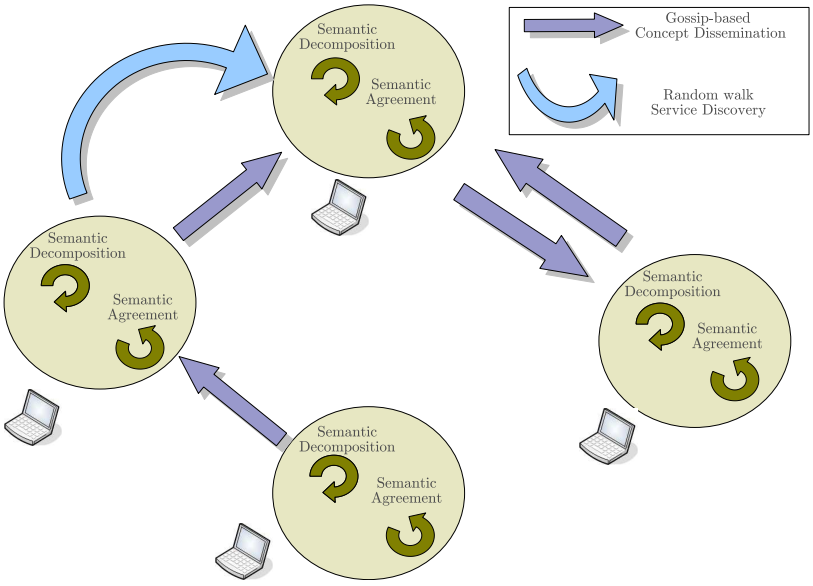


**Fig. 1.3**: The OntoMobil model illustrated as a set of independent processes that cooperate through concept exchange.

to random nodes (Section 1.3.2) and the matching of received concepts in each node (Section 1.3.3). Together, they provide the substrate on top of which a simple random walk protocol (Section 1.3.4) can be layered. This random walk protocol is responsible for routing service queries to appropriate provider nodes. Figure 1.2 shows the OntoMobil architecture in each node. The algorithms for the similarity and reasoner modules can be substituted and are independent of the architecture. The gossip protocol is executed periodically in each node until certain conditions are met. Service discovery is an on-demand process whenever applications require the discovery of new functionality. Figure 1.3 illustrates the model from a process-driven perspective. The semantic decomposition and semantic agreement algorithms run in each participating node. The gossip and the random walk protocols use point-to-point connections and do not require the use of multicast or broadcast facilities.

The advantages of this approach are flexibility, scalability and efficiency. Flexibility derives from allowing nodes to maintain heterogeneous ontologies. Different types of services can then be supported by different domain ontologies, while services in the same domain do not have to agree on common ontologies. The model achieves scalability through randomisation and the utilisation of each participating node in the identification of similar concepts.

Network-wide efficiency is preserved by avoiding the use of broadcast and thus reducing traffic. Additionally, since each node bears part of the responsibility towards semantic agreement and discovery, the use of excessive overhead in select broker nodes is avoided. However, discovery under these circumstances is fundamentally a best effort process. There is an implicit assumption that similar services will be modelled either with ontologies that have similar terms or concept matching will identify similarity even when ontologies are described differently. The following sections briefly describe the constituent processes of OntoMobil.

### 1.3.1 Semantic Decomposition

Concepts, together with relationships and concept attributes, form the constituent elements of an ontology. In the domain of Knowledge Representation (KR), concepts represent sets of individuals that are composed in hierarchies of concepts used to represent domains of interest [Baader et al., 2003, Chapter 1]. More informally, concepts model similar classes of objects. The idea of using concepts to discover services has also been used in different forms in other projects [Skouteli et al., 2005, Chakraborty et al., 2002, Arabshian and Schulzrinne, 2004, Schlosser et al., 2002]. The core idea behind all these efforts is location transparency [Mullender, 1989, Coulouris et al., 2005]. By disseminating concepts to network nodes, the level of discovery abstraction is raised. A discovery request can now avoid the direct location of the nodes that contain a matching semantic description and instead locate the concepts that compose the request.

In OntoMobil, a concept-based approach is used for the process of semantic agreement as well as for the discovery of services. This means that both ontologies and discovery queries are first decomposed into concepts and then routed to a destination. This symmetry is in itself a desired characteristic. Besides the conceptual uniformity, which simplifies and reduces the number of building blocks for potential middleware or application developers, it is more efficient when devices are small and networks vary in size to disseminate small sets of concepts rather than complete ontologies. On the other hand, the issue of semantic agreement becomes more problematic, when semantic matching occurs between concepts instead of ontologies. This trade-off is explored in more detail in Chapter 4.

### 1.3.2 Gossip-based Concept Dissemination

Epidemic or gossip algorithms are based on randomisation. Randomised algorithms generally have good load balancing properties, can model complex problems in an elegant way and are amenable to analytical methods. According to Motwani and Raghavan [1995], two main advantages of randomisation are performance and simplicity. Randomised algorithms outperform deterministic ones in many cases (e.g., skip lists, permutation routing problem), especially as the size of the input grows. Furthermore, they are simpler to specify and implement than some corresponding deterministic algorithms.

In OntoMobil, a specialised protocol is developed to *gossip*[3] a randomised subset of concepts to a randomised subset of participating nodes. This style of epidemic communication has been shown to provide a robust method against the unpredictable environment of MANETs [Haas et al., 2006, Luo et al., 2003]. Each node executes the gossip protocol and stores received concepts in a buffer until certain conditions are met. These conditions are part of a set of parameters that control the scalability properties of the model. Given the right set of values for these parameters, no node stores the full set of concepts from all ontologies but at any one time it contains a random subset of all concepts. This parameterised partial replication of concepts in each node is the key to the scalability of the model.

As new nodes join the network, their concepts are also spread uniformly across all participating nodes. Concept replication is used during discovery to identify concepts similar to those composing the service query. Because of randomisation, a probabilistic analysis is performed on the number of replicated concepts across nodes. The derived predictable model is important because it qualifies processing and memory costs for each node and is used in combination with a random walk to infer the probabilistic bounds for service discovery.

### 1.3.3 Facilitating Semantic Agreement

When services are described by different ontologies, it is necessary to precede discovery with identification of semantically related descriptions. The exact mechanisms to establish semantic closeness is outside the scope of this thesis. OntoMobil only *facilitates* the identification

[3]Transmit in gossip style.

of compatible ontologies and provides the mechanism to distribute matching across nodes. The matching process is progressive and mimics an epidemic style infection. As each concept is disseminated across the network, its semantic content "infects" other ontologies, thereby advancing the network's semantic agreement. Ultimately, this infection spreads throughout all ontologies resulting in global semantic agreement.

The proposed approach guarantees that matching is distributed, which leads to desirable load balancing characteristics. To identify semantic similarity, OntoMobil supports two matching methods, named *dynamic* and *template* matching. Both methods establish transitive relations between compatible ontologies but use different ways to identify concepts that are semantically equivalent.

### 1.3.4 Random Walk for Service Discovery

Random walk is a technique in stochastic modelling that is extensively used in the study of graph properties. In its theoretical setting, a random walk on a graph is defined as a series of discrete steps, where in each step a neighbour of the current vertex is randomly chosen and visited. Since MANETs can easily be modelled as graphs, random walks have a practical applicability in mobile ad hoc networks. In particular, random walks have been used in a group communication setting [Dolev et al., 2006] and to provide a uniform partial membership view [Bar-Yossef et al., 2006].

OntoMobil models discovery as a process that visits random nodes and collects information in order to locate candidate nodes. These candidates are nodes that maintain ontologies that are similar to the ontology of the node that issued the query. Subsequently, the discovery query is routed to these nodes, where it is evaluated against their available services. Services that match are transmitted back to the client node where they can be filtered and used appropriately. Because this process can be mapped into a random walk, probabilistic properties can be derived giving predictable bounds on the number of hops before a discovery occurs. An additional advantage of this process is that service matchmaking takes place at the provider nodes and omits the need for service brokers.

### 1.3.5 Towards an Ad hoc Interaction of Autonomous Services

This chapter introduced a mobile ad hoc network as an open distributed system composed of autonomous mobile nodes. Similar to the dynamic network formation that is typical of MANETs, collaborative ad hoc applications should be able to self-organise and interact in a spontaneous, on-demand and unconstrained fashion.

Semantic services are chosen as the abstraction paradigm for application interoperability. As the term *service* is used to describe a broad variety of activities, we provide here a definition for autonomous services that is close to the definition in Paolucci and Sycara [2003], but is adapted for the MANET environment.

> An autonomous service can represent specific application functionality with a minimum dependency on common and external specifications. It uses a well-defined and public interface that is described in an expressive language, which can capture the service's capabilities. Complex interactions between autonomous services emerge when network peers use mechanisms that can advertise, discover and invoke said interfaces.

In practice, the services considered in this thesis are similar to middleware services [Bernstein, 1996] in that they are *atomic*, can be invoked by means of a service handler and contrary to enterprise web services, encapsulate functionality relevant to mobile devices.

## 1.4 Contribution of Thesis

The contribution of this thesis has the following parts:

1. Current semantic service architectures for mobile ad hoc networks assume a common ontology. In OntoMobil, semantic services can be developed without a priori agreement on common ontologies.

2. Existing semantic topologies do not address the matching of heterogeneous ontologies. In order to facilitate semantic agreement between the different ontologies, this thesis

proposes a novel model that distributes the ontology matching process to all participating nodes.

3. Existing discovery mechanisms in MANETs provide either scalability through syntax-based service queries or semantic queries in networks of limited scale. The OntoMobil service discovery protocol accepts semantic expressions and uses the semantic overlay built by the gossip protocol to perform discovery with probabilistic guarantees without the use of broadcast or flooding.

4. Models amenable to analytical methods are desirable because they provide predictable behaviour under known assumptions. This thesis presents a Markov-based analysis of concept replication across nodes. This analysis results in a predictable stochastic model that incorporates all the important characteristic parameters and can be used to investigate the impact of network scale in each node.

5. Existing semantic overlay topologies that are based on ontology decomposition do not provide an adequate transformation from complex semantic queries to their constituent concepts. This thesis provides a simple set of rules for the syntactical decomposition of Description Logics queries into concepts.

An implementation of OntoMobil confirms the derived analytical bounds. The evaluation also extends to different strategies for concept matching that can illustrate with more accuracy the trade-off between processing requirements in each node and the progress of semantic agreement.

## 1.5 Thesis Layout

The layout for the rest of the thesis is as following. Chapter 2 covers the state of the art in decentralised service architectures for P2P and MANET environments. Chapter 3 specifies the system assumptions, describes the gossip protocol and analyses the concept replication distribution. Chapter 4 specifies a simple service profile based on RDFS and details the process of concept matching and the random walk discovery protocol. A generalisation from

RDFS to a description logic is offered at the end of the chapter. Chapter 5 describes the implementation of OntoMobil, Chapter 6 the evaluation and Chapter 7 the conclusion.

# Chapter 2

# State of the Art

The importance of software services in the design of flexible and dynamic applications can be seen in the proliferation of service architectures. Both in academia and in industry a wealth of specifications have been produced. This chapter reviews existing work in decentralised semantic topologies, ontology matching algorithms and service architectures that have similar goals and features with OntoMobil. Whenever appropriate, we compare and contrast specific features of OntoMobil with those of the systems under review. Although the main emphasis is towards systems designed for mobile environments, other work in related domains, mainly P2P networks is also covered.

## 2.1 Discovery in Decentralised Environments

Assuming a centralised architecture and an intuitive scheme to name available resources, discovery is often a sequence of simple steps. The activity of looking through yellow pages or initiating a google query are typical examples of this. When centralised architectures cannot be supported, specialised architectures and protocols are required. A typical example is P2P networks. Initially, discovery in these systems was confined to naming conventions with limited expressiveness, putting the emphasis on topology properties such as scalability and efficiency [Ratnasamy et al., 2001, Stoica et al., 2001]. Currently, a new generation of P2P networks uses semantic representations in decentralised topologies. This move represents

a shift from the description and discovery of content that is based on attribute value pairs to more expressive semantic annotation and querying. Service discovery can be considered a subclass of the general content discovery problem and has undergone a similar evolution. Service architectures have evolved from those that rely on a small number of providers and use syntactical name-based interfaces to architectures that feature large scale decentralised topologies that support the semantic description and discovery of services.

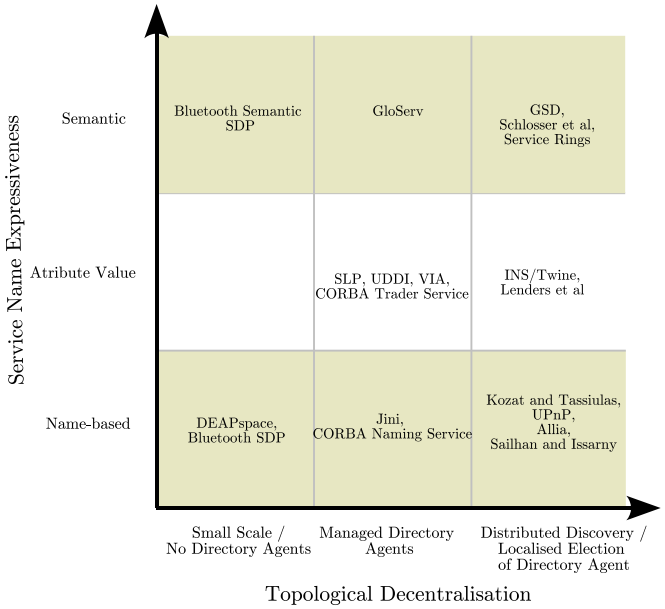


**Fig. 2.1**: The evolution of service architectures in relation to discovery protocols and expressiveness of service representations.

Figure 2.1 illustrates the evolution of service architectures in two axes. The progress in terms of topology is depicted horizontally, while the evolution in the representation of services is shown vertically. The systems presented at the intersection come mainly from the P2P, LAN and mobile domain. This is only a qualitative classification with a more detailed classification for MANET service architectures presented in Section 2.2. The next section focuses on discovery mechanisms for semantic content and services in stationary decentralised networks.

### 2.1.1 P2P Topologies for Semantic Content

In P2P networks, the issue of expressive content representation is receiving increased attention. Usually such networks come under the names of semantic or schema-based topologies. One of the first complete systems to explore this concept was the EDUTELLA project [Nejdl et al., 2002]. EDUTELLA is mainly concerned with semantic content rather than services but its goal is similar to OntoMobil: The distributed discovery of semantic information without the use of common ontologies. EDUTELLA assumes that information is described in heterogeneous schemas and this aspect influences many design decisions in its P2P topology. To address the heterogeneity problem, the authors define an architecture that provides facilities for schema integration and topology control and specify a custom query language [Nejdl et al., 2003].

EDUTELLA takes a layered approach both in terms of content description but also in terms of topology support. To allow discovery of information residing in distributed repositories, RDF is used for the metadata specification and an abstract query service is specified that provides a mapping between peer metadata and a common schema. This allows the distribution of queries to both individual and multiple RDF repositories. In contrast, OntoMobil does not support this extended query model, where different peer repositories are considered connected. In OntoMobil, each node is considered autonomous and although requests can be directed to multiple nodes, there is no mechanism in place to present a unified view of the different ontologies to service requests.

In terms of topology, EDUTELLA is based on super-peers, though the clustering algorithm has several flavours. The layered architecture has allowed the exploration of topologies that produce overlay clusters based on different semantic similarity rules. Two such variations are ontology-based clustering, where nodes with similar ontologies are clustered together and rule-based clustering where super-peers register cluster rules, e.g., schemas rules, IP address constraints and resource rules with nodes joining when they fulfil these rules.

Haase et al. [2004] describe another semantic P2P topology that is focused on efficiency. The authors start from a completely random network topology and eventually derive an efficient topology that closely matches the similarity in the semantic knowledge of peers. Under

their assumptions, each peer has access to a common ontology and routes incoming queries according to a similarity metric that is derived from hierarchical data. The authors evaluate the hypothesis that certain semantic topologies will perform more efficiently than random topologies. Through experimentation and simulation they illustrate that under a "perfect" semantic topology, where the network topology matches the similarity in peer content, precision and recall of queries achieve their maximum ratio. A number of assumptions underlie this hypothesis. Chief amongst them are the assumptions of similarity, stable connectivity and efficiency. It is assumed that participating nodes will have some degree of similarity, from which a semantic topology can be derived. The second assumption is that the latency cost of converging to a "perfect" semantic topology is offset by long-lived connectivity. Finally, efficiency is only achieved when each peer has knowledge about other peers with related semantic content. In OntoMobil many of these assumptions do not hold. Specifically, the assumptions of stable connectivity and efficiency in the face of complete semantic knowledge cannot be justified. Section 2.1.4 further elaborates on the problems of cluster topologies when heterogeneous ontologies must be supported in MANETs.

Castano et al. [2003b] describe a P2P overlay architecture, called H3, where each peer maintains an independent ontology. For scalability, the topology algorithm exploits the semantic relationship amongst the different ontologies, thereby forming a topology that mirrors the semantic similarity of peers. The H3 architecture is split in two layers. The knowledge infrastructure layer, called Helios, processes queries and matches incoming concepts with those maintained in each peer. The communication infrastructure layer, called Hermes, is responsible for the functions of routing queries based on their semantic content and providing membership functionality to peers. Although no concrete protocol is devised and performance results are not presented, the architecture is very similar to OntoMobil. It assumes semantic heterogeneity and an intermediate process that matches concepts disseminated from peers based on an affinity metric. Similar to OntoMobil, routing takes advantage of concepts that are augmented with peer addresses and relationships with other similar concepts. Differences emerge where OntoMobil defines a concrete protocol specification that details the node interactions and provides a analytical model to predict performance. Additionally, the model in

this thesis is specifically targeted for services and its design incorporated the issue of mobility.

REMINDIN' [Tempich et al., 2004] is another project that exploits query history in order to route semantic queries in P2P networks. It accepts that a common schema between peers is a simplistic assumption and assumes that peers describe content with heterogeneous meta-data. REMINDIN' does not define an exact topology. Through observation and recording of meta-information from user queries, it selects a number of peers that are likely to contain relevant results. Initially, when no meta-information has been recorded by peers, routing resorts to bounded flooding. As more queries are being observed, peers augment their knowledge of other peers that can answer queries about particular topics. This progressive acquisition of knowledge is a common characteristic between REMINDIN' and OntoMobil, though the overall model is very different. In OntoMobil, a proactive gossip protocol is used to construct a randomised semantic overlay using concepts that are replicated in each node. REMINDIN' on the other hand, does not define a topology but relies on queries, which are reactive by nature, in order to build a semantic similarity model.

### 2.1.2 P2P Topologies for Semantic Services

GloServ [Arabshian and Schulzrinne, 2004] is a P2P service discovery architecture that is designed to be both expressive and scalable. It uses OWL-DL to describe services and to derive a hybrid network topology that uses ontology concepts to classify services. The proposed topology is hybrid because it is constructed from a high-level service classification hierarchy and a low-level Content Addressable Network (CAN). The CAN topology is used to connect peers with similar services. The extra complexity of connecting and maintaining two different topologies is justified by the stated aim of GloServ, which is the design of a discovery architecture that is scalable and suitable for ubiquitous and pervasive computing.

The topology formation in GloServ occurs through the following steps. A hierarchy of service types, called the *service classification ontology*, is initially devised and is assumed to be immutable. Each service in GloServ inherits from at most one of the concepts in the classification ontology. The ontology is in turn mapped to a stable hierarchical network of GloServ servers. Since this ontology is coarse-grained and GloServ assumes that there will

be thousands of services, each server in the hierarchical network will be responsible for many service instances. Because this approach cannot scale, a second partitioning occurs at this stage. When services register they are classified as subconcept instances of some concept from the classification ontology. A CAN overlay network is created for each of the concepts in the classification ontology, and services can register in any of the CAN nodes. As more nodes join each of the multiple CAN overlays, the dimensions are split to accommodate the new joins. Service queries are handled in a similar fashion to service registrations. Each query is classified and is routed to the server that maps to the query's superconcept. Subsequently, according to the service's classification, it is again routed to the node in the CAN that is responsible for the service's concept.

The idea of routing a semantic query based on its concepts, is similar to OntoMobil. In practise, the two approaches differ as OntoMobil relies on concept replication for routing, while GloServ assumes a classification ontology that does not change and uses that as the initial point of contact. GloServ is still at the prototype level and therefore no evaluation data are available.

Schlosser et al. [2002] present a graph topology for scalable semantic service discovery. Peers are organised into a hypercube, giving the desirable properties of low network diameter, efficient message broadcast and resilience to node failures. The hypercube topology is also used to index ontology concepts, so that semantic queries can be routed to appropriate peers. Specifically, it is assumed that services can be classified according to globally known ontologies and peers may provide an arbitrary number of semantic services. In order to organise this semantic information, peers with similar services are grouped in concept clusters. A concept cluster is a set of ontology concepts that best describes the peers belonging to the cluster. Each concept cluster becomes a vertex into the hypercube graph. A semantic query can now be routed in a two-step process: the concepts composing the query are used for routing to the appropriate concept cluster. Once this is reached, a secondary broadcast occurs inside the concept cluster to identify suitable peers.

Query routing in OntoMobil uses a similar two-step process. Each query is first decomposed into concepts and routing is conducted by following a randomised yet bounded path

until it collects enough information to be dispatched to nodes with similar concepts. The difference between OntoMobil and Schlosser et al. [2002] is based primarily on the construction of different topologies. This stems from different assumptions of environment and service autonomy. While a hypercube topology has good properties as an overlay peer to peer network, the constant mobility and high failure rates of MANETs can result in excessive topology reconfiguration. Additionally, the assumption of global ontologies known by all nodes is different than the basic assumption of OntoMobil.

### 2.1.3 Autonomous Description and Semantic Resolution

The evolution to more expressive and decentralised P2P architectures has been complemented by increased heterogeneity in peer metadata [Nejdl et al., 2002, Löser et al., 2003b, Aberer et al., 2003, 2002]. Where formal metadata are used, it means that peers can describe their data or services using multiple, independently developed ontologies. Less control in content creation can increase peer autonomy and flexibility and broaden the range of content and services that can be provided by each peer. However, the departure from semantic homogeneity to more autonomous descriptions comes at a cost. Methods must be developed to integrate ontologies that were independently developed but describe the same domain.

Such integration is imperative as the idea behind the use of ontologies is to share and formalise knowledge. It is also complex because depending on the assumptions of heterogeneity, integration can happen at various points. Klein [2001] identifies two ways where ontologies can differ, namely at the *language* and the *ontology* level. The former describes mismatches at the language primitives that are used to specify an ontology, while the latter refers to differences in conceptualising similar domains of knowledge. This thesis assumes that nodes use the same ontology language but differences exists at the ontology level. In other words, it is assumed that all nodes share the same upper ontology.

In the context of decentralised discovery, heterogeneous ontologies present two separate issues. First, the type of *methods* that are used to bridge semantic mismatches and second the *network architectures* developed to facilitate semantic bridging.

### Methods for semantic integration

For what is ultimately a problem of codifying meaning in different ways, it is not surprising that a variety of techniques have been proposed to find similarities between different ontologies. One such technique is ontology mapping, which Klein [2001] defines as:

> Relating similar (according to some metric) concepts or relations from different sources to each other by an equivalence relation. A mapping results in a virtual integration.

Kalfoglou and Schorlemmer [2003] give a comprehensive survey for the field of ontology mapping. Apart from ontology mapping, similar techniques bear the names of ontology combination, merge, alignment and transformation [Klein, 2001]. OntoMobil uses the term *ontology matching* to denote both the identification of similarities between ontologies (mapping) and their augmentation with an appropriate relation. Augmentation does not involve alignment (i.e., changing two ontologies to bring them into agreement), rather it is just the insertion of hints at the linking points between different ontologies.

For the ad hoc environment, where OntoMobil operates, it is important for any ontology mapping mechanism to be automated, efficient and able to be executed at runtime. The requirement for automated mapping is necessary so that service discovery is not impeded by the manual resolution of different ontologies. Device characteristics also point to methods that should be able to trade performance for accuracy. Finally, both the periodic concept exchange and the dynamism of the network, call for methods that can be used at run time. Various practical classifications of automated and semi-automated methods to integrate ontologies are given in Klein [2001], Pinto et al. [1999], Noy [2004].

Two prominent approaches to integrate distributed ontologies are represented in the GLUE [Doan et al., 2002] and H-MATCH [Castano et al., 2003a] systems.

GLUE proposes a novel ontology matching process in three layers. At the bottom layer the *distribution estimator* takes as input taxonomies and their instances and computes a probability distribution of similarity and dissimilarity between all concepts. This is the core algorithm and employs machine learning techniques to identify similar concepts between the

input ontologies. The output is used as further input to the *similarity estimator* that establishes equivalence or generic/specific relationships between concepts from the multiple ontologies. Finally, the *relaxation labeler* layer uses common knowledge and domain constraints together with the input from the previous layer to derive appropriate concept mappings. The evaluation results show that GLUE is a very accurate matching algorithm. However, it is also a heavyweight algorithm. This is a function of requiring not only complete ontologies but also instance data in order for the machine learning part of GLUE to work effectively. In the case of OntoMobil, complete ontologies are not available as the protocol relies on the progressive dissemination of random ontology subsets. Instance data is also not available in the current specification, though they it be piggybacked within the description of transmitted concepts. Finally, the multilayer matching architecture is designed for accuracy rather than run time performance.

H-MATCH is a different approach to the problem of ontology matching. It uses four matching models, namely *surface*, *shallow*, *deep* and *intensive*. These models are configurable by a matching policy and use a variety of syntactic and semantic techniques. In particular, surface matching uses only the syntactic features of concept names for matching. Shallow matching is also syntactic, but takes into account concept names and concept properties. Deep matching uses a semantic matching technique that takes into account concept names and the whole context of concepts, i.e., potential relationships to other concepts. Finally, intensive matching is the most accurate matching technique and uses property values aside from the concept's context. Unfortunately, no comparative accuracy results exist between GLUE and H-MATCH. In terms of features, both systems are at par, though the configurable aspect of H-MATCH makes it a better candidate for open mobile environments. The idea of a customised set of matching algorithms is appealing because devices can choose to utilise parts depending on their performance characteristics. Generally, the more accurate a matching technique, the more expensive in terms of resources. Therefore, it is reasonable to customise matching.

OntoMobil defines two different matching methods, i.e., dynamic and template and uses the shallow H-MATCH matching model for the dynamic method. The purpose is to demon-

strate that an existing ontology matching algorithm can be adjusted to work with the concept matching approach proposed in this thesis.

### Network architectures for semantic integration

In a distributed environment, the integration of heterogeneous schemas from autonomous peers has a *temporal* and a *spatial* aspect. Semantic network architectures must resolve when and where (i.e., which nodes) the different peer metadata will be matched. These are important issues as they can affect the scalability and efficiency of the network and the accuracy and response time of content queries.

The temporal matching strategy can be further decomposed into reactive or proactive. A reactive strategy matches heterogeneous schemas only when needed, e.g., during the execution of a query, thus minimising processing cost at the expense of fewer responses. A proactive strategy will continuously match metadata as peers join or leave.

An example of a reactive method is given in Castano et al. [2003b]. The authors specify an architecture where matching occurs during the routing of semantic queries. A special type of query, called the *probe* query is used, containing a set of concepts from the source peer. When a destination receives such a probe, the query concepts are matched against the metadata maintained in the knowledge repository of the destination. The matching strategy can be set either by the source peer or at the destination.

Aberer et al. [2003] describes another reactive strategy. The authors assume that peers describe content without reference to a common schema. A semantic gossiping method is specified where peers progressively converge towards a global semantic agreement by exploiting transitive matching and feedback from other peers. This occurs during query propagation with no separate protocol to guarantee semantic agreement.

A proactive temporal strategy, where peers distribute metadata information before querying is used in Löser et al. [2003a]. Specifically, peers are organised into clusters that are based on the similarity of their semantic content. Before a new peer can query the network, it broadcasts its metadata amongst the super peers. A reply from a receiving host with similar semantic content enables the formation of semantic clusters.

Selecting a spatial matching strategy is equivalent to deciding which peers will act as mediators. Two prominent approaches are to either utilise super-peers or use the capabilities of every participant. Both approaches require the exchange of ontologies between peers that act as mediators. When super-peers are used, there is the potential for reduced communication overhead as exchange occurs only between a subset of available peers. On the other hand, super-peer selection in decentralised networks requires distributed mechanisms for leader election, which can add extra overhead. Additional factors that influence the design and overall performance of these topologies is the assumption about metadata distribution, e.g., the number of peers that are expected to contribute different metadata, and the specific communication protocols that will be used.

The super-peer method is described in Löser et al. [2003a]. Only a subset of peers are selected to serve as decision points for semantic similarity. The aim is to reduce communication overhead by having a limited broadcast only across super-peers. It also simplifies the process of arriving to a consensus on semantic similarity, by having only the designated peers match schemas. The disadvantage of such an approach is that it places a burden on the elected super-peers. Moreover, the protocols required to elect super-peers, especially in the face of failures and high peer churning can be complex.

A different approach is implied by Castano et al. [2003b], Aberer et al. [2003] and Aberer et al. [2002]. In systems discussed by these researchers, semantic matching extends to all peers. Each host is responsible for playing a part in the emergent semantic knowledge derived through peer interaction.

In terms of the temporal matching strategy, OntoMobil uses the periodic execution of the gossip protocol to exchange concepts, which results in a proactive strategy. This leads to a progressive semantic matching idea that is similar to the one found in Aberer et al. [2003], though the two methods of arriving to network-wide semantic consensus are very different. In terms of the spatial matching strategy, OntoMobil uses the "all peer" approach to alleviate the need for an extra super-peer formation protocol and to take advantage of concept replication in each node.

### 2.1.4 Issues in MANETs

As mentioned in Section 1.2, despite similarities between peer-to-peer and mobile ad hoc networks, there are a number of important differences. In an environment where mobile nodes develop services without a common schema, ontology matching and discovery must be robust to node mobility, consider the processing capacity of nodes and adapt to the variation in the size of the network.

In particular, mobility and the reduced processing capabilities of participating devices tend to favour the distribution of heavyweight tasks across nodes. Were such tasks not distributed but executed in elected nodes, the high rate of partitions, merges and link failures would require complex fault-tolerant procedures when those nodes became disconnected or failed. Though clustering algorithms are becoming increasingly sophisticated and efficient [Li et al., 2006], they are usually applied to tasks that do not require high processing cost, e.g., Multi-Point Relays (MPR), which maintain routing tables in the Optimised Link State Routing (OLSR) protocol [Clausen et al., 2001].

Unlike the maintenance of routing tables, ontology matching is a process that can be computationally intensive. It is appealing to centralise such a process especially when the process outcome changes infrequently. For example, in a stable network without new node additions or failures, ontology matching can be executed once and semantic consensus can be derived easily. However, in the ad hoc environments examined in this thesis, the transient connectivity of nodes means that semantic consensus is ever changing and evolving. As new nodes join, new ontologies need to be matched against existing ones, while as connected nodes disconnect or fail, their ontologies must be removed from the network. It is a process that is continuous, terminating only in periods of relative network stability. As such, distributing the process of ontology matching across mobile nodes is a design choice that must be given serious consideration.

Another issue arises with discovery scope and network scale. Certain multicast-based LAN discovery protocols, e.g., SLP and Jini, restrict the propagation of discovery queries by applying filters. This restriction of the propagation is mainly an administrative tool to increase security and limit the use of multicast, which can create significant communication

overhead. In P2P networks, scoping is not used because of administrative requirements but as a method to enhance scalability. These approaches restrict query propagation to peers with similar metadata because of an implicit assumption that peers will query for content based on their own semantic description. P2P networks that use semantic clustering to improve efficiency make this assumption.

In environments where interaction is transient and opportunistic, semantic clustering may not offer similar gains as in large-scale P2P networks. Discovery queries in these setting are best effort and can only get a reply if a suitable provider is currently connected. To maximise the probability for a response it is important to utilise as many providers as possible, while allowing the network to scale to a sufficient degree.

The next section presents a set of criteria to classify mobile service specifications. These criteria are designed to bring forward the various ways with which current service architectures and protocols handle mobility, scale and efficient discovery.

## 2.2 Classification Criteria for Mobile Services

This section reviews services that are mobile-aware and are designed mainly for the pervasive application domain. A number of the specifications vary in their stated aims and system assumptions, e.g., mobility, availability of multicast, etc., but all use a degree of self-organisation to adapt to a decentralised environment and incorporate some management of failures and partitions to reduce the effects of mobility. To aid understanding and compare the various specifications, a classification is provided according to the criteria illustrated in Fig. 2.2.

The purpose of this classification is to decompose important aspects of the systems under review. In comparison to the service discovery survey by Zhu et al. [2005], we focus on categorising common overlay topology structures, assessing the expressive power of the examined services and qualifying the efficiency of service maintenance in a distributed setting.

Services are qualified according to the following criteria:

**Service representation** – The representation of services, especially in mobile systems, must balance efficiency of implementation with expressiveness. Expressive service descrip-

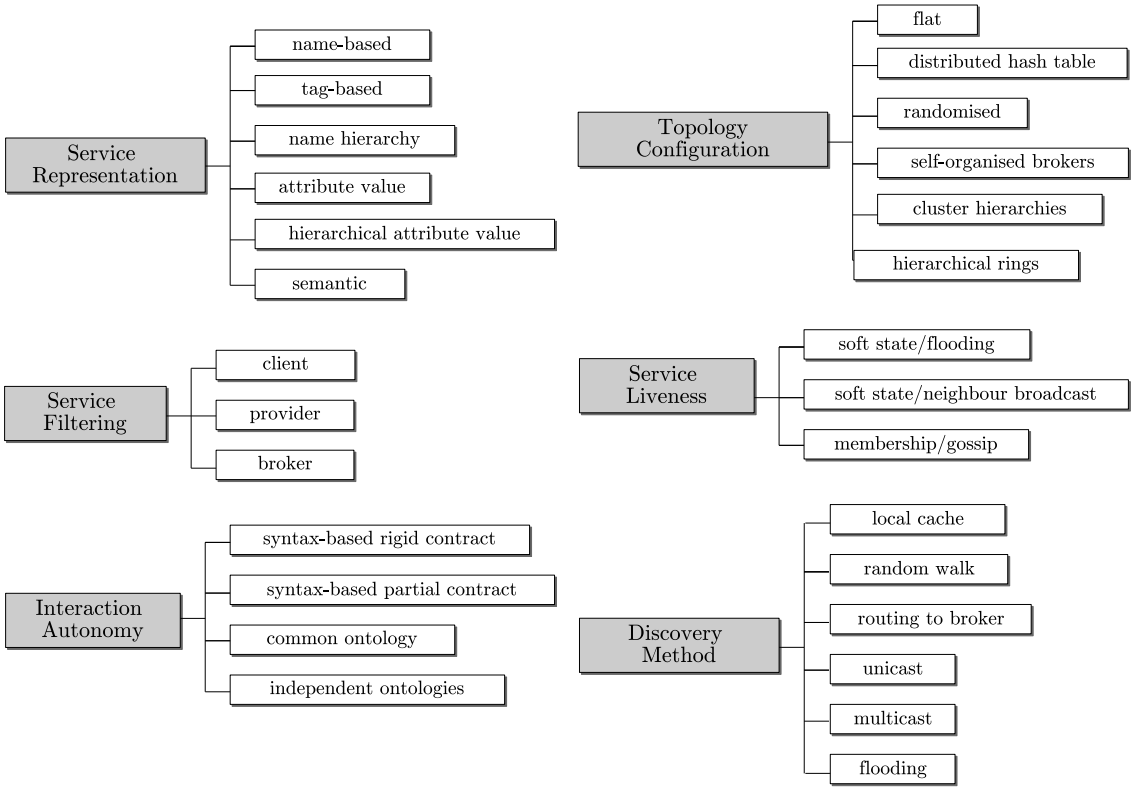


**Fig. 2.2**: Classification criteria for mobile service protocols and architectures.

tions enable richer encoding of functional and non-functional service characteristics. Additionally, service requests can more accurately capture user requirements. However, more complex and formal descriptions are associated with a higher cost in modelling and processing services. Where an overlay topology is needed, the choice of representation is also a significant factor. For example, INS [Adjie-Winoto et al., 1999] has a topology that is very closely linked with its hierarchical attribute value naming syntax. The classification distinguishes six approaches to service description, in ascending order of expressiveness: The use of a service name accompanied by type parameters, which is frequently used to encapsulate implementation details, e.g., execution method; tag-based service description, where services are annotated with a set of names; hierarchical naming conventions, where services are described through a sequence of categories, much like files are qualified by directory names in modern operating systems; attribute value pairs, where arbitrary pairs of properties are used

for description; hierarchical attribute values, where a hierarchy is imposed on the properties and finally semantic descriptions, where an ontology is used for representation.

**Service filtering** – A large number of the service systems assume an environment where many providers offer a multitude of services. Queries in these environments can be generalised to return a large number of services. Whether all of the matched services are returned and the location where the selection process takes places can affect network efficiency. The selection of a node with the responsibility to filter matched services is important in large networks with hundreds of services. In large networks the aggregation and ranking of services has significant processing and communication overhead.

Depending on some general architectural patterns, a number of choices for service filtering emerge. For example, in architectures where all descriptions are cached at broker nodes, it is natural for selections to happen there. Where a distributed architecture is in place, extra communication might be required to gather all the matched services and return them to the client node. In a homogeneous environment, where nodes are not allowed to customise and extend the service middleware, filtering can take place at any node. If custom filters can be supplied at the client side, e.g., QoS or context filters, and their evaluation depends on custom software, aggregation and filtering at the client might be inevitable. Service filtering is not being adequately addressed at the moment, but will become an issue as service architectures scale up their numbers of providers and consumers. This classification has identified the choice of service filtering to be either at client, provider or broker nodes.

**Interaction autonomy** – In the context of service discovery, interaction describes a query-select-respond pattern, while autonomy characterises the level of commitment to a common semantic interpretation of a service representation. In other words, this criterion reveals the implied contract between a provider and a client regarding the service's capabilities. It can be argued that a name-based service query from a node with no prior knowledge of any mapping between the service's functionality and the service's name is an autonomous interaction. It is unlikely however, that such a discovery query will produce any meaningful expected outcomes. Autonomy becomes more useful as the service representation becomes more expressive. When the description of a service allows queries to be expressed in various

combinations, flexibility and hence autonomy is increased.

The following cases are distinguished: In a rigid contract, a service is expressed in syntactic terms with the client knowing a priori what functionality maps to which service. One way to achieve automated query resolution in this context is the standardisation of services. For example, Jini, SLP and UPnP operate in this manner. Partial contract is a variation of the previous case that offers increased flexibility by allowing clients to invoke services based on partial knowledge. For example, support for regular expressions or attribute only queries can provide a less strict and thereby more autonomous interaction.

Semantic modelling also alleviates some of the requirements for a priori knowledge between a service name and its functionality. By querying for the actual service functionality, any implicit knowledge becomes part of the service description. Finally, opportunistic interaction with minimal a priori knowledge between providers and clients can be achieved with the use of different ontologies to describe offered and requested services.

**Topology configuration** – As networks grow, the use of an overlay topology to organise services is a more efficient approach than the use of multicast or broadcast. Section 2.1.2 presented various schema-based P2P topologies, where semantic similarity is reflected in the network topology. This is one possible approach to topology configuration. Often, the choice of topology is a function of providing some desired properties, e.g., certain QoS guarantees or bounded service discovery. Other times, it is shaped by the constraints of the environment, e.g., availability of multicast features at the network layers or use of a certain service representation. For mobile ad hoc networks, some key properties are efficient topology construction, adaptation to mobility and where brokers are in use, timely broker election. Regarding brokers, this classification does not interpret an overlay topology from the presence of brokers only. The rationale is that it is challenging to define precisely what a broker is and how it operates. For example, there are topologies where every provider is a broker (OntoMobil, GSD), brokers and providers partly overlap [Kozat and Tassiulas, 2004] and providers and brokers are completely independent [Adjie-Winoto et al., 1999, Sailhan and Issarny, 2005]. Therefore, topologies are classified based on how providers, clients and brokers are organised and not purely on the existence of brokers. This topology criterion has the following con-

figurations: Flat is the simplest configuration and entails no special structure; distributed hash table entails nodes that are organised in such a topology; randomised topology of which OntoMobil is the only example, assumes a topology where each node maintains contact with a random subset of other nodes; self-organised brokers, where a localised algorithm is executed to elect new brokers; cluster hierarchies where nodes are first clustered according to certain criteria and subsequently organised into a hierarchy to improve inter-cluster access; and finally, the hierarchical rings structure is one where nodes are organised into a multilayer ring topology.

**Service liveness** – In a mobile environment, new nodes constantly join and existing nodes may fail or become separated from the network. If caching is used to replicate service descriptions, a mechanism must be devised to cache new services descriptions when they appear and to identify and if necessary remove stale services. Similar mechanisms can be used to reconfigure the network topology by monitoring variability in the network and the usage patterns of services. The two issues of topology maintenance and service liveness are closely intertwined in most service architectures. They are treated here as a single classification criterion, which gives a qualitative metric on communication overhead, protocol complexity and accuracy of available information. The template of <*liveness signal*/*transmission method*> is used to express the temporal aspect and the form of communication that is used by each node to update other nodes with current information. One of the most common methods to signal status information is the use of soft state. Each node periodically transmits a beacon message that includes some basic information, e.g., the node identifier and a sequence number. This is a simple and effective solution but depends on timing intervals which can be hard to predict in a mobile ad hoc setting. A more complex but adaptable solution requires the use of group membership information, which provides a notification service for new nodes that join the network and existing nodes that have failed. How this information is transmitted to other nodes is effectively orthogonal. Solutions based on flooding, neighbour broadcast messages or gossip have appeared in different systems.

**Discovery method** – Discovery method details the protocol used for service queries. There is an obvious dependency between discovery and both the chosen topology configura-

tion and the underlying network support. Systems that rely heavily on service advertisement and broker support have a simpler discovery protocol. Systems that try to minimise advertisements usually require a more elaborate discovery protocol. The discovery criterion distinguishes five methods: local cache, where clients query their own cache for service identifiers; random walk, where clients query a random sequence of nodes; routing to broker, which implies a broker architecture with nodes storing routing information to locate the brokers and forward queries to them; and finally unicast, multicast and flooding, which can be used in conjunction with an overlay topology or depend on protocols at the lower protocol layers.

## 2.3 Review of Existing Service Systems

Here, the classification criteria from Section 2.2 are applied to the most pertinent service architectures and discovery protocols designed for mobile environments. Since, some of the covered systems have only a partial description of some of the properties, whenever one is not specified, it is explicitly annotated with n/a (not available) in Table 2.1. In the next sections (2.3.1 and 2.3.1) the focus is mainly on academic prototypes where a degree of mobility and self-organisation is implied. Industry standards are briefly covered in Section 2.3.3.

### 2.3.1 Service Systems with Partial Mobility

The following two service architectures are complete specifications supporting mobility of providers and clients within an infrastructure backbone.

#### INS/Twine

Balazinska et al. [2002] describe a scalable and distributed architecture for resource discovery. INS/Twine is an evolution from the original INS [Adjie-Winoto et al., 1999] project, but targeted towards resource discovery, rather than towards a distributed naming facility. The architecture assumes a core network of brokers that replicate resource descriptions, while resource providers are placed at the edges of the network. Resources can represent both

static content, i.e., documents and data but also interactive one, i.e., services. Both INS and INS/Twine rely on the use of hierarchical attribute value pairs to express resource parameters. The use of hierarchies allows queries to be more expressive by customising the level of generality and specificity when requesting information. INS/Twine changes the original INS in several ways. The underlying topology now depends on Chord [Stoica et al., 2001], rather than being a custom one. In terms of mobility, INS adapts by using only the *intentional name* (i.e., the structured name) of a resource. This level of indirection can support both *node mobility* and *service mobility*. Node mobility denotes physical mobility, occurring when a node changes network addresses (e.g., as it crosses between administrative boundaries), while service mobility signifies a change in the mapping between resources and their providers (e.g., when network or device conditions change). Similarly, OntoMobil services are also located using their constituent concepts so both types of mobility can be supported.

