FFI-rapport 2011/00257

Pervasive Web Services Discovery and Invocation in Military Networks

Frank T. Johnsen

Norwegian Defence Research Establishment (FFI)

4. January 2011

FFI-rapport 2011/00257

1176

ISBN
P: 978-82-464-1880-3
E: 978-82-464-1881-0

Keywords

Nettverksbasert Forsvar

Tjenesteorientert arkitektur

Service discovery

Approved by

Anders Eggen	Project Manager
Eli Winjum	Director of Research
Vidar S. Andersen	Director

English summary

In this report we first investigate techniques that can be employed to optimize Web services: We suggest using content filtering to reduce application overhead, data compression to address the Web services overhead, and using specialized transport protocols that can overcome the hardware limitations and disruptive nature of military networks. However, some of the optimization techniques break the compatibility with services and clients implemented with today's development tools. Therefore, we suggest that proxies should be implemented. The optimization techniques can then be put in proxies, which can ensure that clients and services remain interoperable with the Web services standards.

Second, we investigate the Web services standards related to service discovery, and find that they can be used in many military networks, but that they are inadequate for use in disadvantaged grids. Thus, we survey existing service discovery solutions, and from their different implementations we learn about the various techniques that can be employed for achieving service discovery. We choose a set of techniques that are well suited for overcoming the challenges in disadvantaged grids: Decentralized operation to overcome availability issues, periodic service advertisements to ensure an up-to-date view of available services, caching to reduce resource use (no need to query the network), piggybacking to reduce the number of data packets, and compression to reduce the size of the data packets. We implement all these techniques in an experimental solution that we call Service Advertisements in MANETs (SAM). Then, knowing that we can perform service discovery in different networks, we investigate how to achieve service discovery across networks. We survey existing techniques for pervasive service discovery, and identify gateways as the simplest and most cost-effective way of achieving transparent service discovery protocol interoperability. We implement an experimental service discovery gateway, and use it in a military experiment featuring MANETs and fixed infrastructure networks. In this experiment we show that we can achieve transparent discovery between SAM and a Web services discovery standard.

The premises for this report were that *all techniques explored, conclusions drawn and decisions suggested in this report must be compatible with use in a federation of systems* and that *Web services should be employed not only for federating systems, but also as a middleware technology within the military networks as well.* This means that it is beyond the scope of this report to create a unified solution for all networks, unless this solution was based on standards. Given these premises, the report addresses three research claims: 1) To be able to invoke Web services in military networks with constraints, adapting the standards is needed. 2) Networks with different properties require different discovery mechanisms. 3) In a federation of systems there is a need for application level interoperability points for service discovery.

In summary, the contribution of this report is an evaluation of a set of techniques followed by proof-of-concept implementations that enable pervasive Web services discovery and invocation in dynamic, heterogeneous networks.

Sammendrag

Denne rapporten tar for seg to hoveddeler av Web services: Først undersøker vi teknikker som kan benyttes for å optimalisere bruken av tjenestene - filtrering, komprimering, og transportprotokoller for å nevne noe. Optimaliseringsteknikkene vil ødelegge kompatibiliteten mellom klienter og tjenester utviklet med dagens programmeringsverktøy. Vi foreslår derfor at optimaliseringsteknikkene bør implementeres i proxier, noe som gjør at klienter og tjenere kan beholde interoperabiliteten ved å fortsette å følge Web services-standardene.

Deretter undersøker vi metoder for å finne tjenester ("service discovery") i militære nettverk. I sivile systemer foretas service discovery ofte statisk i det systemet blir laget ("design-time discovery"). Det betyr at tjenesten man skal bruke blir valgt når klienten implementeres, slik at tjenestens adresse kan hardkodes i applikasjonen. Dette fungerer i statiske nettverk med høy oppetid, men i dynamiske omgivelser, slik som militære nettverk, så kan tjenester komme og gå. På grunn av dette trenger man å kunne oppdage nye tjenester mens systemet brukes ("run-time discovery"). I denne rapporten ser vi på Web services-standardene som kan benyttes til "service discovery", og påpeker hvilke typer nett de eventuelt kan benyttes i. Standardene er ikke tilstrekkelige for service discovery i disadvantaged grids. Vi ser derfor på en rekke eksisterende protokoller, og identifiserer et sett egenskaper som er ønskelige i en service discovery-protokoll for disadvantaged grids: Desentralisert protokoll for å sikre tilgjengelighet, periodiske oppdateringer for å gi en oppdatert tjenesteoversikt, mellomlagring for å redusere nettverksbruk, sammenslåing av data for å redusere antall datapakker, samt komprimering for å redusere størrelsen til pakkene. Alle disse teknikkene implementeres i en eksperimentell løsning som vi kaller "Service Advertisements in MANETs" (SAM). Videre ser vi på teknikker for å oppnå gjennomgående service discovery, og identifiserer en gateway som den enkleste og mest kosteffektive måten å oppnå transparent service discovery-interoperabilitet på. Vi implementerer videre en slik gateway, og benytter den i et eksperiment som omfatter både MANETs og nettverk med infrastruktur. I dette eksperimentet viser vi at vi kan oppnå transparent interoperabilitet mellom SAM og en av standardene for Web services discovery.

To premisser ligger til grunn for arbeidet i denne rapporten, nemlig at *alle foreslåtte teknikker og løsninger må være kompatible med en føderasjon av systemer* og at *Web services skal benyttes ikke bare mellom, men også innad i de ulike militære nettverkene*. Dette innebærer at rapporten ikke tar sikte på å finne en altomfattende løsning for alle nettverk, med mindre denne løsningen er basert på standarder. Rapporten er bygget opp rundt følgende tre forskningspåstander: 1) For å kunne benytte Web services i militære nettverk med begrensninger er det nødvendig å gjøre tilpasninger av standardene. 2) Nettverk med ulike egenskaper krever ulike mekanismer for "service discovery". 3) I en føderasjon av systemer er det et behov for interoperabilitetspunkter på applikasjonsnivå for "service discovery".

Rapportens bidrag er en evaluering av et sett teknikker som muliggjør gjennomgående bruk av Web services i og på tvers av dynamiske heterogene nettverk.

Contents

Preface

This report is based on Frank T. Johnsen's thesis "Pervasive Web Services Discovery and Invocation in Military Networks" (ISSN 1501-7710), which was written in partial fulfillment of the requirements for the degree of *philosophiae doctor* at the University of Oslo. The work described herein was performed as a part of the research activities on Web services in disadvantaged grids in FFI project 1086 "Secure Pervasive SOA". This work is being built upon in the current project, "Tjenesteorientering og semantisk interoperabilitet i INI" 1176, which is a continuation of the research activities in projects 1086 and 1085.

1 Introduction

Web services technology is becoming increasingly popular. Businesses use the technology not only as a middleware within their own corporate networks, but also across the Internet as a means of providing services to other corporations. Currently Web services are being employed with success by a wide array of companies ranging from banks to bookstores. The technology, being based on standards, makes it a simple and cost effective way to build loosely coupled systems.

Since Web services are so successful in civil applications, it makes sense to attempt to employ this technology for military applications as well. This is a challenge, since military networks are complex and encompass different heterogeneous networking technologies. In this report we focus on the use of Web services technology in military networks.

1.1 Background and motivation

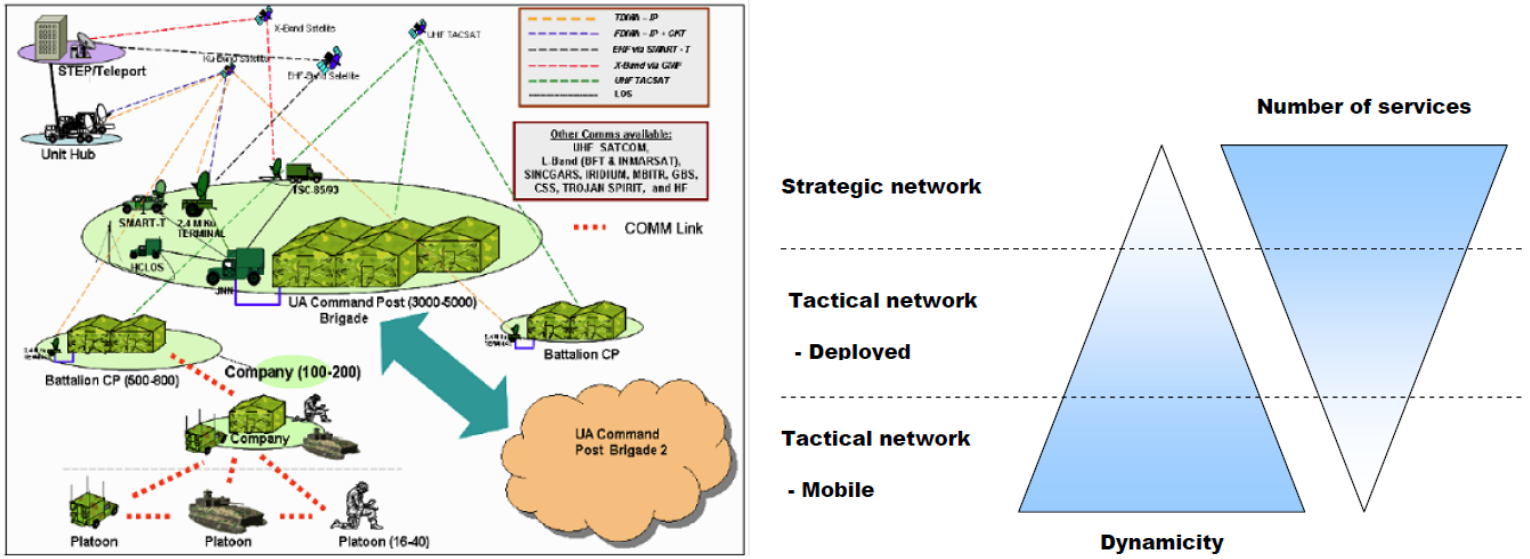
An operational military network is complex (see Figure 1.1(a)), consisting of different operational levels which use different communications technology and equipment. At the lowest level, there are a few highly mobile units. Moving up in the hierarchy there will be more units, but less mobility. The command posts are typically deployed tactical networks. Looking at the different operational levels in context (see Figure 1.1(b)) we can see that the characteristics vary from level to level. A strategic network has fixed infrastructure and is typically not dynamic.

A tactical deployed network is more dynamic than a strategic network. These networks have infrastructure, but typically depend on radio or satellite communication to connect to other deployed and mobile networks. A mobile tactical network (i.e., networked single units taking part in an operation) typically relies only on wireless communications and yields the highest dynamicity seen in an operational network. On the other hand, the total number of services available will be highest in strategic networks. The deployed tactical network will have a more specialized need for services, and thus most likely a lower number of services than on the strategic level.

A mobile tactical network will have and use the least number of services. These networks are often called *disadvantaged grids* since they are characterized by *low bandwidth, variable throughput, unreliable connectivity, and energy constraints imposed by the wireless communications grid that links the nodes* [51]. Not only does the limited bandwidth[1] at this level limit the possible amount of services and communication, but also the need for services will be location and mission specific.

In the Network Based Defense (NBD) concept there is an ambitious requirement for users at all operational levels to seamlessly exchange information. In order to achieve efficient information exchange between these users, one needs to work with different types of information and

[1]The term *data rate* is used when we discuss throughput in terms of bit/s (and multiples thereof). In the networking community, and in this report, the term *bandwidth* is used interchangeably with the terms data rate and throughput. This is in contrast to the signal processing field, where the bandwidth is usually discussed in terms of hertz.



(a) Operational network (from [109]).

(b) Domain complexity (from our report [67]).

Figure 1.1: Military networks are complex and heterogeneous.

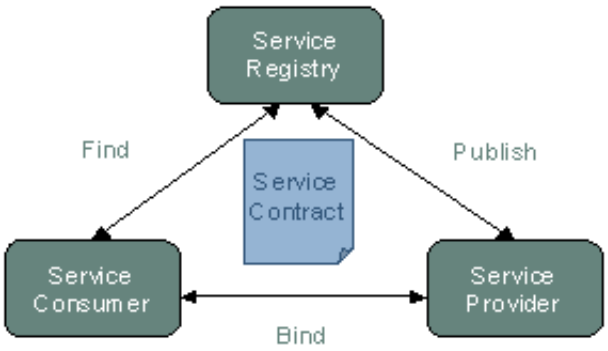


Figure 1.2: The service-oriented architecture (from our report [67]).

communication systems. Systems and equipment used at the various levels are different, and the information exchange must be adapted to fit the capacity of the systems used.

The NATO Network Enabled Capability (NNEC) Feasibility Study [10] presents a discussion of technology, focusing on the needs of future interoperable military communications. An information infrastructure will have to allow for communication across system and national boundaries while at the same time taking legacy systems into account. This leads to a requirement for a flexible, adaptable, and agile information infrastructure which can support all the information needs of national forces, and at the same time support interoperability. The study identifies the Service-Oriented Architecture (SOA) concept and Web services technology as the key enablers for NNEC.

1.1.1 Service-Oriented Computing

The Service-Oriented Computing [62] paradigm is based on the notion of a service, which is a networked piece of functionality (component) offered by a service provider. A service is specified by its service contract, interface, and semantics. The interface should be coarse-grained to accommodate internal change of the service implementation without affecting the interface. Building a system with a SOA [39] means to develop such stand-alone services and to compose them into a system.

In [98], SOA is defined as:

> SOA is a paradigm for organizing and utilizing distributed capabilities that may be under the control of different ownership domains. It provides a uniform means to offer, discover, interact with and use capabilities to produce desired effects consistent with measurable preconditions and expectations.

There are three roles in a SOA; *provider*, *consumer* and *registry*. Three operations define the interactions between these three roles; *publish*, *find* and *bind*, as shown in Figure 1.2.

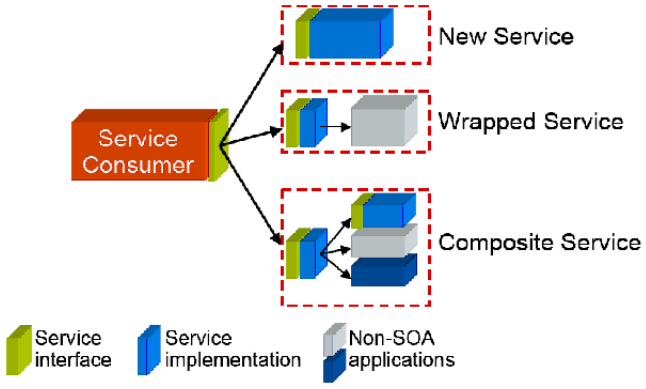


Figure 1.3: Creating services (from [88]).

Interoperability between provider, consumer, and registry is ensured by a standardized *service contract*. Following these principles, we get a loose coupling between the clients and the services, with respect to both time (enabling asynchronous communication) and place (the location of both client and service can be changed without need for reconfiguration):

1. A *service provider* is responsible for creating a service description, *publishing* that description in a *service registry*, and receiving and answering *bind* requests (i.e., service invocations) from service consumers.

2. A *service consumer* is responsible for finding a service description published in a *service registry* and using that description to *bind* to *service providers*. With the *find* operation the *service consumer* states search criteria such as the type of service it needs. The result of the find operation is a list of service descriptions that match the find criteria.

3. A *service registry* is responsible for advertising service descriptions *published* to it by *service providers* and allowing *service consumers* to search for (i.e., *find*) service descriptions within the service registry.

The central concept in a SOA is the *service*, which [98] defines as:

> A service is a mechanism to enable access to a set of one or more capabilities, where the access is provided using a prescribed interface and is exercised consistent with constraints and policies as specified by the service description. A service is provided by one entity — the service provider — for use by others, but the eventual consumers of the service may not be known to the service provider and may demonstrate uses of the service beyond the scope originally conceived by the provider.

Thus, the principle of separating the service interface from the service implementation means that there are several different ways of realizing services. As shown in Figure 1.3, a service can be

defined from scratch, allowing full control of the way the service is implemented. In addition, existing applications can be *wrapped*, and made available as services in a SOA. This approach requires an adaptation layer in order to adapt the existing interface of the application to the service interface. Finally, services of both types can be combined, in order to create new functionality in a more complex, *composite service*.

The SOA paradigm is not very new, it is a widely recognized approach to building loosely coupled distributed systems. In principle, it can be implemented using almost any middleware[2] product. However, when keeping the interoperability requirements in mind, it is preferable to build a SOA using open standards. Even if a SOA can be implemented using several different technologies, Web services stand out as a preferred and widely adopted standard [38].

1.1.2 Web services

There are many definitions of "Web services". The core idea is the same (i.e., using XML [140] formatted data for information exchange), but some of the finer details may vary. For example, so-called *restful* Web services [113] (REST) ignore most of the Web services standards and specifications, providing only synchronous remote procedure call functionality using HTTP [40] over TCP [108]. TCP does not necessarily function in all disadvantaged grids, meaning that REST is too restrictive if one wants to implement a pervasive SOA for military networks. We need the flexibility of a broader spectrum of the Web services specifications.

Definition

In total there are well over 150 specifications related to Web services. However, only a few of these have become standards, or specifications mature enough to be widely supported by industry. Thus, only a select few of all these specifications are actually in use today. The most important standards are SOAP (messaging) and WSDL[3] (service descriptions), since they form the foundation for all other Web services specifications and interactions [27].

When we discuss *Web services* in this report we use the definition by the World Wide Web Consortium (W3C) [142]:

> A Web service is a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format (specifically WSDL). Other systems interact with the Web service in

[2] Middleware is an abstraction layer that can conceal the heterogeneity of applications, operating systems, communication systems and hardware in a distributed system by providing a common interface to applications. In other words, *middleware is a software layer between application and operating system* [8].

[3] The Web Services Description Language (WSDL) provides service descriptions using XML. The abbreviation "WSDL" usually refers to the language, whereas a "WSDL document" refers to a specific XML file adhering to the language, thus providing an interface to a specific Web service. In this report we use the terms interchangeably when talking about service instances.

a manner prescribed by its description using SOAP-messages, typically conveyed using HTTP with an XML serialization in conjunction with other Web-related standards.

Lifecycle

Figure 1.4 shows the lifecycle of Web services. We will now discuss this lifecycle in context of the SOA principles from Figure 1.2:

1. Assuming that a service has been implemented by a service provider, it first needs to make the service available through an *advertisement*. This means that after the service has been made available in the network, it needs to be published in a service registry.
2. Service clients, the so-called consumers, can now query the registry and find this and other available services (i.e., service *discovery*).
3. The list of services found in the discovery phase then need to go through a *selection* phase, where a service is selected. Service selection can be done either manually or automatically [45].
4. If multiple interesting services are found, then one can also create a new service through *composition*.
5. The final step is *invocation*, where the service consumer binds to the service provided.

Only the last step has to be performed at *run-time*. When Web services are used in businesses, the first four steps can be performed at *design-time*, when the SOA enabled system is implemented. This is because enterprises usually have fixed infrastructure where services are permanently available, so that the service address (i.e., binding) that was discovered and selected can be hard-coded in the client software. However, in dynamic systems, there is a need to be able to perform some of these steps at run-time. In this report we pay special attention to steps 1, 2, and 5. The importance of step 5 is obvious, but since tactical networks are dynamic we also need to be able to perform steps 1 and 2 at run-time. Automated run-time selection and composition of Web services is not supported by current Web services standards, and is thus beyond the scope of this report.

1.2 Problem statement

Current Web services solutions are designed for Internet-type networks where bandwidth is abundant and nodes are stationary. Applying such technology directly for military purposes may not be feasible, especially when considering the disadvantaged grids where resources are scarce.

Web services are based on exchanging text based messages over Internet protocols. This means that the existing solutions still have a number of limitations, in the sense that they do not have sufficient

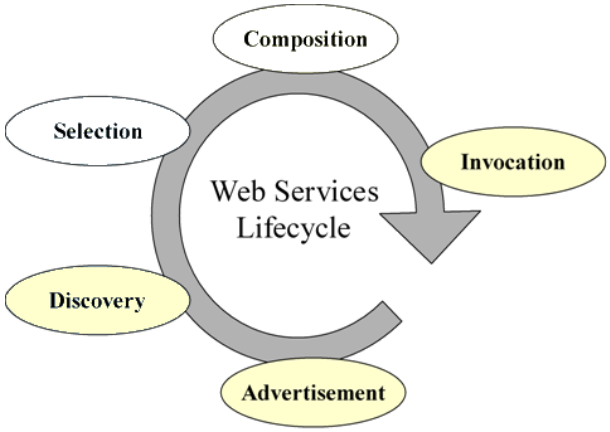


Figure 1.4: The lifecycle of Web services (adapted from [20]).

adaptation capabilities for such dynamic environments as military tactical networks represent, where both resource and service availability may vary continuously.

In a highly dynamic environment, being able to locate and invoke Web services becomes a major challenge. The process of identifying a service, known as service *discovery*,[4] is an important part of any SOA, but it is particularly challenging in dynamic environments such as military tactical networks. A service discovery architecture for such an environment should enable discovery and selection of relevant services, and offer a complete and up-to-date picture of the services available at the given point in time. Moreover, it should be robust in terms of partial failure as well as bandwidth efficient, since nodes in dynamic environments may have bandwidth-constrained wireless connections.

Previously, FFI has performed experiments with Web services in a multi-national scenario at the NATO Coalition Warrior Interoperability Demonstration (CWID) in 2006. These experiments showed that Web services could be used to exchange position tracking data between nations. The object-oriented XML-version of the Command and Control Information Exchange Data Model (C2IEDM) from the Multilateral Interoperability Programme (MIP) was used to exchange messages. Those experiments showed that the utilization of Web services in NBD is feasible for some applications, but it also revealed several challenges [55]. In those experiments Web services were used at the *strategic level*, where bandwidth is abundant (but even so, the Web services traffic consumed a lot of the available bandwidth). In order to achieve full-fledged NBD, the needs of *tactical* network users must be considered as well.

If Web services technology is to be used pervasively in military networks, it must be possible to leverage this technology in some way or another in (and across) all of the different operational levels. That Web services can be used within networks at the strategic level is apparent, since the networking technology used there is no different from that being used in civil enterprises, for which

[4]Discovery is the act of locating a machine-processable description of a Web service-related resource that may have been previously unknown and that meets certain functional criteria [143].

Web services were designed in the first place. The main challenge lies in applying Web services technology in and across tactical networks. Thus, in this report we focus mainly on Web services in the tactical networks.

1.3 Research methodology

The results in this report have been derived from an iterative and incremental process. The main research activities performed as part of this process can be summarized as follows:

1. *State of the art and requirements analysis:* The report addresses three different but related problems (i.e., 1) invoking Web services, discovery of Web services 2) within and 3) across networks). The state of the art is discussed in the context of each of these problems in Chapters 4, 5, and 6. The requirements analysis is summarized in Section 3.5.

2. *Hypothesis:* A set of research claims were formulated based on the motivation and research area identified in Sections 1.1 and 1.2. These claims are presented in Section 1.5.

3. *Specification:* The specification of optimization techniques (for service invocation) as well as the specification for intra and inter network service discovery were derived based on identified requirements and challenges within the problem domains.

4. *Design and implementation:* The solutions in this report were designed and implemented in an incremental and iterative fashion. The evolution of the framework is manifested through a series of publications upon which this report is based. The relevant references are given throughout the report.

5. *Testing and proof of concept:* The design and implementations have been tested and experiments performed mainly through a set of proof-of-concept case studies. Further details are provided in Section 1.6.

1.4 Intentional limitations

It is possible to implement a SOA using other middleware than Web services. However, since NATO is focusing mainly on Web services, we do not consider other middleware in this report. Also, since NATO's key concern is interoperability, we focus only on the most mature of the Web services standards and specifications (i.e., the Web services technologies which are currently implemented and in use in civil systems) as a basis for our research.

The Web services' lifecycle (see Figure 1.4) does not address issues related to *automatic* selection and composition. Automating those aspects are beyond the Web services specifications and standards, and thus beyond the scope of this report. Such issues are in the domain of *semantic* Web services, which we leave for future work.

We only concern ourselves with the core technical aspects of using Web services as a middleware. It is beyond the scope of this report to design or implement a complete SOA architecture for military networks. Thus, issues like governance and service management are not addressed.

Security is very important in military networks. However, today's strict security policies and doctrine require that you encrypt messages at the link or IP layer to secure your communications when information of higher classification is transferred. The application level security mechanisms that can be applied to Web services are not currently approved for securing military communications. Therefore, we do not focus on the security aspects of Web services in this report. It should be noted, however, that none of the solutions discussed here are incompatible with standardized Web services security mechanisms, so if one in the future would like to add such functionality to the middleware as well, then that should definitively be possible.

Optimizations of different parts of the communications stack is important in military networks. However, we focus on using existing radio and networking technologies to carry Web services. We do not address any link layer issues, since this is covered by other projects at FFI. We focus only on *IP networks* [107] and radios capable of transmitting IP packets. It is beyond the scope of this report to create new transport protocols, we build on existing protocols and solutions. Our research is limited to those parts of the systems that are affected by Web services, i.e., the messages exchanged by the Web services middleware, the way these are transported across the network, and also how the applications may employ Web services.

Standardization is important for the interoperability requirement; it can keep costs down since one can avoid vendor lock-in and there is no need to maintain proprietary solutions. Therefore, we want to use standard solutions when possible, and investigate experimental solutions only when the available standards are inadequate.

1.5 Contribution and claims

The contribution of this report is a collection of techniques that allows us to perform pervasive Web services discovery and invocation in military networks. Standard Web services are intended for use over the Internet or in corporate local area networks (LANs), but by employing our techniques it is possible to use Web services technology also in highly dynamic environments with severe data rate limitations. By doing this we show that Web services can be used in and across all operational levels, thus showing the feasibility of employing Web services (with our optimizations) as a middleware in military networks.

By combining existing techniques such as store-and-forward, content filtering, data compression, efficient transport protocols and the concept of proxy servers, we can limit the overhead of Web services enough so that the technology can be used in networks with low data rate and frequent disruptions. Using these techniques enable us to invoke Web services in heterogeneous networks. However, in dynamic networks we also need dynamic service discovery.

We address the issue of dynamic service discovery by identifying a toolkit of different techniques and mechanisms for achieving service discovery in different networks. Different mechanisms have different constraints, and put varying loads on the network. Thus, we argue that one should employ the most efficient mechanism for each network. By doing this we can achieve service discovery within both static and highly dynamic networks by choosing a suitable mechanism for each such network. We can use mechanisms such as registries in static networks, and specially tailored mechanisms for the mobile tactical networks. However, interoperability between the service discovery mechanisms is also needed. We suggest using gateways between the networks for interconnecting the different service discovery protocols. Gateways have the benefit of adding low complexity to the system (i.e., needing deployment only in the connection point between the networks). Low complexity is good in the long run, since it means low development and maintenance costs.

The premises for this report are given below, along with three research claims. These claims express the core of our work, and we prove these claims through the report.

Premises

NATO has identified Web services as a key enabling technology for achieving application interoperability when interconnecting heterogeneous systems. Norway, being a NATO member, acknowledges this and adheres to NATO requirements for interconnecting systems. Since Web services technology is based on standards, interoperability can be ensured between systems from different nations and different vendors. However, NATO only requires nations to use Web services for interoperability — the different nations can employ other technologies for use within their own networks if they so choose. This means that a NATO coalition force will consist of a *federation of systems* rather than a *system of systems*. In a federation of systems, each network has its own administrative domain and each network is under the full control of the owner. An important premise for our work is that *all techniques explored, conclusions drawn and decisions suggested in this report must be compatible with use in a federation of systems.*

Since Web services are based on standards, it is possible to keep development costs low and at the same time avoid proprietary solutions and vendor lock-in. In addition to low development and maintenance costs we get such benefits as being able to use our existing applications and infrastructure in completely new ways, which can lead to more efficient information dissemination and thus more efficient execution of military operations. Thus, our second premise is that *Web services should be employed not only for federating systems, but also as a middleware technology within the Norwegian military networks as well.* This means that it is beyond the scope of this report to create a unified solution for all networks, unless this solution is based on standards.

Keeping these premises in mind, we can discuss the research claims next.

Claim #1

To be able to invoke Web services in military networks with constraints, adapting the standards is needed.

Civil use of Web services has proven that the technology is well suited for use over the Internet and in corporate networks. Thus, that Web services can be used in military strategic networks is obvious, since they use the same networking technology that is used to build civil fixed infrastructure networks. Challenges arise when you attempt to use the technology in dynamic environments with severe limitations on throughput, with high error rates, and frequent disruptions. These issues can arise if one wants to use Web services in civil Mobile Ad Hoc Networks (MANETs), but in military MANETs there are additional constraints as well, such as severely limited packet buffers, little or no support for IP fragments, and maximum transfer units that are much smaller than those available in common civil radios. As a consequence, we think it is necessary to adapt certain aspects of Web services' behavior to accommodate the characteristics of such networks.

This claim is addressed in Chapter 4. To prove this claim we first identify the main characteristics of military networks and discuss aspects of current Web services technology in that context. Comparing current Web services' behavior with the requirements posed by military application, we are able to identify weak points in the technology, and argue that it cannot be employed in networks with constraints without adaptation. Then, to show that adaptation is needed, we evaluate different techniques that can be applied to Web services which enable the technology's use also in military networks with constraints.

Claim #2

Networks with different properties require different discovery mechanisms.

There are many different service discovery protocols, where the different protocols are optimized for different types of networks. At the strategic level we need mechanisms that can scale to a large number of services and a large number of users, but where the underlying network is fixed. At the tactical level we have higher demands for resilience than in the strategic level, due to these networks being deployed in or near the network-centric battlefield. The tactical mobile networks are single or multi-hop MANETs, which warrant other service discovery protocols than fixed networks because of their highly dynamic nature (e.g., spontaneous network partitioning and connectivity loss). In this report we investigate existing solutions, and classify them according to their suitability for use at the different military operational levels. Since there are so many different networking technologies available for use in tactical networks, one solution cannot accommodate all needs. We need a toolkit consisting of several different mechanisms, so that the mechanism best suited for each network can be employed there.

This claim is addressed in Chapter 5. To prove this claim we first identify a set of requirements that we compare the Web services service discovery standards' features with. The main challenge here is discovery in disadvantaged grids. Following a theoretical evaluation, we identify where the current solutions are lacking necessary properties, and attempt to identify mechanisms that can solve service discovery within such networks. Finally, we design and implement an experimental solution that we call *SAM* (short for *Service Advertisements in MANETs*) that is suitable for use with some disadvantaged grids that we currently have access to. Our protocol is evaluated and compared to its standardized Web services counterpart. Finally, the protocol is implemented and shown to function in an actual operational experiment.

Claim #3

In a federation of systems there is a need for application level interoperability points for service discovery.

Assuming claim #2 holds, we can discover services in any network as long as we choose to use a discovery protocol that is suitable for that particular network. However, one needs to discover and invoke services across heterogeneous network boundaries in military operations. In a *joint operation* you need to interconnect systems from several military systems belonging to the army, navy, and air force. In a *combined operation* you need to interconnect your national systems with the systems of the other nations in the coalition force. This means that one needs to discover Web services in an interoperable way across heterogeneous network boundaries, both across operational levels and across national boundaries.

This claim is addressed in Chapter 6. To prove this claim we first investigate different means of achieving pervasive service discovery. We develop models based on related work illustrating the different approaches. The models are evaluated in the context of overcoming network heterogeneity, and the premise that the chosen solution must function in a federation of systems. We show that the only technique which is suited for use in a military context is that of a service discovery *gateway*, because it does not require global changes in the federation of systems. In a federation each nation has ownership of their own system, meaning that we cannot deploy a solution across systems, only inside our own. By employing service discovery gateways between heterogeneous networks we can achieve service discovery interoperability in a transparent way. Also, we can let each network continue to use its service discovery mechanism of choice, while still supporting pervasive service discovery. Finally, a proof-of-concept implementation is provided and evaluated, first in a lab setting, and then in an operational experiment.

1.6 Practical application

The work and techniques described in this report have been tested in three experiments:

- NATO CWID 2007: The NATO Coalition Warrior Interoperability Demonstration (CWID) is an annual event targeted at improving interoperability between command, control, communications, and computer (C4) systems of NATO, NATO nations and partner nations. Its main objective is ensuring interoperability of systems to be deployed in NATO Response Force and Combined Joint Task Force. In addition, the need to perform experimentation with new technology in order to better support future requirements is recognized.

- Multinett II 2008: The national military exercise "Multinett II" (MNII) at Ørlandet and Jåttå was the largest joint experiment activity ever carried out in Norway, and consisted of a number of field experiments. The basic concept of MNII was to connect different communication systems from all the military services into one common network. This included interconnections tried earlier, as well as some interconnections never tried before. The goal of MNII was to experiment with technology that can contribute to further development of NBD.

- Combined Endeavor 2009: The Combined Endeavor (CE) exercise has grown to be one of the main vehicles to demonstrate and propagate new interoperability concepts. CE is a series of US European Command-sponsored workshops planned and executed to identify and document C4 interoperability between NATO and Partnership for Peace nations.

1.7 Outline

The remainder of this report is organized as follows:

There are many challenges related to communications in military networks. These challenges are discussed in Chapter 2. Chapter 3 discusses relevant Web services standards and specifications that can be used to implement Web services as a middleware. We discuss some examples that show how such middleware can be used in enterprises and in military systems. Assuming a SOA built in design-time using Web services, which measures do you need to use to be able to invoke your services in military tactical networks? We address the *bind* operation (invocation of known Web services) in Chapter 4. In dynamic environments, services are transient (i.e., they can come and go). Thus, we need run-time discovery of services in such environments. We discuss service advertisements and discovery (the *publish* and *find* operations) in the context of military networks in Chapter 5. Pervasive service discovery is investigated in Chapter 6. Chapter 7 concludes the report and outlines future work.

2 Military tactical networks

We are concerned with Web services and their application in military communication systems. The goal of this chapter is to give an overview of the communications technologies that are available on various levels in the military organization. We focus mainly on tactical communications in this report. Communication on the tactical level is characterized by wireless networks with low bandwidth, and possibly high delay and high error rates and frequent disconnections. As a consequence, these networks are often called disadvantaged grids.

