Web Services and Service Discovery

Frank T. Johnsen, Joakim Flathagen, Tommy Gagnes, Raymond Haakseth, Trude Hafsøe, Jonas Halvorsen, Nils A. Nordbotten and Magnus Skjegstad

Forsvarets forskningsinstitutt/Norwegian Defence Research Establishment (FFI)

09.05.2008

FFI-rapport 2008/01064

1086

P: ISBN 978-82-464-1379-2
E: ISBN 978-82-464-1380-8l

Keywords

Tjenesteregister

Nettverksbasert Forsvar

Tjenesteoppdagelse

Tjenesteorientert arkitektur

Approved by

Anders Eggen	Prosjektleder
Vidar S. Andersen	Director

English summary

This report is a result of collaboration between the three FFI projects P1086 Secure Pervasive SOA, P1085 SEMANTINI, and P1102 NORMANS R&D. The report describes various requirements that a service discovery solution for Network Based Defense (NBD) needs to support. Furthermore, it gives a non-exhaustive survey of existing service discovery solutions, both standardized and experimental (we have focused on solutions that may be useful in NBD in some way or another). In conclusion we discuss how different service discovery solutions may be used at different levels in the military networks, and identify some important research challenges.

The service discovery mechanism used in any network must take the capabilities and limitations of the network into account; for instance, in low capacity networks it might be needed to sacrifice flexibility in order to reduce resource usage. Due to the large variation in network capabilities on different operational levels it is unlikely that one can find a single mechanism for service discovery that can be used everywhere. Thus, there is a need for a toolkit consisting of different service discovery mechanisms so that each network can use the mechanism that is most suited for that particular network. However, in a NBD scenario, information exchange, and thus service discovery, must be available across network boundaries. This means that the service discovery mechanisms chosen must be able to interact with each other without the need for manual configuration. The interaction between the different mechanisms must be clearly defined, and remains a major challenge when attempting to achieve pervasive service discovery across heterogeneous networks.

Sammendrag

Denne rapporten er utarbeidet som en del av samarbeidet mellom de tre FFI-prosjektene P1086 Sikker Gjennomgående SOA, P1085 SEMANTINI og P1102 NORMANS FOU – Videreføring. Rapporten er et resultat av kollokvievirksomhet mellom de tre prosjektene høsten 2007, der det ble foretatt en kartlegging av de ulike behovene som må tas hensyn til i en service discoveryløsning for Nettverksbasert Forsvar (NbF). Videre tar rapporten for seg en ikke-uttømmende oversikt over eksisterende service discovery-løsninger – både standardiserte og eksperimentelle (det er lagt vekt på løsninger som kan tenkes å være nyttige for Forsvaret). Avslutningsvis gir vi et forslag til hvilke teknologier som kan tenkes å bli benyttet i de ulike nettverkene i Forsvaret, og peker på videre forskningsmessig interessante problemstillinger.

Service discovery-mekanismene som benyttes må ta hensyn til kapabilitetene og begrensningene i nettverket; for eksempel, i nettverk med lav kapasitet kan det være nødvendig å ofre noe fleksibilitet for å få en mindre ressurskrevende løsning. På grunn av den store variasjonen i nettverkskapasitet på de ulike operasjonelle nivåene, så er det lite sannsynlig at man kan finne én enkelt mekanisme for service discovery som kan benyttes over alt. Det vil nok være et behov for en ”verktøykasse” bestående av flere ulike mekanismer, slik at hvert nettverk kan benytte den mekanismen som er best egnet nettopp til det bestemte nettverket. I NbF er det en forutsetning at flere heterogene nett kan samhandle. Dermed er det viktig at også service discovery kan gjøres på tvers av nettverkene. Altså må de valgte mekanismene kunne interagere med hverandre uten alt for store behov for manuell konfigurering. Interaksjonen mellom de ulike mekanismene må være klart definert, og dette er en av de aller største utfordringene man står overfor i arbeidet mot en gjennomgående service discovery-løsning på tvers av heterogene nettverk.

Contents

1 Introduction

One of the main goals of Network Centric Warfare (NCW) [6] and its Norwegian equivalent, Network Based Defense (NBD), is to increase mission effectiveness by interconnecting military entities. Sharing of information between decision-makers can help guide them towards making the right decisions at the right time, and a common information infrastructure is needed to facilitate sharing of relevant information across system and national boundaries. This leads to a requirement for a flexible, adaptable and agile communication infrastructure which can support all the communication needs of national forces, and at the same time support interoperability. The information infrastructure will have to support a number of different usage scenarios, from fairly static environments where services are stable, to dynamic environments where both services and service users come and go in a non-deterministic fashion.

We envision the concept of a Service Oriented Architecture (SOA) to become pervasive in this information infrastructure [7]. In a SOA, networked resources are made available to others as a collection of services, often implemented using a technology called Web services. Current Web service 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 tactical level where resources are scarce (low bandwidth) and the network consists of mobile units leading to frequent topology changes. This means that services can come and go; in other words, they are transient.

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*, is an important part of any SOA, but is particularly challenging in dynamic environments such as military tactical systems. A service discovery architecture for such an environment should reduce the amount of manual configuration, enable automatic 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 wireless connections with low network capacity.

In this report, we present and discuss various aspects of service discovery. There exist many different solutions for service discovery, all designed to solve some specific purpose. The different solutions are implemented according to different strategies [5]; some are based on multicast DNS, whereas others are based on a supporting layer: Application level solutions and network level solutions (e.g., integrating service discovery with the network routing protocol). Previous surveys of service discovery mechanisms at FFI address either application level solutions [3] or network level solutions [14], but do not look at the two in context. This study looks into some existing and proposed solutions for service discovery with particular focus on the challenges that arise when interconnecting networks with different capabilities and topologies.

The remainder of the report is organized as follows: In Chapter 2 we define some central terms. Chapter 3 presents the major requirements for dynamic service discovery. A taxonomy of service discovery models is given in Chapter 4, along with our suggested hybrid model. Web services and service discovery is discussed in Chapter 5, followed by an overview of some existing application level discovery solutions in Chapter 6. Experimental service discovery solutions for mobile ad hoc networks are discussed in Chapter 7. Security issues, an important aspect of any military communication solution, are covered in Chapter 8. Achieving interoperability between different Web service solutions requires standardization, and one should also look towards addressing semantic issues. Chapter 9 discusses semantic service discovery. In conclusion, there are still many open issues regarding service discovery in NBD. Chapter 10 summarizes some of these issues, and gives some suggestions regarding choices of technology for the different military communication networks.

2 Defining central terms

In this chapter we define some terms that are central to understanding this report, namely "service oriented architecture", "service", and "service discovery". There exist many definitions for these terms, some of them conflicting. We have chosen to use the definitions by OASIS, since OASIS is one of the major standardization organizations working with Web services technology today.

2.1 Service Oriented Architecture (SOA)

In [33] SOA is defined as being *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.*

Web services technology is currently the most popular and widespread implementation of SOA.

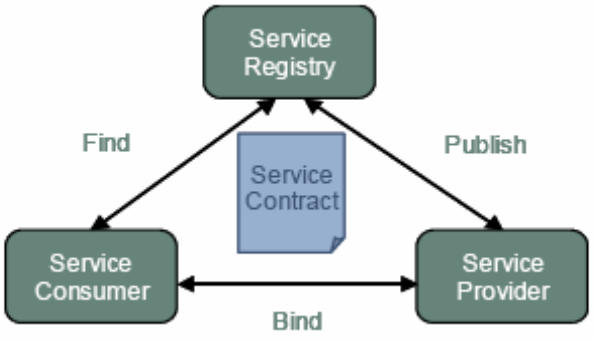

Figure 2.1 Elements of a Web service architecture

Let us examine the parts in *Figure 2.1* in more detail: There are three roles; *provider*, *consumer* and *registry*. Three operations define the interactions between these three roles; *publish*, *find* and *bind*.

1. A *service provider* is responsible for creating a service description, *publishing* that description to a service registry (or several registries), and receiving and answering invocations (*bind* requests) from service consumers.
2. A *service consumer* is responsible for finding a service description published to one or more service registries and using that description to invoke (i.e. *bind* to) service providers. With the *find* operation the service consumer states search criterions 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 service descriptions within the service registry.

2.2 Service

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.

A service is accessed by means of a service interface, where the interface comprises the specifics of how to access the underlying capabilities. There are no constraints on what constitutes the underlying capability or how access is implemented by the service provider. Thus, the service could carry out its described functionality through one or more automated and/or manual processes that themselves could invoke other available services. A service is opaque in that its implementation is typically hidden from the service consumer except for

1. the information and behavior models exposed through the service interface and
2. the information required by service consumers to determine whether a given service is appropriate for their needs.

The consequence of invoking a service is a realization of one or more real world effects (i.e. the actual result of using a service).

This description of the concept of a *service* was penned by OASIS, refer to [33] for further details.

2.3 Service discovery

Service discovery is the process of finding and identifying a service in a network. Service discovery can be divided into two categories [34];

1. manual discovery
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 significantly 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.

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. For Web services, service discovery in its simplest form can be supported by a single centralized registry, as shown in *Figure 2.1*.

3 Requirements for dynamic service discovery

In traditional networks and in the Internet community a lot of work has been done in the terms of service discovery. A number of different mechanisms are 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 makes these generic service discovery techniques unsuitable for tactical networks.

Below, we briefly summarize some high level requirements for a discovery infrastructure for dynamic environments. This builds on work initially presented in [2]:

- Whether the underlying network is a LAN, WAN or a mobile ad hoc network, a unified way to bootstrap and maintain the service discovery infrastructure is needed to avoid frequent manual configuration. There should be automatic discovery of registry nodes in a coherent and transparent way.
- The system should allow flexible resource utilization, since capacity (memory, CPU, storage) and connectivity distribution often are asymmetric. Limited clients should be allowed to delegate service selection to registry nodes (they may return only the best service advertisement) and hereby prevent receiving too many responses to queries. Especially in wireless environments, it is important to use bandwidth efficiently.
- 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, WWW). Additional artifacts needed by clients to evaluate or use services (e.g. XML schema, ontologies) must be obtained from 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 service network and should not advertise services that are no longer present on the network. This is known as *liveness* information.
- The infrastructure should support different kinds of service description mechanisms, ranging from simple (name, id, URI specifying a pre-agreed service type), to rich (e.g. semantic descriptions). Thus, both normal Web services [4] as well as Semantic Web services should be able to use this infrastructure. Also, services not relying on Web services standards as their transport should be able to use the service discovery infrastructure. This could for example be services that broadcast information via UDP according to some custom standard (e.g. Tactical Data Links).
- Security measures must also be taken, as the registry and the capability to discover services is a key component of the information infrastructure.

These issues described above are generic requirements that should be addressed by any service discovery framework, be it for WAN, LAN or ad hoc network. There are, however, some additional issues that must be considered for service discovery in tactical ad hoc networks [14]:

- *Low bit rate* increases the demand for high performance protocols and cross-layer optimizations. Compression and other techniques which reduce communication overhead become more important with lower bit rate.
- *Extended radio coverage* may lead to more infrequent rerouting compared to 802.11 based networks using the same mobility patterns. However, the heterogeneity in a combined soldier, sensor, and vehicular network puts high requirements on the choice of the mobile ad hoc routing protocol.
- *Higher security demands* affect the choice of protocols on all protocol layers.
- *Proprietary protocols* and solutions are more frequent in tactical networks. This could possibly make cross-layer techniques more feasible. The compatibility with legacy protocols and equipment must however be taken into consideration.

4 Taxonomy

There are two main categories of service discovery models, as classified in [8] and [5]:

1. The *client-service model.*
2. The *client-service-directory model*.

Figure 4.1 illustrates the difference between these two models. The current solution for service discovery in Web services, the Universal Description Discovery and Integration (UDDI) specification, falls in the second category. The first model does not rely on a registry for service discovery, instead it relies on a totally decentralized topology. In this client-service model a client broadcasts or, more efficiently, multicasts its queries on the network. The services that

match with the received query return a reply. In this model there is no registry server to keep the information about the services. This model is considered suitable for small networks by [5]. In the client-service-directory model, there is a registry that keeps information about all the services which are on the network. This registry can be either centralized or distributed, i.e. decentralized. A client must send a query to the registry server to find a service.