In terms of resource representation, INS/Twine uses the more expressive hierarchical form of attribute value pairs. Although the semantics of the representation are not standardised, it is expected that descriptions and queries will use similar representations for similar resources.

Service filtering is customisable in INS/Twine, with the query specifying whether one or all replies will be sent back to the client. If a query specifies that a single reply should be transmitted, a user-supplied metric acting as the ranking coefficient must also be supplied. In that respect INS/Twine supports both client and broker filtering.

Using a hierarchy of attribute value pairs and allowing queries to express only a subset of the attributes that describe resources increases the autonomy of peers and provides a more flexible contract between resource providers and clients.

INS/Twine uses brokers, called *resolvers* in INS terminology, to advertise resource descriptions. The resolvers form a distributed hash table topology that is designed to share load and bound the path of discovery queries. To remove stale resources and refresh existing ones, it uses an adaptive soft state approach. Between resolvers the refresh rate is low, so bandwidth is not saturated from liveness messages. On the other hand, to cater for responsiveness in the event of a failed resource provider, the refresh rate between providers and their resolvers is higher than that between resolvers. For discovery, INS/Twine uses unicast

over the routing infrastructure provided by the Chord substrate. INS/Twine is a mature and efficient design, but it is not targeted towards the mobile ad hoc domain as defined in this thesis, i.e., a network with autonomous nodes, absence of a priori agreement and equality of node responsibilities. Furthermore, the reliance of INS/Twine on the Chord distributed hash table means that a MANET implementation would also require a Chord-like system designed for MANETs.

### VIA

VIA [Castro et al., 2001] is another discovery architecture aimed at locating data across service domains. A basic assumption in VIA is the asymmetric nature of communication. It is envisioned that pervasive environments will have a large number of small, low-powered devices connected to different administrative domains. Brokers, *gateways* in VIA terminology, will be used to store most of the data obtained from the multitude of devices. The problem that VIA addresses is an architecture for global access to all the data and services between different brokers. To this end, it considers an overlay network based on the principles of robustness, self-organisation, scalability and autonomy. Contrary to approaches such as INS, VIA does not replicate metadata descriptions. Instead, it considers a model whereby data is clustered according to location. This makes semantically related data distributed across brokers since it is designed to increase access time within domains. Ultimately, VIA relies on a self-organising topology that adapts to queries by having gateways form hierarchies according to a localised algorithm and not according to some predefined policy.

VIA uses a flat attribute value naming scheme to describe any type of resource. Although VIA supports a generalisation/specialisation relation between attribute value pairs, it is not based on inheritance hierarchies. Rather, it is based on the presence or absence of values from attributes. For example, an attribute with no value is considered more general than an attribute value pair.

Similar to INS, VIA allows requests to contain partial data descriptions and also permits the composition of more complex queries using the logical operators of “and” and “or”. This is a more flexible approach than requiring a strict interface between clients and providers.

The topology configuration in VIA is a set of adaptable, self-organised cluster hierarchies. Any peer that becomes the root of such a cluster hierarchy requires the use of multicast facilities to communicate with other root peers. Multicast is also required for the communication between clients and root peers.

Service liveness is not specifically addressed in VIA. Reconfiguration of VIA hierarchies occurs either as a result of failures or due to changing query patterns. A soft state approach is used to identify failures by periodically advertising the broker's status. To reduce the overhead associated with maintaining state for a large numbers of brokers, status information is only transmitted between parent brokers and their immediate children. Finally, discovery is conducted by multicasting to all peers that are cluster roots followed by a unicast inside each cluster.

VIA has a deep-rooted assumption of pervasive fixed infrastructure, which permeates many aspects of its design. The resulting architectural complexity is intended to deal with levels of scalability not envisioned in the OntoMobil scenario of standalone ad hoc networks.

### 2.3.2 Service Discovery in MANETs

This section reviews eight service discovery specifications specifically for the domain of mobile ad hoc networks. These specifications offer various degrees of completeness. Some specify just the discovery protocol, while others specify a more complete system architecture including the service representation and topology construction.

#### DEAPspace

DEAPspace [Hermann et al., 2000, Nidd, 2001] is one of the first service architectures to target transiently connected mobile devices. DEAPspace aims to devise a protocol for low latency service discovery and to represent services in a compact and efficient manner. To further qualify the timeliness requirements for discovery, the authors analyse the available time period for two mobile entities as they cross moving at opposite directions under different mobility scenarios. This represents the window of interaction that a service protocol can exploit. The resulting time periods become an upper bound in the overall design. DEAPspace

is concerned mainly with networks that span a single hop and as such service advertisements are aggressively used. Each advertisement is associated with a timestamp and provides both liveness information and simplified discovery. Discovery is conducted without transmission, using only the device's local repository.

The service representation in DEAPspace is a custom specification designed to satisfy fine-grained service selection, small message size and parsing simplicity. Service selection is based primarily on the service name, although the actual representation contains some mandatory and an arbitrary number of optional attributes.

Since discovery is local to the cache maintained by each device, service filtering is client specific. Interaction autonomy is rigid, since selection is primarily based on the name and the input–output service parameters with any optional attributes being secondary selection criteria. DEAPspace does not use an overlay topology, instead advertising relies on the periodic use of flooding. This limits the scalability and the efficiency of the scheme, though it is a simple and practical solution for the type of scenarios envisioned by the authors.

#### Konark

Konark [Helal et al., 2002] is a service middleware for small scale mobile ad hoc networks. It supports service discovery and service advertisement and assumes that each node is both a provider and a client. Konark is a practical design using a WSDL-like service description and a micro-http server to process SOAP-encoded service requests and advertisements. The Konark specification is not suggestive towards a specific approach related to discovery or representation. Instead, a number of methods are discussed and analysed.

Services in Konark are described in a variety of ways. There is a hierarchical naming structure where generic concepts appear at the top of the tree with more specific ones at the bottom. Services, both those provided by the node and those acquired from advertisements, occupy the leaf spots in the tree. Additionally, services can optionally be associated with a set of tags.

The discovery protocol suggests that service selection takes place solely at the client. Interaction autonomy is rigid when discovery queries specify a service as a path in the Konark

tree. It is assumed that concepts in the tree will require agreement before being used. Tag-based discovery queries provide a more relaxed selection criteria, though their free form syntax makes it likely that precision will be low with a high recall. Konark uses a soft state approach to prevent stale services from being used. It also uses a specialised multicast address for discovery and advertisement, which can limit scalability. Konark is targeted towards an implementation of existing service technologies and as such it does not provide a novel model or an analysis of its various design trade-offs.

**Kozat and Tassiulas**

In Kozat and Tassiulas [2004] and Kozat and Tassiulas [2003] a distributed discovery mechanism for mobile ad hoc networks is presented. Unlike other approaches, it is broker-based but operates very close to the routing layer. The authors argue that using service brokers close to the routing layer can enhance scalability, improve efficiency and reduce discovery latency. Two separate algorithms comprise the system. A backbone management (BBM) algorithm forms a mesh of service brokers using the idea of a dominating set. This results in the localised construction and maintenance of the backbone substrate. Such a scheme has the desirable property that each non-broker node is one hop away from at least one broker. A distributed service discovery (DSD) algorithm uses an additional source-based multicast protocol across the brokers to replicate service descriptions during service registration and increase reliability and access during discovery. A very extensive evaluation is provided comparing the system with other approaches based on multicast and anycast. The results illustrate that despite the high communication overhead, a clustering scheme can be a viable option for service discovery in MANETs.

There is no suggestion as to what type of services can be supported in such a system as the focus is on the specification of the discovery protocol and its performance evaluation. It is mentioned that the frequent broker hand-offs required as the network topology changes and nodes fail, makes it unlikely that costly registration procedures can be supported. Similarly, there is little mention of service filtering, though the use of multicast for discovery suggests that such filtering can only occur at the client side. The topology configuration is a self-

organised mesh with only neighbour broadcast messages being used to maintain the topology.

The use of neighbour broadcast by the BBM algorithm is less prone to message loss and can potentially reduce the aggregate network overhead when compared with the multi-hop gossip unicast used in OntoMobil. However, the two overlay networks serve different functionality making a direct comparison difficult. If the overlay by Kozat and Tassiulas [2004] were to be used to fulfil the requirements in Section 1.2, the broker nodes would have to bear the sole responsibility for concept matching. This would increase their processing overhead and would also require the specialised hand-off procedure to transfer the matching relations between departing and new broker nodes. In case of high mobility, the hand-off procedure and the constant swapping of nodes between brokers and non-brokers would create a lot of unneeded traffic.

#### Sailhan and Issarny

Sailhan and Issarny [2005] describe a scalable service architecture for mobile ad hoc and hybrid networks. The architecture is composed of a service representation and a suite of protocols for advertisement and service discovery. The motivation behind the protocol suite is efficiency and reduced energy consumption as measured by the total number of transmitted messages. These factors have led to a design that incorporates a set of connected broker nodes forming an overlay network, called a *virtual network*. Nodes become directory brokers through a localised protocol that elects those nodes willing to contribute their resources and have the highest number of neighbours in their coverage area. Contrary to Kozat and Tassiulas [2004], overlay formation is not formally specified to create a connected dominating set, but has rather been devised for simplicity and efficiency. Scalability is achieved as follows: First, brokers cache only a subset of the available service descriptions. Specifically, each broker maintains services from providers that are within a configurable number of hops away. Second, the cached services are encoded in a compact directory profile using a novel scheme based on Bloom filters. To enable network-wide service discovery, brokers exchange their directory profiles at frequent intervals.

Services are represented in a WSDL-like, syntax-based interface that is augmented with

a set of attribute value pairs. These pairs are used to describe QoS and resource specific attributes. Primarily however, services are expressed through a name as the attribute value pairs are used for efficient selection and load balancing.

The discovery protocol achieves the stated goals of efficiency and scalability by incorporating load balancing during service requests. As service providers augment service descriptions with resource specific information, brokers can decide whether a particular provider is overloaded or low on battery and route a query to another node that provides the same service. A pricing-based scheme is used at the broker to evaluate the registered providers. Hence, selection filtering is broker-based.

The topology is a self-organised broker configuration. Each broker is responsible for advertising its presence within a configurable number of hops and providers and clients are responsible for maintaining the address of at least one broker node. This requires all nodes to keep some partial state, which can reduce robustness in the face of mobility. It also requires constrained flooding to reach all nodes in the vicinity of the broker, which is optimised by the authors with the use of bordercasting.

A number of methods are used in the effort to maintain accurate service and topology information in the face of mobility. Service providers use an adaptive soft-state mechanism to update their descriptions. Brokers frequently bordercast an advertisement to notify nodes of their presence. Brokers also require communication with other brokers in order to exchange directory profiles. An evaluation of the protocol that takes into account the extra overhead shows that it still bears a reduction over flooding. Finally, discovery is initiated with a simple unicast from a client to its local broker. In the case where the requested service is not found, the broker node forwards the query to a number of other brokers, which are selected based on battery life availability.

The core differences between the OntoMobil overlay and the discovery architecture by Sailhan and Issarny [2005], reside in the representation of services, the contents of the overlay network and the use of broker nodes. Because OntoMobil uses of ontologies to describe services, the overlay is constructed from metadata, rather than services. Furthermore, in OntoMobil every node is considered to be part of the overlay, which simplifies the design by

not requiring a special protocol for node to broker communication.

**Lenders et al.**

Lenders et al. [2005] presents a field theoretic model for service discovery. The idea is to map a physics concept, the potential resulting from electrical point charges, to the domain of mobile ad hoc networks. Specifically, the distribution of potential that peaks at the point charges is mapped to a *scalar field* where nodes in the network calculate the potential distribution with service providers being the point charges. A service request is then routed towards the provider with the highest charge. Two metrics are used to select service providers that also influence the routing of service requests; the number of hops between clients and providers and the *capacity of service* (CoS), a set of attribute value pairs indicating functional and non-functional service properties. Through flooding, each provider disseminates its service instances (charges) to all nodes, so that each node can calculate its own potential for all received charges. An emerging property of this model is that requests tend to gravitate towards providers that are closer to the client's proximity. This increases local communication reducing overall overhead. Simulation results demonstrate a good discovery ratio even under high mobility scenarios.

Services are represented as a set of attribute value pairs. It is expected that all nodes share a common interpretation of these pairs. This work takes an interesting perspective on service filtering. Each request is only ever routed to a single service provider. Assuming that each provider advertises unique service instances, filtering is progressive and a byproduct of routing a request to the highest service charge. In terms of interaction autonomy, the design does not mention the possibility for service requests being based on generalised or partial queries.

No special overlay structure is used to organise the nodes in the network. Flooding is used for charge dissemination, though to counter the expected overhead an optimisation is proposed. Flooding works in coordination with a soft state approach, which is used to keep cached service references up-to-date.

With cached references in every node, each discovery is routed to the provider with the

strongest point charge following a sequence of successive unicasts to neighbouring nodes. This application level routing is similar to OntoMobil, though there are two differences: First, OntoMobil does not use neighbour unicasts to route a discovery, but a random walk approach based on multi-hop unicasts making is less robust and more prone to failures; second the selection of the next hop node in OntoMobil is not based on finding the highest potential amongst neighbours, but is based on a random selection.

### GSD

The Group-based Service Discovery (GSD) protocol [Chakraborty et al., 2002] uses a caching mechanism to improve access times and reduce overhead of semantic service discovery in ad hoc networks. The caching mechanism works by employing an ontology-based classification for all services. Specifically, each service is classified into a hierarchy of service groups, which are defined in a standard ontology. A provider uses constrained flooding to advertise both the service description and its service groups. Both are cached in receiving nodes. When a node that has a non-empty cache wishes to advertise its own services, it also piggybacks some of the service groups found in its cache. This has the effect that nodes cache both exact services and ontology concepts. A request is also classified into service groups and is directed to a provider either following a cached description or through a cached classification. GSD is one of the first discovery protocols that used specific features of semantic services in a mobile ad hoc setting. Simulation results illustrate that a high discovery ratio is maintained while communication overhead is reduced compared to a pure flooding approach. The use of cached service concepts to aid discovery in GSD resembles that of OntoMobil. OntoMobil does not use flooding however and relies on a unicast gossip protocol for concept dissemination. This gives OntoMobil better scalability properties and creates less overhead for nodes that do not participate in the exchange of services.

GSD uses a common ontology to represent services. A service matching approach is also sketched, identifying service groups, input–output parameters and service capabilities as potential matchmaking properties. This can prove suboptimal when the selective forwarding strategy is different than the matchmaking one. For example, as outlined in the design of the

forwarding strategy, service groups are sufficient to route a service to a provider. This does not guarantee a matched service however, as the query signature might still be different than the provided one. OntoMobil addresses this concern by routing requests according to their full semantic signature.

No particular topology configuration is used in GSD. Each connected node participates by caching partial service information. Service liveness relies on a soft state algorithm that can be customised by users depending on the mobility of the network. This information is disseminated through a constrained flooding algorithm. Discovery is symmetric to advertisement and also uses a constrained flooding approach in order to discover either the provider directly or a cached copy of the description or the classification of the service.

#### Service rings

Klein et al. [2003] present a semantic overlay for service discovery in ad hoc networks. The aim is to construct an application overlay that is decentralised, efficient and allows discovery of semantic services. The overlay is based on the idea of a hierarchical ring structure. Each node is placed in a ring and is required to store the identifiers of the two nodes that follow and precede it. Several rings can co-exist in a large ad hoc network since they are formed by exploiting geographical or semantic proximity. Each ring must elect a broker, called the *service access point* (SAP), though the election process is unspecified. A broker is responsible for indexing any service advertisements that arrive from its ring and seeks to participate in higher order rings between other brokers. Communication in service rings, i.e., advertisement and discovery, happens only through the forward and backward links that each node maintains. Although this avoids flooding, it is unclear what the precise gain is, since no evaluation is presented and overlay links in the ring structure do not adapt with mobility. A number of other algorithms are presented mainly to cope with node and link failures and network partitions. Judging from the number of special cases and protocol complexity, there is a high cost associated with maintaining such a rigid structure in the face of mobility.

The service rings specification uses a semantic service representation. It exploits the classification properties inherent in ontologies, so that each service request yields a distance

metric when compared against the cached service descriptions in the brokers of each ring hierarchy. If the semantic distance is less than some user defined threshold, the advertised service constitutes a match.

Filtering takes place at the client side. Each request stores any matching services as it is routed through the different layers of the ring hierarchy, eventually returning to the client node.

The specification does not cover the timing of the cache updates at brokers. It only specifies how clients register their services with the appropriate broker using a service advertisement algorithm. Topology reconfiguration occurs whenever there is a partition or a failed node. Such disconnections are identified using a soft state mechanism.

Discovery uses a unicast transmission across the ring overlay. No analytical or simulation results are offered as to the expected discovery bounds.

#### Allia

Allia [Ratsimor et al., 2002] is a peer-to-peer, policy-based, distributed architecture to support agent-based discovery in mobile ad hoc networks. The aim is to reconcile an agent-oriented abstraction with a distributed service discovery model. To this extent it assumes a "pure" peer model, where no brokers are necessary and nodes are self-sufficient, hosting a complete agent micro platform. This is similar to the assumptions made in OntoMobil. Another similarity is the use of push style advertisements and the use of caching in each node. Allia however, relies on a per node policy-based manager for most decisions regarding advertisement, discovery and caching. This provides flexibility since each node can decide an individual policy that is adapted to its resources. However, when each device can alter parameters like advertisement frequency, cache policies and forwarding strategies, it makes it challenging to assess and analyse the aggregate network behaviour.

Allia focuses on general architectural issues for policy-based peer discovery models and as such it does not prescribe a service representation. An implementation of the architecture over Bluetooth, uses Bluetooth's SDP protocol, which means that services are represented with a universally unique identifier (UUID). The service cache is used at all times, making it

easy to discover and filter services at the client side. In terms of service interaction autonomy, clients and providers follow a strict contract since the implementation assumes that all nodes maintain a mapping from service identifiers to actual service descriptions.

There is no topology structure in Allia, though the architecture elaborates on the issue of service and node liveness. A soft state and a constrained flooding approach is used in both cases. Ultimately however, it is up to the policy manager to decide the frequency and even the flooding strategy (e.g., whether multicast or broadcast will be used). Discovery is a combination of a local cache search and flooding the request in case the local cache gives a negative match.

### 2.3.3 Other Related Systems

The previous section detailed novel models for service discovery in the ad hoc and mobile-infrastructure domains. This section covers two industry-led efforts for service discovery and specification and one research prototype focused on service security. The focus in these systems is not analytical properties or service expressiveness, but rather a firm focus on usable deployments, application development and integration with existing platforms.

#### SLP

The service location protocol (SLP) [Guttman et al., 1999] was one of the first complete service specifications for local area networks. It defines service discovery and advertisement in terms of three agents, the user agent (UA), service agent (SA) and directory agent (DA). Because it is destined for administrative networks, there is little self-organisation and directory agents have to be explicitly configured. It does however prescribe a directory-less mode, where IP multicast can be used for discovery. In SLP, services are represented with a URL and a set of attribute values. The Internet Engineering Task Force (IETF) has released several draft specifications of the protocol and an abstract service specification that can be used as a template for a variety of concrete services, e.g., printers, device drivers. One interesting extension to SLP, which has appeared recently [Authority, 2003], is the mesh-enhanced SLP (mSLP) [Zhao et al., 2003]. This SLP enhancement increases reliability of service registrations

by organising directory agents in mesh overlay.

### Jini

The Jini specification [Arnold et al., 1999] describes a platform-independent distributed architecture and an associated programming model for the discovery and use of a wide range of services. Typical examples include the acquisition of drivers during device boot up, discovery of public key servers and other types of system or user-defined services. At the core of jini is the idea of a *lookup* service that is offered by a broker. Clients or providers that wish to discover or register services need to initially access the lookup service. The sequence of actions for discovery and registration are detailed in three protocols, namely:

- *discovery* is the initial process invoked by any provider or client in order to locate a lookup service. Within discovery, a suite of protocols are responsible for locating the lookup service. Two of the protocols use multicast, while the third uses reliable unicast when the address of the lookup service is known in advance.

- *join* is used by providers to register their own services, after they have located a lookup service. It specifies the interaction pattern between providers and the lookup service and characterises service lease timeouts and fail-over scenarios. During join, providers submit a Java object implementing the offered service. The interaction between providers and the lookup service takes place through a proxy object that is downloaded by each provider once the lookup service has been located.

- *lookup* is used by clients after they have located a lookup service in order to find services. The interaction pattern is also encapsulated inside a Java class (*LookupLocator*) defined by the Jini specification to ease development and reduce the probability of errors.

### SDS

OntoMobil follows the majority of specifications in mobile ad hoc networks in assuming a model where service providers and clients can be trusted. The assumption is that providers advertise an accurate description of their services without "cheating" and clients do not

misbehave by overloading the network with service requests or adversely manipulating their service cache. This is not a realistic assumption in many cases. The Ninja Service Discovery Service (SDS) [Czerwinski et al., 1999] is an architecture for secure and trustworthy access to services. A hybrid of asymmetric and symmetric-key encryption offers privacy, while trust is provided by endpoint authentication and capability-based service disclosure, which is a mechanism to reveal advertisements only to authorised clients. Since one of the stated goals of the Ninja architecture is usage in Wide Area Networks (WAN), scalability and security are explicit design requirements. The architecture consists of five components:

- *SDS servers* are brokers organised in a hierarchical topology and are used to cache encrypted service descriptions and answer client queries.
- *Services* initially contact a Capability Manager (CM) to define the access rights for clients. Subsequently they listen for announcements from SDS servers, select an appropriate server and advertise their descriptions using authenticated, encrypted one-way service broadcasts.
- *Certificate Authority* is a trusted peer that uses certificates to authenticate the binding between services and their public keys.
- *Capability Manager* is another trusted peer that accepts requests from services declaring their capability lists and then distributes these capabilities into requesting clients.
- *Clients* initially discover an appropriate SDS server by listening on a well-known multicast channel. After binding to such a server, they can forward their queries and let the SDS server resolve any pending authorisation and capability requirements.

## 2.4 Summary

This chapter has reviewed several service specifications from the peer- to-peer and mobile ad hoc domains. A summary of related work in the field of ontology and schema matching has also been presented in order to elucidate the requirement for a topology semantic integration in mobile ad hoc networks. Many of the high-level design choices that service specifications

for MANETs must resolve were captured in six classification criteria. A summary between the reviewed systems and OntoMobil based on these criteria is presented in Table 2.1 on the following page.

The related work survey has illustrated the limitations of existing MANET topologies to support ontology matching and discovery of autonomous semantic services. This has led to the development of the proposed model and the randomised overlay topology. OntoMobil avoids clustering topologies, first to remove the need for fragile and complex cluster formation protocols and second to reduce the performance bottleneck of ontology matching on cluster nodes. A third argument is related to the appeal of using a unified approach without requiring different protocols for inter and intra-cluster communication. The next chapter presents the design, specification and analysis of the OntoMobil gossip protocol.

### 2.4.1 Comparison of Mobile Ad Hoc Services

Table 2.1 on the next page provides a summary between of the reviewed systems according to the classification criteria specified in Section 2.2.

| | **Represent.** | **Filtering** | **Autonomy** | **Topology** | **Liveness** | **Discovery** |
|---|---|---|---|---|---|---|
| INS/Twine | hierarchical attribute value | client and broker | partial contract | distributed hash table | adaptive soft state / unicast | overlay unicast |
| VIA | attribute value | n/a | partial | cluster hierarchies | soft state / constrained unicast | overlay multicast and unicast |
| DEAPspace | name-based | client | rigid | flat | soft state / flooding | local cache |
| Konark | tag-based and name hierarchy | client | rigid and partial contract | flat | soft-state / flooding | local cache and flooding |
| Kozat and Tassiulas | n/a | n/a | n/a | self-organised brokers | soft state / neighbour broadcast | overlay multicast |
| Sailhan and Issarny | name-based | broker | rigid | self-organised broker | soft state / optimised flooding and unicast | unicast and overlay multicast |
| Lenders et al. | attribute value | broker | rigid | flat | soft state / flooding | successive neighbour unicasts |
| GSD | semantic | n/a | common ontology | flat | adaptive soft state / constrained flooding | constrained flooding |
| Service Rings | semantic | client | common ontology | hierarchical ring structure | n/a | overlay unicast |
| Allia | name-based | client | rigid | flat | soft state / constrained flooding and policy based | local cache and policy-based flooding |
| OntoMobil | semantic | client | independent ontologies | randomised | membership / gossip | local cache and random walk |

**Table 2.1**: Classification of mobile service systems.

# Chapter 3

# OntoMobil: Model and Analysis

This chapter presents the OntoMobil model and details the gossip protocol, which is responsible for the formation of the randomised semantic overlay used by concept matching and service discovery. The precise specification for matching and discovery is presented in Chapter 4. Here, we describe the model and formally specify and analyse the gossip protocol. The gossip specification includes the selection, transmission and reception of a randomised set of concepts with a dual goal; maintaining a subset of them at each node and at the same time facilitating a network-wide semantic agreement between the different ontologies. For completeness, a join and leave protocol is also included so that nodes can announce their participation or departure in any network.

The model uses ontology concepts as the main abstraction, which are exchanged and randomly replicated between nodes through the gossip protocol. Concepts also serve as the interface to applications that need to advertise or discover services. As shown in the previous chapter, there is no existing solution for the discovery of autonomous semantic services in mobile ad hoc networks that provides scalable discovery and distributed ontology matching. The model described in this chapter aims to fulfil this gap.

## 3.1 Objectives

The objective of the model is to provide an abstract specification that satisfies the requirements for discovery through the formation of a semantic overlay and for eventual semantic agreement through progressive matching. The model is simple enough to be sufficiently specified using the basic notion of randomised concept manipulation.

The formation of the semantic overlay relies upon a gossip protocol that aims to provide a network abstraction and a suitable set of primitives that can be used by both concept matching and service discovery. The network abstraction is provided through the use of a fixed-size partial membership view forming a topology-independent network overlay. Because nodes are mobile and can advertise arbitrary semantic services, nodes are treated uniformly. This means that the formation of the topology does not track the nodes' location or their semantic content. The use of concept dissemination and replication forms another overlay, a semantic one, where concepts are used as the main primitive. The concepts forming the overlay are augmented by the matching process to include references to other concepts that are semantically similar. Because the discovery process also uses concepts as its main primitive, it can use the overlay to first locate and then introspect on concepts and thus locate nodes with compatible ontologies.

The objective of the gossip protocol analysis is to provide a predictive stochastic model that incorporates all the characteristic protocol parameters and to derive a probability distribution for concept replication across nodes and across time. The evaluation in Section 3.7 verifies the experimental results against the outcome of the analytical model.

## 3.2 A Model for Semantic Service Discovery in MANETs

This section describes the model, its rationale and its properties. The section lists some basic model assumptions, provides the abstract model specification and finally gives a more concrete description of the model in terms of its components and the component's interactions.

### 3.2.1 Model Description

The model assumes that each mobile node that provides one or more services becomes a participant once it connects to an ad hoc network. It then starts the execution of the gossip protocol and can initiate discovery queries. Initial connection is handled by a specialised bootstrap protocol, which discovers other participants and advertises the new node's presence. The bootstrap procedure can also be used to establish connectivity between two or more nodes that are not yet part of any network. Without loss of generality, each node may maintain multiple ontologies, which are treated as a single one.

The model specification uses *concepts* as the core abstraction and *randomisation* as the main process. The specification begins by considering a finite set of concepts that are split uniformly across nodes. The aim is to compare all concepts in a pair-wise fashion by using all available nodes, while at the same time providing a replication pattern that scales as nodes and concept sizes increase. The model uses the intuition that both ontologies and semantic queries can be easily decomposed into their constituent concepts. To simplify, each node is assumed to know every other node and nodes initiate actions in parallel. Initially, each node randomly selects a fixed number of concepts and transmits them to a random but fixed number of destinations. In subsequent steps, each node mixes the received concepts with those from its own ontology and performs the same random selection and transmission. To prevent the eventual replication of all concepts into all nodes, the model restricts both the retransmission of received concepts and their propagation. This restriction is necessary so that the model can scale when the number of concepts increase. The restriction is facilitated with the use of two constraint parameters, a transmission threshold (age) and a propagation threshold (time to live). The model requires each participant to contribute a fraction of its resources in order to aid network-wide collaboration. Such collaboration enables pair-wise concept comparison in a progressive and distributed fashion and causes the replication of concepts across nodes in a pattern that is parameterised and normally distributed. Concept comparison can now be complemented by a matching algorithm, while concept replication can be used for concept discovery. For discovery in particular, a query can follow a random path until it identifies concepts that are similar to the query's constituent concepts. Replication guarantees that

similar concepts will be found in a number of hops that is probabilistically bound. This model offers a scalable method for the discovery of autonomous semantic services and an efficient way to match heterogeneous ontologies. It is scalable because each node maintains only partial network metadata and efficient because matching load is progressive and distributed across all nodes.

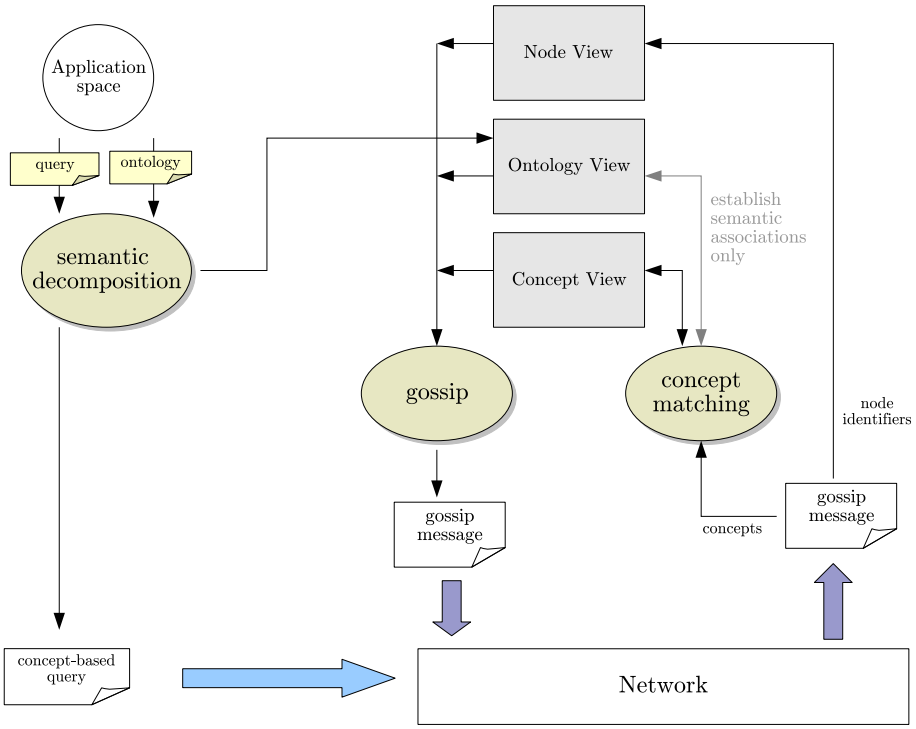


**Fig. 3.1**: OntoMobil model overview. Oval shapes denote processes and rectangular shapes depict data structures. Thin lines with arrows represent the input/ouput flow between structures and algorithms. Thick filled lines represent network communication.

The following sections present a more concrete description of the algorithms and data structures that compose the model. Figure 3.1 presents a graphical overview of the model.

### Semantic decomposition

Semantic decomposition takes a semantic description as input and produces a set of concepts. The decomposition applies to both ontologies and service queries. Ontology decomposition can use a semantic reasoner to produce a hierarchy of concepts. For ontologies based on

Description Logics, concept classification is a standard feature offered by most reasoners. Query decomposition requires a specialised algorithm described in Section 4.7. To reduce the overhead associated with the use of a semantic reasoner, ontology decomposition occurs once during initialisation. Since ontologies are static and do not change at runtime the resulting set of concepts can be stored for subsequent use. In OntoMobil, this set of concepts constitutes the *ontology view*. Concepts that are extracted at this stage need to be augmented with protocol, node and ontology-specific information. Protocol-specific information includes a time-to-live (ttl) parameter controlling the propagation of each concept. A node identifier suffices as node information, allowing replicated concepts to maintain a reference to their source node. As will be explained in Section 4.3, the concept representation also includes some semantic context, e.g., inherited properties, from the ontology where it was defined, so that the matching algorithm can more accurately identify concept similarity.

### Gossip

The gossip protocol, which is described in detail in Section 3.5, periodically exchanges randomised sets of concepts between nodes. A core part of the protocol are three views (buffers), the ontology view, which was described already and two more views with different functionality and properties.

The *concept view* is used to hold received concepts. It is a randomised view with a size that is controlled by the overall number of concepts, the number of participants and the parameters of the gossip protocol. The concept view size is not constant, but varies over time for each node and within a protocol round it also varies between nodes. More importantly, each concept view size is partially dependent on the concept view sizes of the other participants, making it possible to derive a network-wide concept replication factor.

The third view, called the *node view*, stores a partial list of node identifiers. It is initially populated by a bootstrap protocol and subsequently maintained by the gossip protocol. A populated node view is a prerequisite before a node can advertise and discover services. Two extra protocols are devised to handle join and leave requests and enable new nodes to participate and existing nodes to disconnect from a network.

Since the gossip protocol operates on a finite set of data, concept exchange could eventually terminate when the network is in a stable condition, i.e., no new nodes join or existing nodes leave and all concepts have been pair-wise compared. A probabilistic gossip suspension algorithm is introduced for this purpose in Section 4.4.

**Concept matching**

Concept matching, described in Section 4.3 is a per node process that operates on each incoming gossip message. The goal is to identify and establish semantic similarity between the received concepts and those stored in the ontology and concept views. Two different methods are supported that can be used to identify semantic affinity. For the first method, important properties of concepts (e.g., name, semantic context) are compared and matched at runtime, using the well-known ontology matching algorithm H-MATCH [Castano et al., 2003a]. For the second, each node can use a number of predefined mapping relations between its own ontology and other ontologies. This method is less flexible than the previous one, but is more efficient since it alleviates runtime matching costs. Similarity is established with the creation of a new property that is inserted in any concepts identified as semantically equivalent. During discovery this property is examined in order to locate compatible ontologies in remote nodes.

**Service Discovery**

Service discovery is described in detail in Section 4.5. It begins with the formation of a service request. Requests can be based either on the local ontology that is maintained by the source node, or on remote ontologies maintained by other participants. A request that is formed using the source node's ontology is dependent on the progress of the matching algorithm to discover nodes that host compatible ontologies and services. A request that is described by a remote ontology must first locate the concepts that compose the request and through that identify the relevant provider node. Both scenarios use a lookup protocol that follows a random path in the network until it collects enough information about potential destination nodes.

The following steps briefly outline the discovery process:

1. a service request is decomposed and transformed to a concept-based query,
2. due to replication, concepts that match the query concepts can be discovered in a limited number of hops,
3. since all concepts are augmented with node information, matching concepts provide a list of candidate nodes that host compatible ontologies and potential services,
4. the query can now be redirected to the candidate nodes where semantic matchmaking can take place between the required and advertised services.

The first step reformulates a semantic service request to a concept discovery problem. The second and third steps utilise concept replication and the progressive matching approach to bound the number of hops for the discovery of any potential matching concepts. The query is then redirected to any nodes that host compatible ontologies.

### 3.2.2 Rationale

The rationale for the model design lies partly on the specific properties of mobile ad hoc networks and partly on the device characteristics. Currently, a number of semantic service standards (e.g., WSDL-S, OWL-S, WSMO) are targeted towards the Web, complemented by a set of protocols (e.g., UDDI, SOAP) designed to enhance service interoperability and interaction. Two central assumptions about the Web and the Internet environment are the persistence of references and the availability of resources. Even in decentralised P2P network topologies, a modest assumption about longevity of ontology references can be made. Clients need only know a URI reference to obtain an ontology. Additionally, resource availability means that sophisticated schemes can be devised to match ontologies and discover services.

The above assumptions do not hold when mobile nodes form an ad hoc network. The transient nature of communication means that even the weakest assumptions about the existence of ontology references cannot be guaranteed. Only ontologies maintained by currently connected nodes can be referenced and only for the period that a node remains connected. On the other hand, physical resources, (e.g., battery life, CPU power, bandwidth) are scarce and require consideration when trying to exchange and match different ontologies.

Recent standards for semantic web services specify how services are described but are independent of any concrete mechanisms for the discovery of services. To enable the use of such standards in an environment where providers are autonomous and no fixed infrastructure exists, discovery faces two problems that are addressed by OntoMobil:

1. *Discovery queries must be interpreted by nodes with heterogeneous ontologies* – So far a single domain ontology has been assumed for the description of MANET services. This makes possible the direct evaluation of discovery queries between different mobile nodes. Given the opportunistic and unpredictable interaction patterns in MANETs, it is inappropriate to assume that a global ontology is available in every mobile node. Rather, each autonomous node will maintain its own ontology to describe its own services. Since a shared understanding is still required for meaningful semantic interpretation and service interaction, an ontology matching process is needed. To address a similar problem in the context of the Web, the WSMO standards also assume ontology heterogeneity and include the notion of mediators to establish a shared understanding between a service request and an advertisement.

2. *Lack of persistent and centralised service registries* – In connected, fixed networks, certain assumptions can be made about longevity of ontology references. Usually, a URI reference is enough for clients to obtain an ontology from a central repository. Resource availability and infrastructure support in these environments also means that ontology matching and semantic service discovery can employ sophisticated techniques that use central facilities. In MANETs, transient communication makes brokering architectures harder to support, requiring an on-demand approach to the acquisition of ontologies. Hence, only ontologies in connected nodes can be used and discovery is inherently distributed.

### 3.2.3 Properties

The OntoMobil model exhibits the following properties:

### Scalable interaction as mobile nodes increase

As nodes increase, therefore also increasing the total number of concepts, scalability in OntoMobil becomes a challenging issue. It must satisfy the conflicting requirements of keeping device resources fixed against maintaining a constant number of hops for service discovery and a constant latency for semantic agreement. OntoMobil proposes a parameterised model that can shift this trade-off in different directions. Since all three requirements are directly related to the size of the concept view, having a concept view that is not fixed-size, but is parameterised and with predictable bounds, provides an adaptable and configurable solution and is the key scalability feature of OntoMobil. In particular, understanding how the protocol affects replication can provide a useful assessment on the processing and memory requirements for each node and the number of hops required before an unknown concept is discovered.

### Distributed discovery of semantic services

The utilisation of the random walk approach to route service requests simplifies discovery and is a natural "fit" for the randomised topology created by the gossip protocol. It also guarantees that any imposed discovery load will be distributed evenly across all participating nodes. An issue with such an approach is that route traversal is not topology aware. As the node view maintains a randomised subset of node identifiers, the included identifiers may reference nodes that span several physical hops, increasing the probability that service requests are lost before successfully locating any provider nodes. A potential remedy to this situation is to optimise the random walk protocol or use a more reliable unicast protocol. Such optimisations are similar to the ones used by the gossip protocol and are discussed in Section 3.5.6.

### Routing independent design

Even though a number of MANET discovery protocols are tightly coupled with the routing layer (e.g., [Kozat and Tassiulas, 2004]), OntoMobil is routing layer agnostic. Many of the low-level discovery protocols cite efficiency as the motivation for placing the design close to

the routing layer. In OntoMobil, two reasons led to the high-level approach: First, the complex service representation and the requirement for ontology matching would make routing protocol integration overly complex. There is a fundamental mismatch between the aims of an ad hoc routing protocol and OntoMobil. Second, between packet overhead and protocol correctness, OntoMobil veers towards the protocol correctness, expecting that optimisations can be later applied where it seems necessary.

#### Unconstrained discovery

OntoMobil advocates a flat approach to service discovery. There is no hierarchy formation or broker election, resulting in nodes maintaining minimal state. This simplifies node insertion and removal by not requiring an extensive reconfiguration and is more adaptable to volatile mobile environments. Furthermore, traversal of the overlay topology gives uniform access to all provided services and not just those within the geographical or semantic proximity of the requesting node.

#### Progressive matching

Semantic consensus in OntoMobil is based on a fundamental trade-off between consensus latency and processing overhead. The model avoids the costly processing overhead of instantly matching a large number of heterogeneous ontologies by distributing such processing across all participants and across time. Each gossip reception advances the semantic knowledge of the network in an incremental fashion. The parameterised nature of the gossip protocol can also shift this trade-off in different directions depending on the network and node characteristics. For example, a higher replication factor means that each gossip reception will be matched with a larger set of concepts, but at the expense of extra processing. On the other hand, decreasing replication reduces memory and processing load but increases consensus latency and the number of hops required for discovery.

## 3.3 Background: Gossip Protocols

*Gossip*, or *rumour spreading*, or *epidemic* protocols are a class of protocols that bypass traditional deterministic communication schemes in favour of a probabilistic approach [Eugster et al., 2004]. Incorporating a degree of uncertainty into a communication protocol is shown to simplify the algorithmic design and results in overall behaviour that reduces overhead and increases resilience to failures. Epidemic protocols were first developed to manage consistency in the field of databases [Demers et al., 1987]. With the progression from the client-server to the P2P paradigm, epidemic protocols have been rediscovered and are now being used to balance network scalability against strict delivery guarantees (e.g., timeliness, atomic delivery semantics). Their use in the domain of mobile ad hoc networks [Haas et al., 2006, Luo et al., 2004] has also shown that the randomised nature of gossip algorithms is a suitable counter-measure against the random and uncertain topology that is typical of MANETs.

The main characteristics of gossip protocols are summarised below:

- *Distributed operation* – gossip protocols are used in situations where a large number of peers require some form of weak consensus. Reliable broadcast, failure detection, membership maintenance are typical scenarios. In these situations it is either expected that the use of centralised infrastructure cannot adequately provide enough resources to support the envisioned network scale or peers cross administrative boundaries and therefore centralised support is impossible logistically.

- *Point-to-point communication* – gossip protocols derive some of their desirable properties by using a unicast communication pattern. This avoids costly request-response schemes or the use of flooding. In fact, they have been used primarily as a stateless substrate to multicast or broadcast. Stateful multicast approaches, such as tree or mesh-based, require select nodes to maintain state on behalf of other nodes. In the case of large-scale dynamic networks, maintenance of such state can be complex and create overhead. Gossip protocols usually require no other state apart from the maintenance of a group membership view.

- *Resilience to failures* – partial replication is a side-effect of the randomised point-to-

point communication pattern exhibited by gossip protocols. For example, in gossip-based multicast protocols, some nodes might receive the same message multiple times. This requires techniques to discard duplicate messages, such as maintaining old message identifiers. However, the overhead associated with replication is counter-balanced by the resilience to certain types of failures. In the multicast domain, gossip protocols can adapt to message omission, fail-stop and network partition failures. In networks that exhibit a high rate of failures, such as mobile ad hoc networks, the inherent replication stemming from gossip protocols can offer a viable alternative to protocol-specific fault-tolerant procedures.

- *Probabilistic guarantees* – another appealing characteristic of gossip protocols is their abstraction to well-understood analytical methods. Such analysis is usually borrowed from the field of epidemiology [T.J.Bailey, 1975] and typically uses stochastic processes to model the spread of a message through the network. Given the large number of parameters that gossip protocols require, such analysis can be useful as a predictive performance model without using a simulation environment. In Luo et al. [2003], the authors state that gossip protocols strike a good balance between formal specification and scalability as compared to protocols that have a strict specification but scale poorly, or to protocols with practical scalability properties but no formal reliability analysis.

Gossip protocols borrow their terminology from the theory of epidemics. All nodes start and remain in the *susceptible* state until they receive a message representing a "virus". A node receiving the virus (message) shifts from the susceptible state to the *infective*, if it can "infect" other nodes, i.e., further propagate the received virus. It can also become *infected* if the virus cannot be propagated further. It is possible that a node in the infective or infected state can also become *removed*, meaning it no longer holds a copy of the virus and can be later reinfected.