Different tactical communications equipment from different vendors cannot necessarily communicate with each other. Support for different frequencies, cryptographic modules, and other differences mean that the equipment used by different NATO nations is incompatible. NATO is currently working on standardizing waveforms that can be used for interoperability. However, since military equipment is often acquired with an intended lifespan of 20 years, there will still be many years till complete interoperability at all OSI layers between all NATO nations can become a reality. NATO's first step towards interoperability is by defining so-called *interoperability points*, i.e., gateways that can be used to interconnect heterogeneous networks. NATO has also identified the Internet Protocol (IP) as the common network protocol to be used in all coalition networks [10]. A lot of legacy equipment does not support IP, and such systems either have to be equipped with IP-capable modems, or IP needs to be tunneled through the protocol(s) they support (typically X.25 for the previous generation equipment). Currently available tactical hardware typically supports IPv4, but often with limitations. However, due to the long lifespan of procured equipment, a lot of legacy systems will be around for many years to come. In this way, military networks differ significantly from civil ones, in that equipment has a much longer operational life. Also, in civil systems you will want as much uptime as possible to maximize availability and revenue. In military operations, sometimes you need to sacrifice connectivity in order to avoid being detected by the enemy. By going into radio silence, the unit can still receive transmissions, but will not send any responses to avoid electronic detection. This means that protocols which function well on the Internet, such as TCP, will not function at all in a military network if you attempt to use it when a unit is in radio silence: The unit may receive all the packets, but it will not be allowed to send an acknowledgment back, thus triggering TCP's retransmission function. Other examples where TCP cannot be used is across communication diodes (one-way data pumps), and over radios with high turntime. For such cases, specialized tactical protocols can be used, which are designed to function under such conditions.

2.1 Characteristics of tactical radio networks

Several challenges are posed by the tactical communications environment, as identified by [50] and [126]. We summarize the most important characteristics below.

2.1.1 Network topology

The network topology in the disadvantaged grids in the tactical battlefield is unpredictable, since the communications links provided by the broadcast medium (radio) are unreliable. The nodes that are participating entities in the network are highly mobile. There are networks of sub-networks, where each sub-network is on a different base frequency from the others. The nodes can frequently connect/disconnect from sub-networks when needed. A single radio can participate on only one sub-network at a time. Today, participation in multiple networks requires multiple radios, but the number and type of radios that are feasible to bring will be determined by space and weight limitations. For example, a vehicle can carry multiple heavy radios as opposed to a soldier, who may carry one or maybe two comparably light devices.

2.1.2 Connectivity loss

Connectivity loss is an issue. There are two types of connectivity loss:

1. *Planned loss of connectivity*: The node can be re-assigned to a different role in the battlefield, in which case it can disconnect from the network it is currently connected to. Also, a node that needs to participate on multiple networks with a single radio by switching frequencies will have to leave one network in order to connect to another. Another reason can be that the node has to go into radio silence to avoid enemy detection. In this case the node cannot send any information, but it can still receive. If the node is subscribing to some sort of periodic information, i.e., it is only an information sink, then radio silence is not necessarily a problem. For example, the node can still receive orders to perform a certain task, but it cannot acknowledge that the order has been received (at least not right away, it can communicate freely again when radio silence is no longer necessary).
2. *Unplanned loss of connectivity* can be caused for a number of reasons: There can be terrain or atmospheric interference, nodes can move too far apart and exceed radio range, it can be caused by enemy actions (jamming or attack), or it can be caused by equipment malfunction.

In the case of planned connectivity loss, the application can be prepared for this event: You can de-register your published services, and you can finish using the services you need before you disconnect. If you need to continue receiving information after going into radio silence, which is a form of planned connectivity loss, you can signal this up front — if you are using a tactical transport protocol, then it will no longer expect acknowledgments. By setting up the subscriptions you need before entering radio silence, you can continue receiving data and the node can still perform a lot of tasks utilizing the information it receives.

If you are the victim of unplanned loss of connectivity, then that can be problematic for the operation. You will be disconnected from the service(s) you use, and preferably you will want to see if there are other relevant services nearby that you can connect to instead, if possible —

depending on the cause of the connectivity loss. Thus, in dynamic environments you need dynamic service discovery — a way to find the services that are available right now in your area. Also, if the connectivity loss is temporary, you would want your initiated request to go through. If there are temporary fluctuations in connectivity, then a store-and-forward system can be beneficial. We discuss these techniques in the context of Web services in subsequent chapters.

2.1.3 HTTP and TCP in disadvantaged grids

The use of HTTP over TCP originates from the World Wide Web (WWW), and has later been adopted as the primary protocol combination for Web services (see Chapter 3). The SOAP messages exchanged between Web services clients and servers are usually sent using HTTP, which in turn utilizes TCP for reliable transfer of the messages.

HTTP is synchronous, which means that when a SOAP request is sent, the HTTP connection is kept open until the SOAP response is returned in the HTTP "acknowledgment". If the connection times out because of delays or for any other reason, there will be a problem routing the SOAP response back to the service consumer. In disadvantaged grids, the data rate is often less than 1000 bit/s [126]. Our tests have shown that Web services carried over HTTP using the TCP implementation in Windows XP stop functioning when the data rate is 400 bit/s or less [124]. In certain disadvantaged grids, you will frequently be limited to a very low data rate per client, since the network is shared between several units. In practice, 40 bit/s per user is not uncommon, meaning that traditional TCP is non-functional in most tactical wireless environments [50]. Thus, limited data rate can be a problem in disadvantaged grids. In addition to the limited data rates, there are also issues with limited buffer space, limited support for IP fragments, and other equipment-specific limitations that vary from radio to radio and vendor to vendor.

Consequently, HTTP will not work well when used in disadvantaged grids or in a combination of heterogeneous networks. Furthermore, in disruptive networks TCP connections break, making the protocol less suited for the tactical environment. Such networks require asynchronous communications and protocols that are able to cope with the characteristics (e.g., data rates, delays, frequency of disconnections) of military communication networks:

- Protocols that can withstand long and variable round trip times, while at the same time having very little communication overhead.
- Store-and-forward capabilities, where intermediate nodes can store a message until it can be delivered to the recipient rather than discarding the message if immediate delivery is not possible.

The store-and-forward capability is needed for two reasons: Users connected through a disadvantaged grid can experience frequent but short communication disruptions, which can prevent a message from being delivered immediately. Having store-and-forward support can ensure that the message

is not dropped. In addition, store-and-forward can be used in gateways between network types to compensate for differences in link capacity between the networks. An ordinary router risks having to drop packets due to its buffers filling up faster than the packets can be transmitted out onto the lower capacity network.

2.2 Requirements for Network Based Defense

Ideally, we would want interoperable IP-based radios to facilitate communication in NBD. In the future this may become a reality, but currently many legacy systems need to be interconnected. To start implementing the NBD concepts today, we need to start by using interoperability points and overcoming network heterogeneity that way. Legacy systems must be made available as services that can be used in and across networks. It should be noted that services do not need to be globally accessible. Some services may only be of use at the operational level where they are deployed. Other services may need to be invoked across network boundaries, but the services themselves do not usually pass from network to network. There is one case where this may happen, though, and that is in the case we discussed above where a node may participate in several networks with one radio, but only one network at a time (i.e., it switches frequency to move between networks). In the case of radio silence, the node should preferably be able to use some sort of asynchronous publish/subscribe system to be able to continue to receive data asynchronously.

Due to the diversity of military networks, there will be challenges associated with invoking services across the heterogeneous networks. At the tactical level there is dynamicity in the networks, and one needs to discover if new services appear or if existing services become unavailable.

2.3 Summary

We have seen that military communications equipment is heterogeneous and diverse. Tactical networks are characterized by low data rates, variable throughput, unreliable connectivity, and energy constraints imposed by the communications equipment used. This makes communication over tactical networks, disadvantaged grids in particular, a challenge. Standard Internet protocols like HTTP and TCP do not necessarily function in all such networks, and we need to consider alternatives that can cope with the tactical equipment we have to work with. Due to the diversity of military networks, it would be beneficial if one could employ a standard based middleware such as Web services for interoperability between networks, and to implement clients and services. NATO has defined interoperability points, i.e., gateways, for hardware interoperability, and it has chosen IP as the common protocol. This paves the way for communication across heterogeneous military networks, and thus potentially employing a middleware technology to hide the heterogeneity of these networks. In the next chapter we discuss Web services as a middleware.

3 Web services as a middleware

Interoperability, both inter- and intra-nation, is a main concern when attempting to fully realize NBD. The NBD vision implies an information infrastructure that supports prioritized access to information, services, and communications resources from the strategic level, down to the tactical level where communication resources usually are scarce. This encompasses a vast array of different systems, and without interoperability between these heterogeneous systems the NBD vision will be very hard to accomplish.

In this chapter, the middleware aspect of Web services is explored. Web services is the technology identified by NATO as the *key enabler* for NNEC, which is the motivation for investigating this technology for use in NBD as well. However, other middleware technologies exist which potentially could also be used for implementing a SOA. This discussion is beyond the scope of this report, where we consider only Web services. For a comparison of Web services technology with other existing middleware solutions, both commercial and experimental, see our report [81].

3.1 Connecting heterogeneous distributed systems

One well known way of handling heterogeneity and distribution is through the use of middleware. In addition to providing a standardized interface, middleware can offer distribution transparency, i.e., hide the consequences of distribution, with respect to things like location, access, concurrency, replication, errors, and mobility. One common programming abstraction is offered, spanning a distributed system. The middleware encapsulates general solutions to tasks that appear repeatedly in distributed systems, offering building blocks on a higher level than the application programming interfaces (APIs) and sockets offered by the operating system.

Although middleware is usually viewed as a class of software technologies designed to help managing the complexity and heterogeneity found in distributed systems, there is no general consensus on the precise separation between these areas, nor on the required functions and services. In general, the separation will vary with time, as common application functionality is better understood, so that it can be standardized and moved into the middleware as a service.

Although Web services can be seen as yet another middleware, there are differences. One difference is that while other types of middleware are typically used within a local domain, Web services are used as middleware to connect such local domains over the Internet. In other words, they function as entry points into local information systems. Furthermore, Web services can be viewed as an attempt to standardize middleware platforms with respect to language (XML), interfaces (WSDL) and business protocols [3].

Traditional middleware face several problems that Web services have the potential to solve. For instance, for cross-organizational interactions, the problem is where to place the middleware and

who should control it [3]. There are also problems connected with long-lasting transactions and crossing of trust domains.

The concept of Web services takes the middleware abstraction even further, and can be viewed as a "middleware of middlewares". The idea is that while traditional middleware is used within a system or platform, for instance a LAN within a company, Web services are used to connect such systems. In other words, Web services are intended for use between systems, hence the "middleware of middlewares" analogy. Note, however, that this does not preclude the use of Web services also within a system. Even if some core standards are mature, other important topics are at the draft level, or are covered by competing non-interoperable standards. Thus, Web services are still far from being mature enough for playing the role as a full-blown "middleware of middlewares".

3.2 Web services specifications

One of the earliest Web services standards, SOAP, was first introduced in 1999. Since then the number of Web services related standards have been ever increasing and the Web services standards now cover a large range of topics. The core standards, such as XML, SOAP and WSDL are widely supported, but the sheer number of available specifications means that it is difficult for developers to know which of them to adopt. This task is made even more complex when taking into consideration the fact that the maturity of the specifications vary. Some are fully ratified standards and have been released in several versions already, while others are early in their development cycle and are currently working drafts or have status as notes or recommendations.

In addition to the maturity issue, it is worth noting that there is not one single organization that controls all the ongoing Web services standardization work, and thus there is no set standardization process that ensures that all the standards adhere to a common "Web services architecture". As a consequence, some topics are covered by multiple, and in some cases competing standards, while other topics are not covered by a standard at all. Vendor support is crucial, so looking into which standards are currently supported by the major vendors such as IBM and Microsoft can function as a guideline when trying to determine if a standard is likely to ever see widespread use. The Web Services Interoperability Organization (WS-I) [144] is an open industry organization chartered to promote Web services interoperability across platforms, operating systems and programming languages. The organization's diverse community of Web services leaders helps customers to develop interoperable Web services by providing guidance, recommended practices and supporting resources.

This chapter does not attempt to cover every aspect of the Web services standardization due to the ever changing nature of this work. Instead it focuses on the main categories and topics covered by current Web services standards, and point out the core standards within each of these categories. These standards are the ones that are currently most likely to be a part of a SOA deployment, and form a fundament for using Web services as a SOA middleware. Figure 3.1 gives an overview of

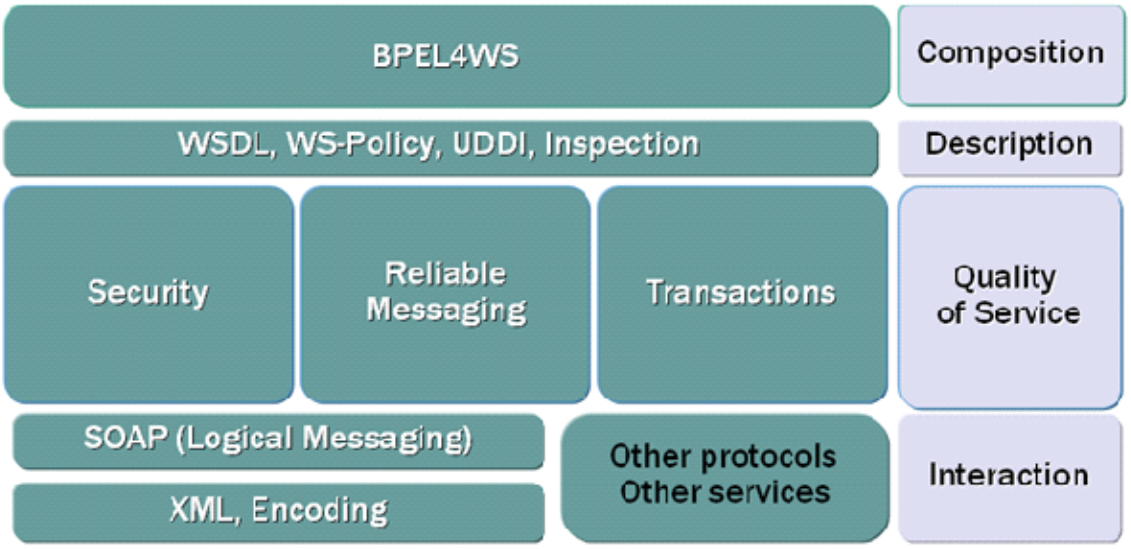


Figure 3.1: Web services middleware components (from [120]).

the core categories of Web services standards, but does not cover all topics. It does, however, serve as a good starting point for a more detailed categorization and description of standards.

3.2.1 Interaction

Extensible Markup Language (XML)

XML is a simple, very flexible text format derived from SGML (ISO 8879) [140]. Originally it was designed to meet the challenges of large scale electronic publishing, but XML is playing an increasingly important role in the exchange of a wide variety of data on the Web and elsewhere. XML is often considered the base standard for Web services, as most of the other standards use the encoding and format rules defined in this standard. There are multiple XML related standards, with the two most important being XML itself, and XML Schema. The latter standard is a description of a type of XML document, typically expressed in terms of constraints on the structure and content of documents of that type, above and beyond the basic syntax constraints imposed by XML itself.

An XML document consists of data that are surrounded by *tags*. Tags describe the data they enclose. A tag may have other tags inside it, which allows for a nested structure. To illustrate the structure of an XML document, consider this simple example:

```
<?xml version="1.0" ?>
<greeting>
   <greeting text>Hello world!</greeting text>
</greeting>
```

One of the benefits of using XML is that an XML document contains metadata, that is, data about the data that are present in the document. In the example above, for instance, we can see that the text string "Hello world!" is a "greeting text", which in turn is a part of "greeting". Such tags can be standardized, which allows for the exchange and understanding of data in a standardized, machine-readable way. An XML document can be defined according to an XML Schema, which enables validation of XML documents according to rules defined in the schema.

SOAP

SOAP [132] is an XML based protocol for information exchange in a decentralized, distributed system. SOAP is an envelope for XML messages, functioning as a transport independent messaging protocol. It was called "simple object access protocol" up to and including its release as a W3C version 1.1 note, but this name did not describe exactly what SOAP was, and so it was later dropped. In its current version, the W3C version 1.2 recommendation, the protocol is just called "SOAP". SOAP messages can be carried by a variety of protocols, the most common being HTTP. Other protocols can also be used — standardized SOAP bindings exist for UDP [100] and SMTP [26] in addition to HTTP. Since SOAP is transport protocol agnostic, it can basically be used with any transport protocol. For example, we have shown that one can run SOAP over STANAG 4406 [66]. Current Web services development tools usually support both SOAP version 1.1 and version 1.2.

A SOAP message contains a header and a body. SOAP is an application level protocol. Compared to for example an IP packet, the SOAP header is the equivalent of the IP header, whereas the SOAP body is the equivalent of the packet payload. Thus, just like an IP header, the SOAP header can contain information that is necessary to achieve the desired per-hop behavior of the message. For example, WS-Addressing [136] information can be present in the SOAP header and be used to achieve transport protocol independent addressing of SOAP messages. Security related standards typically also add fields to the SOAP header [91].

Other protocols

The messaging protocol used for Web services is SOAP, but there exist a large number of specifications that expand the SOAP protocol and add further functionality. An example of one such protocol is the Message Transmission Optimization Mechanism (MTOM) [141]. MTOM describes how to transfer binary data as a part of a SOAP message.

Additionally, the standards that support event notification are of particular interest in a military context. Publish/subscribe is a well known communication pattern for event driven, asynchronous communication. In [58] we discuss the benefits of employing the publish/subscribe paradigm in NBD.

At present there are two standardization efforts regarding publish/subscribe for Web services: OASIS has its Web Services Notification (WS-Notification, WSN) standard [93], whereas W3C has

produced a similar framework called Web Services Eventing (WS-Eventing) [135]. Both of these protocols are based on SOAP, and use the functionality provided by SOAP rather than building their own messaging protocols.

WS-Notification

WS-Notification is a publish/subscribe notification framework for Web services. There are three parts in the specification: WS-BaseNotification, WS-BrokeredNotification and WS-Topics. The WS-BaseNotification specification unifies the principles and concepts of SOA with those of event based programming.

WS-BaseNotification provides the foundation for the WSN family of specifications. It defines the basic roles and message exchanges needed to express the notification pattern. The specification can be used on its own, or it can be used in combination with the WS-Topics and WS-BrokeredNotification specifications in more sophisticated scenarios. The specification defines the message exchanges between notification producer, notification consumer, subscriber, and subscription manager.

The simplest form of a subscribe request message just contains an endpoint reference for a notification consumer. This form of request instructs the notification producer to send each and every notification that it produces to the notification consumer.

The subscribe request message can optionally contain one or more filter expressions. The filter expressions indicate the kind of notification that the consumer requires by restricting the kinds of notification that are to be sent for this subscription.

WS-Notification encompasses the following standards:

- WS-BaseNotification defines standard message exchanges that allow one service to subscribe and unsubscribe to another, and to receive notification messages from that service.
- WS-BrokeredNotification defines the interface for notification intermediaries. A Notification Broker is an intermediary that decouples the publishers of notification messages from the consumers of those messages. This allows publication of messages from entities that are not themselves Web service providers.
- WS-Topics defines an XML model to organize and categorize classes of events into "Topics", enabling users of WS-BaseNotification or WS-BrokeredNotification to specify the types of events in which they are interested.

The WSN specifications standardize the syntax and semantics of the message exchanges that establish and manage subscriptions and the message exchanges that distribute information to subscribers. An information provider, known as a notification producer, that conforms to WSN can be subscribed to by any WSN-compliant subscriber.

WS-Eventing

The WS-Eventing specification defines a baseline set of operations that allow Web services to provide asynchronous notifications to interested parties. WS-Eventing provides basic publish/subscribe functionality. WS-Eventing provides similar functionality to that of WS-BaseNotification so we will not present further details here. The overall concept is the same, but the two are not compatible with each other at the message level.

3.2.2 Quality of Service

In [131], QoS is defined as *the set of those quantitative and qualitative characteristics of a distributed multimedia system necessary to achieve the required functionality of an application*. Such QoS aspects as security, reliability, and transaction management have been addressed, whereas none of the other issues (e.g., prioritization and preemption) have standardized solutions for Web services today (see our report [70] for details). QoS is an important aspect of NBD, and must be taken into consideration to be able to realize the NBD vision. There is currently no standardized QoS model for Web services, but OASIS has a work group dedicated to creating a Web Services Quality Model (WSQM) [94], which aims to secure Web services at a specific level of service quality.

QoS is mentioned here to complete the overview of the most important Web services specifications, but QoS research is beyond the scope of this report, since that is being covered by other ongoing work at FFI (see [77]).

Security

Security is an important part of any middleware. Web services technology is increasingly being employed on the Internet, both within organizations and more importantly, between them, providing business to business services. To protect the confidentiality and integrity of data being transmitted, security measures are a necessity to guarantee the success of a service and to prevent fraud and financial loss. Even more so, to protect national interests one must consider security issues when contemplating using Web services to realize NBD. There is a set of standards for providing security to XML like e.g., XML Digital Signature, XML Encryption, SAML and XACML. These standards are currently under evaluation by a NATO. For a discussion of security in Web services and XML, see [91].

Reliable messaging

Reliable messaging standards focus on improving application reliability by adding functionality such as guaranteed message delivery to SOAP. The purpose of these standards is to add end-to-end reliability which is independent of the transport protocol, so that message delivery is reliable over

multiple hops even when using unreliable transport mechanisms. The goal is to make sure that message delivery occurs exactly once and in order.

There are two reliable messaging standards, WS-Reliability and WS-ReliableMessaging [63]. WS-Reliability was the first to be approved as a standard, but it lacks transaction support, secure session handling and, due to its early release, does not take other Web services standards into consideration. Because of these limitations vendor support for this standard is limited, whereas the other reliability standard, WS-ReliableMessaging, has more industry backing and is widely supported.

WS-ReliableMessaging is not a complete messaging solution in itself. Instead, WS-ReliableMessaging is a building block used to extend other Web Services specifications (i.e., SOAP and WSDL) to build a complete and reliable messaging solution. The protocol defines and supports a number of delivery assurances for a single message or a group of messages: *AtLeastOnce* — Each message will be delivered at least once. If a message cannot be delivered, an error must be raised. Duplicates are allowed, meaning that messages may be delivered more than once. *AtMostOnce* — Each message will be delivered at most once, meaning that a message may not be delivered at all, but if it gets through then it will never be duplicated. *ExactlyOnce* — Each message will be delivered exactly once. If a message cannot be delivered, an error must be raised. Duplicate messages shall not occur. *InOrder* — Messages will be delivered in the order that they are sent. This assurance can be combined with any of the above assurances.

Transactions

Transaction support is often necessary when integrating applications, and protocols for such support is therefore important. WS-Transaction [97] is a set of specifications, built on top of the WS-Coordination framework, that defines protocols for transaction support in Web services. Because Web services in many areas differ from traditional middleware (lack of centralized middleware platform, long running transactions, lack of a fixed resource model), the traditional transactional model (the "ACID"-properties: Atomicity, Consistency, Isolation, and Durability) is relaxed, and instead compensation mechanisms are used [3]. An application requires compensation if its operations cannot be atomically rolled back, but how services do their work and provide compensation mechanisms is not the domain of the WS-Transaction specification.

3.2.3 Description

The standards that are related to Web services description can be divided in two main subcategories, namely *metadata* and *service discovery* standards.

The metadata standards focus on providing a framework that can be used to describe the requirements of a service, including its operations, message format, input and output parameters, location information and which other Web services standards the service supports. The main metadata standard is WSDL, but WS-Policy and WS-Inspection also fall in this category.

Service discovery standards such as UDDI, on the other hand, are used to make services available to potential service consumers.

Web Services Description Language (WSDL)

WSDL [27] is an XML language for describing Web services. The current version is 2.0 [134], available as a W3C recommendation from 2007. However, version 1.1 [133] is still being used a lot, since several development tools work with this version of WSDL.[5] Since XML is used, Web services definitions can be utilized by any implementation language, platform, object model, or messaging system. The specification defines a core language which can be used to describe Web services based on an abstract model of what the service offers.

A WSDL service description indicates how clients are supposed to interact with the described service. It represents an assertion that the described service implements and conforms to what the WSDL document describes. A WSDL interface describes potential interactions with a Web service, not required interactions. The declaration of an operation in a WSDL interface is not an assertion that the interaction described by the operation must occur. Rather it is an assertion that if such an interaction is initiated, then the declared operation describes how that interaction is intended to occur. By using WSDL, it is possible to create a formal, machine-readable description of a Web service, making it possible for clients to invoke it.

WSDL service definitions provide documentation for distributed systems and serve as a recipe for automating the details involved in applications' communication. Thus, WSDLs are a crucial part of Web services, since they define the interface through which you access services, as well as information about where the service can be found in the form of an URL.

We will now take a closer look at a WSDL 1.1 document, since that is the foundation of all the Web services implementations we discuss later in this report. A WSDL 1.1 document uses the following elements in the definition of network services:

- *Types*, which is a container for data type definitions using some type system, such as XML Schema Definition.
- *Message*, which is an abstract, typed definition of the data being communicated.
- *Operation*, which is an abstract description of an action supported by the service.
- *PortType*, i.e., an abstract set of operations supported by one or more endpoints.
- *Binding*, containing a concrete protocol and data format specification for a particular port type.
- *Port*, which is a single endpoint defined as a combination of a binding and a network address.

[5] This is because currently only WSDL 1.1 is addressed by the WS-I basic profile.

```
<?xml version="1.0"?>
<definitions name="PositionUpdate"
             targetNamespace="http://example.com/PositionUpdate.wsdl"
             xmlns:tns="http://example.com/PositionUpdate.wsdl"
             xmlns:xsd1="http://example.com/PositionUpdate.xsd"
             xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
             xmlns="http://schemas.xmlsoap.org/wsdl/">
  <types>
    ...
  </types>

  <message name="GetLastPositionInput">
    <part name="body" element="xsd1:PositionRequest"/>
  </message>
  <message name="GetLastPositionOutput">
    <part name="body" element="xsd1:Position"/>
  </message>

  <portType name="PositionUpdatePortType">
    <operation name="GetLastPosition">
      <input message="tns:GetLastPositionInput"/>
      <output message="tns:GetLastPositionOutput"/>
    </operation>
  </portType>

  <binding name="PositionUpdateSoapBinding" type="tns:PositionUpdatePortType">
    <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
    <operation name="GetLastPosition">
      <soap:operation soapAction="http://example.com/GetLastPosition"/>
      <input>
        <soap:body use="literal"/>
      </input>
      <output>
        <soap:body use="literal"/>
      </output>
    </operation>
  </binding>

  <service name="PositionUpdateService">
    <port name="PositionUpdatePort" binding="tns:PositionUpdateSoapBinding">
      <soap:address location="http://example.com/PositionUpdate"/>
    </port>
  </service>

</definitions>
```

Figure 3.2: An example WSDL file.

- *Service*, which is a collection of related endpoints.

Figure 3.2 presents a WSDL file that we have created as an example. It describes a simple *service* called *PositionUpdateService*. The *types* section has been omitted for brevity, but it would contain the data type definitions being used, i.e., a way of expressing position coordinates, for example longitude and latitude, and perhaps also altitude. The *message* sections contain information about the data being communicated, in this case two message types are defined: We have a message called *GetLastPositionInput* and a message called *GetLastPositionOutput*. The former is used to issue a request to the service, telling it to respond with a position, i.e., with a message of the latter type. This use of the defined messages is described in the *portType* section, where an operation called *GetLastPosition* is defined in terms of *GetLastPositionInput* as the input message

type and *GetLastPositionOutput* as the output message type. In the *binding* section, we see that the service is bound to SOAP over HTTP, which is by far the most common choice for Web services transport. Finally, a *port* called *PositionUpdatePort* is defined as using the binding to SOAP, and thus providing the necessary information to use the *PositionUpdateService*. In practice, all position services provided in a network would have a WSDL like this, with one important exception — the *address location* would change for each deployed instance of the service. For this particular example, the address is "http://example.com/PositionUpdate".

Web Services Policy Framework (WS-Policy)

WS-Policy is an important framework for introducing policy support to Web services. The framework is supported by several companies, such as IBM, BEA Systems, Microsoft, SAP AG, Sonic Software, and VeriSign [138]. WS-Policy provides a flexible and extensible grammar for expressing the capabilities, requirements, and general characteristics of entities in a system based on Web services. WS-Policy defines a framework and a model for expressing these properties as policies. Policy expressions allow for both simple declarative assertions as well as more sophisticated conditional assertions. WS-Policy defines a policy to be a collection of one or more policy assertions. Some assertions specify traditional requirements and capabilities, for example, authentication scheme, and transport protocol selection. Other assertions specify requirements and capabilities that are critical to proper service selection and usage, for example a privacy policy. WS-Policy provides a single policy grammar to allow both kinds of assertions to be reasoned about in a consistent manner. For further details about WS-Policy and other security related standards, see [91].

UDDI

The OASIS standard Universal Description, Discovery and Integration (UDDI) [92] allows service providers to register their services and service consumers to discover these services. UDDI defines entities to describe businesses and their services. UDDI is defined as a Web service itself, meaning that the SOAP protocol must be used to interact with the registry.

Although UDDI is considered to be one of the core Web services standards, its adoption by enterprises has lagged behind that of the other cornerstones of Web services — SOAP and WSDL. Gartner surveys show that fewer than 10 percent of businesses use UDDI for their Web services registries [48]. This is probably because you can easily use Web services in your corporation without using more than SOAP and WSDL — you do not necessarily need a registry present — as we discuss in Section 3.3 below.

UDDI is discussed in more detail later when we address service discovery in Chapter 5.

Web Services Inspection Language (WS-Inspection)

WS-Inspection is a specification by IBM and Microsoft [9]: It provides an XML format for assisting in the inspection of a site for available services and a set of rules for how inspection related information should be made available for consumption. In other words, it provides a static index document of services provided by a node.

WS-Inspection was created in 2001, but has not been widely adopted. It has not been standardized and is no longer considered for service discovery. Three standards (i.e., UDDI, ebXML, and WS-Discovery) have emerged that attempt to fill the service discovery needs (see Section 5.1.3), and they have much more inherent flexibility than a simple index mechanism like WS-Inspection.

3.2.4 Composition

The composition standards address the business process aspect of Web services and add support for modeling process flow and service-to-service integration. There are two main types of service composition, namely orchestration and choreography.

Orchestration is the composition of services controlled by one organization, and is mostly used within a business to control how the services owned by that business should cooperate. The WS-BPEL standard [95] (BPEL4WS) is the leading standard within orchestration, and adds workflow type logic to a set of service operations and also allows for maintaining limited business related state within a process. The standard is intended for use within one business, and has limited functionality when it comes to multi party collaboration.

The second type of service-to-service interaction is choreography, which focuses on composing more advanced services by using basic services provided by several different enterprises. Choreography is focused on interactions that occur between these enterprises rather than on enterprise internal composition. This kind of service composition is covered by WS-CDL [137], a service choreography language, which can be used to describe collaboration interfaces between parties. For a discussion of composition of Web services for military networks, see [114].

3.3 Two civil real world examples

These two examples show how Web services are commonly employed in civil systems today, discussing a composite service and a wrapped service, as shown in Figure 1.3.

3.3.1 Travel agency: A composite service

For business use over the Internet, you can often assume that the endpoint of a Web service is static. Typically, the endpoints do not change. Take for example a Web shop, which is implemented as

a Web application.[6] The user wants to book a complete vacation online, consisting of an airplane ticket, a hotel room, and a rental car. In this case, the back-end of the Web application may invoke several Web services. In fact, it can be a so-called mashup[7] [11,25], where the application invokes a Web service for an airline, a Web service for a hotel, and a Web service for car rental. Each of these services may belong to different companies. These services will offer information on availability. Last but not least, when the customer accepts the information presented in the Web application and enters a credit card number, another Web service is invoked to check that the number entered is valid, and also that there is sufficient credit available. If all these checks pass, then the credit card is charged with the proper amount, and the respective Web services invoked to fulfill the booking of airplane ticket, rental car, and hotel room. A receipt with the relevant information is then prepared and displayed to the customer in the browser by the Web application. For business use on the Internet, this is a typical use case for Web services technology. There is no need to do run-time discovery, because your business contracts give you access to certain services that were made available to you when you created your application, i.e., during design-time. This example illustrates how Web services from multiple companies can be utilized to make one application. The fact that Web services are based on standards makes this interaction possible and the results well defined.

3.3.2 Finance: Wrapping legacy applications

Web services technology can also be of significant benefit when it is employed within a company as well. One example illustrating this is that of the Norwegian bank Storebrand's Web application. This application allows customers to log in and administer their account, pay bills, and so on. By exposing services from the legacy banking system as Web services (i.e., wrapping the legacy system in Web services), the bank was able to create a flexible connection between the customer and the back-end system. Thus, allowing the customer to do more of the job and reducing the need for cashiers. Storebrand is also involved in insurance, and has wrapped that system in a similar fashion. This means that their customers cannot only do their banking themselves, but also handle their insurance policies. Many banks have similar solutions today, but in Norway Storebrand was one of the first. By using Web services, the company gains a lot of flexibility when it comes to future maintenance and development of their Web application. Since Web services are based on standards, and many development tools exist, it is easy to get basically any consultant to maintain or expand the software. In this case it is also sufficient with design-time service discovery, since once the application is created it just needs to access the Web services exposing the various legacy system services, and they will be stable and up and running in the bank's internal network.

[6] A *Web application* is an application that leverages Web technologies and is used through a Web browser such as Internet Explorer or Firefox. A Web application is intended for use by humans. This is a contrast to *Web services*, which are intended for machine-to-machine interactions only.

[7] A *mashup* is a service or application which utilizes other, pre-made services. Naturally a composite service can be made using a formal language such as BPEL4WS, but more often than not the invocation sequence for the external services is merely hard-coded in the application.

3.4 Two military examples

In this section we explore two military applications of Web services. There are many standards and systems for friendly force tracking, but lately a new standard has emerged as one of the first military applications leveraging Web services technology. In other areas of NBD, such as in cooperative electronic support measures operations, service orientation is still lacking. We use it as an example of how you can wrap legacy systems with Web services technology and gain added benefits which were not available before, as we present our use of Web services in an operational experiment.

