FFI-rapport 2009/02289

Information exchange in heterogeneous military networks

Ketil Lund, Trude Hafsøe, Frank T. Johnsen, Espen Skjervold og Anders Eggen
Léon Schenkels, NC3A
James Busch, NC3A

Forsvarets forskningsinstitutt/Norwegian Defence Research Establishment (FFI)

22 December 2009

FFI-rapport 2009/02289

1086

P: ISBN 978-82-464-1792-9
E: ISBN 978-82-464-1793-6

Keywords

Web services

Nettverksbasert Forsvar

Tjenesteorientert arkitektur

Approved by

Anders Eggen	Project Manager
Vidar S. Andersen	Director

English summary

The NATO-equivalent of the Norwegian ”Network Based Defence” is ”Network Enabled Capabilities” (abbreviated NNEC). This concept allows users at all operational levels to seamlessly access the necessary and relevant information when needed. It is not realistic to anticipate that NATO nations will replace their existing systems in favor of new, common systems. Instead, there will be a mixture of heterogeneous systems, where new and old systems need to interoperate. NATO has identified the Service-Oriented Architecture paradigm implemented using Web services as the key enabling technology for NNEC. Here, each system can be exposed as one or more services, which in turn can be accessed through standardized interfaces.

The challenge lies in using Web services in heterogeneous military networks. The technology was developed for use over the Internet where bandwidth is abundant, but in e.g. tactical networks bandwidth can be scarce and there can be frequent disruptions (so-called disadvantaged grids). This report addresses some of the most important Web services components, and we discuss necessary measures for employing Web services in heterogeneous networks. Our experiments show that it is possible to utilize the technology in disadvantaged grids, provided that one can cope with the low bandwidths and also overcome the underlying network instability.

FFI has developed a software prototype called DSProxy, which enables the use of Web services in disadvantaged grids. In this report we present the prototype, and discuss its functionality in context of the joint NC3A and FFI experiments at Combined Endeavor in 2009.

Sammendrag

I Nato har man utviklet konseptet ”Network Enabled Capabilities”, som tilsvarer vårt konsept “Nettverksbasert Forsvar”. Konseptet innebærer at brukere på alle operative nivåer skal ha sømløs tilgang til den informasjonen de trenger, når de trenger den. Samtidig er det urealistisk å forvente at alle Nato-land skal skifte ut eksisterende systemer, og i stedet kun benytte nye, felles systemer. Det vil i stedet alltid være en miks av heterogene systemer, og nye og eldre systemer som skal virke sammen. For likevel å kunne oppnå sømløs informasjonsutveksling i et slikt miljø har Nato valgt å satse på tjenesteorientert arkitektur (SOA – Service Oriented Architecture), realisert ved hjelp av Web services, hvor systemer fremstår som tjenester som aksesseres gjennom grensesnitt. Grensesnittene er beskrevet i standardiserte tjenestebeskrivelser, og ut fra disse beskrivelsene kan man så generere programvare for å kommunisere med tjenestene.

Utfordringen med bruk av Web services er at de er beregnet for bruk i homogene nett med høy båndbredde. En utfordring er å få Web services til å fungere på tvers av heterogene nett, spesielt i taktiske nett, hvor man kan ha lav båndbredde og mange avbrudd (disadvantaged grids) vil Web services ofte fungere dårlig. I denne rapporten ser vi nærmere på en del av de mest aktuelle Web service komponentene, og hva som må gjøres for å få Web services til å fungere i heterogene nett. Våre erfaringer viser at det er fullt mulig å benytte Web services også i disadvantaged grids, selv ved svært lave båndbredder. Dette krever imidlertid at man innfører et mellomvarelag som kan skjule heterogeniteten og ustabiliteten i taktiske nett fra applikasjonene og klientene.

Ved FFI har vi utviklet en løsning kalt DSProxy, som fungerer som en slik mellomvare, og som gjør det mulig å benytte umodifiserte Web services også i disadvantaged grids. I rapporten gir vi en beskrivelse av hvordan denne løsningen fungerer, og vi presenterer eksperimentene vi gjennomførte på Combined Endeavor i 2009, hvor vi demonstrerte bruk av DSProxy i et operativt scenario.

Contents

1 Introduction

In the NATO concept of Network Enabled Capabilities (NEC) there is an ambitious requirement for users at all operational levels to seamlessly exchange information. The first step towards NATO NEC is to integrate legacy strategic and tactical systems into a common network. For such integration the modular concept from Service Oriented Architectures (SOA) is essential. Each legacy system can be viewed as a separate module that needs to be interconnected with others. In order to get the different modules to cooperate one needs a common standardized means of communication between them, and for this, Web services, being the prevalent means for realizing SOA, is a possible technology.

There is also an increasing demand to use Commercial off-the-shelf (COTS) software in military systems, which is a further incentive for focusing on Web services. However, the civil standards being developed for implementation of SOA solutions are usually meant for reliable, static networks with high data rate. In military operational networks, on the other hand, the situation can be very different from this. One of the challenges is therefore to find out where we may use civil standardized solutions, and where we have to develop special military STANAGs.

For Web services, the challenge lies in using these across heterogeneous networks, including tactical communication systems with low available bandwidth and high error rates, so-called disadvantaged grids. In general, data-rate constraints in tactical networks impose great challenges that have to be solved in order to fully deploy a SOA supporting NEC. However, in our view, Web services are well suited for pervasive military use, despite the characteristics of military networks. The mechanisms needed to enable Web services on the tactical level include compression, filtering, and proxy servers to limit bandwidth usage.

Furthermore, on the Internet, Web services use the XML-based SOAP protocol over HTTP and TCP for information exchange, but the properties of HTTP and TCP make these unsuited for use in disadvantaged grids. In order to allow a pervasive use of the SOA concept, information must be able to traverse heterogeneous networks with different characteristics. If we assume the use of Web services as the general information exchange mechanism, this can be achieved by introducing delay-tolerant overlay solutions that enable asynchronous message exchange and store-and-forward capabilities. Such a solution will hide the network heterogeneity from the Web services layer and utilize suitable communication protocols for the different networks.

2 Motivation

Shared situation awareness among military units is essential for NEC operations. This requires increased access to information to ensure that the units that best can utilize the information have access to it. One of the main challenges in NEC is to enable users to exchange information with each other at all operational levels, including users on disadvantaged grids (tactical communications systems with low data rate, high delay, and frequent disruptions). In addition, in the case of international operations, information must be exchanged between nations.

2.1 Characteristics of different operational levels

From a technical point of view, it is sensible to separate between three operational levels for networks, namely strategic, tactical deployed and tactical mobile [13]. Together, these three levels cover all network types that are normally encountered in a military context.

2.1.1 Strategic level

Networks on the strategic level are characterized by high bandwidth and a large number of both services and nodes, and they are normally wired. The configuration is relatively static, meaning that the topology and available services do not change very often. In other words, networks on this level are relatively Internet-like.

The high bandwidth also means that resource-demanding data types like images and video (including streaming) can easily be transported over such networks.

2.1.2 Tactical deployed level

On the tactical deployed level the networks are normally wired within a deployment, and with radio or satellite-based reachback links to strategic networks. The wired parts of such networks can have capacities in the order of megabits or gigabits per second, while the radio-based parts normally have considerably lower bandwidth, typically in the range of tens to hundreds of kilobits.

This means that internally in e.g., a deployed headquarter, there will be sufficient bandwidth for resource consuming services such as video, while the capacity of the external network links will be dependent on the technology used.

2.1.3 Tactical mobile level

Networks on the tactical mobile level are radio-based and typically characterized by low bandwidth, frequent delays and disruptions, and a small and very dynamic set of services. Such networks can also easily get temporarily partitioned, due to lack of radio coverage. In many cases there will be a mix of medium bandwidth, short range radios used internally in, e.g., a squad, and low bandwidth, long range radios used as a reachback to a deployed headquarter.

Due to the need for long range signals and jamming resistance, radio systems such as HF or VHF used for reachback links may have a data transfer rate lower than 1Kb/s in practice. In addition, some radio systems suffer from long turn times for directional changes, plus long setup times for connections.

Consequently, out of the three operational levels described above, the tactical mobile level poses the biggest challenges. In particular over the reachback links, it is relatively limited what services can be offered, and what data types that can be transferred.

A tactical deployed network, typically depending on radio or satellite communication, is more dynamic than strategic networks, and typically represents the most dynamic type of operational networks. 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. This is illustrated in Figure 2.1.

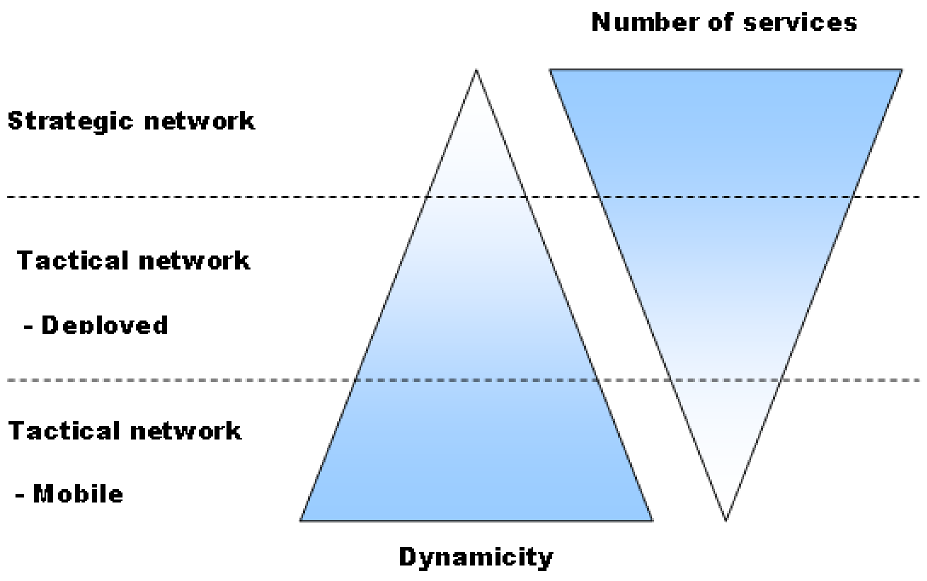


Figure 2.1: Domain complexity (from our report [13])

2.2 Information exchange in military networks

The information exchange between the NATO nations will be provided by a Networking and Information Infrastructure (NII) that, through utilization of new and modern information and communication technology, will provide more flexibility and efficiency. However, what is new and modern is continuously changing and NATO NEC is therefore not a finite state, but rather a continuous process. This means that there will always be a heritage of legacy systems and networks that need to interoperate with the new and modern systems. It is therefore clear that there is a growing need for making existing and new systems work together, in order to meet the ambitions of seamless information exchange in a context of increasing variety and complexity of technological tools and solutions. We currently see interoperability difficulties related to the many heterogeneous communication and information systems being implemented, and this means that we have to make use of technology that makes it possible to integrate heterogeneous systems across heterogeneous networks.

Two important recommendations made in the NATO NEC Feasibility Study (NNEC FS) [4] that should be emphasized in this connection:

- The information infrastructure should be implemented as a SOA.

- IP should be used as a common network protocol in all network types.

These technologies facilitate easier interoperability both across network types and across national systems. Furthermore, the NNEC FS envisions users at all operational levels exchanging information, including users in the field who may only communicate with others over disadvantaged grids. Consequently, NATO focuses on the establishment of a SOA in order to enhance interaction between the allied forces. This is because SOA provides a way of making military resources available as services so they can be discovered and used by other entities that need not be aware of these services in advance.

2.3 Requirements

It is important to remember that the NATO NEC information infrastructure will consist of a federation of systems mainly provided by the member nations. Many nations consider NATO NEC to be a concept to be used for operations in all types and levels of conflicts and in both national and international operations. This puts some requirements on the technologies to be used, and the nations should focus on technology that can be adapted to different types of use (national and international). An additional advantage of such an approach is that cost for procurement, training, and maintenance is reduced.

It will never be realistic to expect that all NATO nations adopt the same systems, and in addition, there will always be legacy systems that must be integrated in order to co-exist with contemporary systems. The NATO nations should therefore instead agree on standardized descriptions of interfaces and data formats, and leave it to each nation to implement the interfaces according to the specifications. If such clients and services are implemented according to the agreed standards and formats, interoperability between nations is ensured. By using Web services for this, COTS software is in many cases a viable solution, contributing to reduced cost and development time.

The most important question is therefore how one can enable pervasive use of Web services, i.e., how to use Web services on all operational levels, including disadvantaged grids and heterogeneous networks. Through our work, we have found that this question can be broken down into three requirements:

- *Reduce the network traffic generated by Web services*. Web services are based on XML, which is a relatively verbose language, producing considerable overhead when used in communication protocols. It is therefore necessary to reduce the amount of data to be transmitted.
- *Remove the dependency on end-to-end connections*. The transport protocol that is normally used in Web services, TCP, is dependent on being able to establish an end-to-end connection between the communicating parties. This makes the communication vulnerable in case of unstable networks. In some cases, it may even be impossible to

establish an end-to-end connection. To enable pervasive use of Web services, it is therefore necessary to remove this dependency on end-to-end connections

- *Hide network heterogeneity*. Web services are originally intended for homogeneous, internet-like networks. In particular, tactical networks are often very different from internet-type networks, displaying long delays and low bandwidth. This is in itself a challenge, and in addition, a connection having to traverse several networks with very different characteristics can be impossible for the standard Web service communication protocols. It is therefore necessary to hide the heterogeneity, and make the network appear to the application layer as one homogeneous network.

In addition, at the same time as these requirements must be met, it is important to remain standards-based as far as possible. Thus, it is a goal that standard, unmodified Web services and – clients should be able to communicate over disadvantaged grids.

3 SOA and Web services

SOA is a concept that enables resources to be provided and consumed as services, allowing for dynamic information sharing between entities (e.g., military units). The main functional components of a SOA are the service provider, service consumer and service registry, as illustrated in Figure 3.1. A service provider may publish the services it is willing to share with others in a service registry. A service consumer may browse the service registry to retrieve the relevant announcements, which describes where and how the services may be invoked. Finally, the consumer invokes the service directly.