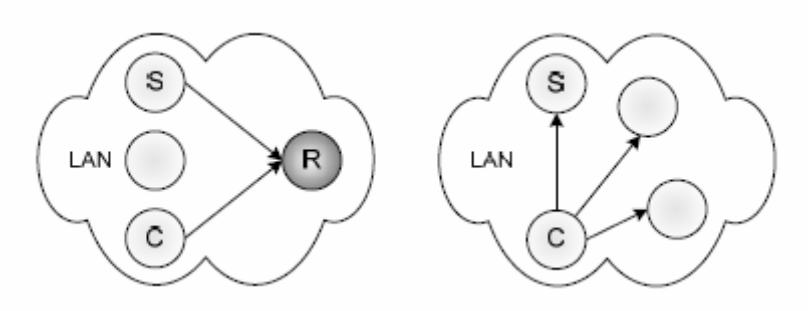

Figure 4.1 Two modes of service discovery:Centralized/decentralized with one or more registries, and decentralized without a registry [1].

Consequently, we can categorize the different discovery models into three basic topologies [1], namely completely decentralized, centralized, and distributed, as shown in Figure 4.2. The centralized topology has one node with more responsibility than the others, whereas in the decentralized topology, all nodes are equally important. The distributed topology is a compromise between the centralized and the decentralized, where a group of nodes has more responsibility than the others. We will now discuss pros and cons of each topology in turn. Furthermore, we introduce the concept of a hybrid topology with aspects from both service discovery models.

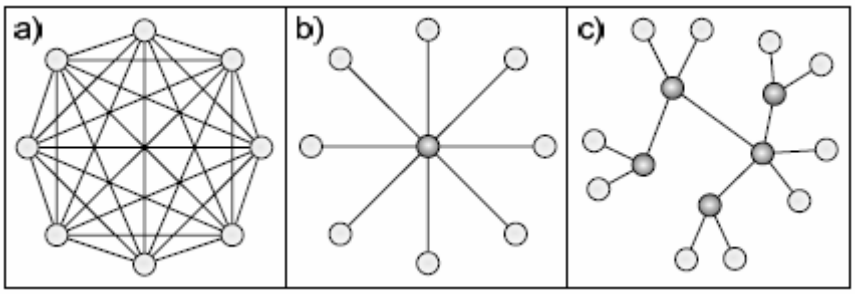

Figure 4.2 Service discovery topology: Decentralized (a), centralized (b), and distributed (c) [1]

4.1 Client-service model (fully decentralized)

The client-service model does not rely on a registry for service discovery, instead it relies on a totally decentralized topology. There are several reasons why a decentralized approach is attractive for service discovery. For instance, most decentralized systems are based on service provider nodes hosting their own service advertisements. This means that updates to these advertisements will be immediate, and that there is no need to republish and propagate any new service metadata. Further, the available service advertisements correspond directly to the state in the service network. This means that if a service provider node goes down, its service will no longer be advertised. Decentralized systems are also robust against failure and attacks, since they

have no weak points that can bring the whole system down if they disappear. There are, however, several problems related to the decentralized approach as well: One problem is that a query must be propagated to all nodes, typically through broadcast. Then, all provider nodes must evaluate the query independently of each other before they return their responses to the querying node. Therefore, the decentralized approach can lead to high bandwidth consumption in the system. It also increases the load on all provider nodes because they have to do processing every time a query is received. This becomes an even greater problem when richer, semantic descriptions are used. Because device capabilities are often different, not all nodes may be able to evaluate queries on semantic service descriptions. This is different from the peer-to-peer world, where simpler (string- or hash-based) advertisements are used. A decentralized topology therefore cannot be used in all cases.

Services, and the real-world resources they represent, should be considered unique. Again, this is different from peer-to-peer systems, where several identical files could be replicated across the system. Therefore, in a service discovery system, all available advertisements should be queried in a deterministic way, not in a random way that does not guarantee discovery of available advertisements. Random querying is often used in peer-to-peer systems. Another effect of using a decentralized approach is that there is no central point where the total number of responses to a query can be controlled. This lack of *query response control* can at worst, if a query is too broad, lead to “response implosion” at the querying node, meaning that the client is bombarded with responses from potentially all provider nodes in the service network. This certainly increases network load, especially as service descriptions become larger. It also increases the processing load on the querying client, because now it must evaluate all the received responses itself, to make a final selection between the candidate service descriptions it has received. Of course, the number of responses from each node can be limited, but still, query response control is very coarse-grained. The total cost in terms of processing and network capacity therefore is high with this topology.

4.2 Client-directory-service model

4.2.1 Centralized

A centralized topology is probably the most efficient in terms of configuration, because clients only have to maintain the location of one registry. Also, since there is only one centralized location with a complete view of the service network state, query response control is not a problem. To maintain an up-to-date view of the service network in a centralized registry, removal of old advertisements should be done, since we have to deal with dynamic conditions where services may disappear abruptly. Should the number of service advertisements grow very large, storage may be a constraint as well, since one node must host all advertisements. Further, by delegating service selection to the central registry, query evaluation may only have to be carried out once. The opportunity to allow service selection support in registries is important to relieve constrained clients. However, a completely centralized solution has problems related to robustness, since we now have a single point of failure. Also, processing load could be high on

the central node, since all query evaluation must be performed on a single node. Moreover, with the centralized approach bandwidth must be shared between all querying clients, potentially leading to a bottleneck situation. This must, however, be compared with a greater total load on the network generated by the decentralized topology, which can be critical for networks with broadcast media.

4.2.2 Distributed

As mentioned, a distributed topology is a compromise between the decentralized and the centralized approach. It is based on a group of intermediate nodes that form a centralized network of nodes that have more responsibility than the rest. Potentially, this can strike a balance between the two “pure” topologies discussed above. Finding the right combination of different properties may enable query response control, robustness, load balancing, and bandwidth-efficiency, and therefore may offer support for dynamic environments. Various systems with a distributed topology place different degrees of responsibility at the intermediate nodes, depending on properties such as query evaluation capability and storage capability. In the peer-to-peer world, intermediate nodes are often termed super-peers, and systems relying on a distributed topology are called hybrid peer-to-peer systems. We have identified three subcategories of distributed topologies: clusters, distributed hash tables, and multiregistries. Clusters are basically one registry replicated on several nodes. This means that the same content is present at different nodes. An example of this is UDDI. Superpeer distributed hash tables are used in several peer-to-peer systems, like in [9]. Such systems are based on storage of hashes in the intermediate nodes, and therefore, semantic query evaluation cannot be performed at the intermediate nodes in such systems. Logically, this makes these systems more decentralized. The multi-registry topology is based on autonomous registries with independent content.

4.3 Hybrid model

When designing a service discovery 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. Thus, a pure client-directory-service model as is currently used in Web services should probably not be employed. Consider the example from [3] (see Figure 4.3) where two tanks driving past each other share a wireless network. One of the tanks provides a service that the other needs to use, but the UDDI registry they both use is currently off-line, thus the service cannot be found. Clearly, there should be some other means for discovering service on a shared local network.

Consequently, service discovery in NBD should support a hybrid model, in which a pure client-service communication model can be used in the absence of a registry service. However, in those cases where a registry is present, one could fall back to using the client-service-directory model. We suggest that the registries should be detectable as any other service, thus enabling the use of the client-service model until a usable registry service is detected.

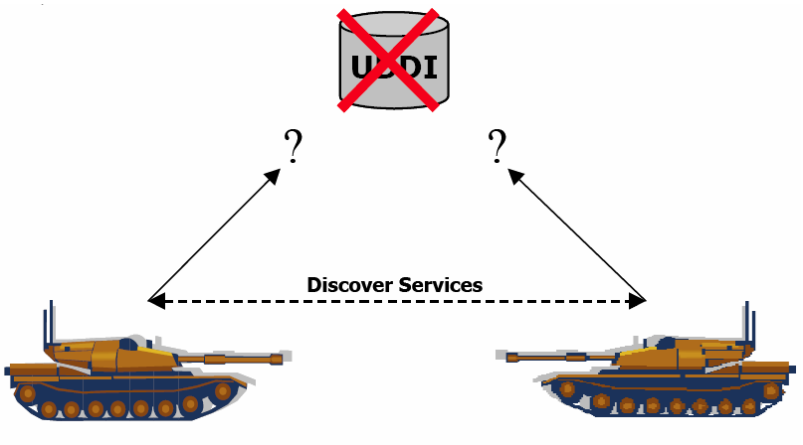

Figure 4.3 The need for decentralized service discovery without registry

5 Web services and service discovery

In this chapter, we will look at some existing Web service specific solutions for service discovery. We can split existing technologies for Web Service discovery into local area network (LAN) discovery and wide area network (WAN) discovery (WAN registries can be used on LANs as well). For WAN discovery, both UDDI and the ebXML registry make it possible to use multiple registries or to let a registry consist of several nodes, which can help achieve robustness and scalability. In UDDI, either replication between registry nodes or a hierarchical model may be used. The ebXML solution supports a nonhierarchical multi-registry topology, facilitating federated queries. LAN discovery solutions such as WS-Discovery are often based on local-scoped multicast. WS-Discovery facilitates registry discovery and local service discovery based on URI matching. However, current technologies target different domains, and no relationship between them exists. This basically means that one technology must be chosen for dynamic LAN discovery and another one to reach out on the WAN.

5.1 Universal Description, Discovery, and Integration (UDDI)

To be able to publish Web services in a service registry, the specification Universal Description, Discovery and Integration (UDDI) [10] can be used. Basically, UDDI allows service providers to register their services and service consumers to discover these services. UDDI is defined as a Web service itself, meaning that the SOAP protocol must be used to interact with the registry. In principle, UDDI is centralized, but mechanisms for federating several registries have also been specified in version 3 of the specification. 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 inside a registry. The replication topology must be configured. UDDI has an advanced API that facilitates querying for businesses, their provided services, and technical information about these services, specified in so-called *bindingTemplates*. It is also possible to subscribe to, and to be notified of changes in the registry.

All core entities in the UDDI information model are assigned unique keys. 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 be used for different purposes. For instance it can include pointers to further description of a service, for instance its WSDL description and bindings. In the UDDI specification, this is called the technical fingerprint. 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 Quality of Service (QoS) issues [37]. 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.

Furthermore, there is no uniform way of querying about services, service interfaces and classifications: The search for services is restricted to Web service name and its classification. Also, there is no liveness information in UDDI, so it is possible to get out-of-date service documents in the registries. Although UDDI is considered to be one of the core Web service 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 Web service enabled businesses use UDDI [36].

5.1.1 UDDI in NBD

For interoperability concerns it is best to choose standards based COTS, and UDDI is standardized and available. However, the basic, standardized components of UDDI are lacking some important features, such as the above mentioned QoS-support and liveness information. UDDI can be expanded with proprietary functionality in a way that such features become available. Such expansion could make UDDI better suited to NBD. For further considerations about using UDDI in NBD, see [3].

Choosing a common standard for service registries enables NATO partners to be interoperable in operations. Currently NC3A is investigating semantic interoperability through the Interoperability Metadata Registry and Object Store (IMROS) team. UDDI, and the below mentioned ebXML, are being considered by IMROS, but UDDI is considered to be less suitable than ebXML since it is not designed for semantic annotation functionality. Both UDDI and ebXML are standards under the OASIS umbrella, but UDDI is not as actively updated for future semantic extensions as is ebXML [42].

5.2 Electronic Business using eXtensible Markup Language (ebXML)

ebXML is the result of an initiative created by UN/CEFACT and OASIS in 1999. The goal of ebXML is to standardize a framework “designed to enable enterprises of any size in any geographical location to conduct business over the Internet” [43].

ebXML is not a single standard, but rather a modular suite of specifications. The ebXML suite currently includes specifications for

- Business processes (ebBP): Defines a standard language for business processes to enable collaboration between business partners [44].
- Core data components: Aims to define a way to discover and analyze core information components when performing electronic business [45]. An example of a core information component can be attributes in a product catalog that is expressed differently by different business partners, but needs to be compared.
- Collaboration protocol profiles and agreements (ebCPPA): Provides definitions for business profiles and how agreements should be negotiated [46]. When using ebCPPA, each business has a profile that defines its capabilities. Business partners may then negotiate agreements based on each other's capabilities [47].
- Messaging services (ebMS): A communications-protocol neutral method of exchanging electronic business messages. It supports reliable and secure delivery. Messages may contain payloads of any format type [48].
- Registries (ebRIM) and repositories (ebRS): The Registry Information Model (ebRIM) defines what types of objects are stored in the ebXML Registry and how they are organized [49]. The Registry Service specification (ebRS) describes how to build registry services that provide access to information content in the ebXML Registry. This includes APIs, interfaces and interaction protocols [50].

This chapter will focus on the functionality provided by the ebXML Registry[1].