3.4.1 Friendly force tracking

Friendly force tracking (also often called *blue force tracking*) is recognized as one of the most important aspects of the NBD concept. In complex endeavors where several different nations take part, blue force tracking is important to avoid possible blue-on-blue situations. To facilitate interoperability between nations, NATO has specified a format for exchanging friendly force tracking information: NATO Friendly Force Information (NFFI). A part of the NFFI specification is an XML schema to allow the exchange of blue force tracking information using Web services.

The current version of NFFI is 1.3 as published in draft STANAG 5527. NFFI consists of a message definition and message protocols. It is currently in use in Afghanistan where coalition force members can share position information between their HQs. That is, it is being used for interoperability between the deployed tactical networks of the coalition forces.

The message format is defined by an XML schema containing both mandatory and optional fields. Each NFFI message can contain one or more *tracks*. A track must contain the mandatory NFFI options, and can also contain an arbitrary amount of optional data. Figure 3.3 gives an overview of the NFFI fields. The position and dynamics data (PDT) is a mandatory part of the document and contains information about position (longitude, latitude, altitude), velocity, a timestamp, and a tracker identifier. Thus, the PDT contains the minimum information needed to draw a symbol on a map. All the other fields are optional and may contain contact information, telephone numbers etc. For example, a 15 character text string from APP-6A [130], a NATO standard for military map marking symbols can be added. Furthermore, a status field may contain the operational status of the object.

To make systems interoperable at all levels, it is desirable to use XML encoded NFFI also on the mobile tactical level. XML, while being a standardized way to structure data, leads to large text documents that need to be exchanged. At the tactical level resources are scarce, and measures must be taken if one is to use an NFFI Web service in a disadvantaged grid. We explore this further in Chapter 4 (see also our experiment report from CWID 2007 [54]).

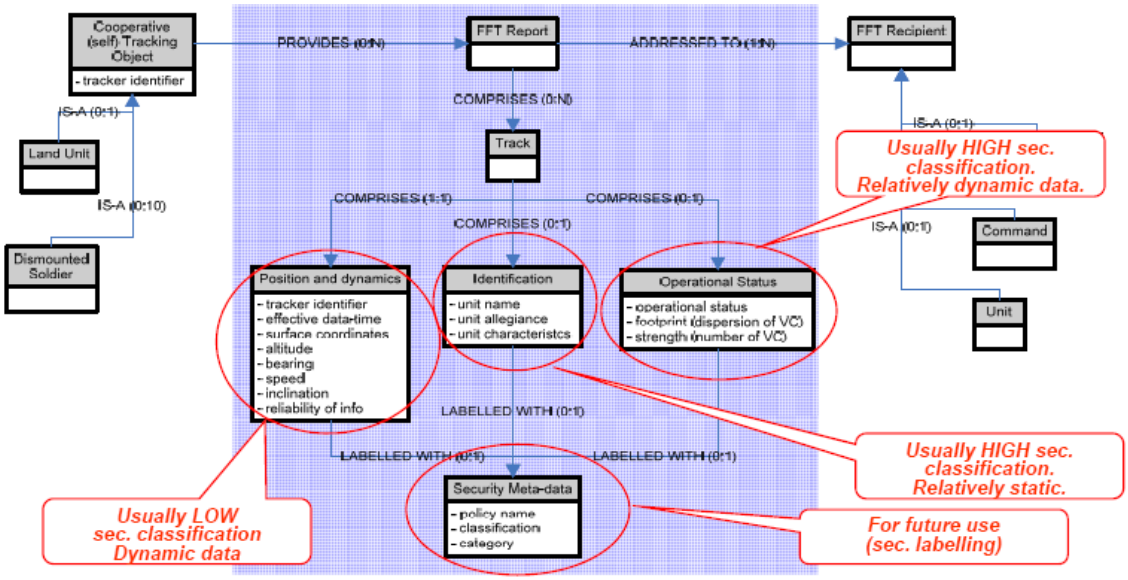


Figure 3.3: Data model showing mandatory and optional NFFI data, along with usual security requirements (from [106]).

3.4.2 Cooperative Electronic Support Measures

Special radio receivers for the detection of emissions from radars and communication systems are called Electronic Support Measures (ESM) by the Electronic Warfare community. The ESM sensors measure the radar frequencies and other characteristic parameters of the signal, and the direction or bearing to the emitter.

By tracking emitters over time based on bearings from one ESM sensor, it can take a long time to establish target solutions, and the solutions may have limited accuracy.

With cooperating ESM sensors one can process simultaneous observations of an emitter from several ESM sensors. Solutions can be obtained more rapidly and more accurately than with one sensor. Additionally, one can use other and better methods that produce a more accurate emitter location. Cooperative ESM operations (CESMO) are mainly concerned with the detection and localization of radar emitters by cooperation between many sensor platforms, as illustrated in Figure 3.4.

According to the present CESMO operational concept, an ESM sensor platform can have two roles:

1. ESM sensor platform
2. Signal Intelligence Identity Authority (SIA)

All platforms can have the role of SIA, but the role is often given to the platform with the best resources, both sensor and operator wise. Depending upon the type of operation, some

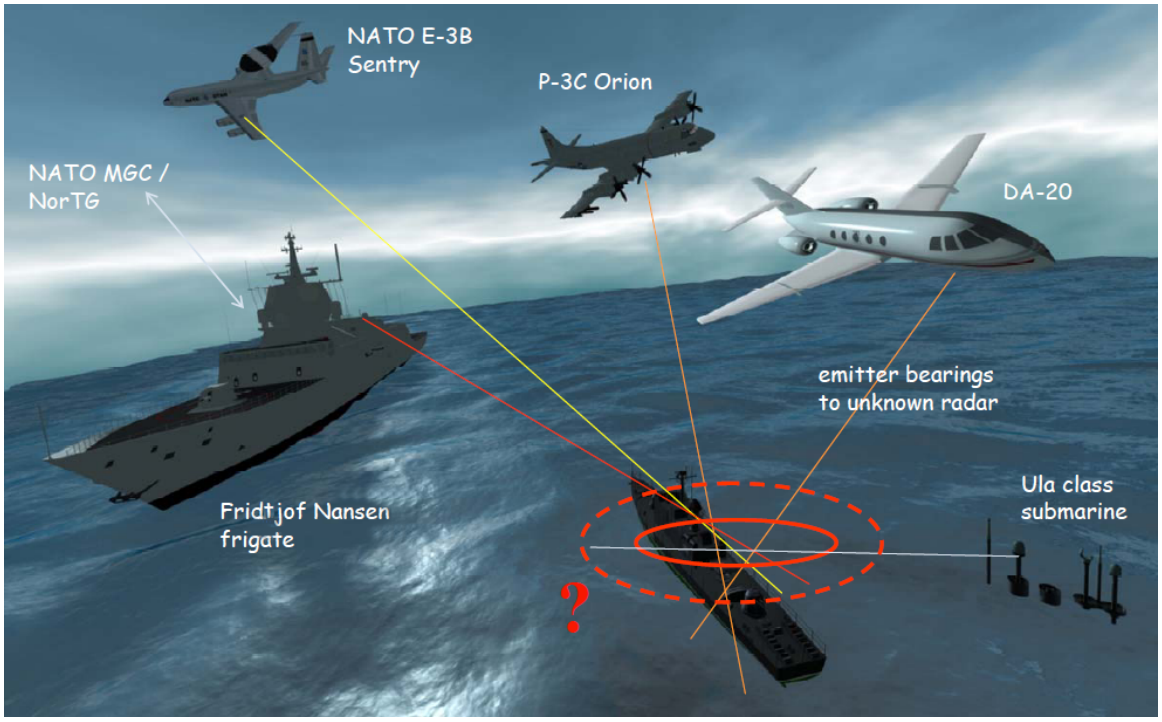


Figure 3.4: The CESMO concept (from our report [71]).

hostile emitters are more interesting than others. When an interesting emitter is detected, Signal Coordination Messages are sent on the network from all sensor platforms that have observed the emitter. Based on the received messages the SIA computes the emitter location, and reports the emitter location in a Geolocation Message to all participating platforms. The participants also send Participant Status Messages at irregular intervals.

The main motivation for using CESMO as a case for Web services was to show that it is feasible to use Web services in existing operational systems, and that it brings substantial benefits. It was also important to show these benefits for operational personnel.

One of our goals was to show that using Web services in the CESMO network can contribute to automate and speed up operations. We added a Web services front-end to each platform (i.e., wrapping them), which made them appear as services. Each platform could then use the services of other platforms.

The network interface of the CESMO software follows the CESMO Interface Control Document (ICD), also referred to as STANAG 4658. The ICD defines byte oriented messages in a fixed format.

CESMO is a good example of operations where sharing of data is essential for the success of the operation. Traditionally, signal intelligence and Electronic Warfare are areas where the sharing of data even between close allies has been limited.

We wanted to show how Web services enable information exchange between stove pipe systems, possibly in different military services. In other words, how they can function as an integrator,

making it possible to find information and make it available across tactical and strategic systems in different military services. This was shown by connecting the CESMO network to a NORTaC-C2IS system[8], and then converting[9] and exporting the geolocations into NORTaC-C2IS. From NORTaC-C2IS, the tracks were further distributed into a strategic command and control system called NORCCIS-II [129]. By doing this, we were able to show how information could be gathered by both air force and naval units, be processed in a central location, and then distributed to interested parties within a different military service.

Another illustration of such information exchange across operational levels is the fact that we connected the CESMO network to the Joint Electronic Warfare Coordination Cell (JEWCC), located at the Joint HQ at Jåttå, Stavanger. The JEWCC is responsible for coordinating all activity related to electronic warfare, and as such it is dependent on receiving information about all radio activity in the operational area. Today, the JEWCC usually receives such information late, but by using Web services, we were able to provide the JEWCC with live information from the operational area.

Lessons learned

We were able to test quite a few aspects of Web services (for a complete overview of our experiment, see [71]). The most important lessons that we learned are summarized below:

- We were able to create Web services by wrapping legacy CESMO software. Additionally, we were able to offer several new Web services based on information from the CESMO system, namely a NFFI service and a CSV service. These new services allowed previously separate systems to receive information from the CESMO network, thus showing the potential added value of employing Web services.
- By employing WS-Eventing we could see the benefits of publish/subscribe, in that one could easily control the flow of information between the different parts of the system. Subscriptions allowed information consumers to subscribe to only those messages which were of interest to that particular system. Published messages went out only to subscribers, and were sent asynchronously with no need to poll for new data. This reduced network load since only relevant and needed information was sent over the network at any time.

However, the solution did not run flawlessly during the entire experiment. At times our experimental software would stop functioning correctly. Monitoring the network traffic and identifying the times when the solution ran into problems, we found that they were related to the military hardware's limitations. The available bandwidth (16 Kbit/s) was not a problem here, but we identified several others.

[8] NORTaC-C2IS [78] is a tactical command and control application used by the Norwegian army.

[9] NORTaC-C2IS does not understand ICD/STANAG 4658 messages, so we converted the geolocations to NFFI-tracks instead. NFFI is discussed in Section 3.4.1.

It turns out that the Universal Improved Data Modem (UIDM) [36] has several limitations that need to be taken into account when using it:

- There is limited support for IP fragments, so the application should not send very large packets (or it should fragment its packets itself).
- The maximum transfer unit (MTU) is much lower than that which is usual for an Ethernet, so the messages/fragments must reflect this.
- There is limited buffer space in the modem. During testing we crashed the modem by sending too many packets to it in rapid succession. Thus, it is necessary to control the communication burstiness of the applications (congestion control).

These limitations mean that standard Web services will not always function well in tactical networks. Thus, we found that we cannot merely employ the standard solutions as they are, we need to investigate possible ways to adapt Web services to tactical networks. This is where the engineering efforts stop and we need to address the research issues of invoking and discovering Web services in military networks, which is the contribution of this report.

3.5 Requirements analysis

When designing an architecture for NBD, it is important to note that constraints on network availability and topology, available services, intended users and required robustness all vary with each deployment, and thus add to the complexity of such an architecture.

Summarizing the discussion of different operational levels (see Figures 1.1(a) and 1.1(b)) from the previous chapters, we can state the following about the three levels:

We know that the strategic level has a large fixed infrastructure and hosts a large number of services. In these networks, there will be hundreds to thousands of nodes. This domain requires solutions that can scale to a large number of users and contain information about a large number of services.

The tactical deployed level can use a mostly fixed infrastructure. Such networks constitute the backbones of the deployed networks, for example a local HQ. However, these deployed networks need to communicate with other networks both at the same operational level and with other networks higher or lower in the hierarchy. For such communications, they employ radio or satellite links, which are prone to disruptions. Thus, at this level, we have fairly large fixed networks, but some dynamicity is expected due to their interconnection with unreliable links. The networks are large, with hundreds to thousands of nodes and services. At this level, we need solutions that can scale to a large number of users, and that can handle some dynamicity.

The tactical mobile level, the lowest level in the hierarchy which constitutes the so-called disadvantaged grids, differs a lot from the two other levels higher up in the hierarchy. Here, the

Architectural property	Operational level
Premise: Solution must support a federation of systems	All
Premise: Solution should use Web services technology	All
Always use standards when possible	All
Support for radio silence	Tactical
Support for limited bandwidth	Tactical
Support sudden disruptions	Tactical
Support low MTU	Tactical
Should not rely on TCP	Tactical
Should not rely on IP fragments	Tactical

Table 3.1: Requirements analysis stating premises and properties that must be fulfilled, and to which operational level(s) each item applies.

networks are small, each network with perhaps four to twenty nodes, and a small, mission specific set of services (sensors, positioning information, etc.). The networks use wireless links with all the drawbacks and limitations we discussed in Chapter 2. At this level, we need a solution that can handle a highly dynamic environment. It should also be resource efficient, since resources are scarce.

As we can see, the different networks may call for different solutions. In the fixed networks, scalability is the most important aspect. In disadvantaged grids, on the other hand, we need solutions that can handle high mobility and at the same time minimize resource use. In these networks, conserving resources and supporting mobility is much more important than scalability, since disadvantaged grids usually are very small networks compared to those higher up in the hierarchy.

Hence, we have identified several key architectural properties necessary for Web services to function in dynamic environments. These properties (i.e., the requirements analysis mentioned in Section 1.3, item number one) are summarized in Table 3.1. In a NATO federation of systems it is unrealistic to expect that one can choose and deploy a unified solution. Furthermore, the demand that standards should always be used when possible is very important for interoperability reasons. In later chapters when we evaluate different solutions, this property is one of the most important ones.

3.6 Summary

Interoperability is crucial when attempting to fully realize NBD, but achieving such interoperability is a considerable challenge given the large number and heterogeneity of the different systems currently being used, and the scarcity of resources on the tactical level. Middleware is an abstraction layer that can conceal the heterogeneity of underlying hardware in a distributed system, and thereby represent a solution to this challenge.

Web services are the most common way of realizing a SOA. However, Web services are still far

from being mature enough for playing the role as a "middleware of middlewares"; especially on the standardization side, much work remains since (e.g., lack of several important QoS aspects). Standards are important for interoperability, and it is important to avoid proprietary solutions to avoid vendor lock-in. The greatest strength of Web services technology is that it is interoperable across different operating systems and different programming languages. You can connect systems running on OS X, Windows, or Unix, and applications implemented using Java, Python, Microsoft .NET, and others. This is the main reason why Web services are in such widespread use today. Thus, in this report we focus on employing established standards when possible.

The core standards, XML, SOAP and WSDL are widely supported. They form the foundation of Web services, and are the absolute minimum one needs to support in order to use Web services as a message oriented middleware. You use the WSDL to specify the service and client interfaces, and SOAP to convey messages between the service and client instances. XML is used for "everything" in the Web services world, since it is the language used to formally define every schema used in the Web services standards and specifications. Some areas, such as interaction and description (see Figure 3.1), have standards and implementations that are well developed and interoperable. However, topics like QoS and composition are still lacking standardization and non vendor specific implementation. Thus, it is possible to start employing the mature standards now, and modularly expand the middleware functionality as additional standards gain widespread use. This is where WS-I plays an important role. To ensure interoperability between implementations from different vendors, a standard should be addressed by a WS-I profile.

In the examples we have seen that commonly only the "bind" operation is performed in civil applications. We have also seen that Web services can bring added value to military applications. The main drawback of Web services is the overhead: XML is text based, and thus a quite verbose way of encoding data. This can be a problem in networks with low throughput. Also, since HTTP/TCP does not always work in tactical networks, there is a need to address the particular requirements of disadvantaged grids as well. As we will discuss in subsequent chapters, there are techniques that can be used to optimize Web services in different ways. In the next chapter, we will investigate techniques that can be employed for optimizing *invoking* Web services for tactical networks.

4 Invoking Web services

The first step towards NBD is to integrate legacy strategic and tactical systems into a common network. For such integration, the modular concept of SOA is essential. Each legacy system can be viewed as a separate module that needs to be interconnected with others (i.e., needing to be wrapped as a service). In order to get the different modules to cooperate, one needs a common standardized means of communications between them. This infrastructure should preferably be built using standards and commercial off-the-shelf (COTS) products to keep acquisition and maintenance costs low. Web services technology is in widespread use on the Internet today, and COTS products are readily available. Adopting Web services also seems to be a general trend in defense since both NATO and the Network Centric Operations Industry Consortium [89] support the Web services standards. As a consequence, we also investigate Web services for use in NBD.

Web services promote interoperability between different systems, but at the same time they increase the information overhead significantly, resulting in higher data rate demands. In corporate networks this is not much of an issue, since bandwidth is abundant, but the situation is different for radio systems in military tactical networks. There, military requirements for long transmission range, security, robustness, anti-jamming and protection from wiretapping often result in limited data rates and other issues affecting communications (i.e., the *disadvantaged grids*).

In this chapter, we address research claim #1, that *to be able to invoke Web services in military networks with constraints, adapting the standards is needed.* We consider methods to reduce the XML and Web services overhead, in order to make it possible to use Web services in tactical networks. Our experiments show that it is possible to use Web services even in networks with low data rates, as long as data compression and efficient communication protocols are employed. Finally, we discuss how proxies can contribute to reduce the overhead, increase the availability and lower the delay of network communication.

4.1 Introduction to Web services invocation

Client applications can use Web services as a middleware to connect to services provided they know their invocation address. In this chapter we assume that this address is available, and investigate invoking Web services. First, a client needs to form a request (i.e., a service invocation following the message format dictated by some WSDL), put it in a SOAP envelope (i.e., forming the SOAP body) and send it to a service. The most common transport protocol for SOAP is HTTP over TCP. The service can then receive the SOAP message, retrieve the XML encoded request from the SOAP body, de-serialize it and treat it as a function call in the service application. If there is an error, then a SOAP fault message will be sent to the client (yet again according to the WSDL definition). If there is no error, and the service needs to return some data (specified by the WSDL), then it will serialize its reply to an XML document, put this document in a SOAP message, and send it back to the client.

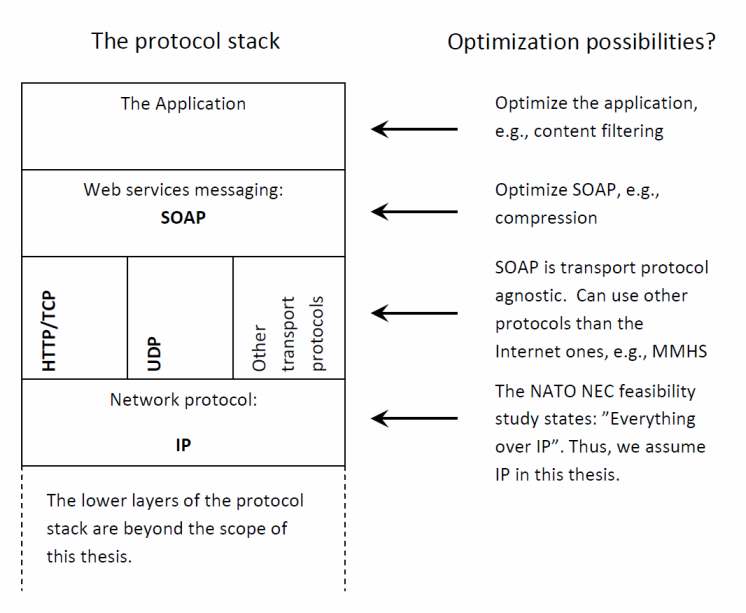


Figure 4.1: Optimizations can be performed several places in the protocol stack.

XML, being text based, is a verbose way to encode data. It is good for human readability, and is easy to parse for a computer, which is why it is suitable for interoperability reasons. However, it is not good for transmission over networks with limited data rate. In Section 3.5 we summarized a set of requirements that must be met in military networks. In this chapter we investigate these requirements in the context of Web services invocation. We need to reduce the resource use of Web services if we are to use the technology in tactical networks. To reduce the overhead, we can try to optimize the application by reducing its need to transmit data. Furthermore, we can try to reduce the overhead of SOAP, the Web services messaging standard. Since SOAP is transport protocol agnostic, we can also attempt to replace HTTP/TCP with other protocols. See Figure 4.1 for an overview of the optimizations that we consider in this chapter.

4.2 Related work

There have been a few research efforts related to both reducing XML overhead and communication overhead for civil applications, since using Web services on mobile phones can be a challenge. Mobile phones have a few of the same restrictions as mobile military tactical systems, in that processing power, battery life, and available data rate is much more limited than that of a PC connected to the Internet. Also, proxies that can perform optimizations on behalf of COTS Web services and clients exist.

There are different ways to optimize Web services for networks with low data rates: You can optimize the application, i.e., reduce the amount of information the application needs to transmit over SOAP. You can optimize the encoding, i.e., address the overhead of XML by for example

compressing the Web services payload. Also, you can address the overhead of the transport mechanisms, and use different protocols to transmit your SOAP messages over the network. Figure 4.1 illustrates where optimizations can be performed. Existing research usually targets one of these areas for optimization. In this report we want to combine the techniques to gain the best overall optimization benefits.

4.2.1 Reducing XML overhead

In [12], an experimental electronic wallet is implemented using Web services on mobile devices communicating over Bluetooth and 802.11. The paper identifies several challenges related to mobile devices, among other things that XML and SOAP are bandwidth expensive. It is possible to use compression to make Web services less bandwidth expensive, thus addressing the issue of XML and SOAP's bandwidth consumption.

In [6] several challenges regarding mobile wireless Web services are identified and addressed. One of the most important optimization factors identified there is compression. The basic idea behind compression is that by applying an encoding algorithm, information can be represented in a more efficient manner (i.e., using fewer bits to represent the same information). XML, being text based, is especially compression friendly, and since XML is a structured format, there also exist several XML specific compression techniques.

In [73] we have shown that it is feasible to use Web services on mobile phones. In addition to performing simulations, we verified our results using real GPRS and UMTS network connections. We tried different compression algorithms, and found that they were necessary to reduce the XML overhead of Web services.

Compression algorithms

There are two types of compression; *lossless compression* and *lossy compression* [116]. Lossless compression is used on data that needs to retain its exact representation when it is decompressed. Lossy compression is used on data that can tolerate some loss such as audio, pictures and video. Lossy compression can, since it is allowed to modify the data, achieve higher compression rates than lossless compression. For documents (in our case XML documents), however, we need all the information to be intact so lossless compression should be used.

We differentiate between two[10] lossless compression techniques:

[10]Note that we focus on application level compression in this report, since we address the overhead of XML and Web services. In addition to the techniques we discuss here, it is also possible to employ compression in lower layers of the protocol stack, e.g., robust header compression [16] at the networking layer. Such optimizations may or may not be present in the network, but optimizations at the networking layer and below are beyond the scope of the project that laid the premises for this report.

- Generic compression
- XML-aware compression

Generic compression methods have a single goal — to reduce the size of any document. Typical examples are GZIP and ZLIB (see [90]). The compressed document cannot be used directly by applications; it has to be decompressed first. GZIP was initially developed to produce a compression utility that was independent of any patents, and its file format is specified in [30]. It is based on ZLIB [31] and DEFLATE compression [29].

XML-aware compression methods operate only on XML documents. Some methods are only concerned with achieving the optimal compression rate. Other methods target preserving the XML properties of the data, while introducing a binary encoding that significantly reduces the data size. This means that the XML information structure and hierarchy are fully preserved, while the information is stored in a machine-readable binary format instead of in a human readable XML text format, leading to a more compact and efficient representation. Applications that support the binary encoding of XML data may edit the binary XML files directly, without converting them back to the original text format. The potential advantages are most obvious in devices with limited computation, battery and memory resources, such as mobile phones: XML data processing is potentially faster, requires less memory and is faster to transfer through the mobile networks with limited bandwidth. Several compression techniques exist. According to the literature [7,90] a select few are comparably much more efficient than the rest. In this report we chose to focus on those particularly promising algorithms: In [90], XMLPPM gave the best average compression ratios of the XML-conscious compression techniques, while GZIP was the best of the generic compression algorithms tested. XMLPPM requires a lot of processing power and memory when compared to GZIP, thus it is not suited for use on mobile devices with limited resources.

Efficient XML (EFX) was one of the formats the W3C XML Binary Characterization Working Group investigated during their work with requirements for a binary XML format. It was later adopted by the W3C Efficient XML Interchange Working Group (EXI) [34] as the basis for the specification of the efficient XML format. The objective of the EXI Working Group is to develop a specification for an encoding format that allows efficient interchange of the XML Information Set, and to illustrate effective processor implementations of that encoding format. In 2007, the group released its third working draft, and in December 2009 EFX became a W3C candidate recommendation [139]. EFX was originally developed by Agile Delta and provides a very compact representation of XML information. It is worth noting that Agile Delta is actively participating in the EXI work, and its EFX product — which was used in this report — conformed to the second working draft at the time we performed our experiments. The studies in [90] and [7] did not investigate EFX, since it is a rather new technique, but [7] mentioned its existence and that it should be evaluated in future work. GZIP and EFX implementations are available for several platforms, including Java SE and ME, making them suitable for use in all kinds of devices. XMLPPM is available as C++ source code (available from http://xmlppm.sourceforge.net/). It is more computationally expensive than GZIP and EFX, thus making it less suitable for use in devices with limited processing power.

4.2.2 Reducing communication overhead

Werner et al. [145] point out that bandwidth and latency are issues when using SOAP. Compression is mentioned as a means of reducing some of the overhead of SOAP, as well as differential coding of SOAP messages. However, one of the main reasons for latency can be attributed to the transport protocol. Connecting and disconnecting TCP when using SOAP over HTTP/TCP contributes to latency. Measuring several suggested SOAP transports such as SMTP, FTP, MSMQ, TCP (SOAP over TCP, not HTTP/TCP), and UDP, the study shows that of these protocols, SOAP over UDP has the least overhead. However, SOAP over UDP has two main drawbacks compared to HTTP/TCP: UDP is unreliable, and the SOAP message is limited to 64 Kb. The unreliability issues of SOAP over UDP can be addressed by using application level mechanisms such as WS-ReliableMessaging, which adds reliability to SOAP. To address the limitations in message size, a protocol called PURE is suggested, which is a multi-packet binding for SOAP. However, outside of this study, we have not encountered PURE. It is not in actual use today, unlike SOAP over UDP, which became a Web services standard during summer 2009.

In [104], Pham et al. introduce Web services as a middleware for mobile devices, using SOAP over HTTP/TCP. They show that it is feasible to use Web services on mobile devices when using lightweight packages such as kSOAP and kXML. As an optimization for mobile devices, Gehlen et al. [49] replace HTTP with WAP and evaluate a SOAP over WAP transport implementation. While WAP is not a relevant protocol for military tactical networks, the idea of replacing the usual HTTP/TCP binding of SOAP with another protocol which is better suited to your network is interesting. For military networks, one could consider using special purpose transport protocols in the tactical networks. This research is complementary to that focusing on compression, in that compression can be combined with any underlying transport protocol. They recognize that compression should be used, but they do not focus on the performance of the exact compression technique used; they use WBXML for compression without any justification for choosing that particular technique.

In [32], the focus is on dynamically replicating Web services in mobile ad-hoc networks. The authors recognize the need for reducing network traffic, but only for reducing power consumption in battery-powered devices. The reason for this is that they assume a WLAN network, which offers considerably higher bandwidth than a typical tactical mobile network. Furthermore, the approach for reducing network traffic is by reducing the number of transmissions. There is no focus on reducing the size of each transmission (i.e., compression), which is also necessary in low-bandwidth networks.

4.2.3 Reducing information overhead

Maintaining *information superiority*, which is *the capability to collect, process, and disseminate an uninterrupted flow of information while exploiting or denying an adversary's ability to do the same* [86], means that proper information management is critical. Ensuring that only relevant

information is transmitted over the network helps maintain information superiority, in that irrelevant information is not allowed to disrupt the information flow by overflowing the network. Content filtering can be used to alleviate network congestion by removing information that is not relevant to the user. Thus, it becomes important to identify the information that is indeed relevant, and transmit only that over the network.

Defence Research and Development Canada has performed a series of technical trials related to the dissemination of operationally important information in congested tactical radio subnets using their Low Bandwidth Test Bed [28]. Among the experiments performed is a test of dynamic reduction of network load by using content filtering techniques, as described in [51]. The experiment involved using an information management rule to determine whether or not to suppress replication messages. These messages contained a unit's report of its own position, and the rule used was based on how far the unit had moved since it previously reported its own position. Each node would use this rule to make an autonomous decision to either broadcast or suppress its own position at given time intervals. This means that the type of content filtering done in this experiment was a type of frequency based filtering, but the information management rules used were geographically based. The decision whether to perform filtering or not was made locally, which means that the required state could be maintained by each unit independently.

In [57] we discuss several types of information filtering, and suggest that filtering should be performed in the origin system (or in intermediate proxies) in order to limit network use.

Also, we have experimented with a combination of geographical and frequency filtering [54]. These techniques ensure that units always have the most recent information about units close to them, but get less frequent updates about units that are further away and thus outside their area of operations. This guarantees that only necessary and relevant information is sent over the network at any time.

Furthermore, colleagues at FFI have experimented with security label filtering in gateways which ensures that no data with a high classification can leak into a low classification domain [54].

4.2.4 Proxies

A *proxy* is *a node in the network between a client and a server through which the network traffic passes*. A proxy can be used for several purposes, such as caching, firewalling and content adaptation [110]. For example, HTTP proxy servers have been popular on the Internet for years, since they lower response times when surfing the Web. Web services proxies follow the same principle as HTTP proxies, in that they function as a "middle man" between the provider and the consumer of the service. However, they do not just understand the HTTP protocol, they must be able to recognize and process SOAP as well.

Agile Delta sells a proxy solution for Web services [1]. Their product allows proxies to be deployed in the network, and applies EFX compression to the messages as they pass between proxies. Thus, this product is a means for COTS (i.e., standard based and uncompressed) Web services consumers

and providers to gain the benefit of compression across the network. The client connects to a proxy, which compresses the request, forwards it to another proxy, which decompresses it and forwards it to the deployed service.

In [118], a solution employing proxy servers for disruption tolerant networking (DTN) is presented. The key concept of this work is to use proxies to translate from applications' native use of end-to-end protocols such as TCP and UDP to DTN messages that can traverse tactical networks despite their disruptive nature. Since DTN provides its own API, the application level proxies add transparent DTN support by leveraging the network level DTN API.

Faucher et al. [109] have created an experimental Web services proxy aiming to overcome the challenges associated with disadvantaged tactical networks. The authors recognize the need for proxy solutions, as tactical networks *are prone to frequent disruptions and high latency, causing applications to fail and producing an adverse effect on information transfer between the Web services and tactical applications*. The authors have conducted several experiments, using simulated network conditions and a scenario related to blue force tracking. The proxy handles SOAP sent over the HTTP protocol, limiting its use to applications based on SOAP over HTTP (i.e., it is tailored for Web services). The authors recognize the need for disruption tolerance, and discuss the fact that XML compression will enable the proxy to perform better in tactical networks (although they have not implemented this).

4.3 Adapting Web services for disadvantaged grids

The scarcity of resources at the tactical level means that it is vital to keep overhead at a minimum. This means that we need to adapt not only the amount of data we need to transmit (i.e., reduce the XML overhead), but also to use the transport protocol that is best suited (i.e., reduce the communication overhead).

As we have seen in the previous section, one can employ different means to reduce the overhead of Web services by:

- Reducing XML overhead with data compression.
- Reducing communication overhead by replacing the transport protocol.
- Reducing information overhead by optimizing the applications' need for information exchange.

The first two techniques can be applied to any Web service, but the latter technique needs to be considered for each and every service in turn. The latter technique should not be a part of the middleware, since the middleware should not be concerned with the application data. Thus, this functionality should be placed in the applications themselves or perhaps in proxies, since proxies

provide a convenient way of adding functionality between COTS clients and services. We apply these techniques to adapt Web services for use in disadvantaged grids, using NFFI as an example case.

4.3.1 Reducing information overhead: Removing optional fields

NFFI has some mandatory and a lot of optional fields (see Section 3.4.1). We remove all optional fields and keep only the mandatory fields of each track. The tracks contained in the NFFI message will, as a result of this removal of optional information, become very uniform (i.e., all the same XML tags are used in all the tracks) and only the data will differ. This makes the NFFI message as compression-friendly as possible, an important aspect for transmission in networks with low data rate.

The motivation for removing the optional fields is that they are unimportant in disadvantaged grids. For example, the experimental tactical soldier system we used in our experiments, NORMANS,[11] is not able to use these fields anyway. The NORMANS visualization is simple, being designed to run on a Windows CE PDA. As such, the NFFI messages contain more than enough information for the application to function after the optional fields have been removed (in fact, even some of the mandatory information in NFFI will not be visualized, since NORMANS makes a distinction only between friend and enemy units, and does not show the type of unit). For further details about the experiments with NORMANS, see our report [54].

4.3.2 Reducing XML overhead: Compression

Interoperability is a key challenge in NBD, so a standard based compression method is preferable. Since, as mentioned above, EFX is continually adapted to conform to the working drafts released by the EXI, we found it important to investigate EFX in the context of NFFI compression.