Gossip protocols are usually message-based and operate in rounds. A round is defined as the time period in which all nodes that wish to propagate information transmit a single message and their corresponding destinations receive it. Round-based communication provides a simple and useful abstraction that is used to derive analytical bounds on the properties of

the gossip protocol.

## 3.4 System Assumptions

This section provides the set of system assumptions and notation used throughout the rest of the thesis. The model considers an ad hoc network as composed of a set $\mathcal{N}$ of mobile nodes. A subset $\mathcal{M} = \{n_1, n_2, \ldots, n_{\mathtt{M}}\} \subseteq \mathcal{N}$ of size $\mathtt{M}$ are considered to be active participants that maintain ontologies and share services. To simplify the description of the model it is assumed that all nodes in $\mathcal{N}$ are initially uniformly placed across a geographical area. It is also assumed that the initial placement of nodes in $\mathcal{M}$ is also uniform with respect to the nodes in $\mathcal{N}$. Uniform node placement simplifies the bootstrap protocol in Section 3.5.1, which aims to provide all nodes with uniform and partial membership views. Subsequent movement of nodes can be arbitrary and we do not pose any restrictions on the mobility model.

All nodes are addressed by a unique identifier (id). Nodes that are not participants need only have the basic capability of forwarding packets. In the general case, forwarding can be facilitated using an optimised flooding protocol such as SMURF [Perkins et al., 2005]. In the current implementation, a custom flooding protocol is used that simply rebroadcasts received packets that have not been seen before to all one-hop neighbours. It is assumed that all nodes communicate using a fixed-range, wireless medium (e.g., IEEE 802.11b) and that a unicast routing protocol (e.g., AODV, OLSR) is available.

Each node $n_i \in \mathcal{M}$ maintains the three different views that were described in Section 3.2.1. The views are denoted in the following notation: $\mathcal{V}_i^{\mathtt{O}}$ for the ontology view, $\mathcal{V}_i^{\mathtt{C}}$ for the concept view and $\mathcal{V}_i^{\mathtt{N}}$ for the node view.

The ontology view represents a node ontology as a fixed set of concepts, $\mathcal{V}_i^{\mathtt{O}} = \{c_{i1}, \ldots, c_{i\mathtt{G}}\}$, where $c_{il} \in \mathcal{V}_i^{\mathtt{O}}, 1 \leq l \leq \mathtt{G}$ is a concept in the ontology of source node $n_i$ and $\mathtt{G}$ represents the maximum number of concepts per node. The model assumes that these ontologies are static so this view allows no additions or deletions of concepts for the duration of the protocol execution. This assumption is reasonable as ontologies are structured metadata, meaning they are specified during application design and do not change often.

A node's concept view includes the set of concepts that are received from other nodes and

does not allow duplicate concepts or concepts that exist already in the node's ontology view. We represent this view as $\mathcal{V}_i^{\mathtt{C}} \subseteq \{c_{kl} \,|\, c_{kl} \in \mathcal{V}_k^{\mathtt{O}}, k \in \mathcal{M} - \{n_i\}, 1 \leq l \leq \mathtt{G}\}$, where $c_{kl}$ represents a concept from any ontology view except the one in node $n_i$. The gossip protocol provides the concept view with the following properties. It is:

- *probabilistically bound* – the concept view is not constrained by a fixed size, rather the protocol guarantees a bound on its size with a certain probability. The intent of the different gossip parameters is to keep the concept view partial, i.e., with a certain probability it should maintain only a subset of the total number of concepts,
- *evolving* – the gossip protocol constantly inserts and removes concepts,
- *simple random sample* – since each view does not contain a set of concepts that concretely describe a knowledge domain, rather it contains randomised concepts that can belong to any of the available ontologies. Furthermore, each view contains a set of concepts that is independent from any other view.

The node view is a set of node identifiers, $\mathcal{V}_i^{\mathtt{N}} \subseteq \{n_k \,|\, k \in \mathcal{M} - \{n_i\}\}$. Like the concept view it does not allow duplicate node identifiers and can not contain the node's own identifier. It is a membership view that has similar properties as those found in recent gossip protocols [Eugster et al., 2003, Luo et al., 2003, 2004]. The node view is:

- *fixed-size* – contrary to the concept view, the node view does not automatically adapt to an ever-increasing group size. More sophisticated protocols have been proposed that adapt the size of the membership view as the group size increases [Ganesh et al., 2001], but this is outside the scope of this thesis. The node view is also intended to be partial, without contain the complete list of all participants,
- *uniform* – the probability that a node identifier exists in a node view is the same for all node identifiers. In other words, if we merge all node views, each node identifier has the same frequency of appearance,
- *randomised* – the distribution of identifiers in the node views is a random distribution.

The node view is populated during a bootstrap phase and is subsequently maintained by the gossip protocol. The consequences of maintaining the two partial views ($\mathcal{V}^{\mathtt{N}}$ and $\mathcal{V}^{\mathtt{C}}$) is that no node holds the complete knowledge of all participating nodes or all ontology concepts. This enables the model to scale in terms of nodes and concepts and forms the core of the matching and discovery processes.

## 3.5 Gossip-based Concept Dissemination

The gossip protocol is completely characterised by the following parameters[1]:

- $\mathtt{F}_c$ – the concept fanout specifies the number of concepts a source node includes in a gossip message,
- $\mathtt{F}_n$ – the node fanout specifies the number of destination nodes a gossip message is sent to,
- $\mathtt{T}_a$ – specifies the number of times a node transmits a received concept before the concept is removed from the concept view. For convenience, we define the function $age(c)$ that takes concept $c$ as input and returns its age,
- $\mathtt{T}_t$ – the time to live (ttl) value is assigned by each source node to any concept that is selected for transmission from the ontology view. It specifies the number of hops that a concept will traverse before being discarded, i.e., when the ttl value reaches zero.

### 3.5.1 Join

Before a node begins execution of the gossip protocol, it must first populate its node view with a partial list of other participants. A simple bootstrap protocol is used that is similar to the one used in Luo et al. [2003] in order to arrive at a partial and uniform membership view. This join protocol is based on a simple *expanding ring search* and is used when a node enters an ad hoc network or after a node failure. Listings 1, 2 and 3 provide the pseudocode

[1] We have used the short form $\mathtt{F}_{<qualifier>}$ and $\mathtt{T}_{<qualifier>}$ for protocol parameters that are used in the stochastic analysis and the longer form $\mathtt{Parameter}_{<\mathtt{qualifier}>}$ for secondary parameters.

for the join protocol. An initial request is flooded through the network with each receiving node sending a reply back to the source node with some probability (line 5 in Listing 2). Using a probabilistic method to acquire replies avoids flooding the network with responses when many nodes are clustered together [Tseng et al., 2002]. During this initial exchange, both the source and the receiving nodes populate their node views with the corresponding node identifiers. Using definitions borrowed from the gossip literature [Demers et al., 1987], the bootstrap protocol is both a *push* and a *pull* gossip protocol. During the push (request) phase, the new node populates the node views of existing nodes, while during the pull (reply) phase, the new node learns the identifiers of the existing nodes. A node enters the gossip protocol execution when it receives enough responses so that its node view size exceeds a threshold ($\texttt{Parameter}_{\texttt{NodeView}} = |\mathcal{M}|/2$). This threshold is currently set to half the size of all participants for reasons that are elaborated in Section 3.5.5. The main differences with the join protocol in route driven gossip [Luo et al., 2003] is the absence of a group identifier (gid) in OntoMobil and the fact that the implementation here is routing protocol agnostic, rather than coupled with the DSR routing protocol [Broch et al., 1999].

**Listing 1** Join Request Transmission

1: // $JoinRequestMessage$ represents the request packet
2: At node $j$
3: **while** $|\mathcal{V}_j^{\mathtt{N}}| < \texttt{Parameter}_{\texttt{NodeView}}$ **do**
4: $\quad JoinRequestMessage.source \leftarrow j$
5: $\quad$ ***broadcast***$(JoinRequestMessage)$
6: **end while**

**Listing 2** Join Request Reception and Join Reply Transmission

1: //$JoinRequestMessage$ represents the request packet
2: //$JoinReplyMessage$ represents the reply packet
3: On reception of $JoinRequestMessage$ at node $j$
4: $\mathcal{V}_j^{\mathtt{N}} \leftarrow \mathcal{V}_j^{\mathtt{N}} \bigcup \{JoinRequestMessage.source\}$
5: **if** $P_{reply} \geq \texttt{Parameter}_{\texttt{Reply}}$ **then**
6: $\quad JoinReplyMessage.source \leftarrow j$
7: $\quad$ ***send***$(JoinRequestMessage.source, JoinReplyMessage)$
8: **end if**

**Listing 3** Join Reply Reception

1: $//JoinReplyMessage$ represents the reply packet
2: On reception of $JoinReplyMessage$ at node $j$
3: $\mathcal{V}_j^{\mathtt{N}} \leftarrow \mathcal{V}_j^{\mathtt{N}} \bigcup \{JoinReplyMessage.source\}$

### 3.5.2 Leave

The current gossip specification requires an explicit disconnection procedure and does not admit unexpected failures. Before a node leaves the network, it declares its intent as part of the periodic gossip transmission. Nodes that receive a gossip transmission from a node that intends to disconnect:

1. remove that node identifier if it exists in their node view,
2. remove any concepts from their concept view that originate from the ontology view of the departing node,
3. remove all references from concepts maintained in their ontology or concept views that point to concepts in the ontology view of the departing node.

In the next gossip round, the nodes that initially received the leave notification will further propagate it, resulting in more nodes executing the same removal sequence as above. Eventually, by gossipping the intention to disconnect, all participants will be reached and all network-wide references to the departing node will be removed. To increase reliability in the face of message loss, the leave specification requires nodes to transmit the identity of the departing node for $\texttt{Parameter}_{\texttt{AgeRemove}}$ rounds. Well-known results in the epidemic literature [Pittel, 1987] guarantee that with the right parameter values, all nodes eventually receive the intention to disconnect message in a logarithmic number of rounds. The pseudocode for the leave process is given in conjunction with the main gossip protocol in Listings 4 and 5.

### 3.5.3 Gossip Protocol

The gossip protocol described here is executed by each participating node with the aim to facilitate semantic agreement and provide a semantic overlay through concept replication.

The specification is given in terms of actions taken during the reception and transmission of a gossip message, which is used to encapsulate a fixed number of random concepts. The protocol uses a gossip push mechanism to disseminate information and contrary to other gossip protocols in domains such as multicast and failure detection, it randomly replicates a potentially large, but finite set of data items (concepts) across all nodes.

Using the epidemic terminology, we consider each concept as a "virus". Initially, all nodes but one are regarded as susceptible. A gossip transmission from the initial infective node will then "infect" a random but fixed number of other nodes. Subsequent infective nodes maintain each virus (concept) until certain conditions are met and subsequently the virus is removed. This matches the Susceptible-Infected-Susceptible (SIS) model as specified in Daley and Gani [2001, page 56]. During the period of infection, a node can infect other susceptible nodes. It follows that nodes with a non-empty concept view are infected by multiple viruses and each node is always susceptible if a virus is not in its concept view. The viral spread achieves two things:

1. infective nodes maintain copies of several unique concepts,

2. through concept matching, each infection propagates concepts that have potential references to semantically related concepts, thereby incrementally progressing the network's shared knowledge.

Listing 4 describes the transmission of a gossip message. Periodically, a node selects $\mathtt{F}_c$ concepts uniformly at random from the union of the ontology and concept views (line 3). A concept that is selected from the ontology view is augmented with the ttl property (line 6), signifying the number of hops the concept will be propagated before being dropped by the receiving node. If a concept is selected from the concept view and has already been transmitted $\mathtt{T}_a$ times, it is removed from that view (line 9). Selecting a fixed but random number of concepts from the union of the two views is a simple algorithm that exhibits a desirable adaptive behaviour. During the initial stages of gossip transmission, a node's priority is to disseminate its own ontology so that its semantic information is diffused throughout the network. Since the concept view contains few elements during the initial rounds, concepts from

**Listing 4** Gossip Transmission

1: //$GossipMessage$ represents the network packet
2: Every $t$ ms at node $j$
3: *Choose* $\{c_{k1}, \ldots, c_{k\mathtt{F}_c}\}, k \in \mathcal{M}$ *random concepts from* $\mathcal{V}_j^{\mathtt{O}} \bigcup \mathcal{V}_j^{\mathtt{C}}$
4: **for all** $c \in \{c_{k1}, \ldots, c_{k\mathtt{F}_c}\}$ **do**
5: **if** $c \in \mathcal{V}_j^{\mathtt{O}}$ **then**
6: $c.ttl \leftarrow \mathtt{T}_t$
7: **end if**
8: **if** $c \in \mathcal{V}_j^{\mathtt{C}} \wedge age(c) = \mathtt{T}_a$ **then**
9: $\mathcal{V}_j^{\mathtt{C}} \leftarrow \mathcal{V}_j^{\mathtt{C}} - \{c\}$
10: **else if** $c \in \mathcal{V}_j^{\mathtt{C}}$ **then**
11: $age(c) \leftarrow age(c) + 1$
12: **end if**
13: **end for**
14: $GossipMessage.concepts \leftarrow \{c_{k1}, \ldots, c_{k\mathtt{F}_c}\}$
15: //Reinforcement of membership view
16: *Choose a random nid from* $\mathcal{V}_j^{\mathtt{N}}$
17: $GossipMessage.nodes \leftarrow \{j, nid\}$
18: $aView \leftarrow \mathcal{V}_j^{\mathtt{N}} - \{nid\}$
19: //Piggyback removal notifications
20: $GossipMessage.removals \leftarrow Buffer_{Removals}$
21: //Select destination nodes
22: *Choose* $\mathtt{F}_n$ *random nodes,* $\{n_1, \ldots, n_{\mathtt{F}_n}\}$ *from* $aView$
23: **for all** $nid \in \{n_1, \ldots, n_{\mathtt{F}_n}\}$ **do**
24: ***send***$(nid, GossipMessage)$
25: **end for**

the ontology view have a higher probability of being selected for transmission[2].

Each time a node transmits a gossip message, it also piggybacks a single node identifier from its own node view (line 16). The random id together with the identifier of the sender (line 17), serve to reinforce the node view of the receiving node and maintain a uniform distribution even after new admissions and disconnections. The idea is to keep a fluctuating node view across nodes, so that initial cluster effects disappear over time. For example, in a situation where multiple nodes that are in close geographical proximity decide to join an existing group, the bootstrap protocol is likely to return the identifiers of the same participants. Even though each new node will have a node view of distinct identifiers, collectively their node views will

[2]The inclusion probability of a concept from $\mathcal{V}^{\mathtt{O}}$ appearing in the concept fanout is actually: $\frac{\binom{|\mathcal{V}^{\mathtt{C}}|+|\mathcal{V}^{\mathtt{O}}|}{\mathtt{F}_c} - \binom{|\mathcal{V}^{\mathtt{C}}|}{\mathtt{F}_c}}{\binom{|\mathcal{V}^{\mathtt{C}}|+|\mathcal{V}^{\mathtt{O}}|}{\mathtt{F}_c}}$ where $|\mathcal{V}^{\mathtt{C}}| > \mathtt{F}_c$ and $|\mathcal{V}^{\mathtt{O}}| > \mathtt{F}_c$. This probability is closer to 1 when $|\mathcal{V}^{\mathtt{C}}|$ is small.

contain very similar identifiers, potentially invalidating the uniform property of the node view. A reinforcement algorithm can help alleviate such problems and provide an eventual uniform spread of identifiers across nodes. Such a reinforcement algorithm for a partial membership view was first introduced in Eugster et al. [2003] and thoroughly analysed in Allavena et al. [2005].

If the sender has received any disconnection notifications, they are also appended to the gossip message using $Buffer_{Removals}$ (line 20). The last action for the sender is to select $\mathtt{F}_n$ distinct nodes uniformly at random from its node view (line 22), making sure that the identifier selected for reinforcement is not selected again as a gossip destination (line 18), and to unicast the gossip message to the target nodes (line 24).

On reception of a gossip message, the receiving node executes the algorithm presented in Listing 5. If the optional field $GossipMessage.removals$ exists then the actions outlined in Section 3.5.2 are executed. The identifiers are then stored in the auxiliary buffer $Buffer_{Removals}$ so they can be further propagated in the next gossip transmission (line 6).

Next, the receiver matches the set of concepts included in the gossip message against its own stored concepts. This is shown in line 10 and is the focus of Section 4.3. Subsequent to matching, lines 11 – 14 show that if a concept is not found in either the concept or the ontology views and the concept's ttl is greater than one, it is stored in the receiver's concept view and the concept's ttl value is decremented by one.

Through the ontology matching algorithm, each message has the potential to create new associations between concepts from different ontologies. This progressive aspect of ontology matching results in the following network behaviour. The longer a node is connected to an ad hoc network, the more likely it is that potential associations between its own and other ontologies will be identified. This prevents nodes that are short-lived or transiently connected from immediately overloading the network with the task of complete ontology matching. A concept view size that is probabilistically bound and a fixed concept fanout also ensure that nodes are not overwhelmed with matching large ontologies. Although the redundancy that is inherent in gossip protocols can seem excessive, it is this very feature that allows progressive matching through the exchange of concepts and scalable discovery through

**Listing 5** Gossip Reception

```
1: //GossipMessage represents the gossip packet
2: On reception of a GossipMessage at node j
3: //Prune entries from removed nodes
4: for all nid ∈ GossipMessage.removals do
5:    Execute algorithm in Section 3.5.2
6:    Buffer_Removals ← Buffer_Removals ⋃ {nid}
7: end for
8: //Process received concepts
9: for all c ∈ GossipMessage.concepts do
10:   Execute ontology matching algorithm between c and V_j^O ⋃ V_j^C
11:   if c ∉ V_j^O ∧ c ∉ V_j^C ∧ c.ttl > 1 then
12:      c.ttl ← c.ttl − 1
13:      V_j^C ← V_j^C ⋃ {c}
14:   end if
15: end for
16: //Node view maintenance
17: for all nid ∈ GossipMessage.nodes do
18:   if nid ≠ j ∧ nid ∉ V_j^N then
19:      if |V_j^N| = Parameter_NodeView then
20:         Prune V_j^N by removing a random node id
21:      end if
22:      V_j^N ← V_j^N ⋃ {nid}
23:   end if
24: end for
```

concept replication.

The last action a receiver undertakes is node view related. A simple algorithm derived from Eugster et al. [2003] is used, which maintains the view's fixed size while constantly updating it with a small number of new identifiers. It is detailed in lines 17 – 23. The algorithm removes an existing node id from the node view only after it checks that the new insertion will exceed the node view's fixed size and that the new insertion is not a duplicate and is not the receiving node's identifier. Removing first and inserting later helps ensure that the newly inserted id cannot be removed, which can potentially occur if the reverse order is used. This change gives marginally better uniform characteristics (Section 3.5.5) and also reflects the fact that even minor adjustments in the execution sequence of a randomised algorithm can influence its behaviour, especially if it is expected to run for a large number

of iterations.

The impact of this randomised concept exchange is twofold: first, the ontology matching component in each node can derive semantic similarity relations between its stored concepts and those received; second, replicated concepts can be used to bound the number of hops required for discovery. The gossip protocol is the basis of the model and provides the foundation on top of which efficient distributed ontology matching and semantic service discovery can take place.

### 3.5.4 Example Operation

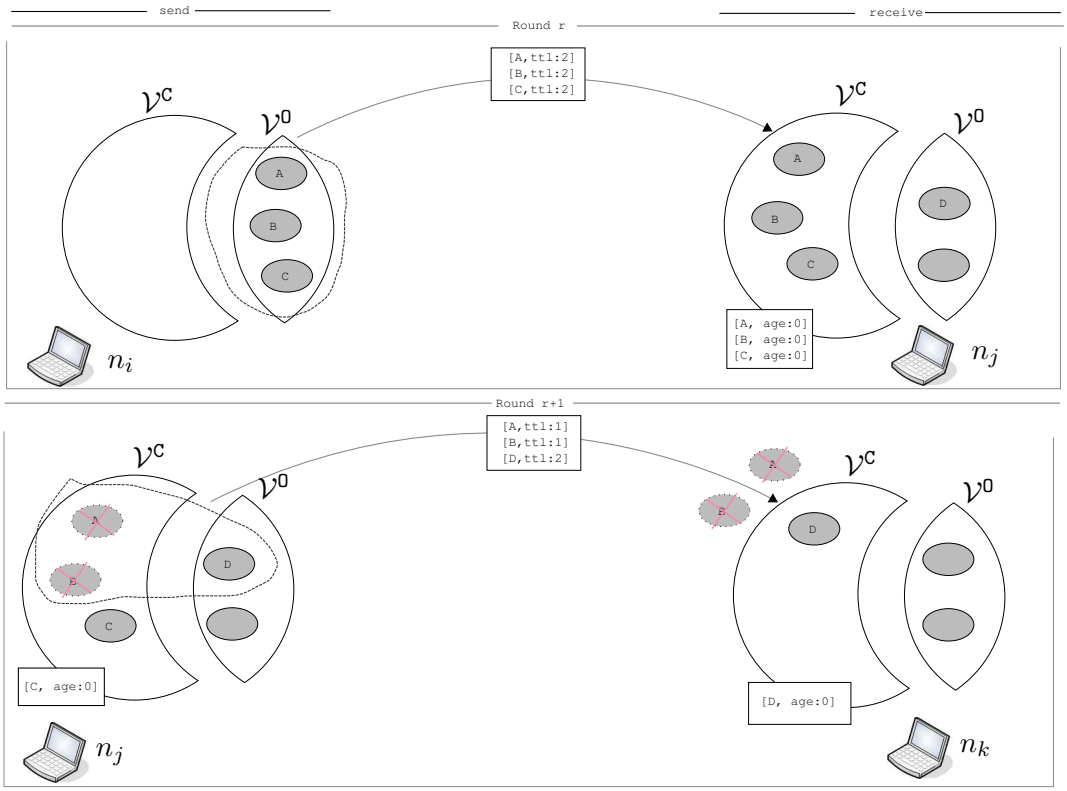


**Fig. 3.2**: An example operation of the gossip protocol over two rounds. The characteristic parameters are $\mathtt{F}_c = 3, \mathtt{F}_n = 1, \mathtt{T}_a = 1$ and $\mathtt{T}_t = 2$.

A hypothetical gossip session with gossip parameters $\mathtt{F}_c = 3, \mathtt{F}_n = 1, \mathtt{T}_a = 1$ and $\mathtt{T}_t = 2$ is illustrated in Fig. 3.2. We assume that at round $r$, node $i$ has an empty concept view and an ontology view that contains three concepts, A, B and C. Node $j$ exists in the node view of

node $i$ and is selected as the destination for $i$'s gossip transmission. Since all three concepts are contained in the ontology view, each one is assigned a new ttl value of $\mathtt{T}_t = 2$. The gossip transmission is received by node $j$ and the three concepts are inserted into the concept view as $j$ contains none of these concepts. With each insertion, the concept's ttl value is decremented by one. An internal table that exists at each participant and maps concepts to their current age is also updated with the name of the concept and a default value of zero.

During round $r + 1$ we observe node $j$. It is assumed that node $k$ exists in $j$'s node view and that $k$ is selected as the gossip destination. Furthermore, the concept selection process yields concepts A, B and D as the gossip message content. Since the age parameter is set to $\mathtt{T}_a = 1$, once a concept gets selected from the concept view it is also removed. Hence, after transmission, only concept C remains in $j$'s concept view, with A and B being removed. Note however that any matching associations that were established between A, B and C and $j$'s ontology view concepts continue to persist. They will only be removed once $i$ disconnects. At the destination side, node $k$ examines the ttl values of the received concepts and *after* matching A, B and D with its own concepts, inserts D in its concept view, since it is the only one with a ttl value greater than one. Similar to the previous reception by node $j$, an entry in the age table is created for the newly added concept.

### 3.5.5 The Issue of Node View Uniformity

The gossip protocol uses and strongly depends on partial membership views that must be uniform at all times. The stochastic analysis undertaken in Section 3.6 illustrates this dependency more clearly.

Partial, fixed-size and uniform views were first introduced in the lightweight probabilistic broadcast (lpb) system [Eugster et al., 2003] to increase the scalability of the gossip-based event dissemination protocol. The membership algorithm in OntoMobil is similar to the one described in the lpb system with some minor modifications in the order of the insertion and removal of identifiers. However, lpb provides no formal analysis on the uniform properties of the view maintenance algorithm. Uniformity is intuitively assumed, though the authors report a deviation between their analytical and simulation results and attribute the deviation

to uniform views that are "imperfect". Their evaluation is not focused on uniformity itself, rather on the impact of various view sizes on certain aspects of their gossip protocol.

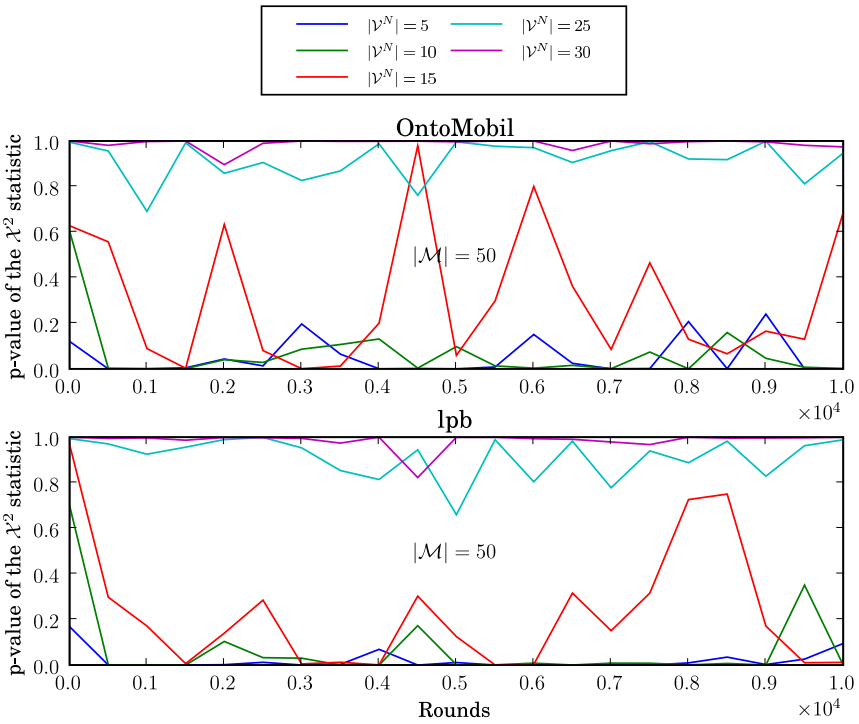


**Fig. 3.3**: Variation in the uniformity of the partial membership views by means of a $\mathcal{X}^2$ test. Experiment invariants: $|\mathcal{M}| = 50$, $\mathtt{F}_n = 2$.

To measure how view size impacts view uniformity, the chi-square ($\mathcal{X}^2$) test was used. The test is conducted to verify the hypothesis that in lpb-like view maintenance algorithms, uniformity and view size are actually correlated and are not independent as claimed. The results are presented in Fig. 3.3 and Fig. 3.4. The chi-square test also included an OntoMobil variation on the lpb algorithm against the original lpb, on networks of 50 and 100 nodes, for 10000 rounds with the node fanout set to two ($\mathtt{F}_n = 2$). The $\mathcal{X}^2$ test is conducted after the end of each round. The $\mathcal{X}^2$ null hypothesis ($H_0$) states that the observed frequencies match the expected ones. Setting expected frequencies as the ideal uniform views (always $|\mathcal{V}^{\mathtt{N}}|$), the test can provide a practical measure of uniformity. The logic of the test is reversed, since we seek to accept the null hypothesis rather than reject it. Therefore, a high p-value is the

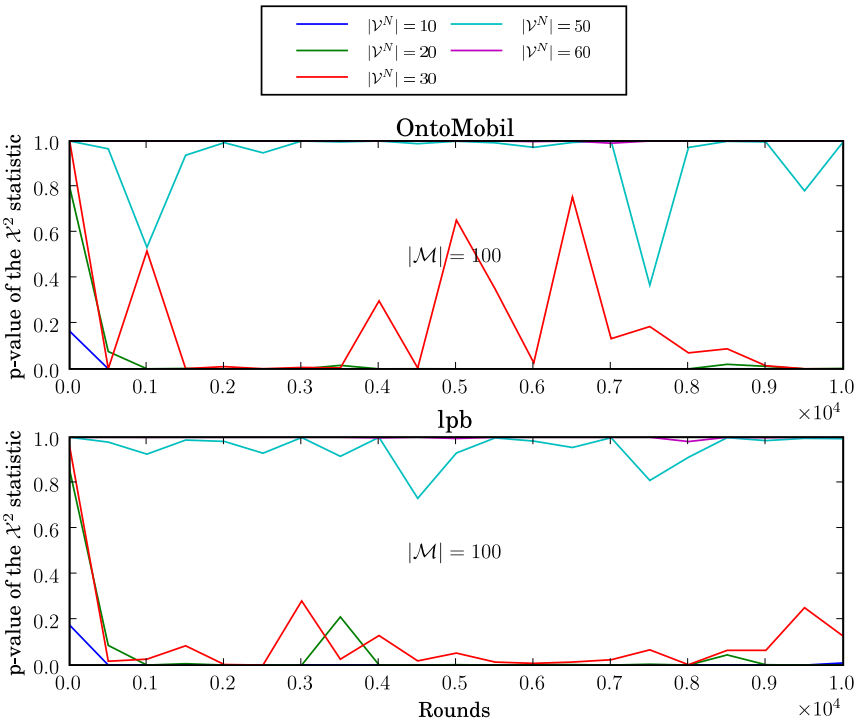


**Fig. 3.4**: Variation in the uniformity of the partial membership views by means of a $\mathcal{X}^2$ test. Experiment invariants: $|\mathcal{M}| = 100$, $\mathtt{F}_n = 2$.

desirable outcome.

As shown, until the view size becomes equal or larger than half the node population, uniformity is very low and varies widely. It is only after the cardinality of the node view exceeds the 25 and 50 identifiers for $|\mathcal{M}| = 50$ and $|\mathcal{M}| = 100$ respectively that consistent and perfect uniform views are maintained. We have used this result in all further simulations with individual node views set at $|\mathcal{M}|/2$.

Since the original inception, other systems have used the idea of partial views as a way to improve scalability [Luo et al., 2003, 2004]. The protocols used in these systems for membership maintenance are largely derived from the original lpb specification. It is not until RaWMS [Bar-Yossef et al., 2006] that a protocol is presented to provide provable partial and uniform views in mobile ad hoc networks. The protocol is based on a random-walk scheme and has the desirable property that the view size can be independently set by each node.

Uniformity in RaWMS has a different definition than OntoMobil. In RaWMS, uniformity is derived by guaranteeing that each view selects identifiers uniformly at random from the total number of nodes. OntoMobil also requires that the frequency of appearance is the same for all identifiers. This can be satisfied in RaWMS when each view is set to the same size. There are other protocols [Ganesh et al., 2001, Allavena et al., 2005] that provide provable partial view properties, without necessarily providing uniformity. In OntoMobil, the gossip protocol requires uniform partial views, but is nonetheless independent of any specific view maintenance protocol. Future work can explore the integration between OntoMobil and RaWMS.

### 3.5.6 Optimisations

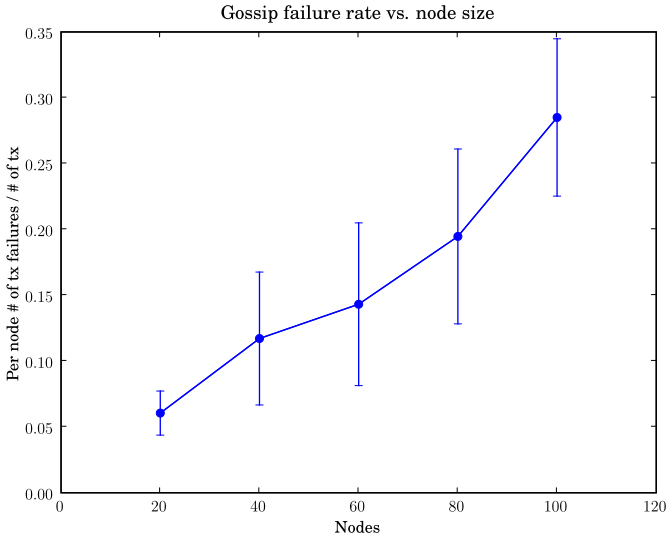


**Fig. 3.5**: The 95% confidence interval of the mean per node ratio of failed gossip transmissions against network size. Each value is averaged over 8 experiments using different permutations of the gossip parameters ($\mathtt{F}_n = \{1, 2\}, \mathtt{T}_a = \{1, 2\}, \mathtt{T}_t = \{2, 3\}$). The routing protocol is OLSR.

As described in Section 3.2.3, the model design has separated the gossip protocol from specific routing features. Simulations have demonstrated that this abstraction results in a noticeable increase in transmission failures in large ad hoc networks ($|\mathcal{N}| \geq 60$). Such failures

are even more pronounced when the routing protocol is reactive (e.g., DSR, AODV). Proactive protocols (e.g., OLSR) adapt better to the OntoMobil traffic flow, but as the average number of hops to destination increases, so does the failure rate. Figure 3.5 illustrates the per node ratio of failed gossip transmissions against the size of the ad hoc network. For this discussion it is useful to know only that the routing protocol is OLSR, though Chapter 6 gives the complete simulation parameters. There is a clear increase in the number and variability of failed transmissions as the network size increases. Such an increase reduces the reliability of the gossip protocol.

Specifically, since the properties of the gossip protocol rely on the correctness of the transmit-receive interaction, high failure rates can invalidate the probabilistic guarantees in concept replication. Simulations have shown that a uniform failure rate, where all nodes exhibit a similar ratio of failed transmissions, of $\sim 15\%$ can be tolerated by the protocol without significance performance degradation. Higher failure rates are followed by a higher failure variability across nodes and thus have a larger affect on the properties of the gossip protocol.

Different optimisations can be used depending on the nature of the underlying routing protocol. When a reactive routing protocol is used, a useful optimisation is described in Luo et al. [2003]. Although the optimisation is used in a group communication setting, the OntoMobil gossip protocol also follows a many-to-many communication paradigm with a more lightweight traffic pattern. The specific optimisation is centered around transmitting only to destination nodes for which the source node has an active route. The idea is to avoid costly route discovery procedures taking place in each transmission and to use routing information to increase reliability. A specification is presented that is coupled with the DSR routing protocol and can potentially be used by OntoMobil.

Proactive routing protocols have better performance characteristics when the traffic pattern resembles that of OntoMobil. The maintenance of routes to all connected nodes makes such protocols ideal candidates when message flow is between constantly changing and arbitrary pairs of nodes. This has motivated the use of OLSR [Clausen et al., 2001] for all simulations. A potential optimisation that can be applied with a proactive protocol is the

combined use of topology-aware membership views with a weighted destination selection mechanism. The idea is to have the node view maintain, in addition to the set of node ids, a corresponding estimation on the number of hops required to reach each of the corresponding nodes. This information can be recorded each time a gossip message is received and maintained together with an associated timer to track the accuracy of the hop distance. As discussed in Listing 4, each gossip transmission piggybacks two identifiers to reinforce the node view of the receiving node; the identifier of the source node and an identifier that is randomly chosen from the source node's node view. It is simple to calculate the hop distance for the source identifier. It requires each intermediate node to increment a counter until message reception. For the identifier that is randomly chosen from the source's node view, hop information will not be included. Having a node view that is partially populated with a distance metric can facilitate an algorithm that does not select identifiers uniformly at random but is biased to those identifiers with a short hop count and timely information. In other words, selection of destination nodes will favour those that are in the geographical proximity of the gossip sender. Note that such an optimisation should still maintain the independence in the contents of the concept view, while also maintaining the probabilistic bound on its size.

## 3.6 Stochastic Analysis of OntoMobil

When considering the applicability of the proposed model, the size and variability of the concept view become important factors. They influence the scalability of the gossip protocol as memory consumption, processing overhead and the probabilistic guarantees for service discovery depend on the distribution of replicated concepts across nodes. In particular, memory consumption depends partly on the number of concepts in the concept view and partly on the verbosity of the chosen concept representation[3]. Processing overhead is also proportional to the size of the view as concept matching is performed between received concepts and concepts in both ontology and concept views. Finally, the random walk service discovery uses concept replication to bound the number of hops before a concept is found.

[3]An XML-based concept representation is likely to consume more memory than a custom binary one.

This section uses a stochastic analysis to derive a predictive model[4] of the proposed gossip protocol, given the total number of concepts, the number of nodes and the characteristic parameters of $\mathtt{F}_n$, $\mathtt{F}_c$, $\mathtt{T}_t$ and $\mathtt{T}_a$. The main aim of the analysis is to formulate a probability measure of the concept view size. This will be used as the foundation to derive a probability measure for the service discovery mechanism. This is discussed with the discovery protocol in Chapter 4.

The variability of the concept view size is modelled using the random variable $V$. Based on $V$, the probability mass function (pmf) $f_V(v)$ is derived, which contains complete information about the distribution of $V$ to satisfy the main goal of the analysis.

### 3.6.1 Analytical Assumptions

To make analytical modelling tractable a number of assumptions and simplifications were made. Similar to the system assumptions in Section 3.4, the analysis considers a fixed set $\mathcal{M}$ composed of $\mathtt{M}$ active participants. It is assumed that all nodes maintain the same number of concepts $\mathtt{G} = |\mathcal{V}^{\mathtt{O}}|$ in their ontology views. The total number of concepts is then $\mathtt{G}_{total} = \mathtt{M} \cdot \mathtt{G}$. The analysis also requires the characteristic parameters to be identical for all nodes.

In reality, the protocol does not constrain nodes with the strict requirement of ontology views having an equal number of concepts. However, when nodes have ontology views with large deviations from the mean view size, it can distort the predictability of the analytical model. This is because the stochastic analysis is an average case analysis that treats all node uniformly.

Recall that concepts are inserted in the concept view if they do not exist in the two views ($\mathcal{V}^{\mathtt{C}}$ and $\mathcal{V}^{\mathtt{O}}$) already. Concepts are also removed when they are selected for transmission from the concept view and their age parameter reaches $\mathtt{T}_a$. The analysis presented here focuses on modelling the variability of the concept view size, based on the probability that a certain number of concepts are inserted and removed in each round.

As is evident from the protocol specification, the size of the concept view varies both across nodes and also across rounds. This means that a node's concept view varies across

[4]In this section, the word model can either mean OntoMobil or the stochastic model that is used for the analysis of the gossip protocol. The context should clarify which model is being referenced.

rounds, while in a certain round there is variability of view sizes across participating nodes. For the analysis it is assumed that the size distribution across both of these dimensions is Gaussian and as the number of rounds and the number of nodes increase the mean and variance of these respective distributions converge.

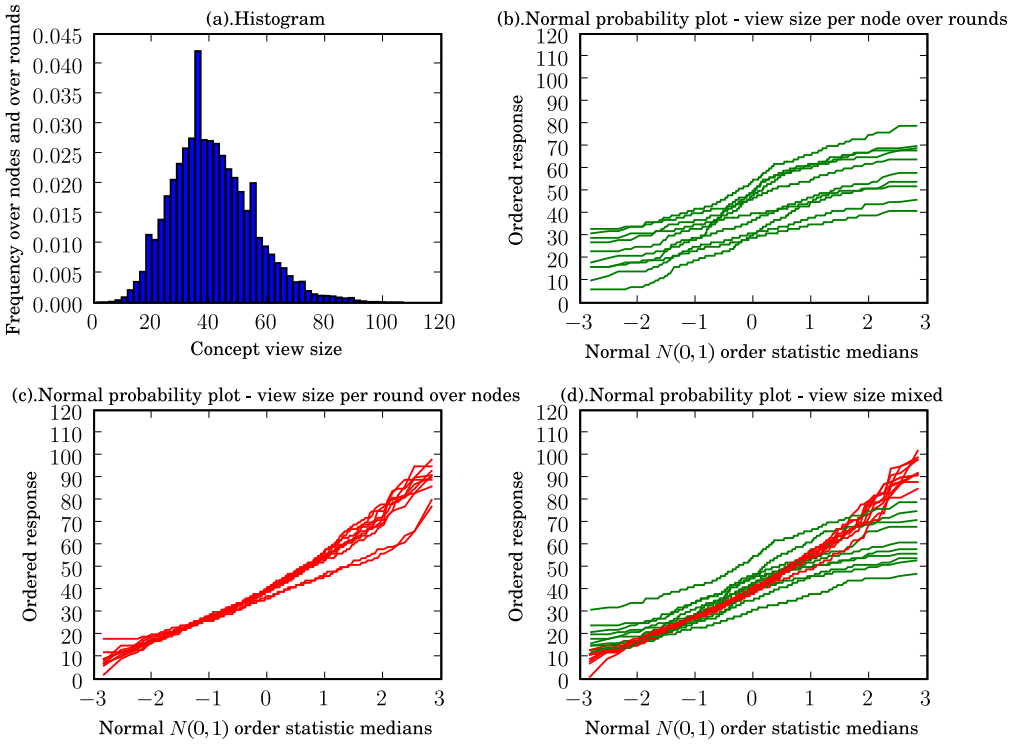


**Fig. 3.6**: Various distributions of the concept view size for 300 nodes after 300 rounds. Experiment invariants: $\mathtt{F}_c = 4$, $\mathtt{F}_n = 2$, $\mathtt{T}_t = 2$, $\mathtt{T}_a = 2$.

Figure 3.6 illustrates different facets of the size distribution through an experiment with 300 nodes running for 300 rounds with gossip parameters of $\mathtt{F}_c = 4$, $\mathtt{F}_n = 2$, $\mathtt{T}_t = 2$ and $\mathtt{T}_a = 2$. The number of nodes was deliberately chosen equal to the number of rounds, so as to validate the assumption of similar distribution across nodes and across rounds. The experiment produced a sample of 90000 data points, since each node recorded the size of its concept view after each transmission. This constitutes the *overall* sample. The overall sample was decomposed in two ways; first, using the recordings of *each node over all rounds*

and second, using the recordings in *each round over all nodes*. The first 30 recordings of each node were discarded, so the first decomposition produced 300 samples, each containing 270 concept view size recordings, while the second decomposition produced 270 samples, each with 300 size recordings. The histogram in (a) is derived from the overall sample. It shows that the size of the concept view over nodes and over rounds follows an approximately normal distribution and provides a graphical assessment for the mean and the variance of the concept view size under the characteristic parameters of the gossip protocol. The histogram is complemented with three normal probability plots (b), (c) and (d) displaying the measure of normality in the two decompositions. Probability plot (b) illustrates the assumption of the normal distribution when each sample is produced from the per node recordings. Ten nodes were then selected uniformly at random and their view sizes were plotted. In (c), the probability plot illustrates the assumption of normal distribution when each sample is constructed from the per round recordings. Similarly, ten rounds are randomly chosen and plotted. Finally, probability plot (d) merges (b) and (c) in order to better illustrate the properties between the two different recordings. The means of the two samples are similar and hence they are also similar to the mean of the overall sample. However, the variance is greater in the per node concept view size distribution. It is possible that with such a large number of nodes, a probability plot with less variance can only be obtained if a larger number of rounds is used, however this is a computationally expensive process.

The stochastic model provides an analytical distribution that approximates both experimental distributions as derived from the per node and the per round samples. This implies that the size distribution of a single node over time is an indicator for all nodes as well as an indicator for the distribution of the concept view size in a single round across all nodes. Therefore, the behaviour of the gossip protocol is observed from a node that is selected uniformly at random from the nodes in $\mathcal{M}$.

Four assumptions that will be used during the analysis are presented below.

$\mathcal{A}1$)*Synchronous gossip transmission.* The stochastic analysis assumes that all nodes transmit in synchronous intervals. Specifically, each node in $\mathcal{M}$ transmits at every round a gossip message to $\mathtt{F}_n$ other nodes. Node failures or message omissions are not taken into

account and it is assumed that message latency between all nodes is negligible. It can be inferred that messages sent in a round will be received within the same round.