5.2.1 The ebXML registry

An ebXML Registry contains both a registry for metadata and a content repository. Architecturally, the registry and repository are separate, but they are both accessed through the Registry Service interfaces as defined in ebRS [50]. The ebXML Registry must be able to contain a wide variety of data objects and is designed more like a database with a defined information model than a service directory. This enables storage of not only pointers to repository items, but also the items themselves. Thus, the registry can be used for governance of any type of objects throughout their lifecycle. The importance of a repository is discussed in more detail in [51]. Objects in the registry are represented by classes descended from a common super class called RegistryObjects. Classes commonly used in business relations are predefined in ebRIM. Such classes include "Organization", "Person", "Service" and so on. RegistryObjects and its descendents can be extended by adding new fields as one or more "slots". "Slots" are composed of a name, data type and one or more values. RegistryObjects may also be associated with other RegistryObjects [49]. To manipulate and retrieve objects in the registry a query management interface is defined in enRS [50].

[1] As specified in [50] and [49]. Some of the features discussed are only available in implementations conforming to "Registry Full". A comparison of the "Registry Full" and "Registry Lite" conformance profiles is available in [50].

Items in the repository are treated similarly to registry objects, but the representing classes are descendents from the class RepositoryItems. Repository items are managed through the lifecycle management interface defined in ebRS [50].

Web services can be described directly in the registry with Service classes defined in the information model.

All objects are assigned an object reference consisting of a globally unique identifier. This ensures that objects can be uniquely referenced across multiple registries [49].

5.2.1.1 Query management

Queries are used to discover items and services or to read metadata from the registry. They may be written as ebXML Filter Queries or SQL92 queries. Filter Queries are not as expressive as SQL and may have some performance issues depending on the implementation.
Queries can be executed ad-hoc or stored in the registry for later use. Stored queries may have user-defined parameters [50].

5.2.1.2 Life cycle management

Items in the repository are managed through the life cycle management interface. The interface supports submission, update, deprecation and removal of objects. All objects are automatically versioned and carry version and status information. The object statuses may be customized.
Each time an object is submitted to the registry its version number is increased. The client or registry may purge older versions of the object automatically when they are no longer in use [50].

5.2.1.3 Cooperating registries

As of version 3 of the Registry Services specification [50] the ebXML Registry supports federations of registries. A federation is a group of registries that cooperate to serve their clients. Federations in ebXML are based on a peer-to-peer model where all participating registries are treated equally. Any registry may join, leave or create a federation at any moment. Registries may participate in any number of federations simultaneously.

When a client sends a query to a registry participating in a federation, the query is forwarded to all federation members. The result sets are merged before they are returned to the client by the registry that received the original query.

Registries participating in a federation may synchronize object metadata with the other participants for fault-tolerance and increased performance, but this is not required. Flexible local caching and replication mechanisms allow adjustments of resource usage in each registry.
Objects may be referenced, relocated or replicated between registries independently of federation membership [50].

5.2.1.4 Events and notifications

When a user subscribes to an event, a stored ad-hoc query must be created that projects the area of interest. If the query result changes, a notification may be sent to a predefined service that supports notifications (push) or it can be stored in the registry if the user wants to check the result later (pull) [50].

5.2.1.5 Content management

Clients may register a Content Management Service that validates content when it is submitted to the registry or repository. Validation may run when the object is submitted (inline invocation model) or sometime after (decoupled invocation model).

Content Cataloging Services may be used to automatically catalog submitted objects. This enables Content-based Discovery and content based queries [50].

5.2.1.6 Security

Authentication in ebXML is based on X.509 digital signatures. It also supports fine grained access control policies using XACML 1.0. Identity management is based on SAML 2.0 profiles, including federated identity management and single-sign on (SSO) [51].

5.2.1.7 Implementations

Several vendors support ebXML in their products, including Sun and IBM. But since many of the features in the specifications are optional they may not be available in all products. OASIS has published an open source reference implementation called freebXML that is fully compliant with version 3.0 of ebRS and ebRIM and is written in Java [52].

A list of products with complete or partial support for ebXML is available at [53]. A list of ebXML implementations is available at the ebXML website [54].

5.2.2 ebXML in NBD

The ebXML Registry is designed to store a wide variety of data formats and the information model is easy to extend to allow proprietary uses by adding new fields to existing classes. The use of classes and associations combined with globally unique object identifiers makes it possible to express complex relationships between objects directly in and in between registries. The superior flexibility of ebXML, when compared to UDDI, makes it interesting in an NBD setting. However, since it is not currently possible to create new classes, some of the expressiveness of ebXML is lost when it is used outside business environments.

Content management enables enforcement of semantic models with automatic validation and cataloging. Event subscriptions can be used to notify clients when a service that fits a specific semantic description becomes available. Support for semantics is important for interoperability, and should be considered a key enabler and feature of NBD. Leasing mechanisms for service

descriptions are unfortunately not included in ebXML, but it may be possible to create events that remove expired objects through the life cycle management interface.

Support for federations provides fault-tolerance, distributed queries and load balancing. Since the federation model is peer-to-peer based, there is no single point of failure in a federation, as long as objects are replicated among the registries.

Object relocation, replication and adjustable local caching may be used by registries to adapt to the resources it has available. For instance, objects could be relocated or replicated to locations closer to where they are needed, thus reducing global bandwidth requirements.

All in all, ebXML seems better suited for use in NBD than UDDI from a theoretical perspective. However, both registries should be subjected to actual practical tests before a choice is made. For a comparison matrix of UDDI v3 and ebXML v3, see [51].

5.3 Web Services Inspection Language (WS-Inspection, WSIL)

WS-Inspection defines a simple document type that can be used to advertise services. A WS-Inspection file contains information such as a short human readable description of available services, links to WSDL files or links to other WS-Inspection files. The root WS-Inspection file is located under a well known name and path.

A short overview of WS-Inspection, sometimes also called WSIL, is given by IBM in [35]: The WS-Inspection specification 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. A WS-Inspection document provides a means for aggregating references to pre-existing service description documents, which have been authored in any number of formats. These inspection documents are then made available at the point-of-offering for the service as well as through references, which may be placed within a content medium such as HTML.

Specifications have been proposed to describe Web services at different levels and from various perspectives. It is the goal of the proposed Web Services Description Language (WSDL) to describe services at a functional level. The UDDI schema aims at providing a more business-centric perspective. What has not yet been provided by these proposed standards is the ability to tie together, at the point of offering for a service, these various sources of information in a manner which is both simple to create and use. The WS-Inspection specification addresses this need by defining an XML grammar which facilitates the aggregation of references to different types of service description documents, and then provides a well defined pattern of usage for instances of this grammar. By doing this, the WS-Inspection specification provides a means by which to inspect sites for service offerings.

5.3.1 WS-Inspection in NBD

Repositories already exist where descriptive information about Web services has been gathered together. The WS-Inspection specification provides mechanisms with which these existing repositories can be referenced and utilized, so that the information contained in them need not be duplicated if such duplication is not desired. This enables simple service discovery to be performed, if the address of a repository is known already. This could perhaps be used for interoperability reasons, where WS-Inspection could be used to query gateways or dedicated interoperability points for services exported from a nation to the coalition. However, it is a very static way of performing service discovery, and its usefulness will be limited by that fact.

5.4 Web Services Dynamic Discovery (WS-Discovery)

WS-Discovery [38] is a proposal from several vendors, and addresses some of the shortcomings of UDDI. UDDI provides discovery for services that are always connected to the network, but one also needs a discovery system for services that are only connected occasionally. Also, UDDI provides discovery only for registered services, but discovery of services that do not exist in any central registry is also needed. Basically, UDDI is suited for use in static, wired networks. WS-Discovery, on the other hand, is a discovery system that can be used in ad-hoc networks. WS-Discovery defines a multicast protocol using SOAP over UDP to locate services, a WSDL providing an interface for service discovery, and XML schemas for discovery messages. It allows dynamic discovery of services in ad-hoc and managed networks, and enables discovery of services by type and within scope. WS-Discovery leverages other Web service specifications for secure, reliable message delivery. Inherently scalability is limited due to the use of multicast, but WS-Discovery can scale to a large number of endpoints by defining a multicast suppression behavior if a service registry, i.e. discovery proxy, is available in the network. The discovery proxy is intended to be a registry for Web services (e.g. UDDI). When the discovery proxy is discovered, clients use a discovery-specific protocol to communicate with it. However, this is not a part of the WS-Discovery specification and details are left to the programmers: WS-Discovery neither defines the discovery-specific protocol nor the interaction between the WS-Discovery service and registry.

WS-Discovery is not one of the core Web services standards; in fact, it is not a standard at all. The WS-Discovery specification is provided as-is and is not a standard, so currently it is not in widespread use. However, Microsoft, BEA, Canon and Intel are contributors, and WS-Discovery is implemented in Windows Vista, so we can expect others to implement support for it in the future as well.

5.4.1 WS-Discovery in NBD

There are some limitations in WS-Discovery that limit its usefulness: Just like UDDI, it does not provide liveness information, and it does not define a rich data model for service descriptions. Also, WS-Discovery it is not suitable for Internet-scale discovery since multicast may not be supported, and it relies on multicast. However, since it is only a draft specification at the moment it may evolve to include some of these lacking features in a later version. WS-Discovery could

perhaps be used to discovery registries in a LAN, or services in a local area if no registry is present. Such functionality will be needed in NBD, as discussed in Section 4.3.

6 Existing application layer solutions

This chapter discusses several existing service discovery protocols. These protocols are generic, and can be used to discover a wide array of services such as printers and hosts on a network.

6.1 Universal Plug and Play (UPnP)

Universal Plug and Play [11] is said to be targeted at ad-hoc or unmanaged networks at home, in small companies, public places, or attached to the Internet, it being networks of intelligent appliances, wireless devices, or PCs. Being a Microsoft initiative it is present in newer versions of Microsoft Windows.

UPnP networking is divided into several phases. The first phase consists of assigning an IP address to the device [15]. After an IP address has been assigned service discovery is carried out. The client is referred to as a control point and when a control point has discovered a service it can obtain further information about the device offering the service by downloading an XML description provided through a URL. This description enables the control point to control the service (i.e. use the service), receive notifications of events, or perhaps bring up a user interface for the service in a browser.

The General Event Notification Architecture (GENEA) is used for notifications, while SOAP is used to send control messages. The Simple Service Discovery Protocol (SSDP) is UPnP's solution for service discovery. SSDP will therefore receive the main attention here.

When an UPnP device is added to the network it multicasts an advertisement for each of its provided services to a standard multicast address and port (239.255.255.250:1900). This discovery message must contain a validity for the service (minimum 30 minutes), and the advertisement will have to be re-sent before this validity expires to remain valid. If a device leaves the network or revokes a service prior to the expiration time it should multicast a message corresponding to each of its advertisements informing that the service is no longer available. Control points (clients) listen to the multicast channel and caches the information received in advertisement messages. When a control point first joins the network it searches for devices of interest by sending a search message to the multicast address. Devices offering the services respond to the control point using unicast. By the combination of first searching actively and then listening passively for advertisements all control points have information about the available services in the network at a given time. Because a node may leave abruptly or crash however it may not always advertise its departure and the expiration time is thus used to remove stale entries. The SSDP messages are sent using multicast or unicast versions of HTTP, depending on whether the message is to be multicast or not. These HTTP messages are again transported over UDP. Because UDP is unreliable each message should be sent more than once.

6.1.1 UPnP in NBD

In UPnP it is assumed that a node remains updated on the services available in the network. This is achieved by requesting information about available services when entering the network and then caching the service advertisements from nodes entering the network at a later point. However, the assumption that a node is kept updated using such a method may not hold in a tactical network. This is due to the fact that if two mobile ad-hoc network (MANET) partitions are interconnected after having been physically separated the nodes in one partition will have missed the advertisements from the nodes in the other partition. While the creation and merging of partitions are relatively rare in traditional wired networks, partition changes can be expected to occur more often in such dynamic networks as tactical networks.

Also, since UPnP is not Web services related, whereas Microsoft's proposed WS-Discovery is (WS-Discovery can replace UPnP), one should probably not base a new solution on UPnP, but rather look towards the development of WS-Discovery.

6.2 Service Location Protocol (SLP)

The Service Location Protocol (SLP) [12] is targeted for use in local area networks (LANs) and is an Internet standard from the Internet Engineering Taskforce (IETF.) SLP messages are normally transmitted over UDP.

The SLP defines three types of agents according to SOA principles; these are the user agent (UA), the service agent (SA), and the directory agent (DA). A user agent is a process trying to locate a service for the user/application by retrieving service information from service agents or directory agents. The service agent is an agent representing one or more services to advertise the services. The directory agent collects service advertisements. A directory agent does not need to be present for SLP to operate, but increases scalability in larger networks.

In the absence of a directory agent a user agent would send service requests for a requested service to the SLP multicast address. Service agents listen to this multicast address and respond with a unicasted message if they provide the service. Services and their locations are represented using service URLs. A user agent submitting a service request for a printer service represented by the service URL "service:printer" could for example receive a reply containing the URL "service:printer:lpr://hostname". The latter URL specifies that the service type line printer is available at hostname.

