Coalition Networks for Secure Information Sharing

CoNSIS Task 2
Final Report

Version 1.1

9 April 2013

EXECUTIVE SUMMARY

The Coalition Network for Secure Information Sharing (CoNSIS) is a multinational group focusing on technologies and methods that will facilitate the partners' abilities to share information and services securely in ad-hoc coalitions, and between military and civil communication systems, within the communications constraints of mobile tactical forces.

The work done within the CoNSIS group has been divided into a number of tasks, each focusing on a different aspect of interoperability issues. In CoNSIS Task 2, we have looked at how to achieve interoperability in practice, by addressing challenges that arise due to limited functionality in standards, ambiguities in standards, and resource limitations in radio-based networks.

Our goal was to verify if using the Web service standards, as specified by the Core Enterprise Services in NATO Network-Enabled Capabilities, as interoperability enablers, independent implementations of information infrastructures are able to interoperate with only the service specifications as a common reference.

The experiments did prove that Web services can be utilized as an enabler for interoperability in radio networks, but it also allowed us to identify a number of technical lessons learned, and also some challenges which should be addressed in greater detail in the future.

Table of Contents

1 INTRODUCTION

The Coalition Network for Secure Information Sharing (CoNSIS) is a multinational group consisting of members from Germany, France, USA, and Norway, with participants from both research institutions and industry. The objectives of this group are to develop, implement, test, and demonstrate technologies and methods that will facilitate the partners' abilities to share information and services securely in ad-hoc coalitions, and between military and civil communication systems, within the communications constraints of mobile tactical forces.

The group has focused on practical application of information infrastructure technologies in a network-of-networks, consisting of a variety of low capability network technologies. The work done within the CoNSIS group has been divided into a number of tasks, each focusing on a different aspect of interoperability issues. This report covers the domain of Service-Oriented Architectures (SOA) and its application in limited capacity networks, with particular focus on interconnection of infrastructures. During June 2012 CoNSIS conducted a large-scale experiment in Greding, Germany, in which all the different aspects of technical interoperability were tested; integrating the work of all the task groups of CoNSIS.

The remainder of this report is structured as follows

- Chapter 2 provides an introduction to CoNSIS and the different task groups
- Chapter 3 introduces Service-Oriented Architectures
- Chapter 4 presents the work done in Task 2
- Chapter 5 describes the experiment performed in Greding, Germany, in June 2012
- Chapter 6 presents the conclusions of our work

2 CONSIS

The CONSIS areas of work are broken down into five major tasks as follows:

- ♦ Task 1 - Communication Services
- ♦ Task 2 - Information and Integration Services
- ♦ Task 3 - Security
- ♦ Task 4 - Management
- ♦ Task 5 - Architecture, Test & Demonstration, and Coordination

Task 1 has focused on activities to support a general goal of an overall NII infrastructure based on IP technology. The focus of this task has been on demonstrating solutions that will work within the communications constraints and dynamic topology imposed by highly mobile tactical networks. Communication services within tactical systems have been analyzed towards their ability to support SOA core services (e.g. discovery), real time services (e.g. VoIP and VTCoIP) and streaming services (e.g. TADIL).

Task 2 has demonstrated the applicability of the SOA approach in a multinational military environment, federating the SOAs of each nation. The task has been taking into account the constraints applicable to highly mobile tactical forces (including limited bandwidth, intermittent communications, high rate of change of network topology, and the need to make decisions quickly).

Task 3 investigated, specified, and demonstrated security mechanisms for use for integration and interoperability of heterogeneous, coalition networks. The likely next expansion of military networks will be into highly mobile platforms on the tactical edge. A second area of interest has been to develop and demonstrate practical, yet secure, black-core, network topologies and architectures for coalition interoperability. A third area of interest has been to investigate the use of Multilevel Security (MLS) including the use of virtual terminal technology for cross-domain solutions that support network communications operating at multiple security levels without separate infrastructure being required for each security level. The goal was also to allow coalition network access via networks operating at national security levels.

Task 4 has explored, specified, and demonstrated mechanisms for automatic management services and service levels in coalition networks. The main challenge is to automate the end-to-end management across multiple security domains during changing operational and network situations. This requires mechanisms to detect changes and operational policies that define the actions to be taken. A second area of interest has been the detection of a jammer attack autonomously per vehicle or on a cooperative basis.

Task 5 has developed an overall Experimentation Architecture for CoNSIS. This architecture defines the way in which the deliveries of tasks 1 to 4 are integrated. The task has also carried out the overall co-ordination and planning of the CoNSIS Project. It has provided reporting and dissemination of the results of CoNSIS during and upon completion of the Project. The intention has been to demonstrate technical results that can transition to an operational demonstration/scenario (outside this MoU).