$\mathcal{A}2$)*Simple random sample.* The second assumption is that concepts in both the concept and ontology views represent a simple random sample selected from the set of all concepts. This assumption is verified by the discovery results in Section 4.6 and it means that the set of concepts in each node's $\mathcal{V}^{\mathtt{C}}$ and $\mathcal{V}^{\mathtt{C}} \cup \mathcal{V}^{\mathtt{O}}$ represent a simple random sample from $\bigcup \mathcal{V}_i^{\mathtt{O}}, \forall i \in \mathcal{M}$.

$\mathcal{A}3$)*Independence of gossip reception and transmission.* The transmission and reception of gossip messages are independent events that happen simultaneously. In practice, transmission and reception of gossip messages have an arbitrary order within a round and occur sequentially. However, treating them as simultaneous events allows the composition of reception and transmission into a single well-ordered gossip action. This is shown in Fig. 3.7. Such an approximation brings subtle differences to the computation of the insertion and removal probabilities. For the analysis, both insertion and removal are computed based on the view size of the previous round, rather than the view size after the last insertion or removal of concepts. This approximation is used to simplify modelling.

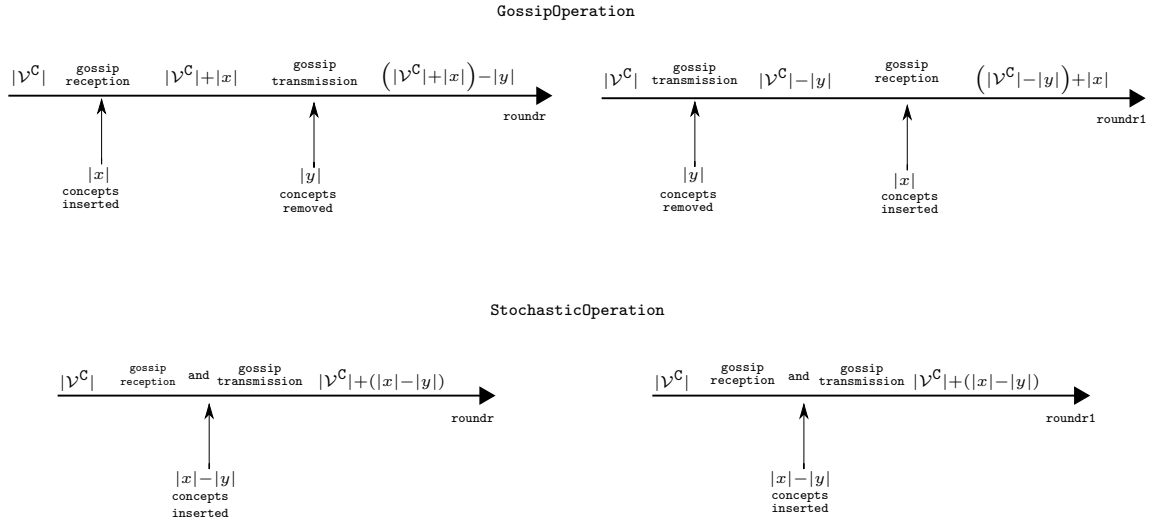


**Fig. 3.7**: Ordering differences between the gossip protocol and its modelling counterpart (Assumption $\mathcal{A}3$). The concept view size is shown after each concept insertion and removal event. For modelling, the two events are coalesced into a single one. This brings differences to the computation of the insertion and removal probabilities as they depend upon the view size. Note that $x, y \in \mathcal{P}(\{0, \ldots, \mathtt{F}_c\})$, where $\mathcal{P}$ is the powerset of $\{0, \ldots, \mathtt{F}_c\}$.

$\mathcal{A}4$)*Sequential insertion of concepts when* $\mathtt{F}_n > 1$. The last assumption concerns the reception of multiple gossip messages from different senders, i.e., $\mathtt{F}_n > 1$. When $\mathtt{F}_n > 1$, we show that on average $\mathtt{F}_n \cdot \mathtt{F}_c$ concepts are received by each node in a single round. This assumption excludes the case where similar concepts may exist in the different $\mathtt{F}_n$ transmissions, essentially treating this case similar to the reception of a single gossip message containing $\mathtt{F}_n \cdot \mathtt{F}_c$ *distinct* concepts.

### 3.6.2 Stochastic Model Specification

The stochastic model considers a network of nodes with empty concept views at the initial round $r = 0$. Let $V_r$ be a random variable that represents the concept view size of an arbitrary chosen node, so that $V_r$ models $|\mathcal{V}^{\mathtt{C}}|$ when $r \geq 0$. From the gossip specification, $V_r$ has range $\mathcal{E}_V = \{0, 1, \ldots, \mathtt{G}_c\}$ where $\mathtt{G}_c = \mathtt{G}_{total} - \mathtt{G}$. The range includes the values from zero to the total number of concepts minus the number of concepts in the node's ontology view. The sequence of random variables $\{V_r\}_{r \geq 0}$ can now be considered as a discrete-time and finite-space Markov chain that has $\mathcal{E}_V$ as its *state space*.

Markov chains are characterised by a set of states and a transition matrix representing the probability of transitioning from one state to another. The stochastic model is described in terms of the state probability vector, the transition probability matrix and the probabilities composing the transition probability matrix.

#### State probability vector

The states in a Markov chain represent the set of all potential source and destination states that a stochastic process can transition from and to. In the current model, the state space represents all applicable concept view sizes in unit increments. To represent the state probabilities at round $r$ we use the vector

$$\mathbf{p}^{(r)} = \left[\, p_0^{(r)} \;\ldots\; p_{\mathtt{G}_c}^{(r)} \,\right]' \tag{3.1}$$

where $p_i^{(r)} = P[V_r = i]$ for $i \in \mathcal{E}_V$ is the probability that the concept view has size $i$ *after* round $r$. The vector notation is a compact representation, which allows the use of the position

index $i$ to represent a state space value, while the corresponding vector value $p_i^{(r)}$ represents the probability for state $i$. As an initial condition $p_0^{(0)} = P[V_0 = 0] = 1$ is assumed.

A desirable property for vector $\mathbf{p}^{(r)}$ is the *stationary* property [Yates and Goodman, 1999]. If the stationary vector for $\mathbf{p}^{(r)}$ exists, the state probabilities reach a steady-state and the stationary vector represents the required probability measure of the concept view size. Since the stationary vector can be mapped to the probability mass function $f_V(v)$, then if it shown that $\mathbf{p}^{(r)}$ has the stationary property, $f_V(v)$ will have been derived.

**Transition probability matrix**

Given a random node and its concept view having size $V_r = i$ after round $r$, the transition to size $V_{r+1} = j$ after round $r+1$, where $j \leq i$ or $j > i$, is subject to the following conditions:

- $\mathcal{C}1$. the number of concepts received by the node,
- $\mathcal{C}2$. the number of concepts inserted in the concept view,
- $\mathcal{C}3$. the number of concepts removed from the concept view because of the node's transmission and the age parameter threshold.

Condition $\mathcal{C}1$ is approximated by computing the mean number of received concepts in each round. Under the assumptions of uniform node views and synchronous gossip transmission ($\mathcal{A}1$), each node will receive on average $\mu_c$ concepts from $\mathtt{F}_n$ gossip transmissions. The computation of $\mu_c$ uses the intuition that during each round the number of concepts arriving at a given node can be calculated by (i) the concept fanout $\mathtt{F}_c$, (ii) the number of transmitting nodes minus the given node (since a node does not gossip to itself), $\mathtt{M}-1$, (iii) the probability that the receiver exists in the node view of a transmitting node, $P_{inNodeView}$ and (iv) the probability that the receiver exists in the node fanout of the transmitting node, $P_{inNodeFanout}$:

$$\begin{aligned}
\mu_c &= \overbrace{\mathtt{F}_c}^{\text{i}} \cdot \overbrace{(\mathtt{M}-1)}^{\text{ii}} \cdot \overbrace{P_{inNodeView}}^{\text{iii}} \cdot \overbrace{P_{inNodeFanout}}^{\text{iv}} \\
&= \mathtt{F}_c \cdot (\mathtt{M}-1) \cdot \frac{\binom{\mathtt{M}-2}{|\mathcal{V}^{\mathtt{N}}|-1}}{\binom{\mathtt{M}-1}{|\mathcal{V}^{\mathtt{N}}|}} \cdot \frac{\binom{|\mathcal{V}^{\mathtt{N}}|-1}{\mathtt{F}_n-1}}{\binom{|\mathcal{V}^{\mathtt{N}}|}{\mathtt{F}_n}} \qquad (3.2) \\
&= \mathtt{F}_c \cdot (\mathtt{M}-1) \cdot \frac{|\mathcal{V}^{\mathtt{N}}|}{\mathtt{M}-1} \cdot \frac{\mathtt{F}_n}{|\mathcal{V}^{\mathtt{N}}|} = \mathtt{F}_c\,\mathtt{F}_n
\end{aligned}$$

To compute $P_{inNodeView}$ and $P_{inNodeFanout}$, it suffices to formulate the inclusion probability of a receiving node existing in the node view and the node fanout of a sender node. For $P_{inNodeView}$, this is the probability that a node exists in $\mathcal{V}^{\mathtt{N}}$, if $|\mathcal{V}^{\mathtt{N}}|$ node identifiers are randomly selected from a set of $\mathtt{M}-1$ identifiers. Similarly, $P_{inNodeFanout}$ is the probability that a node exists in the set composed of $\mathtt{F}_n$ identifiers, if $\mathtt{F}_n$ of these identifiers are randomly selected from the node view. As shown in Eugster et al. [2003], uniform node views imply that $\mu_c$ is not dependent on the size of the node view, $|\mathcal{V}^{\mathtt{N}}|$. Although Section 3.5.5 has shown that node view size affects uniformity and hence would effect the calculation of $\mu_c$, increasing the node view size bypasses this problem.

Having calculated the mean number of concepts that each node receives in a round, it can be inferred that the variability of the concept view size between two rounds ranges from $-\mathtt{F}_c$ to $\mu_c$. This variability lies in the interval of the boundary cases where no concepts are inserted and $\mathtt{F}_c$ concepts are removed and all $\mu_c$ received concepts are inserted but no concepts are removed. Note that this is not the variation for the corner cases of $V_r < \mathtt{F}_c - 1$ and $V_r > \mathtt{G}_c - \mu_c - 1$ as the concept view size cannot be negative or greater than $\mathtt{G}_c$.

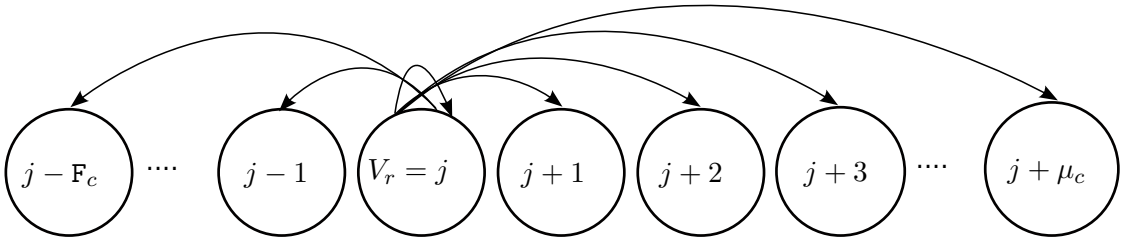


**Fig. 3.8**: States and corresponding transitions for a Markov chain representing a subset of concept view sizes.

Figure 3.8 illustrates more clearly the possible transitions of the concept view size after a round. Each of these transitions can be represented by a measured event that has zero or more outcomes. Two numbers compose each outcome, the number of concepts inserted in the view and the number of concepts removed from the view. These two numbers correspond to conditions $\mathcal{C}2$ and $\mathcal{C}3$.

For example, the transition to the state of maximum view size reduction occurs when there are zero insertions and $\mathtt{F}_c$ concept removals, while moving to the state representing the

maximum increase happens when there are zero removals and $\mu_c$ concept insertions.

The transition to other states can be computed in a similar way, remembering that most transitions are represented by multiple outcomes. An increase of one in the view size can happen because two concepts are inserted and one is removed or three concepts are inserted and two are removed, etc.

This relationship is revealed if the difference in view sizes between consecutive rounds, i.e., $V_{r+1} - V_r$, is further decomposed into two random variables. Each random variable represents an outcome, with $X$ denoting the number of inserted concepts during a round ($\mathcal{C}2$) and $Y$ denoting the number of concepts removed during the same round ($\mathcal{C}3$). The two random variables have ranges $\mathcal{E}_X = \{0, 1, \ldots, \mu_c\}$ and $\mathcal{E}_Y = \{0, 1, \ldots, \mathtt{F}_c\}$. It follows that $X - Y = V_{r+1} - V_r$.

As stated in the assumption about the independence of gossip reception and transmission ($\mathcal{A}3$), insertion and removal are considered as simultaneous and independent events. Let $P_X(x, i) = P[X = x | V_r = i]$ and $P_Y(y, i) = P[Y = y | V_r = i]$ be used to express the probabilities that $x$ concepts are inserted and $y$ concepts are removed when the concept view size is $i$. The probability that a view change is $V_{r+1} - V_r$ can then be computed by using the sum of products between $P_X(x, i)$ and $P_Y(y, i)$ for all $x \in \mathcal{E}_X$ and $y \in \mathcal{E}_Y$ where $x - y = V_{r+1} - V_r$.

The view change probability can be generalised to express the transition probability for the Markov chain as:

$$
\begin{aligned}
P_{ij} &= P[V_{r+1} = j \,|\, V_r = i] \\
&= \begin{cases}
\displaystyle\sum_{\substack{\forall x \in \mathcal{E}_X, \forall y \in \mathcal{E}_Y : \\ x - y = j - i}} P_X(x, i)\, P_Y(y, i) & \text{if } i - \mathtt{F}_c \leq j \leq i + \mu_c \\
0 & \text{if } j < i - \mathtt{F}_c \text{ or } j > i + \mu_c
\end{cases}
\end{aligned}
\tag{3.3}
$$

For the cases where $V_r < \mathtt{F}_c - 1$ and $V_r > \mathtt{G}_c - \mu_c - 1$, the ranges of $Y$ and $X$ are respectively reduced to the appropriate bound so that 0 and $\mathtt{G}_c$ are never exceeded.

Consider a stochastic model with $\mathtt{G}_c = 10$ having $\mathcal{E}_X = \{0, 1, 2, 3\}$ and $\mathcal{E}_Y = \{0, 1\}$. The probability that the concept view size remains at zero can be computed as the probability

that zero concepts are inserted and zero removed *or* one concept is inserted and one concept is removed. The resulting transition probability can be written as $P_{00} = P_X(0,0)\,P_Y(0,0) + P_X(1,0)\,P_Y(1,0) = \sum_{\substack{\forall x \in \mathcal{E}_X, \forall y \in \mathcal{E}_Y: \\ x-y=0}} P_X(x,0)\,P_Y(y,0)$.

The transition probability matrix for the above example can be constructed from equation (3.3):

$$\mathbf{P} = \begin{bmatrix} P_{00} & P_{01} & P_{02} & 0 & 0 & \ldots\ldots & 0 \\ P_{10} & P_{11} & P_{12} & P_{13} & 0 & \ldots\ldots & 0 \\ 0 & P_{21} & P_{22} & P_{23} & P_{24} & \ldots\ldots & 0 \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\ \ldots\ldots & P_{k(k-1)} & P_{kk} & P_{k(k+1)} & P_{k(k+2)} & \ldots\ldots & \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\ 0 & \ldots & 0 & P_{8,7} & P_{88} & P_{89} & P_{8(10)} \\ 0 & \ldots & 0 & 0 & P_{98} & P_{99} & P_{9(10)} \\ 0 & \ldots & 0 & 0 & 0 & P_{(10)9} & P_{(10)(10)} \end{bmatrix} \tag{3.4}$$

The transition matrices formed from equation (3.3) have all similar structure to the matrix in (3.4). This similarity will be used to show that the transition matrices formed are irreducible and aperiodic and therefore a stationary distribution exists as $r \to \infty$.

Before proceeding to compute the values $P_{ij}$ of the probability matrix by forming a probability measure for $Y$ and $X$, we define the convenience function $\zeta(i) = \mathtt{G} + i$ to return the number of concepts in a node's ontology and concept views when the concept view has size $i$. Here, $i$ can also be considered as a row index in the transition matrix generated from equation (3.3). The concept removal probability is presented first, since it involves a much simpler analysis, followed by the more complex analysis for the concept insertion probability.

### Concept removal probability

The concept removal probability is easily calculated assuming that concepts with different age values are uniformly distributed in the concept view. The probability of removing a concept from the concept view is the probability that the selected concept is picked from the concept view and that its age value is $\mathtt{T}_a - 1$.

The probability of removing $y$ concepts, when the concept fanout is $\mathtt{F}_c$ can be calculated as:

$$P_Y(y,i) = \frac{\binom{\lceil i/\mathtt{T}_a \rceil}{y}\binom{\lfloor \zeta(i)-(i/\mathtt{T}_a)\rfloor}{\mathtt{F}_c - y}}{\binom{\zeta(i)}{\mathtt{F}_c}} \quad \text{for } y \in \mathcal{E}_Y \tag{3.5}$$

Equation (3.5) expresses the probability of selecting $y$ concepts from the subset of concepts in the concept view that have an age value of $\mathtt{T}_a - 1$, while the remaining $\mathtt{F}_c - y$ are selected either from the ontology view or from the subset of concepts having an age value different to $\mathtt{T}_a - 1$.

### Concept insertion probability

To calculate the concept insertion probability, it is necessary to identify the number of received concepts that ($\mathcal{F}1$) have a ttl value that is greater than one and ($\mathcal{F}2$) do not exist in the concept and ontology views of the receiver. Since both ($\mathcal{F}1$) and ($\mathcal{F}2$) depend on the sender of the gossip message, this section considers two distinct nodes with the roles of a sender and a receiver in a hypothetical gossip transmission. Both nodes are selected uniformly at random from $\mathcal{M}$. Let $\eta(i)$ be a function that returns the number of concepts in the sender's views that have ttl value different to one and do not exist in the receiver's views.

The probability of inserting $x$ concepts having received $\mu_c$ can be calculated as:

$$P_X(x,i) = \frac{\binom{\lceil \eta(i) \rceil}{x}\binom{\lfloor \zeta(i)-\eta(i)\rfloor}{\mu_c - x}}{\binom{\zeta(i)}{\mu_c}} \quad \text{for } x \in \mathcal{E}_X \text{ and } \mu_c < \zeta(i), \mu_c < \eta(i) \tag{3.6}$$

Equation (3.6) expresses the probability that $x$ concepts will be selected from $\eta(i)$ concepts, while the remaining $\mu_c - x$ concepts will either have a ttl value of one or exist in the receiver's views and therefore will not be inserted. This probability distribution assumes that all $\eta(i)$ concepts are uniformly distributed across the $\zeta(i)$ concepts.

In (3.6), insertion into the receiver's concept view is related to the selection of concepts during transmission. When $\mathtt{F}_n = 1$, from (3.2), $\mu_c = \mathtt{F}_c$ and thus one sender easily maps to one receiver. When $\mathtt{F}_n > 1$, under the assumption of sequential insertion ($\mathcal{A}4$), a single sender can still be related to a single receiver by having the sender select $\mu_c$ concepts for transmission, while the receiver selects $\mathtt{F}_c$ concepts. This introduces an asymmetry and is counter to the gossip protocol, which does not distinguish sender from receiver. However, its use greatly simplifies the stochastic analysis.

### Computation of $\eta(i)$

The computation of $\eta(i)$ requires the calculation of two factors:

- $\mathcal{F}1$, the distribution of ttl values across the ontology and concept views of a single node,
- $\mathcal{F}2$, the distribution of replicated concepts between a sender and a receiver.

Since both of these factors depend on the size of the concept view, the calculation of ($\mathcal{F}1$) and ($\mathcal{F}2$) must take place for all concept view sizes.

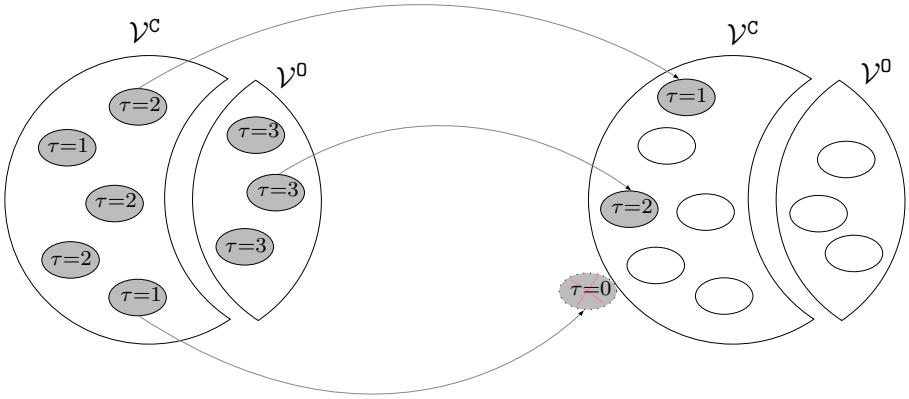


**Fig. 3.9**: The distribution of ttl values across concepts in a concept view is proportional to the ttl distribution across concepts in *both* views of the previous round.

To find the number of concepts with a ttl value different to one, the following approach is used. Knowing that concepts in the ontology view have a ttl value of $\mathtt{T}_t$, the concept view will contain concepts with ttl values from one to $\mathtt{T}_t - 1$. We rely on the intuition that in each

round, the number of concepts with a ttl value of $\tau$ is proportional to the number of concepts with a ttl value of $\tau + 1$ in the previous round. Figure 3.9 illustrates the process.

Since the probability matrix only incorporates the notion of concept view size and not of rounds, a method is required to map the ttl distribution from rounds to matrix rows. This is achieved by assuming that the ttl distribution, when the concept view has size $i$, derives from the ttl distribution of the immediately preceding size $i - 1$. In other words, this method replaces gossip rounds with unit increments in the size of the concept view. This approximation yields surprising accuracy. A recurrent function, $g_i$, can now be used to compute the number of concepts having ttl value $\tau$ when the size of the concept view is $i$. Function $g_i$ has domain and range:

$$g_i : \{1, \ldots, \mathtt{T}_t\} \rightarrow \{0, \ldots, max\{\mathtt{G}, i\}\}$$

and specified as:

$$g_i(\tau) = \begin{cases} 0 & \text{if } i = 0 \text{ and } \tau \in \{1, \ldots, \mathtt{T}_t - 1\} \\ \mathtt{G} & \text{if } i \in \{0, \ldots, \mathtt{G}_c\} \text{ and } \tau = \mathtt{T}_t \\ i \cdot \dfrac{g_{i-1}(\tau + 1)}{\sum_{\kappa=2}^{\mathtt{T}_t} g_{i-1}(\kappa)} & \text{if } i \in \{1, \ldots, \mathtt{G}_c\} \text{ and } \tau \in \{1, \ldots, \mathtt{T}_t - 1\} \end{cases} \tag{3.7}$$

Equation (3.7) computes the number of concepts with ttl value $\tau$, when the size of the concept view is $i$. It does so by assuming that for each view size, the distribution of ttl values from one to $\mathtt{T}_t - 1$ derives recursively from previous view sizes, terminating by giving the size of the ontology view ($\mathtt{G}$) when $\tau = \mathtt{T}_t$. By calculating the ratio of concepts with ttl value $\tau$ for the previous view size, the normalising factor of the current concept view size $i$ is used to adjust the ttl ratio to the current view size. The denominator in the last case of equation (3.7) expresses the constraint that concepts with a ttl value of 1 cannot be inserted in the receiver's view. This is used to derive an accurate ratio of ttl values from previous view sizes that will be used for the distribution of the current view size.

To compute the probability that a received concept exists in the receiver's views (i.e., either the concept or the ontology view), we first need to calculate the number of identical (replicated) concepts in both views between the sender and the receiver. Using an overall

set to represent all concepts, under the simple random sample assumption ($\mathcal{A}2$), two distinct subsets of equal size selected randomly and with replacement from this overall set can be used to represent each node's concepts. Note that this includes the concepts from the union of both views. The probability distribution of identical concepts between the two subsets can now be calculated, further assuming that the selection of concepts for each subset occurs without replacement.

The selection of the two equal-sized concept subsets makes the assumption that the concepts views of the sender and the receiver have the same size. This is necessary in order to avoid the large number of permutations that would result from taking into account size variations between senders and receivers. These permutations would lead to a combinatorial explosion thus making the analysis infeasible.

Let the random variable $Z$ represent the number of replicated concepts between two nodes. The range of $Z$ is $\mathcal{E}_Z = \{0, \ldots, \zeta(i)\}$ and its probability distribution can be expressed as:

$$P_Z(z, i) = \frac{\binom{\zeta(i)}{z}\binom{\mathtt{G}_{total}-\zeta(i)}{\zeta(i)-z}}{\binom{\mathtt{G}_{total}}{\zeta(i)}} \quad \text{for } z \in \mathcal{E}_Z \tag{3.8}$$

Equation (3.8) expresses the probability that $z$ concepts are identical when both the sender's and the receiver's views have size $\zeta(i)$. We derive the distribution knowing that $\zeta(i)$ concepts have already been selected from $\mathtt{G}_{total}$ for the first subset and formulating the probability distribution of identical concepts based on the second subset. Since the size of the second subset is also $\zeta(i)$, the probability distribution is computed by having the second subset be composed of $z$ items chosen from the first subset, therefore considered identical, while the rest $\zeta(i) - z$ are selected from the remaining set of all available concepts. The expected value $E[Z]$ is a good approximation to the mean number of identical concepts in both views between the sender and the receiver.

Assuming that the expected number of identical concepts are spread uniformly across the two views in each node, the probability that a transmission contains a certain number of identical concepts can be easily determined.

Factors $\mathcal{F}1$ and $\mathcal{F}2$ can now be expressed from equations (3.7) and (3.8) resulting in the following expression for $\eta(i)$:

$$\eta(i) = \sum_{\tau=2}^{\mathtt{T}_t} g_i(\tau) - \left( \frac{\sum_{\tau=2}^{\mathtt{T}_t} g_i(\tau)}{\zeta(i)} \cdot E[Z] \right) \tag{3.9}$$

Equation (3.9) computes the number of concepts in the sender's views that do not exist in the receiver's views and have ttl value different to one. When using $\sum_{\tau=2}^{\mathtt{T}_t} g_i(\tau)$ to calculate the number of concepts that have ttl value different to one, the result also contains a number of concepts that are identical (since identical concepts are spread uniformly across the two views). The second part of the formula computes this proportion, which is then subtracted from the first estimation to identify a more accurate result. Function $\eta(i)$ tends to overestimate the number of replicated concepts for low values of $i$. This is because the distribution $P_Z$ does not distinguish between concepts in the ontology view, which are unique amongst nodes, but instead treats all $\zeta(i)$ concepts as selected from the set of all concepts.

**Stationary distribution**

Theorem 12.12 in Yates and Goodman [1999] shows that a vector has a stationary probability vector, if the associated Markov chain is *irreducible*, *aperiodic* and *finite*. This theorem will be used to show that $\mathbf{p}$ from equation (3.1) is stationary.

An irreducible Markov chain has the property that it is possible to go from any state to any other state, potentially in more than one transition. In addition, a Markov chain is aperiodic if all states have a period of 1.

By construction the considered probability matrix derived from equation (3.3) is finite. It can be shown that the Markov chain is irreducible and aperiodic, if for some power of the matrix, all its elements are positive.

Each transition matrix constructed under the current model has elements that have either positive or zero values. Each matrix is formulated with a set of well defined rules, with the following property:

$$\mathbf{P}[ij] = \begin{cases} > 0 & \text{if } i - \beta < j \leq i + \alpha \\ 0 & \text{otherwise} \end{cases} \tag{3.10}$$

Note, that $\mathbf{P}[ij]$ represents the matrix element at position $ij$, which has a value calculated from $P_{ij}$. Specifically, $\alpha$ corresponds to $\mu_c$ and $\beta$ to $\mathtt{F}_c$. When $\alpha > 1$ and $\beta > 1$, it is easy to show that for some $r > 1$, the transition probability matrix $\mathbf{P}^r$ will have all its elements positive.

Knowing that the state probability vector $\mathbf{p}^{(r)}$ results in a stationary vector and having formulated the transition matrix, the probability distribution of $V$ is the stationary vector $lim_{r\to\infty}\mathbf{p}^{(r)}$, which can be computed from:

$$\mathbf{p}^{(r)} = \mathbf{p}^{(r-1)}\,\mathbf{P} \tag{3.11}$$

## 3.7 Evaluation of Stochastic Analysis

This section presents the evaluation of the stochastic properties of the gossip protocol. The analytical results are compared against two types of simulation. One uses perfect uniform views with no message failures ("perfect views"), while the second is based on the ns2 mobile ad hoc network simulator [McCanne and Floyd, 1997]. A detailed setup of the ns2 simulation environment is given in Chapter 6. The aim of this section is to study:

- the accuracy of the analytical model against the simulation of the gossip protocol,
- the impact of view uniformity,
- the impact of message loss.

The perfect views simulation uses an implementation of the gossip protocol where each node view maintains the full membership. Variability in view uniformity is therefore not an issue. Furthermore, the network simulation layer is removed, resulting in no transmission and receive failures. This additional simulation environment satisfies the stated purpose of demonstrating how much variability is introduced because of failures and imperfect views.

The large number of parameters that characterise both the gossip protocol and the mobile ad hoc environment (population size, mobility, traffic and failure parameters) make it difficult to exhaust all possible permutations in an experimental evaluation. A characteristic set of parameters was therefore chosen and each experiment used a parameter permutation from

| $\lvert\mathcal{V}^0\rvert$ | $\mathtt{F}_n$ | $\mathtt{F}_c$ | $\mathtt{T}_a$ | $\mathtt{T}_t$ | $\lvert\mathcal{M}\rvert$ |
|---|---|---|---|---|---|
| 10 | $1, 2$ | 4 | $1, 2$ | $2, 3$ | $20, 40, 60$ |

**Table 3.1**: Summary of gossip parameters for stochastic evaluation.

this set. Experimental results are presented graphically in this section and also numerically in Appendix A. Overall, 40 experiments were executed using the perfect views simulator and 24 using ns2. Each experiment was repeated five times.

Simulation experiments and analytical results were obtained against the same set of parameters. The number of participants varied from 20 to 100 nodes. Experiments with the ns2 simulation were only conducted to a maximum of 60 nodes, since the high rate of message loss ($> 20\%$) that occured in larger networks prevented the correct functioning of the protocol. The varied gossip parameters were node fanout ($\mathtt{F}_n$), age ($\mathtt{T}_a$), and the concept ttl ($\mathtt{T}_t$), while the concept fanout ($\mathtt{F}_c$) and the ontology view size ($|\mathcal{V}^0|$) were kept constant at four and 10 concepts correspondingly. Table 3.1 summarises the parameter sets.

Figures 3.10 and 3.11 compare the results between the two simulation types and the stochastic analysis. For both simulation types, the 95% confidence interval is calculated over 350 rounds of protocol execution. In each experiment, all nodes record the size of the concept view after every transmission. To illustrate results closer to the steady state distribution of the concept view, the initial 30 recordings are discarded. Each point in the graphs represents the median value from a sequence of average concept view sizes. For example, in the case where the per node over rounds distribution is measured, each experiment results in $|\mathcal{M}| \times 5$ samples, each containing 320 (350 total rounds - 30 initial rounds) data items representing concept view sizes. From each sample, the average and standard error is extracted and the median of these averages is finally selected and plotted. For the stochastic analysis, what is depicted is the expected value of the state vector ($E[V]$) after the Markov chain has converged.

The difference between the two figures is in the chosen sample. In Fig. 3.10, each sample contains concept view sizes that are recorded over all gossip rounds. In Fig. 3.11, each sample is composed from the per round snapshot of concept view sizes over all nodes.

For each parameter permutation, the stochastic analysis gives an accurate prediction of the behaviour of the gossip protocol. The expected values obtained from the analysis are closer to the mean values of the perfect views simulation, while the ns2 simulation presents greater variability. Between the analytical results and the two simulations the source of variability is mainly the approximations that were made during the derivation of the transition matrix. This may explain why certain parameter permutations are more closely aligned than others.

The variability in the ns2 simulation has two likely causes: imperfect node views and failed transmissions. Both must be seen in context of the already high statistical variability, especially in the case where the stochastic model is a predictor for the per node variation of the concept view (Fig. 3.10). Figure 3.6 has also illustrated that such samples have greater variability.

Imperfect uniform views can affect concept view variation by favouring one subset of nodes against another. Although fixing the node views to a size that is half the node population results in good uniformity properties, residual noise can still affect uniform characteristics. Specifically, since node views are fixed in size, the frequency of occurrence of certain nodes may increase amongst the node views of the participants, causing more transmissions to be directed to those nodes and fewer transmissions to nodes with lower frequency. One source of this variation is the node view maintenance protocol that was described in Section 3.5.5. However, another source of imperfect node views is also the bootstrapping protocol (Section 3.5.1), which cannot always guarantee initial uniformity.

Message loss affects concept replication by lowering the average replication value. This becomes clear when comparing the perfect views simulation against ns2. In each ns2 experiment, there is a general downward trend as network size increases with the mean concept view size in the 60 nodes case always lower than that predicted by the analysis and what is observed with no message failures. This is expected and is related to the transmit-receive dependency of the protocol. A large number of lost messages cause nodes to eventually remove transmitted concepts from their concept views without these concepts being replicated in other nodes. Although the protocol, through the age parameter, can tolerate a certain ratio of message loss, high failure rates cause the performance of the protocol to deteriorate.

Variability is also different between the two concept view size distributions, per node over rounds (Fig. 3.10) and per round over nodes (Fig. 3.11). One influencing factor for this difference is sample size. While in the first case the mean and the standard error are calculated over a sample of 320 values, in the second case the confidence interval is calculated over a sample size that equals the node population ($|\mathcal{M}|$).

Concept replication is tightly linked to the characteristic parameters and network size. For a constant ttl value ($\mathtt{T}_t = 2$), the node fanout and age parameters influence replication in a similar fashion. By doubling either the fanout or the age, the mean concept view size is also doubled, though it remains constant as the number of participants increase. Since the figures display the concept view size in absolute values, the ratio of a constant size against an increasing number of total concepts decreases linearly. This highlights the main scalability trade-off in OntoMobil; with increasing number of nodes, certain gossip parameters (e.g., $\mathtt{F}_n = 1, \mathtt{T}_a = 1, \mathtt{T}_t = 2$) can maintain a constant *concept view size* keeping the processing and memory overhead constant but also reducing the rate of matching and increasing the number of hops for discovery. This is in contrast to maintaining a constant *replication ratio* at each node with increasing the total number of concepts and nodes. This would give a constant matching rate and a constant number of hops during discovery at the expense of an ever growing concept view size.

With a low ttl value, different parameter combinations can be used under different network conditions. Using a low node fanout, e.g., $\mathtt{F}_n = 1$, reduces communication overhead and the replication factor. However, there will also be a corresponding increase in the concept matching latency and the number of hops before a concept is discovered.

If the ttl value is increased to three, the concept view size shows a linear increase as the node population grows. The ttl value is now the controlling parameter resulting in a higher propagation range, which in turn allows each concept to reach a larger number of nodes. Each concept can now infect a higher percentage of nodes before it is removed and can thus be used to reduce the concept matching latency.

## 3.8 Summary

This chapter has described the OntoMobil model and provided a detailed specification and stochastic analysis of the gossip protocol. The protocol forms a randomised semantic overlay that is designed to interact with the progressive ontology matching algorithm and the concept-based discovery protocol. The replication of concepts across the participating nodes is completely characterised by a set of parameters and is amenable to stochastic analysis. The evaluation has confirmed that analytical results are aligned with simulation results.

Further analysis has revealed that the properties of the gossip protocol are sensitive to uniform membership views and transmission failures. Recently published work can be used to address both of these issues. Specifically, the gossip literature now features at least one protocol that provides uniform partial views with provable properties [Bar-Yossef et al., 2006]. Unicast optimisation undertaken in the field of MANET group communication can also be used to reduce the number of failed transmissions as the network size grows [Luo et al., 2003]. Both of these issues, integration with a different membership protocol and utilisation of a more reliable unicast protocol, are considered future work.

At the moment, the number of nodes and the number of characteristic parameters require manual setting. Moving closer to the vision of ad hoc opportunistic interaction would require a higher layer protocol to set and adapt the values of these parameters depending on the network size, application and node requirements.

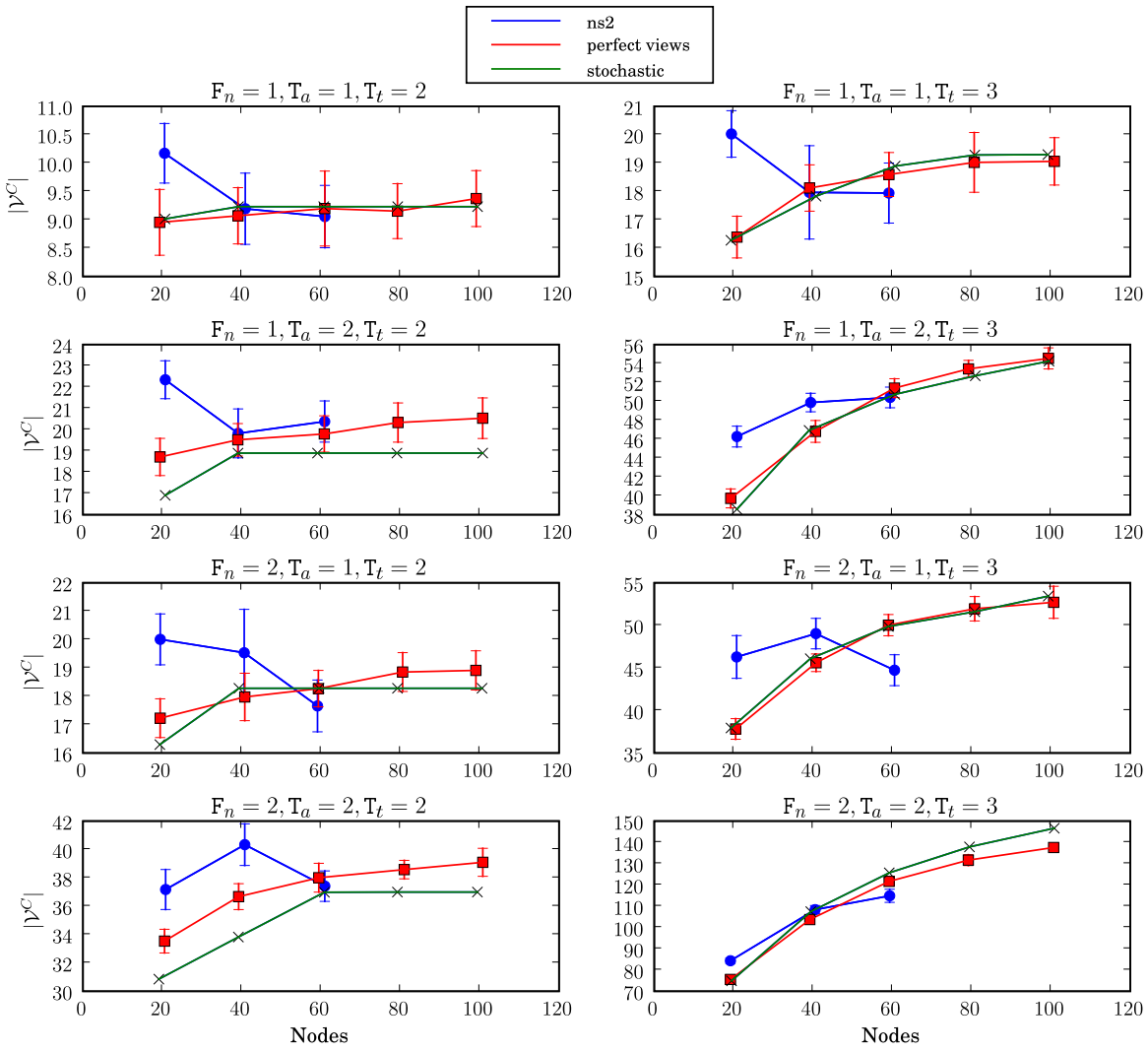


**Fig. 3.10**: Comparative results between the stochastic analysis, the simulation with perfect views and no failures, and the ns2 simulation. Each point in the graphs represents the median from a list of sample means, where each sample contains concept views sizes taken **per node and over all rounds**. Experiment invariants: $\mathtt{F}_c = 4, |\mathcal{V}^0| = 10$.

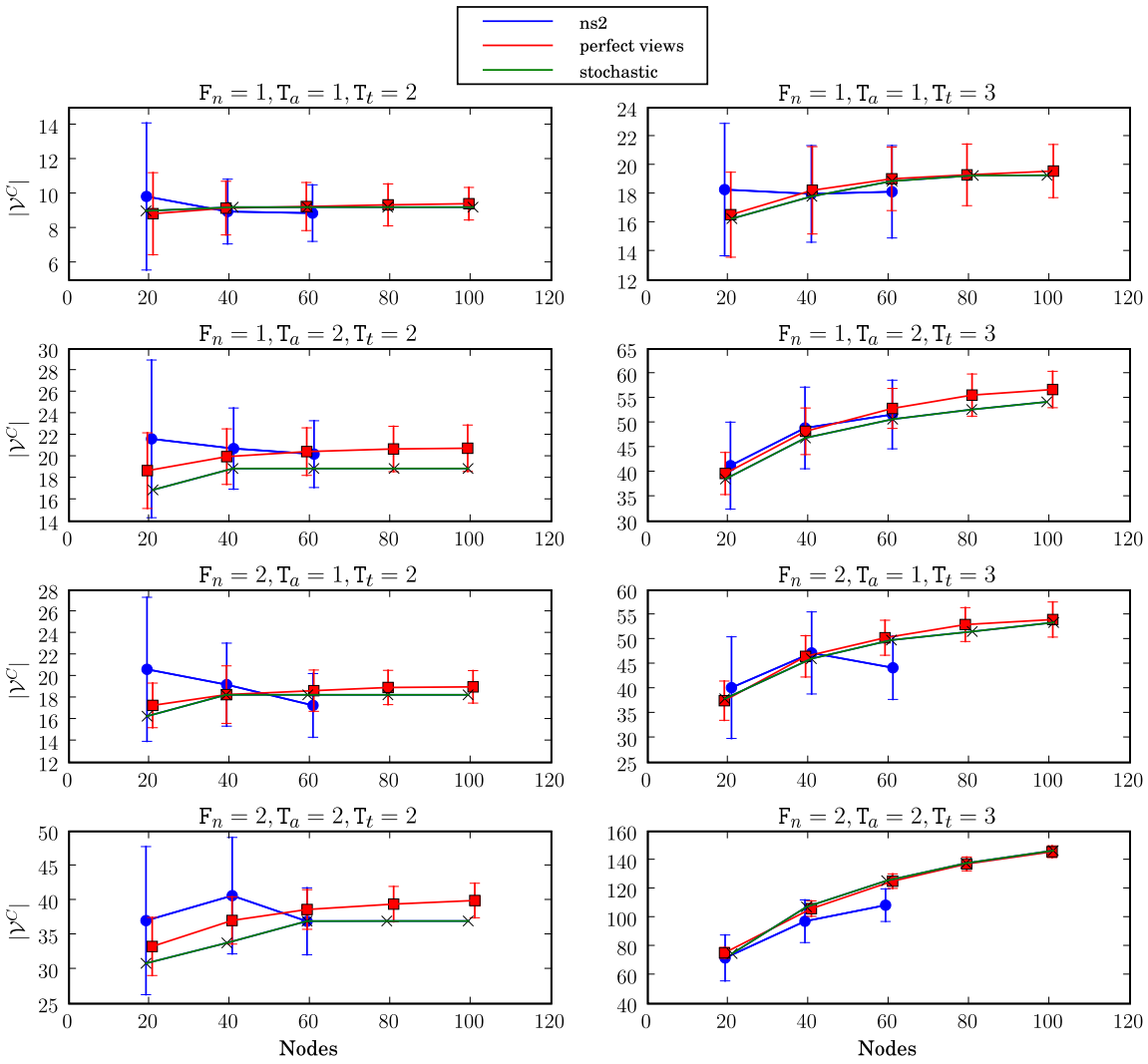


**Fig. 3.11**: Similar to Fig. 3.10. Each sample now contains concept views sizes taken **per round and over all nodes**. Experiment invariants: $\mathtt{F}_c = 4, |\mathcal{V}^{\mathtt{O}}| = 10$.