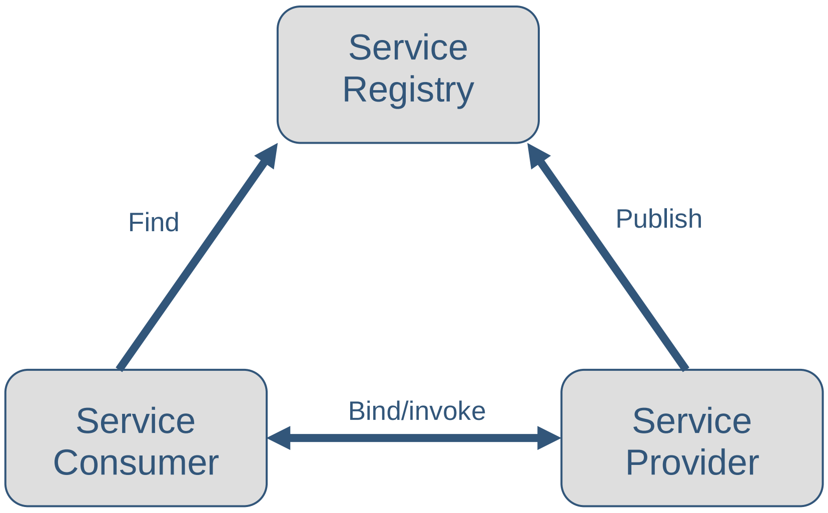


Figure 3.1: The SOA triangle

There exists no agreed-upon, common definition of SOA, and different people can have different understandings of the concept. In our work, we use the “10 Principles of SOA” [15] as a basis. In our view, these principles provide a good foundation for understanding the concept of SOA, and we therefore include them in this report:

1. *Explicit boundaries*: Everything a service needs for providing its functionality should be passed to the service as part of the invocation, and all access to the service should go through the publicly exposed interface.
2. *Shared contract and schema, not class*: A service description should be all both service consumer and service provider needs to consume or provide a service. The service provider should not be dependent on the consumer being able to reuse specific code
3. *Policy-driven*: Policies are needed for describing non-functional capabilities and needs of providers and consumers.
4. *Autonomous*: A service only relates to the outside world through its interface.
5. *Wire formats, not programming language APIs*: All message formats should be described using an open standard or human-readable description, and it should be possible to create correctly formed messages without a specific programmer's library. Furthermore, semantics and syntax for additional communication information (e.g., security headers) should follow a public specification or standard, and at least one of the transport/transfer protocols used should be a standard network protocol.
6. *Document-oriented*: Data to and from the service should be passed as documents, i.e., an explicitly modelled, hierarchical data container. This implies some degree of redundancy, in order to isolate the service interface from the data models of the service consumer and –provider.
7. *Loosely coupled*: Systems can be loosely coupled in many dimensions: Two examples are time (whether both parties have to be online at the same time) and location (whether the address of the participants are looked up, so they can relocate without the need for reconfiguration).
8. *Standards-compliant*: Following standards (as opposed to relying on proprietary APIs) is a key principle in SOA.
9. *Vendor-independent*: The choice of specific products should not have implications on an architectural level.
10. *Metadata-driven*: All metadata must be stored in a way that makes it possible to discover, retrieve and interpret it both at design time and runtime. Examples of metadata include descriptions of service interfaces, endpoints and binding information.

3.1 Web services

In order to achieve shared situation awareness among different organizational units it is essential that the mechanisms for information exchange are based on standards. Requiring all organizational units to use the same systems is not a realistic alternative. By establishing a set of standards for interconnections and information exchange instead, each organizational unit is free to realize their own systems, as long as they adhere to the agreed standards and formats. This is why SOA is a very suitable concept in a military context. Web services are one of the most widely adopted approaches for achieving such standardization, and also the preferred technology for implementing a SOA. Web services also fit very well with the 10 SOA principles listed above.

Web services use the XML-based SOAP[1] protocol for information exchange, and are in widespread use on the Internet today, with civil, commercial products and development tools readily available. To keep acquisition costs low, it makes sense to attempt to utilize this technology for military purposes as well. This is widely recognized, as the NATO NEC FS identifies Web services as a key enabling technology for NEC.

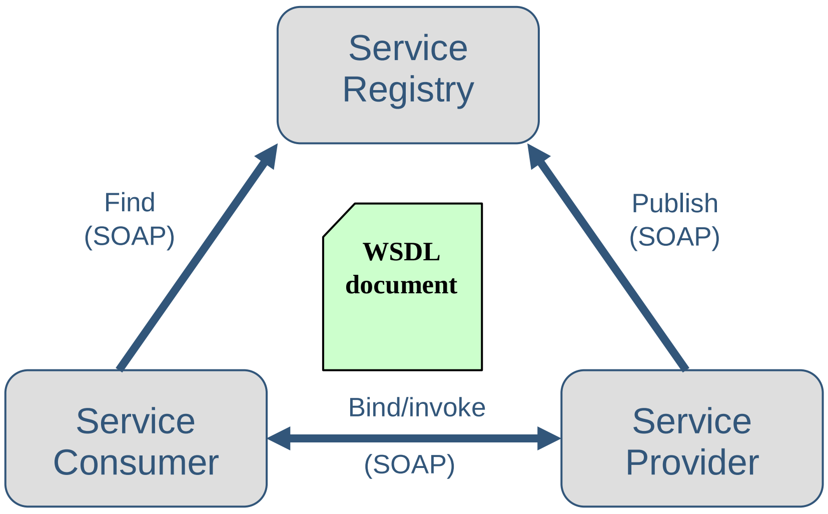


Figure 3.2: SOA realized with Web Services

Web Services are based on a set of XML based standards, with XML itself often considered the base standard for Web services. Most of the other standards use the encoding and format rules defined in this standard, and there are over 80 specifications and standards related to Web services.

Three of the basic specifications are SOAP, Web Services Description Language (WSDL), and Universal Description, Discovery and Integration (UDDI).

SOAP is the XML messaging protocol used for transport of information between the Web services components. An important property of SOAP messages is that they can be delivered over a number of application, transport and network protocols. Today, HTTP is normally used for transporting the SOAP messages between hosts, but in particular in tactical military networks other protocols must also be considered.

WSDL is an XML language for describing Web services. The current version is 2.0, available as a W3C recommendation. Since XML is used, Web service definitions can be mapped to any implementation language, platform, object model, or messaging system. The specification defines a core language which can be used to describe Web services based on an abstract model of what the service offers. It also defines the conformance criteria for documents in this language.

[1] Originally, SOAP was an acronym for “Simple Object Access Protocol”, but in recent revisions this name has been dropped. It is now simply referred to as “SOAP”, with no special implied meaning, though some unofficially refer to it as the “SOA Protocol”, since SOAP is the messaging protocol of Web services.

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

The service registry is often realized using UDDI, which provides mechanisms for publishing and discovery of services. More specifically, UDDI provides access to the WSDL documents describing the protocol bindings and message formats required to interact with the Web services listed in its directory. UDDI can be used for both design time and run time discovery of 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. 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, such as its WSDL description and bindings. It is important to note that the technical documents referenced by a tModel are not stored in UDDI registries themselves. Instead, the tModels contain addresses to where the actual information can be found, which means that you need a separate repository to store the actual data in.

It should be noted that a registry is not strictly necessary in Web services, as there exist alternative forms of service discovery. In particular for disadvantaged grids, a registry based solution is usually not desirable, and sometimes not even possible. For more information on the service discovery part of Web services, we refer to [13].

3.2 Publish/subscribe

Publish/subscribe is a well known communication pattern for event-driven, asynchronous communication. The pattern is particularly well suited in situations where information is produced at irregular intervals. Simply speaking, publish/subscribe means that you will only receive the information that you have subscribed to. This concept utilizes a combination of push and pull. As opposed to a general ”push” mechanism there are benefits in that you may select (subscribe) to the information sent to you, and when using pure “pull” principles you are not able to notify listeners when events occur.

A typical example of this is a sensor network, where information is produced each time a sensor makes an observation. Using a traditional pull-based pattern, the consumers of this information would have to poll the sensor at regular intervals, to check if there is new information available. By using the publish/subscribe pattern, the consumers subscribe to information from the sensors instead, and receive a notification each time new information is available. This means that the consumers receive the information the moment it is available (instead of having to wait for the next polling), and the network traffic is reduced (since polling is not necessary). Thus, the asynchronous nature of the publish/subscribe paradigm makes it a very important mode of communication in NEC.

Regular Web services are based on the traditional pull ("Request/Response") pattern, so the client sends a request to a service, and the service returns a response to the client. As described above, this is not always the best choice for disadvantaged grids. Especially for time-critical information, clients have to poll Web services frequently, which may lead to wasted polls and added network traffic. In such situations, it makes much more sense to have the information producers push the information to the consumers as soon as it becomes available.

In this chapter, we therefore discuss the Publish/subscribe paradigm in a military context. The importance of publish/subscribe for military use can be seen from the fact that the NATO NEC FS puts considerable emphasis on this paradigm, and defines it as a required capability. Also the Core Enterprise Services Working Group (CESWG) focus on this paradigm, and has proposed publish/subscribe as a core service [12] . This working group is now bringing the framework forward by defining service descriptions, which also will include the actual mandatory standards. It will therefore also be of importance what the CESWG decides regarding the two competing specifications for publish / subscribe messaging.

3.2.1 Publish/subscribe patterns

The subscription mechanism allows the consumer to specify what kind of information that is of interest. This means that the consumer does not need to filter the incoming information – the publisher will only send information that the consumer has expressed interest in. In general, there are two types of filtering used in publish/subscribe; *topic-based* and *content-based*:

- *Topic-based filtering* implies that notifications are associated with *topics*, and subscribers only receive notifications on topics that they have subscribed to. The publisher is responsible for defining the different topics that can be subscribed to, and for classifying the notifications according to these topics.

- *Content-based filtering* implies that the subscriber defines a filter that the publisher compares each notification against. Only those notifications that match the filter are passed on to the subscriber. Thus, in this case it is the subscriber that is responsible for classifying the notifications. However, the definition of the service (the WSDL for Web services) will typically define a "dialect" that defines what can be filtered on.

A combination of the two filtering types is also possible, so the subscriber can do content-based filtering on notifications within a topic.

In addition to the filtering types, the topology of the publish/subscribe mechanism is important. In Web services there exist two different publish/subscribe topologies, *direct* and *brokered*:

- *Direct topology* means that the subscribers connect directly to the servers, and that the publishers send individual notifications to each subscriber. In practice, this is nothing more than reversed Web services, where the notification producer calls a Web service offered by the client. This is a relatively simple topology that does not require any specialized infrastructure. On the other hand, the direct connection means that the dissemination of notifications is not very efficient, especially if there are many subscribers (although this can in some cases be remedied using multicast or broadcast). In addition, the direct topology requires both the subscriber and the publisher to implement publish/subscribe functionality. This is typically done by using some framework, and such frameworks can have a relatively large footprint.

- *Broker topology*: Brokering represents a decoupling of notification producers and – consumers. By operating as intermediaries, notification brokers can improve scalability. In addition, consumers and producers no longer need to know about each other; it suffices that they know the broker. The broker itself acts as both a producer and a consumer, and can relieve the producers of notification management (handling subscriptions, filtering, etc.).

In a military context a brokering topology represents several advantages. On platforms with few resources, being relieved of subscription management and notification dissemination implies an increased range of application for notification services. In addition, the notification brokers have the potential to distribute the notification more efficiently than the point-to-point transmissions of basic notification, which is beneficial in disadvantaged grids. For instance, a sensor service connected to a low bandwidth radio network, and distributing notifications using the basic Publish/subscribe mechanisms in Web services would have to set up a separate connection to each subscriber. Using brokered notification instead, it might suffice with one connection out of the radio network, and then a broker can distribute the notifications over a high-speed fixed network. However, this would require that many of the subscribers had subscribed to the same topic.

3.2.2 Notifications and proxy servers

When a client subscribes to a service at a certain server, this server will send new information to the client as soon as the data becomes available. Basically, request/response is traditional “pull” communication, whereas notifications use a combination of “push” and “pull” communication. We will now describe how proxies can be employed to improve both “pull” and “push” communication in Web services. This is an illustration of how the broker topology can be used in a military context, and where proxies play the role of brokers[2].

A notification service will, in its basic form, lead to the transmission of one notification to each subscriber. This means that many copies of the same message will have to be sent over the same physical network connection. In a disadvantaged grid this should be avoided if possible to save bandwidth, and multicast should be employed over shared channels which support it. Proxies can be used to introduce multicast communication even if the notification service itself does not support it.

The following example illustrates how a proxy can be used to increase the efficiency of publish/subscribe communication by reducing network load. A notification service has a publisher which handles subscriptions and the dissemination of new information to its subscribers. When a client wishes to subscribe to information from such a service it needs to register its interest with the publisher. This is done by sending a "subscribe" message to the publisher, which then confirms that the message is received and that a subscription has been established. The illustrations below show how a proxy can be employed to optimize various aspects of the notification service.

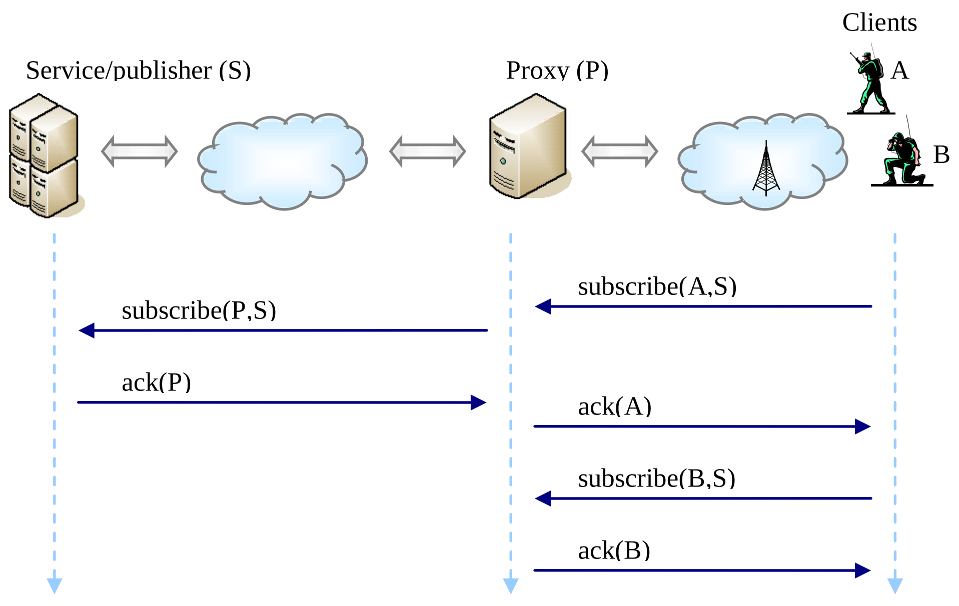


Figure 3.3: Subscription by proxy

When a subscription has been established further communication will be made asynchronously; the publisher publishes the content by sending a "notification" message to each and every subscriber, as illustrated in Figure 3.4: S has new information for its subscribers (which in this case is only P), and sends a "notification" message. P has established this subscription on behalf of clients A and B, and will forward this message to both those clients. An important part of this is the concept of "late copying", which means that the notification messages are copied as late

[2] A proxy can also have other functions, such as transcoding or intermediate storage (store-and-forward)

(i.e., close to the clients) as possible, in order to minimize the amount of data sent over the network.