User agents and service agents discover the presence of a directory agent by sending a service request for the directory agent service at start-up, and directory agents also periodically advertise their presence on the SLP multicast address. By default user agents and service agents are required to wait for 15 minutes before repeating a directory agent discovery attempt, and the default frequency of the directory agent discovery advertisement is every three hours. These values would have to be set lower to accommodate more dynamic networks, at the expense of

more network traffic. The location of the directory agent(s) can also be preconfigured through DHCP or static configurations.

Service agents register their services with directory agents using unicast messages, and reregister them periodically to keep them from timing out. User agents can then unicast service requests to their directory agent(s). Several directory agents can exist within a network and user, service and directory agents can be grouped by the use of scopes. Service location can then be carried out within a certain scope, for example only within a certain department.

6.2.1 SLP in NBD

SLP is based on SOA principles, which makes it interesting. The fact that it can be used to find registries and other pre-defined services is also a plus, but its model for expressing the services – using only a service URL – limits its usefulness in an NBD setting. Using an URL to describe services is very limiting, and a framework with more flexibility and semantics support is preferable.

6.3 Jini

Jini [13] is targeted at the workgroup, and aims to federate groups of devices and applications into a single distributed system. Jini is a product from Sun Microsystems and is heavily based on Java and Java RMI. Thus, devices need a Java Virtual Machine (VM) to fully utilize Jini. In addition to service discovery Jini also provides service invocation, transactions and event notifications. When a service provider is plugged in it performs discovery to locate a lookup service. Upon locating a lookup service its services can be registered with the lookup service, a process referred to as join in the Jini jargon. Discovery is conducted by multicasting a request for a lookup service on the local network. The join is conducted by uploading a service object to the lookup service. The service object contains the Java programming language interface for the service, including the methods that users and applications will invoke to execute the service along with any other descriptive attributes. The service object then functions as a proxy for the service. After registering with the lookup service the service provider obtains a lease, and will periodically have to renew the lease to keep the service registered at the lookup service. When a client wants to look up a service it first discovers the lookup service in the same way as the service provider. It then looks up the desired service by its interface and descriptive attributes. The service object is then downloaded into the client. The client can then interact with the service provider through the service objects which functions as a proxy. The actual communication between the service object and the service provider is thus transparent for the client, as this is implemented by the service object. In the case where no lookup service exist a client can perform peer look up by pretending to be a lookup service and thus have service providers attempt to register their services, from which the client can pick the one(s) desired and drop the rest. As can be imagined this method could be quite bandwidth consuming.

6.3.1 JINI in NBD

The JINI concept would be very suitable in NBD were it not for the bandwidth consumption and the fact that JINI is limited to Java, excluding other frameworks.

6.4 DNS Service Discovery (DNS-SD)

Apples *Bonjour* (formerly Rendevouz) is included in MAC OS X, and is also supported by the KDE and Gnome desktop environments found on Linux and BSD platforms. Bonjour uses a combination of link-local address choosing [15], Multicast DNS (mDNS), and DNS-Service Discovery (DNS-SD).

DNS-SD is a way of using the existing DNS records to locate services. Since a ZeroConf (or Bonjour) implementation most likely will have a multicast DNS responder for the name-to-address translation, the amendment of service discovery can be implemented in quite a lightweight manner. DNS-SD is considered simpler than SSDP because it uses DNS rather than HTTP. The protocol can be used to obtain names, service type, port numbers and other attribute information.

Since Apple first launched Bonjour in 2002, every major maker of network printers has adopted Bonjour and uses DNS-SD to advertise the printer service to the local area network.

6.4.1 DNS-SD in NBD

A proposal to use DNS-SD and Zeroconf technology to interconnect soldier wearable computers and Ethernet-enabled devices is presented in [16].

6.5 JXTA

JXTA [39] is a hybrid topology with decentralized query evaluation. It is a framework by Sun for building peer-to-peer applications. It is very general, and does not require applications to be service-oriented. Neither does it have its own service description vocabulary. Nevertheless, it could easily be used to carry any XML-based service descriptions. JXTA is what we call a hybrid peer-to-peer technology, which means that some peers are more important than others in that they store pointers to advertisements that reside on edge peers. Such super-peers are called Rendezvous peers. JXTA is based on a loosely consistent Distributed Hash Table (DHT) mechanism [40]. One particularly interesting research project based on JXTA is the Service-oriented Peer-to-Peer Architecture described below.

6.5.1 Service-oriented Peer-to-Peer Architecture (SP2A)

In traditional, client/server based Grids a considerable amount of resources remain unused. By using a service-oriented peer-to-peer network, resources can be made available across the Grid through services. SP2A is a lightweight Service-oriented framework for Peer-to-Peer based resource sharing in Grid environments, proposed by the Distributed Systems Group at the University of Parma, Italy [56].

A service-oriented peer (SOP) in SP2A is constructed of several basic modules as described in [56]:

- *Message Handler*: Allows peers to create virtual network overlays on top of existing infrastructure.
- *Resource Provision Services*: Implements resource management mechanisms for local and remote resources.
- *Resource Monitor*: Maintains a list of available resources and reports resource failures.
- *User Interface*: Allows the user to interact with the system.
- *Router*: Handles routing of messages between the peers in the network.
- *Scheduler*: Schedules task execution requests based on resource availability and workload. Handles load distribution when the same resource is available from more than one peer.
- *Security Manager*: Responsible for protection of shared resources, identity management, privacy and confidentiality.
- *State Manager*: Peers can have different states in different groups of peers, i.e. supernode in one and leaf node in another. The State Manager checks and changes the peer state.

Resources in SP2A are published as Resource Provision Services (RPS) and are described by name, a textual description, and a uniquely identified owner. In addition, an interface is exposed that specifies how to access it. A service may also have semantic descriptions of the expectations associated with it. These are divided into three parts; the data model, the process model, and the behavior [56].

Discovery is based on both value matching and semantic search. Discovered RPSs are ranked against the reference ontology using a semantic matcher. Since SP2A is a peer-to-peer system, discovery of an existing service is never guaranteed as there is no guarantee that a query will be broadcasted to the whole network [56].

SP2A has been implemented as a lightweight Java API with Sun's JXTA framework functioning as a Message Handler and Router module. It currently supports service deployment with Web services and semantic service descriptions in OWL-S [56]. The software is intended to be small and portable, thus functioning on a variety of platforms including servers, laptops, PDAs and mobile phones: See [57] for specific platform requirements.

Even though the Java API has been created with portability in mind, it does have some compatibility issues – especially with mobile phones. Mobile platforms have evolved considerably the last years and even when running the same operating system, devices may have incompatibilities that affect the framework. Besides compatibility issues the implementation has been demonstrated to correctly establish a virtual network, discover services and dynamically invoke them even with limited resources available [57].

6.5.2 JXTA in NBD

The peer to peer paradigm seems to be suited for use in NBD from a theoretical point of view. Currently, we are planning on performing further experiments with this architecture as part of the collaboration between NC3A and the FFI project “Secure Pervasive SOA”.

7 Service discovery in MANETs

A mobile ad-hoc network (MANET) is a collection of mobile nodes connected by wireless links able to dynamically form an autonomous multihop radio network, without the use of any pre-existing infrastructure. Intermediate nodes in a MANET can act as routers that forward packets on behalf of other nodes. Further features of MANETs are that they are self-forming, automatically configurable and adaptive to rapid changes of the network topology.

As found in the literature, a variety of applications can benefit from ad-hoc technology. Nevertheless, the flexibility, cost benefit, ease of maintenance, auto-configuration and all the other advantages come at a price. Ad hoc networking introduces a great many challenges and imperatives, and adopts the side effects of wireless computing: Wireless links are significantly less reliable than wired media, having unpredictable signal quality and transmission range, the channel can be time-varying and possibly asymmetric, and the wireless links suffer from security problems not found in wired networks. Further, a MANET introduces a lot of challenges due to its multihop nature, topology dynamics, heterogeneity, variations of node availability and power constraints.

Ad hoc technology has proved to be a very useful tool for meeting the tactical battlefield communication requirements [19]. Self-configuring networks are extremely effective; they lower the user interaction to a minimum and enable quick deployment of troops in urban environments or battlefields. Utilizing soldiers, sensors and vehicles as MANET routers, ad hoc technology will extend the operational range both in line of sight and in non line of sight scenarios and enable radio connectivity in previously inaccessible environments. With the combination of lightweight and user-friendly combat PDAs, sensors, automatic service discovery and ad hoc technology, we can anticipate a revolutionary advance in situational awareness and the combat power of individual soldiers on the digitized battlefield.

Recent research has brought the ad hoc networking concept closer towards realization beyond the academic circles. The industry is now embracing this emerging technology, and in the recent years, several vendors have provided handheld soldier radios with MANET capability. The possibility for individual soldiers to be part of the entire NBD and to both provide and request services in the network, paves the way for a new and evolving service-oriented way of thinking. This will require a need for optimized solutions to disseminate service information in the network in an efficient matter.

7.1 Design issues

We know the characteristics of mobile ad hoc networks, and remember that the idea behind a MANET is based on a non-infrastructure approach. Most of the existing service discovery architectures are primarily designed for fixed networks, and are due to the special characteristics of mobile networks not directly applicable for MANETs. Hence, tailor-made solutions specific to MANETs are often chosen in favor of more generic solutions. However, since actual implemented MANETs differ both in size, equipment, applications and objectives, we can find a great variety of proposed service discovery architectures for MANET to solve specific purposes. Some of the solutions focus purely at scalability in order to support hundreds or even thousands of nodes. Some solutions seek to minimize latency in the discovery process, while other are focused on reducing the control message overhead to support bandwidth-constrained environments.

Considering the three basic service discovery categories presented in Chapter 4 (decentralized, centralized, and distributed), service discovery mechanisms for MANETs can be further subdivided in the following points:

1. General mechanism *independent of* the underlying routing protocol.
2. Specialized mechanism *integrated with* the routing protocol.

Routing protocols in MANETs can be either *reactive* or *proactive*. A reactive routing protocol finds a route through the network on-demand and on-the-fly, whereas a proactive routing protocol calculates and stores the route at regular intervals. Thus, point two above can be further subdivided into the following two categories:

1. Mechanism integrated with a *reactive* routing protocol.
2. Mechanism integrated with a *proactive* routing protocol.

Most of the MANET service discovery proposals belong to the first category, and place service discovery at a layer above routing, i.e. the application layer. The greater part of the proposed service discovery solutions in this category are based on a service coordinator based architecture as a mode of operation, while others are based on a decentralized architecture. In a centralized scheme, as we recall from Section 4.2.1, services are registered in a central entity, and clients search for services in this registry. In MANETs, a fixed and stable infrastructure cannot be assumed, and the link to the service coordinator may be unpredictable and represent a single point of failure. Thus, a distributed or hybrid scheme is more suitable both for MANETs and for dynamical tactical networks.

The latter two categories in the listing assume that, since an ad-hoc routing protocol such as OLSR [21] or AODV [24] must be present in a multi-hop ad-hoc network, it is convenient to use a mechanism to exploit the routing layer for efficient dissemination of service control messages. In the literature, this strong coupling between routing protocol and service discovery is also named *cross layer* service discovery. Cross layer approaches demand proprietary solutions and modifications of existing standards, as well as being of limited use if IP cryptography is present. These drawbacks have limited cross layer service discovery from becoming widespread so far,

and might limit its usability in an NBD setting. However, cross layer solutions have low overhead and should thus be investigated for use in disadvantaged grids where bandwidth is the limiting factor.

Given the outlined service discovery categories for MANETs we will now first discuss different generic (or application layer) approaches, and then cross layer techniques.

7.2 Application layer service discovery mechanisms

The general mechanisms made for MANETs places service discovery in the application layer. These mechanisms will create an overlay on top of the network layer to disseminate service advertisements, requests and replies in the network. Some advantages using such solutions can easily be listed:

1. It is feasible to create pervasive service discovery architecture across different network domains, since no assumption about the underlying network is made.
2. The architecture can easily be based on existing standards.
3. A modular and layered approach is maintained.

However, application layer service discovery architectures also introduce some disadvantages that will be discussed later. We present PDP and Konark as examples of application layer service discovery, and Sailhan05 as a directory based application layer service discovery solution. Even if Sailhan05 can be characterized as an application layer protocol, it does to some extent introduce the need for cross layer optimizations. Pure cross layer protocols for both reactive and proactive routing protocols are covered in the last section of this chapter.