EFX can be used in one of two modes of operation; *generic* and *schema specific* compression. The generic option can compress any valid XML document without knowledge of the schema. The schema specific option must have access to the XML schema when it performs compression and decompression, thereby sacrificing generality for an increase in compression rate. We used the generic option in our experiments, enabling us to compare EFX directly with XMLPPM (which provides only non-schema specific XML compression). When evaluating the efficiency of the algorithms, we focused on compression results and not resource usage during compression (memory and CPU usage). The reason for this is that for our intended use, i.e., in tactical networks, the bandwidth is the most limiting resource. Power consumption is also an issue when using battery powered communication equipment. However, we have shown in an earlier study [56] that the difference in computation time between the various techniques is in the millisecond range.

[11] FFI develops concept, requirements and technology for the future network enabled soldier. An overview of the Norwegian Modular Network Soldier (NORMANS) is given in [80], and the C2I system is presented in [42].

Transmission of data also requires power, and by using a compression technique with a high compression ratio, we can reduce the transmission time by several seconds. Typically, more power is consumed during the transmission of packets than the reception or during "listening" periods [52]. Since radio transmission consumes a lot more power than local computation, the reduction in power consumption caused by reduced transmission time greatly outweighs the benefits of saving a few milliseconds when performing compression and decompression.

Details of the evaluation are given in [65], but overall EFX is the "winner" followed by GZIP as the second best. A recent study from Poland [105] which repeats our experiments substituting Agile Delta's EFX with EXIficient,[12] the open source version of Efficient XML, supports our findings.

4.3.3 Reducing communication overhead: SOAP over STANAG 4406 Annex E

Data-rate constraints in tactical networks impose great challenges that have to be solved in order to fully utilize Web services technology. On the Internet, Web services use the XML-based SOAP protocol over HTTP and TCP for information exchange. However, properties of these protocols make them unsuited for use in some disadvantaged grids.

In order to allow for use of services on different operational levels, information needs to traverse heterogeneous networks with different characteristics. This requires a message based transport system with optimized protocols and store-and-forward capabilities. Our suggestion is that one should consider replacing HTTP/TCP with the Military Message Handling System (MMHS) implementing STANAG 4406. MMHS has both specially designed tactical protocol profiles and store-and-forward capabilities. It is already present, or in the process of being implemented, in many tactical military communication systems, and using an already existing messaging system such as MMHS can potentially reduce the time needed to deploy Web services based solutions in tactical networks.

STANAG 4406

In NATO, Formal Military Messaging is standardized in STANAG 4406 edition 2 (S4406). The MMHS is responsible for the delivery, formal audit, archiving, numbering, release, emission, security, and distribution of received formal messages. In NATO, the formal messaging service is seen as the vehicle for secure mission-critical operational, military applications. It is the only agreed standard to achieve interoperability between the formal messaging systems of NATO nations. Systems compatible with the S4406 standard have been and are being implemented widely by the NATO nations and by the NATO organization. S4406 encompasses several annexes, where Annex C and E cover the communication protocols.

[12] EXIficient is an open source implementation of the W3C Efficient XML Interchange (EXI) format specification written in the Java programming language. It is available for download at "http://exificient.sourceforge.net/".

Protocol profile	Domain	Message overhead	Change in transmission directions per message transmission
STANAG 4406 Annex C	Strategic	2700 bytes	8
STANAG 4406 Annex E TMI-1	Tactical	700 bytes	2
STANAG 4406 Annex E TMI-4	Tactical	20 bytes	0 (1 using the retransmission option)

Table 4.1: Overhead and change in transmission directions for the different S4406 protocol profiles.

S4406 defines three protocol profiles adapted to different communication networks. The original connection oriented protocol stack defined in S4406 Annex C was developed for strategic high data rate networks, and is not suitable for channels with low data rate and high delays. The protocol profiles TMI-1 and TMI-4 have therefore been developed for use between Message Transfer Agents (MTAs) over disadvantaged grids. With the inclusion of these protocol profiles in Annex E of S4406, a common baseline protocol solution exists that opens for the use of MMHS in both the strategic and tactical environments.

Table 4.1 shows, for each of the three protocol profiles, approximate overhead in bytes per message together with the number of changes in transmission directions during one message transmission.

In addition to military messaging, MMHS may also be used as an infrastructure for interconnection of other applications, including Web services, by use of a standardized API. In this perspective, the MMHS can be viewed as a replacement for HTTP/TCP that can enable the use of Web services in communication systems with different quality and data rate. The benefits of using MMHS in this way can be summarized as follows:

- Reuse of an already established messaging infrastructure in NATO and the NATO nations.
- Three different protocol profiles that enable tailoring of the transport system to the communication networks (simplex, half duplex or duplex). The Annex E protocol profiles are also very bandwidth efficient. These tactical protocols also support nodes in radio silence, and networks with low MTUs.
- Support for both reliable and unreliable transmission modes.
- An asynchronous store-and-forward system able to traverse different communication networks.
- Support for priority and preemption mechanisms for handling time critical information.
- Support for both multicast and unicast of messages.

A key component in MMHS is the MTA, which is a switch in the message transfer system. This switch provides store-and-forward functionality, and may be used as a gateway between strategic and tactical messaging systems. The MTA may have a triple protocol stack implementing both the

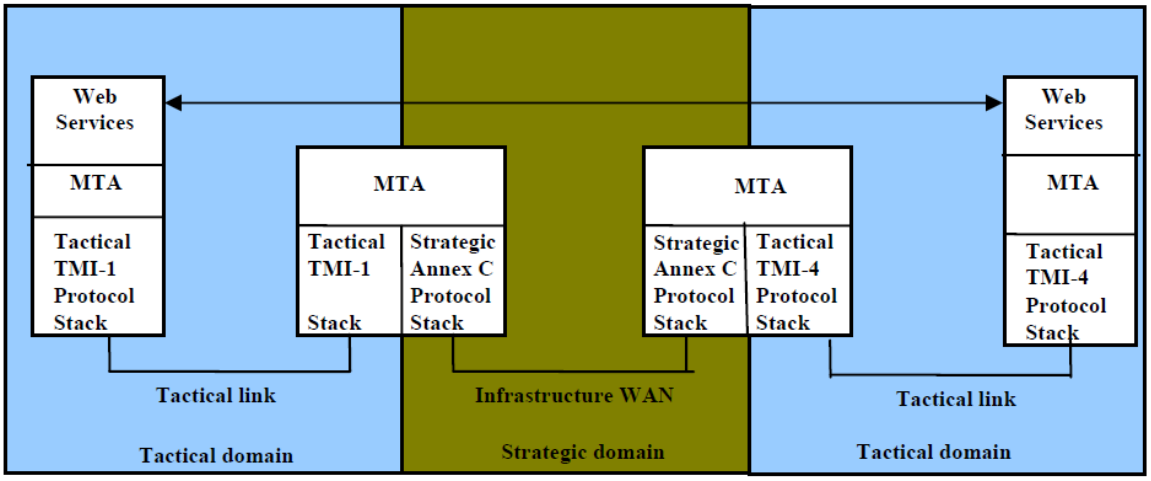


Figure 4.2: Seamless interconnection of Web services over heterogeneous communication networks by using MMHS as an overlay network (from our paper [66]).

strategic connection oriented protocol profile and the two tactical protocol profiles, and can therefore route messages between infrastructure WANs and low data rate tactical links (see Figure 4.2). When using the MMHS for transfer of SOAP messages, the MTAs and the Web services functionality are integrated, which implies that there is no additional delay for checking or connecting to a message store.

Evaluation

As a part of our interoperability experiments at NATO CWID, (see our report [54]) we tested and evaluated the use of MMHS as a carrier for Web services traffic. MMHS has many of the qualities that are needed to ensure delivery of messages between strategic and tactical communication systems, and has the benefit of being able to reuse an existing infrastructure for a new purpose.

Our goal in performing these tests was twofold; we wanted to evaluate the use of store-and-forward in general, and MMHS specifically. We also wanted to compare and evaluate the two tactical protocol profiles of S4406, TMI-1 and TMI-4, in order to investigate how the differences in overhead and directional changes affect transmission delay.

We compared the two tactical profiles of S4406, TMI-1 and TMI-4, in order to establish the efficiency of the two profiles. For the measurements, we transferred documents containing NFFI-tracks, and we compressed the documents using efficient XML with built-in compression enabled. In order to simulate a disadvantaged grid, we used NIST Net configured with a bandwidth of 2.4 Kbps.

In Figure 4.3, we show the results of our experiments. The graph shows the overall average transfer time, the average transfer time from NORMANS Advanced to HQ, and the average transfer time

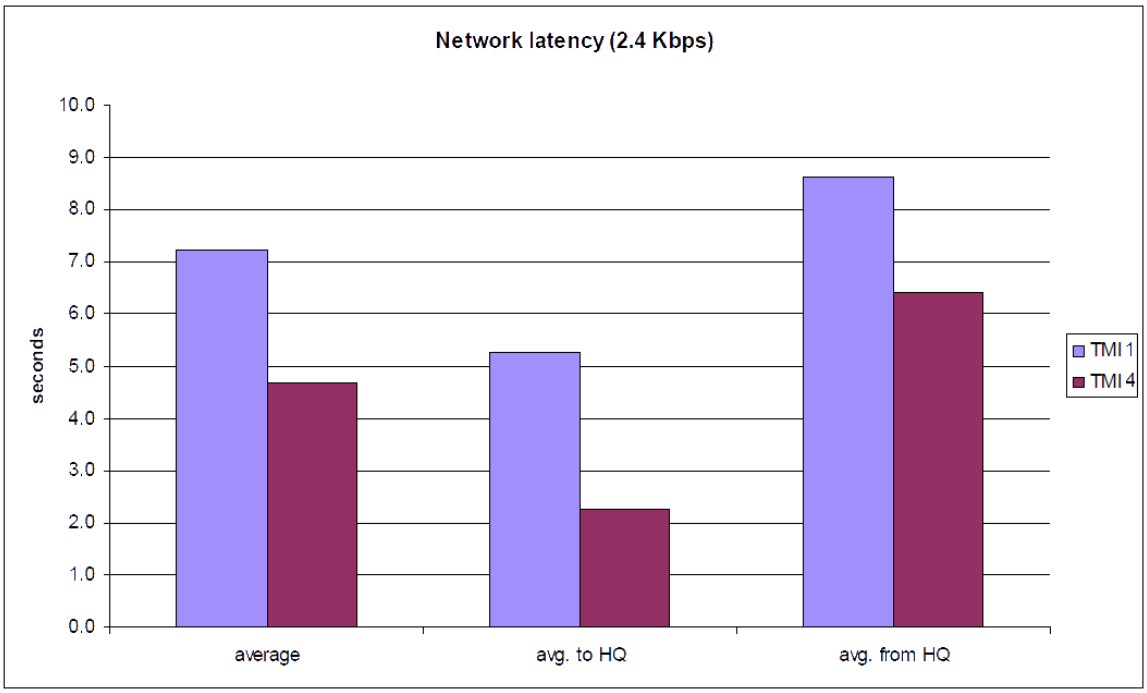


Figure 4.3: Message transmission delays.

from HQ to NORMANS Advanced. It should be noted that the documents sent to HQ contained only one NFFI-track (the soldier's position), while the documents sent from HQ to NORMANS Advanced contained 20 to 25 NFFI-tracks each. Thus, the graph clearly shows the effect of compression; sending 20 tracks only takes about twice the time of sending one track.

It should be noted that the implementation of MMHS we used, XOmail from Thales, refers to TMI-1 as "TMI" and TMI-4 as "Direct Messaging Profile" or "DMP". When comparing the bars for TMI and DMP, it is clear that TMI has considerably more overhead than DMP. This is particularly noticeable for small documents (from NORMANS Advanced to HQ), since there is less data over which to amortize the overhead.

Through our tests we have:

- Confirmed our hyporeport that MMHS can be used as a replacement carrier for Web services.
- Shown that using MMHS avoids the time-out problems that arise when using standard HTTP over TCP in tactical networks.
- Introduced the ability to use Web services even when communication links fail temporarily.

Furthermore, we were able to show that an existing proprietary service can be adapted to a Web services environment; and that by taking the necessary measures, Web services can be used in disadvantaged grids similar to the one we emulated (e.g., MRR).

4.3.4 Proxies

At CWID 2007 we implemented a special purpose NFFI proxy that employed all of the above mentioned techniques. This proxy filtered NFFI, compressed it with EFX using Agile Delta's ZIP, and transmitted it across an emulated disadvantaged grid using SOAP over the STANAG 4406 annex E protocols [54]. It is important to combine techniques in order to maximize the total optimization benefits. For example, the systems that needed the unfiltered NFFI tracks since they needed all the information were unable to get the high compression rates that we did. Typically, the compression rates were between 30 and 50 percent of the original document when using just compression. To achieve better compression results than that, making the NFFI messages more uniform through content filtering was necessary.

Since that experiment colleagues at FFI have continued to develop proxy solutions (see our paper [58] for a discussion of proxy techniques). The current implementation supports compression and multiple transport protocols in a transparent way, and adds delay and disruption tolerance to SOAP. This prototype solution, which we call a *Delay and Disruption Tolerant SOAP Proxy* (DSProxy) [123, 124], is intended for use across heterogeneous military networks, and has been used in an operational experiment where it enabled us to use Web services in an actual disadvantaged grid [82].

4.4 Claim review

In Section 3.4.2, we presented an operational experiment and discussed how we had used Web services in a CESMO operation. In that experiment we noticed that COTS Web services cannot always be reliably invoked in military tactical networks due to their inherent characteristics. Based on these characteristics, we summarized a set of requirements (see Section 3.5) that must be met in military networks. Now we discuss those requirements (from Table 3.1) in the context of the techniques investigated in this chapter.

The first premise was that *a solution must support a federation of systems.* Provided we implement our suggested optimizations in proxies, then the solutions can be employed in a federation of systems. A proxy placed at a network interoperability point can consume COTS Web services calls, and perform optimizations necessary for the network behind it.

The second premise was that a solution *should use Web services technology*. In this chapter we have considered existing Web services, and how we could best adapt the existing solutions to work in tactical networks.

To ease interoperability we should *always use standards when possible*. In this chapter we presented optimizations based on standards (e.g., compression using EFX, using a NATO standardized tactical transport protocol). Furthermore, we suggested implementing the optimizations in proxies in order to remain compatible with COTS (i.e., unmodified) Web services clients and servers.

The set of requirements specific to tactical networks were support for: *Radio silence, limited bandwidth, disruptions,* and *low MTUs*. Also, one *should not rely on TCP or IP fragments*. In this chapter we presented content filtering and compression to better cope with limited bandwidth. Furthermore, by employing a tactical protocol (e.g., TMI-1) rather than HTTP/TCP to convey SOAP messages, we get a transport layer that can function with nodes in radio silence and low MTUs. Finally, STANAG 4406 implements store-and-forward functionality, which allows Web services to function even in networks with disruptions.

In the previous chapter, we saw that COTS Web services do not necessarily function in tactical networks. In this chapter we have shown that it is possible to use Web services in tactical networks, provided a set of optimization techniques is used. Thus, we have proved claim #1, that *to be able to invoke Web services in military networks with constraints, adapting the standards is needed.*

4.5 Summary

In this chapter we have addressed claim #1. We have investigated how to adapt Web services in order to make them function in military networks with constraints, while at the same time fulfilling all the requirements from the requirements analysis in the previous chapter.

Using overhead reducing techniques such as content filtering, data compression, and efficient transport protocols allows us to employ Web services in networks with severely limited data rates. Adding store-and-forward capabilities to the transport system allows us to overcome temporary network disruptions as well, thus giving us the possibility of using Web services technology in military tactical networks. However, modifying or adapting the Web services breaks the compliance with the standards. By employing proxy servers, we can limit the proprietary modifications to the inter-proxy communication, whereas the clients and servers still can be implemented using COTS tools and be completely compatible with the Web services standards.

Being able to invoke Web services across heterogeneous networks is a necessity if one is to use Web services as a middleware technology. However, in dynamic environments such as tactical networks, being able to invoke services we know about is not enough. In cases where a network is unstable, so that services may come and go for any number of reasons, then we cannot hard-code the service addresses in the applications. If a service we want to use becomes unavailable, then the application depending on it will not be able to perform its task. If, however, we can discover another service that can do the same work, then we can use that service instead. Service discovery issues are covered in the next chapter.

5 Discovering Web services

In this chapter we address claim #2, that *networks with different properties require different discovery mechanisms*, and investigate Web services discovery in the context of military networks. We concern ourselves with disseminating the information about services in the different networks (i.e., service advertisements), and querying for available services (i.e., service discovery).

In previous chapters, we have seen that military networks are heterogeneous and complex, but that we can employ several techniques that will allow us to invoke Web services across such networks provided we know their invocation address. In static networks, this address can be hard-coded in the applications, since the address is not expected to change (i.e., design-time service discovery). However, in dynamic networks, services can come and go. Since tactical mobile networks are highly dynamic, there is a need to perform dynamic service discovery (i.e., run-time service discovery). This means that our clients and services are still implemented by the usual Web services standards, but that we need to be able to perform the advertisement of new services and look them up at run-time. This allows us to discover the necessary service addresses as they change, and invoke the services where and when they are available.

In this chapter we discuss the Web services discovery standards. We pay special attention to WS-Discovery, which is particularly interesting since it is decentralized and thus without a single point of failure — a property that is desirable in military networks. We evaluate WS-Discovery, and point out drawbacks associated with that protocol. Furthermore, we explore related work, and discuss different ways of categorizing service discovery protocols. We identify a set of properties that should be supported by a service discovery solution for disadvantaged grids. Finally, we develop and evaluate an experimental service discovery protocol that we have designed with these properties in mind.

5.1 Introduction to service discovery

The term *service discovery* denotes the act of discovering services available for consumption. The purpose of doing service discovery can range from finding devices such as printers on a LAN, to finding trading services such as e-commerce providers on the World Wide Web (WWW).

To discover services, one can choose from different mechanisms, ranging from a network location provided by out-of-band means, via a decentralized peer-to-peer (P2P) based registry, to a centralized registry. In this report, we use the term service *advertisement* to denote the *service description* itself. An important part of the service advertisement is the service contract, which includes the interface of the service. By specifying services at a coarse granularity, giving room for change in the way services are implemented without affecting the external interfaces, one can achieve loose coupling. It enables reuse of services as building blocks in systems not even planned yet, thus allowing for a more flexible architecture. This is the concept behind Web services, where a

WSDL provides the service description (see Section 3.2.3). The use of a service registry facilitates late binding between consumers and services, which also contributes to loose coupling.

5.1.1 Design-time vs run-time discovery

Service discovery and selection can be performed at two instances in time, *design-time* or *run-time*. By design-time service discovery and selection, we mean a client software developer who browses for the desired services to support in the client software, or an administrator looking for the service address to configure at deployment. Web services client software typically comes with a configuration file where the endpoint address can be set manually (e.g., .NET applications), or they access the entire WSDL from which the endpoint is parsed at invocation time (e.g., Java applications). In the case of design-time discovery, the service endpoint is set once in either configuration file or WSDL, and it does not change during subsequent invocations unless an administrator changes the address at some later point, for example when deploying in a different network, etc. This is the least complex case, because there is a human in the loop to read and interpret the descriptions of the services' semantics.

It is run-time service discovery that is our main concern in this report, because in a dynamic environment, such as a military MANET, we can have services that come and go. Run-time service discovery can be divided in two cases: The most common is the case where a client is designed to consume services of a certain interface, and can only consume services that provide this interface. Since we are concerned with discovering Web services in this report, we limit ourselves to the case of finding services that are specifically supported by our clients. This means that while all the static metadata in the WSDL are valid the entire time, the *address location* of the service may change (see Figure 5.1). This means that in dynamic networks, we should be able to discover the current state of the network and find the current addresses of all available services.

A more complicated case is when new services, previously unknown to the client, are encountered. Such behavior demands that the computer is able to reason about discovered service descriptions in order to both select a proper service and to select a service that it is able to invoke. This goes beyond what can be achieved with WSDLs. It requires the use of machine-processable semantics. This is beyond the scope of this report, and is left for future work (see Section 7.2.5).

5.1.2 Service discovery in dynamic environments

In traditional networks, and in the Internet community, a lot of work has been done in terms of service discovery. A number of different mechanisms have been proposed and implemented for local area networks (LANs) and wide area networks (WANs). However, fundamental differences in terms of computing resources, network bandwidth, and issues such as mobility and stability in the network environment make these generic service discovery techniques unsuitable for tactical networks. A service discovery solution for military networks must adhere to the requirements we

```
<?xml version="1.0"?>
<definitions name="PositionUpdate"
             targetNamespace="http://example.com/PositionUpdate.wsdl"
             xmlns:tns="http://example.com/PositionUpdate.wsdl"
             xmlns:xsd1="http://example.com/PositionUpdate.xsd"
             xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
             xmlns="http://schemas.xmlsoap.org/wsdl/">
  <types>
    ...
  </types>

  <message name="GetLastPositionInput">
    <part name="body" element="xsd1:PositionRequest"/>
  </message>
  <message name="GetLastPositionOutput">
    <part name="body" element="xsd1:Position"/>
  </message>

  <portType name="PositionUpdatePortType">
    <operation name="GetLastPosition">
      <input message="tns:GetLastPositionInput"/>
      <output message="tns:GetLastPositionOutput"/>
    </operation>
  </portType>

  <binding name="PositionUpdateSoapBinding" type="tns:PositionUpdatePortType">
    <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
    <operation name="GetLastPosition">
      <soap:operation soapAction="http://example.com/GetLastPosition"/>
      <input>
        <soap:body use="literal"/>
      </input>
      <output>
        <soap:body use="literal"/>
      </output>
    </operation>
  </binding>

  <service name="PositionUpdateService">
    <port name="PositionUpdatePort" binding="tns:PositionUpdateSoapBinding">
      <soap:address location="http://example.com/PositionUpdate"/>
    </port>
  </service>

</definitions>
```

Figure 5.1: This is an example WSDL from a simple Web service we have created. A WSDL definition has two parts: The abstract description *— i.e., static metadata — covers the service interface, including the methods that it supports, the input parameters and return types. The* concrete implementation *— i.e., address location — defines how to physically bind to the service and use its supported operations (highlighted). The whole WSDL is needed for design-time discovery, whereas finding the current address location is the main concern in run-time discovery.*

discussed in Section 3.5. Keep in mind that the use of standards is mandated where it is possible and reasonable to use them, even if that means that there is no unified solution.

Previous work at FFI has addressed service discovery to some extent, and we summarize the necessary properties for service discovery identified by those studies below. These properties can be seen as a specialization of parts of our requirements analysis, and we use these properties when we evaluate the Web services standards. Gagnes [46] specifies four high-level properties for a discovery infrastructure for dynamic environments that we summarize briefly:

To ensure discovery of the services available, robustness and survivability against registry failure or disappearance is important. This means that the system cannot depend on centralized components like a single registry.

Service discovery should work in environments disconnected from the Internet (e.g., DNS and WWW). Additional artifacts needed by clients to evaluate or use services (e.g., XML schema) must be obtained elsewhere. Such functionality could be provided by the discovery service.

The discovery infrastructure must provide a fresh view of available services. Responses to queries should mirror the current state in the network and should not advertise services that are no longer present in the network. This is known as *liveness* information.

The infrastructure should support different kinds of service description mechanisms, ranging from simple (e.g., name, id, and URI specifying a pre-agreed service type) to rich (e.g., semantic descriptions). Thus, both normal Web services (see Chapter 3) as well as Semantic Web services (see Section 7.2.5) should be able to use this infrastructure.

The issues described by Gagnes are generic properties that should be addressed by any service discovery framework in a dynamic environment. Flathagen has identified additional properties that must be considered for service discovery in tactical mobile networks [41]:

Low bit rate increases the demand for specially tailored protocols and cross-layer optimizations (see Section 5.2.1). Proprietary protocols and solutions are therefore more frequent in tactical networks. Also, low bit rate increases the importance of compression and other techniques that reduce communication overhead. However, the compatibility with legacy protocols and equipment must be taken into consideration.

Summarizing the properties

We have identified several key architectural properties necessary for service discovery in dynamic environments. All in all, the solution should:

- Be based on standards, if possible.
- Be able to discover services in a LAN, WAN, and/or military MANET.

- Mirror the network state (i.e., a query should return only available services).
- Be robust.
- Support service descriptions that are rich enough to support run-time discovery (and preferably design-time discovery) of Web services.
- Ideally have no DNS and WWW dependency. This is important for military networks, which must function disconnected from the Internet.

This set of properties along with the requirements from Section 3.5 must be fulfilled within each operational level. However, there is also a need for interoperability across different networks. This need has not been addressed by Gagnes and Flathagen. In this chapter we focus on discovery for each network separately, whereas interoperability issues (i.e., pervasive discovery) is covered in the following chapter.

5.1.3 Web services discovery standards

There are three standards related to service discovery, all by OASIS: Universal Description, Discovery and Integration (UDDI), electronic business using XML (ebXML), and WS-Dynamic Discovery (WS-Discovery). UDDI is the oldest and most widely known, which is why it is almost always mentioned when one talks about service discovery in the context of Web services. So far, it is the only discovery related standard that is addressed by the WS-I. This means that UDDI is also the most mature and widely supported of these standards.

UDDI

UDDI [92] allows service providers to register their services and service consumers to discover these services both at design-time and run-time. In principle, UDDI is centralized, but mechanisms for federating several registries have also been specified in newer versions of the specification. Having multiple registries, or letting a registry consist of several nodes that replicate data, increases the robustness of the discovery solution. In UDDI, replication between registry nodes must be configured manually. It is also possible to let several separate UDDI registries exist independently of each other, but information will not be replicated unless a custom scheme is designed. Additionally, a hierarchical model may be used, using a root registry and affiliate registries. In this case, a root registry must be chosen, and affiliate registries may be defined as child registries of the root registry. This must be done to avoid duplicate identifiers, or keys. A replication scheme for intra-registry replication between nodes is defined, which allows for fault tolerance. The replication topology must be configured.

UDDI supports rich service descriptions, and one can find services by name, type, binding and according to a taxonomy. UDDI provides a flexible model in that specific service types can be

registered with the registry and referenced by service instances that implement the service type. This is called a technical model, or tModel, in the UDDI information model. A tModel can include pointers to further description of a service, such as a WSDL description and bindings. Since UDDI is designed to be general, OASIS describes ways to map WSDL documents and also BPEL4WS abstract processes to the tModel fields of the UDDI registry. Especially the former facilitates a number of interesting queries, where searches can be based on WSDL port types and bindings, that is, the signature of the service. This is very important for run-time selection of services.

The tModels give flexibility, but that is also one of the drawbacks with UDDI, as proprietary use of the field can occur. Many solutions use the tModels to store such proprietary information, for example about QoS issues [87]. However, if such proprietary solutions are to work, all publishers and consumers using the registry must understand the information in the tModel and know how to handle it. Another limitation is that tModels are not stored in UDDI registries themselves. A unique identifier referencing a tModel is contained in the registries, and you need a separate repository to store the actual data in.

The UDDI registry supports reconfiguration as long as services do not go down unexpectedly. If so, advertisements will be in the registry forever because there is no liveness information in the current versions of UDDI. UDDI does not include a repository mechanism. It can only hold URIs pointing to content which has to be stored somewhere else, such as on the WWW.

Since UDDI is the most mature of the Web services registry standards, several vendors offer UDDI implementations. Some support the newest specification UDDI version 3, whereas others implement the older UDDI version 2. There also exist several open source implementations that implement UDDI. For example, Apache jUDDI, which is a Java implementation of UDDI v3. It is freely available from http://ws.apache.org/juddi/.

ebXML

Another service discovery standard is ebXML [99]. It is a collection of specifications for conducting business-to-business integration over the Web. It allows registering services in a similar way as UDDI according to its own registry specification. The ebXML registry also defines inter-registry interaction, or cooperation between registries, a so-called federation of registries. Note that an ebXML federation is different from that of a UDDI federation, because ebXML supports a non-hierarchical multi-registry topology. Here, each registry has the same role, and registries may join or leave a federation at any time. This allows flexible deployment. Federated queries are supported, enabling query forwarding to other registries without the need to replicate data first. Since the federation model is P2P based, there is no single point of failure. The ebXML registry is meant to support both discovery and business collaboration, as opposed to UDDI, which mainly targets discovery.

The ebXML registry information model is similar to that of UDDI but somewhat more flexible. Business capabilities such as processes and services can be published in the registry. It is possible

to use taxonomies to classify the registered items. Support for rich service descriptions in ebXML is currently very similar to that of UDDI. Both support finding services by name, type, binding and according to a taxonomy. However, ebXML supports more advanced (i.e., SQL based) queries.

Just like UDDI, ebXML has issues with liveness, in that it supports reconfiguration as long as services do not go down unexpectedly. Unlike UDDI, the ebXML registry can store vocabularies like XML Schema or ontologies since it also specifies a repository for such items.

This standard is less mature than UDDI and is not as widely supported by industry as UDDI is. There exists an open source reference implementation of ebXML 3, which is the most recent specification. That implementation, called *Omar*, is freely downloadable from http://sourceforge.net/projects/ebxmlrr/files/.

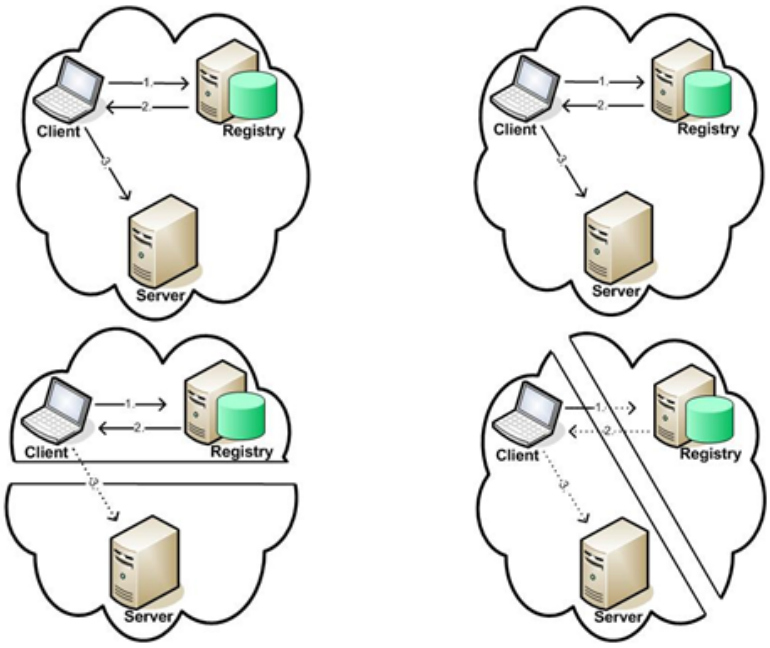
WS-Discovery

WS-Discovery is a standardized Web services discovery mechanism. After being a draft since 2005 [117], it became a standard in 2009 [96]. This makes WS-Discovery the most recent of the Web services discovery standards, meaning that there are few (if any) implementations available that adhere to the standard. For example, the implementation of WS-Discovery in Windows Vista pre-dates the standard, and is thus only draft compliant.

An open source implementation of WS-Discovery written in Java is available from http://code.google.com/p/java-ws-discovery/, and it too supported only the draft standard at the time of writing this report. Thus, for our experiments, we based our code on the draft standard of WS-Discovery. The differences from draft to standard are minor, involving changes in some attributes in some XML schemas, and also changing the required number of messages to send per query. In the experiments presented below with the open source implementation of WS-Discovery, we modified the messaging frequency to match that of the standard. Thus, our results should be comparable to that of the standard when an implementation of that becomes available.

WS-Discovery is based on local-scoped multicast, using SOAP over UDP [100] as the advertisement transport protocol. Query messages are called probe messages. Services in the network evaluate probes, and respond if they can match them. To ease the burden on the network, WS-Discovery specifies a *discovery proxy* (DP) that can be used instead of multicast (e.g., if a registry such as UDDI or ebXML is present, it should be used instead). This means that WS-Discovery can run in two modes, depending on whether there is a DP available or not. However, this DP is not well-defined in the standard. The standard fully describes the decentralized operation of WS-Discovery, but the functionality of (and integration with) the DP is left to be implemented in a proprietary manner for now.

WS-Discovery is suited for service discovery in a LAN only, since it is based on local scoped multicast. However, if a DP is present, then WS-Discovery enables you to find that DP in your LAN. That DP can in turn allow you to find services in the WAN. WS-Discovery, when operating



(a) The *liveness* problem. (b) The *availability* problem.

Figure 5.2: Issues in MANETs when network partitioning occurs.

without a DP, is fully decentralized. However, when a DP is present, it should be used instead, and then the robustness of the DP solution is important. As none of the registry standards take liveness into account, WS-Discovery will only accurately reflect the service network in its decentralized mode.

With WS-Discovery, service matching is based mainly on the WSDL port type supported by the service and administrative scope. The port type is described by a namespace URI, and some scope limitation can be done through a simple filter. Since WS-Discovery cannot provide a complete WSDL, it is not suited for design-time discovery. WS-Discovery has no repository mechanism, and information referenced in discovered descriptions (i.e., WSDL includes and XML schema includes) need to be fetched from somewhere else.

5.1.4 Evaluating the standards

In this section we evaluate the standards with respect to disadvantaged grids. The standards function well in WANs and LANs. Currently, they are deployed in many such networks for civil applications around the world. Thus, we should be able to employ the standards within strategic networks and deployed tactical networks, which leverage the same technology as civil networks. In the tactical mobile networks, however, they are not necessarily suitable, as we argue below.

The registries

Registries were created for use in large, fixed infrastructure networks. They are not suitable for use in MANETs, due to the dynamic nature of such networks. MANETs, being characterized by mobile units and unstable links, are prone to network partitioning where not all nodes can communicate with each other all the time. In such networks, you can encounter problems with service liveness and service availability:

- The liveness problem (see Figure 5.2(a)) occurs when a service has been published in a registry, but the service has become unavailable. In this case, a client can still look up the service in the registry, but the service cannot be reached. No matter how many times the service discovery is performed — the result is still the same. This occurs because the standardized Web services registries require that you actively register and de-register services to keep them up-to-date.

- The availability problem (see Figure 5.2(b)) occurs when the registry is in a different network partition from that of the client. If this occurs, then the client will be unable to look up any services at all, since it cannot contact the registry. Thus, even if the service the client wants to use is actually present and available in the same network partition, there is no way of discovering and using it.