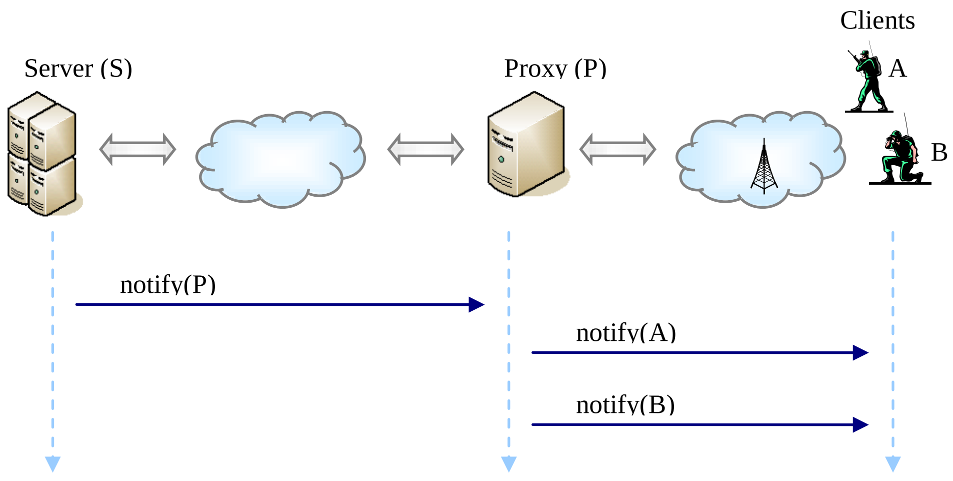


Figure 3.4: Notification via proxy, unicast

Using a proxy has both advantages and disadvantages: An important advantage is that bandwidth is saved between server and proxy. Furthermore, the proxy can adapt the mode of transmission to better utilize the transmission medium between itself and the clients by using multicast, for example. This can lead to a substantial reduction in bandwidth needs, since there will be only one message traversing the network per “notification” sent – no matter how many subscribers there are. This gives the system greater scalability.

Further bandwidth savings can be achieved using filtering, which will be discussed below. Introducing a proxy also has some disadvantages. The proxy needs to keep state about subscriptions, and becomes a possible point of failure in the network. If a proxy crashes and loses its information, then it will not know what to do with incoming notifications, i.e., where to forward them. It is necessary to investigate various solutions for subscription renewal to overcome this problem.

3.3 Web services and Publish/subscribe

In Web services, there exist two competing mechanisms for realizing Publish/subscribe, namely WS-Notification [1] and WS-Eventing [2] . Neither of these are currently in widespread use, but they provide functionality that is required if widespread use of Web services is to be achieved in military networks.

In this section, we present both mechanisms, and compare and evaluate them, in order to provide a recommendation for WS Publish/subscribe in military contexts.

3.3.1 WS-Notification

WS-Notification (WSN) [1] is a Publish/Subscribe notification framework for Web services. There are three parts in the specification: WS-BaseNotification, WS-BrokeredNotification and WS-Topics.

- **WS-BaseNotification** [4] is based on the direct topology, and defines standard message exchanges that allow one service to subscribe and unsubscribe to another, and to receive notification messages from that service.

- **WS-BrokeredNotification** [6] is based on the broker topology, and defines the interface for notification intermediaries. A Notification Broker is an intermediary that decouples the publishers of notification messages from the consumers of those messages; among other things, this allows publication of messages from entities that are not themselves Web service providers.

- **WS-Topics** [7] is used for topic-based filtering, and defines an XML model to organize and categorize classes of events into "Topics," enabling users of WS-BaseNotification or WS-BrokeredNotification to specify the types of events in which they are interested.

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

WSN defines a set of terms, the most important of which we list below:

- **Situation:** A situation is an occurrence (something has happened) that is noted by one party and is of interest to other parties.

- **Notification:** WSN uses this term to refer to the one-way message that conveys information about a situation to other services. The association between a situation and the type of corresponding notification message is not necessarily one-to-one. It is possible that an application might associate several different notification message types with a given situation. This could be the case if there are multiple receivers and the aspects of a situation that are of interest to the receiver vary from receiver to receiver.

- **Publisher:** A publisher is an entity that creates notification message instances (commonly known as events) based on a situation.

- **Notification Producer:** A service that is responsible for sending notifications to the appropriate consumers. If the notification producer does not act as publisher, it is referred to as a notification broker. The notification producer is responsible for maintaining a list of interested consumers and arranging for notification messages to be sent to those receivers.

- **Notification Consumer:** The counterpart of a notification producer is an entity that receives the notifications distributed by notification producers. The most common kind of consumer is a push consumer, which is able to receive notifications sent directly from the notification producer. WSN also supports pull consumers, which interact with the notification producer (or some intermediary) when they wish to receive information. The pull communication is performed with pre-defined pull-points at the service. Pull-points can for example be used by clients that join a network to synchronously “catch up” by fetching an aggregated report of previous events.

- **Subscription:** This is an entity that represents the relationship between a notification consumer and a notification producer. It records the fact that the notification consumer is interested in some or all of the notifications that the notification producer can provide. In loosely coupled environments such as SOAs, it is often desirable to apply a finite lifetime to a subscription so as to avoid situations where consumers disappear or lose interest in a subscription without cancelling it. The subscription lifetime is a tradeoff between refresh rate and amount of “dead” subscriptions. Short-lasting subscriptions means that the clients frequently have to renew their subscriptions, but avoid the problem of “dead” subscriptions running for a long time, and vice versa.

- **Subscriber:** Although subscriptions may be defined statically as part of a system design, event-driven architectures typically involve dynamic subscriptions. WSN uses the term subscriber to refer to an entity that requests creation of a subscription. Note that a subscriber may play the roles of both consumer and subscriber. WSN separates the two roles to allow third-party subscriptions.

- **Subscription Manager:** The subscription manager is a service that manages requests to query, delete, or renew subscriptions. Each subscription manager is subordinate to the notification producer that owns the subscriptions in question. It is possible for a single service to take both the subscription manager and notification producer roles.

3.3.2 WS-BaseNotification

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

WS-BaseNotification does not specify the details of how notification message instances are created and does not define any interface between a publisher and the notification producer. In a tactical system one can use this functionality to integrate legacy systems into NEC. The legacy system can be considered a standalone publisher, and be Web services enabled by allowing it to publish through a WSN compliant notification producer.

The simplest form of a subscribe request message just contains an endpoint reference for a notification consumer. This form of request instructs the notification producer to send all notifications it produces to the notification consumer. In addition, the subscribe request message can optionally contain one or more filter expressions. Such filter expressions indicate the kind of notifications that the consumer is interested in, thereby restricting the kinds of notification that are to be sent for this subscription.

WS-BaseNotification defines the following three kinds of filter expressions:

- **Topic filters** provide a convenient way of categorizing notifications. A topic is a concept used to categorize kinds of notification and their associated notification message types. A topic filter excludes all notifications which do not correspond to the specified topic or topics. Topics are standardized in WS-Topics, which defines topic trees and the use of namespaces.
- **Message filters** are Boolean expressions evaluated over the content of the notification message (content-based filtering). For example, that one only is interested in notifications covering a limited geographical area.

- **Producer state filters** are based on some state of the notification producer itself. In order to use this kind of filter expression, the subscriber needs to know something about the properties of the notification producer.

3.3.3 WS-BrokeredNotification

The specification defines the concept of a notification broker as an intermediary Web service that decouples publishers and notification producers. Thus, a notification broker is itself a Web service, a notification producer and a notification consumer. It can be hosted remotely from both the publisher and the notification consumers, if required.

An advantage of WS-BrokeredNotification is that the publisher does not need to implement notification producer functionality. Since the brokers handle all publish/subscribe functionality, the publisher actually does not need to be a Web service at all.

Furthermore, since the brokers implement both producer and consumer roles, notification dissemination can be done more efficiently; the number of interconnections is reduced, and the brokers administer subscriptions on behalf of the publishers.

3.3.4 WS-Eventing

The WS-Eventing (WSE) specification defines a baseline set of operations that allow Web services to provide asynchronous notifications to interested parties. The specification provides similar functionality to that of WS-BaseNotification, but they are not compatible with each other. The WSE specification also uses approximately the same set of terms as WSN. The most important differences are that WSE does not make a distinction between publisher and notification producer, using the term event source for the combined entity; and that the notification consumer is called event sink in WSE.

On the other hand, the management of subscriptions can be delegated to a separate *subscription manager*, which is a Web service that handles creation, renewal and deletion of subscribtion on behalf of the event source. Subscriptions normally have an associated expiration time, and if the subscription is not renewed within the expiration time, it is deleted.

Finally, WSE has a boolean filtering mechanism, which corresponds to message filtering in WS-BaseNotification (although a much simpler mechanism). A WSE filter is a Boolean expression, and if filters are used, a notification is only sent to a consumer if the expression evaluates to true. This solution means that the declaration of event filters is tightly coupled with content format. This in contrast to the more loosely coupled subscription method based on topics, used in WSN.

3.3.5 Implementation and Usage of WSN and WSE

Over the last years, it seems to have been relatively little activity around the two publish/subscribe proposals. However, WSE has recently become part of the W3C Web services Resource Access Working Group (WSRA), launched in November 2008, and which has the following scope [8] :

"*The Web Services Resource Access Working Group is chartered to standardize a general mechanism for accessing and updating the XML representation of a resource-oriented Web service and metadata of a Web service, as well as a mechanism to subscribe to events from a Web service.*"

Thus, it seems that WSE is currently gaining some momentum. As for WSN, there seems to be few independent implementations, but the standard is being used in products such as IBM WebSphere and Apache ServiceMix. IBM, although being one of the main forces behind WSN, now also supports WSE. Still, it is clear that they still consider WSN as their main publish/subscribe interoperability standard [9] :

"*IBM remains committed to use of the OASIS Standard WS-Notification 1.3 as the interoperability standard for enabling a broad range of applications that include publish/subscribe and event notification in the context of business and complex event processing scenarios. However, we are supportive of the W3C standardization of WS-Eventing because we appreciate that some in the Web services community are using it in certain domains such as* *device management and we are interested in ensuring that such a standard compose effectively with other Web services specifications including WS-Notification.*"

In the NATO Friendly Force Information (NFFI) Interface protocol definition IP3[10] , a publish/subscribe mode was defined. This mode was based on WSN, but not compliant with it. Recently, this specification has been renamed NFFI Service Interoperability Profile 3 (SIP3) [11] In SIP3 the WSN resemblance has been abandoned, and instead, the specification states that the publish/subscribe mode (push mode) shall be *conform to* WS-Eventing. SIP3 then defines a custom *filtering dialect* to be able to express queries on the NFFI data structure.

3.3.6 Recommendation

In this chapter we have looked at the two alternative specifications for Publish/subscribe in Web services, WSN and WSE. The current situation for these specifications is that WSN is an accepted standard, while WSE is only a draft specification. On the other hand, it seems that there are more implementations available for WSE, while WSN is primarily available as an integrated part of larger frameworks. For both specifications, however, it is important to be aware of the licenses under which the implementations are released. For several implementations, GPL-type of licenses are used, and this could represent a problem in some cases, for instance, if software is to be delivered by commercial actors.

A factor that speaks in favour of WSE is that NFFI has abandoned WSN in SIP3 and instead will be conform to WSE. However, the standard push-mode of WSE is not used, and an alternative delivery mode is used instead. It is not clear to us what this means in practice with respect to compatibility with WSE. In addition, translating between the message formats of WSE and WSN is a relatively straightforward task. Thus, NFFI being conformant to WSE does not rule out WSN.

As explained in this report, we consider brokering to be an important part of Publish/subscribe. In order to realize brokering in WSE one would have to design this oneself, and since such functionality is not specified in WSE, this will necessarily become a proprietary solution. WSN, on the other hand, has brokered notification as part of the specification. Combined with the fact that only WSN is an agreed standard, we therefore consider WSN as the best alternative for Publish/subscribe in Web services[3].

4 DSProxy

Because Web services represent such big advantages with respect to interoperability, reuse and reduction of cost and development time, it is our goal to make the use of Web services as pervasive as possible within military networks.

[3] After this report was finished, it also became clear that CESWG has decided to use WSN to realize the publish/subscribe core service

In a previous survey of techniques for adapting Web services to disadvantaged grids, we have identified proxy servers [14] as one of the most important components. Through the use of proxies one can utilize unmodified Web services at the client and server machines, and only the intermediate nodes in the network need use the proxy software. This reduces complexity in the development of applications and servers, and thereby also costs. We have addressed many of the issues presented in [14] by designing and implementing the Delay and disruption tolerant SOAP Proxy (DSProxy).

The DSProxy is a novel prototype system implementing a range of concepts, ideas and mechanisms which aim to tackle the challenges associated with utilizing Web services in tactical environments and disadvantages grids. In practice, the DSProxy is a middleware component enabling delay and disruption tolerant Web services for heterogeneous networks. The Java-based software uses acknowledged standards and is designed to work with existing COTS Web services clients and services. It is designed to be a cross-platform, lightweight and pluggable system with a small installation footprint. All DSProxy components share the same core functionality, but can be configured with different sets of plug-ins.

The DSProxys are compliant to standard Web services (SOAP over HTTP/TCP). This means that Web services and Web services clients communicate as though they were directly connected. The only requirement for using DSProxys is that it must be possible to change the URL in the HTTP header, in order to make the SOAP messages go through the DSProxys rather than directly between client and service.