7.2.1 Pervasive Discovery Protocol (PDP)

The pervasive discovery protocol (PDP) was specially designed to work in ad hoc networks [17]: The protocol does not need a central server, so no infrastructure is required. One of the key objectives of the PDP is to minimize battery use in all devices, especially in the most limited ones. This means that the number of transmissions necessary to discover services is reduced as much as possible. A device announces its services only when other devices request the service. Service announcements are broadcast to all the devices within range, all of which will get to know about the new service simultaneously at that moment, without having to actively query for it. Devices have a cache where these announcements are stored. Service announcements include not just the services offered by the device itself, but all the known services of the requested type. PDP takes into account the heterogeneity of the devices, reducing the consumption of the most limited ones by prioritizing the reply of less limited devices and allowing the others to abort their answer if they have nothing new to say.

Each device has a PDP User Agent (PDP_UA) and a PDP Service Agent (PDP_SA). The PDP_UA is a process working on behalf of the user to search information about services offered in the network. The PDP_SA is a process working to advertise services offered by the device. The PDP_SA always includes the availability time *T* of the device in its announcements. Each device

has a cache containing a list of the services that have been heard from the network. Each element of the cache has two fields: The service description and the service expiration time. The service expiration time is the estimated time for the service to remain available. This time is calculated as the minimum of two values: The availability time of the local device, and the service announced lifetime. Entries are removed from the cache when they time out.

PDP has two mandatory messages: PDP_Service_Request, which is used to send service requests, and PDP_Service_Reply, which is used to answer a PDP_Service_Request, announcing available services. Additionally, PDP has one optional message: PDP_Service_Deregister, which is used to inform that a service is no longer available. PDP supports two different kinds of queries; *one query – one response* (1/1, i.e. the application is interested in the service, not in which device offers it) and *one query – multiple responses* (1/n, i.e. the application wants to discover all devices in the network offering the service).

The listening port for PDP is 3000 (it is not reserved by the IANA). This is the destination port for all PDP messages. All PDP messages are multicast to the local-scope multicast address 239.255.255.253. The default TTL to use for multicast is 255. In isolated networks, such as one hop ad hoc networks, broadcast will work in place of multicast. PDP is freely available in a Java implementation and for simulations in the NS2 network simulator.

7.2.1.1 PDP in NBD

PDP, being designed for low overhead and ad hoc networks, could be useful in an NBD setting. For example, it could be used to detect registries and other services in a local area. The protocol has low overhead, and being an application layer protocol it does not have any of the drawbacks associated with cross layer solutions. On the other hand, it is not Web services related, and its simplicity leaves us wanting when it comes to semantic issues (as has been mentioned for many of the other solutions as well, e.g. SLP). Also, it is an experimental protocol and not a standard so far. Scalability is also an issue; PDP would be suitable only in small scale networks since it is based on broadcast/multicast.

7.2.2 Konark

Konark [4] is a decentralized service discovery architecture based on a peer-to-peer model using lightweight HTTP servers on every participating node. Conceptually, as Konark uses IP-multicast to advertise and discover services, the underlying network must support an IP-multicast enabled ad-hoc routing protocol. Konark defines its own description language based on XML, loosely based on WSDL. HTTP and SOAP are utilized to handle service delivery, and the HTTP servers are used to handle service delivery requests. The Konark architecture maintains a tree-based structure to cope with service scoping and classification. The XML format support search for either “all”, “generic” or “specific” services in each category. The requests can be made using either keywords or a more fine-grained service description if a specific service is desired. Hence, a node can choose to offer its services based on different levels of detail, and other nodes in the network can search for services based on their desired level of detail. Using this classification, a node can do a brief search for “a general printer”, or choose to do a more detailed search for “a

color laser printer on the second floor". As Konark supports both proactive and reactive service advertisements, both service producers and consumers can actively discover and advertise services on a need basis. In the discovery process, the service request is sent to a fixed multicast group, and all the nodes with a matching service ought to respond. A time-to-live (TTL) field is specified in the service advertisement process, enabling local caching of service descriptors on each network node. The service provider should refresh services before their TTL expires, otherwise the descriptor will be deleted from the caches around the network. The use of timeouts and caching is a way to make the system more robust against unexpected failures such as loss of connectivity or hardware failure.

The Konark architecture includes an algorithm for distributing the service descriptor tree in the ad hoc network using a kind of gossip protocol. Incoming service messages are combined with the cached view of services and further distributed to the network using IP multicasting. An implementation for Konark exists for both Java and J2ME CLDC/MIDP.

7.2.3 Sailhan

Sailhan et al [20] proposed a directory based service discovery architecture aimed at large scale ad hoc networks. In this survey, we take the liberty to name the architecture *Sailhan05*. Directories in Sailhan05 are distributed and deployed dynamically, and form a virtual backbone of nodes exchanging service requests and replies using WSDL service descriptors. Sailhan05 outlines an architecture independent of the routing protocol, and communication between directories is done using a bordercasting technique between directories (in contrast to flooding). The bordercast technique used in Sailhan05 works exactly like the familiar multi point relay (MPR) selection and MPR flooding that is used in the OLSR routing protocol [21]. In fact, the architecture can take advantage of the OLSR MPR election itself instead of creating the bordercast overlay on its own.

The dynamically allocated directory agents are deployed so as at least one directory is reachable in at most a fixed number of hops. Directories then cache the descriptions of services available in their vicinity and use Bloom filters [22] to summarize the content of the directory by hashing the set of WSDL-based service descriptions. One important feature is that Bloom filters do not require a predefined static set of keywords to define certain services while they provide a compact summary of the directory's content.

Using the virtual backbone of directories, efficient service descriptors, and bordercasting of service descriptors, Sailhan05 proposes a feasible solution for service discovery in large-scale ad hoc networks. However, it is unclear how well the system will cope with highly dynamic and small networks. Also, WSDL, being XML based, is verbose and could perhaps be too bandwidth consuming for use in disadvantaged grids if used indiscriminately.

7.3 Cross layer service discovery mechanisms

Generally speaking, *cross layer design* refers to protocol design done by actively exploiting the dependence between protocol layers to obtain performance gains [23]. By doing this, cross layer solutions violate the modular layered approach. However, in many situations, cross layer interactions are inevitable to eliminate the redundancies associated with repeating similar tasks found on adjacent layers, which normally will lead to increased control message overhead [32]. Two examples of important cross layer optimizations can be shown:

1. Routing protocols need to do neighbor- and link sensing using beaconing on the network layer. As link sensing is faster and more precise on the link layer, routing protocols can exploit link layer information using a cross layer API.
2. Some application layer protocols need to disseminate topology information to gain knowledge about the network structure. As routing control messages disseminate the entire network and outline the topology on the network layer, services at the layer above can utilize this functionality through cross layer interchange in contrast to create a separate overlay.

As implied by those two examples, violation of a layered architecture involves giving up the luxury of designing protocols at different layers independently. With respect to service discovery, a cross layer integration of the service discovery architecture with the routing protocol seems to bring considerable optimizations. This will, however, require thorough knowledge about the routing protocol functionality and implementation. The remaining part of this section focuses on proposals involving cross layer service discovery in mobile ad-hoc networks.

7.3.1 Routing

To get a full understanding of the cross layer service discovery proposals, we will now give a short introduction to the different approaches regarding routing in mobile ad hoc networks in this section for the future reference. In the development of mobile ad-hoc networks, mainly two different routing approaches are considered: *reactive routing*, and *proactive routing*. Reactive routing protocols will only attempt to discover routes between nodes on-demand. Proactive routing protocols on the other hand, seek to maintain routes to all nodes regardless of upper layer communication demands. A variety of protocols are proposed in both categories. Some of the routing protocols are considered purely academic, while others are heavily deployed in real-life applications. Some variations also combine the reactive and proactive approach in a hybrid or multimode solution. In this report we will focus on proposals supporting the work done by the Internet Engineering Task Force (IETF).

The IETF is chartered to develop one reactive, and one proactive routing protocol through the MANET working group. The work has concluded in two RFC standards, namely AODV [24] and OLSR [21]. The group is now working on the development of the successors to AODV and OLSR, namely DYMO [25] and OLSRv2 [26]. DYMO is built upon AODV's distance vector

routing. OLSRv2 evolves from OLSRv1 and improves the initial OLSR version with increased extensibility, more efficient messaging and lower complexity.

7.3.2 Reactively Routed MANETs

SEDRIAN [27] is a decentralized service discovery approach using AODV as routing protocol. Service information can be based both on an optimized description using a 128-bit Universal Unique Identifier (UUID) for generic services and a more descriptive language. The latter format is used to advertise special services that cannot be described in the UUID.

SEDRIAN exploits AODV by encapsulating three specific packets in the AODV RREP message: Discovery Request (DREQ) contains a request for a UUID-based service. Discovery Reply (DREP) contains a reply to a discovery request. The last message is an Advertisement Message (ADVM) that contains the advertisement of a special service that is provided.

Since SEDRIAN uses the AODV RREP message to disseminate discovery requests, replies, and advertisements, the proposal brings some changes to the original AODV protocol. The RREP is not originally used as a broadcast message, and some additions are needed to avoid packet loops. The drawback by introducing this extension to AODV is that all nodes in the network must support SEDRIAN in order to make the discovery process work.
The proposal by Engelstad et al [28] utilizes AODV piggybacking in a similar, but more coherent matter. The article shows how both distributed and hybrid service discovery architectures can be integrated with the AODV routing protocol. Service discovery requests (SREQ) are piggybacked on routing request packets (RREQ), and service discovery replies (SREP) are piggybacked on routing reply (RREP) packets. Hybrid architecture is also supported. Using the hybrid scheme, the service coordinator announcements are piggybacked on RREQ packets and service registrations are piggybacked on RREQ packets. Thus, there is no need to change the original AODV code, besides implementing the functionality to allow piggybacking of service discovery messages.

7.3.3 Proactively Routed MANETs

Although proactive routing introduces more control message overhead, the overall overhead including cross layer service discovery is fairly comparable for the two different routing approaches. This is due to the fact that routes are maintained continuously and no extra route computation is necessary as a cause of the service discovery and advertisement process. The proactive process leads to less latency involved in the discovery process, compared to service discovery in reactively routed networks.

Most of the proposals involving cross layer service discovery using proactive routing protocols are built based on similar ideas. In this survey we present three different, albeit quite resembling proposals.

Jodra et al [29] presents a solution integrating service discovery with the OLSR routing protocol. The different OLSR messages [26] share a common message header. Utilizing this header, a new

message can be described. In Jodra's proposal, the new message is called Service Discovery Message (SDM). The SDM packet can contain either a service advertisement or a query. The proposal also introduces a service cache for each node in the network. The cache will store all services that are available for the node, both local and foreign. Whenever a node wants to use a service not stored in its local cache, it will send a query asking for that particular service using the SDM query message. The SDM messages will be forwarded by the MPRs in the network. Upon receiving such a SDM query message, a node will check whether its local services correspond to the service asked for in the query SDM. If this is the case, it will send an advertisement message announcing the requested service. The answer will be MPR flooded.

As the complete SDM message is only 8 bytes, and thanks to piggybacking of the SDM on the OLSR packets, simulations show that SDM adds an almost insignificant overhead to the network. However, it should be noted that this efficient, albeit limited message format, cannot cope with more advanced and detailed service descriptors.

A similar proposal utilizing OLSR as routing protocol is Lightweight Service Discovery [30]. CrossROAD [31] emphasizes the need for cross layer solutions and introduces an overlay structure exploiting OLSR that can be used for a variety of applications, such as service discovery and peer to peer applications.

7.4 Cross layer service discovery and NBD

As previously mentioned, using cross layer approaches violate standards and can cause problems when using IPsec. But, cross layer techniques are still worth considering in military tactical networks, due to the scarcity of resources in such networks.

Sailhan05 and Konark use XML and may be too bandwidth consuming for use in tactical networks. However, seeing as XML compression could reduce a fair amount of the overhead [7], these solutions could perhaps still be useful.

Even if the majority of the service discovery solutions and proposals address service discovery at the application layer, a cross-layer integration of the service discovery architecture with the routing protocol seems to bring considerable optimizations. Although this solution will violate a modular layered approach and hinder easy interchange of routing protocols, the benefits are indisputable both in proactively and reactively routed MANETs.

It is important to search for a service discovery architecture that makes it possible for intermediate nodes, not running any service discovery code, to forward requests, replies and advertisements on behalf of other nodes. It is obvious that using an application layer design, this is feasible. But even if the service discovery architecture is based on a cross layer design, one should aim for such a transparent concept.