# Chapter 4

# Discovery of Semantic Services

This chapter describes the specification of concept matching and service discovery. The two processes use the gossip protocol to complete the OntoMobil model. Concept matching facilitates eventual semantic agreement, which is necessary when services are described by heterogeneous ontologies. It is a distributed process that uses the concept exchange mechanism provided by the gossip protocol with a matching algorithm in each node. The result is the progressive creation of equivalence relations between the different ontologies. Two different concept matching methods are described, namely *dynamic matching* and *template matching*, which use different assumptions and algorithms. The chapter explores the trade-offs between their resource usage and their matching accuracy.

Assuming a finite set of nodes and concepts and a stable network, the process of concept matching will eventually establish all possible relations. The gossip protocol can then be suspended by all nodes since no additional semantic knowledge can be derived by further disseminating concepts. An initial investigation into such a suspension algorithm is given in this chapter.

This chapter also describes the random-walk service discovery protocol. It uses the semantic overlay built by the gossip protocol to discover concepts and their matching relations. Two variations are presented, based on whether service requests are derived from the ontology of the node issuing the request or from a remote ontology. The chapter concludes by showing how queries based on Description Logics can be decomposed into concept-based queries.

```
<rdf:RDF xml:lang="en"
  xmlns:om="http://www.dsg.cs.tcd.ie/OntoMobil/0.1/"
<om:Service rdf:ID="aService">
  <om:hasInput rdf:resource="#ConceptA"/>
  ...
  <om:hasOutput rdf:resource="#ConceptB"/>
  ...
</om:Service>
</rdf:RDF>
```

a. A service instance.

```
<rdfs:Class rdf:ID="C">
  <om:source rdf:resource="192.168.1.2"
  <om:hasSuperClass rdf:resource="#B"/>
  ...
  <om:hasProperty rdf:resource="P1"/>
  <om:hasProperty rdf:resource"P2"/>
  ...
</rdfs:Class>
```

b. The network representation of a concept.

**Fig. 4.1**: The RDF representation of a service and a concept instance.

## 4.1 Objectives

This chapter has four main objectives. First, the role of concept matching is to identify potential semantic similarities and establish an association between equivalent concepts. Concept matching partly addresses the requirement for discovery when services are represented by heterogeneous ontologies. Second, in a stable network where no existing nodes fail or disconnect and no new nodes join, concept matching will eventually identify all potential semantic relations. A suspension algorithm pauses the gossip protocol in every node in order to reduce the associated communication and processing overhead. The gossip protocol can restart in response to a number of events such as nodes requesting to join or leave the network. Third, the objective of the discovery protocol is the distributed lookup of concepts that compose a service request and the identification of nodes that maintain compatible ontologies with the ontology maintained by the source node of the request. The discovery protocol completes the requirement for locating semantically heterogeneous services. Finally, the fourth objective is to allow queries based on arbitrary Description Logics expressions to be decomposed into a set of concepts, so that OntoMobil can be integrated with more complex and expressive service queries.

## 4.2 Semantic Service Description

The Resource Description Framework Schema (RDFS) [Brickley and Guha, 2000] was chosen to model ontologies and to represent services and queries. Although limited in its modelling constructs, RDFS provides adequate expressiveness for the initial OntoMobil prototype and

simplifies the evaluation scenarios with its intuitive mapping between concepts and RDFS classes. There is currently no service standard in RDFS so a minimal service specification was defined for this thesis, based on the service profile of OWL-S. An example of an RDFS-based service instance is shown in Fig. 4.1a. The schema for this instance is included in Appendix B.

In OWL-S, services are specified and discovered based on an Input, Output, Precondition, Effects (IOPE) tuple. Input and output properties specify the service's behaviour, while pre-condition and effects describe functional service aspects [Li and Horrocks, 2003b]. Capturing functional service requirements usually requires the integration of a rule-based language and is not covered by the current OntoMobil specification. Instead, a simplified version is considered where only the input and output concepts are used for service description and discovery. It is also expected that mobile ad hoc services will be similar to OWL-S atomic processes, which represent services that are invoked and executed in a single step, as opposed to services that require multi-step composition and orchestration before execution.

### 4.2.1 Concept Representation

The premise of semantic services is the description of a service's capabilities using a number of concepts that are combined with a set of semantic operators (*constructors* in the terminology of Description Logics). In the proposed model it is concepts rather than services that are advertised and discovered. However, the decomposition of ontologies into concepts and the distribution of concepts across the network imposes certain requirements on the concept syntax. Ontology languages like RDFS and OWL are not designed for the task of concept decomposition and dissemination so the syntax of advertised concepts had to be augmented. This augmented syntax is called the *network representation* of a concept to distinguish it from the representation that a concept has within an ontology. Figure 4.1b shows an instance of a concept's network representation.

The network representation syntax needs to contain enough information to satisfy three distinct requirements:

1. *Discovery of nodes with compatible ontologies* – The first step to service discovery is

the identification of nodes with compatible ontologies. If the network representation of each concept includes the source node identifier, the discovery protocol can introspect concepts that match those composing the query and use this information to identify any nodes with compatible ontologies. This requirement is satisfied with the inclusion of the `om:source` RDFS predicate. Only a single such predicate per network representation is allowed. Discovery is also aided by the inclusion of references to matching concepts. This allows concepts to aggregate matching information as they propagate across the network, which is then used to locate compatible ontologies in remote nodes. This requirement is satisfied with the inclusion of two RDFS predicates, `om:matchesConcept` and `om:matchesConceptInNode`. The former is a reference to a concept URI, while the latter is a reference to the corresponding node address. The cardinality of this pair of predicates is left unspecified.

2. *Pair-wise concept matching* – To interpret discovery queries that are formed without reference to a single common ontology, OntoMobil relies on concept equivalence relations introduced by the process of concept matching. The problem then is to include enough information in the concept's network representation to make matching produce meaningful equivalence relations. Since properties in most ontology languages are first-class entities and are defined outside the scope of concepts, it means that a standalone concept representation must have explicit support for concept properties. By embedding properties and additional concept information (e.g., super and sub-concepts) in the network representation, concept matching can be facilitated in any participating node. This requirement is satisfied with the inclusion of the `om:hasSuperClass`, `om:hasSubClass`, `om:hasProperty` and `om:hasSuperProperty` RDFS predicates. Section 4.3 details the approach.

3. *Discovery of service providers* – A possible optimisation to avoid visiting nodes that have compatible ontologies but do not provide matching services, is to augment the network representation with service information. Any concept used by a service to define the service's behavioural characteristics, can include an indication in its network

representation that summarises its usage in the corresponding service. Service requests can then examine concepts and only retrieve the source node identifier if the matching concept indicates that it is being used in the definition of a service. Evidently, this approach is only an approximation, which cannot substitute the semantic matchmaking capabilities offered by ontologies.

The current specification of the network representation addresses the first two requirements. A future extension may include support for incorporating service usage indicators.

## 4.3 Concept Matching

OntoMobil lifts the restriction of a single common ontology and in doing so is faced with the challenge of semantic integration. To reach semantic consensus in an efficient manner, OntoMobil uniformly distributes the matching process across all nodes and uses results from the field of ontology matching to establish semantic similarity between heterogeneous ontologies. Since the contribution of OntoMobil is not in the field of ontology matching, the problem of semantic integration has been simplified with two basic assumptions:

- Some connected participants will always maintain related ontologies for similar domains. In other words, not all ontologies are dissimilar.
- Between related ontologies, there is a set of concepts that when compared through the concept matching methods proposed in this section, yield a matching association.

OntoMobil enables a semantic matching architecture that is distributed and proactive (Section 2.1.3). Within the resulting architecture, two different methods are described that can be used to integrate heterogeneous ontologies. The first is the *dynamic matching* method, where each node is responsible for creating matching associations at runtime having no pre-existing semantic knowledge other than what the matching algorithm (H-MATCH) provides. The second is the *template matching* method, where nodes are assumed to maintain, apart from their own ontology, a mapping to a number of other ontologies. Template matching does not require a concept matching algorithm, instead it relies on a scheme to unambiguously identify concepts and ontologies.

| | Properties | | | | | |
|---|---|---|---|---|---|---|
| **Concepts** | **0** | **1** | **2** | **3** | **4** | **5** |
| **10** | 0.2410 | 0.4455 | 0.7162 | 1.0374 | 1.3829 | 1.8425 |
| **20** | 0.5557 | 1.2816 | 2.2467 | 3.3018 | 4.5969 | 6.0430 |
| **30** | 1.0649 | 2.5754 | 4.5777 | 6.9349 | 9.6372 | 12.8218 |
| **40** | 1.6876 | 4.2554 | 7.7056 | 11.5536 | 16.2711 | 21.9438 |

**Table 4.1**: Time in seconds required for the pairwise matching between concepts of two ontologies in a Compaq IPAQ H3870 running linux. Ontologies are described in XML/RDF(S) and the matching algorithm is implemented in Python using the RDFlib library. The matching algorithm is a simple string matching of concept names, properties and their types.

Both methods address the following requirements, which are derived from the constraints of the mobile ad hoc environment and the proposed model. These requirements are generic and should be addressed in case a new matching method is designed. Specifically, matching in OntoMobil should be:

1. Lightweight. The continuous dissemination of concepts by the gossip protocol and the per node matching approach demands processing time by each mobile node. Therefore, limited device resources dictate that any matching algorithm be fast and efficient. Table 4.1 demonstrates the processing cost of concept matching in a mobile device. The table records the time in seconds required to compare two ontologies of progressively increasing number of concepts and properties. In each row, the number of concepts per ontology is constant, while the number of properties per concept increases. In the case where each ontology has size $G$ with no concept properties, the number of comparisons is $G^2$. This simple scenario demonstrates that there is a significant increase in the required time as properties per concept increase.

2. Automated. Since semantic services enable the mechanising of the discovery process, it is imperative that ontology matching is also automated. Otherwise, the transient nature of communication in MANETs and the constant introduction of new nodes and ontologies would quickly turn manual semantic conflict resolution into a laborious and

error prone task [Noy and Musen, 2000].

3. Operating at the schema level. Schema level matching operates solely on metadata without taking into account any metadata instances. Given the gossip protocol, this matching technique is the most appropriate. It usually results in multiple matches with an output that provides a similarity range [Rahm and Bernstein, 2001]. The dynamic matching method proposed here simplifies the process by only considering a boolean similarity output.

4. Working with partial ontologies. Since the OntoMobil approach trades centralised and resource intensive matching with a progressive and distributed process, it is not possible to match complete ontologies. Therefore, instead of ontology matching OntoMobil uses the idea of concept matching.

According to the taxonomy of schema matching approaches defined in Rahm and Bernstein [2001], dynamic matching and template matching can be classified under the *Schema-only based/Element-level* branch. In particular, since the H-MATCH [Castano et al., 2003a] algorithm is highly configurable, it could potentially be classified under either the *Linguistic* or the *Constraint-based* leaves of this taxonomic branch.

The next sections specify the two methods, detail how matching is established and introduce a set of optimisations to reduce the overhead from concept comparison.

### 4.3.1 Dynamic Matching

A runtime matching algorithm executing in each mobile node necessitates a practical approach for identifying concept similarity. Syntactical matching is more appropriate because it requires less resources. Semantic matching on the other hand can produce more accurate integration, but requires complex inferencing over the candidate ontologies. For the initial implementation of OntoMobil, an algorithm similar to *shallow matching* of H-MATCH [Castano et al., 2003a] was used. Each node records a match between two concepts when their respective names are syntactically the same and each of their properties match in type and name. In addition, in order to incorporate publicly available ontologies, matching uses the

following rule: when the ontology namespace between two concepts from different nodes is the same and the concept names are also syntactically the same, then this should be used as a hint that both concepts belong to the same ontology. The ontology namespace is used in the `om:namespace` predicate in each concept's network representation.

There is a dependency between the completeness of the network representation and the accuracy of the matching algorithm. If the network representation of a concept contains only its name, it is difficult for any algorithm to produce an accurate match between concepts. As explained in Section 4.2.1, the network representation of a concept includes any properties that have the corresponding concept as their domain concept in addition to properties inherited through the `rdfs:subClassOf` and `rdfs:subPropertyOf` predicates. Furthermore, the use of `om:hasSubClass` and `om:hasSuperClass` predicates, which represent references to a concept's sub and super concepts can provide each concept with a set of references to its semantic context. Algorithms such the *deep matching* variation of H-MATCH can the utilise such information in order to produce a more accurate match, albeit at the cost of additional processing and in the case of OntoMobil increased bandwidth utilisation due to larger gossip messages.

### 4.3.2 Template Matching

The template matching method assumes that each node can use predefined mappings (mapping templates) from its own ontology to a number of other ontologies. These mappings [Kalfoglou and Schorlemmer, 2003] can be created in any way (e.g., automated or semi-automated), but they are trusted to be valid. Such mappings do not have to be complete. In fact, as argued in Madhavan et al. [2002], most are partial and there are usually several related mappings that are task-dependent.

By assuming a more constrained environment, the template matching method reflects a more realistic model than the dynamic method. Although nodes still maintain their own ontologies they now acknowledge the existence of other ontologies that can be similar or complementary. By allowing partial mappings, ontologies can also evolve by reusing concepts from other ontologies. For example, a leaf concept can be mapped to a concept at the top of a

concept hierarchy, thereby providing a natural way for ontologies to grow. The opportunistic aspect of interaction is still preserved, as these mappings are purely intentional and utilised only when the target ontologies are found in a node that is currently connected.

The advantage of this approach is that it is no longer necessary to maintain the resource consuming concept matching module. In addition, such mappings can exploit the transitivity of the matching relation, which is described in the next section (4.3.3). Finally, it provides a practical method to evaluate the decentralised concept matching approach in OntoMobil without the need to construct partially compatible ontologies, which would be required for evaluation if the dynamic matching module was used. Chapter 6 uses randomly generated ontologies and randomly generated mappings to evaluate the progressive matching approach.

### 4.3.3 Establishing Semantic Similarity

Semantic similarity is established with the use of a transitive and symmetric relation between two concepts. This occurs only after the concept matching process has compared the two concepts and has identified that they are semantically equivalent. If $c_{ix}$, $c_{jy}$ and $c_{kz}$ represent concepts in $\mathcal{V}_i^0, \mathcal{V}_j^0, \mathcal{V}_k^0$; $\mathbf{M}$ represents the matching relation, and a match exists between $c_{ix}$, $c_{jy}$ and also between $c_{jy}$, $c_{kz}$, the following matches are also established:

1. $c_{ix}\mathbf{M}c_{jy} \Rightarrow c_{jy}\mathbf{M}c_{ix}$ (symmetry)
2. $c_{jy}\mathbf{M}c_{kz} \Rightarrow c_{kz}\mathbf{M}c_{jy}$ (symmetry)
3. $c_{ix}\mathbf{M}c_{jy} \wedge c_{jy}\mathbf{M}c_{kz} \Rightarrow c_{ix}\mathbf{M}c_{kz}$ (transitivity)

The exact semantics of the matching relation can vary and ultimately depend on the strength of the matching mechanism. For this thesis, the scope of matching is reduced to an equivalence relation that is similar to the `owl:equivalentClass` property in OWL. Other relation types are also possible, such as the `owl:kindOf` or `owl:partOf` OWL predicates.

Figure 4.2 illustrates the matching of concepts between two trivial ontologies. The network representation for a concept in each ontology is shown in the corresponding box. It includes the concept's name, its source node identifier and a list of properties and their types. Similarity in the names and properties results in a bidirectional matching relation between the

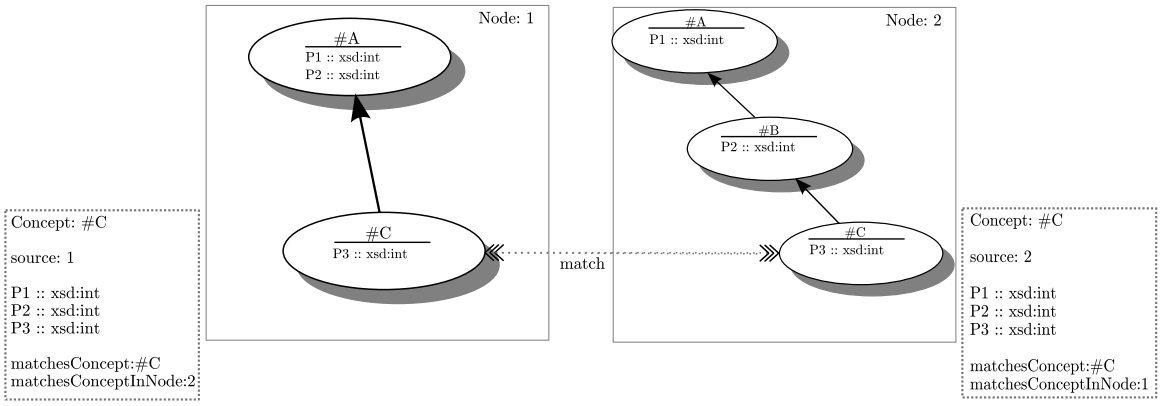


**Fig. 4.2**: Establishing matching associations between concepts.

two concepts. Such a match is established with the addition of two predicates in the network representation of each concept, i.e., `om:matchesConcept` and `matchesConceptInNode`. These predicates take as values the URI and the source identifier of the other concept and can be inserted in any concept whether it resides in the ontology or the concept view.

Establishing this kind of transitive and symmetric relations in a decentralised way can easily lead to inconsistencies. Matching ambiguities can arise especially if a more flexible matching algorithm is used, e.g., one that uses a thesaurus or wordnet [Miller, 1995], instead of strict string matching. For example, consider the distinct concepts $c_{ix}, c_{iy} \in \mathcal{V}_i^0$ and concept $c_{jz} \in \mathcal{V}_j^0$. It is possible for a node to match $c_{ix}$ with $c_{jz}$, while another node matches concept $c_{jz}$ with $c_{iy}$. Eventually, the transitive relation $c_{ix}\mathbf{M}c_{jz}\mathbf{M}c_{iy}$ will be established in nodes $n_i$ and $n_j$. However, the two distinct concepts in node $n_i$ are now matched through concept $c_{jz}$. This problem can appear when matching individual concepts rather than complete ontologies.

Aside from practical solutions to matching ambiguities (e.g., use of a matching relation that is not transitive), such inconsistencies can be resolved and do not effect the correctness of the model. The discovery protocol described in Section 4.5 uses a two-phase approach in which the initial phase identifies nodes that maintain compatible ontologies with the ontology that the query is defined against. This means that from each matching relation, only the `matchesConceptInNode` predicate is initially used. Such circular inconsistencies are therefore

not an issue during the first phase. Although the second discovery phase is not supported in the current implementation, it is assumed that the matching relation will also be used at the provider node during the service matchmaking phase. Its use there is to identify and bridge the semantic gap between the query's source ontology and the candidate ontologies. Circular dependencies can complicate the process if for example a single query concept matches two different concepts from the target ontology. A potential solution would be to conduct multiple matchmaking trials depending on the number of concept similarities that are identified.

### 4.3.4 Matching Usage

The distributed approach to matching requires that each node compares every incoming gossip message with the union of its concept and ontology views. This task imposes a processing penalty that may not be sustainable by all nodes. Clearly, the dynamic matching approach induces a higher overhead than template matching, though all matching approaches eventually rely on concept introspection and pair-wise comparison. To further reduce the impact of concept matching, each node can parameterise its own level of participation in the process of semantic agreement. This is achieved by adjusting the number of concepts each received gossip message is compared against. Every node can now customise its cooperation to a level that is aligned with its processing capabilities. We distinguish between three usage approaches, namely *union*, *ontology-only* and *probabilistic*, which can be employed with either dynamic or template matching. It is worth noting that increasing the per node flexibility and customisation comes at the expense of reducing the predictability of the required number of rounds before network-wide semantic agreement is established.

#### Union

The matching algorithm that is referenced by the gossip reception specification in Listing 5 of Section 3.5.3 assumes that each incoming gossip message is compared against the contents of both the concept and the ontology views. This is the union approach, which guarantees that any concepts that are semantically similar will be identified. This approach promptly advances the semantic knowledge of the node and the network, but also incurs a high processing

cost. For example, in the scenario where the network size and the characteristic parameters of the gossip protocol are $\mathcal{M} = 60, \mathtt{F}_n = 2, \mathtt{T}_a = 2$ and $\mathtt{T}_t = 3$, the concept view has a mean size of $\sim 125$. This means that in each round and in each node, 4 concepts are compared against $\sim 135$ concepts[1]. This has a large performance overhead, not least because the protocol constantly removes concepts in order to guarantee the partial replication property. The resulting churning means that some of the expended effort of the concept matching process is inevitably lost when augmented concepts are removed in the next transmission. In this case, although a concept is removed from the transmitting node there is still the potential that it can "infect" other concepts in the receiving node. Even if the concept's ttl has expired, the protocol guarantees that received concepts will first be matched and only subsequently compared against the list of conditions that ascertain their insertion into the receiver's concept view.

**Ontology-only**

To alleviate the problems of excessive processing overhead due to large concept view sizes and non-utilised matching associations due to concept churning, one approach is to execute the matching algorithm only between received concepts and those in the receiver's ontology view. This approach has two benefits; first, by using the ontology view the number of concepts that are compared at each round is kept small and fixed. In terms of performance, fixed size comparisons are easier to predict and optimise and this is further supported by the fact that the concept fanout is also a fixed parameter. Second, any associations that are established between concepts are unique and persist in the ontology view. This reduces the redundant matches that occur because of replication (e.g., when the concept view is used, many nodes will establish the same matching relation between the same concepts) and limits the problem of non-utilised matching associations. However, it is expected that if all nodes use the ontology-only approach, more rounds will be required to establish network-wide semantic agreement.

[1] $|\mathcal{V}^0 \bigcup \mathcal{V}^c|$

### Probabilistic

The probabilistic approach is an attempt to strike a balance between the reduced latency in semantic agreement when both views are used and the reduced processing overhead when only the ontology view is used. It achieves this by using the ontology view and a random number of concepts that are chosen uniformly from the concept view. The parameter $\texttt{Parameter}_{\texttt{MatchComp}}$ represents the ratio of concepts from the concept view that will be used to compare received concepts against. Clearly, when $\texttt{Parameter}_{\texttt{MatchComp}} = 1$, all concepts are used for comparison and the probabilistic approach is the same as the union one, while when $\texttt{Parameter}_{\texttt{MatchComp}} = 0$, it reduces to the ontology-only approach.

## 4.4 Gossip Suspension

This section presents a prototype algorithm that suspends the operation of the gossip protocol. The algorithm is probabilistic and is intended to be invoked in a situation where a mobile ad hoc network reaches stability, i.e., joining of new nodes or disconnection of existing nodes does not occur. Under this scenario, any potential matching associations will eventually be established and the continuous execution of the gossip protocol will bring no additional benefits. What is required in this case is a mechanism to notify all participants that the operation of the gossip protocol can be suspended. The problem can be trivially solved with the assumption of a group communication protocol that can reach all participants. However, this introduces an extra dependency in OntoMobil with extra complexity and cost. Furthermore, the existence of the partial membership view in OntoMobil means that the group communication protocol must be able to use partial views or a full membership view must also be incorporated. An ideal algorithm to suspend the process of concept exchange would address the following requirements:

1. OntoMobil fitness. The algorithm should fit with the probabilistic model and follow the same decentralised and epidemic approach in order to make integration with OntoMobil easier.

2. Maintainance of concept distribution. In the effort to suspend all participants, it is not

desirable to eliminate the replication properties of the concept view. This means that even after suspension it should still be possible to execute discovery queries and obtain the same guarantees as when the gossip protocol was running.

3. Guaranteed eventual agreement. If a node suspends the gossip protocol, the rest of the participants should eventually suspend execution or the suspended node should eventually resume the gossip operation.

4. Simple state transition between suspended operation and normal execution. It is expected that networks will interleave between periods of stability and periods of high churning. Therefore, gossip resumption should not require extensive reconfiguration, e.g., execution of the join protocol.

The proposed algorithm relies on a simple counter-based index. The index increases in each gossip reception that does not establish a new match and returns to zero in case a reception establishes a matching association between two or more concepts. When the index exceeds a lower threshold, it is interpreted as a first estimation that further gossip receptions will not establish more associations. It is a heuristic that each node employs to decide whether more matching associations will be derived in the future, given that so far a consecutive number of gossip receptions have not produced a match. Although this heuristic enables a node to decide independently whether more matches are likely, the decision to suspend gossip execution needs to be in close succession with the decision of the other participants. If no coordination between the participants occurs and each node suspends gossip independently, the inevitable variation in the timing between the nodes may distort the replication properties.

To enable network-wide eventual suspension, the gossip protocol is used to announce a node's state. The core idea is to use the paradigm of epidemic dissemination to bring nodes into eventual agreement. There are three different states that a node can be in as shown in Fig. 4.3. When the counter-based index crosses the threshold, a node moves from the `Gossiping` state to the `Can Suspend` state. Once in this state, every subsequent gossip transmission will include a flag, which indicates that this node can suspend its gossip. If a node in the `Can Suspend` state receives a gossip message that causes a matching association

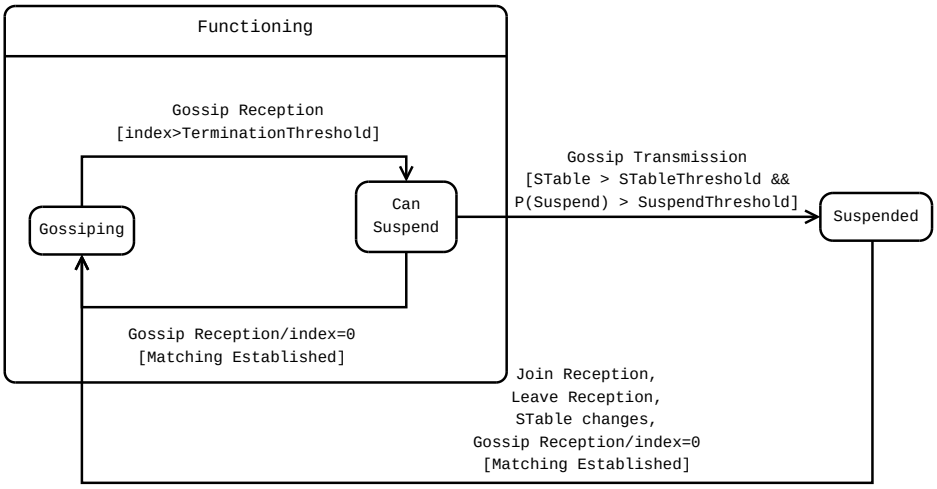


**Fig. 4.3**: State transition diagram for the suspend algorithm.

to be established, it reverts back to the `Gossiping` state and stops including the flag in subsequent gossip transmissions. This intermediate state enables the roll-back to the initial state in case the counter-based scheme has prematurely detected that no matching associations will occur. Nodes receiving a gossip transmission process it as before only now each node also maintains an additional *suspend table* (*STable*) that stores the node identifiers that are in the `Can Suspend` state. To increase the rate of dissemination, nodes in the `Functioning` state will randomly select a number of node ids from the suspend table and further propagate suspend indications in subsequent gossip transmissions. Entries in the suspend table are removed when a gossip message is received, which includes a node identifier that exists in the table but has reverted to the `Gossiping` state.

A node in the `Can Suspend` state moves to the `Suspended` state when the following two conditions are satisfied:

1. The number of nodes in the suspend table exceeds a threshold (*STableThreshold*). This threshold is currently set to the size of the node view, i.e., half the number of all participants. The aim is for a node to suspend gossip execution once it discovers that a number of other nodes have similar intentions. This is the first part of the condition

to suspend gossip execution.

2. A suspend probability $P(Suspend)$ is true. A number is selected uniformly at random between $[0, 1)$ and compared to a threshold ($SuspendThreshold$). A number higher than the threshold yields a true condition. This is a probabilistic method to defer suspension. Due to the eventual delivery property inherent in the gossip protocol, not all nodes will satisfy condition 1 at the same round. Deferring suspension is a trade-off between the probability that some nodes will not suspend and increasing the gossip traffic. The condition prevents situations where a large majority of nodes are suspended, leaving a few nodes gossiping without ever exceeding the required threshold of the suspend table. The probability for such a situation can be reduced considerably if all nodes keep transmitting despite having satisfied condition 1.

Both of these conditions are evaluated when a node is in the `Can Suspend` state and before each gossip transmission. If both conditions yield true the node aborts its transmission and enters the `Suspended` state. To resume gossip execution one of four events can cause a transition back to the `Gossiping` state:

1. the reception of a join request,
2. the reception of a leave request,
3. a gossip message that causes a new matching association to be established or
4. when an entry in the suspend table is removed, i.e., a node has changed its status and reverted from the `Can Suspend` to the `Gossiping` state.

Note that regardless of the state a node is in, it can still receive and process gossip messages.

The proposed algorithm is push-based and therefore fits well with the OntoMobil model. An evaluation presented in Chapter 6 suggests that eventual agreement has a strong dependency on the suspend probability. Where a low threshold reduces overhead but might block certain nodes from suspension, a high threshold increases overhead and may cause nodes to transmit for many rounds before suspension. This algorithm is only an initial investigation and not all issues have been explored at adequate depth.

## 4.5 Random Walk Discovery

In centralised architectures like UDDI or where broadcast facilities are available (e.g., LANs), mechanisms for service discovery can use the existing infrastructure. A query to a well known URL or a broadcast request can return any available services that match certain criteria. Neither of these facilities are available in MANETs. Persistency of identifiers cannot be guaranteed and broadcasting queries in large networks of mobile providers can generate excessive traffic.

The proposed service discovery mechanism in OntoMobil accepts a set of concepts that describe the input and output parameters of a service. It exploits the randomised overlay constructed by the gossip protocol to locate nodes with ontologies compatible to the ontology maintained by the node that issued the service request. The discovery specification in this section covers this mechanism and assumes that a matchmaking process at each destination node will identify services similar to the requested one and transmit the reply to the source node. In case the service query is formulated as a Description Logics expression (e.g., OWL-S), the algorithm in Section 4.7 can be used to decompose the query into a set of concepts.

The discovery process has two phases:

1. the identification of candidate nodes with compatible ontologies,
2. the redirection of the discovery query to the candidate nodes for the final service matchmaking.

During the first phase, a random walk is used to traverse a graph built by considering each node a vertex having outgoing edges to the nodes maintained in its node view. The random walk mechanism aggregates the identifiers of nodes that maintain ontologies with concepts that match the concepts in the discovery query. The node identifiers are addresses of potential service providers since matching relations indicate partially compatible ontologies. When a query condition is satisfied, i.e., at least one or all query concepts are located, or when the query's ttl expires, the second phase is initiated where the query is redirected to the aggregated nodes.

A service request comes in two variations that are differentiated based on whether the query concepts are derived from the ontology of the source node or from a remote ontology. Both variations use concept replication to guarantee that as the number of hops increases so does the probability that a concept is eventually discovered. However, as discovery ultimately depends on the replication factor, the number of hops can be quite large if the size of the concept view is small. This is one of the reasons why both variations bound the random walk with a ttl parameter ($\texttt{Parameter}_{\texttt{RequestTTL}}$). The query ttl avoids the case where the condition of a query cannot be satisfied and a query consumes resources by traversing a long path of nodes. Additionally, since a query can be composed of concepts that do not exist in the network, the ttl is also used as a terminating condition. Since the graph traversed by the random walk contains cycles, each request also piggybacks the visited nodes of the random walk in order to avoid routing loops.

### 4.5.1 Preliminaries

Some common definitions are provided below followed by the description of the two variations.

- A node that initiates a discovery request becomes the *source* node of the request.
- A service request (or service query) is composed from a set of `In` and `Out` concepts. The following notation is used:

  $$\mathcal{Q} = \{(c, T), \dots \mid c \in \mathcal{V}_k^0, k \in \mathcal{M} \wedge T = \{\texttt{In}, \texttt{Out}\}\}$$

  where $c$ represents a *query concept* that comes either from the ontology view of the source node (Section 4.5.2) or from the ontology view of some other participant (Section 4.5.3), while $T$ is the parameter type corresponding to the `om:hasInput` and `om:hasOutput` RDFS properties as presented in example 4.1 and defined in Appendix C.
- The *semantic context* of a concept is defined as a set that includes its super and sub concepts.
- As the semantic context of a concept can be the complete ontology, it is bounded with a parameter $l$.

- If $super_l(c)$ and $sub_l(c)$ give the set of super and sub concepts for concept $c$, then the following set represents the semantic context for all query concepts in $\mathcal{Q}$:

$$\mathcal{H}_l = \{super_l(c) \cup sub_l(c) \,|\, \forall c \in \mathcal{Q}\}$$

- Since the concept view of each node constitutes an independent and identically distributed (iid) simple random sample (assumption $\mathcal{A}2$ in Section 3.6.1), $P_{InCView} = |\mathcal{V}^{\mathtt{C}}|/\mathtt{G}_{total}$ represents the probability that a concept is found in the concept view of any node. Since the same assumption applies for the set of concepts that derives from the union of the ontology and concept views, $P_{InBothViews} = (|\mathcal{V}^{\mathtt{C}}| + |\mathcal{V}^{\mathtt{O}}|)/\mathtt{G}_{total}$ represents the probability that a concept is found in either views of any node.

### 4.5.2 Local Concept Service Request

This section describes the variation of the random walk protocol that handles queries composed from concepts in the ontology of the source node. This represents the most common type of query that appears in the literature [Li and Horrocks, 2003b, Schade et al., 2004, Tangmunarunkit et al., 2003], though existing efforts assume a single common ontology between source and destination. To handle semantic heterogeneity in service discovery, the first step in OntoMobil is to identify nodes that maintain ontologies with concepts that match the concepts in a query. A simple mechanism would identify candidate nodes by examining only the source's ontology view and would immediately redirect the query to the identified nodes. This is straightforward since even concepts in the ontology view embed predicates for matching concepts and their corresponding source node identifiers.

There are two problems with this approach however. Because progressive matching can take time to terminate, concepts in a node's ontology may initially contain only partial matches. This can result in discovery queries with fewer hits. Additionally, semantic services provide a more flexible discovery mechanism because of subsumption-based discovery. This makes provided services described by concepts that are subsumed or subsume concepts in a query still valid. Subsumption-based service matchmaking is the most frequently used inference type in semantic services and has been investigated by Paolucci et al. [2002] and Li

and Horrocks [2003a].

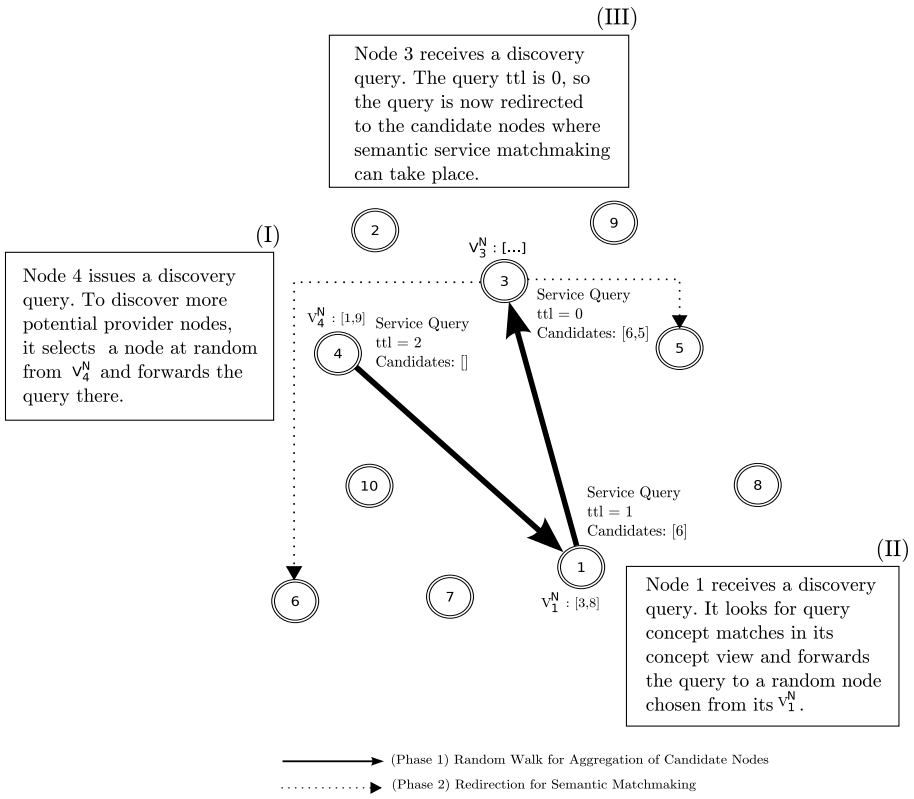


**Fig. 4.4**: The two phase local concept service discovery.

To increase the probability of finding candidate nodes, the query is forwarded to a small subset of nodes using a random walk. Starting from the query source, a node selects a destination at random from its node view and transmits the query to that node. The aim of the random walk is then two-fold; first to locate replicated instances of the query concepts and second to use the matching relations contained in the replicated instances in order to aggregate information about candidate nodes. Depending on the matching progress, the expectation is that replicated instances will contain additional matching relations than the source node.

Each node in the path of the random walk searches for matches not only for the query concepts, but also for the concepts that constitute the semantic context of the query concepts.

The rationale is that if the parent of a query concept has a match, while the actual query concept has no matches, a provided service can still be compatible when defined against the concept matched by the parent. In other words, a candidate node is one that has an ontology with concepts that match *any* concept from the semantic context of *all* query concepts. Figure 4.4 illustrates the process.

**Listing 6** Local Concept Discovery

1: $\#RequestMessage$ represents the service query
2: $\#\{\mathcal{Q}, \mathcal{H}, \mathcal{R}\}$ correspond to $RequestMessage.\{\mathcal{Q}, \mathcal{H}, \mathcal{R}\}$
3: At node $j$
4: $\mathcal{Q} = \{(c_j, T), \dots \mid c \in \mathcal{V}_j^{\mathtt{O}}, T = \{In, Out\}\}$
5: *Populate* $\mathcal{H}$
6: **for all** $c \in \mathcal{Q} \cup \mathcal{H}$ **do**
7: $\quad \mathcal{R} \leftarrow f_{\mathcal{V}_j^{\mathtt{O}}}(c)$
8: **end for**
9: $RequestMessage.ttl \leftarrow \mathtt{Parameter}_{\mathtt{RequestTTL}}$
10: *Choose a random nid from* $\mathcal{V}_j^{\mathtt{N}}$
11: $\boldsymbol{send}(nid, RequestMessage)$
12: *At each node receiving* $RequestMessage$ *during the random walk:*
13: **for all** $c \in \mathcal{Q} \cup \mathcal{H}$ **do**
14: $\quad$ **if** $c \in \mathcal{V}^{\mathtt{C}}$ **then**
15: $\quad\quad \mathcal{R} \leftarrow f_{\mathcal{R}}(c) \cup f_{\mathcal{V}^{\mathtt{C}}}(c)$
16: $\quad$ **end if**
17: **end for**
18: **if** $RequestMessage.ttl = 0$ **then**
19: $\quad \mathcal{D} = \bigcap_{c \in \mathcal{Q}} (f_R(c) \cup \forall x \in sub_l(c).f_R(x) \cup \forall x \in super_l(c).f_R(x))$
20: $\quad$ **for all** $nid \in \mathcal{D}$ **do**
21: $\quad\quad$ *#redirect query to node nid*
22: $\quad\quad \boldsymbol{send}(nid, RequestMessage)$
23: $\quad$ **end for**
24: **else**
25: $\quad RequestMessage.ttl \leftarrow RequestMessage.ttl - 1$
26: $\quad$ *Choose a random nid from* $\mathcal{V}^{\mathtt{N}}$
27: $\quad \boldsymbol{send}(nid, RequestMessage)$
28: **end if**

The exact algorithm for discovery is specified in Listing 6. A service query is formulated by selecting a number of concepts from the source node's ontology view (line 3). Note that $\mathcal{Q}$ and $\mathcal{H}$ do not contain the network representation of the query concepts. Instead, each concept uses a qualified name, which includes the node's address and an optional ontology

namespace, e.g., the URI is of the form `node://<nid>/[<ontology namespace>]/<concept name>`, so that concepts can be uniquely identified and name clashes avoided. Apart from $\mathcal{Q}$ and $\mathcal{H}$, a discovery query requires a third set recording the results of the random walk protocol. Matches are stored in $\mathcal{R} = \{(c, \{id_1, \dots\}), \dots \mid c \in \mathcal{Q} \cup \mathcal{H}\}$, where $id$ is the identifier of a node that has an ontology with a concept that matches $c$.

To simplify the description of the random walk, we define the function $f_S(c)$, where $c$ is a concept with $c \in \mathcal{S}$ and $S$ can be either of the two views, i.e., $\mathcal{V}^{\mathtt{O}}$, $\mathcal{V}^{\mathtt{C}}$ or $\mathcal{R}$. This function returns the set of node identifiers of all matched concepts currently embedded in $c$. In other words, for each concept $c$, it returns all values given by traversing the RDFS property `om:matchesConceptInNode`.

After formulating the discovery query, line 7 shows a first set of matches being recorded at the source node. In a situation where matching has been completed between all ontologies and the termination protocol has paused the gossip protocol, this first set represents all possible matches. In cases where the gossip protocol continues to execute, the query is forwarded to a small number of nodes to increase the probability that all potential matches are identified. These nodes are selected using the random walk.

In particular, each node in the path checks whether the concepts in the query's $\mathcal{Q}$ and $\mathcal{H}$ exist in the node's concept view (line 14) and whether it can augment the query's result set $\mathcal{R}$ with additional matching information (line 15). Thus, this discovery variation has a dual dependency on both the location of replicated concepts but also on the matching progress. This has the implication that in the early stages of concept exchange, concepts that are located will not contain much additional matching information. Therefore, because of the fact that concept discovery alone does not mean that all matching information has been discovered, the condition to redirect the service query becomes the ttl parameter. When the query's ttl reaches zero (line 18), a set $\mathcal{D}$ of candidate nodes is extracted from $\mathcal{R}$ and the query is redirected to these provider nodes. Line 19 lists the condition to identify the set of nodes that have concepts compatible with all the query concepts or their semantic context. Finally, if the query's ttl is not zero, its ttl is decremented (line 25) and it is forwarded to another node selected uniformly at random from the current node's $\mathcal{V}^{\mathtt{N}}$ (line 26).

For clarity, the extra step of recording node ids by transitively following matching relations is omitted. This is an optimisation that is employed in order to discover additional matches. The initial set of identifiers can be potentially augmented if matching relations from the query concepts are also checked against the concepts in the receiving nodes.

We can now derive a probabilistic bound on concept discovery by using the expected value $E[V]$ of the probability measure from equation (3.11) in place of $|\mathcal{V}^{\mathfrak{C}}|$ in $P_{InCView}$ and $P_{InBothViews}$. The probability that $|\mathcal{Q}|$ concepts are found when $n$ nodes are traversed (not including the query source), and each concept $c \in \mathcal{Q}$ has $s = |super_l(c) \cup sub_l(c)|$ super and sub concepts is:

$$P_{Disc} = \prod_{|\mathcal{Q}|} \left(1 - \left((1 - P_{InCView})^n\right)^s\right), \; n \geq 1, \tag{4.1}$$

Equation (4.1) assumes that $\texttt{Parameter}_{\texttt{RequestTTL}}$ will be at least 1. The equation is a measure of the location of replicated concepts through the random walk and does not hint at the number of the matching relations discovered, as that depends on the matching progress. It shows that the semantic context of a concept increases the probability of discovery. This is a direct consequence of subsumption matching, since we treat the discovery of the semantic context of a concept in equal terms to the concept itself.