4.1 Concept

The DSProxy system consists of deployed proxy nodes, which are organized in an overlay network. By routing Web services requests and responses through the DSProxy overlay network, COTS Web service clients and services can be utilized on disadvantaged grids, without modification (see Figure 4.1). This means that network heterogeneity and instability are hidden from the users.

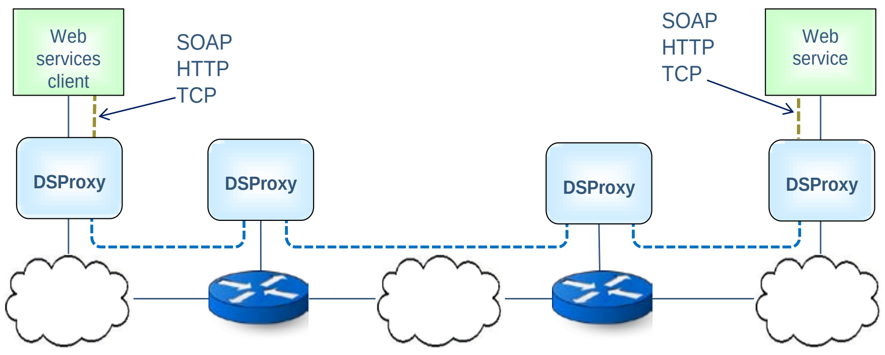


Figure 4.1: The DSProxy concept.

4.2 Functionality

The ability to store-and-forward SOAP messages is the most fundamental functionality found in the DSProxy core. When a DSProxy receives a SOAP message, the message is stored locally before being forwarded to the next node. Thus, if the forwarding fails, the DSProxy can retry transmission until it succeeds. The originating node of the SOAP message, typically a Web service client uses HTTP over TCP when sending the message to the nearest DSProxy.

Upon receiving the request from the client, the DSProxy instance assumes responsibility for relaying the invocation message further on through the chain of DSProxys, in order to invoke the actual end Web service. The DSProxy does this by determining which of its neighboring DSProxys is next in the chain, and then initiates a TCP connection to it, forwarding the invocation request. If the connection for any reason fails, the DSProxy may try an alternative route, or if no such route exists, it will wait for a specified number of seconds (configurable) before retrying. Once the connection is established and the request is successfully forwarded, the DSProxy will wait for the response (the result returned by the actual Web service), and return it to the client. This is done over the TCP connection originally initiated by the client, which is held open for as long as it takes to complete the invocation roundtrip[4].

This mechanism allows ordinary COTS Web service clients to interact with the DSProxy overlay network, hiding the fact that several point-to-point connections may be required to reach the target Web service. Between any two DSProxys, other transport protocols may be used, such as UDP or PMUL (ACP-142v2). When using UDP, an acknowledge mechanism needs to be built on top of the protocol, in order to ensure that all messages are received and to detect failing nodes.

By using this approach Web services invocation becomes possible in and across unreliable networks, effectively hiding disruptions[5] from the calling client. While the Web service or the network it resides in may be temporarily unavailable at the time of the invocation, the DSProxy(s) between the client and the service will store the Web services request, and retry the invocation at regular intervals, returning the Web services response when finally successful.

A network can have any number of DSProxys deployed, but a single DSProxy will most likely be less common than a configuration consisting of several DSProxys working together. As part of the core functionality, the DSProxy has mechanisms for self organizing into an overlay network consisting of any number of DSProxy components, based on a mechanism which relies on UDP multicast. Alternatively, where UDP multicast is not possible, a static overlay network can be configured. All DSProxys regularly advertise their presence in the network, and include a list of services they can invoke, either directly in the same network, or indirectly via other DSProxys. They also provide the number of hops to the advertised services, enabling other DSProxys to

[4] The initial TCP request from the client is immediately acknowledged by the DSProxy to prevent the connection from timing out.
[5] Delays will still be noticeable for the caller, but as explained above, these delays will not cause TCP to time out.

choose the shortest network path. The system is based on an addressing scheme where all services across all connected networks are required to have unique IDs.

The DSProxy overlay network also enables another key property of the system; the ability to traverse multiple and heterogeneous networks. A Web services invocation from a client may have to be relayed through many DSProxy components situated in many different types of networks before the Web service is invoked (see Figure 4.1), and may require different transport mechanisms on the way. Different types of networks have diverse properties and require different types of transport protocols, and the DSProxy system can utilize pluggable transport protocol adapters. Currently, TCP-adapters and UDP-adapters are available, and there are also plans for a PMUL-adapter.

4.3 DSProxy Network Configurations

Figure 4.2 illustrates a network configuration where two DSProxy components enable the traversal of three networks using different transport protocols. The DSProxy components are placed in an *edge proxy configuration*, effectively functioning as SOAP gateways between the networks and enabling Web services requests and responses to cross multiple networks.

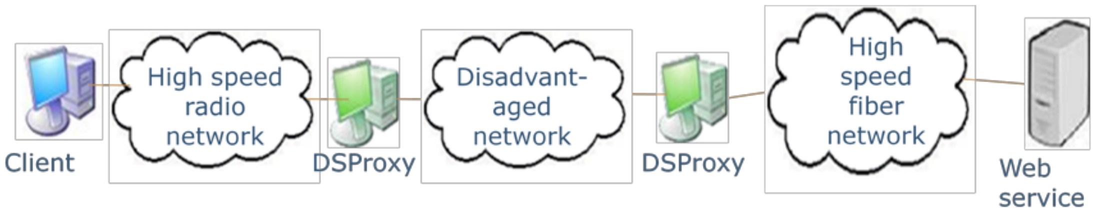


Figure 4.2. Edge proxy configuration.

While the edge proxy configuration bridges networks and provides store-and-forward capability at all network edges, it remains vulnerable to disruptions occurring between the client and the first DSProxy component. An alternative to this configuration is therefore the *locally deployed proxy configuration*, illustrated in Figure 4.3. In this configuration, additional DSProxys are placed on the same physical nodes as the Web services client and the Web service. By placing a DSProxy component locally to the client, we ensure that the Web services client will always reach the first DSProxy component, and store-and-forward capability is provided from the client node and onwards into the networks. This configuration offers advantages for clients operating in unreliable conditions such as mobile ad hoc networks (MANETs), where neighboring nodes and services may appear and disappear frequently. In addition, the traffic between DSProxys can be compressed, as explained in Section 4.5. By placing a DSProxy component locally to the server, the traffic between this DSProxy and the actual service can go uncompressed, without adding traffic load to the network.

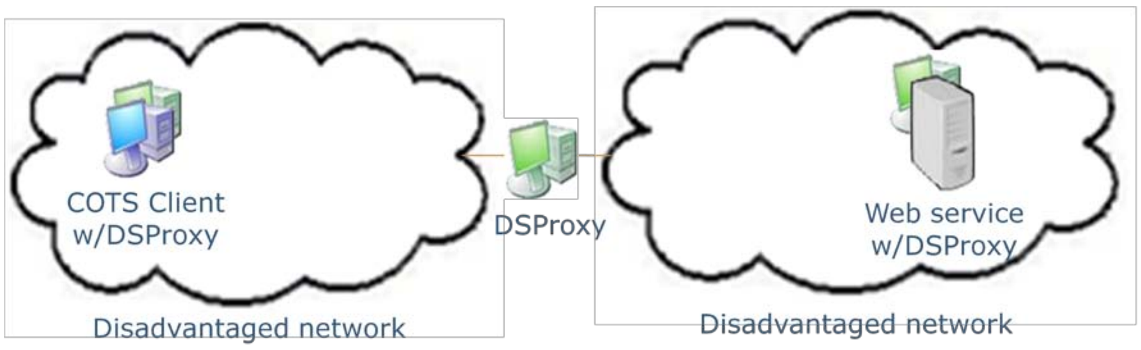


Figure 4.3. Locally deployed proxy configuration.

4.4 COTS Web Services Compatibility

Maintaining compatibility with Commercial off-the-shelf (COTS) Web services clients and Web services has been emphasized when developing the DSProxy system. Most COTS Web services today are based on SOAP, HTTP and TCP. The DSProxy system supports this, and only requires existing applications to be *reconfigured*, not modified. All that is required of the clients is that they replace their original Web services URL in the HTTP header with a URL connecting them to a DSProxy component. In many cases, the service URLs are placed in WSDL files or in application configuration files. In such cases, no changes to the COTS Web services client applications are required. However, sometimes the URL is embedded in the web application, and in such cases, the source code must be touched in order to update the URL (the business code itself is not changed). Below we show examples of URLs used without and with DSProxys:

Original Web services URL: http://133.196.33.75:4756/weatherService.asmx.

URL routing the invocation through the DSProxy system:
http://localhost:/7000/?uniqueServiceName=us.mycompany.departmentx.projecty.weatherService

The example URL above assumes a locally deployed proxy configuration, where the first DSProxy component can be found at the *localhost* hostname. In an edge proxy configuration, the hostname would be substituted with the correct IP address of the nearest edge proxy. By placing the service end point reference in the URL as a parameter (together with any other additional information that should be interpreted by the DSProxy overlay network), the message payload (the SOAP message) needs no modification whatsoever, and is not interpreted by the DSProxy components.

4.5 Pluggable Functionality

In addition to the DSProxy core functionality, the lightweight system architecture offers pluggable and configurable functionality such as data compression and response data caching.

The ready-made, priority plug-in provides role-based traffic prioritization through the DSProxy overlay network. As with the service ID, the role and user IDs are provided through modification of the DSProxy URL, and under high traffic loads, Web services invocations made by users having a role with high priority are prioritized over invocations with lower priority. Information about roles and users is stored in an embedded object database in each DSProxy.

In order to avoid exhausting network radio buffers and losing data, the DSProxy utilizes a Bandwidth Manager plug-in in order to detect possible network congestion through a passive mechanism. This works in the following way: When DSProxy *A* forwards a request to DSProxy *B*, it measures the elapsed time from when it is forwarded to when a response is received. Together with the response, DSProxy *B* also reports the elapsed time from *it* received the request, to when it returned the response to it. DSProxy *A* now subtracts this time from the time it measured itself, and the resulting number is the time the message spent on the network (including the time it spent on the network adapters and radio buffers etc). When these numbers are monitored over time, fluctuations from an established baseline are detected, and the DSProxy will respond by adjusting its transmission rate. Under heavy loads, this leads to its internal message queues accumulating large volumes of requests, which actually facilitates the prioritization of traffic mentioned above. By enqueuing the requests in the application rather than on the network radio, QoS aspects can be considered for a larger set of requests, which yields more effective prioritization.

Compression of Web services invocation messages and response messages are available for traffic sent between two DSProxys. In order to maintain compatibility with COTS systems, the last mile (to the actual Web service and back to the actual client) is transmitted in plain text (i.e., not compressed). The provided compression plug-in is based on the GZIP algorithm [7] using DEFLATE [8] compression, but custom compression plug-ins can also be created. In a locally deployed proxy configuration with DSProxys running on the client node and the Web services node, compression can be employed between all physical nodes, making it particularly suitable for disadvantaged grids and congested networks.

In order to minimize network traffic, the DSProxy system offers response data caching. By altering the DSProxy URL and adding the “allowResponseCachedForSeconds = x” parameter, the Web services consumer can itself specify whether cached data will suffice (and how old), or whether fresh data are required. The DSProxy component does this by caching all response data in a ring buffer, matching cached response data against requested content when new requests arrive. This feature is expected to be particularly useful in environments where many Web services consumers request the same information, and where the information either changes infrequently, or the aspect of freshness is of less importance.

4.6 Addressing and Routing

As previously described, the DSProxy overlay network requires all participating services to be uniquely identified. Rather than introducing a hierarchical, cross-domain addressing scheme for all participating organizations to adopt, resolving a service is done by simple string-matching. A service ID may be e.g. “organisationX.org/locationService” or “uk.businessY. departmentZ.locationservice”. The mapping between service IDs and actual Web services is done in the configuration file belonging to the DSProxy component that is responsible for invoking the actual service. In a locally deployed proxy configuration, this is typically the DSProxy component running on the same physical node as the Web service itself. In the edge proxy configuration, the gateway DSProxy component closest to the Web service typically handles this responsibility. However, multiple DSProxy components may assume this responsibility for redundancy.

DSProxy components that are directly connected to two or more networks (as edge proxies are) will listen to advertised services on one network, and broadcast the received services to the DSProxys on the other networks. As the services are propagated from one DSProxy component to the next, the hop counter for each service is incremented by one. DSProxy components that are only connected to a single network (as "normal" DSProxy components inside one network are), will listen for and accumulate a list of all advertised services, but will only broadcast those services (if any) it is itself responsible for directly invoking.

No single DSProxy component knows the entire route to a Web service. Routing is accomplished by each DSProxy component forwarding the request on to the neighboring DSProxy component that reports being able to reach the Web service in the lowest number of hops. When the request eventually reaches the DSProxy component responsible for invoking the Web service, the Web service will be invoked using a standard TCP connection, and the response will backtrack the request's route.

5 Combined Endeavor

In 2009, NC3A, as the primary technical R&D authority for NATO, and FFI established a collaboration with focus on SOA aspects such as security, service discovery and information dissemination. Two major goals of this collaboration were to perform (federated) cross-domain experimentation of service-oriented implementations and to prepare for a coordinated exercise/demo to take place during the Combined Endeavor exercise in September 2009.

5.1 Background

The Combined Endeavor exercise has been called "the world's largest communications exercise", and is a U.S. European Command (EUCOM) sponsored communications and information systems interoperability event between and among Partnership for Peace and NATO nations, focusing on the deliberate planning process, communications information systems interoperability, and development of a road map for future interoperability improvement among participants. In 2009, 40 nations and 1200 participants, spanning EUCOM and Central Command (CENTCOM) and three continents, conducted approximately 1000 communication and information systems interoperability tests during the exercise's two week period.

Combined Endeavor 2009 objectives are as follows: Evaluation of systems interoperability by the provided framework within a simulated operational setting that addresses operational requirements.

Within the scenario command structure, the scenario should promote the maximum number of different types of information exchanges among systems. The operational Commanders should have the possibility of exchanging information with all available services (e.g. MIP, mail, voice) and observe how it could speed up their Command and Control procedures.