This means that in dynamic networks where partitions can occur, such as in tactical mobile networks, one should preferably use service discovery mechanisms that address these issues. Tactical mobile networks usually contain a few but highly mobile participating nodes. This means that it is feasible to use fully decentralized service discovery mechanisms in such networks. A fully decentralized mechanism addresses the availability problem by distributing the same information about services to all the nodes that it can reach. If the mechanism is coupled with a lease mechanism or just lets service advertisements time out from its cache, then it can also address the liveness problem in that there is no need to actively de-register unavailable services any more — the mechanism removes such stale information itself.

Thus, existing Web services discovery standards are insufficient in disadvantaged grids. The basic, standardized components of UDDI are lacking some important features, most notably liveness information. However, UDDI can be expanded with proprietary functionality in a way that such features become available. Such expansion could make UDDI better suited to NBD. However, the prototype for the NATO metadata registry and repository [76] is built on ebXML. This means that there is an incentive in NATO for choosing ebXML in preference to UDDI, despite UDDI being the most mature standard of the two. This can be attributed to ebXML's superior federation capabilities (i.e., federated queries vs UDDI's replication), and its more flexible query support. Thus, in this report, we utilize ebXML as the registry of choice in our experiments. For further considerations about using UDDI in NBD, see [46].

Minimum size	Maximum size	Average	Std.dev.	Median
1643	149342	13830	19202	8514

Table 5.1: Sizes in bytes for our 100 WSDLs.

WS-Discovery

WS-Discovery is intended for use in LANs. Its discovery scope is limited to a single network hop, and services are required to send UDP multicast *HELLO* messages that advertise when they become available. Also, services should send UDP multicast *BYE* messages when they go away. If services are able to do this, then each node will have an up-to-date view of the available services in a LAN. In a dynamic network, we cannot rely on receiving such messages. Thus, we do not evaluate that part of WS-Discovery. It is also possible to actively query the network by sending *PROBE* messages. In order to accurately mirror the current network state of a dynamic network probing must be used, and no DP must be present, in which case each node replies with UDP unicast *PROBE MATCH* messages. This generates a lot of data packets, which is undesirable in a disadvantaged grid. We evaluate the open source WS-Discovery implementation release version 0.2.0 with message frequencies changed to match that of the standard: The draft requires all multicast packets to be sent four times, and all unicast packets to be sent two times. The standard has cut resource use in half. It requires all multicast packets (i.e., *HELLO*, *PROBE*, and *BYE* messages) to be sent twice, and the unicast *PROBE MATCH* messages to be sent once.

We evaluated WS-Discovery using WSDLs for services such as finance, news, weather services, etc. These WSDLs were fetched from http://www.webservicex.net/ and http://www.webservicelist.com/, which provide lists of freely available Web services. Also, the WSDLs from Google and Amazon's search services were included, yielding a set of 100 WSDLs in total. This provided us with a representative set of interfaces (see Table 5.1) for a wide array of Web services which we could use in our evaluation.

WS-Discovery is based on a query-response model, where a multicast query (probe) triggers unicast responses (probe match). The load incurred on the network by the number of querying nodes (q) in a network with a total number of n nodes can be calculated using the formula in Equation 5.1. If all nodes should have an up-to-date view of the currently available services, then $q = n$, conversely, if only one node is querying, then $q = 1$.

$$LOAD_{WSD} = (sizeof(probe) + sizeof(probe\ match) * (n - 1)) * q \tag{5.1}$$

Minimum size	Maximum size	Average size	Std.dev.	median
990	1070	1017	17.08	1013

Table 5.2: HELLO message payload statistics (in bytes), calculated from HELLO messages corresponding to the Web services described by our 100 test WSDLs. For each Web service that is published, WS-Discovery sends two identical such messages.

Number of services	PROBE MATCH message size
1	1175
10	5153
20	9459
40	18188
60	26879
80	35589
100	44817

Table 5.3: The size (in bytes) of the unicast PROBE MATCH message payload sent by a node publishing a certain number of Web services with WS-Discovery.

In our tests[13] the *HELLO* messages yielded different sizes (see Table 5.2) depending on the different WSDLs that were published (two HELLO messages generated per WSDL). A *PROBE* message was always 597 bytes. According to the standard it had to be transmitted twice, meaning that $sizeof(probe) = 2 * 597\ bytes$. The *PROBE MATCH* varied in size with the number of services published, since it contained all the services published by a node. Table 5.3 shows the different sizes of *PROBE MATCH* messages sent by a node with 1 to 100 services published. We see that publishing just one service incurs a lot of overhead (1175 bytes to disseminate information about it), whereas for a larger number of services this overhead is reduced (more actual Web services porttype information in the response compared to SOAP headers, etc). Calculating the average when publishing multiple services (i.e., the average of the message sizes divided by the number of services) yields 464 bytes. UDP can carry a payload of 65507 bytes, meaning that WS-Discovery has a theoretical upper limit of publishing approximately $65507/464 \approx 141$ Web services when considering results from our 100 WSDLs. Naturally, the number is approximate, because in practice varying namespace lengths in different WSDLs can affect the PROBE MATCH size.

Using Equation 5.1, we fill in values for $sizeof(probe\ match)$ using values from Table 5.3, as well as the above mentioned $sizeof(probe)$. The number of nodes in the network, n, is varied from 1 to 250. First, we set $q = 1$, meaning that only one node is querying. Figure 5.3 illustrates WS-Discovery's resource use (in megabytes) in this case when one node is querying in networks with

[13] We used Wireshark version 1.2.1 for Windows from http://www.wireshark.org/ to capture data traffic between a small network with two nodes. This enabled us to capture actual WS-Discovery traffic, and examine the packet payload sizes, thus giving a foundation for further analytical results.

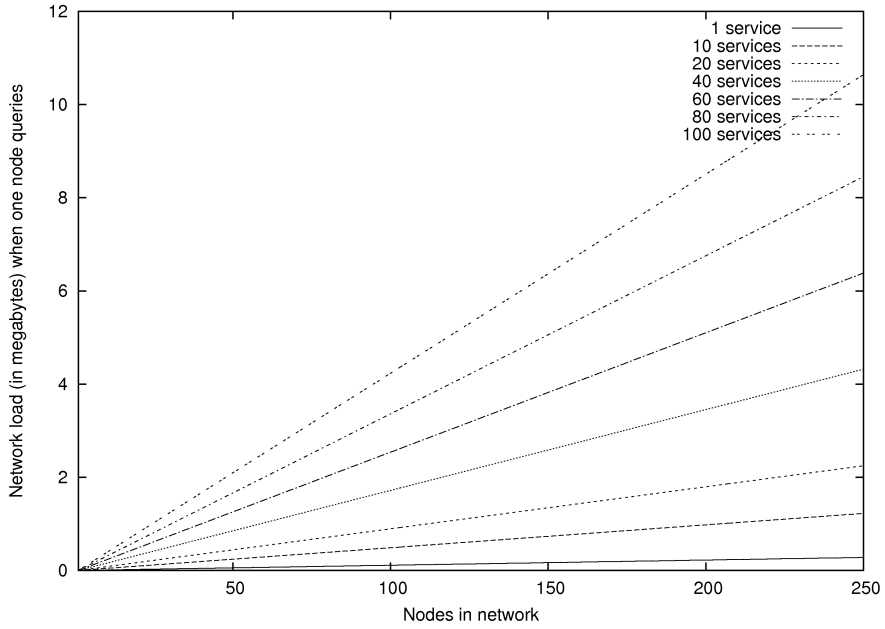


Figure 5.3: WS-Discovery's resource use when one node queries.

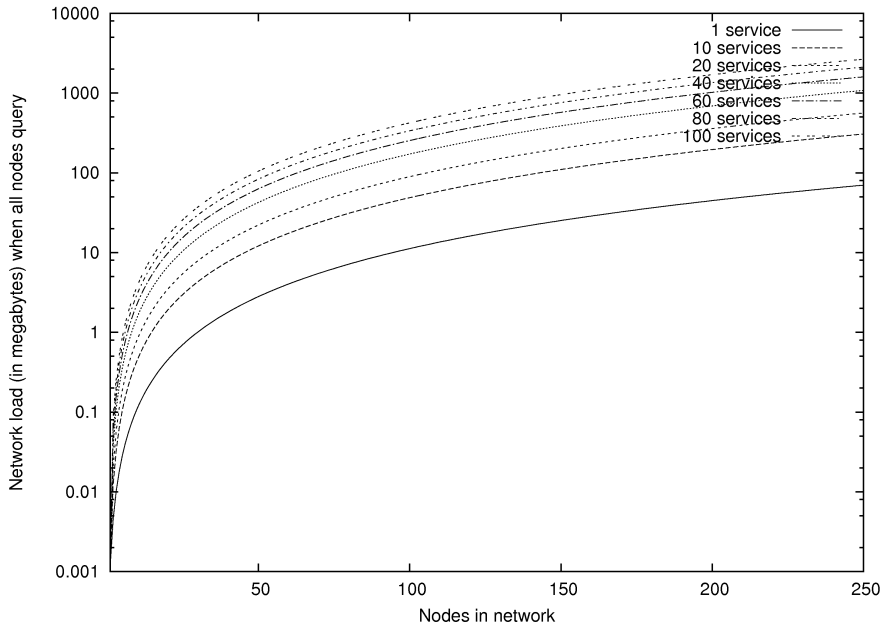


Figure 5.4: WS-Discovery's resource use when all *nodes query.*

up to 250 nodes. This means that in such a network, for every query issued, we get the resource use indicated by the graph. Next, we set $q = n$, so that in a network of a given size, *all* nodes query. Figure 5.4 shows WS-Discovery's resource in this latter case. In both graphs, all nodes are equal and publish the same number of services. We see that with an increasing number of nodes and published services, the overall resource use increases substantially. These results indicate that WS-Discovery may not be suited for use in disadvantaged grids. However, we did not know whether it would work until we attempted to use it in practice.

We attempted to use WS-Discovery in a disadvantaged grid in the CESMO experiment, and found that it was unsuitable for use there since it generated too much traffic in the network and flooded the modem buffers (see Section 3.4.2 or our report [71]). Some disadvantaged grids are not simply single-hop MANETs like in our CESMO experiment. They are multi-hop MANETs and, in those cases, WS-Discovery would only discover services that are within one hop away from you anyway. Both these issues indicate that a different discovery protocol is needed in disadvantaged grids. Since the Web services standards are insufficient, we must consider other alternatives to achieve service discovery in such networks.

5.2 Related work

There are many service discovery protocols, and each protocol has been designed with a set of properties that are beneficial in some specific application scenario. In the following overview, we make the distinction between cross-layer service discovery and application level service discovery.

5.2.1 Cross-layer service discovery

In the battlefield, MANET technology is well suited to meet the demands for dynamic networking. However, limited resources (most notably data rate constraints) in such tactical networks call for specialized solutions. In fact, approaches that work in wired networks may well be counterproductive in MANETs [127]. Thus, one could try a different approach to protocol design in such networks. *Cross-layer design* refers to protocol design done by actively exploiting the dependence between protocol layers to obtain performance gains [125].

There are several experimental service discovery solutions for MANETs that leverage cross-layer design in one way or another: Jodra et al. [64] present a solution integrating service discovery with the OLSR [23] routing protocol. Another solution, Mercury [43, 69], implements a very resource efficient service discovery integration with OLSR. SEDRIAN [101] is a decentralized service discovery approach using AODV [103] as the routing protocol. A proposal by Engelstad et al. [37] utilizes AODV piggybacking in a similar manner.

Current solutions in this category are experimental and tailored for use in civil MANETs. What we need are protocols that can support networks with low data rate, generate few data packets

when communicating, and ideally support both design-time and run-time discovery. Current cross-layer solutions do not support design-time discovery due to their limited expressive power. But, piggybacking is an important technique to keep in mind, since it lowers the number of data packets that need to be sent.

Cross-layer solutions have low overhead and can be of use in military tactical networks where bandwidth is the limiting factor. Application level solutions may also be feasible at this level. If one is capable of utilizing an application level solution then that is preferable, since *binding a service discovery protocol tightly to the underlying network protocols can limit its discovery capability* [146]. Also, *architecture violations can have a detrimental impact on system longevity*, as has been argued for the case of cross-layer design in [75]. Cross-layer approaches demand proprietary solutions and modifications of existing standards. These drawbacks have so far limited cross-layer service discovery from becoming widespread. We discuss application level solutions next.

5.2.2 Application level service discovery

A lot of the existing service discovery protocols are suitable for fixed infrastructure networks or single-hop home or office ad hoc networks. Such protocols (see e.g., the survey by Zhu et al. [146] and our report [67]) are unsuitable for use in highly dynamic environments such as multi-hop MANETs. This goes for protocols such as e.g., the Service Location Protocol (SLP) and Universal Plug and Play (UPnP), which were designed for use in an office environment. UPnP can be seen as a predecessor of WS-Discovery, which we discussed in Section 5.1.3. See [112] for details about SLP and UPnP. Apple's Bonjour (formerly Rendezvous) [22] uses a combination of link-local address choosing [21], Multicast DNS (mDNS), and DNS-Service Discovery (DNS-SD). A proposal to use DNS-SD and Zeroconf technology to interconnect soldier wearable computers and Ethernet-enabled devices is presented in [102]. SLP, UPnP, and Rendezvous share many of the same basic design principles, and are geared towards device discovery rather than Web services discovery. The protocols are worth mentioning because they are widely supported and used for device discovery. However, WS-Discovery exhibits similar behavior as these protocols, and should be used instead of them for the discovery of Web services. As we have seen, WS-Discovery is not suitable for use in disadvantaged grids, so we need to consider experimental protocols as an alternative.

JXTA [53] is a framework by Sun for building P2P applications. Communication is done over TCP/IP and HTTP with XML as the underlying message format. One particularly interesting research project based on JXTA is the Service-Oriented P2P Architecture (SP2A) [4]. It currently supports service deployment with Web services. The NATO C3 Agency (NC3A) is experimenting with a prototype called Service-Oriented Peers (SOP) [83], which builds on code from the SP2A project. SP2A and SOP remove the loose coupling of Web services, demanding that the services be published, searched for and invoked through the JXTA P2P overlay.

Another experimental P2P based Web services framework which also integrates discovery and

service invocation is WSPeer [60]. WSPeer is a component based solution, aiming at providing an API for hosting and invoking Web services. Being component based, the solution is extensible, allowing users to develop application specific service capabilities. However, being a complete framework, it too, like SOP, removes the loose coupling of Web services, tightly coupling service implementation, discovery and invocation to their proprietary API.

Search+ is an experimental P2P based system that we have designed for service discovery in tactical networks (see our publications [69, 122] for an overview, and our report [121] for a more complete discussion regarding the suitability of P2P networks in military networks). What separates this work from that of SOP and WSPeer is that Search+ maintains the loose coupling of Web services. Search+ uses P2P techniques for disseminating service information, but the overlay is not used for service invocation. Invocation is left to the standard Web services mechanisms. This work shows that P2P techniques are beneficial in and across tactical deployed networks. However, in such networks the standardized Web services registries can be employed, and so we do not consider P2P discovery further in this report. In fact, using ebXML in a federation of registries gives you the benefits of P2P technologies, since the ebXML federation functions as a P2P network. This is in contrast to UDDI, which does not allow federated searches, and only supports replication among other registries.

Konark [61] is a fully decentralized service discovery mechanism for multi-hop MANETs. It is a complete middleware for service description, discovery and invocation. The framework is loosely based on Web services, in that XML is used for descriptions, and SOAP over HTTP is used for service invocation. However, the service descriptions are a proprietary twist on WSDL's, meaning that COTS Web services cannot be used with this solution. Konark uses periodic advertisements with a TTL to handle liveness. Konark multicasts a subset of its service knowledge to reduce bandwidth. This means that you may not be able to access all services in your network partition, since service advertisements are made in a random fashion.

The protocol by Sailhan et al. [115] is a solution for service discovery in large networks, both fixed and MANETs. It relies on a backbone (i.e., overlay network) of registries. Each registry is responsible for service discovery in its vicinity, and service discovery on a "global" scale is made through the registry overlay network. Service descriptions use WSDLs with added information (i.e., QoS information about services). Directories advertise their presence in their vicinity by broadcasting advertisements. A directory is not pre-designated, but is chosen through an election process. The protocol has been evaluated in NS-2, and later implemented in Java. However, by combining WS-Discovery and ebXML as a DP, where the registries can be configured in a federation, we could get close to the same functionality as Sailhan, depending on how the DP interface to the registry is implemented. Using the standards in this fashion is preferable to Sailhan, since it is a proprietary solution.

PDP by Campo et al. [19] is a service discovery protocol for ad hoc networks. It is based on sending multicast queries for services to the network. All devices within transmission range listen, and when answering they send a list of services offered by themselves and others (cached earlier query responses). Service advertisements contain a service description and an expiry time. When the

expiry time is reached, the service is purged from the cache. The authors argue that one of the most important issues when designing a service discovery protocol for ad hoc networks is to minimize the number of transmissions, especially of the most limited devices. This was the motivation for combining timeouts, caching, and active queries in the way they do. However, this gives neither the accuracy of pure service querying, nor the low latency of pure advertisements. Combining these techniques in this way gives the drawbacks of both techniques. We argue that if you use querying, the responses should not include services from other nodes, only the one answering in order to be fully accurate (for example, this is the way WS-Discovery operates when you send a *PROBE* message). PDP has been simulated, and later implemented in Java. It supports very simple service descriptions, i.e., a text string representing a service ID or name.

Gagnes [46] assesses several methods for discovering dynamic services in NBD. According to Gagnes, current Web services discovery technologies are not sufficient in such dynamic environments. Especially liveness information, coherence between LAN and WAN service discovery and dynamic registry cooperation is lacking. The paper lists a series of key properties such as robustness, registry autonomy, liveness information, and pluggable service description models, and a hybrid topology with autonomous, federated, cooperating registries is proposed. In this report, we augment these ideas by suggesting a service discovery architecture approach for all operational levels of military networks.

Service discovery protocols suitable for pervasive computing environments are surveyed by Zhu et al. [146]. Several other service discovery initiatives related to multi-hop MANETS are discussed in a recent survey by Mian et al. [85]. It is beyond the scope of this report to discuss them all, it suffices to say that none of these protocols are related to Web services discovery or have properties that make them suitable for use in disadvantaged grids.

5.2.3 Classifying service discovery mechanisms

Different service discovery surveys classify the mechanisms in different ways. Seno et al. [119] make a simple division which reflects whether all nodes are equal or if some do more work than others. For example, with WS-Discovery in decentralized mode, all nodes are equal, whereas with a registry there is a central point that clients connect to. We call this the *decentralized vs centralized* approach. Zhu et al. [146] are mostly concerned with the mechanisms that constitute different service discovery solutions. They classify service discovery solutions by the techniques they implement and the capabilities they offer. That is, they are classified by the service discovery *protocol components*. Another way of classifying service discovery protocols is by their *suitability for use in different networks* (e.g., large vs small and static vs mobile), which is the approach taken by Mian et al. [85].

Decentralized vs centralized

Seno et al. [119] identify two main categories of service discovery mechanisms (see Figure 5.5):

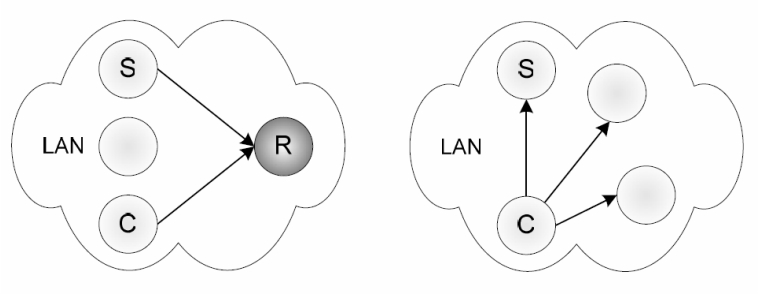


Figure 5.5: Service discovery with a registry (left) and without a registry (right) (from [47]). These two cases illustrate the client-directory-service *model and the* client-service *model.*

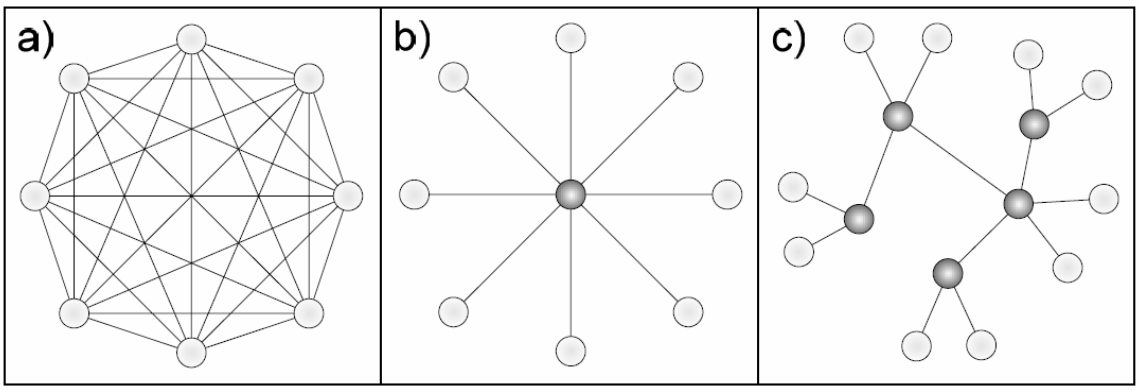


Figure 5.6: Fully decentralized (a), centralized (b), and distributed (c) topologies (from [47]).

1. The client-service model.

2. The client-directory-service model.

The first model does not rely on a registry for service discovery; instead it relies on a totally decentralized topology. In the client-service model, a client may broadcast or multicast its queries in the network. The services that match the received query return a reply. In this model, there is no registry to keep the information about the services. This model is considered suitable for small networks [119]. In the client-directory-service model, there is a registry that keeps information about all the services which are in the network. This registry has to be queried about available services before a service can be used. This approach is most suitable for environments with hundreds or thousands of services [146].

A registry can be either centralized or distributed [47]. The main idea in the second model is that some nodes (directories) maintain the service information, while other nodes (clients) query them. In Figure 5.6 we can see the difference between fully decentralized, centralized, and distributed topologies. The client-service model is fully decentralized, whereas the client-directory-service model can be implemented with either a centralized or a distributed registry solution.

Protocol components

Zhu et al. [146] survey service discovery protocols in the context of pervasive computing. They observe that existing protocols have various design goals and solutions, and develop a taxonomy for major service discovery components. We summarize the components below, and discuss the properties in the context of Web services standards:

Service and attribute naming: Typically, this will vary from protocol to protocol. Some support templates, i.e., they define a format for service attributes and names. Other protocols also offer a predefined set of common attributes or frequently used service names. For Web services discovery we must be able to relate to a WSDL, or a part of a WSDL.

Initial communication method: Some protocols rely solely on unicast. Others use multicast or broadcast and may switch to another communication mode (e.g., unicast). UDDI and ebXML fall in the former category, WS-Discovery in the latter.

Discovery and registration: Some protocols are query based, supporting a request/response pattern for discovering services. Another paradigm is using announcements, i.e., a "push" pattern. Typically, the Web services registries are query based, whereas WS-Discovery supports both patterns.

Service discovery infrastructure: The solution can be either non-directory-based or directory-based. This property is the same as the client-service and the client-directory-service models that we discussed above.

Service information state: Protocols can keep soft state or hard state. WS-Discovery keeps soft state, whereas the registries keep hard state.

Discovery scope: Scope can be determined by network topology, user role, or context. Proper use of discovery scopes can minimize unnecessary computation in the discovery process. The registries support discovery by context, whereas WS-Discovery supports discovery by topology (within a single hop) and context (i.e., XML namespace and WSDL port type).

Service selection: Service discovery protocols can offer manual or automatic selection. The W3C WS-Architecture document [143] refers to this distinction as manual and autonomous discovery. Manual selection leaves all control to the user, and is the type of selection that the Web services standards offer.

Service invocation: The *first level* protocols provide only service locations (addresses), leaving the application responsible for defining the communication protocol. For example, PDP is a first level protocol, whereas for Web services discovery we ideally want a second level protocol. The *second level* protocols define the address and the communication mechanism. The standardized mechanisms for discovering Web services are second level, because WSDLs do not only contain the invocation address, but also describe the transport protocol, operations, message flows, and fault

handling that a service supports. There are also *third level* protocols, that define application specific operations in addition to all the features offered by a second level protocol. UPnP is an example of a third level protocol (e.g., it allows automatically opening TCP and UDP ports in UPnP enabled firewalls).

Service usage: This property, if supported by a service discovery protocol, is either based on explicitly releasing a service, or providing access in a lease-based fashion. Leases are most suitable under dynamic conditions. The Web services discovery standards support neither of these, since Web services are intended to be used by several clients simultaneously. This is an important distinction from that of pervasive computing, where a service typically corresponds to a device.

Service status inquiry: The discovery mechanism can either poll or receive notifications about a service's status. The Web services discovery standards support neither of these.

Suitability for use in different networks

Discovery mechanisms can also be categorized by the type of network that they are suitable for. Mian et al. [85] classify networks and mobility as follows:

A *small* network has up to 10 nodes. A *medium* network has from 10 to 100 nodes, whereas a *large* network has more than 100 nodes. Node velocities up to 5 km/h constitute *low* mobility. Nodes exhibiting *medium* mobility travel with velocities between 5 and 50 km/h. Networks with *high* mobility have nodes with velocities in excess of 50 km/h.

Further, discovery protocols are classified in these four categories:

1. Directory-based with overlay support.
2. Directory-based without overlay support.
3. Directory-less with overlay support.
4. Directory-less without overlay support.

They argue that for large networks with low mobility category 1 is most optimal. For medium networks with medium mobility, the categories 2 or 3 are best. Finally, in small highly mobile networks, protocols in category 4 are most suitable.

Typically, fully decentralized solutions (i.e., using the client-service model) fall in category 4. P2P networks form overlays [121], and typically fall in category 1 or 3. Drawing a parallel to the Web services standards, UDDI and ebXML fall in category 1 when they are federated, and in category 2 when they are used as stand-alone registries. However, as we discussed above, the registries are not suitable in dynamic environments unless liveness is handled in some way (or that the multi-hop MANET is never partitioned, as it seems is the assumption in [85]). WS-Discovery falls in category

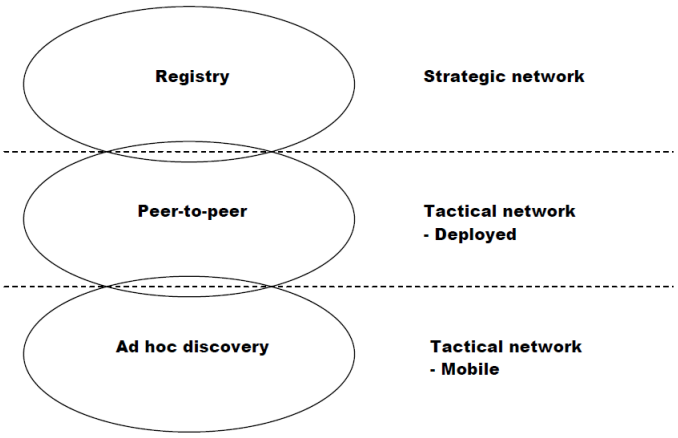


Figure 5.7: Suggested service discovery mechanisms for each operational level (from our report [67]).

4 when it is operating fully decentralized. Category 3 is not supported by current Web services standards. We know that WS-Discovery is not very suitable for disadvantaged grids, meaning that it could be worth investigating other category 3 and 4 solutions.

Strategic networks are not mobile, but if we want to classify them according to this scheme, then they would be in the large network, low mobility category. The deployed tactical networks would fall in the same category. However, the mobile tactical networks would fall in the small or medium network category, and their nodes would exhibit everything from low mobility (e.g., soldiers on foot) to high mobility (e.g., vehicles). In [67] we have surveyed several existing service discovery protocols, and conclude from a theoretical standpoint that registries are suitable for use in static networks, P2P based systems (including the ebXML federation) are suitable in tactical deployed networks, and that specially tailored ad hoc network discovery mechanisms must be used in disadvantaged grids. However, there is not necessarily a hard limit between these levels when it comes to discovery solutions, as it will depend on the actual communications hardware being used in the deployment. This is illustrated in Figure 5.7. Keeping this in mind, we can attempt to use ebXML in both strategic and tactical deployed networks. However, we will have to develop a suitable Web services discovery mechanism for use in disadvantaged grids, since none of the standards or existing experimental solutions available are adequate.

5.3 Towards service discovery in disadvantaged grids

We have seen that several key properties are missing when deploying today's standards for Web service discovery in dynamic environments. See Table 5.4 for a summary. Thus, we need to research solutions which can accommodate the network-centric battlefield, while at the same time ensuring interoperability with other systems. From Section 5.2.3, we learned that a category 4 solution should

Standard	Service Discovery on LAN/WAN	Mirrors network state	Resilient towards partial network failure	Descriptions	No DNS and WWW dependency
UDDI	WAN	NO	Data replication between registries	UDDI information model	NO
ebXML	WAN	NO	Registry federation	ebXML information model	YES, repository mechanism
WS-Discovery	LAN	YES, but not on discovery proxy	OK	Namespace, port type and scope	NO

Table 5.4: Capabilities of standardized Web services discovery mechanisms.

be suitable for disadvantaged grids. Lacking such a solution for Web services, we create our own protocol that we call *Service Advertisements in MANETs (SAM)*.

5.3.1 Our novel protocol: SAM

As we have seen earlier, we lack a suitable Web services standard for use in tactical mobile networks. Ideally, we want to be able to discover Web services in such networks. We have implemented the necessary functionality in an experimental solution. We chose a set of techniques that are well suited for overcoming the service discovery challenges in small, highly mobile networks [85, 146]:

- Decentralized operation to overcome availability issues,
- periodic service advertisements to ensure an up-to-date view of available services, and
- caching to reduce resource use (no need to query the network).

Further, we introduce piggybacking of NFFI position information to reduce the overall number of data packets, and compression to reduce the size of the data packets. The PDT from NFFI (see Section 3.4.1) is incorporated in the service discovery mechanism. By distributing the PDT at regular intervals together with service information, the units can know where other friendly units are, and also the services they provide. By using such a proactive push model, we also ensure that nodes in radio silence will benefit from the data being sent. We implement all these techniques in SAM.

5.3.2 Observations and system design

Supporting service discovery in the tactical battlefield is a major challenge. The idea of combining service discovery with friendly force tracking (see Section 3.4.1) is based on the following observations:

You need to know where your mobile units (e.g., squad members) are, thus there is a need to exchange position data. Also, you need to know which services are available where and when, thus there is a need to exchange service advertisements.

Previously, cross-layer solutions have been suggested as a resource efficient service discovery mechanism (e.g., Mercury). However, cross-layer solutions violate layered design, making an application level service discovery solution preferable.

In some cases it is beneficial to not only discover a service, but also the position of the service. For example, this could be the case if the output of the service is linked to the position of the unit (sensor area coverage, etc). Since we need to send NFFI information anyway, we might as well piggyback it with service information in the advertisement messages. This gives the benefit of getting the unit position and all active services provided by the unit in each update. Also, it limits the number of data packets that need to be transmitted.

The information about the currently active services in each unit is described by a WSDL, which is the standardized way to describe Web services. The WSDL contains enough information about a service not only to invoke it, but also to implement a client to the service. In an operation, this latter function is not important, since each unit will already have the software it needs to fulfill its role. Thus, it is finding the endpoint of the WSDL, i.e., the invocation address of the service, that is of interest in an operation. The software needs this endpoint in order to address and invoke the service properly.

WSDLs are large XML documents which contain a lot of static information, and some dynamic information. We want to separate the two, since the endpoint is our main concern. Preferably, the service discovery mechanism should exchange something more compact than the WSDL to reduce communication overhead. This is important in a multi-hop MANET, since the data rate effectively is cut in half per hop [84] for nodes that cannot send and receive at the same time. We want as much bandwidth as possible available for use by the applications' traffic. To achieve this, we can for example distribute a complete but compressed WSDL document. The entire WSDL is needed if you want to create new client software for previously unknown service types. However, the entire WSDL is not needed in an operation, because new clients will not need to be implemented on-the-fly; nodes will come with pre-made software. The problem here is identifying and invoking *known* services. This leads us to consider using a hash of the WSDL document, since a hash will uniquely identify an already known WSDL and represent it in a very space efficient manner. Also, since a WSDL defines a service, a hash can be used to uniquely identify a service type.

Adopting the hashing approach in our SAM solution, we use a hash over a WSDL with the endpoint removed. Since the endpoint can change, it cannot be a part of the hash. The remainder of the WSDL, on the other hand, is static and identifies the interface of the service. The service discovery mechanism disseminates the hash together with the current endpoint to uniquely identify a service and its invocation address.

5.3.3 Implementation

SAM is tailored to the specific requirements for service discovery in small, highly dynamic multi-hop MANETs. It can function disconnected from the Internet since it hosts all the necessary data and metadata in a local repository. This repository must be populated with the WSDLs of the service types a node wishes to discover.

SAM is resilient to partial failure since it is a fully decentralized solution. Thus, it will still function if the network becomes partitioned — the clients in each partition will have a fully operational service discovery mechanism at hand. The mechanism is based on periodic dissemination of service advertisements, and the advertisements are cached locally in each recipient. Whenever a new update is received the cache is refreshed, and any outdated services are removed. Each service exists in the cache only a limited time before it is deleted, thus ensuring that a query in the local cache will give a fairly up-to-date picture of the available services. An advertisement contains a list of services being provided by the node that sent the advertisement. This list contains entries uniquely identifying the service, and any metadata associated with the service. If the node chooses to report its position (e.g., if it is fitted with a GPS), then the position data (latitude, longitude, node ID, and timestamp) is sent along with the list of services.

Basically, an advertisement contains the following XML-encoded data:

- A position (optional). This is the PDT from NFFI.
- A list of services, containing one or more of the following entries: A unique service hash ID (required), an endpoint URL (required), and metadata (optional).

This means that an example advertisement containing a position and two Web services (but no metadata) can look like the one in Figure 5.8.