### 4.5.3 Remote Concept Service Request

Another variation in the service request is the composition of a query from concepts that do not exist in the ontology view of the source node. This serves two purposes; first, it can be used in a situation where an application requires a service defined with an ontology that is known in advance but it is not maintained by the source node. This can be due to storage considerations or because there is only an optional need for such a service. For example, a mobile device can be loaded with predefined requests that are activated in specific locations, where it is known that the predefined requests can be answered. Such well-known locations may even be connected to the Internet and can be universities, hospitals, etc. Second, it allows the demonstration of the concept discovery properties of the randomised overlay, without the dependency on concept matching. This enables easier evaluation of the randomised overlay.

The algorithm in Listing 7 is a simpler version of the local concept discovery in Section 4.5.2. It is assumed that the concepts composing the request are derived from a single ontology and that this ontology is available in a node connected to the network. The primary objective of this random walk is to locate the node that maintains the query concepts in its ontology view. In contrast to the algorithm in Listing 6, which uses the matching associations from *all* query concepts to identify a *set* of candidate nodes, the current algorithm identifies *one* candidate node if it locates *at least one* of the query concept in the ontology or concept views of the nodes in the random walk.

A service request is initially decomposed into a set of concepts, which, along with their parameter types, constitute set $\mathcal{Q}$ (line 4). The source node checks whether any of the concepts in $\mathcal{Q}$ exist in its own concept view (line 6). If a concept is found, the concept's source identifier is acquired and the service request is redirected to that node (line 7). Alternatively, the request is populated with a ttl threshold (line 10) and forwarded to a node selected uniformly at random from the node view of the source node (lines 11–12).

Each node that receives the request examines whether any of the concepts in $\mathcal{Q}$ exist in its ontology view (line 15). If a concept exists, then this node is the ultimate destination and the service request is matched against its provided services. Otherwise, the receiver checks the query concepts against its concept view (line 17) and proceeds by reducing the query's ttl and further forwarding the query (lines 20–22).

It is also possible to compose a request from concepts coming from different ontologies. This would require a request to be redirected to a set of provider node when *all* query concepts have been located. To generalise, when concepts are derived from a single remote ontology the condition that satisfies the request is a logical "or", while concepts that are derived from multiple remote ontologies require a condition that is a logical"and".

Similar to equation (4.1), this variation on the random walk has a probabilistic bound on concept discovery that can be derived as follows:

$$P_{Disc} = 1 - \Big((1 - P_{InCView}) \cdot (1 - P_{InBothViews})^{n-1}\Big)^{|\mathcal{Q}|},\ n \geq 1 \tag{4.2}$$

where $P_{InCView}$ is the probability that a concept exists in the source node, $P_{InBothViews}$ the probability that a concept exists in a node from the random walk path and $|\mathcal{Q}|$ the number

**Listing 7** Remote Concept Discovery

1: $\#RequestMessage$ represents the service query
2: $\#\mathcal{Q}$ correspond to $RequestMessage.\{\mathcal{Q}\}$
3: At node $j$
4: $\mathcal{Q} = \{(c, T), \ldots \mid c \in \mathcal{V}_k^{\mathtt{O}}, k \in \mathcal{M} - \{n_j\} \wedge T = \{In, Out\}\}$
5: **for all** $c \in \mathcal{Q}$ **do**
6: **if** $c \in \mathcal{V}_j^{\mathtt{C}}$ **then**
7: *Redirect query to node c.source*
8: **end if**
9: **end for**
10: $RequestMessage.ttl \leftarrow \mathtt{Parameter}_{\mathtt{RequestTTL}}$
11: *Choose random nid from* $\mathcal{V}_j^{\mathtt{N}}$
12: $\boldsymbol{send}(nid, RequestMessage)$
13: *At each node receiving RequestMessage during the random walk:*
14: **for all** $c \in \mathcal{Q}$ **do**
15: **if** $c \in \mathcal{V}^{\mathtt{O}}$ **then**
16: *Node Found*
17: **else if** $c \in \mathcal{V}^{\mathtt{C}}$ **then**
18: *Redirect query to node c.source*
19: **else**
20: $RequestMessage.ttl \leftarrow RequestMessage.ttl - 1$
21: *Choose random nid from* $\mathcal{V}^{\mathtt{N}}$
22: $\boldsymbol{send}(nid, RequestMessage)$
23: **end if**
24: **end for**

of concepts in the query.

### 4.5.4 Service Matchmaking

The second phase of discovery takes place in each of the candidate nodes identified during the first phase. The evaluation of service discovery queries at provider nodes can provide good load balancing and scalable properties. One advantage is that broker nodes are not needed. Additionally, processing in a provider node can be proportional to the number of offered services and the size of its ontology. It is expected that nodes with larger ontologies and more services will be able to handle an increased number of queries. Finally, MANETs have a high degree of transient connections so a reply from a provider node back to the source node is a guarantee that the provider node is still connected. In contrast, a reply from a

broker node that some provider contains a service does not guarantee that the provider is still connected or has not failed, unless an additional liveness protocol is used.

In each provider node, a subsumption-based matchmaking process can be used to identify compatible services. Before matchmaking takes place, an intermediate step is required to associate the concepts composing the query to the matching concepts of the destination ontology. This is achieved by either converting the query to use concepts from the target ontology or transform the target ontology with concept equivalence relationships. Each provider node has the flexibility to choose its own conversion based on parameters such as resource consumption or response accuracy. Similar issues are examined in the area of concept unification and matching in Description Logics [Baader et al., 1998].

## 4.6 Evaluation of Concept Discovery

This section presents the analytical results obtained from the two discovery methods, local and remote, against the two simulation types of perfect uniform views with no message failures ("perfect views") and the ns2 simulator[2]. Since the focus is on measuring the probability of locating the query concepts using the random walk discovery protocol, the influence of the progressive matching approach on discovery is not taken into account. This is deferred to Chapter 6, where an evaluation of the different matching methods is conducted. This section evaluates

- the accuracy between the analytical and simulated discovery ratio,
- the impact of the concept view size,
- the impact of network size.

Because of the large set of gossip parameters, most parameters are kept constant ($\mathtt{F}_n = 2, \mathtt{F}_c = 4, \mathtt{T}_a = 2$) and only the gossip ttl is varied ($\mathtt{T}_t = 2, 3$) as a way to influence the size of the concept view. The simulation experiments tested network sizes that ranged from 20 to 60 nodes. Each simulation ran for approximately 350 rounds and was executed five times. In

[2]The detailed setup of the ns2 simulation environment is given in Chapter 6.

| Discovery Variation | $\lvert\mathcal{V}^0\rvert$ | $\mathtt{F}_n$ | $\mathtt{F}_c$ | $\mathtt{T}_a$ | $\mathtt{T}_t$ | $\lvert\mathcal{M}\rvert$ | $\lvert\mathcal{Q}\rvert$ | $\lvert\mathcal{H}\rvert$ | $\mathtt{Parameter}_{\mathtt{RequestTTL}}$ |
|---|---|---|---|---|---|---|---|---|---|
| Local | 10 | 2 | 4 | 2 | 2, 3 | 20, 40, 60 | 2 | 2 | 5 |
| Remote | 10 | 2 | 4 | 2 | 2, 3 | 20, 40, 60 | 1 | - | 5 |

**Table 4.2**: Summary of gossip and random walk parameters in the concept discovery evaluation.

both simulation types, discovery began after the first 30 rounds so that initial buffer affects were minimised and the concept view had reached a steady state. After that, in the ns2 simulation every node transmitted a discovery query each second, while in the perfect views simulator every node transmitted a query at the end of each round. In all experiments the query ttl was set to five. For the analytical results, the values for $E[V]$ are in Appendix A. Table 4.2 summarises the parameters used. In Fig. 4.5 and 4.6, the y-axis depicts the discovery ratio per node in the random walk. This represents the cumulative number of queries satisfied in each successive node and divided by the total number of queries, including failed queries. It represents the probability of locating the query concepts at the various nodes that compose the random walk.

Figure 4.5 presents the discovery ratio for the local concept service request (Section 4.5.2). For each query, the source node selects uniformly at random two concepts from its ontology view ($|\mathcal{Q}| = 2$). Each query concept has also one corresponding context concept ($|\mathcal{H}| = 2$), bringing the total number of concepts to four. In all cases, apart from when the network size is 60, the discovery probability calculated from equation (4.1) is aligned with the experimental results. In the case of 60 nodes, there is a big disparity between the ns2 discovery ratio and the other two. This is explained by the high failure rate caused by the succession of unicasts required by the random walk protocol. In a MANET of 60 nodes, the failed discovery queries, i.e., queries that were not satisfied and did not expire because of the ttl parameter, were in the order of $\sim 88\%$. This is observed by the fact that the discovery ratio remains constant as the number of nodes in the random walk increases. The high ratio of failed queries reflects the sequential reliability nature of the random walk. In order for a query to succeed, *all*

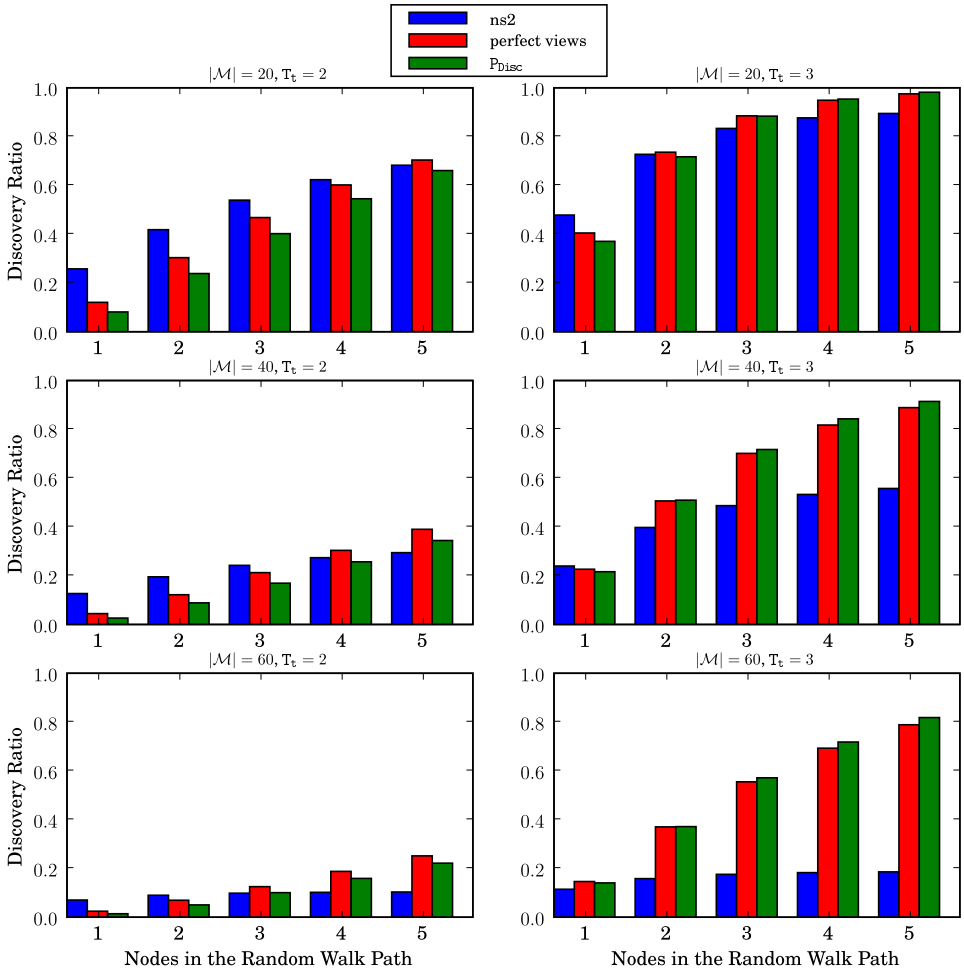


**Fig. 4.5**: Comparative discovery ratio for the local concept service request method.

independent transmissions must be successful. This compounds the already high failure ratio of unreliable unicasts in network sizes of 60 nodes. Potential optimisations to remedy this problem could include the utilisation of a more reliable unicast protocol (e.g., TCP) as well as the optimisations discussed in Section 3.5.6. When a reliable unicast protocol is not possible, these results suggest that a high discovery ratio can still be obtained by using a small query ttl value combined with a high value for the average size of the concept view.

Figure 4.6 shows the discovery ratio for the remote concept service request (Section 4.5.3). In each query, the source node selects one concept ($|\mathcal{Q}| = 1$) uniformly at random from a

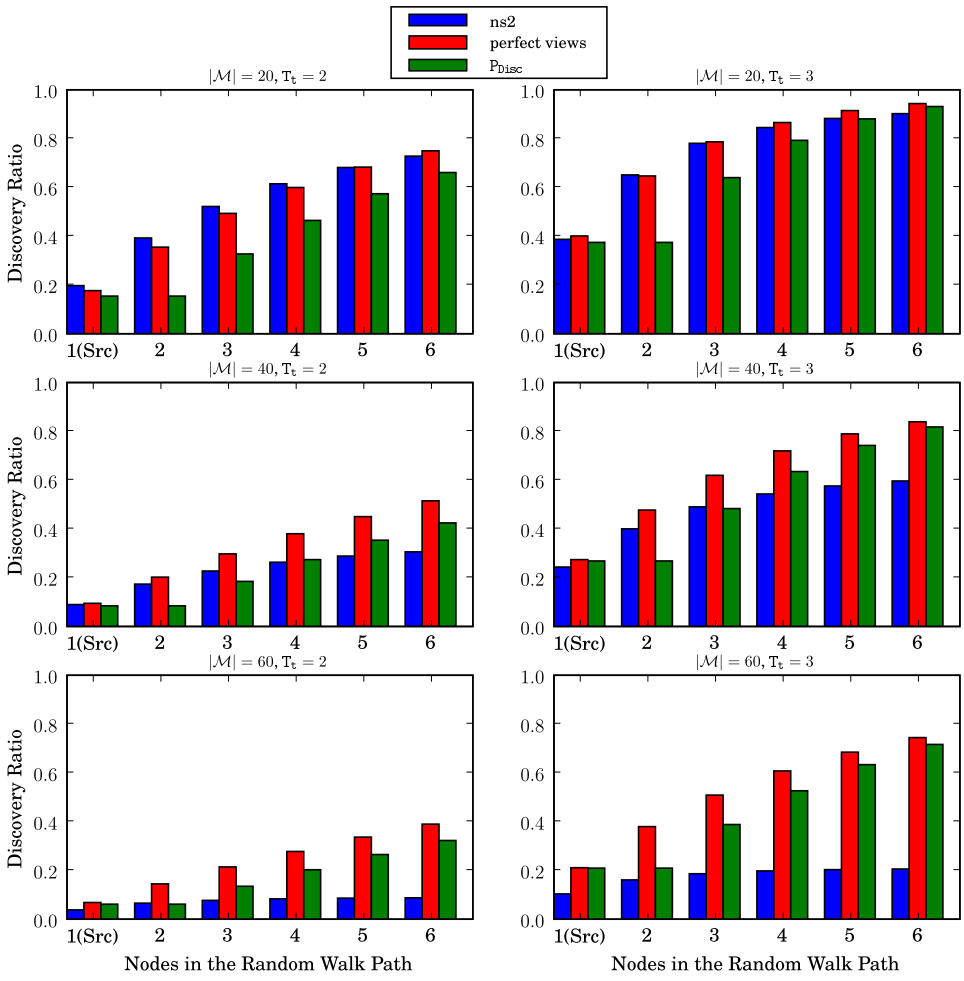


**Fig. 4.6**: Comparative discovery ratio for the remote concept service request method.

set that includes all available concepts, except the ones in the ontology view of the source node. The source node subsequently searches its concept view for that concept and if it is not found, the random walk protocol is executed. This means that the first node in Fig. 4.6 represents the lookup at the query source. Concept discovery for this second variation follows similar trends as the first one. Analytical results match the experimental ones apart from the case of 60 nodes.

The discovery ratio increases with the size of the concept view in both figures. With respect to network size, the discovery ratio shows a linear decrease with increasing number

of nodes and hence increasing number of concepts. This is a good results, which verifies the scalability of the discovery protocol. Note that the current model assumes an intrinsic relationship between nodes and content (concepts), which is contrary to the usual treatment of nodes and content (keys) as separate entities in P2P distributed hash table overlays. This intrinsic relationship is in accordance with the model assumption of autonomous provider nodes.

The alignment between the analytical and the experimental results also verifies the assumption that the union of the concept and ontology views constitute a simple random sample from the population of all concepts (assumption $\mathcal{A}2$ in Section 3.6.1). In other words, the gossip protocol and the construction of the concept view do not produce a correlation between the concepts in the different concept views and hence the discovery probability model that is captured by equations (4.1) and (4.2) is further validated.

## 4.7 Query Decomposition in Description Logics

The main OntoMobil abstraction is concepts. This is reflected in both the gossip and discovery protocols and in the choice of RDFS, which is used as a simple ontology and because of the convenient mapping between concepts and RDFS classes. However, semantic descriptions and more specifically ontologies, represent knowledge using additional syntactic elements. For example, OWL-S permits the composition of service description and advertisements not only from concepts but also from Description Logics (DL) expressions. Since RDFS is not based on Description Logics, it lacks the ability to compose concepts with DL constructors. This limits the applicability of the current approach to service requests that are not very expressive.

In order for OntoMobil to use complex service queries and service description standards such as OWL-S, a simple and efficient set of rules is required to decompose DL expressions into concepts. This section provides such a set of rules that are syntactic in nature and thus do not have the overhead stemming from the use of a reasoner. The trade-off between a syntactic and a semantic transformation is that the syntactic one offers only an approximate decomposition. In the case of OntoMobil this is sufficient. Concepts constitute only the first

| | | | |
|---|---|---|---|
| $C$,$D$ | $\longrightarrow$ | $A$ \| | (atomic concept) |
| | | $\top$ \| | (universal concept) |
| | | $\bot$ \| | (bottom concept) |
| | | $\neg A$ \| | (atomic negation) |
| | | $C \sqcap D$ \| | (intersection) |
| | | $\forall R.C$ \| | (value restriction) |
| | | $\exists R.\top$ | (limited existential quantification) |

**Table 4.3**: Description Logics $\mathcal{AL}$ language constructors [Baader et al., 2003]

step in the discovery process and are only used to identify provider nodes. The approximate decomposition is also balanced by the mandatory query evaluation at the provider node.

The basic elements of Description Logics are *atomic concepts* and *atomic roles*. Abstract DL notation traditionally uses the letters $A$ and $B$ to represent atomic concepts and the letter $R$ for atomic roles. Complex DL expressions can be inductively built from atomic elements using *concept constructors*. Complex concept descriptions are usually denoted with the letters $C$ and $D$. Different DL languages are distinguished by the constructors they provide. A minimal DL language that is of practical interest is the $\mathcal{AL}$ language (Attributive Language), which has the constructors in Table 4.3.

The basic transformation principle is to recursively evaluate a DL expression in a left-to-right manner and extract the concepts that form the expression. The extracted concepts do not have to be atomic as that would require inference. In the simplest case, an expression that contains only atomic concepts and the intersection constructor, i.e., $C \sqcap D$ will yield a result set containing both $C$ and $D$. The same rule applies for the atomic negation constructor. When an expressions contains roles, the rule is to extract concepts from both the first argument (the role source) and the second (the role filler).

Certain role expressions are easily decomposed if the source precedes the role. For example, in $C \sqcap \forall R.D$, the transformation rule will again yield concepts $C$ and $D$. In the case where the universal and bottom concepts are used as role fillers, the rule is to extract only the source concept. For example, the following two expressions, $C \sqcap \exists R.\top$ and $D \sqcap \forall R.\bot$, will

yield concepts $C$ and $D$ correspondingly. Similar rules apply when multiple nested expressions are used. In an expression like $((C \sqcap D) \sqcap \exists R.C) \sqcap \exists R.(E \sqcap F)$, when a role immediately follows another role, the source concept of the immediately preceding role is recursively used. The previous expression yields the result set of $C, D, E$ and $F$. In expressions where the role source is implicitly assumed from the ontology, e.g., $\exists R.\top$, the transformation must find the base object where the role is defined and use that as the role source.

Description Logics has produced more expressive languages than $\mathcal{AL}$ by adding more constructors. For example, other members of the $\mathcal{AL}$ family include the *union* constructor and *number restrictions* constructors. The same set of rules apply to these constructors and can be used to extract the corresponding concepts.

## 4.8 Summary

This chapter has described the processes of concept matching and service discovery. It presented the OntoMobil concept network representation followed by a set of requirements for concept matching in Section 4.3. In accordance to these requirements, the specification of two different matching methods was given demonstrating the feasibility of the proposed decentralised matching model. From the two matching methods, template matching will be used in Chapter 6 to evaluate the latency between the different usage matching approaches.

Due to the lack of an RDFS-based service representation, a custom one was developed specifically for OntoMobil. It is derived from the atomic process representation in OWL-S and describes a service as a set of input and output parameters. In Section 4.5, the service description was used together with a random walk protocol to specify a discovery process that takes advantage of the overlay created by the gossip protocol. Two variations were presented, local and remote, which construct services based on different assumptions and give different probabilistic guarantees. An evaluation confirmed the correspondence between the simulation and analytical results of the random walk up to 40 participants. However, it also very clearly illustrated a problem with the random walk approach in large network sizes ($|\mathcal{M}| > 40$). This does not invalidate the discovery approach. In the case where the random walk is used to aggregate additional concepts matching the query ones (1st variation), the

random walk is an optional component during the initial stages of concept matching. As rounds progress, matches will be disseminated across nodes, until each node contains the complete set of matches for all concepts that compose its ontology view. Where the random walk cannot be made optional (2nd variation), a more reliable unicast is necessary for larger network sizes.

Finally, we demonstrated a simple syntactic set of rules that can be used to decompose complex Description Logics queries into a set of concepts for inclusion in the OntoMobil discovery request. Future work in service discovery for OntoMobil may include the integration of additional reliability into the transmission of the service requests and the introduction of further optimisations in the identification of services rather than provider nodes. As mentioned in Section 4.2.1, service usage indicators can be incorporated as an extra property in the concept network representation.

# Chapter 5

# Implementation

This chapter describes an implementation of the OntoMobil model as specified in Chapters 3 and 4. The implementation was written in a combination of C++/Python/TCL code using the ns2 network simulator as the network transport, Python for some parts of the protocol logic and the Python RDFlib[1] library to manage the concept and ontology views. A basic programming model is used to group the different protocols together and simplify the writing of the scenarios that were necessary for the simulation experiments. Section 5.1 discusses the programming model, Section 5.2 the OntoMobil architecture and Section 5.3 describes the message formats of the gossip and discovery protocols. The chapter concludes with a summary in Section 5.4.

## 5.1 Programming Model

The OntoMobil model offers an intuitive implementation because it uses concepts as a unified abstraction and is symmetric; both service ontologies and service queries are first decomposed into concepts and then randomly disseminated across nodes. The current implementation supports a rudimentary programming model on top of the "bare" OntoMobil protocols. A lot of configuration options still require manual setting and it is expected that an additional

[1] http://rdflib.net

middleware component will be needed to simplify the development of applications[2]. A session-based approach was chosen as the basis of the programming model. Currently, a session is started by each node to initialise the OntoMobil and RDFlib related structures. Subsequent actions occur through the session object. Since gossip is a background activity, it is initiated automatically after a join request returns the number of node identifiers that exceed the predefined node view threshold ($\texttt{Parameter}_{\texttt{NodeView}}$).

The following API calls are provided through the Python bindings:

- `OMSession()`, returns a session object used to invoke subsequent methods. It also creates and initialises a number of other Python objects that are used by each node in the execution of the various processes.
- `OMSession.registerOntology()`, loads an ontology from a given file or a runtime representation and populates the ontology view.
- `OMSession.queryTransmit()`, accepts a set of input and output concepts representing the requested service. It then selects a random identifier from the node view and forwards the query to that address.
- `OMSession.join()`, takes as input the number of nodes that must be discovered by the bootstrap protocol, before the gossip protocol is initiated.
- `OMSession.leave()`, announces the node's intent to disconnect.

A number of callback functions are also supported in order to notify the application of certain events. At the moment these callback events do not allow the modification of the execution flow. They are merely used for debug purposes and for evaluation scenarios.

- `nodeViewPopulatedCB()`, the callback is raised when the join process populates the node view with the number of node identifiers given to the join method as an argument.
- `gossipMessageReceivedCB()` is called when a gossip message is received. A read only copy of the received message is also passed to the caller function.

[2]Such a middleware layer has been built as part of a Master's thesis and is available as a technical report from the Department of Computer Science, Trinity College Dublin.

- `queryReceivedCB()`, is called when a service request is received. A read only copy of the received message is also passed to the caller function.
- `querySatisfiedCB()`, is called when the query's condition is satisfied.

### 5.1.1 Concept Naming

In a distributed environment, the identification of resources that are autonomously developed must rely on a set of syntactical naming conventions. In OntoMobil, this issue affects the naming of concepts and in particular the naming of concept references. While the gossip protocol uses the network representation to disseminate concepts, during the discovery process only a concept reference is used. This means that there must be a way to compare a concept reference against its network representation.

A naming convention for concepts in OntoMobil must satisfy two requirements. First, each concept has to be uniquely identified. Currently, all concepts are considered unique and must be recognised as such even when concepts belong to the same ontology but maintained in different nodes. Conversely, the second requirement is posed by the concept matching process and the need to identify concepts from the same ontology (Section 4.3.1) as well as identify concepts that have identical names but belong in different ontologies.

The two requirements are currently addressed with the inclusion of two predicates in the concept's network representation, namely the `om:source` and the `om:namespace` predicates. The two predicates are based on the assumptions that nodes have unique identifiers and nodes do not change the namespaces of well-known ontologies. Together with the concept's name, the concept network representation allows the

1. unique identification of each concept,
2. identification of identical concepts that belong to the same ontology.

For the efficient lookup of concepts during the discovery process, OntoMobil has devised a new URI structure that uses the source and namespace predicates in a concise format. Each concept is prefixed with the source node's address, while if a concept is part of a well-known ontology, its namespace is optionally included. This gives a concept URI like the following:

```
node://<nid>/[<ontology namespace>]/<concept name>
```

To maintain consistency with the URI standard, the protocol prefix (e.g., http://) is stripped from the ontology namespace URI.

The concept URI is solely used for the first variation of the random walk (Section 4.5.2), where concepts are selected from the source node's ontology view. For the second variation (Section 4.5.3), since the address of the provider node is not known, the concept URI removes the `node://<nid>` prefix and uses the `<ontology namespace>/<concept name>` convention. This preserves the location transparency offered by the semantic overlay so that requests are still specified semantically and not through the location of the provider nodes.

## 5.2 OntoMobil Architecture

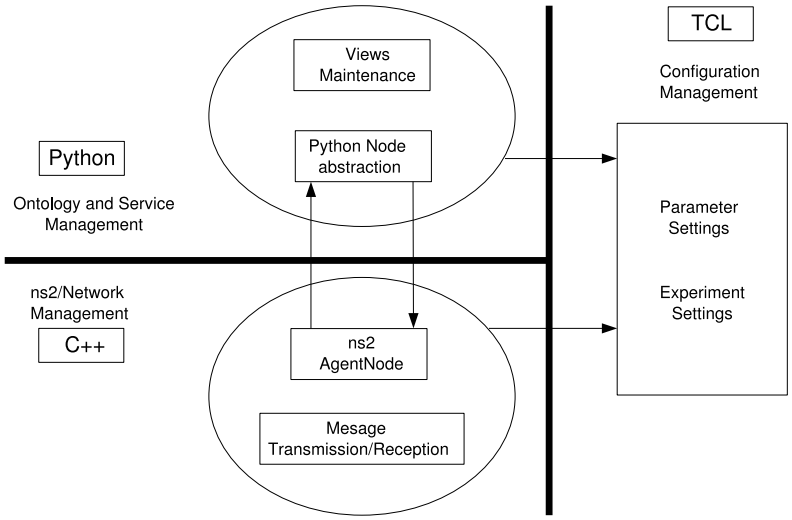


**Fig. 5.1**: Implementation architecture.

The OntoMobil architecture is split in two layers. The low level layer contains the protocol implementation, which is written in C++ using the ns2 API, while the upper layer contains the ontology and service management written in Python. This decomposition between the network and the ontology aspects of the model should ease the porting from the ns2 network

simulator to a real protocol stack. Figure 5.1 illustrates the system architecture.

The ns2 simulator offers the abstraction of an *agent* for network protocol development. The ns2 agent that was developed for OntoMobil also has a corresponding object in the Python environment. This permits the placement of functionality where appropriate, e.g., network functions in ns2 and concept manipulation in Python. TCL code, an integral part of the ns2 environment, is used to store configuration parameters that can be accessed from both the Python and the C++ environment.

The node abstraction in Python stores concepts in two separate RDFS repositories managed by the RDFlib library. These two repositories correspond to the ontology and concept views. When a ttl property must be inserted or when a match is established, an RDF statement is inserted in the corresponding RDFS class that represents the concept. Recall that an RDF statement consists of a subject, a predicate and an object (e.g., "ttl hasValue 3"). The flexibility of RDF makes it possible to augment each concept with meta-information.

The node view, suspend list and other auxiliary buffers are all implemented using Python set structures. Figure 5.6 on page 152 shows the diagram of the most important classes composing the current implementation. The principal class is `OntoMobilAgent`, which is a C++ class derived from the `Agent` class in ns2. Each instance represents a mobile node and is responsible for parts of the protocol logic but mainly it uses the ns2 framework to transmit and receive network packets. Each `OntoMobilAgent` instance maintains a single reference to a `PythonNode` instance, which is the intermediate layer between the ns2 and Python layers. It is a utility class used for conversion between the type systems of C++ and Python. Each node instantiation in ns2 results in the instantiation of the Python `Node` class. This contains the majority of the protocol logic, including concept maintenance and service request processing. At the moment there is no back-reference from the `Node` instance to the corresponding ns2 agent instance. The Python classes are controlled directly from the ns2 environment, which is a design choice imposed by the tight coupling between the evaluation scenarios and ns2. Such scenarios are generally written in TCL, which in ns2 is integrated with C++, so that TCL statements can control aspects of C++ agent classes. Therefore, in the current implementation, it is the ns2 code that controls the `OMSession` class, using it to

acquire information stored in the Python environment, rather than an application.

The `Node` Python class incorporates most of the logic for the gossip and discovery protocols. It contains a reference to the `ConceptView` and the `Ontology` classes. The `ConceptView` stores the ontology and concept views and some utility functions to access and update concepts. The `Ontology` is a factory class that creates a per node runtime representation of an ontology. Each newly created node accesses this factory class in order to obtain an ontology and populate its ontology view. The two evaluation ontologies that are used for the thesis experiments derive from this class. The `DiscoveryOntology` incorporates logic to create random mappings between concepts in different node ontologies. It is used in the evaluation of discovery versus matching progress scenarios (Section 4.5.2). The `AnalysisOntology` creates unique concepts prefixed with the node's identifier and is used in the evaluation of the concept view size variability.

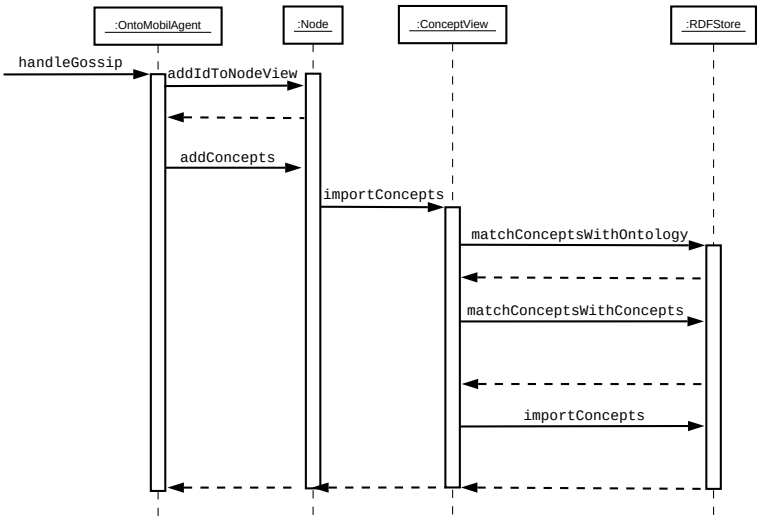


**Fig. 5.2**: Sequence diagram representing the reception of a gossip message.

Figure 5.2 shows a sequence diagram representing the reception of a gossip message. Once a message is received by a node, control is passed to the `handleGossip` method of the `OntoMobilAgent` class. Node view management occurs first and the subsequent call to the `addConcepts` method matches received concepts to the concepts in the two views. Finally,

the `importConcepts` call to the `RDFStore` instance imports the appropriate concepts into the concept view.

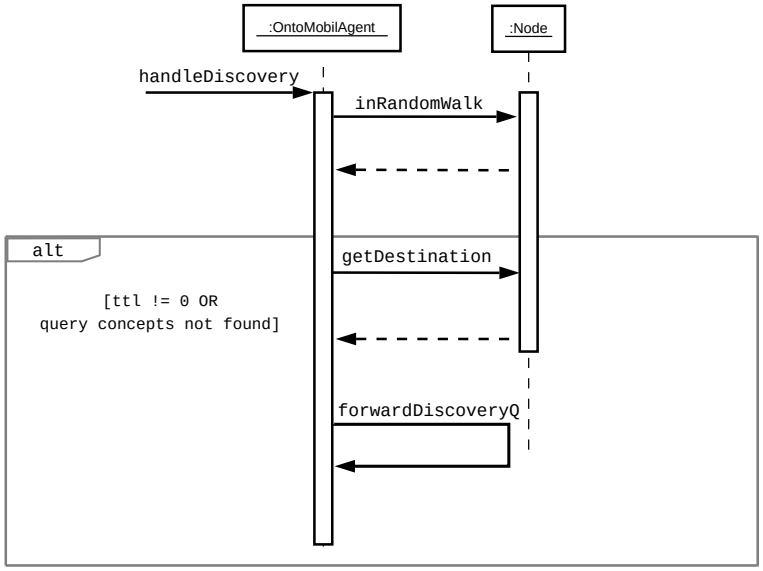


**Fig. 5.3**: Sequence diagram representing the reception of a discovery query.

Figure 5.3 shows the sequence diagram corresponding to the reception of a request during the random walk discovery route. Control is dispatched to the `handleDiscovery` method, which locates the request concepts and potential matches by passing the query to the `inRandomWalk` method of the `Node` class. After the `inRandomWalk` method completes, the query's condition and ttl are examined in order to establish whether to further forward the query or identify the set of provider nodes and redirect the query to those destinations for the final matchmaking phase.

## 5.3 Protocol Description

In the current implementation, protocol messages are verbose and a more compact representation is certainly possible to reduce processing time and network traffic. The main message payload is the transmitted concepts, which are marshaled in XML and handled by the Python environment. Other protocol fields, such as the list of node identifiers that reinforce the re-

Gossip Message

| reinforcement node ids | disconnected node ids | suspended node ids | concept payload |
|---|---|---|---|
| [<nid>,...] | [<nid>,...] | [[<nid>,<status>],...] | <xml> |

**Fig. 5.4**: Gossip message format.

Request Message

| visited node ids | ttl | concepts | query condition | original query |
|---|---|---|---|---|
| [<nid>,...] | <int> | [[<URI>,[<nid>,...]],...] | <string> | <xml> |

**Fig. 5.5**: Request message format.

ceiver's node view and the ttl field of the discovery requests are encoded as C++ structures so that they can be manipulated directly by the ns2 runtime environment. However, since most of the protocol logic already occurs in the Python code, a potential optimisation could refactor the message handling and move any remaining processing tasks from the C++ layer to Python. This would have the effect of unifying the message structure.

OntoMobil defines two different protocol messages corresponding to the gossip and discovery protocols. Figures 5.4 and 5.5 display the protocol fields and their associated types for the two messages. All fields are mandatory, but since not all fields are used in every transmission, an additional protocol optimisation may consider making certain fields optional. Each field that requires a sequence of items (e.g., `[<nid>,...]`) has an explicit prefix that denotes the length of the sequence. Such sequences are used in the gossip message to represent the reinforcement and disconnected node identifiers, while a sequence of sequences is also used to gossip the suspend state of node identifiers. The last field in the gossip message is the concept payload containing the XML-formatted network representation of a parameterised number of concepts ($\mathtt{F}_c$).

The request message in Fig. 5.5 stores the sequence of visited nodes so that loops are avoided during the random walk. The sequence is stored in the "visited node ids" field, which

always contains the source identifier of the request as the first item in the list. Subsequent fields contain:

- the query ttl ($\texttt{Parameter}_{\texttt{RequestTTL}}$),
- the sequence of concepts that are derived from the request. Each item in this sequence is itself a sequence that maps a query concept to a sequence of node identifiers that contain similar concepts ($\mathcal{Q}, \mathcal{H}$),
- the query condition that must be satisfied before a query is redirected to the provider nodes,
- the initial service request in XML representation. Since the service matchmaking is performed at the provider nodes during the second phase of the random walk, the service request needs to be present.

The query condition can be an empty string. In this case the ttl field is used as the sole condition to identify the node in the random walk, which will decide to redirect the query to the provider nodes. If the condition field is not empty, a condition is composed by using the two logical operators "and" and "or" with concept URIs as operands. Each node in the random walk can then check the query condition and depending on whether it is satisfied, will redirect the query to the provider nodes obtained from the "concepts" field or simply forward the query to another node.

## 5.4 Summary

This chapter has presented the implementation approach of the OntoMobil model. We have used the ns2 simulation environment to abstract the details of mobile ad hoc networks, such as routing and MAC layer protocols and traffic and mobility models. Parts of the OntoMobil model are implemented within the simulation environment, while the parts that deal with concept management and the logic of concept matching have been implemented in Python using the RDFlib library. This decomposition has yielded an architecture that is easily adaptable to future protocol extensions and experimentation. It also permits the utilisation

of a network layer other than the one provided by ns2, in case a real-world implementation of OntoMobil is undertaken.

The implementation is described in terms of a basic session-based programming model together with the details of the ns2 and Python classes and the format of the messages used by the gossip and discovery protocols. There are a lot of potential optimisations and refactoring that can be carried out in the current implementation. The main concern has been the creation of an initial prototype to evaluate the properties of the protocols, rather than a middleware to build service-oriented applications. It is expected that such a middleware would benefit from the existence of the programming model described in Section 5.1 and the concept abstraction provided by OntoMobil.

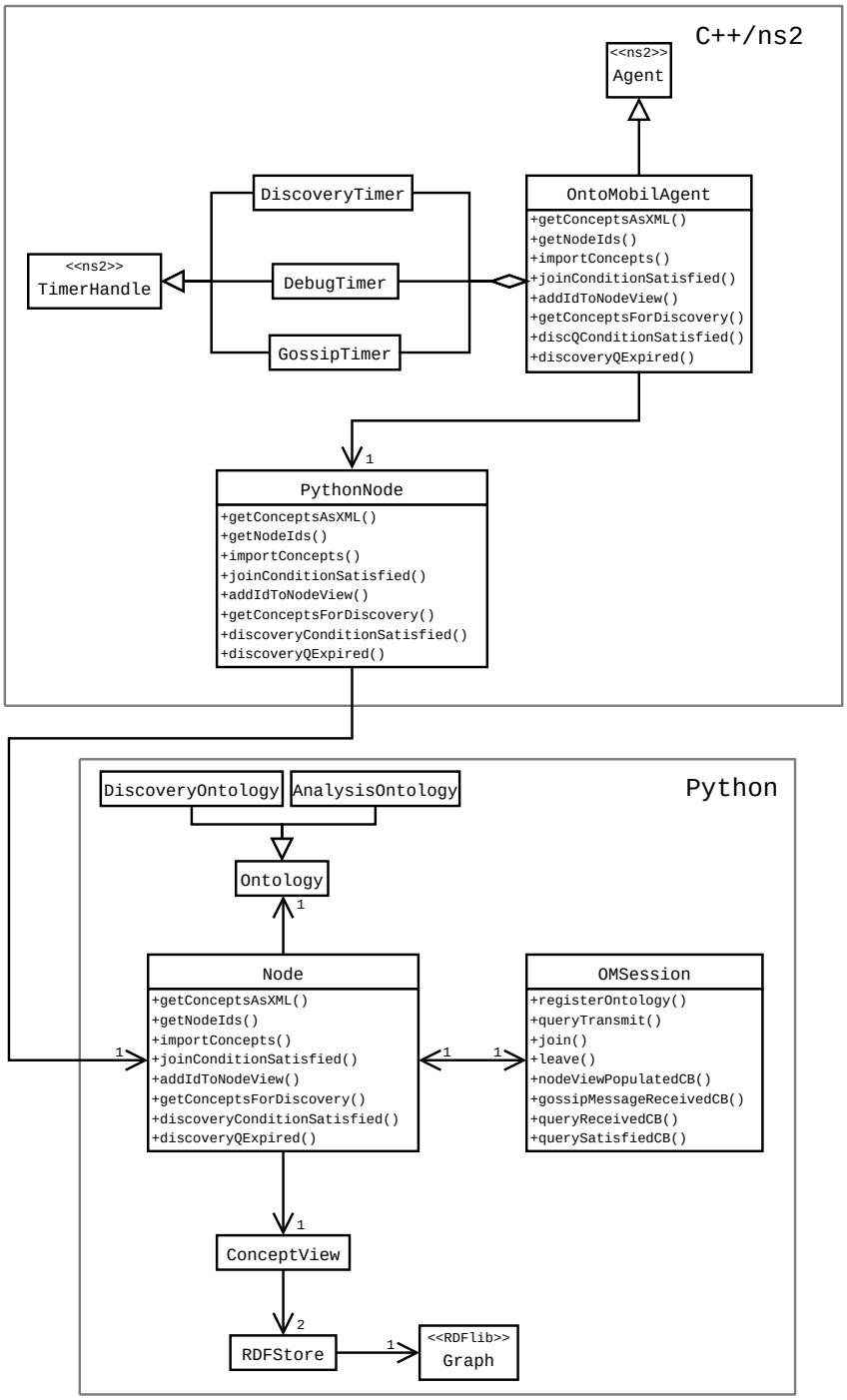


**Fig. 5.6**: Class diagram of the OntoMobil implementation in ns2 and Python.

# Chapter 6

# Evaluation

This chapter concludes the evaluation of OntoMobil. Chapters 3 and 4 included evaluation results that focused on the correspondence between analysis and simulation for the variability of the concept view size and the probabilistic bounds on discovery. In this chapter, the evaluation presents aspects of the model that are outside the scope of the analysis. Instead, simulations are used to evaluate the different scenarios and carry out the necessary experiments.

The objective of the evaluation is to show performance results for the auxiliary protocols, i.e., join, leave and suspend and the three concept matching approaches, i.e., union, ontology-only and probabilistic. The evaluation also presents discovery results where the random walk protocol depends both on concept discovery and concept matching. Overall, the performance evaluation is primarily focused on latency. Because of the eventual delivery characteristics that are associated with the gossip approach, it is important to quantify the delay that is involved for complete semantic matching and discovery. This can then be contrasted against the overhead induced by the different matching approaches.

## 6.1 Experimental Setup

This section describes the ns2 configuration used by all experiments throughout the thesis. It includes the configuration for the experiments performed in Chapters 3 and 4 and also in this

chapter. Since a network simulator like ns2 requires the configuration of an overwhelming number of parameters, the presentation is modularised and is based on recent results on the conduct of simulation studies in MANET protocols. Specifically, Kurkowski et al. [2005] have studied a number of published papers with simulations of mobile ad hoc networks and have devised a set of guidelines for the comprehensive presentation of simulation results. These guidelines are used in this section.

- *RF propagation* – The physical layer used the 2-ray ground reflection model with an omni-directional antenna and transmission power of approximately $\sim 25dBm$. The wireless range of each node was kept constant at $250m$.

- *Mobility model* – The random trip [PalChaudhuri et al., 2005] mobility model was chosen for all scenarios. This is a generalisation of the random waypoint model for which the distribution of node mobility state converges to a time-stationary distribution, starting from any initial state. It avoids well-known problems [Yoon et al., 2003] with the random waypoint model such as average speed decay and highly skewed location probability distributions. The respective parameters of velocity and pause time are independent for each node and were selected from the uniform distribution of $U[1,3]$ m/sec for velocity and $U[1]$ seconds for pause.