Combined Endeavor 2009 involved three separate physical locations: Bosnia-Herzegovina, Denmark and the Netherlands joined over a common network.

5.2 SOA Interoperability Experiment

Our SOA interoperability experiment focused on federation in diverse SOA infrastructures, particularly at the Core Enterprise Service (CES) level, in a realistic, multi-national setting, with a particular emphasis on deployed and radio (possibly constrained) networks and limited-functionality client devices.

One of the key principles of the NNEC vision is that of “federation”: the idea that NATO and National solutions will be able to be linked together to provide a comprehensive, multi-domain capability. In theory, CES services should be possible to federate as well, to provide (for example) an end-to-end service discovery [13] , service security, transformation, and/or collaboration capability across the whole SOA-enabled coalition.

Thus, Combined Endeavor 2009 provided an opportunity to focus on a tactical environment, whereas SOA research and development to date has focused primarily on the static domain. This has left the usability of these technologies in a deployed setting relatively unexplored.

5.3 Scenario description

Testing and demonstrating of new technologies within realistic scenarios, such as the Combined Endeavor 2009 demonstrations, is crucial for architects and developers of new NNEC/SOA solutions. Such activities are necessary to better understand the needs of end users, demonstrate functionality to these users and decision makers, and determine the added value of research and experimentation as well as new products and technologies.

5.3.1 Scenario overview

The Combined Endeavor exercises were executed according to a Counter-Terrorist Operation (CTO) scenario and various vignettes described in this section. The CTO takes place in the western “DELPHINIAN” provinces as part of a larger Stabilization Operation by a NATO Combined Joint Task Force. The NATO Response Force (NRF) under the command of a Deployed Joint Task Force (DJTF) headquarters (HQ) has deployed as the lead force to support the CT portion of the mission.

Phase 1 – Base Protection / Situation Awareness
The purpose of this phase is to receive and analyze intelligence information provided to the DJTF strategic command by the deployed forces. Specifically, the DJTF would receive and analyze

intelligence, which then is shared between DJTF strategic command and the deployed forces. Furthermore, the DJTF HQ receives status reports (e.g., location, photos) from deployed forces, and it is also able to communicate with the deployed forces via text chat.

Phase 2 - Neutralization of Target
The NATO objective in this phase is neutralization of leadership personnel either at their command facilities, accommodation or in transit. In this case, neutralization of suspicious individuals occurs at their present location.

First, the DJTF HQ viewer (see Phase 1) is used to mark certain items as “red”, while deployed forces are already marked as “blue” (Phase 1). The deployed forces are then tasked to eliminate ambushers, and chat/voice used to confirm commander’s intentions (HQ – deployed). Next, the deployed forces move to location and engage ambushers, while the DJTF HQ “watches” the action on its viewer. Finally, chat/voice is used to confirm success.

5.3.2 Scenario Actors

The exercise scenario featured four classes of actors, each using one of the four sections of the network (see Figure 5.1):

- “NATO DJTF HQ”: Located in the NATO tent on the Combined Endeavor compound, containing a deployed, but static network, viewer(s) on laptops and service(s) on servers.

- “Norwegian Static Command”: Located in the Norwegian tent on the Combined Endeavor compound. Similar setup as the NATO HQ, with a deployed, static network, viewers and services on laptops in the tent

- “NATO Liaison Officer”: This is a roaming actor (on foot, near the Combined Endeavor compound), which were using a PDA connected to the NATO Mobile Ad Hoc Network (MANET). He also had a viewer, running on his PDA.

- “Norwegian Deployed Forces”: Consist of two vehicles, each equipped with viewer and services on laptops, and connected to the Norwegian MANET

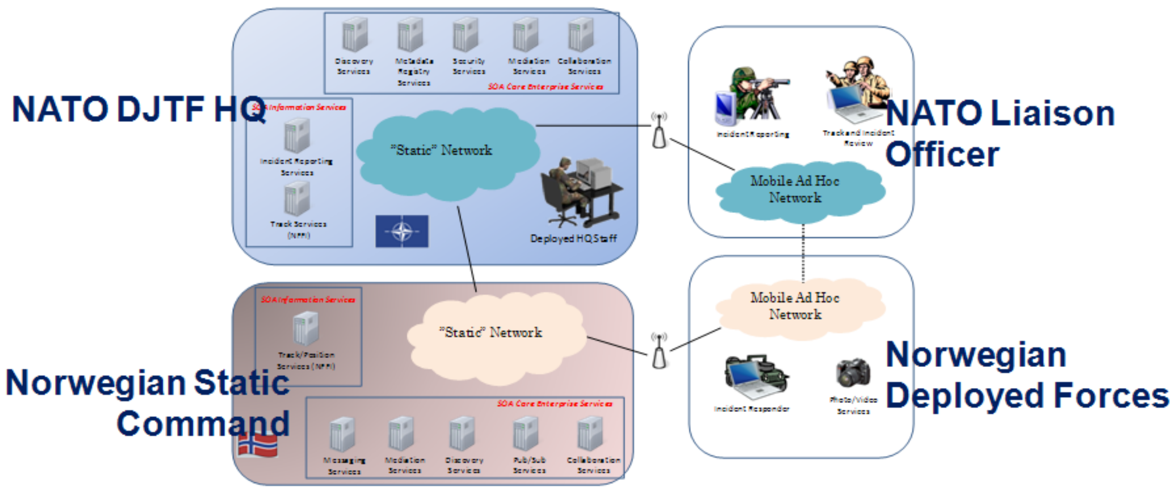


Figure 5.1: Scenario actors

5.3.3 Vignette steps

The scenario above was enacted via a vignette intended to leverage the SOA services in a realistic manner. This section introduces the steps of the vignette.

Starting positions

The "NATO DJTF HQ" is located in the NC3A Tent on the Combined Endeavor compound at 53° 21' 54.08 N, 6° 15'53.53 E, while the "Norwegian Deployed Forces" are positioned on patrol, unit 2 near camp: 53° 21' 53.73 N, 6° 15' 32.15 E, and unit 3, several km to the east: 53° 20' 38.54 N, 6° 17' 55.16 E. Finally, the "NATO Liaison Officer" is located about 1.5km SE of the camp, conveniently in the vicinity of the incident about to occur. The positions are shown in Figure 5.2.

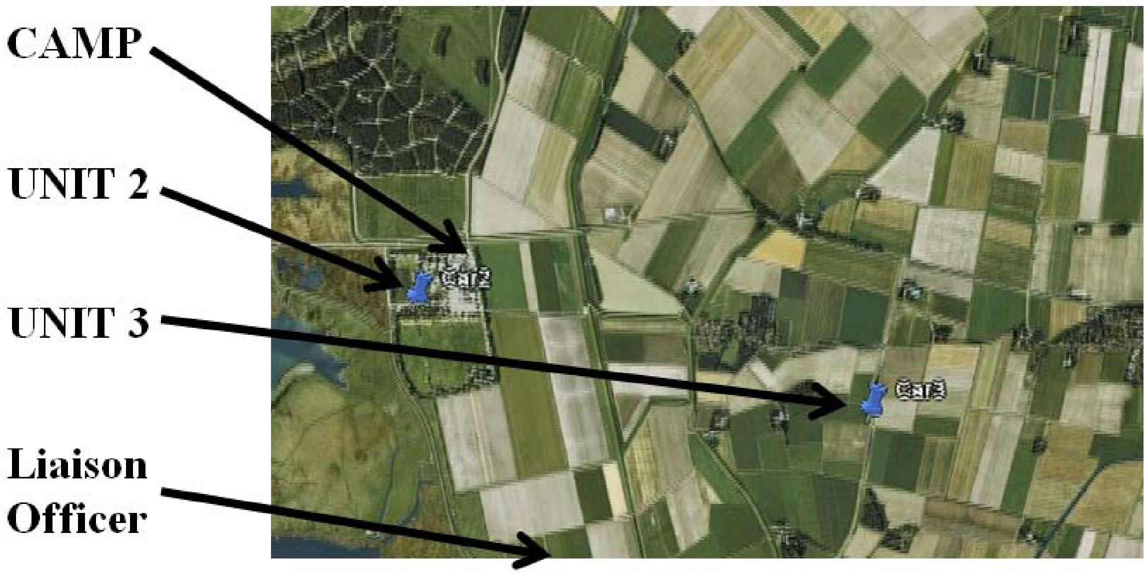


Figure 5.2: Starting position of the actors

The incident

The vignette begins when the "NATO Liaison Officer" spots suspicious behaviour (see Figure 5.3) and records his observation into a NATO system (JOCWatch) via his PDA device:
"Group behaving suspiciously, side of road near Zoutkamp. Appear to be digging hole, possible IED, group is armed, treat with extreme caution."

The incident is located at 53° 20' 51.23 N, 6° 16' 08.96 E (about 1.5 km south of camp).

Figure 5.3: NATO Liaison Officer spots suspicious behaviour

The incident reported by the "NATO Liaison Officer" appears on the screen[6] at "NATO DJTF HQ" (see Figure 5.4) and is reviewed on their viewer.

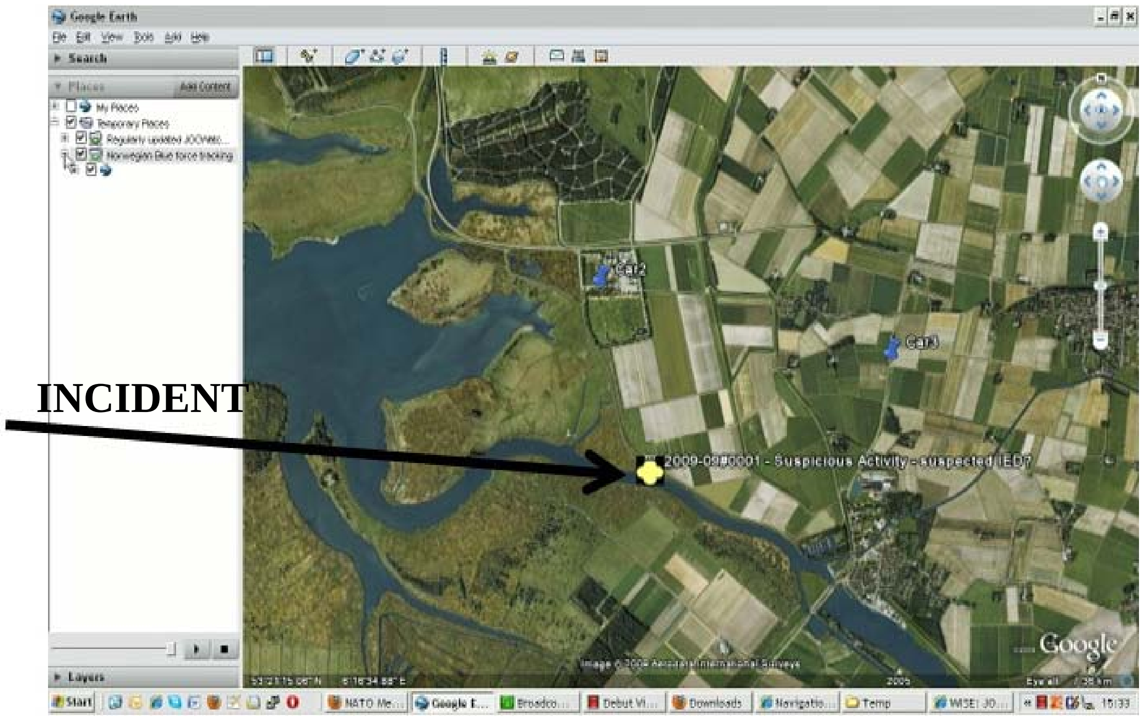
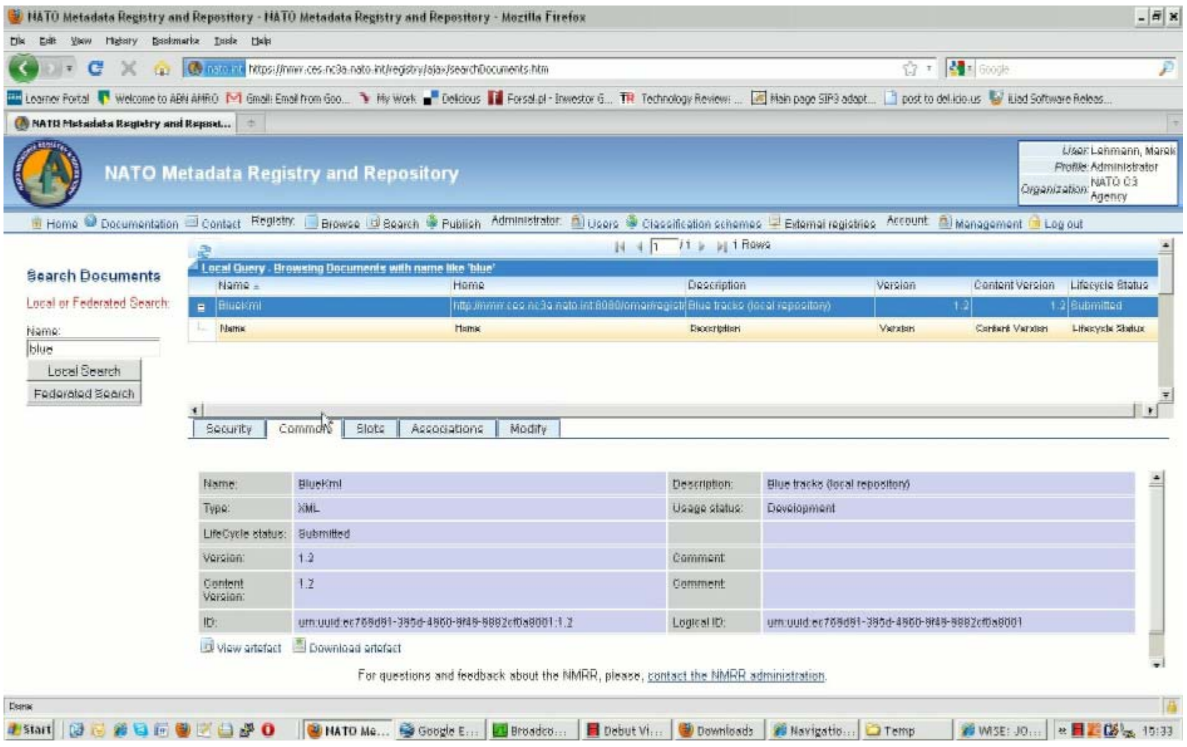

Google Earth
Search
Places
My Places
Temporary Places
Regularly updated JOCWatch
Norwegian Blue force tracking
Layers
Car2
Car1
INCIDENT
2009-09#0001 - Suspicious Activity - suspected IED?
Google

Figure 5.4: Incident appears on screen at NATO HQ

[6] APP-6A symbol is not necessarily correct, but was used here only for demonstration purposes.