8 Security considerations

Due to the central role of service discovery in a dynamic architecture, a service discovery service may be subject to strong security requirements. From a security perspective there are two resources to be considered for protection. The first resource is the service discovery service itself. The availability of this service is vital to the success of a dynamic SOA, and as such will be an attractive target for potential enemies and must be protected. The second resource that needs protection is the data stored and supplied by the service discovery service. This is metadata describing the services and it may contain information which should not be displayed to all users of the service discovery service. This might be sensitive metadata that for instance describes the capabilities, like location and coverage, of a sensor service and thus also reveals the sensor itself. From this we can derive security requirements that typically include:

- Confidentiality:
  - Of the information possessed by the service discovery service.
  - Of the information supplied by a service requester when requesting a service.
  - Of the information supplied by service providers to the service discovery service.
- Integrity:
  - Of the information stored within the service discovery service. (This may again impose integrity requirements on the information supplied to the service discovery service by service providers.)
  - Of the communication between service providers, service requesters, and the service discovery service.
- Availability:
  - Of the service discovery service and the information intended therein.

The confidentiality requirements depend on the sensitivity of the data that is communicated through service discovery. Traditionally, military systems enforce the principle of need-to-know and the confidentiality requirements include requirements on only releasing classified information to those with sufficient privileges. In addition to the confidentiality of the resource content/description, the mere existence of the resource may also be confidential. In order to fulfill the confidentiality requirements, encryption and access control are both central mechanisms. Also notice that encryption again requires some type of key distribution, while access control typically requires authentication to be performed. In addition to the service discovery service authenticating service requesters/providers, it may also be required for service requesters/providers to authenticate the service discovery service.

The integrity requirements on a service discovery service also heavily depend on the scenario. A minimalistic requirement would be that the service discovery service correctly responds to service requests, based on the information provided to it by service providers. Strict integrity requirements, on the other hand, could require that the information provided to service requesters is both correct and complete. While integrity during communication and/or storage can be ensured by for instance using digital signatures, the trustworthiness of the original information

depends on the underlying trust relationships. Furthermore, in order to ensure that the information is both complete and correct, mechanisms to remove stale entries and obtain new updates in a timely fashion may be required.

The availability of the services that are to be located through service discovery directly depends on the availability of the service discovery service itself. Thus, failing to ensure service discovery availability could jeopardize the effectiveness of the system as a whole. In order for a service discovery registry not to become a single point-of-failure, replicated or distributed approaches are preferable. Still, availability also depends on a high number of other factors. For instance, the availability of a service discovery service also depends on network connectedness and capacity, as well as on the availability of the potentially required supporting services (e.g., identity/authentication providers or key management services). Depending on the environment, mechanisms to prevent denial of service attacks may also be necessary. Examples of denial of service attacks may include overloading the discovery service with requests or registering a high number of bogus services.

While fully distributed service discovery (e.g., based on multicast/broadcast of service requests) is advantageous from the point of view that a service does not depend on a service registry in order to be discovered, such architectures may impose additional complexity if the confidentiality of service requesters is to be ensured. While a request sent from a service requester to a centralized and known service registry may be encrypted using the public key of the registry, the exact recipient of a service request in a fully distributed solution is unknown. Thus, ensuring confidentiality of service requests in a fully distributed architecture may require group keys or comparable mechanisms.

The distributed scenario is illustrated in the bottom half of Figure 8.1, and may start with every service provider publishing the availability of their services by broadcasting or multicasting a message to the rest of the group. The service provider can not easily know in advance who will receive the message, thus confidentiality protection may be difficult to provide (apart from encrypting the announcement using a group key). The service provider can digitally sign the message in order to provide integrity protection. The similar situation exists when a consumer issues a request. When multicasting or broadcasting a request there is no way of knowing who will receive it. The reply can however be protected as the provider can identify the client. The obvious advantage of the distributed model is that it might provide a more robust service discovery and thus from a security perspective enhance the availability.

The top half of Figure 8.1 shows service discovery using a centralized registry. In this case all requests are sent to a known node in the network and can thus be protected. This applies to both the publication and discovery steps. From a security perspective, the obvious disadvantage of this centralized solution is the danger of reduced availability due to a single point of failure, although a replication strategy may alleviate this problem. Both models have pros and cons, and the added value of each must be considered against the threat scenario.

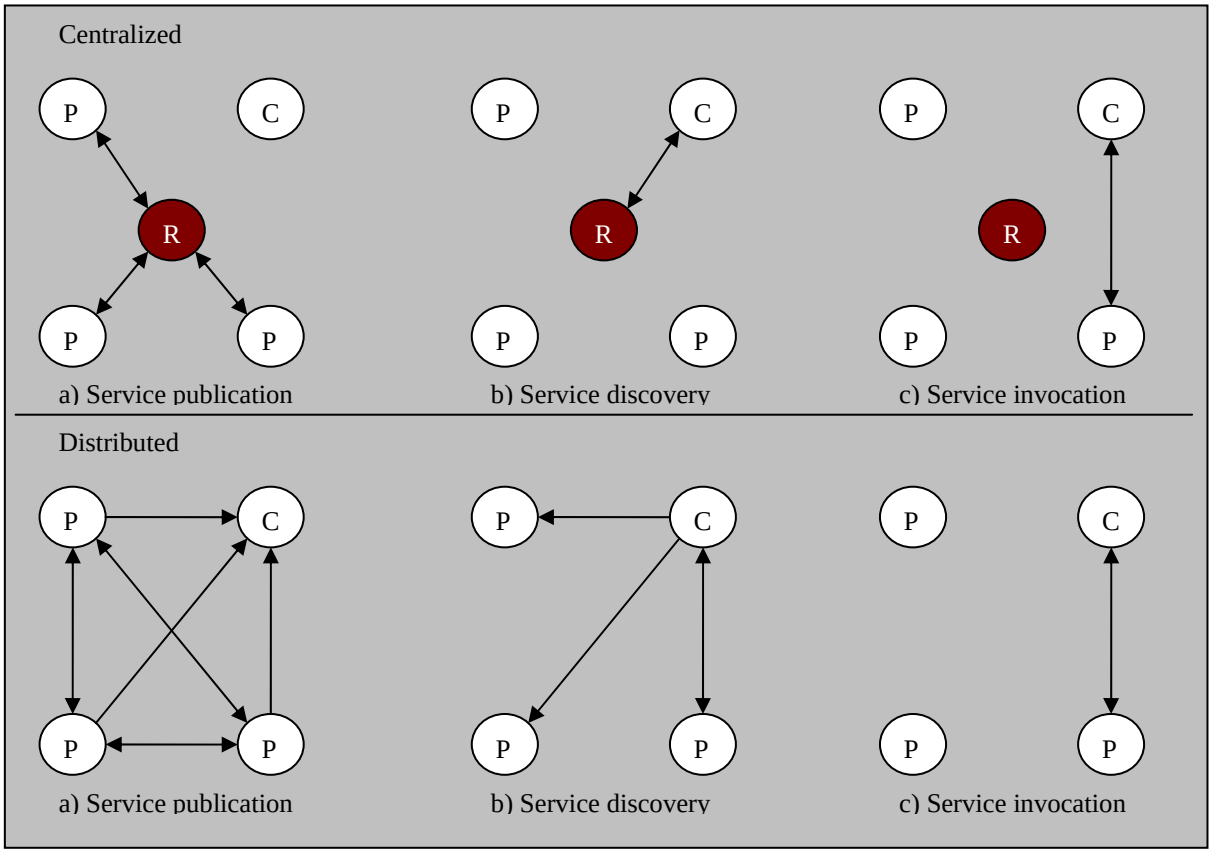

Figure 8.1 Centralized vs. distributed service discovery. Service registries(R), service consumers (C), and service providers (P) are shown.

The previous paragraphs have outlined some high level requirements for service discovery security. Implementations of these requirements should, as far as possible, use open standards to ensure interoperability. XML and related security standards, like for instance XML Signature, XML encryption, SAML, XACML, and Web Services Security, should provide the foundation for securing the service discovery functionality.

Earlier work at FFI involving both service discovery and security has concentrated on the use of centralized registries such as UDDI [10]. In an experiment conducted at CWID 2006, all records stored within the UDDI registry were labeled and signed. The label included a representation of the sensitivity of the information stored in the record, while the signature provided protection against unauthorized changes during storage. Access control to the UDDI records was performed by comparing the privileges of the user against the sensitivity label. In addition, by signing and encrypting all messages to and from the service discovery service, we provided both confidentiality and integrity protection to messages in transport. The signatures also provided client and server side authentication by being used in combination with X.509 certificates and a PKI. As this work provided protection for both the service discovery service and its contents, it serves as a good example of what mechanism a future service discovery service may deploy. For further details on this work, please refer to [18].

9 Semantic service discovery

There has been much effort in the research community on Semantic Web Services (SWS) [41], which is believed to facilitate run-time, capability-based discovery of services. Annotation of service advertisements with rich, explicit semantics can improve service selection and invocation significantly. Therefore, SWS will be essential for solving the problem of run-time service discovery in dynamic environments. However, the efforts in this research area are focusing mostly on description of services, whereas the distribution of service advertisements is based on Web services efforts. We argue that focusing on both these aspects will give the best result for dynamic environments.

By SWS discovery in dynamic environments we mean a way for a client to programmatically, at run-time, discover available services according to their capabilities, and utilize them. A SWS discovery mechanism must provide a means for services to specify that they are an instance of a given class, according to an ontology. This can enable the client and the service to "understand" each others' interaction protocol, interface and message semantics, enabling on-the-fly service invocation.

A recurring issue in service-oriented systems has been that service descriptions mostly are based on syntactical descriptions, rather than semantic meaning. This means that it is impossible to create a fully dynamic SOA, because one has no means to understand the semantics, that is, what to expect from an on-the-fly service invocation.

An issue with the Web services standards of today is that one has no way of describing what the service is capable of doing: At least not in a well-defined, machine-processable way, allowing the computer to reason about a service and compositions of services. SWS are an extension of current Web services technology, and facilitate dynamic service selection and service utilization, according to our framework. This is done by describing services using a formal language, or ontology. One of the driving factors of SWS is automation, especially in the area of service discovery. Keyword-based searches are ambiguous, and say nothing about the capabilities of a service, at least not for computers.

The model for SWS discovery initially is that users have goals or needs that must be resolved. There will probably be some abstraction from the concrete goals of a user to a more general query for services with the wanted capability. This is the service selection part, and it is often called matchmaking. Because the matchmaking process is based on one or more ontologies, it is possible to either broaden the search (generalization) or narrowing the search (specialization). The resulting service descriptions (or recommendations) then serve as the basis for service selection, which is either done at the client, or by some middle agent, e.g. a broker. When a service has been selected, it can be invoked. This is usually done by means of Web services standards like WSDL and SOAP.

9.1 The NATO Discovery Metadata Specification (NDMS)

The NATO Discovery Metadata Specification (NDMS) has been developed in order to support resource discovery in a NATO context. NDMS consists of a set of discovery metadata elements intended to describe data assets (i.e. systems and services that provide access to data, such as relational databases and Web services) that are available in NATO shared spaces. NDMS is based on the Dublin Core Metadata Element Set (DCMES), with additional elements added in order to provide specific NATO information requirements (e.g. additional security metadata). NDMS is intended to be a minimum set of metadata elements that software developers should use to describe their data assets. The vision is that these metadata descriptions will thus allow both human users and autonomous systems to find, access and use data assets [58].

However, being merely an extension of the DCMES, NDMS suffers from the same shortcomings as its parent element set when it comes to being used for machine interpretation. The Dublin Core elements, and thus the NDMS elements, lack explicit formal semantics which are crucial in order to perform semantic service discovery through the use of machine interpretation and reasoning. Dublin Core elements are described using informal semantics in terms of mere textual specifications, which lay open to ambiguity and misinterpretation that can result in loss of reliability and reduced benefit for service discovery [60].

In contrast, machine reasoning requires that the semantics of the vocabulary used to describe the data are defined in a formal language (e.g. first-order logic). As a result that NDMS lacks formal semantics, there is limited ability for machines to make direct use of the metadata elements in terms of performing semantic service discovery through the use of machine reasoning.

There exist a number of vocabularies with formal semantics intended to facilitate semantic service discovery (e.g. OWL-S, WSMO) which are more suited to the task than NDMS. Although none of them seem so far to have been much used outside of research environments, it is fair to say that OWL-S is the one that has received most attention.

OWL-S is an ontology for describing Web services, written in W3C's Web Ontology Language (OWL), and aims to facilitate service discovery, invocation and composition. An OWL-S description describes the input, output, preconditions and effects of invoking a specific Web service, and provides adequate formal semantic descriptions needed for machines to reason over. The inputs and outputs are described as instances of concepts in imported domain ontologies [59].

An OWL-S service description consists of three sub-parts; profile, process model and grounding:

- The profile gives a high level view of the service in terms of what it does (the inputs, outputs, preconditions and effects).
- The process model describes how the service works in terms of the background process performed when invoked. The service might be atomic (one single service is invoked), or it might be a complex process similar to a BPEL model that involves invoking several services and combining the results.

- The grounding describes how to access the actual service. Commonly the grounding binds the OWL-S process model to the WSDL description, in order to be invokable by an OWL-S engine. Note that an OWL-S description of a Web services does not replace the WSDL description, but it co-exists with it and complements it.