- *Traffic model* – There was no other traffic generated other than that of the routing, gossip and discovery protocols. One of the parameters of the gossip protocol that was kept constant in all experiments was the timeout parameter, which was selected independently for each node from the uniform distribution of $U[0.69, 1.11]$ seconds. Each node started to gossip independently of other nodes only after its node view reached the threshold of $\texttt{Parameter}_{\texttt{NodeView}} = |\mathcal{M}|/2$.

- *Protocol stack* – The physical and medium access control layers used the IEEE 802.11 DSSS PHY and IEEE 802.11b DCF protocols. The Address Resolution Protocol (ARP) was disabled so that no extra traffic was generated. The OLSR routing protocol was used for all simulations with the default parameters described in Clausen et al. [2001].

| Nodes ($\|\mathcal{N}\|$) | Participants ($\|\mathcal{M}\|$) | Area |
|---|---|---|
| 30 | 20 | $915 \times 915$ |
| 50 | 40 | $1180 \times 1180$ |
| 70 | 60 | $1400 \times 1400$ |

**Table 6.1**: Mobile nodes and simulation area in ns2.

- *Area size* – Different experiments tested different protocol parameters against three network sizes. In experiments where the network size varied, the area of the simulation was also adjusted to accommodate the changing number of nodes. The rationale was to keep the neighbour count number constant in all scenarios. The neighbour count[1] was kept constant to a value of seven. Table 6.1 shows the corresponding values between network size and area dimensions.

- *Scale* – Most experiments in ns2 were conducted with participants that ranged in numbers between 20, 40 and 60. The number of participants always obeyed the formula $|\mathcal{M}| = |\mathcal{N}| - 10$, resulting in simulations where the total number of nodes was correspondingly 30, 50 and 70. In each experiment, participants were selected uniformly at random from the set of all nodes. Recall that this setup was necessary in order to honour the requirement of OntoMobil functioning even when all nodes connected in an ad hoc network are not running the OntoMobil protocol stack (Section 3.4). Nodes that were not participants only forwarded join requests to their neighbouring nodes and did not otherwise participate in the execution of the gossip or discovery protocols.

- *Concept Fanout and Ontology Size* – Throughout all experiments the concept fanout ($\mathtt{F}_c$) and the size of the ontology view ($|\mathcal{V}^{\mathtt{0}}|$) were kept constant at 4 and 10 concepts respectively. The choice for the concept fanout reflects the desire to encapsulate each gossip transmission in a single unicast message. To this end, the maximum transmission unit (MTU) for the 802.11 MAC was considered to be 1500 octets[2], which means that

[1] Defined in Kurkowski et al. [2005] as $\frac{(\pi \times r^2)}{(\frac{w \times h}{n})}$, where $r$ is the wireless range, $w, h$ the width and height of the area size and $n$ the number of nodes.

[2] The 802.11 1999 standard actually defines the size of the frame body in the MAC frame format to be 2312

it can comfortably encapsulate the XML network representation of 4 concepts. On the other hand, it is impossible to derive an average-sized ontology. A statistical analysis of 95 DAML+OIL ontologies that was conducted in Volz [2004] revealed huge variability in the number of concepts. Therefore, the value of 10 concepts was chosen as a trade-off between an ontology size that on one hand was large enough to allow the expression of discovery queries, while on the other it offered acceptable performance in terms of simulation processing time when used in networks with many service providers.

## 6.2 Summary of Analytical Evaluation

This section provides a brief summary of the evaluation results from previous chapters. In general, results obtained from the simulator developed as pat of this thesis that uses perfect uniform views with no message failures are more closely aligned with analytical results, rather than those obtained from the ns2 simulator. This is expected and is related to the increased statistical variability that mobility introduces in ns2. Nodes that are mobile and have a short-range wireless interfaces will form transient connections, which results in message loss even when a proactive routing protocol such as OLSR is used. In order to strengthen the claim of scalability, a shift must occur from the use of raw unicast in OntoMobil to more reliable protocols. Note that such protocols do not have to offer strong reliability guarantees such as those offered for example by Pagani and Rossi [1997] and Vollset and Ezhilchelvan [2005], but need only reduce the message failure rate to a level similar to that of 40 participants. The threshold of 40 participants is selected as it represents a network of average scale that is also closely aligned with the analytical results.

The comparison of the variability of the concept view size between the analytical model and the protocol implementation was carried out by interpreting the overall data set as two different samples. The samples that correspond to the recordings of each node after every gossip transmission constitute the *per node* samples and are assumed to belong to a common normal distribution. It was observed that the statistical variability between these samples was higher than expected (Fig. 3.6), which partly explained the fact that the deviation between

octets, though 1500 is the practical compatibility limit since it is the size of the frame body in 802.3.

the median ns2 values was higher than the median values obtained from the perfect views simulator (Fig. 3.10). This comparison used the values obtained from the stochastic analysis as baseline values. The two other factors that contributed to the deviation in the outcome of the ns2 simulator were imperfect views and message loss.

The samples that correspond to recordings of concept view sizes across all nodes over a single simulation round constitute the *per round* samples. These samples are also assumed to belong to a common distribution, which is the same distribution as the one derived from the *per node* samples. The per round derived distribution has less variability and the analytical results fall within the confidence interval of the simulation results (Fig. 3.11). Most experiments in this section depend on the size of the concept view across both dimensions (per round and per node). For example, matching progress occurs both across nodes and across rounds. Although an analytical model has not been derived for the metrics that are measured in this chapter, it is expected that the evaluation scenarios in Section 6.3 can provide a sufficient indication about the emerging trend, which together with the stochastic model in Chapter 3, can provide adequate predictive power.

The evaluation results in Chapter 4 verify the claim that discovery of services has probabilistic bounds. The correspondence between the analytical and simulation results offers an additional confirmation to the validity of the stochastic model. Specifically, it validates the assumption that each concept view contains concepts that are not correlated with concepts maintained in other concept views and in fact the set of concepts in the concept view constitute an independent and simple random sample. The evaluation in Chapter 4 mapped service discovery to concept discovery and it did not evaluate how concept matching influences the discovery of semantically heterogeneous services. This interaction is evaluated here with the aim of quantifying the affect of matching latency to service discovery.

## 6.3 Experiments

The following experiments evaluated the latency of the auxiliary gossip protocols in addition to the latency of the matching approaches and the discovery protocol.

### 6.3.1 Latency of the Join and Leave Protocols

The aim of this experiment was to evaluate the latency, in terms of rounds, of the join and leave protocols in OntoMobil. All experiments have so far only considered the case where a group of nodes have initiated the bootstrap process at the same time. Therefore, the execution of the gossip protocol was closely synchronised between all nodes. This experiment expanded the assumption by considering two separate cases. First, that of a new node that has joined an existing network and second that of an existing node that has disconnected from the network. In the first case, the experiment was designed to demonstrate the delay before a new node is integrated into the semantic overlay by measuring the latency before the new node reached the average concept view size of the existing participants. In the second case, the experiment was designed to demonstrate the delay of removing stale references by measuring the latency involved when the gossip protocol is used in the propagation of the leave notification.

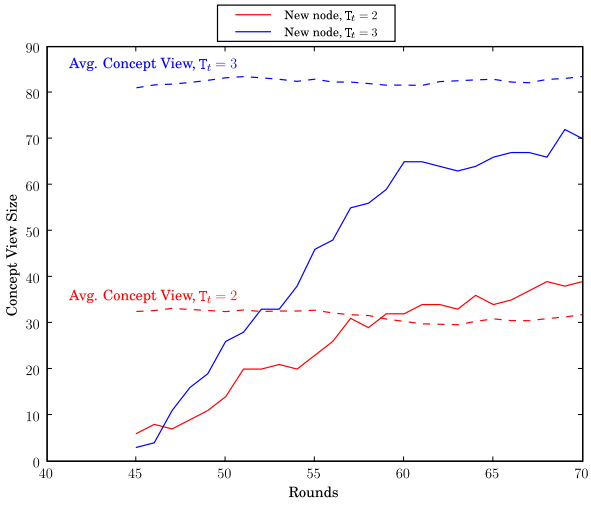


**Fig. 6.1**: The estimated latency in rounds until the concept view size of a new node levels-off as it approaches the average concept view size of the existing participants. Experiment invariants: node fanout ($\mathtt{F}_n = 2$) and age ($\mathtt{T}_a = 2$).

For the join protocol, the ns2 experimental setup used a network of $|\mathcal{N}| = 50$ nodes

with $|\mathcal{M}| = 40$ participants. The results represent the outcome of a single experimental run. After 40 seconds of simulation time, corresponding approximately to 45 rounds, a node that was selected randomly from the set of non-participants ($\mathcal{N} - \mathcal{M}$) initiated the join protocol. Figure 6.1 displays the round index against the corresponding values of the concept view size of the new node. This is an indication of the estimated latency until the concept view of the new node acquires a certain number of concepts. The measurements were made against two different gossip configurations that varied in the value of the concept ttl parameter. For the new node, recording of the concept view size started after the execution of the gossip protocol, which is after the join protocol had returned 20 unique node identifiers therefore crossing the join threshold ($\texttt{Parameter}_{\texttt{NodeView}} = 20$).

In the case where the ttl parameter had a value of two and therefore the average concept view size was $\sim 33$, the solid red line indicates that after $\sim 12$ rounds the concept view size of the new node reached the average, indicated by the dotted red line, and the size increase began to level. In the case where the ttl parameter was set to three and the average concept view size was $\sim 82$, the solid blue line began to level after approximately $\sim 15$ rounds. This represents a clear difference in the rate of increase between the two ttl values. As expected, a ttl value of three expands the propagation of concepts to more nodes, causing less received concepts to be dropped that would otherwise be inserted in the concept view. It is important to realise that the variability and the number of parameters that are inherent parts of the gossip protocol require the use of such measurements as mere approximations rather than accurate predictions.

The ns2 setup for the leave protocol used a combination of three different node populations ($|\mathcal{M}| = 20, 40, 60$) and two different values for the node fanout parameter ($\texttt{F}_n = 1, 2$). For each combination, an existing participant, was selected at random and initiated the leave notification after 40 seconds of simulation time. Figure 6.2 displays the number of rounds until all nodes received the leave notification at least once. The results represent the outcome after a single experimental run. When a node received the identifier of the departing node, all its references were cleared according to the algorithm in Section 3.5.2. The dotted lines, which represent the infection rate when the node fanout is two, show a higher rate of infection for all

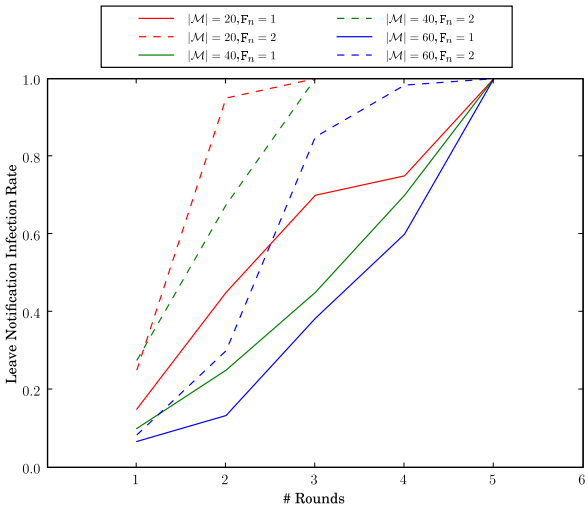


**Fig. 6.2**: The infection rate of the leave notification protocol for different number of participants and node fanout sizes. Experiment invariants: age ($\mathtt{T}_a = 2$) and ttl ($\mathtt{T}_t = 3$)

participant numbers. This is the expected behaviour since a higher node fanout means more participants receiving the leave notification. The $\mathtt{Parameter}_{\mathtt{AgeRemove}}$ was constant throughout all experiments to a value of 4. Although this is a high value, the combination of a lower $\mathtt{Parameter}_{\mathtt{AgeRemove}}$ value with a high number of participants ($|\mathcal{M}| > 20$) would sometimes cause the leave notification to terminate before it infected all participants. That finding revealed a dependency between the network size and the $\mathtt{Parameter}_{\mathtt{AgeRemove}}$ parameter. This is in-line with infection models from the gossip literature [Pittel, 1987, Eugster et al., 2004, Daley and Gani, 2001], where certain parameters have a phase transition effect, i.e., different parameter values result in different model behaviour. Specifically, the model followed by the leave protocol lies between the extremes of the *infect forever* and the *infect and die* models. As such, the number of rounds each participant remains infective and propagates the notification is important for the succesful completion of the epidemic.

### 6.3.2 Latency of the Suspend Protocol

The gossip suspend protocol was evaluated in terms of the associated latency from the moment at least one node transitions to the `Suspended` state to the moment when all participants have transitioned to that state and have ceased the gossip transmission. Figure 6.3 shows

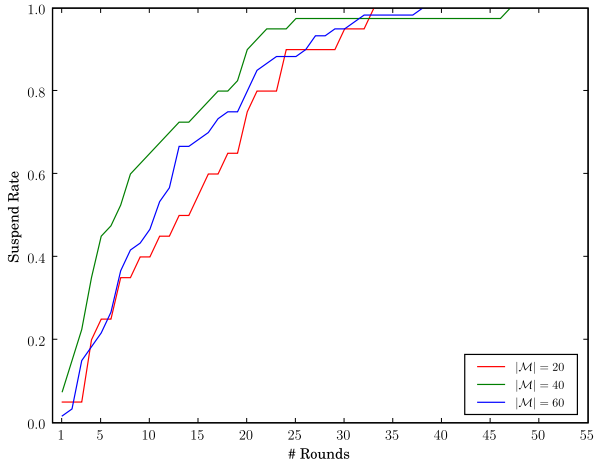


**Fig. 6.3**: Rate of gossip suspension against the number of gossip rounds. Experiment invariants: $SuspendThreshold = 0.9, STableThreshold = |\mathcal{M}|/2$

the suspend rate for different participants against the number of rounds. The results were obtained after a single experimental run. Round 1 in the x-axis represents the entry point where at least one node has suspended gossip. The y-axis depicts the ratio of suspended nodes versus the total number of nodes. The experiment was conducted with the perfect views simulator and therefore does not take into account failures and the affect of partial views. The $SuspendThreshold$ parameter was set to 0.9, while the $STableThreshold$ was set to half the size of the total number of participants, i.e., $STableThreshold = 10, 20, 30$ for $|\mathcal{M}| = 20, 40, 60$. Because the suspend protocol exploits the gossip protocol for communication, the figure shows that the suspend rate is logarithmic and very similar for the different number of participants. This behaviour is desirable and shows that the suspend protocol can scale. However, the high value for the suspend threshold parameter in this experiment

posed a potential problem for the remaining few nodes that continued the execution of the gossip protocol. Even though it guaranteed that the suspend protocol always terminated successfully, i.e., all nodes suspended, is also showed that the concept replication factor can be distorted by the continuous execution of few nodes. This is clearly indicated in the case of 40 nodes, where that last node that has not been suspended continued the gossip transmission for $\sim 20$ additional rounds before suspension. It is possible that this behaviour could have been mitigated by increasing the range in which the suspend probability $P(Suspend)$ was true, i.e., lowering the $SuspendThreshold$ parameter. A better alternative would be the further study of the correlation between network size and the $SuspendThreshold$ parameter. Specifically, what is the maximum range that $P(Suspend)$ can return true, while the probability that the suspend protocol terminates is close to 1.

### 6.3.3 Matching Latency

This experiment measured the latency of the progressive matching approach against three variables:

1. the number of participants,
2. the concept ttl value,
3. the three usage approaches for matching, i.e., union, ontology-only and probabilistic.

The template method was used to measure the matching progress, which required the construction of ontologies with predefined mapping relations. To this end, random mappings between different ontologies were incorporated in a way that was designed to approximate ontology mapping in a decentralised environment. A set of random unique concepts are generated that populate the ontology view of each participant. To establish a notion of similarity, a number of concepts are randomly selected from the ontology view of each node and a mapping is created between each selected concept (the source), and one target concept. The target concept was also randomly selected but this time from the ontology views of all nodes, excluding the node that contained the source concept. Each such mapping is bidirectional with both concepts having a reference to each other and is established with the addition of

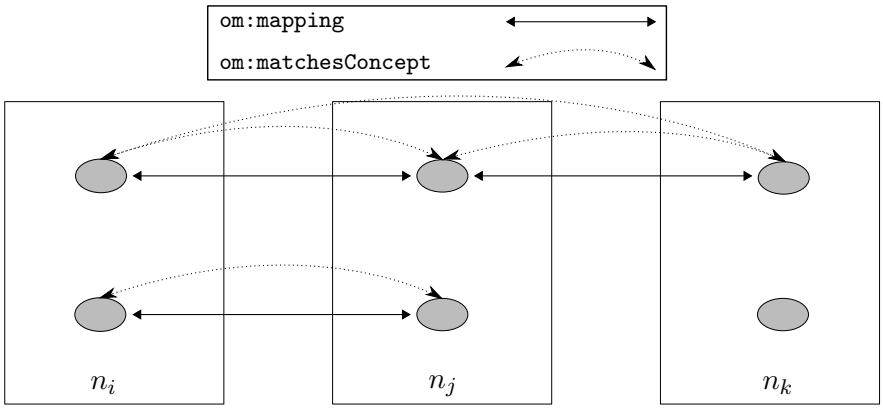


**Fig. 6.4**: The overlap in the predefined mapping relations facilitates the emergence of complete matching relations through the continuous concept dissemination.

an extra predicate in the concept's network representation. The mapping allowed the appearance of *transitive mapping* relations satisfying the definition of the *transitive matching* relation in Section 4.3.3. Note that a concept does not contain a reference to all other concepts in the transitive mapping. Instead, a concept maintains a mapping to only a subset of the concepts that compose the transitive relation. Figure 6.4 illustrates the difference. Because of the overlapping mapping relations, concept dissemination results in the eventual matching of all parts of a transitive mapping relation. Constructing the mapping relation in such a way is more realistic in a distributed environment, since each ontology is not required to have complete knowledge of all other ontologies. On the other hand, such an approach induces additional latency in order to facilitate complete semantic agreement.

For this experiment and the one described in Section 6.3.4, the concept matching process was adapted to establish matching relations only in the presence of predefined mappings. Figure 6.5 and Fig. 6.6 illustrate the latency, in terms of rounds, associated with the progressive matching approach for different number of nodes and ttl values. The results are averaged over 3 experimental runs. The y-axis represents the ratio of concepts that have established a matching relation versus the total number of concepts that *participate in all predefined mappings*. Furthermore, the recorded matching relations were the ones established in the ontology view of each node and not in the concept view. The aim was to show persistent matching relations, rather than matching relations in the concept view that can be discarded

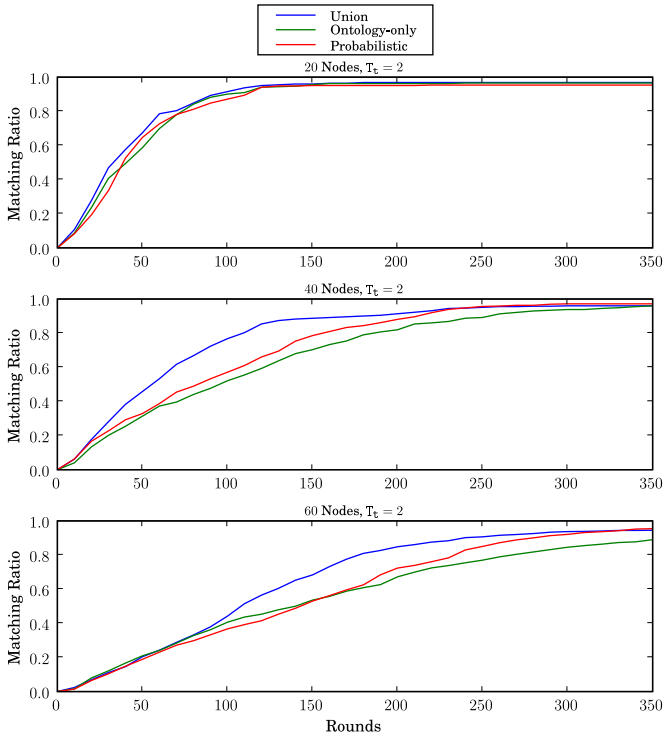


**Fig. 6.5**: Matching progress when $\mathtt{T}_t = 2$.

if the corresponding concepts were removed. This means that the presented ratio is always an underestimation since it does not take into account the matching relations established in the concept view. Accounting for matching relations in both views is demonstrated in Section 6.3.4 and is related to concept discovery.

Figure 6.5 shows the matching latency when the gossip ttl value was set to two. In the case of 20 nodes, there is a close to logarithmic increase in the number of matches, with 50% of the matches established after 50 rounds. The effect of the different usage matching approaches does not appear significant, with the union approach having a slightly higher rate of increase than the ontology-only and probabilistic approaches. As the number of nodes increases, therefore also increasing the number of concepts that must be matched, the union approach shows a higher rate of increase, though the matching rate of all three approaches is reduced to an approximately linear relationship.

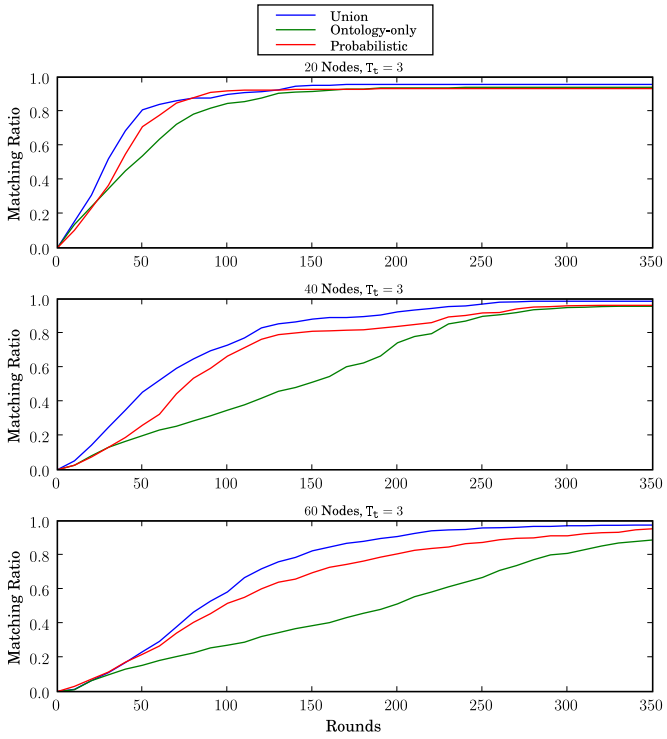


**Fig. 6.6**: Matching progress when $\mathtt{T}_t = 3$.

In Fig. 6.6, where the gossip ttl value is set to three, the difference in the rate of increase between the three usage approaches becomes more pronounced. In the case of 40 and 60 nodes, the ontology-only approach shows a linear rate of increase, while the union and the probabilistic approaches are closer to logarithmic rate. The results in both figures are close to expectations, with a rate of increase that is proportional to the ttl value and inversely proportional to the total number of concepts. As already mentioned, the difference between the three usage approaches does not appear as significant, which is a direct result of measuring concept matching only in the ontology view.

Both figures illustrate another interesting aspect, which is attributed to the construction of the predefined mapping relations. Because they were constructed in a randomised fashion, each experiment produced a different ratio between the number of concepts participating in mapping relations and the total number of concepts. In fact, in each experiment this

ratio was varied between approximately 0.4 and 0.6. A higher rate in this case implies that more comparisons are needed before the *matching ratio* reaches 1. However, since the measurements are consistent with expectations it indicates that such variations have little affect in the overall matching progress.

### 6.3.4 Discovery versus Matching

The aim of this experiment was to evaluate the impact of the progressive matching approach on the discovery of candidate ontologies. As a result, the first variation of the random walk discovery protocol was used, namely the local concept service request (Section 4.5.2) in combination with the union matching approach (Section 4.3.4). The number of nodes varied from 20 to 60 with each simulation running for approximately 350 rounds. Each simulation produced results that are averaged over 3 experimental runs. After the first 10 gossip rounds, each node transmitted a discovery query every second. The experimental setup in this section is similar to the one used for the discovery evaluation in Section 4.6. There were three differences however that reflect the different aim of this experiment:

1. the gossip parameters were all kept constant ($\mathtt{F}_n = 2, \mathtt{F}_c = 4, \mathtt{T}_a = 2, \mathtt{T}_t = 3$). The rationale was to reduce the number of variables so that the dependency between discovery and progressive matching becomes more apparent.

2. the query ttl was set to three instead of five ($\mathtt{Parameter}_{\mathtt{RequestTTL}} = 3$). This was necessary in order to reduce the high query failure rate that was demostrated in the experiments from Section 4.6.

3. the number of concepts used in each query was set to three instead of two ($|\mathcal{Q}| = 3$) with no semantic context attached to any of the query concepts ($|\mathcal{H}| = 0$). This change was done in order to simplify discovery and increase the concept sample size.

Each node constructed a discovery query by selecting three random concepts from the ontology view and then executing the algorithm in Listing 6. Two metrics were used to assess the performance of the discovery protocol against the progressive matching approach.

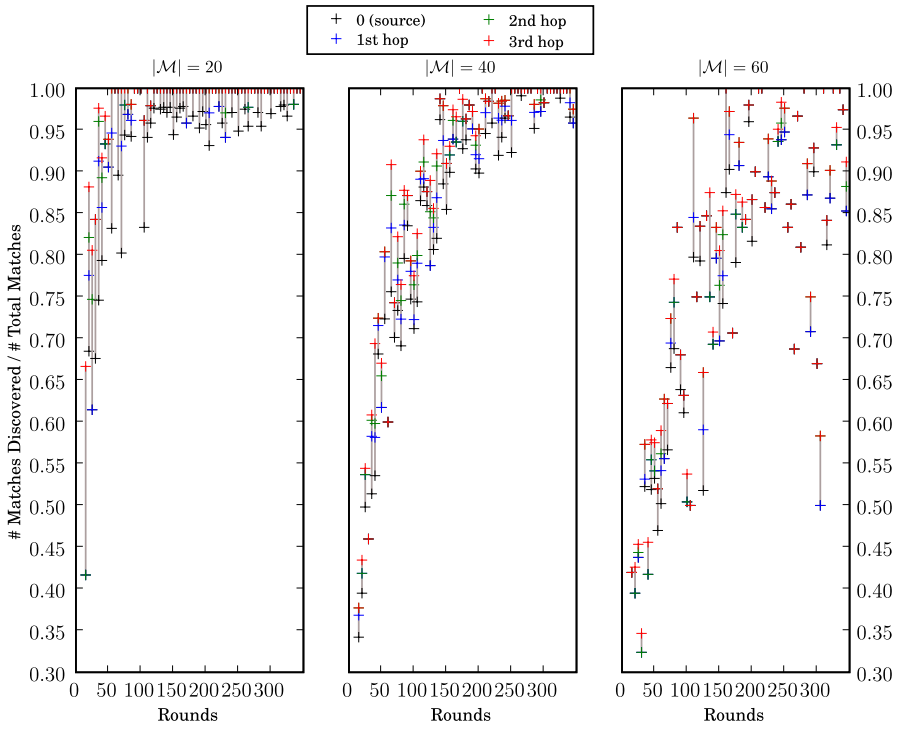


**Fig. 6.7**: Total matches ratio versus gossip rounds.

The first is the *total matches*. This metric indicates the degree to which all predefined mappings have been discovered. It is a ratio calculated by dividing the number of matching concepts recorded by each query concept in each hop of the random walk, i.e., the number obtained by $f_R(c)$, to the total number of concepts in the transitive mapping relation for which the query concept was member.

The second metric is the *current matches*. This is an alternative measure for the performance of the random walk since it reflects the current state of the matching progress rather than the ideal state that is captured by the total matches metric. It is calculated by dividing the number given by $f_R(c)$ in each hop against the number of matches that exist for concept $c$ across the network. In other words, if $c$ is replicated across $k$ nodes, the divisor is the cardinality of the set obtained from: $\bigcup_{i \in k} f_{\mathcal{V}_i^{\mathfrak{C}}}(c)$.

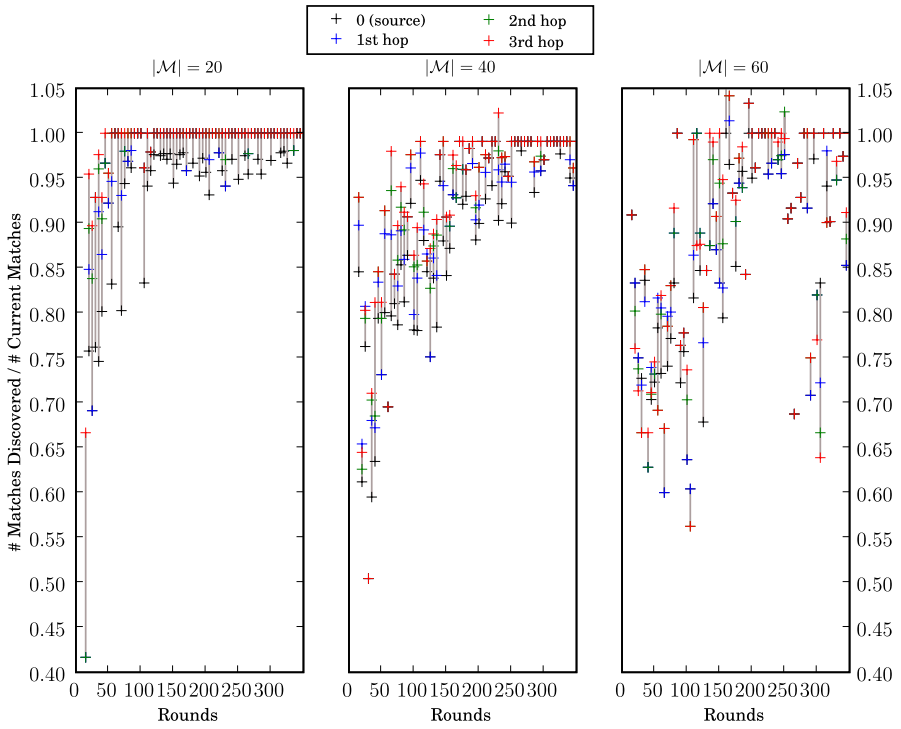


**Fig. 6.8**: Current matches ratio versus gossip rounds.

Figure 6.7 depicts the *total matches ratio* averaged over all queries and all query concepts. Since query concepts were randomly selected, it was possible that some would not belong to a mapping relation. These were excluded from the figure. Crosses represent the hops in the random walk with the bottom cross in each vertical line representing the ratio at the source node. Each plot reveals two types of discovery trends. The round-based discovery, which is the general trend as rounds increase and the hop-based discovery, which is depicted in each vertical line and spans a single round. The results indicate that the random walk had a greater impact on discovery during the initial rounds. In terms of the average hop-based ratio, the 20 nodes case has the least statistical variability with each hop having a significant gain in the plotted ratio. The 40 and 60 nodes cases showed higher statistical variability with each hop having a marginal increase in the matches that were discovered for the query concepts. The higher variability and the smaller per-hop gain can be attributed to the higher message loss

and the fact that due to the slower matching progress, it is the gossip rounds rather than the individual hops that have a stronger dependency on the discovery ratio. For all network sizes the initial increase was reduced as rounds progressed and more matching relations were being discovered at the query source. The round-based discovery trend shows the same logarithmic increase with the number of rounds as that observed in Section 6.3.3. The round-based trend better captures the dependency between discovery and matching. During the initial rounds, matching associations have not yet being established, so discovery ratio is low. This is more prominent in the 40 and 60 nodes cases, where at 30 rounds approximately 35% of potential matches have been discovered. As more concepts are progressively disseminated the concept matching process augments the network's shared knowledge increasing the discovery rate at the query source.

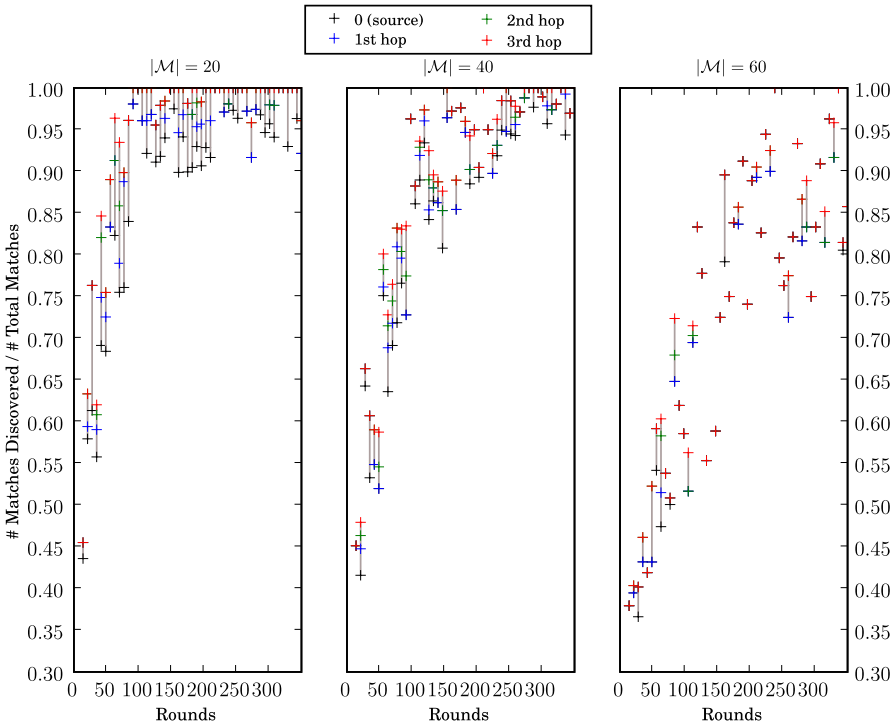


**Fig. 6.9**: Total matches ratio versus gossip rounds when the matching usage is probabilistic.

Figure 6.8 depicts the *current matches ratio* also averaged over all queries and all query concepts. In terms of the round-based discovery trend, results from this scenario would be expected to show a high ratio initially, since few matching relations would have been established. This means that a query discovering a small subset of matches would obtain a high ratio since this subset represents all the matches that may exist in the network. As rounds progress, a lower ratio would be expected since more matching relations are being established across the network but they are not being disseminated to all nodes. Finally, the ratio would reach 1 as matching relations are widely disseminated and become persistent across the participants. This trend is easier to observe in the 40 and 60 nodes cases. It is also possible for the current matches ratio to exceed 1, which means that a query might record more matches than those that have been established across the network. The cause of this is the matching relations already contained within each query (in $\mathcal{R}$). These are also used during the extraction of similar concepts in each hop of the random walk and the transitive nature of the matching relation may augment some concepts in the receiving node. Other than the higher round-based discovery trend, the current matches ratio is similar in terms of statistical variability and hop-based ratio to Fig. 6.7.

Figure 6.9 depicts the *total matches ratio* in an experiment where the matching usage approach was probabilistic. The aim was to show the impact of using a slower progressive matching approach on discovery. The figure shows a round-based discovery trend that has increased latency compared to the union approach and in addition the average hop-based gain is also lower than what is depicted in Fig. 6.7. The increased discovery latency and the smaller per hop gain constitute the trade-off for the use of reduced resources during the matching process.

## 6.4 Summary

This chapter provided a performance evaluation for several features of OntoMobil. The evaluation was conducted with the ns2 simulation and focused mainly on latency. Because of the eventual delivery semantics of many aspects of OntoMobil, i.e., suspend, matching and discovery, latency is an important trade-off.

In particular, the evaluation of the suspend protocol demonstrated that the number of nodes being suspended over the total number of nodes grows logarithmically with the number of rounds. This probabilistic state transitioning showed the scalability of the suspend protocol, but it also highlighted a potential issue where a few nodes would continue the gossip execution and thus remove all concepts from the concept view before they also suspend execution.

The matching evaluation was based on the comparison of the three usage approaches against two different sizes of the concept view. As expected the matching approach that utilised both views (union matching) showed the maximum latency reduction at a rate that grows close to logarithmic against the number of rounds. Although figures 6.7 and 6.8 tend to underestimate the matching ratio, since they capture only relations established in the ontology view, a larger concept view size trades-off increased processing with network-wide semantic agreement in less rounds. A similar trade-off can be made between the three different matching usage approaches.

Finally, the discovery evaluation compared the random walk protocol against the progress of semantic matching. It was observed that the random walk variation, which accepts requests from the ontology of the source node, only has a noticeable effect during the early stages of matching. This illustrated the strong correlation between the discovery of candidate ontologies and semantic matching.

# Chapter 7

# Conclusion

This thesis introduced the problem of discovering autonomous semantic services in mobile ad hoc networks. The thesis argued that a broad class of applications for MANETS, e.g., mobile games or pervasive computing, are inherently collaborative and would need functionality that is distributed across nodes. However, the dynamic nature of the environment means that the location of information cannot be hard-wired. Instead, the service paradigm can be used for on-demand discovery of required functionality. The use of a semantic service representation can further address the requirements for flexible service queries and automated interaction.

Contrary to existing discovery models, where it is assumed that all nodes share a common predefined service ontology, the model proposed in this thesis expands this assumption by allowing nodes to employ different ontologies for the description of services. The rationale is that in an open distributed system of which a mobile ad hoc network is an instance, node autonomy and absence of a priori agreement do not justify the assumption of a centralised semantic service representation. The admission of this expanded assumption and the nature of MANETs posed two problems that this thesis addressed. Specifically, the use of heterogeneous ontologies requires an efficient method to achieve semantic consensus, while the open and dynamic ad hoc environment requires a discovery protocol that scales.

## 7.1 Thesis Contributions

This thesis proposed OntoMobil, a model for semantic services that caters for mobile ad hoc networks and semantic decentralisation. Semantic decentralisation means that autonomous nodes can express services using different ontologies that are not a priori defined. The model relies on the decomposition of ontologies into concepts and the dissemination of these concepts through a novel gossip protocol. This randomised concept dissemination mechanism builds a semantic overlay that facilitates eventual semantic agreement between heterogeneous ontologies and provides a substrate for the discovery of services. A random walk protocol is used for the actual discovery. The protocol uses a two phase approach where semantic queries are first routed and subsequently evaluated at any provider node with a compatible ontology.

The model provides an efficient method for semantic matching, since it distributes the task of concept matching to all available nodes, achieving load balancing and trading matching latency over performance. The model has provable probabilistic bounds on discovery, maintaining a high discovery ratio as the scale of the network grows.

## 7.2 Thesis Summary

The motivation for this thesis stemmed from the need to allow uncoordinated interoperability of different applications in mobile ad hoc networks. The specific instance that was examined was the discovery of service-based applications that use ontologies to describe the functional characteristics of services. This thesis rejected the assumption of a common ontology and assumed that different mobile nodes provide services that are described by different ontologies. An analogy can be drawn between the spontaneous formation of MANETs without agreement on a common location and the spontaneous interaction of semantic services without agreement on a common ontology. In that respect, both location and ontology are seen as the context that is not agreed a priori but is necessary before communication is established.

The current state of the art in the domain of mobile ad hoc networks has not considered the problem of discovering semantically heterogeneous services in a mobile distributed topology.

All existing discovery protocols assume agreement on a common ontology as the fundamental contract between the provider and the consumer of services. Currently, the issues of semantic heterogeneity and topological decentralisation are being explored only by a number of P2P systems [Nejdl et al., 2002, Haase et al., 2004, Castano et al., 2003b, Tempich et al., 2004, Schlosser et al., 2002]. Although these systems provided a starting point in the design of a solution, mobile ad hoc networks have important differences with P2P networks. This thesis examined three such differences in Section 2.1.4, namely, robustness to node mobility, consideration of the processing capacity of nodes and adaptation to the number of mobile nodes.

The requirement for semantic integration that derived from the expanded assumption of heterogeneous ontologies together with the requirements for mobility, efficiency and scalability that derived from the domain of mobile ad hoc networks were used in the design of a distributed service discovery model called OntoMobil. OntoMobil relies on the decomposition of ontologies into concepts and the dissemination of these concepts through a novel gossip protocol. The gossip protocol forms the core of the OntoMobil model by building a randomised semantic overlay, which is the foundation for the semantic integration and service discovery processes of OntoMobil. Because the overlay relies on the randomisation properties of the gossip protocol, it is robust to mobility since it is essentially stateless. Distributing the process of semantic integration to all participating nodes provides the model with good load balancing capabilities. Finally, the model scales by being parameterised allowing to shift the trade-offs between latency and performance. A stochastic analysis of the gossip protocol produced a probability measure describing the variability in the replication of concepts across nodes and across rounds. This stochastic analysis provided a theoretical justification for the scalability claims of the model by incorporating all the characteristic parameters of the gossip protocol.

The stochastic analysis was evaluated against two different simulation types, one that was based on the ns2 simulator and implemented the full gossip specification and a second that removed message loss and assumed full membership views. The comparison showed that the stochastic model is a good approximation for the gossip protocol. The ns2 simulation

displayed a higher variability in the comparison, which was attributed to imperfect partial membership views and message loss. Message loss is caused because an unreliable unicast protocol is used for gossip transmission and the protocol does not include topology optimisations. Two remedies to the problem of unreliable gossip transmissions were proposed as future work in Section 3.5.6.

In order to achieve eventual semantic agreement between the different ontologies, the gossip protocol is complemented with a progressive concept matching algorithm. As concepts are periodically disseminated through the gossip protocol, matching relations are formed between the received concepts and those maintained in each node. Utilising the "infective" properties of the gossip protocol, such relations are further disseminated until complete pairwise concept matching is achieved. In this case the gossip protocol can be suspended and an initial investigation into such a suspension protocol was also provided.

Discovery of services is based on a random walk protocol that accepts as input a set of concepts composing the requested service and visits a sequence of random nodes with the aim of discovering concepts that match the query concepts. An evaluation between the probabilistic model of the random walk and simulation results have confirmed the correctness of the discovery process. High failure rates at large network sizes have also exposed reliability problems with the multi-hop nature of the random walk protocol.

Finally, Chapter 6 provided a performance evaluation of different aspects of the model, with the most important being latency of concept matching and concept discovery. The rate of concept matching increases logarithmically with rounds in networks of average scale (20, 40 nodes), reducing to linear matching in the 60 nodes case. With a gossip ttl parameter of three there is a more pronounced difference between the different matching methods. In terms of concept discovery, there is a strong correlation between the progress of concept matching and the discovery of matching relations for the query concepts. It was also observed that the random walk protocol obtained a higher discovery ratio in the early stages of matching. When matching nears completion, all matching relations can already be found in the ontology view of the node issuing the request

## 7.3 Discussion

The current implementation of OntoMobil is a a set of protocols in the ns2 simulator. However, it is envisioned that a real implementation would be a middleware component that is placed between service-based applications and the network layer. Such a middleware layer would need to have access to application specific details, so that it can route service requests and populate the ontology view and certain network events in order to set the characteristic parameters of the gossip protocol, i.e., the concept fanout ($\mathtt{F}_c$), the node fanout ($\mathtt{F}_n$), the age ($\mathtt{T}_a$) and the concept ttl ($\mathtt{T}_t$).

OntoMobil is based on two fundamental assumptions concerning the service-based applications and the network layer. These assumptions, the heterogeneity of the application semantics and the network-wide agreement on the characteristic parameters of the gossip protocol, need to be examined in order to judge the feasibility of OntoMobil.

It is true that currently the connection between ontologies, services and applications is still rudimentary. OntoMobil envisions a near-automated method, where application functionality can be transformed into a service with an associated semantic description. This will avoid the currently laborious and error-prone task of manual ontology creation. It will also enlarge the service ecosystem, further motivating the development of mobile applications as a set of interacting services. The transformation of standalone applications into service-based ones is already under way (e.g., IBM's WebSphere , Eclipse's web services plugin), aided by the paradigm shift that web services and service-oriented computing has brought. The mapping from service functionality to automated semantic description faces harder technical problems, mainly related to associating software functionality to a semantic encoding. Although such initial efforts to map software to semantic models have taken place since the early '90s [Devanbu et al., 1990], most efforts required the manual adjustments of the semantic model by domain experts. It is hoped that the semantic web will address this problem, assuming that as ontologies become more ubiquitous, a more automated integration with software services will emerge.