The operator at the “NATO DJTF HQ” searches the service registry for ‘sensors’ in the vicinity (see Figure 5.5), and discovers the position service provided by the “Norwegian Static Command” (registered in the Norwegian domain but federated with the NATO registry). “NATO DJTF HQ” then subscribes to the service. The Norwegian blue units coming from the “Norwegian Static Command” service then appear on the “NATO DJTF HQ” viewer.


NATO Metadata Registry and Repository

User: Lehmann, Marek
Profile: Administrator
Organization: NATO C3 Agency

Search Documents
Local or Federated Search:
Name: blue

Local Query . Browsing Documents with name like 'blue'

Name	Home	Description	Version	Content Version	Lifecycle Status
BlueKml	http://nmrr.ces.nc3a.nato.int:8080/omar/registr	Blue tracks (local repository)	1.2	1.2	Submitted

Name:	BlueKml	Description:	Blue tracks (local repository)
Type:	XML	Usage status:	Development
LifeCycle status:	Submitted		
Version:	1.2	Comment:	
Content Version:	1.2	Comment:	
ID:	urn:uuid:ec768d81-385d-4860-9f49-8882cf0a8001:1.2	Logical ID:	urn:uuid:ec768d81-385d-4860-9f49-8882cf0a8001

For questions and feedback about the NMRR, please, contact the NMRR administration.

Figure 5.5: Searching for services in the NATO Metadata Registry

The operator at the “NATO DJTF HQ” again searches the registry to see if there are any other services providing information he can use, and discovers the (federated) photo publishing service that the “Norwegian Deployed Forces” are providing. He subscribes to the service and begins to receive pictures that the units are generating. This is shown in Figure 5.6

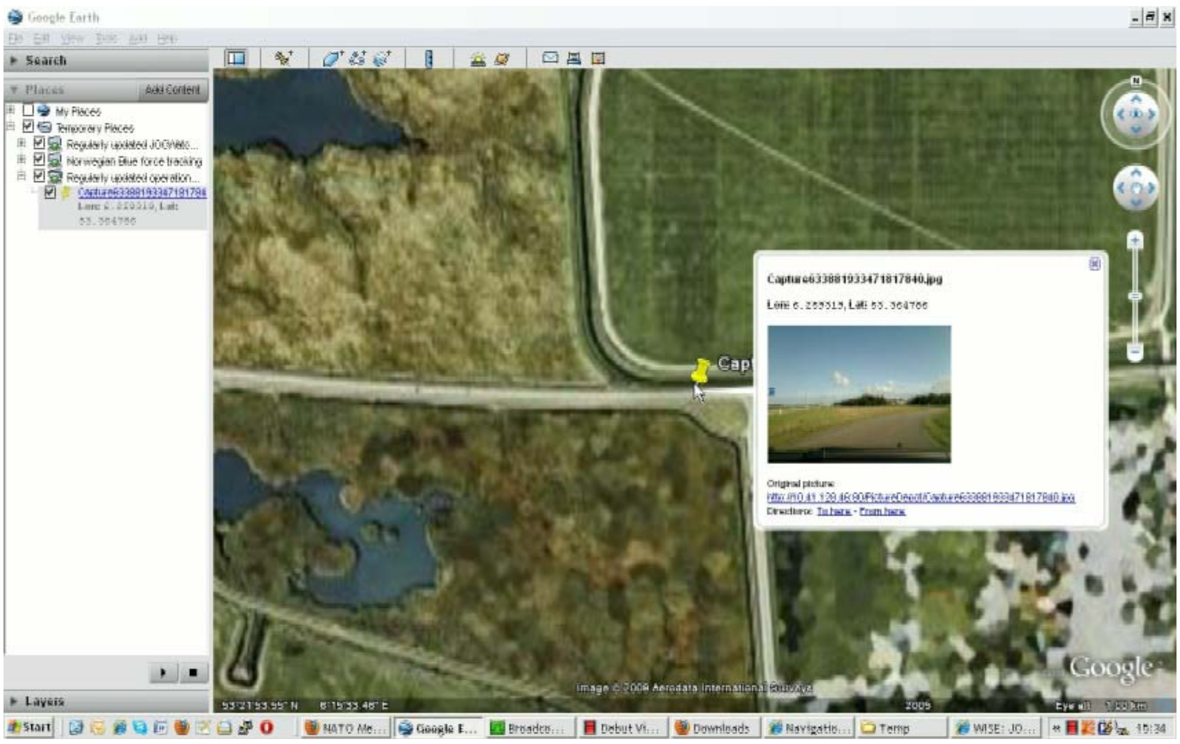
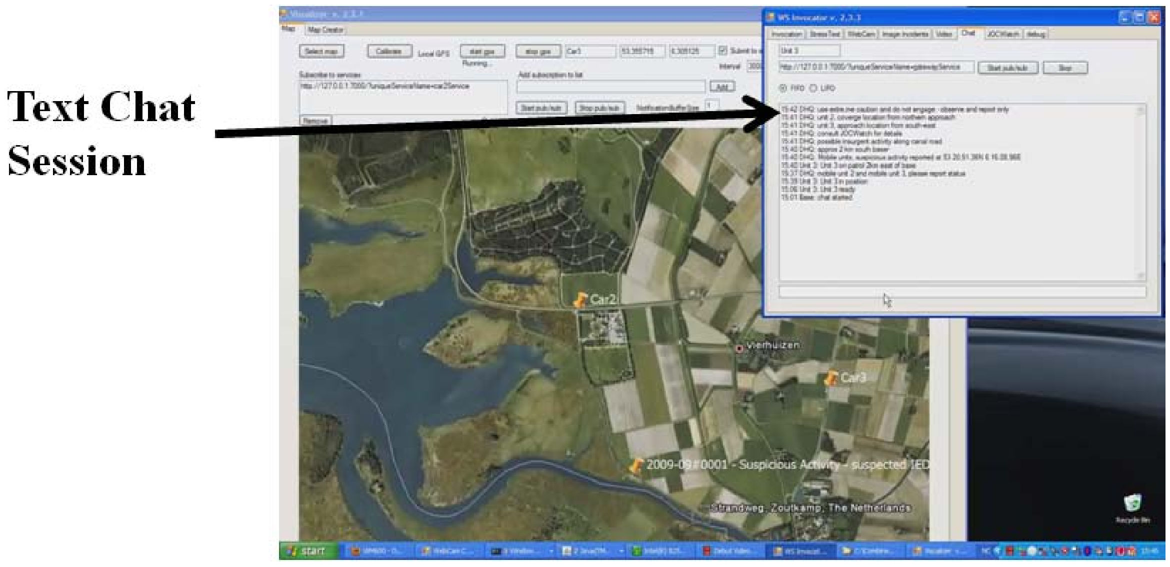

Google Earth
Search
Places
Add Content
My Places
Temporary Places
Regularly updated JOCWatc...
Norwegian Blue force tracking
Regularly updated operation...
Capture633881933471817840.jpg
Lon: 6.255315, Lat: 53.364786
Original picture:
Directions: To here - From here
Layers
Google

Figure 5.6: Picture received from the Norwegian Deployed Force



```
15:42 DHQ: use extreme caution and do not engage - observe and report only
15:41 DHQ: unit 2, coverge location from northern approach
15:41 DHQ: unit 3, approach location from south-west
15:41 DHQ: consult JOCWatch for details
15:41 DHQ: possible insurgent activity along canal road
15:40 DHQ: approx 2 km south base
15:40 DHQ: Mobile units, suspicious activity reported at 53 20.51.36N 6 16.08.96E
15:40 Unit 3: Unit 3 on patrol 2km east of base
15:37 DHQ: mobile unit 2 and mobile unit 3, please report status
15:39 Unit 3: Unit 3 in position
15:06 Unit 3: Unit 3 ready
15:01 Base: chat started
```

Figure 5.7: Chat session between NATO HQ and Norwegian deployed forces

The operator at the "NATO DJTF HQ" begins a text chat session with "Norwegian Deployed Forces" Unit 2 and Unit 3. This is shown in Figure 5.7. Through the chat session, the "Norwegian Deployed Forces" units are tasked to subscribe to the "NATO DJTF HQ" incident reporting service in order to get the details about the suspicious activity. They are then tasked to cautiously converge on the coordinates of the suspected insurgent activity.

In order to get more information, the “Norwegian Deployed Forces” Unit 2 and Unit 3 subscribe to the NATO incident service, and then start moving towards the location. The operator at the “NATO DJTF HQ” can watch the movements of the units on his viewer.

Next, the “Norwegian Deployed Forces” Unit 2 and Unit 3 arrive in position near incident, take pictures of the scene, and publish these via their photo service. In addition, each unit enters a “red force” notation in their position service. The “NATO DJTF HQ” operator sees the photos and “red force” notations appear on his viewer (Figure 5.8)

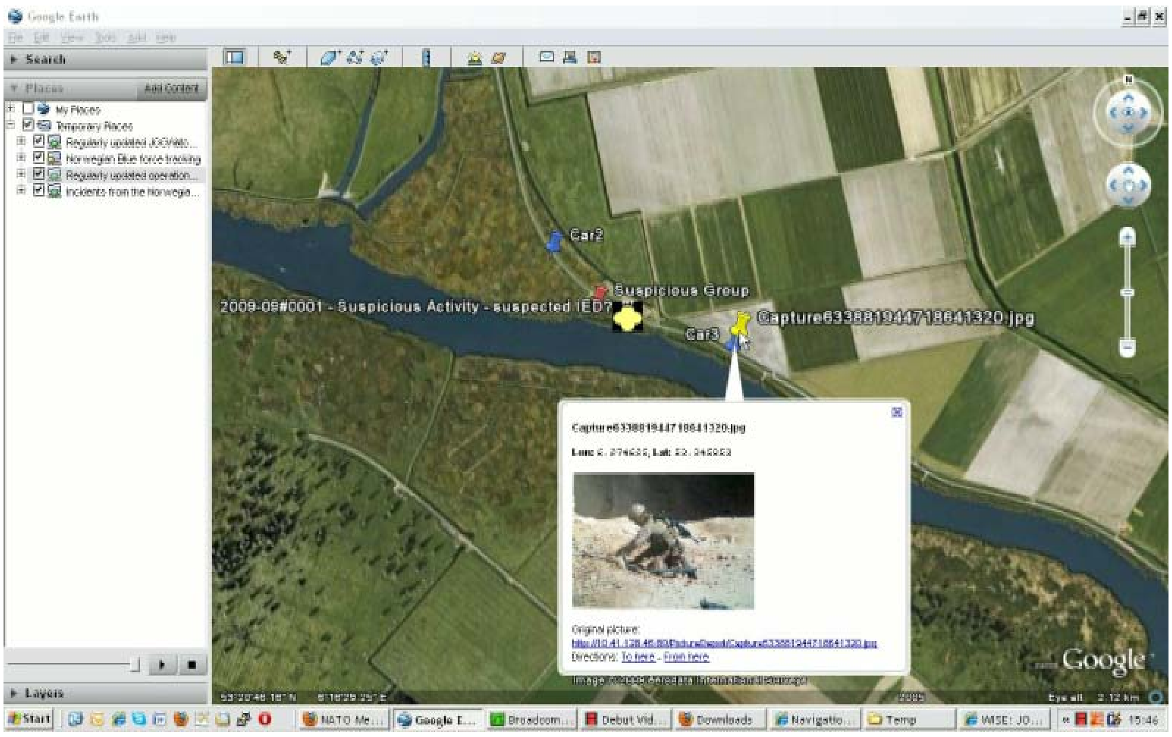

Google Earth
Search
Places
Add Content
My Places
Temporary Places
Regularly updated JOCWatc...
Norwegian Blue force tracking
Regularly updated operation...
Incidents from the Norwegia...
Layers
Car2
Suspicious Group
2009-09#0001 - Suspicious Activity - suspected IED?
Car3
Capture633881944718641320.jpg
Google

Figure 5.8: Pictures from Norwegian deployed forces displayed at NATO HQ

The operator at the “NATO DJTF HQ” now conducts a text chat with the “Norwegian Deployed Forces” units. The mobile units report shots fired, and they are authorized by the “NATO DJTF HQ” to engage.

During the engagement of the enemy, the “Norwegian Deployed Forces” take photos of the scene and publish these. The “NATO DJTF HQ” then automatically receives photos, which are displayed on the viewer (see Figure 5.9).

Finally, the “Norwegian Deployed Forces” Mobile Unit #3 accesses the NATO incident service and makes a new entry:

“4 individuals, suspected insurgents, captured 2 km south Zoutkamp. Confirm presence of IED.” This Incident service entry then appears on the viewer of the operator at the “NATO DJTF HQ”.