OWL-S facilitates service discovery and matching through reasoning over sub/super- and equivalent classes, as well as over specific goals that one wishes to achieve.

10 Research challenges

The process of locating and identifying a service, known as service discovery is an important part of any SOA. However, it is a great challenge to achieve this in dynamic environments such as military tactical systems. Summarizing the previous chapters: We need solutions suitable for mobile environments and low bandwidth networks, and preferably a discovery solution that can function both with and without the support of a registry.

Current Web service discovery is not sufficient for the NBD battlefield. Registries today are designed for high bandwidth, wired networks and function well only in static environments. In a dynamic environment both UDDI and ebXML will leave stale data in the registry if a service does not de-register and leave the system cleanly. Ideally, the registries should have no single point of failure (i.e. use a distributed solution), they should contain liveness information to avoid the problem of stale data, and they should have a functional equivalent in MANETs. If the registry is unavailable then service discovery should still be possible using a fallback mechanism. WS-Discovery, when it has matured into a standard, could perhaps be used as such a fallback. However, this would be suitable only in small networks, since its broadcast/multicast model does not scale to a large number of users. Furthermore, it may incur too much overhead in low bandwidth MANETs, where one should most likely consider a cross layer solution.

In Chapter 4 we introduced the fully decentralized client-service model, along with the client-service-directory model which can be centralized or distributed. Today, Web services use the latter of the two models. We suggest a hybrid model; use a registry (i.e. a client-directory-service model) if available, and use a client-service model as fallback.

An operational network is complex (see Figure 10.1). There are different levels with different communication needs and solutions. It is apparent that a single service discovery mechanism will not meet all demands. 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.

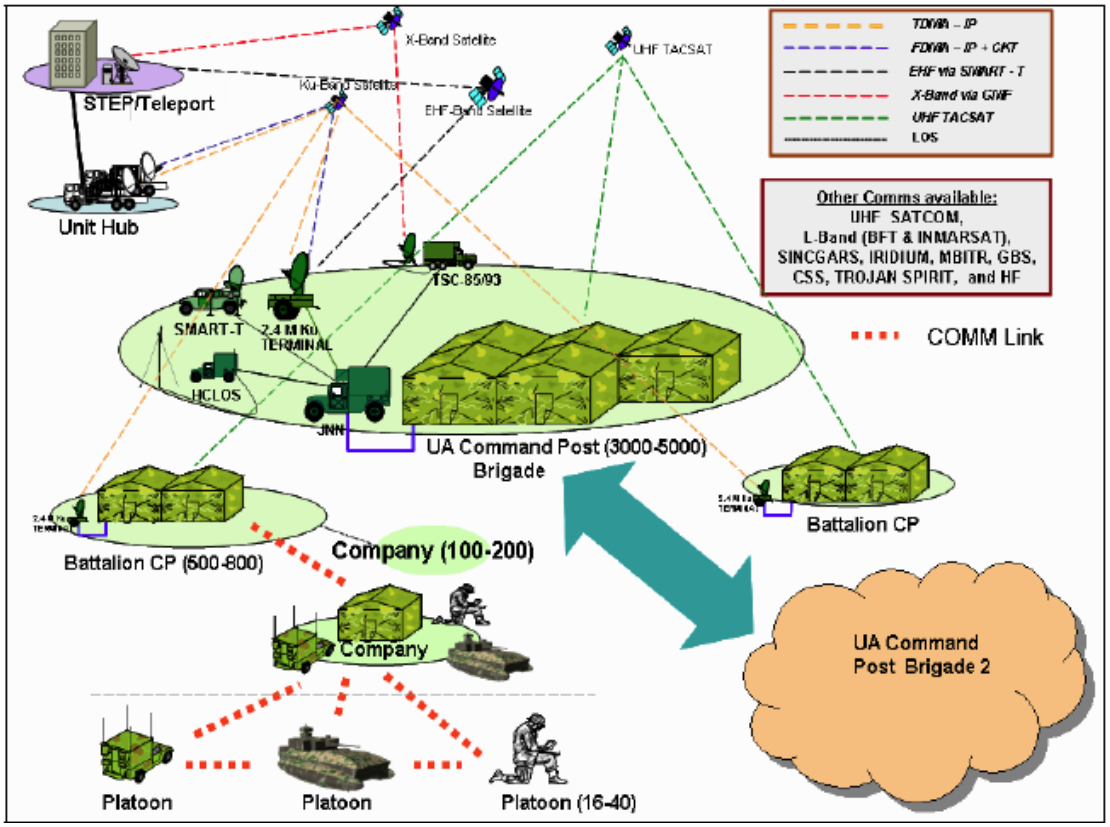

Figure 10.1 Operational network (fetched from [55])

Looking at the different operational levels in context (see Figure 10.2), we can see that the characteristics vary from level to level. A strategic network has infrastructure and is typically not dynamic. A tactical deployed network is more dynamic than a strategic network, typically depending on radio or satellite communication. A mobile tactical network, i.e. networked single units taking part in an operation, typically yield 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 than on the strategic level. A mobile tactical network will have and use the least number of services. Not only does the limited bandwidth at this level limit the possible amount of services and communication, but also the need for services will be location and mission specific.

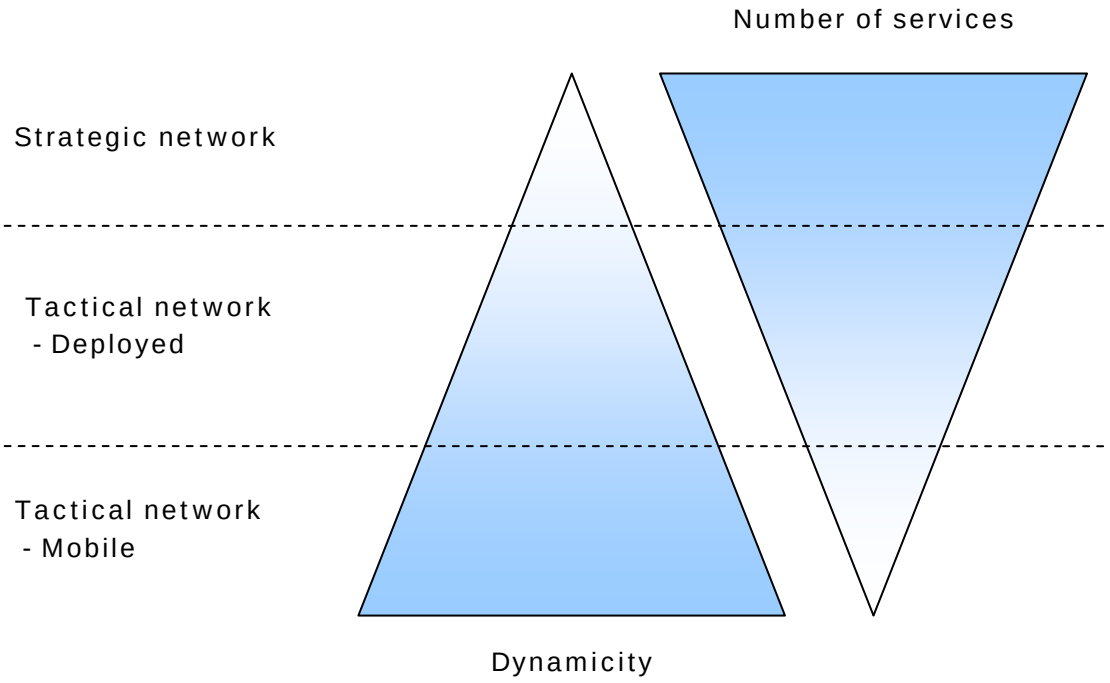

Figure 10.2 Domain complexity

Considering the differences between the different operational levels, we suggest that one should use one discovery mechanism per level, as shown in Figure 10.3. Naturally, there are still many open issues that need addressing, especially regarding interoperability.

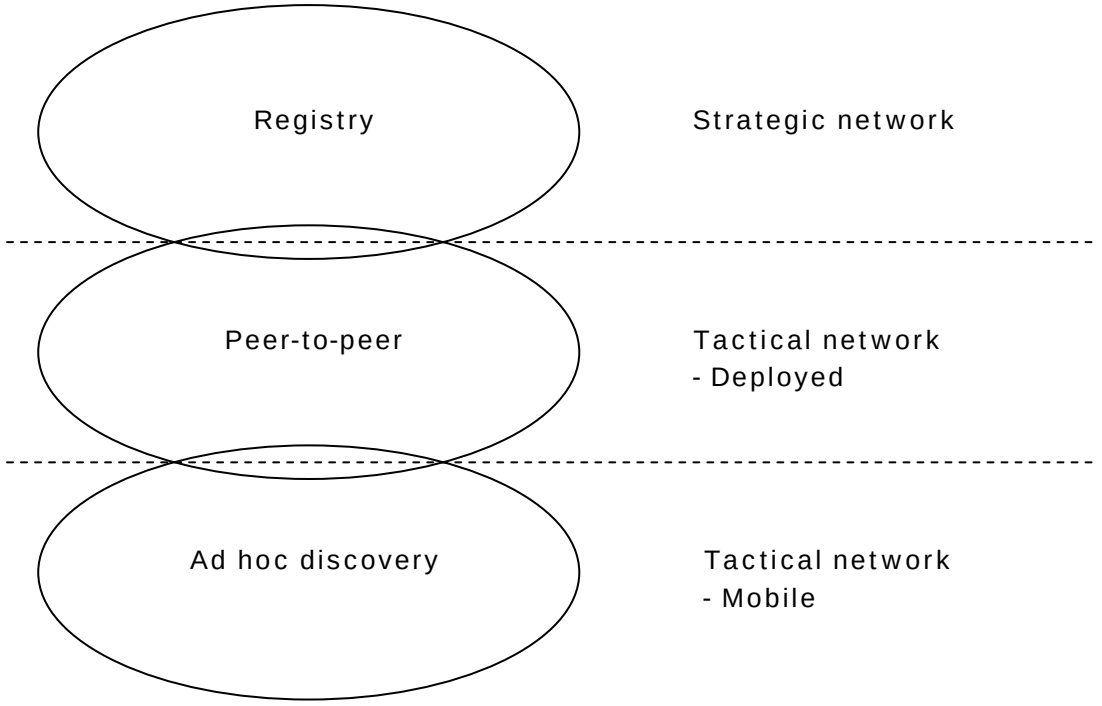

Figure 10.3 Suggested service discovery mechanisms for each operational level

Below is a non-exhaustive list of aspects that need to be addressed:

- How should the different systems interact with each other, across the different types of networks?
    - How should the system behave with/without a registry present?
- How do you express services and how do you search for them?
    - Issues with querying and semantics.
- What about cooperation with other nations? At which level(s) will the other nation(s) be allowed to connect their systems?
    - This leads to several new issues regarding network load and security considerations.

Let us now look at the different operational levels in turn, relating possible solutions for each level to current industry standards and current experimental solutions that are still subject to further research.

In the strategic network, which is static and has high available bandwidth and is Internet-like, one can use COTS based on standards to accommodate the service discovery needs. Both UDDI and ebXML are current OASIS standards, and implementations are available from several vendors. Both solutions are suitable for business use over the Internet, and should also be usable in a strategic network where there is no mobility and stale data is less of an issue. Both UDDI and ebXML can be configured as federations of registries, thus avoiding having a single point of failure and gaining tolerance to partial failure. From a NATO perspective, interoperability is a very important issue of service descriptions and service discovery. FFI project "Semantini" is currently researching Web services and semantics, and lean towards ebXML as the preferred registry technology due to its superior flexibility and functionality when compared to UDDI.

In a tactical deployed network, on the other hand, the current registry solutions are too static. A solution that scales to thousands of users and yet can handle some mobility is needed. Also, bandwidth is lower than in a strategic network, so federations of fully or partially replicated registries are probably not desirable, even if one could overcome the problem of stale data. At this level there is currently no standardized solution available that is suitable. We are currently investigating experimental solutions that can accommodate the needs. One solution that seems promising and that we will experiment with is the Service Oriented Peers solution based on peer-to-peer techniques that is mentioned in Section 6.5. This is currently part of the focus of FFI project "Secure Pervasive SOA".

In a MANET there is potentially high mobility, and bandwidth is low. Also, there are few services and few participants in the network at this level. To limit overhead, one should investigate cross layer service discovery solutions, even when considering the drawbacks discussed in Section 7.1. Even though you lose some of the flexibility (and standard compliance) of the service discovery solution by combining it with the routing layer, you gain the ability to express a few pre-defined services and distribute information about them with low overhead. At this level there is also a lack of COTS and standards, so any solution that emerges here is

currently experimental. Also, even if one uses a seemingly stand-alone service discovery mechanism at this level that does not rely on a registry, it should still be able to use a registry (or peer-to-peer service discovery mechanism) when available. Cross layer service discovery is currently being investigated as part of FFI project "NORMANS", which focuses on the future Norwegian soldiers' equipment and needs.