Since the main premise behind semantic services and ad hoc networks is the uncoordinated and spontaneous interactions between applications, OntoMobil nodes must be able to initially

agree on a set of values for the characteristic gossip parameters and subsequently adapt these values as the number of nodes changes. Currently, this is future work and it is expected that a different protocol will address parameter setting. Though it is possible for nodes to select individual values, concept replication properties similar to the ones derived from the stochastic analysis can only be obtained when parameter agreement is reached. Fortunately, such agreement can be eventual and can use a protocol similar to the gossip suspend protocol described in Section 4.4.

The overlay established by the gossip protocol requires a subset of node identifiers that are uniformly selected from the set of all participating nodes. This dependence of the gossip protocol on a partial membership service requires a membership protocol that is both efficient and has provable correctness. This thesis explored to some depth the affect of an lpb-like partial membership protocol and showed that under this class of protocols uniformity is correlated with view size. Other work by Bar-Yossef et al. [2006], Allavena et al. [2005] has shown that this is a recognised problem in the gossip community and novel solutions are being proposed to address both correctness and efficiency. OntoMobil depends on the partial membership services in order to scale as the number of network nodes increases. This decision follows recent results in the literature for scalable membership services [Luo et al., 2003, Bar-Yossef et al., 2006, Eugster et al., 2003, Allavena et al., 2005]. It can be envisioned that in situations where the network size is expected to be small, a full membership could be used, alleviating the requirement for an additional protocol and another source of variability.

A final remark must be made about the usefulness of randomisation to the extent that it was used in OntoMobil. A randomised membership view is indeed elegant with proven scalability properties. However, its use with a unicast transmission mechanism in MANETs has shown a clear demand either for topology optimisation in the membership view or for a reliability layer in the unicast transmission. The use of randomisation in other aspects of OntoMobil, i.e., during the selection of the concept fanout, adds uniformity and load balancing to the model with a simple algorithm. In hindsight, the design of a randomised protocol is a relatively easier task compared to its implementation. This is mainly due to the difficulty in debugging a protocol with built-in uncertainty, where a mismatch between

analytical properties and simulation results can either be due to statistical variance requiring further runs or a problem with either the analysis or the implementation or both.

## 7.4 Future Work

The need for a reliability layer for OntoMobil has been mentioned several times (Section 3.5.6 and 3.7). Another aspect of OntoMobil that requires future attention is the alteration of the leave protocol to use a soft state approach, rather than rely on an explicit disconnection procedure (Section 3.5.2). Incorporating this change into the current model can be made if a full membership service is assumed. One approach is to rely on the use of the randomised nature of transmission and use the property that with some probability each node will eventually contact every other node, unless failure occurs. A heuristic can associate the number of rounds a node identifier has not been seen with failure. This approach is similar to the one described in Renesse et al. [1998], but it requires keeping state for all nodes. A significant result would constitute the integration of partial uniform views with a failure detector.

At the moment, the random walk discovery protocol relies on the use of the membership view and as a result each visited node can be located multiple hops away (Section 4.5). This poses a problem in networks of large scale that exhibit a high rate of message failure ($> 15\%$). A discovery optimisation can use a random walk with single-hop neighbour unicasts, rather than the current multi-hop random walk. This would possibly offer a much improved reliability, though it would alter one of the model's realistic assumptions; that OntoMobil should operate without all nodes maintaining the OntoMobil stack.

At the moment, the stochastic model does not include a failure model (Section 3.6). This has an impact in the accuracy of the stochastic model in scenarios with high message failure rate ($> 15\%$). Integrating the probability of lost messages, will enhance the predictive power of the model, increasing the accuracy of the expected concept view size. The failure model introduced in Luo et al. [2003] can provide the initial framework that can be adapted for OntoMobil. Part of the changes that may be required would be to introduce rounds into the current model, since failures are round-based.

The random walk discovery protocol (Section 4.5) is specific to services. However, the

OntoMobil model is more generic and as such it can be used as an overlay for all semantic content. This is possible provided that the basic assumption of semantic heterogeneity is preserved and metadata constitute a large enough sample.

The current evaluation has not focused explicitly on different mobility scenarios. System performance results under various conditions of velocity and use of other routing protocols apart from OLSR would greatly enhance the understanding and applicability of OntoMobil in mobile ad hoc networks.

## 7.5 Conclusion

This thesis binds together many different aspects from software engineering, semantic modelling and epidemic protocols in a problem that is unusual at first but on closer inspection it hides a fundamental assumption that has not been questioned or explored before in the domain of mobile ad hoc networks. Since the inception of MANETs, the vision of decentralised and uncoordinated network connectivity was always central and this thesis has helped to extend this vision to the domain of semantic connectivity.

As with all technical work, there are advantages and disadvantages in the proposed approach. The decision to use a partial group membership view as the foundation of both the gossip and the discovery protocols gave a powerful abstraction without much additional overhead. The use of the randomised overlay offered another powerful abstraction, though the combination of the overlay with a multi-hop unreliable unicast protocol should, in retrospect, have enjoyed greater criticism in the beginning of this work. However, the most serious issue in OntoMobil is the assumption of multiple ontologies and the resulting requirement for semantic agreement. When the uncertainty in network conditions is combined with the challenging task of matching heterogeneous ontologies, it makes eventual agreement and discovery of information problematic. If instead of ontologies, a less formal metadata framework could be used that allowed flexibility in description but less “fuzziness” in matching, OntoMobil could provide a practical approach.

# Appendix A

# Stochastic vs. Simulation Results

Simulation results were obtained using the per round across nodes sample.

| parameters | | | | analysis | perfect views | | ns2 | |
|---|---|---|---|---|---|---|---|---|
| $\mathcal{M}$ | $\mathtt{F}_n$ | $\mathtt{T}_a$ | $\mathtt{T}_t$ | $E[V]$ | $\mu$ | StdError | $\mu$ | StdError |
| 20 | 1 | 1 | 2 | 9.0 | 8.8 | 1.9 | 9.9 | 3.6 |
| 20 | 1 | 1 | 3 | 16.3 | 16.6 | 3.1 | 18.0 | 5.1 |
| 20 | 1 | 2 | 2 | 16.9 | 18.8 | 3.2 | 21.6 | 6.6 |
| 20 | 1 | 2 | 3 | 38.6 | 39.6 | 5.6 | 41.4 | 9.7 |
| 20 | 2 | 1 | 2 | 16.3 | 17.3 | 2.5 | 20.6 | 6.9 |
| 20 | 2 | 1 | 3 | 37.9 | 37.5 | 5.2 | 40.0 | 10.4 |
| 20 | 2 | 2 | 2 | 30.9 | 33.2 | 5.0 | 37.1 | 11.2 |
| 20 | 2 | 2 | 3 | 74.9 | 75.2 | 3.8 | 72.0 | 14.4 |
| 40 | 1 | 1 | 2 | 9.2 | 9.2 | 1.7 | 8.9 | 2.2 |
| 40 | 1 | 1 | 3 | 17.8 | 18.2 | 2.8 | 18.0 | 4.1 |
| 40 | 1 | 2 | 2 | 18.9 | 20.0 | 2.5 | 20.8 | 3.9 |
| 40 | 1 | 2 | 3 | 46.9 | 47.9 | 5.1 | 49.1 | 7.7 |

| parameters | | | | analysis | perfect views | | ns2 | |
|---|---|---|---|---|---|---|---|---|
| $\mathcal{M}$ | $\mathtt{F}_n$ | $\mathtt{T}_a$ | $\mathtt{T}_t$ | $E[V]$ | $\mu$ | $\sigma$ | $\mu$ | $\sigma$ |
| 40 | 2 | 1 | 2 | 18.3 | 18.3 | 2.7 | 19.2 | 3.1 |
| 40 | 2 | 1 | 3 | 46.1 | 46.8 | 4.2 | 47.4 | 7.6 |
| 40 | 2 | 2 | 2 | 33.8 | 37.1 | 3.3 | 40.5 | 7.3 |
| 40 | 2 | 2 | 3 | 107.4 | 106.4 | 3.9 | 97.5 | 14.8 |
| 60 | 1 | 1 | 2 | 9.2 | 9.3 | 1.3 | 8.9 | 1.6 |
| 60 | 1 | 1 | 3 | 18.9 | 19.1 | 2.7 | 18.1 | 3.0 |
| 60 | 1 | 2 | 2 | 18.9 | 20.4 | 2.3 | 20.3 | 3.2 |
| 60 | 1 | 2 | 3 | 50.7 | 52.9 | 3.3 | 51.4 | 6.7 |
| 60 | 2 | 1 | 2 | 18.3 | 18.7 | 1.9 | 17.3 | 2.9 |
| 60 | 2 | 1 | 3 | 49.9 | 50.2 | 3.4 | 44.3 | 5.6 |
| 60 | 2 | 2 | 2 | 37.0 | 38.5 | 3.5 | 36.8 | 5.6 |
| 60 | 2 | 2 | 3 | 125.7 | 124.8 | 4.8 | 109.9 | 12.1 |
| 80 | 1 | 1 | 2 | 9.2 | 9.3 | 1.2 | - | - |
| 80 | 1 | 1 | 3 | 19.3 | 19.4 | 2.2 | - | - |
| 80 | 1 | 2 | 2 | 18.9 | 20.7 | 2.0 | - | - |
| 80 | 1 | 2 | 3 | 52.7 | 55.5 | 3.5 | - | - |
| 80 | 2 | 1 | 2 | 18.3 | 18.9 | 1.7 | - | - |
| 80 | 2 | 1 | 3 | 51.6 | 52.9 | 3.6 | - | - |
| 80 | 2 | 2 | 2 | 37.0 | 39.3 | 2.5 | - | - |
| 80 | 2 | 2 | 3 | 137.9 | 137.2 | 5.1 | - | - |

| parameters | | | | analysis | perfect views | | ns2 | |
|---|---|---|---|---|---|---|---|---|
| $\mathcal{M}$ | $\mathtt{F}_n$ | $\mathtt{T}_a$ | $\mathtt{T}_t$ | $E[V]$ | $\mu$ | $\sigma$ | $\mu$ | $\sigma$ |
| 100 | 1 | 1 | 2 | 9.2 | 9.4 | 1.1 | - | - |
| 100 | 1 | 1 | 3 | 19.3 | 19.7 | 1.9 | - | - |
| 100 | 1 | 2 | 2 | 18.9 | 20.8 | 1.8 | - | - |
| 100 | 1 | 2 | 3 | 54.2 | 56.7 | 3.8 | - | - |
| 100 | 2 | 1 | 2 | 18.3 | 19.0 | 1.6 | - | - |
| 100 | 2 | 1 | 3 | 53.4 | 54.1 | 3.3 | - | - |
| 100 | 2 | 2 | 2 | 37.0 | 40.1 | 2.5 | - | - |
| 100 | 2 | 2 | 3 | 146.6 | 145.6 | 5.5 | - | - |

# Appendix B

# Concept Specification

```
<?xml version='1.0' encoding='ISO-8859-1'?>
<rdf:RDF xml:lang="en"
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xml:base="http://www.dsg.cs.tcd.ie/~nedosa/om/1.0/CNR/">
<rdf:Property rdf:about="namespace">
   <rdfs:comment xml:lang="en">
      "Placeholder for the concept's namespace"
   </rdfs:comment>
   <rdfs:domain rdf:resource="rdfs:Class"/>
   <rdfs:range rdf:resource="rdfs:Literal"/>
</rdf:Property>
<rdf:Property rdf:about="ttl">
   <rdfs:comment xml:lang="en">
      "Placeholder for the concept's ttl"
   </rdfs:comment>
   <rdfs:domain rdf:resource="rdfs:Class"/>
   <rdfs:range rdf:resource="rdfs:Literal"/>
</rdf:Property>
<rdf:Property rdf:about="source">
   <rdfs:comment xml:lang="en">
      "Placeholder for source node identifier"
   </rdfs:comment>
```

```
   <rdfs:domain rdf:resource="rdfs:Class"/>
   <rdfs:range rdf:resource="rdfs:Literal"/>
</rdf:Property>
<rdf:Property rdf:about="matchesConcept">
   <rdfs:comment xml:lang="en">
      "Placeholder for concept URI"
   </rdfs:comment>
   <rdfs:domain rdf:resource="rdfs:Class"/>
   <rdfs:range rdf:resource="rdfs:Class"/>
</rdf:Property>
<rdf:Property rdf:about="matchesConceptInNode">
   <rdfs:comment xml:lang="en">
         "Placeholder for node id. To be used with the matchesConcept
         in coordination with matchesConcept placeholder"
   </rdfs:comment>
   <rdfs:domain rdf:resource="rdfs:Class"/>
   <rdfs:range rdf:resource="rdfs:Literal"/>
</rdf:Property>
<rdf:Property rdf:about="hasSuperClass">
   <rdfs:comment xml:lang="en">
      "Placeholder augmenting the concept's network representation
      with information about the concept's superclasses"
   </rdfs:comment>
   <rdfs:domain rdf:resource="rdfs:Class"/>
   <rdfs:range rdf:resource="rdfs:Class"/>
</rdf:Property>
<rdf:Property rdf:about="hasSubClass">
   <rdfs:comment xml:lang="en">
         "The concept's subclasses."
   </rdfs:comment>
   <rdfs:domain rdf:resource="rdfs:Class"/>
   <rdfs:range rdf:resource="rdfs:Class"/>
</rdf:Property>
<rdf:Property rdf:about="hasProperty">
   <rdfs:comment xml:lang="en">
         "The concept's properties"
```

```
   </rdfs:comment>
   <rdfs:domain rdf:resource="rdfs:Class"/>
   <rdfs:range rdf:resource="rdfs:Literal"/>
</rdf:Property>
<rdf:Property rdf:about="hasSuperProperty">
   <rdfs:comment xml:lang="en">
         "Placeholds augmenting the concept's network representation
         with information about the concept's inherited properties"
   </rdfs:comment>
   <rdfs:domain rdf:resource="rdfs:Class"/>
   <rdfs:range rdf:resource="rdfs:Literal"/>
</rdf:Property>
</rdf:RDF>
```

# Appendix C

# Service Specification

```
<?xml version='1.0' encoding='ISO-8859-1'?>
<rdf:RDF xml:lang="en"
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xmlns:om="http://www.dsg.cs.tcd.ie/~nedosa/om/1.0/SR/"
  xml:base="http://www.dsg.cs.tcd.ie/~nedosa/om/1.0/SR/">
<rdfs:Class rdf:about="Service">
  <rdfs:comment>The service class.</rdfs:comment>
  <rdfs:subClassOf rdf:resource="http://www.w3.org/2000/01/rdf-schema#Resource"/>
</rdfs:Class>
<rdf:Property rdf:about="hasInput">
   <rdfs:domain rdf:resource="om:Service"/>
   <rdfs:range rdf:resource="rdfs:Class"/>
</rdf:Property>
<rdf:Property rdf:about="hasOutput">
   <rdfs:domain rdf:resource="om:Service"/>
   <rdfs:range rdf:resource="rdfs:Class"/>
</rdf:Property>
</rdf:RDF>
```

# Bibliography

K. Aberer, P. Cudré-Mauroux, and M. Hauswirth. A framework for semantic gossiping. *SIGMOD Rec.*, 31(4):48–53, 2002. ISSN 0163-5808.

K. Aberer, P. Cudré-Mauroux, and M. Hauswirth. The Chatty Web: Emergent Semantics Through Gossiping. In *Proceedings of the 12th International Conference on World Wide Web (WWW'03)*, pages 197–206, New York, NY, USA, 2003. ACM Press. ISBN 1-58113-680-3.

K. Aberer, P. Cudré-Mauroux, A. M. Ouksel, T. Catarci, M.-S. Hacid, A. Illarramendi, V. Kashyap, M. Mecella, E. Mena, E. J. Neuhold, O. De Troyer, T. Risse, M. Scannapieco, F. Saltor, L. de Santis, S. Spaccapietra, S. Staab, and R. Studer. Emergent Semantics Principles and Issues. In *DASFAA 2004*, pages 25–38, 2004.

W. Adjie-Winoto, E. Schwartz, H. Balakrishnan, and J. Lilley. The design and implementation of an intentional naming system. In *Proceedings of the 17th ACM Symposium on Operating Systems Principles (SOSP'99)*, pages 186–201. ACM Press, December 1999. ISBN 1-58113-140-2.

A. Allavena, A. Demers, and J. E. Hopcroft. Correctness of a gossip based membership protocol. In *Proceedings of the twenty-fourth annual ACM symposium on Principles of distributed computing (PODC '05)*, pages 292–301, New York, NY, USA, 2005. ACM Press. ISBN 1-59593-994-2. doi: http://doi.acm.org/10.1145/1073814.1073871.

K. Arabshian and H. Schulzrinne. Gloserv: Global service discovery architecture. *mobiquitous*, 00:319–325, 2004.

K. Arnold, R. Scheifler, J. Waldo, B. O'Sullivan, A. Wollrath, B. O'Sullivan, and A. Wollrath. *Jini Specification*. Addison-Wesley Longman Publishing Co., Inc., 1999. ISBN 0201616343.

I. A. N. Authority. Service location protocol extensions, version 2 (slpv2), May 2003. URL `http://www.iana.org/assignments/svrloc-extensions`.

F. Baader, A. Borgida, and D. McGuinness. Matching in description logics: Preliminary results, 1998.

F. Baader, D. Calvanese, D. L. McGuinness, D. Nardi, and P. F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, 2003. ISBN 0-521-78176-0.

Z. Baida, J. Gordijn, and B. Omelayenko. A shared service terminology for online service provisioning. In *Proceedings of the 6th international conference on Electronic commerce (ICEC '04)*, pages 1–10, New York, NY, USA, 2004. ACM Press. ISBN 1-58113-930-6.

M. Balazinska, H. Balakrishnan, and D. Karger. INS/Twine: A Scalable Peer-to-Peer Architecture for Intentional Resource Discovery. In *International Conference on Pervasive Computing*, Zurich, Switzerland, August 2002.

Z. Bar-Yossef, R. Friedman, and G. Kliot. Rawms -: random walk based lightweight membership service for wireless ad hoc network. In *MobiHoc '06: Proceedings of the seventh ACM international symposium on Mobile ad hoc networking and computing*, pages 238–249, New York, NY, USA, 2006. ACM Press. ISBN 1-59593-368-9.

J. Beatty, G. Kakivaya, D. Kemp, T. Kuehnel, B. Lovering, B. Roe, C. S. John, J. Schlimmer, G. Simonnet, D. Walter, J. Weast, Y. Yarmosh, and P. Yendluri. Web services dynamic discovery (ws-discovery), April 2005.

D. Beckett. Rdf/xml syntax specification (revised), February 2004. URL `http://www.w3.org/TR/rdf-syntax-grammar/`. W3C Recommendation.

A. Beitz, M. Bearman, and A. Vogel. Service location in an open distributed environment. In *Proceedings on the Second International Workshop on Services in Distributed and Networked Environments*, pages 28–34, June 1995.

T. Berners-Lee. Www past and future. Presentation, 2003. URL `http://www.w3.org/2003/Talks/0922-rsoc-tbl/`.

T. Berners-Lee. Universal resource identifiers in www, 1994. URL `http://www.w3.org/Addressing/URL/uri-spec.txt`.

P. A. Bernstein. Middleware: a model for distributed system services. *Communications of the ACM*, 39(2):86–98, 1996. ISSN 0001-0782.

*Specification of the Bluetooth System*. Bluetooth Consortium, December 1999. URL `http://www.bluetooth.com`.

D. Booth, H. Haas, F. McCabe, E. Newcomer, M. Champion, C. Ferris, and D. Orchard. Web services architecture. W3C Technical Report, February 2004. `http://www.w3.org/TR/2004/NOTE-ws-arch-20040211/`.

D. Brickley and R. Guha. Resource Description Framework RDF Schema Specification 1.0, March 2000. URL `http://www.w3.org/TR/2000/CR-rdf-schema-20000327`. W3C.

J. Broch, D. B. Johnson, and D. A. Maltz. The dynamic source routing protocol for mobile ad hoc networks. Internet-Draft Version 03, IETF, October 1999.

S. Capkun, L. Buttyan, and J.-P. Hubaux. Self-Organized Public-Key Management for Mobile Ad Hoc Networks. *IEEE Transactions on Mobile Computing*, page 17, 2003.

S. Castano, A. Ferrara, and S. Montanelli. H-match: an algorithm for dynamically matching ontologies in peer-based systems. In *SWDB*, pages 231–250, 2003a.

S. Castano, A. Ferrara, S. Montanelli, E. Pagani, and G. Rossi. Ontology-Addressable Contents in P2P Networks. In *Proceedings of the 1st WWW International Workshop on Semantics in Peer-to-Peer and Grid Computing (SemPGRID '03)*, Budapest, Hungary, May 2003b.

S. Castano, A. Ferrara, and S. Montanelli. Ontology-based interoperability services for semantic collaboration in open networked systems. In *Proceedings of the 1st International Conference on Interoperability of Enterprise Software and Applications (INTEROP-ESA 2005)*, Geneva, Switzerland, February 2005.

P. Castro, B. Greenstein, R. Muntz, P. Kermani, C. Bisdikian, and M. Papadopouli. Locating application data across service discovery domains. In *Proceedings of the 7th annual international conference on Mobile computing and networking (MobiCom'01)*, pages 28–42, New York, NY, USA, 2001. ACM Press. ISBN 1-58113-422-3.

D. W. Chadwick. *Understanding X.500 (The Directory)*. International Thompson Publishing, unknown 1996. URL `http://www.cs.kent.ac.uk/pubs/1996/2051`.

D. Chakraborty, A. Joshi, T. Finin, and Y. Yesha. Gsd: A Novel Group Based Service Discovery Protocol for MANETs. In *Proceedings of the 4th IEEE Conference on Mobile and Wireless Communications Networks (MWCN'02)*. IEEE Press, September 2002.

T. Clausen, P. Jacquet, A. Laouiti, P. Minet, P. Muhlethaler, A. Qayyum, and L. Viennot. Optimized Link State Routing Protocol. Internet Draft (draft-ietf-manet-olsr-06.txt), IETF, September 1 2001. Work in Progress.

M. Corporation. Universal plug and play: Background, 1999.

G. Coulouris, J. Dollimore, and T. Kindberg. *Distributed systems (4th ed.): concepts and design*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2005.

W. Cox, F. Cabrera, G. Copeland, T. Freund, J. Klein, T. Storey, and S. Thatte. Web services transaction (ws-transaction), 2004. URL `http://dev2dev.bea.com/pub/a/2004/01/ws-transaction.html`.

S. E. Czerwinski, B. Y. Zhao, T. D. Hodes, A. D. Joseph, and R. H. Katz. An architecture for a secure service discovery service. In *Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking (MobiCom'99)*, pages 24–35, New York, NY, USA, 1999. ACM Press. ISBN 1-58113-142-9.

D. J. Daley and J. Gani. *Epidemic Modelling: An Introduction*. Cambridge University Press, 1st edition, 2001. ISBN 0521014670.

J. de Bruijn, C. Bussler, J. Domingue, D. Fensel, M. Hepp, U. Keller, M. Kifer, B. Knig-Ries, J. Kopecky, R. Lara, H. Lausen, E. Oren, A. Polleres, D. Roman, J. Scicluna, and M. Stollberg. Web service modeling ontology (wsmo). W3C Member Submission, June 2005. `http://www.w3.org/Submission/2005/SUBM-WSMO-20050603/`.

A. Demers, D. Greene, C. Hauser, W. Irish, J. Larson, S. Shenker, H. Sturgis, D. Swinehart, and D. Terry. Epidemic Algorithms for Replicated Database Maintenance. In *Proceedings of the 6th annual ACM Symposium on Principles of Distributed Computing (PODC'87)*, pages 1–12, New York, NY, USA, 1987. ACM Press. ISBN 0-89791-239-4.

P. T. Devanbu, R. J. Brachman, P. G. Selfridge, and B. W. Ballard. Lassiea knowledge-based software information system. In *Proceedings of the 12th international conference on Software engineering (ICSE '90)*, pages 249–261, Los Alamitos, CA, USA, 1990. IEEE Computer Society Press. ISBN 0-89791-349-3.

A. Doan, J. Madhavan, P. Domingos, and A. Halevy. Learning to map between ontologies on the semantic web. In *Proceedings of the 11th international conference on World Wide Web (WWW '02)*, pages 662–673, New York, NY, USA, 2002. ACM Press. ISBN 1-58113-449-5.

S. Dolev, E. Schiller, and J. L. Welch. Random walk for self-stabilizing group communication in ad hoc networks. *IEEE Transactions on Mobile Computing*, 5(7):893–905, 2006. ISSN 1536-1233.

J. Dowling, E. Curran, R. Cunningham, and V. Cahill. Using feedback in collaborative reinforcement learning to adaptively optimize manet routing. *IEEE Transactions on Systems, Man and Cybernetics, Part A,*, 35(3):360–372, May 2005. ISSN 1083-4427. URL `https://www.cs.tcd.ie/publications/tech-reports/reports.05/TCD-CS-2005-38.pdf`.

M. Dumas, J. O'Sullivan, M. Hervizadeh, D. Edmond, and A. H. M. ter Hofstede. Towards a semantic framework for service description. In *Proceedings of the IFIP TC2/WG2.6 Ninth*

*Working Conference on Database Semantics*, pages 277–291, Deventer, The Netherlands, 2003. Kluwer, B.V. ISBN 1-4020-7351-8.

P. T. Eugster, R. Guerraoui, S. B. Handurukande, P. Kouznetsov, and A.-M. Kermarrec. Lightweight Probabilistic Broadcast. *ACM Transactions on Computer Systems*, 21(4):341–374, 2003. ISSN 0734-2071.

P. T. Eugster, R. Guerraoui, A.-M. Kermarrec, and L. Massoulié. Epidemic information dissemination in distributed systems. *IEEE Computer*, 37(5):60–67, 2004.

H. Füßler, M. Käsemann, M. Mauve, H. Hartenstein, and J. Widmer. Contention-based forwarding for mobile ad-hoc networks. *Elsevier's Ad Hoc Networks*, 1(4):351 – 369, 2003.

A. J. Ganesh, A.-M. Kermarrec, and L. Massouli. Scamp: Peer-to-peer lightweight membership service for large-scale group communication. In *Third International Workshop on Networked Group Communications (NGC 2001)*, London, UK, November 2001.

M. Gudgin, M. Hadley, N. Mendelsohn, J.-J. Moreau, and C. H. F. Nielsen. Soap version 1.2 part 1: Messaging framework, June 2003. URL `http://www.w3.org/TR/soap12-part1/`. W3C Recommendation.

E. Guttman, C. Perkins, J.Veizades, and M.Day. *Service Location Protocol, Version 2*. IETF, 1999.

H. Haas and A. Brown. Web services glossary. W3C Technical Report, February 2004. `http://www.w3.org/TR/2003/WD-ws-gloss-20030808/`.

Z. J. Haas, J. Y. Halpern, and L. Li. Gossip-based ad hoc routing. *IEEE/ACM Transactions on Networking*, 14(3):479–491, 2006. ISSN 1063-6692.

P. Haase, R. Siebes, and F. van Harmelen. Peer selection in peer-to-peer networks with semantic topologies. In M. Bouzeghoub, editor, *Proceedings of the 1st International Conference on Semantics in a Networked World (ICNSW'04)*, volume 3226 of *Lecture Notes in Computer Science*, pages 108–125. Springer Verlag, June 2004.

J. Heflin. Owl web ontology language use cases and requirements. URL `http://www.w3.org/TR/webont-req/#onto-def`. W3C Recommendation, 2004.

S. Helal, N. Desai, V. Verma, and C. Lee. Konark – a Service Discovery and Delivery Protocol for Ad-Hoc Networks. In *Proceedings of the 3rd IEEE Conference on Wireless Communication Networks (WCNC'02)*. IEEE, March 2002.

R. Hermann, D. Husemann, M. Moser, M. Nidd, C. Rohner, and A. Schade. Deapspace: Transient ad-hoc networking of pervasive devices. In *Proceedings of the 1st ACM International Symposium on Mobile Ad Hoc Networking & Computing*, pages 133–134. IEEE Press, 2000. ISBN 0-7803-6534-8.

M. N. Huhns and M. P. Singh. Service-oriented computing: Key concepts and principles. *IEEE Internet Computing*, 9(1):75–81, 2005.

M. Jelasity and Ö. Babaoglu. T-man: Gossip-based overlay topology management. In S. Brueckner, G. D. M. Serugendo, D. Hales, and F. Zambonelli, editors, *Engineering Self-Organising Systems*, volume 3910 of *Lecture Notes in Computer Science*, pages 1–15. Springer, 2005. ISBN 3-540-33342-8.

Y. Kalfoglou and M. Schorlemmer. Ontology mapping: the state of the art. *The Knowledge Engineering Review*, 18(1):1–31, 2003. ISSN 0269-8889.

P. Kinney. Zigbee technology: Wireless control that simply works. www.zigbee.org/resources, October 2003. Whitepaper.

M. Klein. Combining and relating ontologies: an analysis of problems and solutions. In A. Gomez-Perez, M. Gruninger, H. Stuckenschmidt, and M. Uschold, editors, *Workshop on Ontologies and Information Sharing, IJCAI'01*, Seattle, USA, August 2001.

M. Klein, B. Konig-Ries, and P. Obreiter. Service rings - a semantic overlay for service discovery in ad hoc networks. In *14th International Workshop on Database and Expert Systems Applications (DEXA'03)*, Los Alamitos, CA, USA, 2003. IEEE Computer Society.

G. Kortuem, J. Schneider, D. Preuitt, T. G. C. Thompson, S. Fickas, and Z. Segall. When peer-to-peer comes face-to-face: Collaborative peer-to-peer computing in mobile ad hoc networks. *p2p*, 00:0075, 2001.

U. C. Kozat and L. Tassiulas. Network layer support for service discovery in mobile ad hoc networks. In *IEEE INFOCOM*, volume 22, pages 1965–1975, March 2003.

U. C. Kozat and L. Tassiulas. Service discovery in mobile ad hoc networks: an overall perspective on architectural choices and network layer support issues. *Ad Hoc Networks*, 2 (1):23–44, 2004.

S. Kurkowski, T. Camp, and M. Colagrosso. Manet simulation studies: the incredibles. *SIGMOBILE Mobile Computing and Communications Review*, 9(4):50–61, 2005. ISSN 1559-1662.

H. Lausen, J. de Bruijn, A. Polleres, and D. Fensel. Wsml - a language framework for semantic web services. In *Rule Languages for Interoperability*, 2005.

V. Lenders, M. May, and B. Plattner. Service discovery in mobile ad hoc networks: A field theoretic approach. In *International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM)*. IEEE, June 2005.

J. H. Li, M. yu, R. Levy, and A. Teittinen. A mobility-resistant efficient clustering approach for ad hoc and sensor networks. *SIGMOBILE Mobile Computing and Communication Review ($MC^2R$)*, 10(2):1–12, 2006. ISSN 1559-1662.

L. Li and I. Horrocks. A Software Framework for Matchmaking based on Semantic Web Technology. In *Proceedings of the 12th International Conference on World Wide Web (WWW'03)*, pages 331–339, New York, NY, USA, 2003a. ACM Press. ISBN 1-58113-680-3.

L. Li and I. Horrocks. A software framework for matchmaking based on semantic web technology. In *Proceedings of the 12th international conference on World Wide Web (WWW'03)*, pages 331–339, New York, NY, USA, 2003b. ACM Press. ISBN 1-58113-680-3.

A. Löser, F. Naumann, W. Siberski, W. Nejdl, and U. Thaden. Semantic overlay clusters within super-peer networks. In K. Aberer, V. Kalogeraki, and M. Koubarakis, editors, *DBISP2P*, volume 2944 of *Lecture Notes in Computer Science*, pages 33–47. Springer, 2003a. ISBN 3-540-20968-9.

A. Löser, W. Siberski, M. Wolpers, and W. Nejdl. Information integration in schema-based peer-to-peer networks. In *Proceedings of the Conference on Advanced Information Systems Engineering*, June 2003b.

J. Luo, P. T. Eugster, and J.-P. Hubaux. Route Driven Gossip: Probabilistic Reliable Multicast in Ad Hoc Networks. In *Proceedings of the 22nd IEEE INFOCOM*, pages 2229–2239, 2003.

J. Luo, P. T. Eugster, and J.-P. Hubaux. Pilot: Probabilistic Lightweight Group Communication System for Ad Doc Networks. *IEEE Transactions on Mobile Computing*, 3(2): 164–179, 2004.

J. Madhavan, P. A. Bernstein, P. Domingos, and A. Y. Halevy. Representing and reasoning about mappings between domain models. In *Eighteenth national conference on Artificial intelligence*, pages 80–86, Menlo Park, CA, USA, 2002. American Association for Artificial Intelligence. ISBN 0-262-51129-0.

D. Martin, M. Burstein, J. Hobbs, O. Lassila, D. McDermott, S. McIlraith, S. Narayanan, M. Paolucci, B. Parsia, T. Payne, E. Sirin, N. Srinivasan, and K. Sycara. OWL-S: Semantic Markup for Web Services, November 2004. URL `http://www.w3.org/Submission/OWL-S`. W3C.

S. McCanne and S. Floyd. ns, Network Simulator, 1997.

D. L. McGuinness and F. van Harmelen. Owl web ontology language overview, February 2004. URL `http://www.w3.org/TR/owl-features/`. W3C Recommendation.

B. McKee, D. Ehnebuske, and D. Rogers. Uddi version 2.0 api specification, June 2001.

Microsoft, IBM, Hitachi, IONA, Arjuna Technologies, and BEA Systems. Specification of WS-Coordination, 2005. URL `http://www.iona.com/devcenter/articles/pub_aug_2005_RC1.1/WS-Coordination.pdf`. `http://www.iona.com/devcenter/articles/pub_aug_2005_RC1.1/WS-Coordination.pdf` (accessed Oct. 3, 2005).

G. A. Miller. Wordnet: a Lexical Database for English. *Communications of the ACM*, 38 (11):39–41, 1995. ISSN 0001-0782.

R. Motwani and P. Raghavan. *Randomized algorithms*. Cambridge University Press, New York, NY, USA, 1995. ISBN 0-521-47465-5.

S. Mullender, editor. *Distributed systems*. ACM Press, New York, NY, USA, 1989. ISBN 0-201-41660-3.

C. S. R. Murthy and B. Manoj. *Ad Hoc Wireless Networks: Architectures and Protocols*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 2004. ISBN 013147023X.

W. Nejdl, B. Wolf, C. Qu, S. Decker, M. Sintek, A. Naeve, M. Nilsson, M. Palmér, and T. Risch. Edutella: A P2P Networking Infrastructure Based on RDF. In *Proceedings of the 11th International Conference on World Wide Web (WWW'02)*, pages 604–615, New York, NY, USA, 2002. ACM Press. ISBN 1-58113-449-5.

W. Nejdl, W. Siberski, and M. Sintek. Design issues and challenges for rdf- and schema-based peer-to-peer systems. *ACM SIGMOD Record*, 32(3):41–46, 2003. ISSN 0163-5808.

Y. Ni, U. Kremer, A. Stere, and L. Iftode. Programming ad-hoc networks of mobile and resource-constrained devices. *SIGPLAN Not.*, 40(6):249–260, 2005. ISSN 0362-1340. doi: http://doi.acm.org/10.1145/1064978.1065040.

M. Nidd. Service discovery in deapspace. *IEEE Personal Communications*, 8(4), August 2001.

N. F. Noy. Semantic integration: a survey of ontology-based approaches. *ACM SIGMOD Record*, 33(4):65–70, 2004. ISSN 0163-5808.

N. F. Noy and M. A. Musen. Prompt: Algorithm and tool for automated ontology merging and alignment. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on Innovative Applications of Artificial Intelligence*, pages 450–455. AAAI Press / The MIT Press, 2000. ISBN 0-262-51112-6.

O2. Odmg oql user manual, 1998. URL `http://www.cis.upenn.edu/~cis550/oql.pdf`.

OMG. Query service specification, 2000. URL `http://www.omg.org/cgi-bin/doc?formal/2000-06-23`.

E. Pagani and G. P. Rossi. Reliable broadcast in mobile multihop packet networks. In *MobiCom '97: Proceedings of the 3rd annual ACM/IEEE international conference on Mobile computing and networking*, pages 34–42, New York, NY, USA, 1997. ACM Press. ISBN 0-89791-988-2.

S. PalChaudhuri, J.-Y. L. Boudec, and M. Vojnovic. Perfect Simulations for Random Trip Mobility Models. In *Proceedings of the 38th annual Symposium on Simulation (ANSS'05)*, pages 72–79, Washington, DC, USA, 2005. IEEE Computer Society. ISBN 0-7695-2322-6.

M. Paolucci and K. Sycara. Autonomous semantic web services. *IEEE Internet Computing*, 07(5):34–41, 2003. ISSN 1089-7801.

M. Paolucci, T. Kawamura, T. R. Payne, and K. P. Sycara. Semantic Matching of Web Services Capabilities. In *Proceedings of the 1st International Semantic Web Conference ISWC'02*, pages 333–347, London, UK, 2002. Springer Verlag. ISBN 3-540-43760-6.

C. E. Perkins. *Ad hoc networking*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2001. ISBN 0-201-30976-9.

C. E. Perkins, T. Clausen, and P. Jacquet. *Multicast With Minimal Congestion Using Connected Dominating Sets*. IETF, 2005. http://tools.ietf.org/wg/manet/draft-perkins-manet-smurf-00.txt.

H. Pinto, A. Prez, and J. Martins. Some issues on ontology integration, 1999. URL `citeseer.ist.psu.edu/pinto99some.html`.

B. Pittel. On spreading a rumor. *SIAM Journal on Applied Mathematics*, 47(1):213–223, 1987. ISSN 0036-1399. doi: http://dx.doi.org/10.1137/0147013.

E. Rahm and P. A. Bernstein. A survey of approaches to automatic schema matching. *The VLDB Journal*, 10(4):334–350, 2001. ISSN 1066-8888.

S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Schenker. A scalable content-addressable network. In *Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications (SIGCOMM'01)*, pages 161–172, New York, NY, USA, 2001. ACM Press. ISBN 1-58113-411-8.

O. V. Ratsimor, D. Chakraborty, S. Tolia, D. Khushraj, A. Kunjithapatham, A. Joshi, T. Finin, and Y. Yesha. Allia: Alliance-based Service Discovery for Ad-Hoc Environments. In *ACM Mobile Commerce Workshop*, September 2002.

R. V. Renesse, Y. Minsky, and M. Hayden. A gossip-style failure detection service. In *Proceedings of Middleware '98*, September 1998.

A. Rowstron and P. Druschel. Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems. In *IFIP/ACM International Conference on Distributed Systems Platforms (Middleware)*, pages 329–350, November 2001.

F. Sailhan and V. Issarny. Scalable Service Discovery for MANET. In *Proceedings of the 3rd IEEE International Conference on Pervasive Computing and Communications (PERCOM'05)*, pages 235–244, Washington, DC, USA, 2005. IEEE Computer Society. ISBN 0-7695-2299-8.

S. Schade, A. Sahlmann, M. Lutz, F. Probst, and W. Kuhn. Comparing approaches for semantic service description and matchmaking. In R. Meersman and Z. Tari, editors, *CoopIS/DOA/ODBASE (2)*, volume 3291 of *Lecture Notes in Computer Science*, pages 1062–1079. Springer, 2004. ISBN 3-540-23662-7.

F. A. Schaeffer. *Genesis in Space and Time; The Flow of Biblical History (Bible Commentary for Layman)*. Regal Books, 1972.

M. Schlosser, M. Sintekand, S. Decker, and W. Nejdl. A scalable and ontology-based p2p infrastructure for semantic web services. In *Proceedings of the Second International Conference on Peer-to-Peer Computing (P2P 2002)*, pages 104–111, 2002.

C. Schmidt and M. Parashar. A peer-to-peer approach to web service discovery. *World Wide Web*, 7(2):211–229, 2004. ISSN 1386-145X.

N. Shadbolt, T. Berners-Lee, and W. Hall. The semantic web revisited. *IEEE Intelligent Systems*, 21(3):96–101, 2006. ISSN 1541-1672.

P. Shvaiko and J. Euzenat. A survey of schema-based matching approaches. *Journal of Data Semantics*, pages 146–171, 2005.

C. Skouteli, G. Samaras, and E. Pitoura. Concept-Based Discovery of Mobile Services. In *Proceedings of the 6th International Conference on Mobile Data Management (MDM'05)*, pages 257–261, New York, NY, USA, 2005. ACM Press. ISBN 1-59593-041-8.

M. Stal. Web services: beyond component-based computing. *Communications of the ACM*, 45(10):71–76, 2002. ISSN 0001-0782.

I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications (SIGCOMM'01)*, pages 149–160, New York, NY, USA, 2001. ACM Press. ISBN 1-58113-411-8.

K. Sycara, S. Widoff, M. Klusch, and J. Lu. LARKS: Dynamic Matchmaking Among Heterogeneous Software Agents in Cyberspace. *Autonomous Agents and Multi-Agent Systems*, 5(2):173–203, June 2002.

C. Szyperski. *Component Software*. Addison-Wesley Professional, November 2002. ISBN 0201745720.

H. Tangmunarunkit, S. Decker, and C. Kesselman. Ontology-based resource matching in the grid - the grid meets the semantic web. In D. Fensel, K. P. Sycara, and J. Mylopoulos,

editors, *International Semantic Web Conference*, volume 2870 of *Lecture Notes in Computer Science*, pages 706–721. Springer, 2003. ISBN 3-540-20362-1.

C. Tempich, S. Staab, and A. Wranik. Remindin': semantic query routing in peer-to-peer networks based on social metaphors. In *Proceedings of the 13th international conference on World Wide Web (WWW'04)*, pages 640–649, New York, NY, USA, 2004. ACM Press. ISBN 1-58113-844-X.

N. T.J.Bailey. *The Mathematical Theory of Infectious Diseases and its Applications*. Griffin, 2nd edition, 1975. ISBN 0-8564264-231-8.

D. Trastour, C. Bartolini, and J. Gonzalez-Castillo. A semantic web approach to service description for matchmaking of services. In I. F. Cruz, S. Decker, J. Euzenat, and D. L. McGuinness, editors, *SWWS*, pages 447–461, 2001.

Y.-C. Tseng, S.-Y. Ni, Y.-S. Chen, and J.-P. Sheu. The broadcast storm problem in a mobile ad hoc network. *Wireless Networks*, 8(2/3):153–167, 2002. ISSN 1022-0038.

E. W. Vollset and P. D. Ezhilchelvan. Design and performance-study of crash-tolerant protocols for broadcasting and reaching consensus in manets. In *SRDS '05: Proceedings of the 24th IEEE Symposium on Reliable Distributed Systems (SRDS'05)*, pages 166–178, Washington, DC, USA, 2005. IEEE Computer Society. ISBN 0-7695-2463-X.

R. Volz. *Web Ontology Reasoning with Logic Databases*. PhD thesis, Universität Karlsruhe (TH), Universität Karlsruhe (TH), Institut AIFB, D-76128 Karlsruhe, 2004.

D. Winer. Xml-rpc specification, 2001. URL `http://www.xmlrpc.com/spec#update3`.

R. D. Yates and D. J. Goodman. *Probability and Stochastic Processes: a friendly introduction for electrical and computer engineers*. Wiley, 1999. ISBN 0-471-17837-3 (cloth).

J. Yoon, M. Liu, and B. Noble. Random waypoint considered harmful. In *INFOCOM*, 2003.

W. Zhao, H. Schulzrinne, and E. Guttman. Mesh-enhanced service location protocol (mslp), 2003.

F. Zhu, M. W. Mutka, and L. M. Ni. Service discovery in pervasive computing environments. *IEEE Pervasive Computing*, 04(4):81–90, 2005. ISSN 1536-1268.