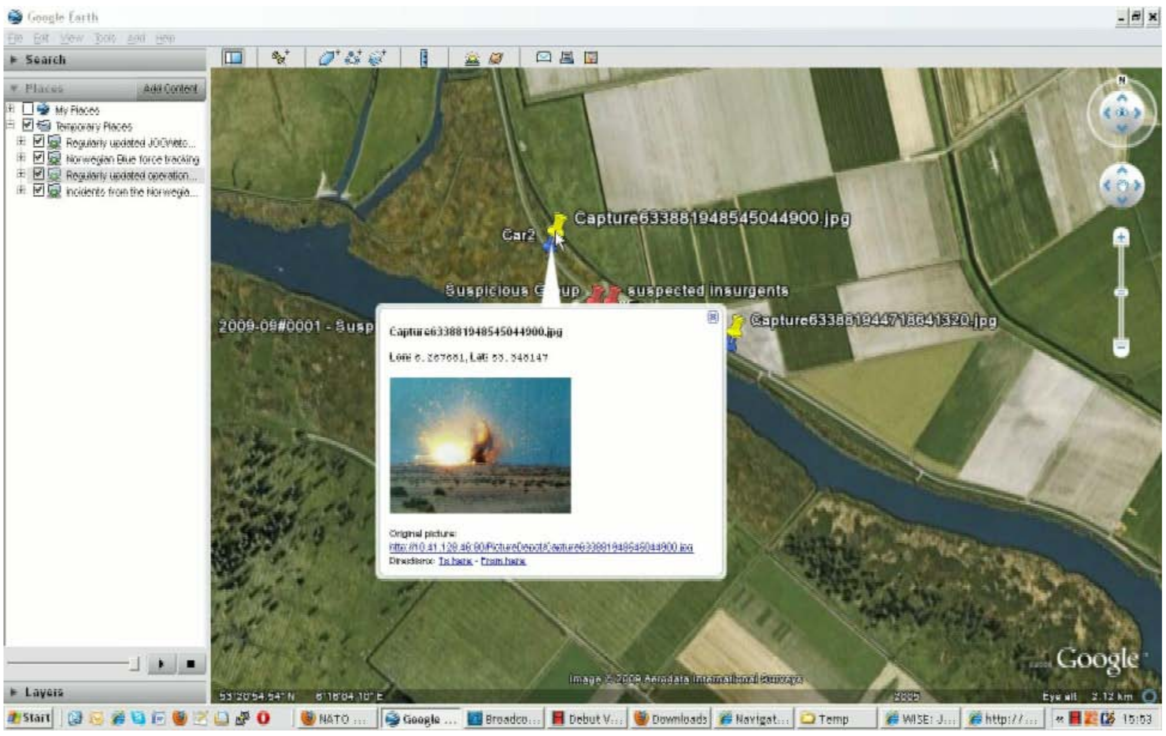


Figure 5.9: More pictures received in NATO HQ

5.4 Format translation

Several services have been designed to transform information from one format (such as that generated by a service) into another (such as that required by a client viewer).

5.4.1 NVG to KML Translation Service

The Google Earth viewer used by NATO staff geo-renders objects using a format called Keyhole Markup Language (KML), while JOCWatch shares information using NVG. As a result, a translation service was created to “wrap” the JOCWatch web service interface and allow it to return data as KML. This enabled the Google Earth viewer to receive and plot data that came from the JOCWatch service.

5.4.2 NFFI to KML Translation Service

Since the Norwegian position services export their data in NATO Friendly Force Information (NFFI) format, a second translation service was created to transform that data into a format that was readable for the NC3A.

5.5 Architecture

For the Combined Endeavor exercise, FFI brought 7 laptops (see Figure 5.10), while the NC3A configuration brought 7-8 (virtual) servers, 3-4 laptops and PDAs. All nodes were at the same security level (UNCLASSIFIED).

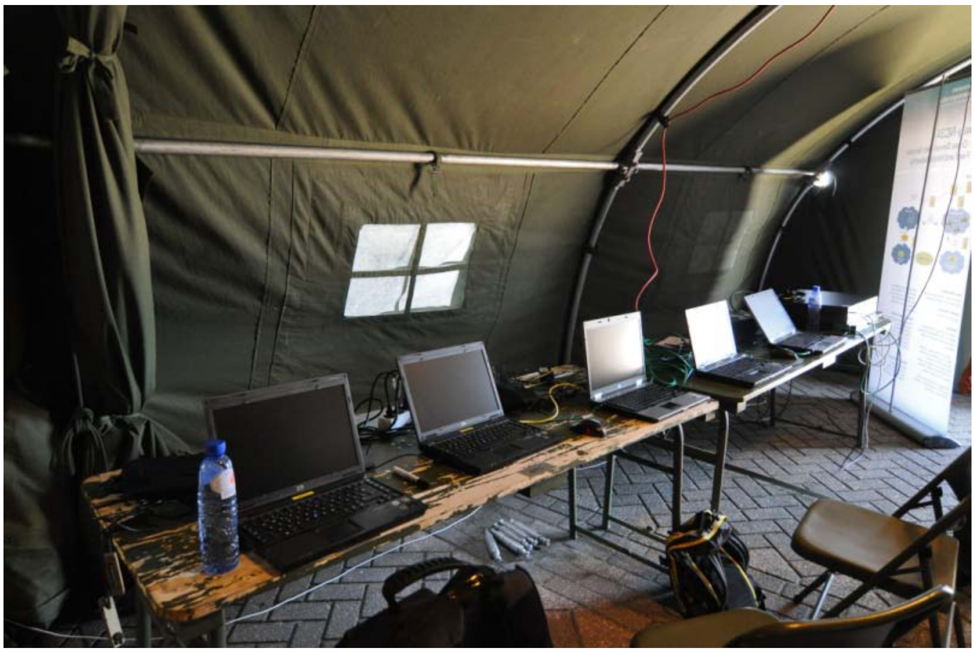


Figure 5.10: From the Norwegian HQ

FFI used two cars, each equipped with a laptop, GPS, web camera, and a Kongsberg WM600 radio (see Figure 5.11). In addition, the Norwegian tent was equipped with a WM600 in order to "ground" the MANET and connect it to the RG C network (via a laptop).

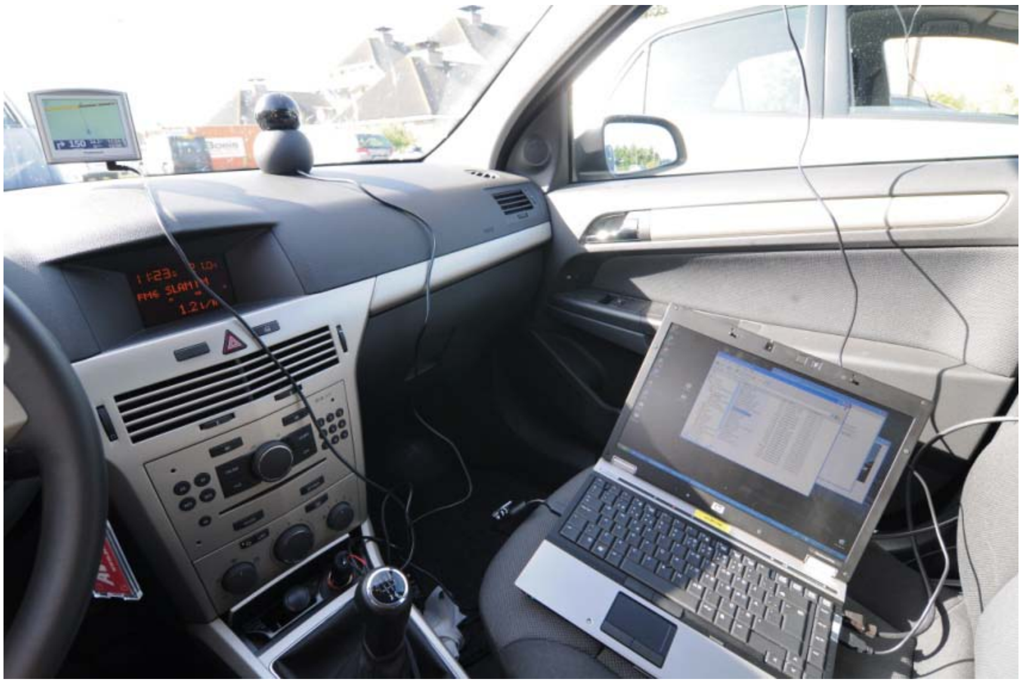


Figure 5.11: Norwegian "deployed forces"

5.5.1 Mobile Ad Hoc Networks (MANET)

Mobile ad-hoc networking (MANET) was identified as a flexible technology to facilitate communications in areas without a pre-deployed infrastructure. By multi-hop networking, communications coverage can be achieved wherever (military) users jointly operate. Furthermore, by interconnecting one or more MANET gateway devices to networks of opportunity in the vicinity (such as GSM, UMTS, BGAN or others), all mobile devices in the MANET could achieve a reachback to their higher commands. This could be exploited to achieve not only enhanced communications for the military users themselves (which could be seen as disadvantaged users in a service oriented architecture (SOA) environment) but also to improve the situational awareness of the military commander.

The Mobile Ad Hoc Network (MANET) as provided by NC3A consisted of "BreadCrumb" MANET nodes from Rajant Corp, which basically are ruggedized components using civil 802.11b/g technology for communications. In addition, the reachback to the core network consisted of a Multi-Function Access Gateway (MFAG) client and gateway, together with commercial UMTS and/or BGAN provisioned Internet service.

The Norwegian MANET consisted of WM600 tactical radios capable of forming a multi-hop MANET. The NC3A used Breadcrumb radios from Rajant, Both the NC3A and the FFI MANET technologies are IP-enabled, and can carry Web services traffic. However, the radios are not compatible on the air, since WM600s typically are configured to use a military frequency, whereas 802.11b/g uses the civil 2.4GHz ISM band. WM600 supports IP multicast. The Breadcrumbs do not support multicast.

5.5.2 Detailed Architecture

As mentioned earlier, the exercise scenario featured four classes of actors, the "NATO DJTF HQ" (deployed), "Norwegian static command" (deployed), "NATO Liaison Officer" (roaming), and "Norwegian deployed forces" (roaming). This setup implied that there were four distinct networks, namely the NATO fixed network, the Norwegian fixed network, the NATO MANET, and the Norwegian MANET.

For security reasons, we were not allowed to interconnect all four networks, as this would have created a "short circuit" (the two fixed networks would have been connected both through the Combined Endeavor core network *and* through the MANETs). We therefore ran our experiments in two different configurations, as illustrated in Figures 5.12 and 5.13.

In configuration 1 (see Figure 5.12 and 5.13), the FFI Combined Endeavor System consists of the following subsystem/groups: The FFI MANET was distributed over two cars and two fixed locations (the Norwegian tent and the NATO tent). In the NATO tent, a Kongsberg WM600 UHF radio was connected to the Norwegian MANET and a Rajant Breadcrumb radio was connected to the NC3A MANET. These two radios were connected through a laptop (Win XP) with two network interface cards that functioned as a gateway between the two MANETs, using the DSProxy software. In addition, the laptop ran a service discovery gateway that enabled federated service discovery across the networks. In the Norwegian tent another Kongsberg WM600 UHF radio was connected to the Norwegian MANET. This radio was only connected to a laptop (Win XP) running DSProxy and software for visualizing tracks and images from the cars. There was no connection from the MANET radio in the Norwegian tent and into the Norwegian fixed network.

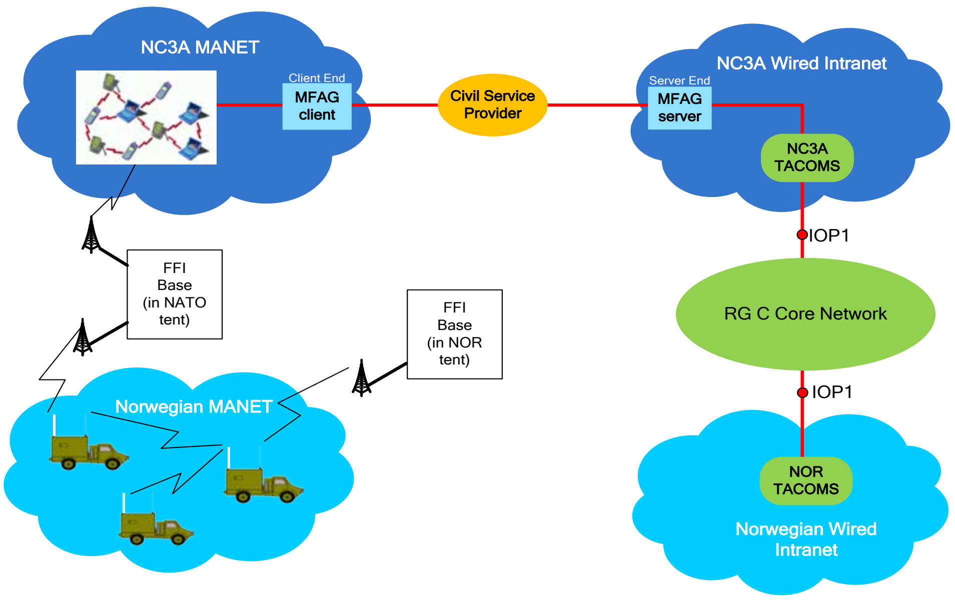


Figure 5.12: Major sub-system groups in the FFI and NC3A Combined Endeavor systems, and their interconnections in Configuration 1

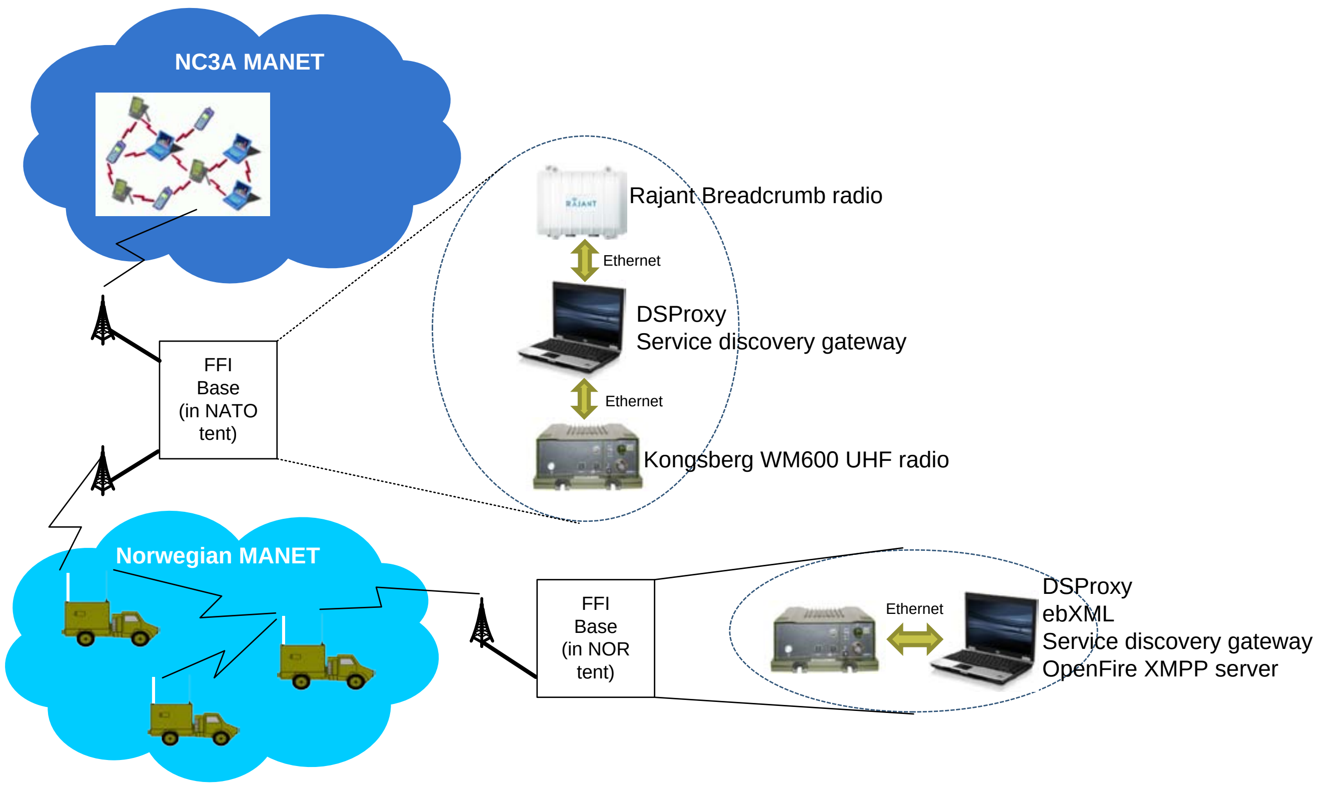


Figure 5.13: Systems and interconnections for Configuration 1

Finally, we had a connection from the Norwegian fixed network into the Combined Endeavor core network through a TACOMS router. A PC running an ebXML registry was also connected to the router. This registry was federated with the NATO registry, to enable search for services across domains (see Figure 5.5).