The endpoint URL is the invocation address of the service being identified by the unique service ID in that particular node. In other words, this is the endpoint address as it can be found in the service definition of the WSDL (see Figure 5.1). This URL is the only dynamic aspect of a Web service description, as the rest of the WSDL can remain static from deployment to deployment. Based on this observation, we found that by removing the endpoint URL from the WSDL, we would get a WSDL which could uniquely describe all services of the type defined therein. To reduce the bandwidth requirements of the advertisement distribution mechanism, we chose to use a SHA-1 [33] hash over the WSDL without endpoint to uniquely identify a service. Metadata associated with the service is optional. The metadata field is a flexible measure provided by the solution, in that it can be used by regular Web services for discovery (i.e., just ignore the field) or by Semantic Web services for added reasoning capabilities [59]. This makes our protocol support both regular and Semantic Web services (see Section 7.2.5), which was one of the desired properties discussed in Section 5.1.2.

Caching of service advertisements with timeouts is used to address the liveness issue. In civil systems, using caching may be undesirable: For example, using IEEE 802.11b you have high

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ns2:ServiceDiscovery xmlns:ns2="urn:no:ffi:servicead"
    xmlns="urn:nato:fft:protocols:nffi13">
    <ns2:position>
        <trackSource>
            <sourceSystem>
                <country>NOR</country>
                <system>FFI</system>
            </sourceSystem>
            <transponderId>BASE1</transponderId>
        </trackSource>
        <dateTime>20091116094916</dateTime>
        <coordinates>
            <latitude>53.3618627551</latitude>
            <longitude>6.26719155956</longitude>
        </coordinates>
    </ns2:position>
    <ns2:service hash="E67AE486E32BB2FB2F551BD14EC57D56D258AD3C"
      address="http://192.168.3.3:8080/NffiService/SIP3_Service_ReqRes"/>
    <ns2:service hash="7E0119A31D7244A51C939A46A685271BC82944F2"
      address="http://192.168.3.3:8080/PingWS/PingService"/>
</ns2:ServiceDiscovery>
```

Figure 5.8: An example service advertisement with position data and two Web services.

bandwidth but short radio range. In such cases, querying the network may be preferable to caching depending on the mobility pattern. In disadvantaged grids, on the other hand, we need to reduce bandwidth use as much as possible. Also, military units move in what can be described as a *hierarchical group* pattern [44], meaning that clusters of units typically move around together. In other words, units in a squad tend to stay together, and can communicate with each other. On the other hand, they may move out of range of their HQ, losing their reachback link. This means that using caching in a disadvantaged grid has less negative impact than in a civil network, because of different characteristics and different mobility patterns. Partitioning may occur if the group divides into smaller groups, or for any other number of reasons such as for example hostile activity or entering radio silence (see Section 2.1.2), in which case the timeout ensures that old entries are purged from the cache.

Two caches are maintained: A local cache (i.e., a cache of the services being provided by this particular node), and a remote cache (i.e., a cache of services that have been discovered through advertisements sent by other nodes). Updates are sent at regular intervals (this is configurable; we used an interval of 30 seconds). The update is sent in the form of an XML service advertisement. The list of services in the advertisement is the set of services contained in the local cache.

Upon receiving an update over the network, the remote cache is updated with the services from the advertisement. The services in the remote cache time out if they are not refreshed through new advertisements. The timeout is configurable, but in the prototype the value is set to two times the advertisement interval, so that the service is deleted from the cache the second time the service misses its advertisement slot. This achieves the desired liveness, in that liveness is defined by the

lack of advertisements, rather than requiring the service provider to actively de-register its services. This solution is preferable in a dynamic environment.

Our solution is resource efficient in that querying for services is always made locally: A lookup in the local cache is all that is needed to get the current state of the network, and no network packets need to be sent to fulfill the request. Also, the advertisements are made as small as possible before transmission by using GZIP compression. We used GZIP in our implementation because it was freely available. Efficient XML would yield better results, but due to restrictive product licensing, we chose to adopt the second best alternative in the prototyping of SAM as a proof-of-concept.

5.3.4 Evaluation

We designed SAM to be an alternative to WS-Discovery in tactical mobile networks. SAM combines NFFI position descriptions with advertisements of available Web services described by their WSDLs. Our experimental solution is tailored for use at the lowest tactical level, the disadvantaged grids, where disruptions are frequent and bandwidth is scarce. It can be used in environments where the standardized Web services discovery mechanisms are unsuitable, provided that the network supports IP multicast. Hashing, caching and compression techniques are employed to keep bandwidth usage low, while at the same time being able to reap the benefits of XML technology.

Proof-of-concept

We used SAM for service discovery in a military multi-hop MANET at the Combined Endeavor exercise (further details from this experiment is discussed in Section 6.3.2, and in our report [69]). There, SAM functioned flawlessly for both service discovery and node position tracking. We had four Web services in our network, and comparing SAM to WS-Discovery gave the following results:

Compared to WS-Discovery, SAM is more bandwidth efficient. In our tests, WS-Discovery had to rely on IP fragmentation. IP fragments are undesirable for two reasons: 1) They increase the impact of packet loss, and 2) not all military equipment supports IP fragments (see Chapter 2). Conversely, by using compression SAM kept all advertisements well under the MTU, thus avoiding IP fragments. This means that for typical use (i.e., few published services at the lowest tactical level), we did not need to implement application level fragmentation for use at the exercise. We used the draft version of WS-Discovery in our evaluation, since a standard-compliant implementation was not available at the time — however, we changed the number of message transmissions to adhere to the number specified by the standard.

WS-Discovery sends *HELLO* messages during startup. For four Web services, this generates eight packets, i.e., two hello messages per service (the payloads are 1008, 983, 1074, and 1017 bytes respectively). Searching for services involves sending *PROBE* messages. For each search, two probes are sent (2x 597 bytes payload). WS-Discovery enabled nodes with services then reply by

	Four Web services AND position information	Just the four Web services
Uncompressed	1122 bytes payload	681 bytes payload
Compressed	728 bytes payload	516 bytes payload

Table 5.5: Example SAM service advertisement sizes.

sending *PROBE MATCH* messages. In our case one node has four services, resulting in one big probe match message (fragmented packet, payload total 2506 bytes). This is not resource efficient in a disadvantaged grid where bandwidth and buffer space limitations come into play. The WS-Discovery draft specification performs even worse, in that it generates twice as many packets in all the cases discussed above.

SAM, on the other hand, sends periodic broadcasts every 30 seconds (this is configurable). SAM's resource use for our four Web services is shown in Table 5.5.

This served as a proof-of-concept, showing that SAM can work in a tactical mobile network. SAM has potential as a Web services discovery protocol for MANETs, and could be of interest to others. However, since the prototype used at Combined Endeavor uses parts of NFFI, it cannot be freely distributed outside NATO. Following that exercise we have designed our own PDT replacement for SAM, which is capable of carrying the same position information, but does not rely on NATO schemas and namespaces. This modified Java implementation has been released as an open source project and can be downloaded from:

http://sourceforge.net/projects/servicead/

In the following section we perform a comparative evaluation of WS-Discovery and the first open source release of SAM.

WS-Discovery vs SAM

SAM utilizes periodic advertisements, so calculating the total load on the network can be expressed as a simple equation where the variables are the number of nodes (n) in the network, the number of advertisement intervals (i), and the sizes of the service advertisements (see Equation 5.2).

$$LOAD_{SAM} = n * i * sizeof(advertisement) \tag{5.2}$$

Table 5.6 shows the resource use of one SAM node when sending a service advertisement for a varying number of published services and position information (using the same WSDLs as we used for the evaluation of WS-Discovery in Section 5.1.4).

Calculating the average number of bytes to represent services in SAM (using compression and including position information), we get approximately 71 bytes. Thus, SAM has a theoretical upper limit of publishing approximately $65507/71 \approx 923$ Web services with position information per node

Services	SAM	Compressed SAM	SAM w/position	Compressed SAM w/position
100	11742	5228	12183	5460
80	9472	4240	9913	4472
60	7245	3284	7686	3516
40	4890	2308	5331	2544
20	2510	1372	2951	1600
10	1349	884	1790	1108
1	260	300	701	520
0	N/A	N/A	565	392

Table 5.6: Size of SAM advertisements (in bytes) with compression enabled or disabled, and with position information missing or present.

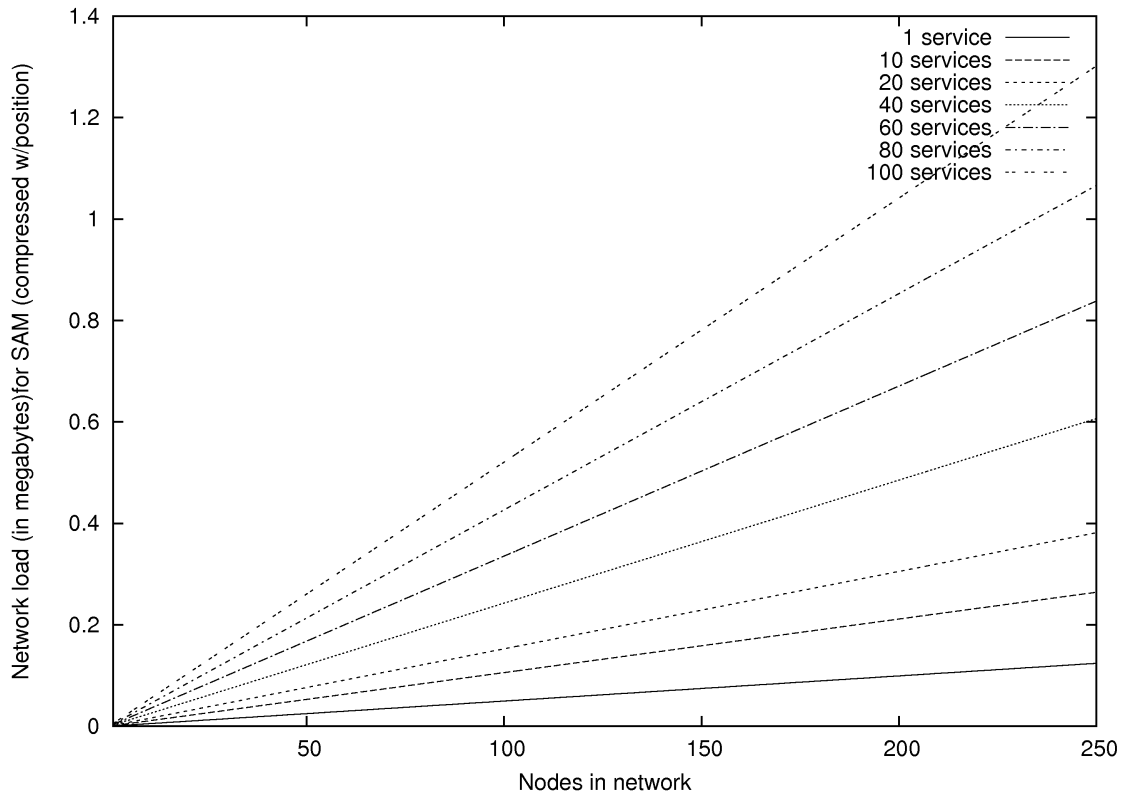


Figure 5.9: Resource use (in megabytes) per service advertisement.

when considering results from our 100 WSDLs. We see that, since neither SAM nor WS-Discovery supports application level fragmentation, SAM scales to a much larger number of published services per node than WS-Discovery. We compare SAM's network resource use with that of WS-Discovery next.

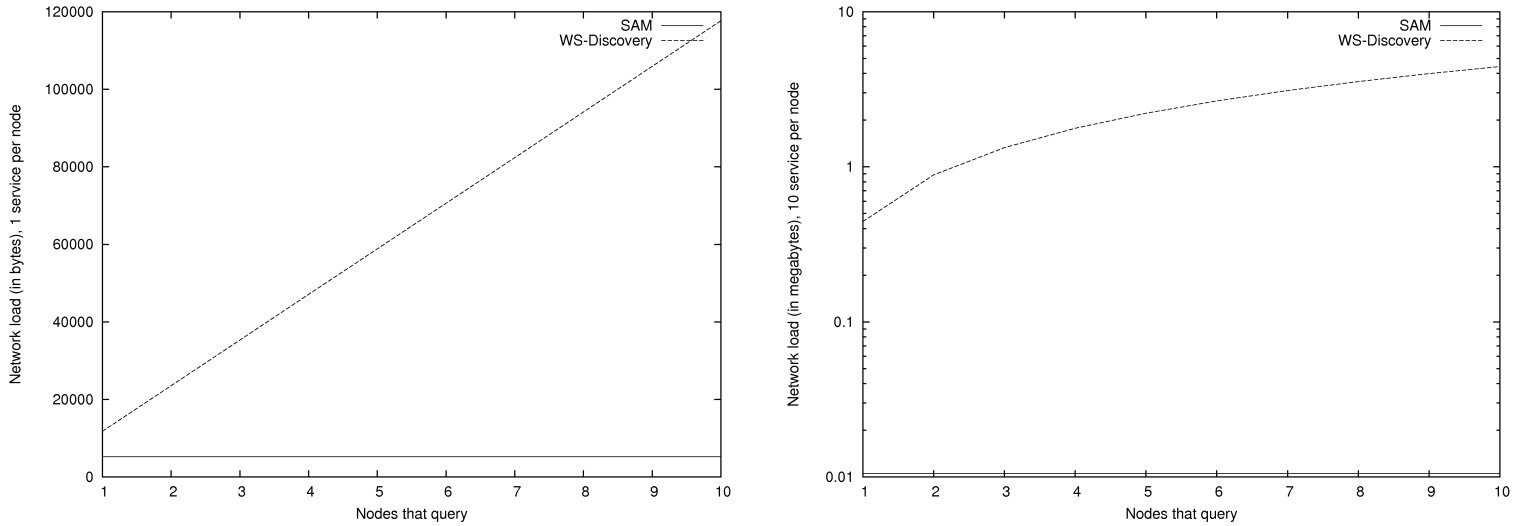
Figure 5.9 shows plots for equation 5.2 using values from the "Compressed SAM w/position" column in Table 5.6. The graph shows resource use per service advertisement (i.e., $i = 1$), meaning that this resource use will be constant over time for each advertisement interval and independent of the number of queries issued by the nodes. SAM requires a fixed and deterministic resource use over time. This is due to SAM's proactive operation, in contrast to WS-Discovery's reactive operation, where the number of queries issued directly influences the network load.

In Section 5.2.3, we discussed the network classification by Mian et al. [85]. According to their definition, a small MANET has 10 nodes or less. Using their definition, we perform an analytical comparative study of WS-Discovery and SAM in a network with 10 nodes. Figures 5.10(a) and 5.10(b) illustrate this for networks with 1 service per node and 10 services per node within one SAM interval (i.e., $i = 1$ in Equation 5.2), respectively. Over time, the resource use WS-Discovery will increase based on the number of queries issued, whereas SAM's resource use is constant. Figures 5.11(a) and 5.11(b) illustrate the differences between the two protocols for the duration of an hour (here $i = 120$ since SAM by default issues two advertisements per minute). WS-Discovery(1) shows the results for one node querying, whereas WS-Discovery(10) shows the results for all nodes querying. Thus, we can see the upper and lower bounds for WS-Discovery's resource use. We see that SAM performs best in terms of overall bandwidth use when there are many queries issued, whereas WS-Discovery performs best when there are few queries issued. Keep in mind that all these results show discovery payloads only. In the case of WS-Discovery, one would need to run an NFFI service for node tracking in addition to the service discovery protocol, but in our evaluation of SAM position information is included.

5.4 Claim review

SAM follows the *client-service* model, and implements a *second level* protocol for Web services discovery. It is a *category 4* solution according to the classification by Mian et al. [85] (see Section 5.2.3), since it is directory-less and without overlay support. Comparing SAM with the properties we want a service discovery protocol for military networks to meet (as identified by Gagnes), we see that our protocol satisfies all the demands. We wanted a solution to (see Section 5.1.2):

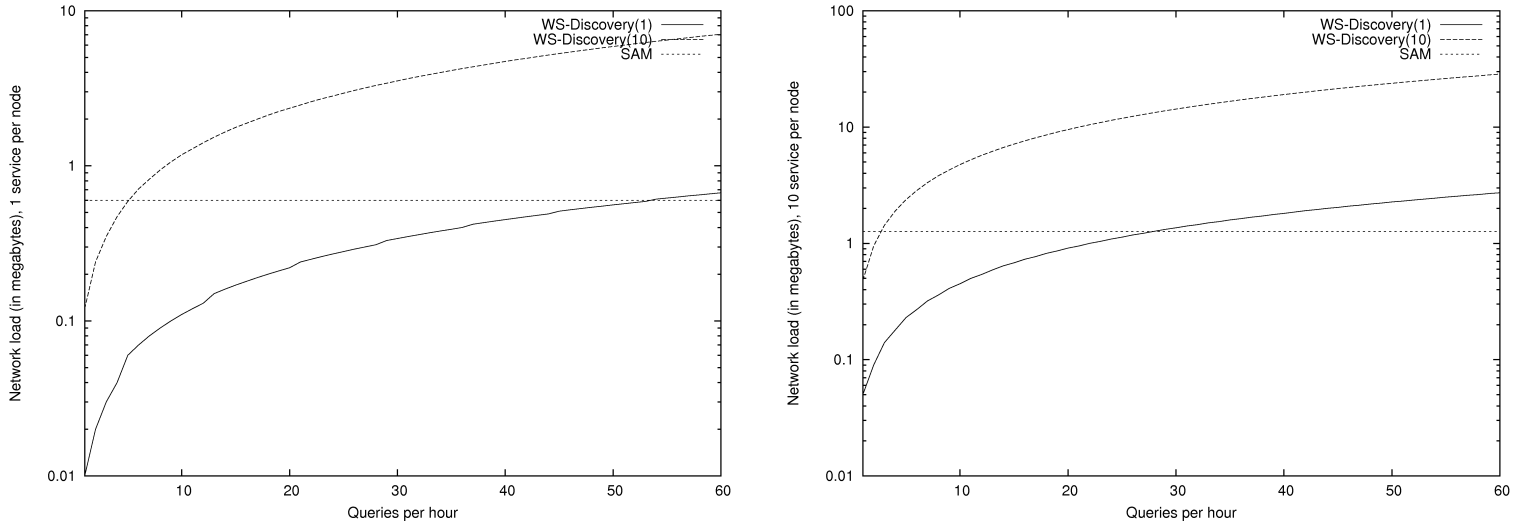
- *Be based on standards, if possible.* SAM is not a standard (however, the standards do not provide a suitable category 4 solution), but it utilizes standards to the fullest extent: XML is used as the language for describing the service advertisements. The PDT from NFFI is used



(a) 1 service per node.

(b) 10 services per node.

Figure 5.10: Resource use of SAM and WS-Discovery in a network with 10 nodes.



(a) 1 service per node.

(b) 10 services per node.

Figure 5.11: Resource use of SAM and WS-Discovery for a duration of one hour. WS-Discovery(1) denotes that one node queries X number of times, whereas WS-Discovery(10) denotes that all nodes query X number of times.

to describe a position, if present. WSDLs are used for service discovery, in that the service advertisements disseminate a hash over a WSDL and its service endpoint.

- *Be able to discover services in a LAN, WAN, and/or military MANET.* Discovery in LAN and WAN is covered by current Web services standards. The standards are not suitable for use in military MANETs, and a new solution is needed for such networks. SAM is designed to function in military MANETs.

- *Mirror the network state (i.e., a query should return only available services).* SAM mirrors the network state as it was when the last service advertisement was received. Using cache timeouts removes services that are no longer available from the cache. This means that querying the local cache returns the currently available services.

- *Be robust.* SAM is a fully decentralized solution and has no single point of failure.

- *Support service descriptions that are rich enough to support run-time discovery (and preferably design-time discovery) of Web services.* In its current implementation, SAM supports run-time discovery. However, it would be possible to extend the implementation to also support design-time discovery. In that case, it would require us to implement a resolve mechanism, so that if a hash for a WSDL we do not know about is encountered, then this WSDL can be retrieved from the node that sent the advertisement. The sender node must have the WSDL in order to create the hash, so asking it to send a compressed WSDL should be feasible to implement.

- *Ideally have no DNS and WWW dependency. This is important for military networks, which must function disconnected from the Internet.* SAM utilizes a local repository in each node which hosts the necessary information. This requires storage space in each node, but solves the important requirement of having no DNS and WWW dependency.

Furthermore, we consider the general requirements analysis from Section 3.5. The requirements that a solution should use Web services technology and always use standards when possible have been covered in the above discussion. In the tactical mobile networks we need support for radio silence (SAM nodes can receive positions and service information without having to transmit data after the multicast group has been joined), support for limited bandwidth (SAM uses compression), support for sudden disruptions (SAM is designed for military MANETs, as we elaborated on in the above discussion), should not rely on TCP (SAM uses UDP). Furthermore, we wanted support for low MTU and that a solution should not rely on IP fragments. Since SAM is based on UDP it may be prone to IP fragmentation in networks with low MTU and a large number of published services. In practice (at Combined Endeavor) this was not a problem, since we had few deployed services which made small datagrams (typically, the lowest tactical level has few services, as shown in Figure 1.1(b)). SAM is modular, so in the future it would be possible to implement support for one of the tactical transport protocols that we discussed in Section 4.3.3, which would overcome this issue. Finally, we wanted a solution to support a federation of systems. SAM is intended

to be a bandwidth efficient replacement for WS-Discovery. Using SAM in networks where the standards cannot function, and the standards in the other networks means that we can achieve service discovery within each sub-network of the federation. With this, we have shown that *networks with different properties require different discovery mechanisms*. However, we also need to solve the interoperability aspect so that the different solutions can exchange information across network boundaries. This is the topic of the next chapter.

5.5 Summary

In this chapter, we have investigated Web services discovery. Making the distinction between design-time and run-time discovery, we have discussed the need for run-time discovery in dynamic environments. Further, we have surveyed the standards related to Web services discovery, and concluded that they are suitable in WANs and LANs, but not in military MANETs. Since none of the standards are suitable for disadvantaged grids, we have created our own mechanism, SAM, which implements properties that are desired of a service discovery mechanism in a small, highly dynamic MANET. SAM cannot be used in a WAN, where it does not scale. Also, IP multicast is often not supported across WANs. Thus, we have proved research claim #2 from Section 1.5, and shown that *networks with different properties require different discovery mechanisms*. Having seen that we can discover services within different operational levels, we need to address the issue of interoperability across operational levels. This topic is covered in the next chapter.

6 Pervasive service discovery

In the previous chapter, we saw that different networks required different discovery mechanisms. Choosing a service discovery mechanism that can cope with a certain network's characteristics allows us to discover services within that network. However, in a military network we can have several interconnected heterogeneous sub-networks, and we must be able to discover services across these networks. This demands that we solve the issue of interoperability between different service discovery protocols, thus achieving *pervasive service discovery*.

In this chapter we address research claim #3, that *in a federation of systems there is a need for application level interoperability points for service discovery*. We investigate different proposed solutions for pervasive service discovery from the literature, and identify the approach that should be employed for military networks. Finally, we perform proof-of-concept experiments that solve interoperability between standardized Web services discovery mechanisms and proprietary discovery mechanisms.

We focus on one particular registry standard, ebXML, for two reasons: First, it is the most flexible of the registry standards, in that it provides the best federation mechanism. Second, it is the standard that the NATO metadata registry and repository (NMRR) is built on. This makes ebXML a sensible choice for a registry standard, in that it eases interoperability with NATO. However, a registry is not suitable for use in a disadvantaged grid, so we need to find other ways to discover services there. For the disadvantaged grids, we solve interoperability between proprietary protocols and WS-Discovery, where WS-Discovery further ensures interoperability with the ebXML registry in the tactical deployed network. Finally, we bring our experimental software to an operational experiment, and evaluate the functionality in an operational context in a field trial.

6.1 Introduction to pervasive service discovery

Interoperability is important not only between the different networks in a coalition force, but also between all the operational levels used within a nation. This means that we must be able to discover Web services across different networks in a coherent manner, i.e., we need support for pervasive service discovery. Furthermore, we must solve pervasive discovery in a manner that can function in a federation of systems, which is an important premise from our requirements analysis (see Section 3.5). This is illustrated in Figure 6.1.

Zhu et al. [146] state that:

> Existing protocols have various design goals and solutions. Each has its advantages and disadvantages in different situations, so it seems unlikely that a single protocol could dominate in pervasive computing environments. With current protocols, this means clients and services can't discover each other if they don't use a common protocol. We

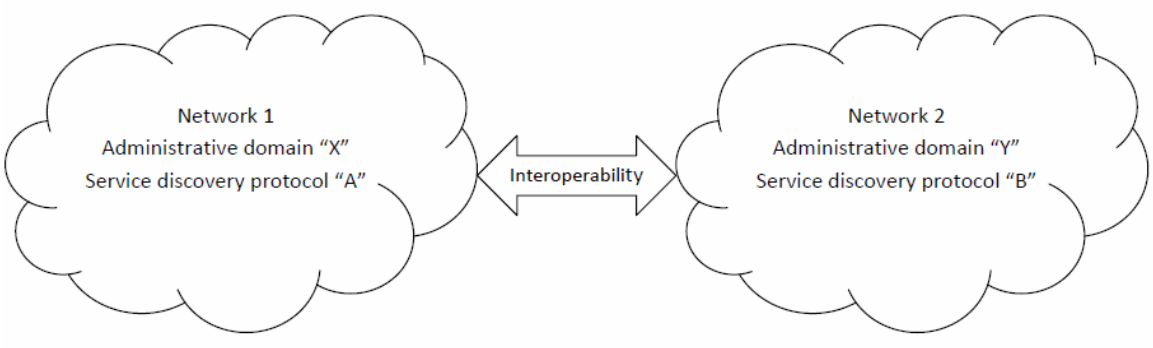


Figure 6.1: Pervasive service discovery in a federation of systems. Here two different networks are interconnected, and clients in one network should be able to discover and invoke services in the other. To achieve interoperability, both networks could use the same discovery mechanism. However, this may not be possible due to policy or hardware limitations. Thus, we must find a means of achieving interoperability between discovery mechanisms A *and* B *which is suitable in a federation of systems (here illustrated by administrative domains* X *and* Y*).*

should therefore establish a common platform to enable interoperability among service discovery protocols.

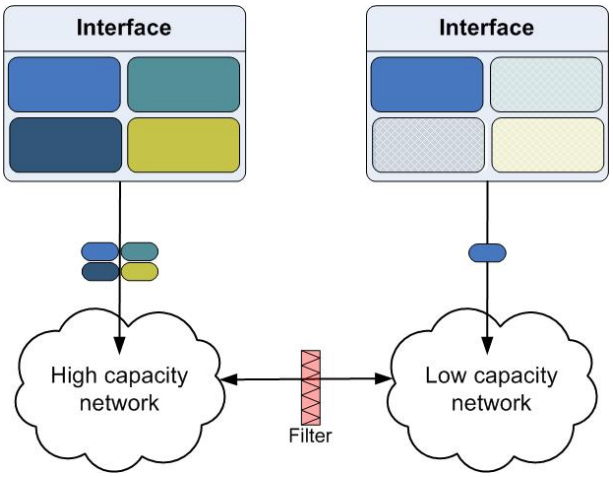
Thus, after we find a protocol (or set of protocols) to use in the different military networks, we need to find a means of interconnecting the service discovery mechanisms across the heterogeneous networks.

6.2 Related work

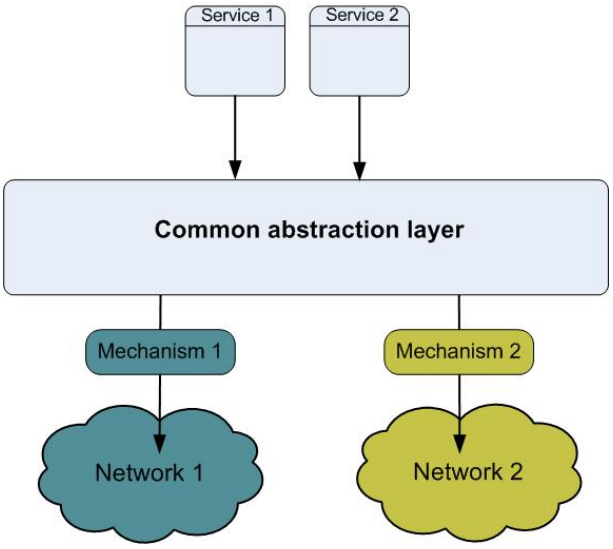
As we have seen in Chapter 5, existing discovery protocols are very diverse, and different networks require different protocols. This means that there is a need for pervasive service discovery. In recent years, several experimental solutions have appeared which attempt to address the challenge of service discovery across different network domains. Lacking a standardized solution, we explore which solutions the research community has identified:

Some protocols are adaptive and can be used in different domains simultaneously. A design for such an adaptive protocol is discussed by Gagnes et al. in [47] and [46]. Bethea et al. [13] investigate the use of ontology-based reasoning for the purpose of developing a general service discovery capability in multi-provider tactical networks.

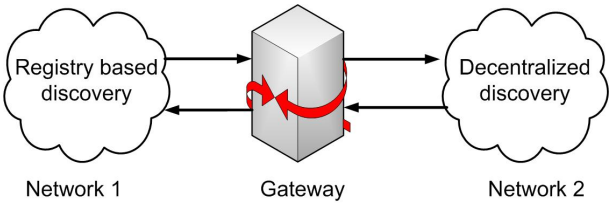
Another approach involves creating a layered structure that can allow different legacy protocols to coexist by adding a service discovery abstraction layer above the others [18]. Raverdy et al. [111] have implemented MUSDAC, a middleware platform addressing the lack of interoperability between existing discovery protocols in a pervasive environment. MUSDAC achieves service



(a) Adaptive service discovery



(b) Layered service discovery



(c) Service discovery gateways

Figure 6.2: The three ways of achieving pervasive service discovery

discovery interoperability through "manager" and "bridge" components: A manager handles discovery requests within the network for local clients. Service discovery plugins implement specific service discovery protocols, and connect to the manager which forms an abstraction layer above the plugins. Finally, bridges assist the manager in expanding service discovery to other networks, by forwarding data to corresponding managers there.

It is also possible to implement service discovery gateways to support transparent interoperability between different protocols. Allard et al. [2] create a gateway which allows Jini clients to use UPnP services and UPnP clients to use Jini services transparently and without modification to service or client implementations. A similar suggestion is that of Kang et al. [74], who present an architecture to provide simple interoperability among various service discovery protocols using a dynamic service proxy. Gateways are considered more efficient than a layered architecture, and also make it possible to use legacy client applications and services unmodified in the network [18].

6.2.1 Classifying the approaches

We can classify the different ways of achieving pervasive service discovery in these three categories that we discovered in the literature review:

1. *Adaptive service discovery*, meaning that one single service discovery protocol is used across all network domains. The chosen protocol must be able to adapt its behavior according to the limitations and capabilities of each underlying network. All applications in the network must be able to interact with this protocol. Going from a network with high capacity to a network with low capacity, a filter is needed that discards excess information. This is illustrated in Figure 6.2(a).

2. *Layered service discovery*, meaning that each network domain can use a dedicated protocol, but an overlaying protocol controls and connects the different service discovery protocols. Here, a network local protocol is translated to the abstraction layer model and offered to the client (e.g., MUSDAC's manager). Interoperability across networks is handled by the abstraction layer, in that it can disseminate its model to other instances in other networks (e.g., MUSDAC's bridge). This case is shown in Figure 6.2(b).

3. *Service discovery gateways*. Using this method, each network domain can choose the most suitable protocol, and interoperability is ensured by using service discovery gateways between the domains (see Figure 6.2(c)). Irrespective of the chosen architecture, interoperability is ensured by the creation and interpretation of service descriptions in clients, servers and in gateways. The structure of the different service descriptions determines whether interoperability is fully, or only partially possible.

All these approaches can solve the pervasive service discovery requirements for heterogeneous networks. However, some of these approaches have been developed for civil networks and may not

be suitable for use in military networks. We investigate these techniques in the context of military networks next.

6.2.2 Discussion

Comparing Figure 6.1 with Figures 6.2(a), 6.2(b), and 6.2(c), we can see that a gateway is the only of the three approaches that is suitable in a federation of systems. This is because it is the only approach that does not require changes within each respective administrative domain — each domain can use the discovery protocol of their choice. Instead, a gateway must be inserted between the networks in order to translate between protocols. Conversely, an adaptive protocol requires every network to adopt this protocol, requiring that all clients in every network are modified to support this adaptive protocol. The same is true for a layered discovery protocol, where every client must support the common abstraction layer.

Thus, considering the three techniques identified above, we argue that *the gateway approach is best suited for military networks*. The gateway approach will keep costs and complexity low, since the only modification needed is a gateway deployed between networks. NATO has specified the need for network level interoperability points between heterogeneous networks anyway, so that could be a convenient spot for deploying such gateways. Having identified an approach which can enable pervasive service discovery in military networks, we investigate the feasibility of implementing this solution below.

6.3 Towards pervasive service discovery

Above, we identified the gateway approach as the preferred way to achieve pervasive service discovery in military networks. Now we will show that this approach is indeed feasible for discovering Web services through a proof-of-concept implementation and experiments.

First, we implement a gateway which solves interoperability between the experimental and proprietary Mercury protocol and the standardized WS-Discovery mechanism (see Chapter 5). Then, having shown the feasibility of implementing the gateway approach, we extend our prototype with support for more service discovery protocols and test it in an interoperability field trial. This final experiment shows that pervasive service discovery (and invocation) is possible across national and network boundaries in a federation of systems.

6.3.1 Our gateway prototype

The Web services discovery standards can be used in networks with fixed infrastructure. This means that in for example a deployed HQ we can use a registry and WS-Discovery. By allowing the registry to be a discovery proxy (DP) for WS-Discovery, we can let clients discover not only services, but also the registry. However, we need to be able to discover services across network boundaries as

well. This is the requirement for interoperability that we discussed in Section 6.1. As we discussed in Section 6.2, using a service discovery gateway is the most efficient means of achieving this.

By using an interoperability gateway, service discovery is feasible across network boundaries, connecting mobile soldier systems and deployed tactical systems. No modification of the existing services or clients is needed, as the gateway maintains discovery protocol transparency between the two network domains. In this section we will explore the details of our case study of implementing a service discovery gateway solving transparent protocol translation between WS-Discovery and Mercury. We chose this as our initial case study, because WS-Discovery is a standardized mechanism, and Mercury is an experimental discovery solution with very little expressive power, which makes it challenging to employ.

Design

The idea is that the gateway resides on a node that is connected to two different networks with two different service discovery mechanisms. It is based on periodic querying for services in each connected network. The gateway is designed as a simple thread-based discovery message router and translator, finding services in one domain, translating the service descriptions from one type to another and then republishing the service in another domain using its native mechanism. Caching is used to determine if a service is new and should be published, or if a service that was available before has disappeared since the last time the discovery mechanism was invoked. We address these concepts below, where we discuss the details of the implementation.

Implementation

Our interoperability gateway prototype is implemented using Java. In our evaluation of the gateway, it was set up on a Linux machine with two network interfaces, one Ethernet interface (eth0) which was connected to the WS-Discovery enabled domain, and one wireless adapter (wlan0) which was connected to the MANET running Mercury, see Figure 6.3.

This setup required one minor modification to the WS-Discovery library, as it was set up to multicast arbitrarily, meaning that WS-Discovery messages would be multicast on both network interfaces. By changing only a few lines of code in the library, we were able to bind the WS-Discovery multicast socket to interface eth0, thereby making sure that WS-Discovery specific traffic was kept on the LAN only. Mercury, due to its tight coupling with the MANET routing protocol, did not exhibit this behavior, and could be used as it was without any modification.

WS-Discovery allows generic probing (i.e., service queries) and thus the gateway could obtain a complete list of WS-Discovery services from that domain. This list of services could then be parsed, and the service names could be used to publish the services in the Mercury domain. This conversion was easy to achieve, since WS-Discovery retains a lot of information about each service.

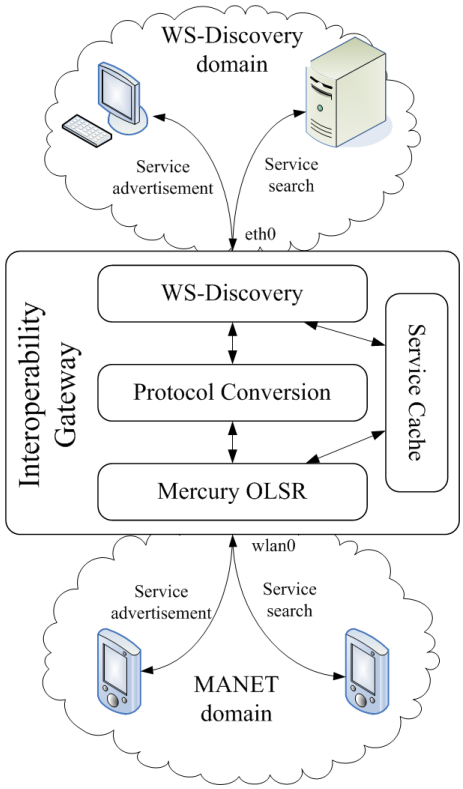


Figure 6.3: The interoperability gateway connects two different service discovery domains. It allows WS-Discovery clients to use Mercury services, and Mercury clients to use WS-Discovery services (from our paper [68]). An updated view of the available services in the network is maintained in the cache.

In the Mercury domain, on the other hand, we were faced with two important challenges:

1. The first challenge was related to querying: Since Mercury is based on Bloom filters [15], you have to search for specific services — you cannot obtain a list of all available services as with WS-Discovery.

2. The second challenge was related to service translation: Mercury describes its services using a simple tuple, i.e., the service name and the IP address where the service can be invoked. This is less information than is needed by WS-Discovery to describe its services. In fact, this is only a part of the dynamic service information.

We addressed these issues as follows: Since Mercury did not support listing all available services, we had to approach that domain differently from that of WS-Discovery. First, we created a list of all names of all supported services in the Mercury domain. Then, we added this list to the query functionality in our gateway, so that instead of issuing a non-specific query, the gateway would search for services from that list in the Mercury domain. This allowed us to find all services in the Mercury domain that had been pre-defined in the gateway. However, the issue of translating these services to WS-Discovery still remained. We needed a way to obtain not only the dynamic aspect of the services, but also the static service information. This meant that our gateway would also have to function as a metadata repository for the known Mercury services, allowing us to supply WS-Discovery with both the static and the dynamic service information that it needed. Having done this, we were able to successfully show the discovery of — and conversion between — arbitrary WS-Discovery services and a set of pre-defined Mercury services.

The gateway used the algorithm shown in Algorithm 1 for each network interface. Basically, the gateway would periodically (e.g., every 30 seconds, as we used in our evaluation, but this is configurable) query all services in the WS-Discovery domain and the specified list of services from the Mercury domain. The services that were found (if any) would then be looked up in the local service cache. We used the local cache to distinguish between services that had been discovered, converted and published before, and new services that had appeared in each domain. If the service was already present in the cache, then it had been converted and published before, and nothing needed to be done. However, if the service was not in the cache, then it would be added to it. The service would then be translated from one service description to the other, and published in the network. Also, during each query iteration we would compare the local cache containing all previously found services with the list of services found now. If any service had disappeared from its domain since last time (i.e., the service was present in the cache but not in the current set of discovered services), then we would delete the service from the other domain as well by using its native service deletion mechanism. After being removed from the network, the service would also be removed from the local cache. This behavior allowed the gateway to mirror the active services from one domain in the other, and remove any outdated information. Thus, assuming that the service discovery mechanism employed in each domain has an up-to-date view of the services in the network, and then this view would be propagated through our interoperability gateway.

Algorithm 1 Service discovery gateway pseudo code.