11 Summary

NATO has so far not chosen any solution for service discovery; some initiatives lean towards UDDI whereas others suggest using ebXML. If one chooses a proprietary national solution, then one will need to use gateways for interoperability with other forces in NATO. This would facilitate participation in coalition forces with cooperation at all the operational levels. Interoperability is important, so the solution NATO eventually chooses should be employed. However, considering only the technical suitability of service discovery solutions, we can summarize the report in the following theoretical evaluation:

Currently, registries are the most mature of the service discovery solutions for Web services. Both UDDI and ebXML are standardized by OASIS, and provide basic static registry functionality. Due to ebXML's flexibility when compared to UDDI, we suggest that one should look into using that solution for NBD, at least as a registry solution at the strategic level. Discovery solutions for use in tactical deployed and mobile networks are another issue, and there are currently no clear candidates for use at these two levels. We suggest further experimentation with peer to peer and both application layer and cross layer solutions for service discovery for use on these operational levels. In tactical networks you should have support for service discovery that functions both if a registry is present and also if it is not. There is currently no clear candidate for use at these levels, since using a registry such as ebXML may prove to be too bandwidth consuming, and also it does not support liveness information which is crucial for services at the tactical level. The tactical level typically consists of multi hop ad hoc networks, and solutions suitable for such networks need to be investigated. We are not aware of any standardized solution that can be employed for this task , since current standardized LAN discovery mechanisms such as SLP and uPNP are tailored for use in a local area office environment, where you have wired connections or single hop ad hoc networks. They would probably not scale and perform well in a tactical multi hop ad hoc network. Furthermore, these standardized LAN mechanisms do not implement any security mechanisms, which would be necessary in an operational network. Also, SLP and uPNP are not Web service related, so one should keep an eye on developments around WS-Discovery, a LAN discovery solution for Web services that is currently a draft specification. At the personal network level, however, it would be possible to employ a standardized service discovery mechanism like for example Zeroconf to interconnect the soldier's different equipment and wearable computer, as was suggested in a previous FFI report (see [16]). That, however, is a solution for interconnecting and configuring equipment for network use, and is not Web service related. So, even if Zeroconf were to be applied for this purpose, one would need to investigate how discovery of Web services and their utilization would fit in.

References

[1] T. Gagnes et al, ”A Conceptual Service Discovery Architecture for Semantic Web Services in Dynamic Environments”, 22nd International Conference on Data Engineering Workshops, 2006, pp.74-74

[2] T. Gagnes, “Assessing Dynamic Service Discovery in the Network Centric Battlefield”, IEEE Military Communications Conference (MILCOM), October 2007

[3] T. Gagnes et al, “An Architecture for Service Discovery in a Network Based Defence”, FFI-Notat-2006/00115

[4] S. Helal et al, “Konark – A Service Discovery and Delivery Protocol for Ad-hoc Networks”, Proceedings of the Third IEEE Conference on Wireless Communication Networks (WCNC), New Orleans, March 2003.

[5] S.A.H. Seno et al, “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

[6] D. S. Alberts et al, “Network Centric Warfare: Developing and Leveraging Information Superiority”, CCRP Publications Distribution Center, 1999.

[7] K. Lund et al, “Using Web Services to Realize Service-Oriented Architecture in Military Communication Networks”, IEEE Communications Magazine, Special issue on Network-Centric Military Communications, October 2007.

[8] F Zhu et al, “PrudentExposure: A Private and User-centric Service Discovery Protocol”, Proceedings of the Second IEEE International Conference on Pervasive Computing and Communications, PerCom’04, Orlando, Florida, USA, March 2004

[9] L. Gong, “JXTA: A Network Programming Environment”, IEEE Internet Computing, vol. 5, no. 3, May/June 2001.

[10] OASIS. UDDI Version 3.0.2. http://uddi.org/pubs/uddi_v3.htm

[11] UPnP Forum, “UPnP Device Architecture 1.0”, http://www.upnp.org/specs/arch/UPnP-DeviceArchitecture-v1.0.pdf

[12] J. Veizades et al, “RFC 2165: Service Location Protocol”, June 1997. http://www.faqs.org/rfcs/rfc2165.html

[13] Sun Microsystems, “Jini Network Technology”, http://www.sun.com/software/jini/

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

[15] S. Cheshire et al, “RFC 3927 - Dynamic Configuration of IPv4 Link-Local Addresses”, http://www.faqs.org/rfcs/rfc3927.html

[16] L. E. Olsen et al, “A study of computer interfaces for soldier systems”, FFI/RAPPORT 2005/03043.

[17] C. Campo et al, “PDP: A lightweight discovery protocol for local-scope interactions in wireless ad hoc networks”, Elsevier 2006, Computer Networks 50 3264-3283

[18] R. Rasmussen, A. Eggen, D. Hadzic, O.-E. Hedenstad, R. Haakseth, and K. Lund, "Experiment report: "Secure SOA supporting NEC" - NATO CWID 2006," FFI rapport 2006/00325

[19] NATO RTO Technical Report: "Awareness of Emerging Wireless Technologies: Ad-hoc and Personal Area Networks Standards and Emerging Technologies", RTO-TR-IST-035; AC/323(IST-035)TP/32, 2007

[20] Sailhan, F.; Issarny, V., "Scalable Service Discovery for MANET," Pervasive Computing and Communications, 2005. PerCom 2005. Third IEEE International Conference on , vol., no., pp.235-244, 8-12 March 2005

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

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

[23] Srivastava, V.; Motani, M., "Cross-layer design: a survey and the road ahead," Communications Magazine, IEEE , vol.43, no.12, pp. 112-119, Dec. 2005

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

[25] I. Chakeres and C. Perkins. "Dynamic manet on-demand (dymo) routing", INTERNET-DRAFT draft-ietf-manet-dymo-09, May 2007.

[26] T. Clausen, C. Dearlove, and P. Jacquet. "The optimized link state routing protocol version 2", INTERNET-DRAFT draft-ietf-manet-olsrv2-03, February 2007

[27] Obaid, A., Khir, A., and Mili, H. 2007. "A Routing Based Service Discovery Protocol for Ad hoc Networks". In Proceedings of the Third international Conference on Networking and Services, 2007

[28] 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.

[29] Jodra, J.L.; Vara, M.; Cabero, J.M.; Bagazgoitia, J., "Service discovery mechanism over OLSR for mobile ad-hoc networks," Advanced Information Networking and Applications, 2006. AINA 2006. 20th International Conference on , vol.2, no., pp. 6 pp.-, 18-20 April 2006

[30] Li Li; Lamont, L., "A lightweight service discovery mechanism for mobile ad hoc pervasive environment using cross-layer design," Pervasive Computing and Communications Workshops, 2005. PerCom 2005 Workshops. Third IEEE International Conference on , vol., no., pp. 55-59, 8-12 March 2005

[31] Delmastro, F. "From Pastry to CrossROAD: CROSS-Layer Ring Overlay for AD Hoc Networks". In Proceedings of the Third IEEE international Conference on Pervasive Computing and Communications Workshops, 2005.

[32] Stine, J.A., "Cross-Layer Design of MANETs: The Only Option", Military Communications Conference, 2006. MILCOM 2006 , vol., no., pp.1-7, 23-25 Oct. 2006

[33] OASIS, "Reference Model for Service Oriented Architecture", Committee draft 1.0, 2006, http://www.oasis-open.org/committees/download.php/16587/wd-soa-rm-cd1ED.pdf

[34] G. Babakhani et al, “Web Trends and Technologies and NNEC Core Enterprise Services – Version 2.0”, NATO C3 Agency, Technical Note 1143 (draft), December 2006

[35] Web Services Inspection Language http://www.ibm.com/developerworks/library/specification/ws-wsilspec/

[36] Gartner research, “Core Web Service Standard UDDI Evolves With Version 3.0.2”, February 2005, ID Number: G00126170, http://www.gartner.com/resources/126100/126170/core_web_servic.pdf

[37] Min Tian, “QoS integration in Web services with the WS-QoS framework”, Freie Universität Berlin 2005, http://www.diss.fu-berlin.de/2005/326/indexe.html

[38] J. Beatty et al, “Web Services Dynamic Discovery (WS-Discovery)”, April 2005, http://specs.xmlsoap.org/ws/2005/04/discovery/ws-discovery.pdf

[39] Gong, L., “JXTA: A Network Programming Environment,” IEEE Internet Computing, vol. 5, no. 3, May/June 2001.

[40] Traversat, B., Arora, A., Abdelaziz, M., Duigou, M., Haywood, C., Hugly, J., Pouyoul, E., and Yeager, B., “Project JXTA 2.0 Super-Peer Virtual Network,” http://www.jxta.org/project/www/docs/JXTA2.0protocols1.pdf, 2003.

[41] McIlraith, S. A., Son, T. C., and Zeng, H., “Semantic Web Services,” IEEE Intelligent Systems, vol. 16, no. 2, pp. 46-53, March/April 2001.

[42] Kol, N., “Interoperability Metadata Registry and Object Store (IMROS)”, presentation at NC3B Ontology workshop, March 11-13th 2008, The Hague, Netherlands

[43] OASIS. Webområde for ebXML/OASIS. [Online] [Cited: April 10, 2008.] http://www.ebxml.org/geninfo.htm.

[44] OASIS Business Process TC (Dale Moberg and Monica J. Martin). ‘The ebBP’ (ebXML Business Process Specifcation Schema). [Online] April 26, 2006. [Cited: April 10, 2008.] http://www.oasis-open.org/committees/download.php/17857/ebxmlbp-v2.0.3-WhitePaper-wd-r01-en.pdf

[45] ebXML Core Components Team. Core Component Overview (version 1.05). [Online] May 10, 2001. [Cited: April 10, 2007.] http://www.ebxml.org/specs/ccOVER.pdf

[46] OASIS ebXML Collaboration Protocol Profile and Agreement Technical Commitee. Collaboration-Protocol Profile and Agreement Specification Version 2.0. [Online] September 23, 2002. [Cited: April 10, 2008.] http://www.oasis-open.org/committees/download.php/204/ebcpp-2.0.pdf

[47] OASIS. OASIS ebXML Collaboration Protocol Profile and Agreement TC - FAQ. [Online] [Cited: April 10, 2008.] http://www.oasis-open.org/committees/ebxml-cppa/faq.php

[48] OASIS ebXML Messaging Services TC. OASIS ebXML Messaging Services, Version 3.0: Part 1, Core Features. [Online] July 12, 2007. [Cited: April 10, 2008.] http://www.oasis-open.org/committees/download.php/24618/ebms_core-3.0-spec-cs-02.pdf

[49] OASIS ebXML Registry Technical Committee. ebXML Registry Information Model, v3.0.1. [ed.] Kathryn Breininger, Farrukh Najmi and Nikola Stojanovic. s.l. : OASIS, February 22, 2007.

[50] ebXML Registry Services and Protocols, v3.0.1. [ed.] Kathryn Breininger, Farrukh Najmi and Nikolah Stojanovic. s.l. : OASIS, February 22, 2007.

[51] Sun Microsystems. Effective SOA Deployment Using an SOA Registry-Repository. A practical guide. [Online] September 2005. [Cited: April 14, 2008.] http://www.sun.com/products/soa/registry/soa_registry_wp.pdf

[52] freebxml.org. freebXML Registry. [Online] [Cited: April 14, 2008.] http://ebxmlrr.sourceforge.net/

[53] ebXML Products. [Online] [Cited: April 14, 2008.] http://ebxml.xml.org/products

[54] OASIS. ebXML Implementations. [Online] [Cited: April 14, 2008.] http://www.ebxml.org/implementations/

[55] R. Faucher et al, "Guidance on Proxy Servers for the Tactical Edge", MITRE technical report MTR 060175, September 2006.

[56] M. Amoretti et al, "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

[57] D. Marco-Mompel, "SERVICE ORIENTED PEER PROTOTYPE FOR MOBILE USERS", NC3A Technical Note Draft under project SPW001495, November 2007.

[58] "Guidance on the use of metadata element descriptions for use in the NATO Discovery Metadata Specification (NDMS)", AC/322-D(2006)0007, NC3B, 14. February 2006

[59] "OWL-S: Semantic Markup for Web Services.", http://www.w3.org/Submission/OWL-S/ (accessed 2008-05-05)

[60] M. Uschold, "Where are the semantics in the semantic web?", AI Mag. vol. 24 number 3, pages 25-36, 2003, American Association for Artificial Intelligence, Menlo Park, CA, USA, ISSN 0738-4602