Configuration 2 is largely similar to configuration 1, with the following differences: In the NATO tent, the laptop was only connected to the Kongsberg WM600 radio, and not to the Rajant Breadcrumb radio. This means that there was no longer a connection between the Norwegian MANET and the NC3A MANET. Furthermore, in the Norwegian tent, the laptop connected to the WM600 radio was also connected to the TACOMS router. This was possible because the laptop was equipped with two network interface cards. Configuration 2 is shown in Figure 5.14 and 5.15.

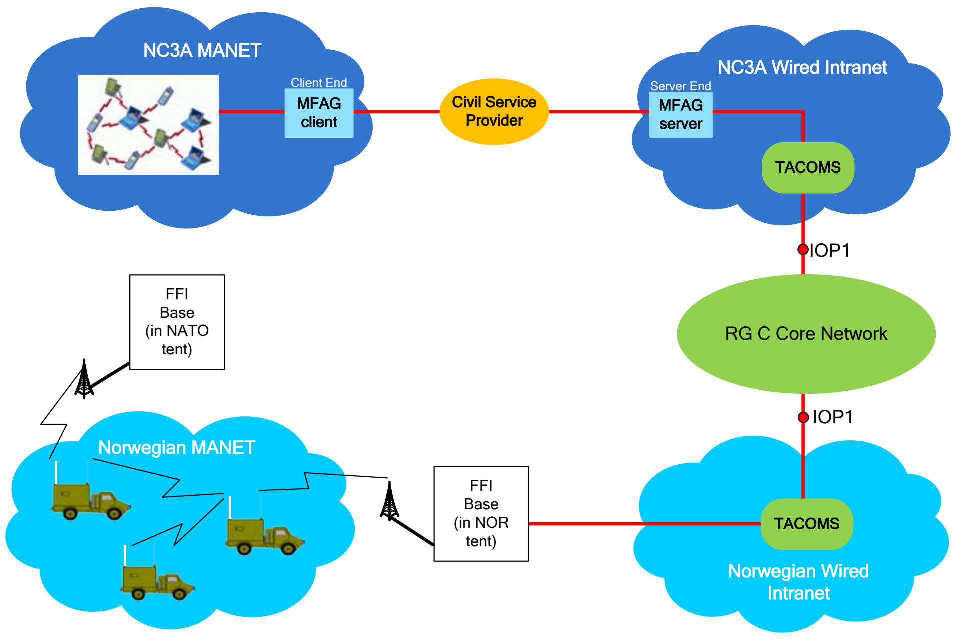


Figure 5.13: Major sub-system groups in the FFI and NC3A Combined Endeavor systems, and their interconnections in configuration 2

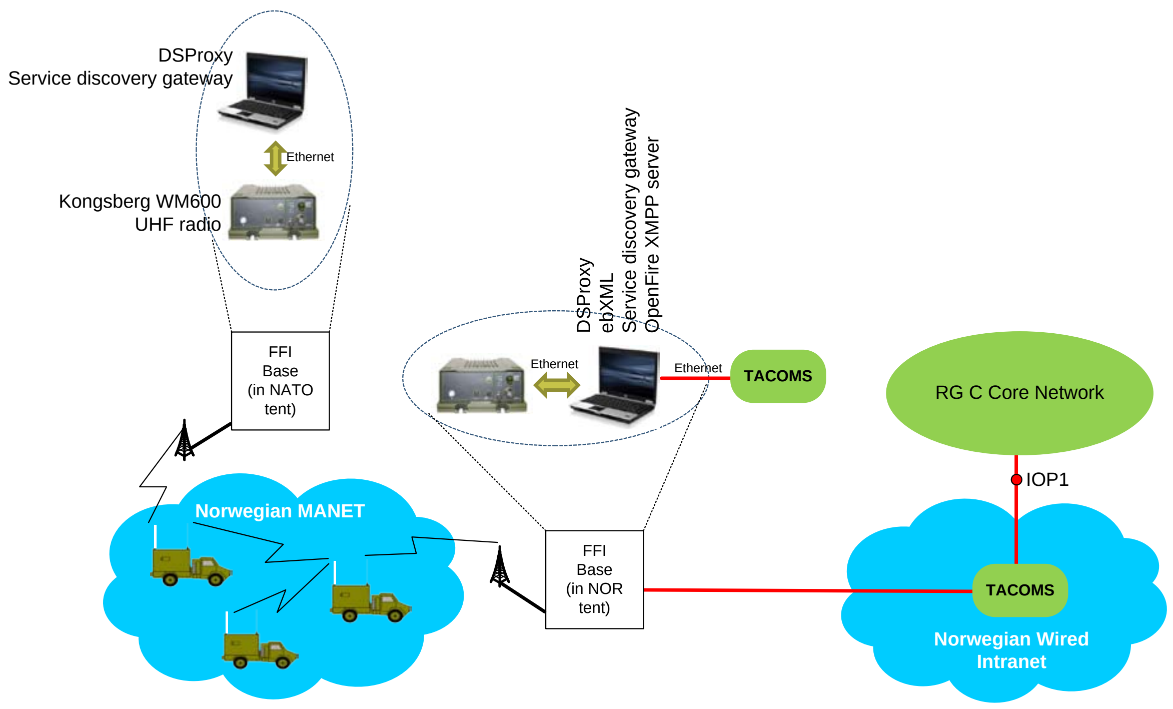


Figure 5.14: Systems and interconnections for Configuration 2

Finally, as mentioned earlier, the two cars representing “Norwegian deployed forces”, were each equipped with a laptop, GPS, web camera, and a Kongsberg WM600 radio, as illustrated in Figure 5.15.

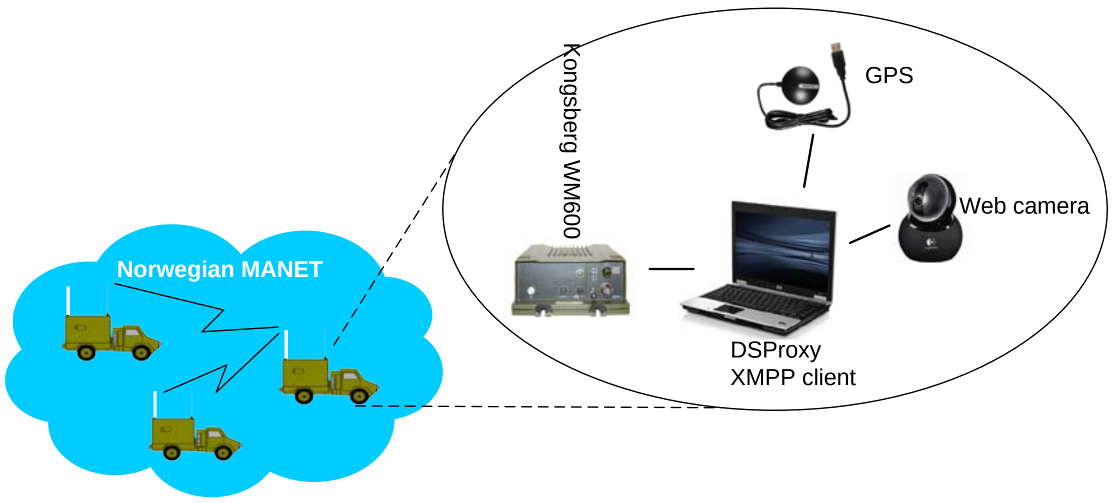


Figure 5.15: System and interconnections in the vehicles

5.6 Results and experiences

The experiments performed by at Combined Endeavor took considerable effort for the participants. In addition to several months of preparation at both FFI and NC3A, we had personnel present on site in Zoutkamp for three weeks.

Overall, the experiments performed by FFI and NC3A at Combined Endeavor were very successful, and provided valuable insight.

First of all, we showed that SOA implementations can be federated across heterogeneous networks, but that following open and agreed civilian standards – and NATO profiles – is important to ensure interoperability. This is best illustrated by the fact that in the preparation period, NC3A and FFI agreed on which standards and interfaces to use during the experiment. When we arrived at Combined Endeavor, we were able to connect our systems with a relatively low effort, and within a couple of days, the systems were interconnected.

Furthermore, we showed that the NATO NEC Core Enterprise Services (as proposed by the NC3B CESWG) successfully provided an underlying SOA foundation. Among other things, we showed how dynamic service discovery adds flexibility with respect to dynamically invoking services (see [13]), and we demonstrated how data can be transformed via Mediation services

Finally, through our experiments with Web services over MANETs, we found that low bandwidth[7] is not necessarily a big issue for Web services in disadvantaged grids, as this can be effectively dealt with. Unreliable connectivity, on the other hand, is a much bigger concern, since the communication protocols used in standard Web services are unsuited for such environments. However, we were able to demonstrate that, by introducing an overlay network (provided by the DSProxy) that hides the unreliable connectivity from the application layer, Web services can successfully be used in disadvantaged grids.

In particular, the store-and-forward mechanism provided by the DSProxy overlay proved to be very important in unreliable networks, in order to avoid having to re-establish end-to-end connections each time the network has lost connection. We therefore recommend the use of a store-and-forward mechanism such as the DSProxy in such an environment.

6 Conclusions

NATO NEC presents an ambitious requirement that users at all operational levels should be able to seamlessly exchange information. This implies that legacy strategic and tactical systems must be integrated into a common network, and SOA, realized using Web services, is an essential mechanism for achieving such integration. In this report, we have therefore focused on how to enable the use of standard, unmodified Web services as far out on the tactical level as possible, since this is where the biggest challenges are.

[7] In the order of Kbit/s

One important mechanism for enabling Web services in disadvantaged grids is Publish/subscribe. We have therefore presented the two existing Web services publish/subscribe specifications, and through our analysis of these chosen WS-Notification as our preferred mechanism.

Furthermore, we have introduced the DSProxy, which is an proxy-based overlay mechanism that hides the heterogeneity and instability of tactical networks from the application layer, and makes it possible to use unmodified Web services across heterogeneous, including disadvantaged, networks.

Finally, we presented the experiments we performed together with NC3A at Combined Endeavor 2009 in the Netherlands, where we demonstrated the use of DSProxys to enable service invocation across both fixed and wireless networks (MANETs). These experiments showed that it is possible to use Web services even in highly unstable networks, but this requires that an overlay mechanism such as the DSProxy is present.

References

[1] OASIS WS-Notification (2006) TC
http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsn

[2] W3C Web Services Eventing (WS-Eventing) public draft release
http://www.w3.org/Submission/WS-Eventing/

[3] http://www.ibm.com/developerworks/webservices/library/specification/ws-roadmap/

[4] P. Bartolomasi, T. Buckman, A. Campbell, J. Grainger, J. Mahaffey, R. Marchand, O. Kruidhof, C. Shawcross, K. Veum, "NATO Network Enabled Capability Feasibility Study", Version 2.0, October 2005

[5] WS-BaseNotification 1.3 OASIS Standard, approved October 1st 2006
http://docs.oasis-open.org/wsn/wsn-ws_base_notification-1.3-spec-os.pdf

[6] WS-BrokeredNotification 1.3 OASIS Standard, approved October 1st 2006
http://docs.oasis-open.org/wsn/wsn-ws_brokered_notification-1.3-spec-os.pdf

[7] WS-Topics 1.3 OASIS Standard, approved October 1st 2006
http://docs.oasis-open.org/wsn/wsn-ws_topics-1.3-spec-os.pdf

[8] http://www.w3.org/2008/11/ws-ra-charter.html#scope

[9] Mail from Steve Holbrook (IBM) to the W3C mailinglist,
http://lists.w3.org/Archives/Public/www-ws/2008Jul/0006.html

[10] NATO FRIENDLY FORCE INFORMATION (NFFI) INTERFACE PROTOCOL DEFINITION IP3 VER. 2.0, April 2007

[11] NFFI Service Interoperability Profile 3 (SIP3) Technical Specifications (Version 1.0.0), NC3A, Nov. 2008

[12] Core Enterprise Services Framework V1.2, ISSC Core Enterprise Services Working Group, Jan 2009

[13] F. T. Johnsen, J. Flathagen, T. Hafsøe, M. Skjegstad, N. Kol, "Interoperable Service Discovery: Experiments at Combined Endeavor 2009", FFI-Rapport 2009/01934, November 2009.

[14] T. Hafsøe, F. T. Johnsen, K. Lund, and A. Eggen, "Adapting Web Services for Limited Bandwidth Tactical Networks", 12th International Command and Control Research and Technology Symposium (ICCRTS), Newport, RI, USA, June 2007.

[15] S. Tilkov, "10 Principles of SOA, A frame of reference – for –SOA-related discussions, SOA World Magazine, March 2007, http://soa.sys-con.com/node/346363