```
loop
  cache ← CacheOfPublishedServices
  servicesFound ← SearchUsingNativeMechanism
  for all S ∈ servicesFound do {Translate and publish}
    if S ∈ cache then
      {The service is already published}
    else
      cache ← cache + S
      newService ← translate(S)
      publishService(newService)
    end if
  end for
  for all S ∈ cache do {Unpublish services that are unavailable}
    if S ∈ servicesFound then
      {The service is available}
    else
      unpublishService(S)
      cache ← cache − S
    end if
  end for
end loop
```

Discussion

An interoperability gateway is a simple and efficient means of interconnecting two heterogeneous networks, as it provides transparent service discovery description translation from one domain to the other. It is the only of the three approaches we considered in Section 6.2 that also has the benefit of allowing legacy clients and servers to function in their respective domains, since it does not require the systems to be adapted to support a new solution. Each system can continue to use the service discovery mechanism that it is designed to use, and that is best suited to its network.

These considerations formed the basis of our decision to pursue a gateway solution. We were able to successfully show that it is feasible to implement and utilize such an interoperability gateway between two heterogeneous networks using their respective service discovery mechanisms. Thus, we have shown that it is feasible to solve the requirement for interoperability between different service discovery protocols and networks (see Section 6.1).

6.3.2 Interoperability field trial

In Chapter 5 we have shown that it is possible to invoke Web services *in* military networks. At Combined Endeavor (CE), we wanted to explore the use of Web services technology in a combined operation by employing service discovery and invocation both *in and across* heterogeneous military networks. CE is an annual multinational communications exercise. We cooperated with the NC3A at the CE location in the Netherlands in this exercise.

We wanted to employ Web services standards as much as possible, augmenting the system with proprietary, experimental solutions only when necessary. We wanted to test FFI's prototype DSProxy, which can enable COTS Web services clients and services to operate across heterogeneous environments (see Section 4.3.4). By deploying the proxy software in each network node, the proxy can intercept the standard Web services invocation locally. This means that SOAP over HTTP/TCP is used between client and proxy, and between proxy and server. However, the proxy supports compression, multiple transport protocols and adds delay and disruption tolerance, meaning that the communication between the proxies (i.e., invoking Web services) can be performed across heterogeneous networks.

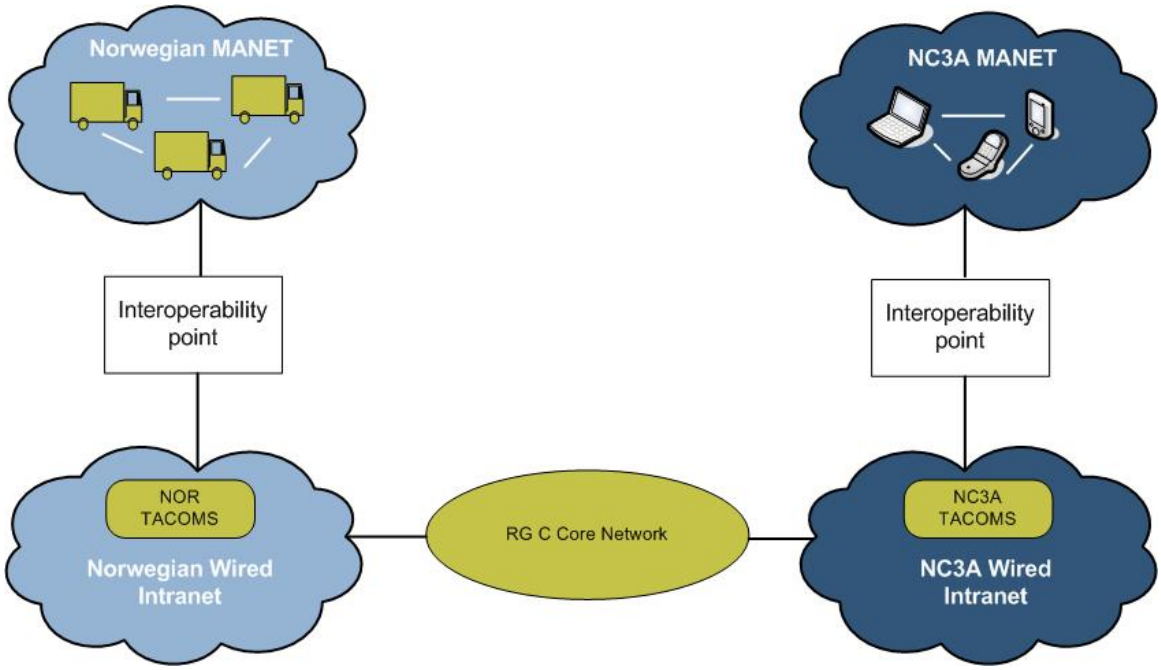
Also, we wanted to achieve pervasive service discovery. In previous experiments [54, 71], we have performed the service discovery at design-time (i.e., the service endpoints used in the experiments have been hard-coded and static in the applications). This way of using Web services is common in civil applications (see Section 3.3), where the services and the network infrastructure are stable. In tactical networks, however, there is a need for run-time discovery, since the dynamic nature of the system means that services are transient. We wanted to explore two cases:

- First, we wanted to show pervasive use of Web services (i.e., discovery and invocation) across network and national boundaries. We used a traditional setup, where direct communication between the two MANETs was not possible (see Figure 6.4(a)). Instead, all communication had to go via the interoperability point between the two HQs. Interoperability between the nations was provided using TACOMS (i.e., a communications standard for joint operations [128]) and the common CE backbone.
- Second, we wanted to try another use case: That of direct interoperability between the two MANETs (see Figure 6.4(b)). This required the use of another interoperability point to connect the two different technologies.

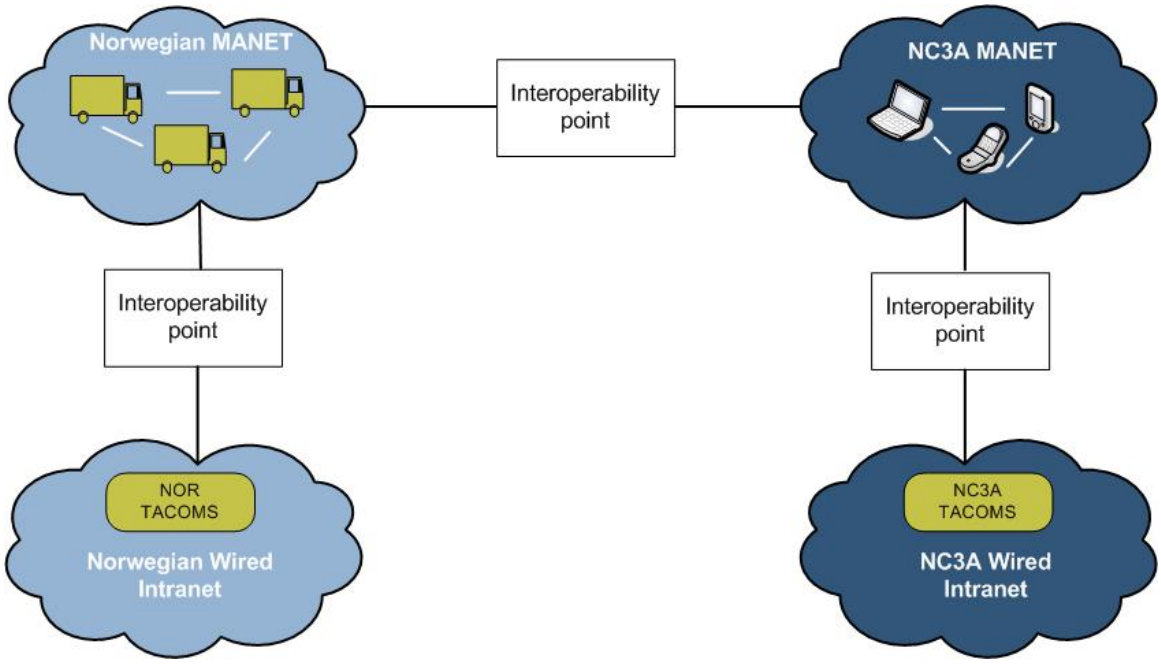
Setup

At CE, we had no strategic network (see the levels in Figure 5.7); we had a setup with two deployed HQs (i.e., tactical deployed networks) and two tactical mobile networks. To address the different characteristics of the networks, we chose to use registries in our HQs and interconnect them using the registries' federation mechanism. This allowed us to perform federated searches, meaning that querying your local registry would propagate the query also to the other registries in the federation. Setting up a registry federation requires the registries to be interoperable. Since NMRR is built on ebXML, we chose to use ebXML in the Norwegian HQ to ease interoperability.

The prototype gateway that we discussed in Section 6.3.1 integrated WS-Discovery with Mercury. This was our original proof-of-concept implementation that was evaluated in the lab. Later, the prototype was extended: At CE, we used it to interconnect WS-Discovery and SAM, integrating the



(a) Experiment run 1: Connection through the backbone (traditional setup).



(b) Experiment run 2: Connection through the MANETs.

Figure 6.4: CE experiment setup.

Norwegian MANET with the Norwegian deployed network. Also, we used it to interconnect SAM with SOP between the Norwegian MANET and the NC3A MANET.

Our first experiment setup at CE was as shown in Figure 6.4(a). We had two MANETs. In the Norwegian MANET, the nodes were vehicles equipped with a tactical radio. These units were mobile, and were out and about reporting incidents (e.g., text and images) back to the base.

To make Web services work in the MANET, we utilized the DSProxy prototype. Furthermore, we used our SAM service discovery mechanism to discover the available services. NC3A used similar techniques in their MANET. In the Norwegian base, we were able to use unmodified Web services and we could also use the standardized discovery mechanisms such as ebXML and WS-Discovery there. The ebXML registry was connected to the NMRR in a registry federation through the RG C core network. Between the Norwegian MANET and Norwegian HQ we had our experimental gateway featuring transparent service discovery protocol translation for interoperability. In this case, the gateway was responsible for interoperability between these two operational levels.

Our second experiment setup was similar, but then the connection between the two HQs was removed. The interoperability between the MANETs was done through direct communication via a second service discovery gateway. This is shown in Figure 6.4(b). In this case, a service discovery gateway was deployed in the interoperability point between the Norwegian MANET and the NC3A MANET as well.

Software

SAM supports Web services discovery in a fully decentralized manner, with support for disseminating position information along with the service advertisements. Based on NFFI, this positioning information can be collected and assembled into full NFFI tracks in the HQ, providing the added value of blue force tracking together with disseminating the Web services information. SAM uses techniques such as data compression, caching and timeouts to minimize the data rate requirements while at the same time addressing the liveness and availability problems. By using this experimental mechanism (Section 5.3.1), we can achieve Web services discovery in our MANET. However, since the mechanism is experimental, it is not interoperable with any of the standardized mechanisms. To address this, we implemented the service discovery gateways discussed above that we placed in the interoperability points between the networks:

- The gateway between the Norwegian MANET and the Norwegian HQ translated between SAM and the standardized WS-Discovery mechanism. This allowed us to be interoperable with a standard, which through a DP could provide further integration with the ebXML registry in our HQ.
- The gateway between the Norwegian MANET and the NC3A MANET translated between SAM and the SOP mechanism NC3A used in their MANET.

We summarize the details of our choice of discovery mechanisms in the Norwegian network in Table 6.1.

Hardware

The Norwegian and NC3A HQs used civil technology: COTS switches, network cables, and routers. Forming two separate networks deployed in two separate tents at the experiment location, these networks were interconnected through the RG C backbone using TACOMS nodes.

The Norwegian MANET consisted of WM600 [79] tactical radios capable of forming a multi-hop MANET. The NC3A used Breadcrumb radios from Rajant [24], which basically are rugged wireless mesh routers using civil 802.11a and b/g technology for communications. Both technologies are IP-enabled and can carry Web services traffic. However, the radios are not compatible on the air, since WM600s typically are configured to use a military frequency, whereas Breadcrumbs use the civil ISM band. WM600 supports IP multicast, meaning that we can use SAM for discovery. The breadcrumbs do not support multicast, but NC3A's SOP relies on unicast to a central node or set of nodes (i.e., JXTA's P2P mechanism).

Execution

Since the MANETs are not compatible on the air, we needed to use an interoperability point. At CE, we solved this by deploying both a WM600 and a Breadcrumb in the Norwegian HQ, where the radios were both connected to a laptop computer. This computer functioned as a router between the two MANETs. This gateway could basically be placed anywhere, it could even have been mobile joining one of the mobile nodes in a MANET. However, for convenience (i.e., continuous power supply for our laptop and radios), we chose to co-locate this gateway with our HQ at CE. Following the same principle, we connected our Norwegian MANET to the HQ. Both the router laptops were running our experimental software: The DSProxy for delay tolerant invocation, and the service discovery gateway software for transparent pervasive service discovery.

The experiment was performed in two iterations:

- First (see Figure 6.4(a)), we used the backbone to communicate between the Norwegian HQ and the NC3A's HQ. There was no connection between the two MANETs in the first iteration.
- Second (see Figure 6.4(b)), we used an interoperability point to facilitate communications directly between the two MANETs. We severed the connection between the two HQs in this experiment.

In both iterations, the participants aimed to solve a simple mission: The NC3A units were scouting an area, and reported an observation into JOCWatch, which was an incident report Web service and database in the NC3A HQ. Upon receiving this incident in the HQ, the NC3A observed the

	WS-Discovery	ebXML	SAM
Category	Decentralized LAN mechanism	Centralized WAN mechanism	Decentralized multi-hop MANET mechanism
Service descriptions	Port types and service names	WSDL and optional metadata	WSDL and optional position and optional metadata
Standardized	Yes	Yes	No, experimental
Operation	Fully decentralized or multicast suppression using central DP.	Centralized registry and repository. Offers federated queries by forwarding queries to other registries.	Fully decentralized using IP multicast.
Suitable in highly dynamic environments	Yes, but only in decentralized mode.	No	Yes
Suitable in disadvantaged grids	No	No	Yes
Application in the Norwegian network at CE	This Web services discovery mechanism is tailored for LAN usage, and is well suited for use in the deployed HQ. It provides integration with the Norwegian registry by wrapping it in a DP.	This registry is used in the Norwegian HQ. It contains all the static Web services offered from the HQ. By connecting the registry to the NMRR we can perform interoperable federated queries.	The mechanism is tailored for disadvantaged grids, where it provides optimized Web services discovery in a fully decentralized manner. We use this mechanism in the Norwegian tactical MANET.

Table 6.1: Service discovery mechanisms in the Norwegian network.

blue force tracking system, and since the Norwegian units were in the area, they were notified via chat (made possible with the DSProxy), and told to investigate. The Norwegian units could then discover the JOCWatch service, connect to it, and download all the details regarding the incident (e.g., description and position). Following this, the units would converge on the position, providing images from the area to the NC3A HQ via a publish/subscribe Web service. A publish/subscribe Web service is discovered in the same manner as a regular request/response service, in that we discover the point to subscribe to. After this point has been discovered we make a subscription, and any new information pushed by this service (in this case, new images) will be delivered to us.

The subscriptions were set up when the Norwegian units were dispatched, and continued for the duration of the experiment. The chat service was also implemented as a publish/subscribe service, where subscriptions were set up prior to starting the experiment. When the units reached the position, they would report back to base via chat. Following chat coordination, the units would return home to base.

The Norwegian units, being in a MANET using SAM for service discovery, were able to provide a periodic update of available services and unit positions. Since SAM provides NFFI position data, we could assemble these positions in the Norwegian HQ and offer an NFFI blue force tracking Web service to the NC3A. The NC3A units, using SOP for service discovery, exposed no positioning information, and we were unable to see their whereabouts during the experiment.

Lessons learned

Unreliable connectivity is a problem for Web services. This can be mitigated by store-and-forward techniques as implemented in the DSProxy. However, with the potential for an unstable network, Web services are not suitable for real-time data.

We have seen that service discovery is possible in and across heterogeneous networks. However, by using a transparent gateway to translate between discovery protocols you may lose some service information going from one network to the other. For example, SAM supports both service and position information, but WS-Discovery supports only service information. This meant that our NFFI tracks had to be assembled and built by the gateway, since this was the point receiving the position information. The NFFI tracks could then be exposed as a Web service. The important thing about using gateways for interoperability is that it is sufficient to know the interface used by another network; you do not need to know the functionality details. This was the case with SOP, where we were able to extract service information, despite being unaware of the NC3A's network topology and how SOP was deployed in their network.

We noticed some issues when using Omar, the open source ebXML reference implementation: First, it was not easy to install. You need several old Java libraries to get it to work, since it is incompatible with some of the newer ones. Thus, you need to use exactly the same library versions that are mentioned on the ebXML website. Second, you have to use Sun's own Java. We attempted to

install Omar on a PC using Ubuntu Linux, and the default Java was OpenJDK. That implementation does not implement security, and therefore compiling Omar failed. After uninstalling OpenJDK and installing Sun's own JDK, we were able to compile Omar. Third, there were issues configuring Omar properly. Omar comes with two graphical user interfaces (GUIs); one Web interface and one Java interface. The Java GUI and the Web GUI support different operation sets. In practice, you need to use both. However, neither of the GUIs set the resource's "HOME" attribute, which is needed in a federation. This attribute tells the registry where the resource belongs. If this attribute is empty, then all responses in a federated query will be treated as if the resources belong to the local registry. If this is not the case, then looking at the XML artifacts will fail, since the identifier will not be resolved to the proper repository's address. To overcome this, we had to update the repository database manually, since neither of the provided GUIs supported setting the HOME attribute. This is problematic, but if you want to use Omar you have to either live with it or write a new GUI that supports all the necessary functionality. The NC3A had remedied this situation by creating NMRR — their GUI to ebXML.

Our use of registries shows that they can be employed in the deployed HQ, and they can also be used in a federation between HQs. The NC3A has shown that P2P can be employed (i.e., the SOP in their network), and while this technology is mostly suitable in large fairly static networks, it can also be employed to some degree in dynamic networks. In highly dynamic networks fully decentralized mechanisms should preferably be used, since they address the aspect of service availability and liveness. We addressed these issues by using our experimental SAM mechanism in our MANET.

Interoperability between heterogeneous networks and mechanisms can be achieved by:

- Using service discovery gateways which translate between discovery protocols, and
- deploying proxies that optimize service invocation across the networks.

The issues we encountered with the ebXML reference implementation clearly show that while standards are important for *interoperability*, the maturity of the available products is equally important for system *usability*.

6.4 Claim review

In Chapter 5, we saw that different networks require different service discovery solutions. This means that in a heterogeneous environment we may encounter different discovery solutions that are incompatible with each other. In this chapter we have investigated techniques for achieving pervasive service discovery in military networks. Now we discuss our chosen technique, service discovery gateways, in the context of the requirements from Table 3.1.

Our first premise was that *a solution must support a federation of systems.* Provided we implement service discovery gateways, they can be employed in a federation of systems. A gateway placed at

a network interoperability point can provide transparent service discovery protocol interoperability between the networks. In fact, a service discovery gateway can be seen as the service discovery counterpart to the Web service proxies we introduced for invocation between networks in Chapter 4.

Our second premise was that a solution *should use Web services technology*. In this chapter we have considered how we can discover Web services across heterogeneous networks using different discovery protocols, and how we can achieve transparent interoperability between these protocols. By using service discovery gateways we were able to achieve pervasive Web services discovery across national and network boundaries.

To ease interoperability we should *always use standards when possible*. In Chapter 5 we saw that it was possible to use the Web services discovery standards in the tactical deployed networks. The disadvantaged grids, on the other hand, had to use proprietary mechanisms. However, the gateway approach can provide compatibility with the existing standards (like we did when we translated between WS-Discovery and proprietary protocols in our proof-of-concept prototype). This means that we can use standards when possible, and fall back on proprietary solutions when we have no other choice.

The set of requirements specific to tactical mobile networks were support for: *Radio silence, limited bandwidth, disruptions,* and *low MTUs*. Also, one *should not rely on TCP or IP fragments*. The gateway approach does not in itself address these requirements. Instead, we should implement support in the gateway for service discovery protocols that can function in tactical networks. In our experiments we used SAM in our disadvantaged grid, and that protocol solves all these requirements. In the tactical deployed network these requirements did not have to be met, so there we could use WS-Discovery.

In the previous chapter, we saw that the Web services standards do not necessarily function in tactical networks. In this chapter we have shown that it is possible to discover Web services across heterogeneous networks, provided that service discovery gateways are deployed at the network interoperability points. NATO has specified network level interoperability points providing IP support across networks. However, we have seen that we need application level service discovery gateways to provide application level interoperability, i.e., transparent translation between different service discovery protocols. Thus, we have proved claim #3, that *in a federation of systems there is a need for application level interoperability points for service discovery.*

6.5 Summary

In this chapter we have addressed claim #3. Interoperability is important and, to achieve pervasive service discovery, we have identified three approaches in this chapter: Adaptive protocols, layered protocols, and service discovery gateways. We argue that gateways are the simplest and most cost-efficient way of achieving transparent service discovery interoperability between heterogeneous networks. Also, it is the only approach suitable for deployment in a federation of systems.

First, we implemented a proof-of-concept prototype gateway solving transparent interoperability between Mercury and WS-Discovery. The solution was promising, and we learned that the gateway should function as a repository for "missing" information. Then, it is possible to translate from a protocol with very limited expressive power such as Mercury to a rich protocol such as WS-Discovery.

Finally, we tried our experimental SAM and interoperability gateway solutions in an operational experiment, and found that it is indeed feasible to achieve pervasive service discovery across heterogeneous networks. We found that ebXML could be used in and between the tactical deployed networks, since its federation mechanism is based on query forwarding rather than replicating data. We have not experimented with UDDI, but since UDDI federations are based on replication, we would expect such federations to suffer from liveness problems in addition to having a lot of transmission overhead in keeping the replication mechanism up-to-date. Since none of the Web services discovery standards are suitable for disadvantaged grids, proprietary solutions must be used in the mobile tactical networks. However, even if there may be no need for such a solution in civil systems, NATO could investigate specifying a STANAG addressing discovery in such networks. Currently, FFI is participating in NATO initiatives such as IST-090 [5] and the Core Enterprise Services Working Group (CESWG), where standardization of discovery mechanisms is an issue. We have shown that there is a need for application level interoperability points, and that could also be subject to standardization in these working groups. At CE we have shown that we can discover and invoke Web services in and across heterogeneous military networks — not only across the national operational levels, but also for interoperability with other nations in a coalition (NATO being represented by NC3A in the experiments). In the next chapter, we discuss our findings in a broader perspective.

7 Conclusion

In Chapters 4, 5, and 6 we discussed our findings and the research claims, and concluded, in the claim review section of each chapter, that we succeeded in what we set out to prove. In this chapter, we conclude the report by summarizing our contribution and by presenting the open issues we have identified for future investigation.

7.1 Contribution

The goal of this report was to achieve pervasive Web services discovery and invocation in and across heterogeneous military networks.

We have investigated several aspects of adapting Web services technology for use in military networks. At NATO CWID we experimented with Web services in an emulated disadvantaged grid. We SOA-enabled a legacy system with Web services, allowing it to provide NFFI tracks to a central HQ which could then visualize the track information and build a common operational picture. The central HQ was connected to our experiment partners. In other words, we used Web services for point-to-point connections between different systems. In these experiments, we found that using optimizations such as XML compression and optimized transport protocols, as well as store-and-forward functionality were a necessity in order to enable Web services in disadvantaged grids.

Further, we have surveyed central Web services standards and specifications, and found that Web services are well suited for building not only traditional "pull" type systems (i.e., request-response operations), but also "push" type systems (i.e., event driven operations). Also, we investigated proxy servers (i.e., intermediate nodes between clients and services) and found that they can be used to provide added value operations, such as compression, content filtering, and so on.

A key concern when adopting NBD is to keep costs down by using COTS technology when possible. Some of the techniques discussed in this report break Web services standards, but this is necessary to get Web services to work in disadvantaged grids. By implementing the optimizations in proxies, we can continue to implement and use COTS technology in clients and servers. The proxies intercept standard Web services and perform the necessary optimizations on inter-proxy traffic.

The most common way of implementing Web services is by using HTTP/TCP for transport. Since TCP does not work well in many disadvantaged grids due to low data rates, varying throughput, disruptions, and high error rates, we have investigated and shown that it is possible to use military message handling systems (STANAG 4406) as a carrier for the Web services protocol SOAP. Using the XML version of NFFI (draft STANAG 5527) as a case study, we have investigated optimizing the information overhead by performing application specific content filtering. Content filtering reduces the total information overhead, leading to less information that needs to be transmitted across the network. In addition, we have investigated various ways of reducing the XML overhead,

by comparing the compression performance of several algorithms. We found that a generic compression algorithm like GZIP compresses XML well, but that the emerging W3C standard for XML compression, EFX, performs slightly better than GZIP. Being an XML conscious compression technique, EFX uses knowledge of the XML structure to perform its compression, thus having an advantage over the generic algorithms. No matter which algorithm we use, we found that compression in some form should definitely be employed, since it significantly reduces the size of XML documents.

Using Web services as a means of integrating stove pipe systems is a requirement of NBD, and we have attempted that in the MNII joint national experiment. There we were able to interconnect systems from the navy and air force, as a part of a cooperative electronic support measures operation. Using Web services as a means of integration and interoperability, we could show the added value of employing Web services in military operations. We found that standardized Web services discovery mechanisms do not necessarily function well in disadvantaged grids (we attempted to use WS-Discovery, but found that it generated too much traffic in our network, flooding buffers and disrupting other traffic).

Service discovery is important in dynamic environments because services can come and go, and we need to know which services are available at any time. We discussed the requirements and challenges of service discovery in different military networks. We concluded that due to the diversity of the networking technologies used in military networks, one single mechanism cannot be used in all networks. A toolkit of different mechanisms is needed, where the mechanism that is best suited is used in each network. By doing this (for example by using specially optimized solutions such as our experimental SAM in disadvantaged grids) we can solve the problem of service discovery in military networks. However, interoperability is a key concern, so there is also a need for pervasive service discovery across heterogeneous networks. We investigated pervasive service discovery and concluded that using gateways for interoperability is the simplest and most cost-efficient means of achieving the needed protocol interoperability. The gateways must be placed in the connection points between heterogeneous networks (i.e., the so-called interoperability points that the NNEC feasibility study discusses).

We reached our goals. We were able to prove the three research claims in Section 1.5:

1. To be able to invoke Web services in military networks with constraints, adapting the standards is necessary.

2. Networks with different properties require different discovery mechanisms.

3. In a federation of systems there is a need for application level interoperability points for service discovery.

7.2 Future work

In this report, we have shown that it is feasible to advertise, discover, and invoke Web services in military networks. However, there are still many open issues that need to be addressed. In this section we present ideas that arose during the work with this report, but that we were unable to pursue within the limited time frame of FFI project 1086. These ideas are subject to future work in project 1176.

7.2.1 Architectural concerns

In order to implement a complete SOA for military networks, one needs to address not only the technical but also the management issues of Web services. We have not investigated issues such as service level agreements, governance, etc. that would need to be considered in a production system.

7.2.2 QoS

QoS is one of the most important aspects of Web services governance. Military applications require end-to-end QoS. There is QoS support in the networks, since Intserv [17] and Diffserv [14] are often used in military networks today. However, there is a need to bring this out into the end applications and the middleware. Thus, we need support for prioritization and preemption of SOAP messages. We also need to map the model to the lower network layers, so that QoS specified in the middleware can be enforced by network mechanisms (e.g., mapping Web services QoS requirements to Diffserv classes [87]). Some of these QoS aspects are currently being addressed at FFI (see [77]), but many open issues still remain that need to be investigated.

7.2.3 UDDI

In this report, we focused on ebXML in preference of UDDI because ebXML has better query capabilities and a more flexible federation mechanism than UDDI: ebXML supports federated queries, meaning that the ebXML registries in a federation can be seen as the backbone of a structured peer-to-peer overlay network. Queries and responses are forwarded through this overlay. If a registry becomes unavailable, then new federated queries will not list the services from the now unavailable registry, thus providing a coarse liveness support. UDDI, on the other hand, will replicate data between the registries in the federation, thus requiring explicit cleanup if one member registry and its affiliated services disappear. These properties, and the fact that ebXML has better support for semantic descriptions than UDDI, made us conclude that ebXML should be the registry of choice for interoperability in NATO coalition forces.

However, recently it seems that a shift of focus in NATO has started: The NATO Core Enterprise Services Working Group (CESWG) has started leaning towards recommending UDDI v2 and v3 as the interface for NATO's planned Service Discovery Service (SDS) [35]. The CESWG does not

exclude ebXML entirely, stating that perhaps future implementations of the SDS should support ebXML and WS-Discovery as well. Their incentive for choosing UDDI over ebXML in their work is that UDDI is more mature (since it is addressed by the WS-I, interoperability can be ensured between different vendors). In fact, their main incentive for mentioning future ebXML support is because the NMRR is built on that standard. So far the CESWG has not started addressing the needs of disadvantaged grids.

Considering this, future work should also investigate UDDI further in a military context.

7.2.4 Further experiments in disadvantaged grids

Web services technology is a great means of implementing interoperable systems for NATO, but the optimizations necessary for each sub-system is something that each nation needs to explore in the context of their own current equipment. It is desirable to continue experimenting in this fashion, and try other communications hardware as well in future experiments. Only by using the actual equipment can we uncover all the specific limitations related to that particular technology, which in turn is necessary in order to suggest specific optimization techniques or discovery protocols for that network.

Our approach to optimizations has been broad, in that we survey and apply several different techniques to achieve our goal: Using Web services technology in military networks. We have successfully demonstrated these techniques in several large military experiments. However, due to the diversity of the military communications technologies, we have been able to suggest a few specific optimizations suitable for a few specific networking technologies.

Further work related to service discovery could be adding support for Efficient XML to WS-Discovery. Efficient XML is a legal representation of XML "on the wire", allowing it to be applied to SOAP messages and still be compliant with the SOAP standard. This means that attempting to use it together with WS-Discovery could be beneficial, in that it would reduce the overhead of the discovery protocol. Thus, it would be worth experimenting with this combination to see if it could potentially be usable in some disadvantaged grids.

7.2.5 Semantic Web services

Service discovery is the process of finding and identifying a service in a network. Service discovery can be divided into two categories [45]:

1. Manual discovery, and
2. autonomous discovery.

Under manual discovery, a requester human uses a discovery service to locate and select a service description that meets the desired functional (and other) criteria. Manual discovery is typically used at design-time. Under autonomous discovery, a requester agent performs this task, either at design-time or run-time. The steps are similar in either case, but the constraints and needs are different:

- The interface requirements for something that is intended for human interaction are different from the requirements for something that is intended for machine interaction.
- There is less of a need to standardize an interface or protocol that humans use to communicate with humans, compared to those interfaces and protocols intended for use by machines.
- There is also an issue of trust — people do not necessarily trust machines to make decisions that may have significant consequences. Trust needs to be addressed along with other security related issues.

As such, one can consider the autonomous discovery process as being a subset of the manual discovery process in terms of functionality. In the case of autonomous discovery, there is also a need for machine-processable semantics. This calls for *Semantic Web services* [20]:

> Web services + semantic technologies = Semantic Web services

Semantic Web services are Web services *annotated* with semantics. The annotation terms adhere to formal terminologies (i.e., ontologies) which include service description, provider details, service operations, service execution model, service parameters, service data flow, service invocation details, etc. See our paper [72] for further thoughts on how Semantic Web services can facilitate interoperability.

7.3 Summary

In this chapter, we have seen that it is both feasible and desirable to use Web services in military networks. We have been able to both discover and invoke Web services in and across heterogeneous military networks by applying different optimization techniques. Though many issues remain open, we have solved some very fundamental aspects of using Web services in military networks in this report.

Appendix A List of abbreviations

C2IEDM	Command and Control Information Exchange Data Model
COTS	Commercial Off-The-Shelf
CWID	Coalition Warrior Interoperability Demonstration
DSProxy	Delay and Disruption Tolerant SOAP Proxy
ebXML	electronic business using XML
GUI	Graphical User Interface
HTTP	Hypertext Transfer Protocol
IP	Internet Protocol
LAN	Local Area Network
MANET	Mobile Ad Hoc Network
MIP	Multilateral Interoperability Programme
MMHS	Military Message Handling System
MTOM	Message Transmission Optimization Mechanism
NC3A	NATO C3 Agency
NDB	Network Based Defense
NFFI	NATO Friendly Force Information
NNEC	NATO Network Enabled Capability
OASIS	Organization for the Advancement of Structured Information Standards
P2P	Peer-to-Peer
QoS	Quality of Service
SAM	Service Advertisements in MANETs
SMTP	Simple Mail Transfer Protocol
SOA	Service-Oriented Architecture
SOAP	Called the "Simple Object Access Protocol" up to and including version 1.1, but from version 1.2 the name is just "SOAP".
STANAG	NATO Standardized Agreement
TCP	Transmission Control Protocol
UDDI	Universal Description Discovery and Intergration
UDP	User Datagram Protocol
UIDM	Universal Improved Data Modem
W3C	World Wide Web Consortium
WAN	Wide Area Network
WS-Discovery	Web Services Dynamic Discovery
WSDL	Web Services Description Language
WS-I	Web Services Interoperability Organization
WSN	Web Services Notification (WS-Notification)
WSQM	Web Services Quality Model
WWW	World Wide Web
XML	Extensible Markup Language

References

[1] Agile Delta. Integrate efficient XML into your application without changing code. http://www.agiledelta.com/product_proxies.html, accessed 2009-12-31.

[2] J. Allard, V. Chinta, S. Gundala, and G. G. Richard III. Jini Meets UPnP: An Architecture for Jini/UPnP Interoperability. In *SAINT '03: Proceedings of the 2003 Symposium on Applications and the Internet*, Washington, DC, USA, 2003.

[3] G. Alonso, F. Casati, H. Kuno, and V. Machiraju. Web services concepts, architectures and applications. Springer-Verlag, 2004.

[4] M. Amoretti, F. Zanichelli, and G. Conte. SP2A: a service-oriented framework for P2P-based grids. In *proceedings of the 3rd International Workshop on Middleware for Grid Computing (MGC05), Grenoble, France*, 2005.

[5] F. Annunziata, B. Ardic, X. Denis, G. Fletcher, T. Hafsøe, I. Hernández Novo, N. Jansen, F. T. Johnsen, P.-P. Meiler, I. Owens, B. Sasioglu, J. Sliwa, J. Stavnstrup, and A. Tokuz. IST-090 SOA challenges for disadvantaged grids. 15th International Command and Control Research and Technology Symposium (ICCRTS), Los Angeles, CA, USA, July 2010.

[6] N. Apte, K. Deutsch, and R. Jain. Wireless SOAP: Optimizations for mobile wireless web services. *WWW 2005, Chiba, Japan*, May 2005.

[7] C. J. Augeri, B. E. Mullins, L. C. Baird, D. A. Bulutoglu, and R. O. Baldwin. An analysis of XML compression efficiency. *Workshop on Experimental Computer Science (ExpCS '07)*, 2007.

[8] D. E. Bakken. Middleware. Chapter in Encyclopedia of Distributed Computing, Urban, J. and Dasgupta, P. (eds.), Kluwer academic press., 2001.

[9] K. Ballinger, P. Brittenham, A. Malhotra, W. A. Nagy, and S. Pharies. Web services inspection language (ws-inspection) 1.0. http://download.boulder.ibm.com/ibmdl/pub/software/dw/specs/ws-wsilspec/ws-wsilspec.pdf, accessed 2009-12-28.

[10] P. Bartolomasi, T. Buckman, A. Campbell, J. Grainger, J. Mahaffey, R. Marchand, O. Kruidhof, C. Shawcross, and K. Veum. NATO network enabled capability feasibility study. Version 2.0, October 2005.

[11] D. Benslimane, S. Dustdar, and A. Sheth. Services mashups: The new generation of web applications. In *IEEE Internet Computing, vol. 12, no. 5*, pages 13–15, 2008.

[12] S. Berger, S. McFaddin, C. Narayanaswami, and M. Raghunath. Web services on mobile devices — implementation and experience. *in proceedings of the Fifth IEEE Workshop on Mobile Computing Systems and Applications*, 2003.

[13] W. Bethea, R. Cole, and P. Harshavardhana. Automated discovery of information services in heterogeneous distributed networks. In *Military Communications Conference, 2008. MILCOM 2008. IEEE*, pages 1–6, 2008.

[14] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss. An Architecture for Differentiated Service. RFC 2475 (Informational), Dec. 1998. Updated by RFC 3260.

[15] B. H. Bloom. Space/time trade-offs in hash coding with allowable errors. *Communications of the ACM*, 13(7):422–426, 1970.

[16] C. Bormann, C. Burmeister, M. Degermark, H. Fukushima, H. Hannu, L.-E. Jonsson, R. Hakenberg, T. Koren, K. Le, Z. Liu, A. Martensson, A. Miyazaki, K. Svanbro, T. Wiebke, T. Yoshimura, and H. Zheng. RObust Header Compression (ROHC): Framework and four profiles: RTP, UDP, ESP, and uncompressed. RFC 3095 (Proposed Standard), July 2001. Updated by RFCs 3759, 4815.

[17] R. Braden, D. Clark, and S. Shenker. Integrated Services in the Internet Architecture: an Overview. RFC 1633 (Informational), June 1994.

[18] Y. Bromberg, V. Issarny, and P. Raverdy. Interoperability of Service Discovery Protocols: Transparent versus Explicit Approaches. *15th IST Mobile and Wireless Communication Summit, Myconos, Greece*, June 2006.

[19] C. Campo, C. García-Rubio, A. M. López, and F. Almenárez. PDP: a lightweight discovery protocol for local-scope interactions in wireless ad hoc networks. *Comput. Networks*, 50(17):3264–3283, December 2006.

[20] J. Cardoso. *Semantic Web Services: Theory, Tools and Applications*. IGI Global, 2007.

[21] S. Cheshire, B. Aboba, and E. Guttman. Dynamic Configuration of IPv4 Link-Local Addresses. RFC 3927 (Proposed Standard), May 2005.

[22] S. Chesire and D. H. Steinberg. *Zero Configuration Networking*. O'Reilly Media, Inc, 2006.

[23] T. Clausen and P. Jacquet. Optimized Link State Routing Protocol (OLSR). RFC 3626 (Experimental), Oct. 2003.

[24] R. Corporation. Rajant corporation home page. http://www.rajant.com/, accessed 2010-07-04.

[25] J. Crupi and C. Warner. Enterprise mashups part i: Bringing SOA to the people. SOA Magazine Issue XVIII: May, 2008.

[26] R. Cunnings, S. Fell, and P. Kulchenko. SMTP transport binding for SOAP 1.1. http://www.pocketsoap.com/specs/smtpbinding/, accessed 2009-12-26.

[27] F. Curbera, M. Duftler, R. Khalaf, W. Nagy, N. Mukhi, and S. Weerawarana. Unraveling the Web Services Web, An Introduction to SOAP, WSDL, and UDDI. *Internet Computing, IEEE*, 6(2):86–93, April 2002.

[28] Defence R&D Canada Valcartier. High capability tactical communications network-HCTCN TD. DRDC Valcartier Fact Sheet IS-226-A.

[29] P. Deutsch. DEFLATE Compressed Data Format Specification version 1.3. RFC 1951 (Informational), May 1996.

[30] P. Deutsch. GZIP file format specification version 4.3. RFC 1952 (Informational), May 1996.

[31] P. Deutsch and J.-L. Gailly. ZLIB Compressed Data Format Specification version 3.3. RFC 1950 (Informational), May 1996.

[32] S. Dustdar and L. Juszczyk. Dynamic replication and synchronization of web services for high availability in mobile ad-hoc networks. In *Service Oriented Computing and Applications, Vol. 1, No. 1*, pages 19–33, April 2007.

[33] D. Eastlake 3rd and P. Jones. US Secure Hash Algorithm 1 (SHA1). RFC 3174 (Informational), Sept. 2001. Updated by RFC 4634.

[34] Efficient XML Interchange Working Group. Efficient XML interchange working group public page. http://www.w3.org/XML/EXI/, accessed 2009-12-22.

[35] A. Eggen. Discussion about the NATO core enterprise services work group. Personal communication, January 2010.

[36] Elbit Systems of America. Uidm v2 modem for operation centers and vehicles. http://www.innocon.com/PDF/UIDM_V2_2009.pdf, accessed 2009-12-31.

[37] P. E. Engelstad, Y. Zheng, R. Koodli, and C. E. Perkins. Service discovery architectures for on-demand ad hoc networks. *International Journal of Ad Hoc and Sensor Wireless Networks, Old City Publishing (OCP Science)*, 2(1):27–58, March 2006.

[38] T. Erl. *Service-Oriented Architecture — A Field Guide to Integrating XML and Web Services*. Prentice hall, 2004.

[39] T. Erl. *Service-Oriented Architecture — Concepts, Technology, and Design*. Prentice hall, 2005.

[40] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee. Hypertext Transfer Protocol – HTTP/1.1. RFC 2616 (Draft Standard), June 1999. Updated by RFC 2817.

[41] J. Flathagen. An introduction to service discovery in ad hoc networks and tactical soldier networks. FFI-Notat 2007/02004, 2007.

[42] J. Flathagen and L. Olsen. NORMANS KKI. FFI Facts (in Norwegian), http://www.mil.no/multimedia/archive/00086/Faktark-NORMANS-KKI-_86445a.pdf, November 2006.

[43] J. Flathagen and K. Øvsthus. Service Discovery using OLSR and Bloom Filters. In *proceedings of the 4th OLSR Interop & Workshop, Ottawa, Canada*, October 2008.

[44] A. Fongen, M. Gjellerud, and E. Winjum. A military mobility model for manet research. In *Parallel and Distributed Computing and Networks (PDCN 2009), February 16 – 18*, Innsbruck, Austria, 2009.

[45] G. Babakhani et al. Web trends and technologies and NNEC core enterprise services — version 2.0. NATO C3 Agency, Technical Note 1143, December 2006.

[46] T. Gagnes. Assessing dynamic service discovery in the network centric battlefield. In *IEEE Military Communications Conference (MILCOM) 2007, Orlando, FL, USA*, pages 1–7, 2007.

[47] T. Gagnes, T. Plagemann, and E. Munthe-Kaas. A conceptual service discovery architecture for semantic web services in dynamic environments. *22nd International Conference on Data Engineering Workshops (ICDEW'06), Atlanta, GA, USA*, 2006.

[48] Gartner research. Core web service standard UDDI evolves with version 3.0.2. Charles Abrams, Ray Valdes, and David Mitchell Smith, http://www.gartner.com/resources/126100/126170/core_web_servic.pdf, accessed 2009-12-28.

[49] G. Gehlen and R. G. R. Bergs. Performance of mobile web service access using the wireless application protocol (wap). *in Proceedings of World Wireless Congress. San Francisco, CA, USA*, 2004.

[50] A. Gibb. Challenges for middleware imposed by the tactical army communications environment. NATO IST-030/RTG-012 Workshop on 'Role of Middleware in Systems Functioning over Mobile Communication Networks', 2003.

[51] A. Gibb, H. Fassbender, M. Schmeing, J. Michalak, and J. E. Wieselthier. Information management over disadvantaged grids. Final report of the RTO Information Systems Technology Panel, Task Group IST-030 / RTG-012, RTO-TR-IST-030, 2007.

[52] J. Gomez, A. T. Campbell, M. Naghshineh, and C. Bisdikian. Conserving transmission power in wireless ad hoc networks. In *ICNP '01: Proceedings of the Ninth International Conference on Network Protocols*, Washington, DC, USA, 2001. IEEE Computer Society.

[53] L. Gong. JXTA: a network programming environment. *Internet Computing, IEEE*, 5(3):88–95, May/Jun 2001.

[54] R. Haakseth, T. Gagnes, D. Hadzic, T. Hafsøe, F. T. Johnsen, K. Lund, and B. K. Reitan. SOA — cross domain and disadvantaged grids — NATO CWID 2007. FFI report 2007/02301, ISBN 978-82-464-1272-6, 2007.

[55] R. Haakseth, D. Hadzic, K. Lund, A. Eggen, and R. Rasmussen. Experiences from implementing dynamic and secure web services. 11th International Command and Control Research and Technology Symposium (ICCRTS), Cambridge, UK, September 2006.

[56] D. Hadzic, T. Hafsøe, F. T. Johnsen, K. Lund, and K. Rose. Web services in networks with limited data rate. FFI report 2006/03886 (in Norwegian), ISBN 978-82-464-1049-4, 2006.

[57] T. Hafsøe and F. T. Johnsen. Reducing network load through intelligent content filtering. 13th International Command and Control Research and Technology Symposium (ICCRTS), Seattle, WA, USA, June 2008.

[58] T. Hafsøe, F. T. Johnsen, K. Lund, and A. Eggen. Adapting web services for limited bandwidth tactical networks. *12th International Command and Control Research and Technology Symposium (ICCRTS), Newport, RI, USA*, 2007.

[59] T. Hafsøe, F. T. Johnsen, and M. Rustad. Semantically enabled QoS aware service discovery and orchestration for MANETs. 15th International Command and Control Research and Technology Symposium (ICCRTS), Los Angeles, CA, USA, July 2010.

[60] A. Harrison and I. Taylor. WSPeer — An Interface to Web Service Hosting and Invocation. In *HIPS Joint Workshop on High-Performance Grid Computing and High-Level Parallel Programming Models, IPDPS*, Denver, Colorado, USA, 2005.

[61] S. Helal, N. Desai, V. Verma, and C. Lee. Konark — a service discovery and delivery protocol for ad-hoc networks. *Proceedings of the Third IEEE Conference on Wireless Communication Networks (WCNC), New Orleans*, 2003.

[62] M. N. Huhns and M. P. Singh. Service-oriented computing: key concepts and principles. *Internet Computing, IEEE*, 9(1):75–81, Jan-Feb 2005.

[63] IBM et al. Web services reliable messaging protocol (ws-reliablemessaging). http://download.boulder.ibm.com/ibmdl/pub/software/dw/specs/ws-rm/ws-reliablemessaging200502.pdf, accessed 2009-12-22.

[64] J. L. Jodra, M. Vara, J. M. Cabero, and J. Bagazgoitia. Service discovery mechanism over OLSR for mobile ad-hoc networks. *Advanced Information Networking and Applications, AINA*, 2:534–542, 2006.

[65] F. T. Johnsen, A. Eggen, T. Hafsøe, and K. Lund. Publications related to nato cwid 2007 soa experiments. FFI report 2008/01430, 2008.

[66] F. T. Johnsen, A. Eggen, T. Hafsøe, and K. Lund. Utilizing military message handling systems as a transport mechanism for soa in military tactical networks. *NATO IST-083 Symposium on Military Communications with a special focus on Tactical Communications for Network Centric Operations, Prague, Czech republic*, April 2008.

[67] F. T. Johnsen, J. Flathagen, T. Gagnes, R. Haakseth, T. Hafsøe, J. Halvorsen, N. A. Nordbotten, and M. Skjegstad. Web services and service discovery. FFI report 2008/01064, 2008.

[68] F. T. Johnsen, J. Flathagen, and T. Hafsøe. Pervasive service discovery across heterogeneous tactical networks. In *IEEE Military Communications Conference (MILCOM 2009)*, pages 1–8, Oct. 2009.

[69] F. T. Johnsen, J. Flathagen, T. Hafsøe, M. Skjegstad, and N. Kol. Interoperable service discovery: Experiments at combined endeavor 2009. FFI report 2009/01934, 2009.

[70] F. T. Johnsen, T. Hafsøe, and K. Lund. *Quality of Service considerations for Network Based Defence*. FFI report 2006/03859, 2006.

[71] F. T. Johnsen, T. Hafsøe, E. Skjervold, K. Rose, K. Lund, and N. A. Nordbotten. Multinett II : SOA and XML security experiments with Cooperative ESM Operations (CESMO). FFI report 2008/02344, ISBN: 978-82-464-1504-8, 2008.

[72] F. T. Johnsen, M. Rustad, T. Hafsøe, A. Eggen, and T. Gagnes. Semantic service discovery for interoperability in tactical military networks. The International C2 Journal, SPECIAL ISSUE: Agility and Interoperability for 21st Century Command and Control, February 2010.

[73] L. Johnsrud, D. Hadzic, T. Hafsøe, F. T. Johnsen, and K. Lund. Efficient web services in mobile networks. *The 6th IEEE European Conference on Web Services (ECOWS), Dublin, Ireland*, November 2008.

[74] S. H. Kang, S. Ryu, N. Kim, Y. Lee, D. Lee, and K. Moon. An Architecture for Interoperability of Service Discovery Protocols Using Dynamic Service Proxies. *Information Networking*, pages 786–795, 2005.

[75] V. Kawadia and P. R. Kumar. A cautionary perspective on cross layer design. IEEE Wireless Commun., vol 12, number 1, February 2005.

[76] N. Kol. NATO metadata registry and repository. http://si.nc3a.nato.int/projects/metadata-registry, accessed 2010-01-24.

[77] Ø. Kolbu. QoS related admission control for web services. Master's thesis, FFI/University of Oslo, Norway, work in progress (due for delivery February 2011).

[78] Kongsberg Defence and Aerospace. NorTac-C2IS. http://www.kongsberg.com/eng/kda/products/Armyc2is/NORTaC-C2IS/, accessed 2008-05-20.

[79] Kongsberg Defence Systems. WM600 datasheet. http://www.kongsberg.com/en/KDS/Products/~/media/KDS/Files/Products/Defence%20Communication/wm600_datasheet_rev_rc_small.ashx, accessed 2009-05-01.

[80] R. Lausund and S. Martini. Norwegian modular network soldier (NORMANS). FFI Facts, http://www.mil.no/multimedia/archive/00086/FFI-FACTS-NORMANS-EN_86447a.pdf, November 2006.

[81] K. Lund, T. Hafsøe, and F. T. Johnsen. *A survey of middleware with focus on application in network based defence*. FFI report 2007/02683, 2007.

[82] K. Lund, T. Hafsøe, F. T. Johnsen, E. Skjervold, L. Schenkels, and J. Busch. Information exchange in heterogeneous military networks. FFI report 2009/02289, 2009.

[83] D. Marco-Mompel. Service oriented peer prototype for mobile users. NC3A Technical Note Draft under project SPW001495, November 2007.

[84] MeshDynamics, Inc. Performance analysis of three mesh networking architectures. http://www.meshdynamics.com/performance-analysis.html, accessed 2010-02-23.

[85] A. N. Mian, R. Baldoni, and R. Beraldi. A survey of service discovery protocols in multihop mobile ad hoc networks. In *IEEE Pervasive computing*, pages 66–74, January-March 2009.

[86] J. E. Miller and J. Dussault. Accessing and sharing: Facets of addressing information overload. 11th International Command and Control Research and Technology Symposium (ICCRTS), Cambridge, UK, 2006.

[87] T. Min. *QoS integration in Web services with the WS-QoS framework*. PhD thesis, Freie Universität Berlin, 2005.

[88] Y. Natis. Gartner research "service-oriented architecture under the magnifying glass". Application Integration & Web Service, Summit 2005, April 18-20, 2005.

[89] Network Centric Operations Industry Consortium. NCOIC — home. https://www.ncoic.org/home, accessed 2009-12-31.

[90] W. Ng, W.-Y. Lam, and J. Cheng. Comparative analysis of XML compression technologies. In *World Wide Web 9(1), Kluwer Academic Publishers*, pages 5–33, March 2006.

[91] N. A. Nordbotten. *XML and Web Services Security*. FFI report 2008/00413, 2008.

[92] OASIS. UDDI Version 3.0.2, UDDI Spec Technical Committee Draft. Luc Clement, Andrew Hately, Claus von Riegen, and Tony Rogers (eds.), http://uddi.org/pubs/uddi_v3.htm, 2004.

[93] OASIS. WS-Notification (2006) tc. http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsn, accessed 2009-12-22.

[94] OASIS. Quality model for web services working draft september 2005. Eunju Kim, and Youngkon Lee (eds.), http://www.oasis-open.org/committees/download.php/15910/WSQM-ver-2.0.doc, accessed 2009-12-28.

[95] OASIS. Web services business process execution language version 2.0 OASIS standard 11 april 2007. Alexandre Alves, Assaf Arkin, Sid Askary, Charlton Barreto, Ben Bloch, Francisco Curbera, Mark Ford, Yaron Goland, Alejandro Guízar, Neelakantan Kartha, Canyang Kevin Liu, Rania Khalaf, Dieter König, Mike Marin, Vinkesh Mehta, Satish Thatte, Danny van der Rijn, Prasad Yendluri, and Alex Yiu (eds.), http://docs.oasis-open.org/wsbpel/2.0/wsbpel-v2.0.html, accessed 2009-12-28.

[96] OASIS. Web services dynamic discovery (ws-discovery) version 1.1 OASIS standard 1 july 2009. Vipul Modi, and Devon Kemp (eds.), http://docs.oasis-open.org/ws-dd/discovery/1.1/wsdd-discovery-1.1-spec.pdf, accessed 2009-12-28.

[97] OASIS. OASIS web services transaction (ws-tx) tc. http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=ws-tx, accessed 2010-01-03.

[98] OASIS. Reference model for service oriented architecture 1.0 OASIS standard, 12 october 2006. C. Matthew MacKenzie, Ken Laskey, Francis McCabe, Peter F. Brown, and Rebekah Metz (eds.), http://docs.oasis-open.org/soa-rm/v1.0/soa-rm.pdf, accessed 2010-01-03.

[99] OASIS. ebXML registry information model version 3.0 OASIS standard, 2 may, 2005. Sally Fuger, Farrukh Najmi, Nikola Stojanovic (eds.), http://docs.oasis-open.org/regrep/v3.0/specs/regrep-rim-3.0-os.pdf, accessed 2010-01-17.

[100] OASIS. SOAP-over-UDP version 1.1 OASIS standard 1 july 2009. Ram Jeyaraman (ed.), http://docs.oasis-open.org/ws-dd/soapoverudp/1.1/os/wsdd-soapoverudp-1.1-spec-os.pdf, accessed 2010-02-23.

[101] A. Obaid, A. Khir, and H. Mili. A Routing Based Service Discovery Protocol for Ad hoc Networks. In *Proceedings of the Third International Conference on Networking and Services, ICNS '07, Athens, Greece*, 2007.

[102] L. E. Olsen and J. Flathagen. A study of computer interfaces for soldier systems (A look at Ethernet, FireWire and USB). FFI Report 2005/03043, 2005.

[103] C. Perkins, E. Belding-Royer, and S. Das. Ad hoc On-Demand Distance Vector (AODV) Routing. RFC 3561 (Experimental), July 2003.

[104] L. Pham and G. Gehlen. Realization and performance analysis of a SOAP server for mobile devices. *The 11th European Wireless Conf. 2005, vol. 2, Nicosia, Cyprus, VDE Verlag*, 2005.

[105] T. Podlasek, J. Sliwa, and M. Amanowicz. Efficiency of compression techniques in SOAP. Military Communications and Information Systems Conference (MCC), Wroclaw, Poland, September 2010.

[106] R. Porta. Friendly force information sharing — lessons learned and way towards NNEC. Presentation at the 7th NATO CIS Symposium, Prague, Czech Republic, October 2008.

[107] J. Postel. Internet Protocol. RFC 791 (Standard), Sept. 1981. Updated by RFC 1349.

[108] J. Postel. Transmission Control Protocol. RFC 793 (Standard), Sept. 1981. Updated by RFCs 1122, 3168.

[109] R. Faucher et al. Guidance on proxy servers for the tactical edge. The MITRE corporation, MITRE Technical Report, MTR 060175, September 2006.

[110] M. Rabinovich and O. Spatscheck. *Web caching and replication*. Addison Wesley, 2002.

[111] P.-G. Raverdy, V. Issarny, R. Chibout, and A. de La Chapelle. A multi-protocol approach to service discovery and access in pervasive environments. *Annual International Conference on Mobile and Ubiquitous Systems, San Jose, CA, USA*, 0:1–9, 2006.

[112] G. G. Richard III. *Service and Device Discovery: Protocols and Programming*. Mc Graw Hill, 2002.

[113] L. Richardson and S. Ruby. *RESTful Web Services*. O'Reilly Media, 2007.

[114] K. Rose, T. Hafsøe, and K. Lund. Composition of web services. FFI Report 2009/01699 (NATO Unclassified), 2009.

[115] F. Sailhan and V. Issarny. Scalable service discovery for MANETs. In *Third IEEE International Conference on Pervasive Computing and Communications (PERCOM)*, pages 235–244. IEEE Computer Society, March 2005. ISBN 0-7695-2299-8.

[116] D. Salomon. *Data Compression — The Complete Reference, 2nd edition*. Springer, 2000.

[117] J. Schlimmer (editor). Web Services Dynamic Discovery (WS-Discovery). http://specs.xmlsoap.org/ws/2005/04/discovery/ws-discovery.pdf, April 2005.

[118] K. Scott. Disruption tolerant networking proxies for on-the-move tactical networks. In *IEEE MILCOM 2005, Vol 5*, pages 3226 — 3231, October 2005.

[119] S. A. H. Seno, R. Budiarto, and T.-C. Wan. Survey and new approach in service discovery and advertisement for mobile ad hoc networks. *International Journal of Computer Science and Network Security (IJCSNS), VOL. 7, No.2*, February 2007.

[120] A. Sheth. Web services to semantic web processes: Investigating synergy between practice and research. The First European Young Researchers Workshop on Service Oriented Computing (Keynote Address), Leicester, U.K., 2005.

[121] M. Skjegstad and F. T. Johnsen. Search+: An efficient peer-to-peer service discovery mechanism. FFI Report 2009/01610, 2009.

[122] M. Skjegstad, U. Roedig, and F. T. Johnsen. Search+: A resource efficient peer-to-peer service discovery mechanism. IEEE MILCOM, Boston, MA, USA, October 2009.

[123] E. Skjervold, T. Hafsøe, F. T. Johnsen, and K. Lund. Publish/subscribe with COTS web services across heterogeneous networks. IEEE 8th International Conference on Web Services Computing (ICWS 2010), Miami, FL, USA, July 2010.

[124] E. Skjervold, T. Hafsøe, F. T. Johnsen, and K. Lund. Delay and disruption tolerant web services for heterogeneous networks. IEEE MILCOM, Boston, MA, USA, October 2009.

[125] V. Srivastava and M. Motani. Cross-layer design: a survey and the road ahead. *Communications Magazine, IEEE*, 43(12):112–119, 2005.

[126] J.-C. St-Jacques. Challenges for a distributed collaborative environment functioning over mobile wireless networks. NATO IST-030/RTG-012 Workshop on 'Role of Middleware in Systems Functioning over Mobile Communication Networks', 2003.

[127] J. A. Stine. Cross-Layer Design of MANETs: The Only Option. *Military Communications Conference, MILCOM 2006, Washington, DC, USA*, pages 1–7, 2006.

[128] TACOMS member nations. TACOMS standardizing federated networking (home page). http://www.tacomspost2000.org/, accessed 2010-01-30.

[129] Teleplan Globe AS. NORCCIS II. http://www.teleplanglobe.com/index.php/products-and-solutions/c4is/norccis-ii, accessed 2009-12-31.

[130] D. U. Thibault. Commented APP-6A — military symbols for land based systems — NATO's current military symbology standard. Technical Note, DRDC Valcartier TN 2005-222, September 2005.

[131] A. Vogel, B. Kerherve, G. von Bockmann, and J. Gecsei. Distributed Multimedia and QoS: A Survey. IEEE Multimedia, Vol. 2, No. 2, pp. 10-19, 1995.

[132] W3C. SOAP specifications. http://www.w3.org/TR/soap/, accessed 2009-12-22.

[133] W3C. Web services description language (wsdl) 1.1 - W3C note 15 march 2001. Erik Christensen, Francisco Curbera, Greg Meredith, and Sanjiva Weerawarana (eds.), http://www.w3.org/TR/wsdl, accessed 2009-12-22.

[134] W3C. Web services description language (wsdl) version 2.0 - W3C recommendation 26 june 2007. Roberto Chinnici, Jean-Jacques Moreau, Arthur Ryman, and Sanjiva Weerawarana (eds.), http://www.w3.org/TR/wsdl20/, accessed 2009-12-22.

[135] W3C. Web services eventing (ws-eventing) working draft. Doug Davis, Ashok Malhotra, Katy Warr, and Wu Chou (eds.), http://www.w3.org/TR/ws-eventing/, accessed 2009-12-22.

[136] W3C. Web services addressing (ws-addressing) W3C member submission 10 august 2004. Don Box, Francisco Curbera (eds.), http://www.w3.org/Submission/ws-addressing/, accessed 2009-12-26.

[137] W3C. Web services choreography description language version 1.0 W3C candidate recommendation 9 november 2005. Nickolas Kavantzas, David Burdett, Gregory Ritzinger, Tony Fletcher, Yves Lafon, and Charlton Barreto (eds.), http://www.w3.org/TR/ws-cdl-10/, accessed 2009-12-28.

[138] W3C. Web services policy 1.2 - framework (ws-policy) W3C member submission 25 april 2006. Jeffrey Schlimmer (ed.), http://www.w3.org/Submission/WS-Policy/, accessed 2009-12-28.

[139] W3C. Efficient XML interchange (EXI) format 1.0 W3C candidate recommendation 08 december 2009. John Schneider and Takuki Kamiya (eds.), http://www.w3.org/TR/exi/, accessed 2009-12-30.

[140] W3C. Extensible markup language (XML). http://www.w3.org/XML/, accessed 2010-01-03.

[141] W3C. SOAP message transmission optimization mechanism W3C recommendation 25 january 2005. Martin Gudgin, Noah Mendelsohn, Mark Nottingham, and Hervé Ruellan (eds.), http://www.w3.org/TR/soap12-mtom/, accessed 2010-01-03.

[142] W3C. Web services glossary W3C working group note 11 february 2004. Hugo Haas and Allen Brown (eds.), http://www.w3.org/TR/ws-gloss/, accessed 2010-01-03.

[143] W3C. Web services architecture W3C working group note 11 february 2004. David Booth, Hugo Haas, Francis McCabe, Eric Newcomer, Michael Champion, Chris Ferris, and David Orchard (eds.), http://www.w3.org/TR/ws-arch/, accessed 2010-01-16.

[144] Web Services Interoperability Organization (WS-I). WS-I organization's web site. http://www.ws-i.org/, accessed 2009-12-22.

[145] C. Werner, C. Buschmann, T. Jäcker, and S. Fischer. Bandwidth and latency considerations for efficient SOAP messaging. *International Journal of Web Services Research, Vol. 3, Issue 1*, 2005.

[146] F. Zhu, M. W. Mutka, and L. M. Ni. Service discovery in pervasive computing environments. In *IEEE Pervasive computing*, pages 81–90, October-December 2005.