2.1 Motivation

NATO Network Enabled Capability is first and foremost about achieving better interaction between the different actors involved in military operations. This implies more efficient exchange of information. Consequently, the NATO information infrastructure will consist of a federation of systems, including a plethora of different information and communication systems, as well as a mix of new and legacy systems. NATO recommends a service-oriented architecture approach based on Web services to enable such a federation.

For Task 2, the goal of the CoNSIS experimentation was to show that by using the Web service standards, as specified by the Core Enterprise Services in NATO Network-Enabled Capabilities (NNEC CES), as interoperability enablers, independent implementations of information infrastructures are able to interoperate with only the service specifications as a common reference. The reasoning behind focusing on standards as a means of achieving interoperability was that it enables us to evaluate not only the implementations used, but also the standards themselves. Standards are not always sufficient as a specification for implementation, since there may be room for interpretation. The best way to discovery such ambiguities in the standards is to actually build systems based on the standards, and test these against other implementations, which is exactly what we have done in Task 2.

In addition, by focusing on tactical networks with low data rate, we are able to assess how suitable the standards specified in the NNEC CES are for use in such networks.

3 SERVICE-ORIENTED ARCHITECTURES

Military systems are often realized as stove-pipe systems. That is, large integrated systems comprising everything from hardware to presentation software, and designed to perform a specific military function. As such stove-pipe systems grow larger, they become complex and difficult to maintain, and they make sharing of information and functionality difficult.

A better solution is to split the systems into functional modules, and make these communicate with an infrastructure over standardized interfaces and using standardized protocols. In addition, by interconnection infrastructures, both information and functionality can be made available across domains and communities of interest. However, for this to work, standardized interfaces, protocols and data formats must be used, both within and between infrastructures.

In the NNEC CES, NATO has chosen this approach, and based it on service-oriented architectures (SOA).

3.1 What is SOA

In the SOA Reference Model by Oasis [1], SOA is defined as follows:
SOA is a paradigm for organizing and utilizing distributed capabilities that may be under the control of different ownership domains. It provides a uniform means to offer, discover, interact with and use capabilities to produce desired effects consistent with measurable preconditions and expectations.

The core concept in any SOA is that of a *service*, which Oasis defines as follows [1]:
A mechanism to enable access to one or more capabilities, where the access is provided using a prescribed interface and is exercised consistent with constraints and policies as specified by the service description.

Figure 1 shows the three central entities in the SOA paradigm, and how these interact with each other. The service provider has a capability that it wants to make available to others, and it does this by offering the capability as a service. In addition to the service itself, the service provider also has to supply a description of the service. The service provider can make its services known to potential users by publishing the service using a service discovery mechanism. The responsibility of the service provider is limited to providing the service discovery mechanism with the service description, while the distribution of this description is handled by the service discovery mechanism. A service consumer wanting to use a service first has to find the services that are available to it using the service discovery mechanism. The consumer then retrieves the service description for the desired service, and uses the information therein to invoke the service directly from the service provider.

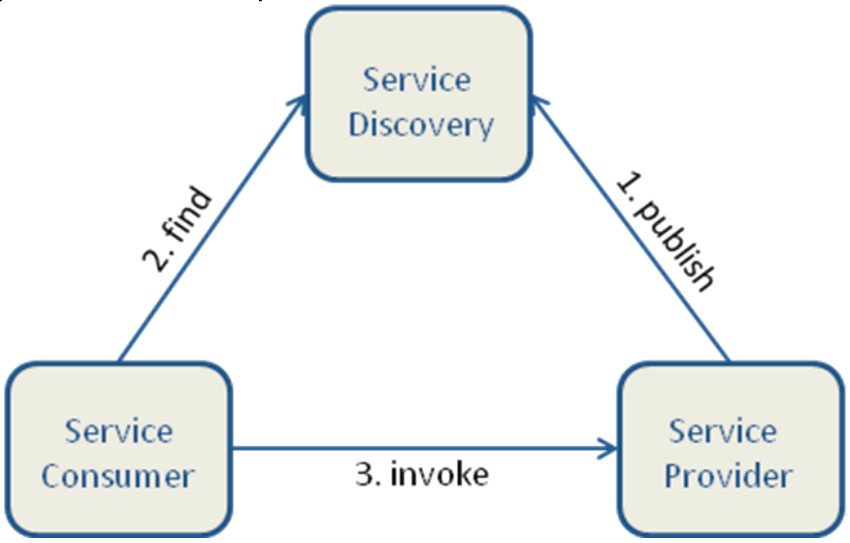


Figure 1: The SOA triangle

Services in a SOA can be realized in three different ways, as shown in Figure 2:

- They can be implemented as services from the start. This is what is normally done when implementing new systems.
- Existing (non-SOA) systems can be “wrapped”, in the sense that a piece of software is placed in front of the system, presenting services to the outside world, while communicating “natively” with the wrapped system.
- Existing services and systems can be combined into new, aggregated services, by using software that coordinates access to the individual systems and services, and presents the aggregate as a new service.

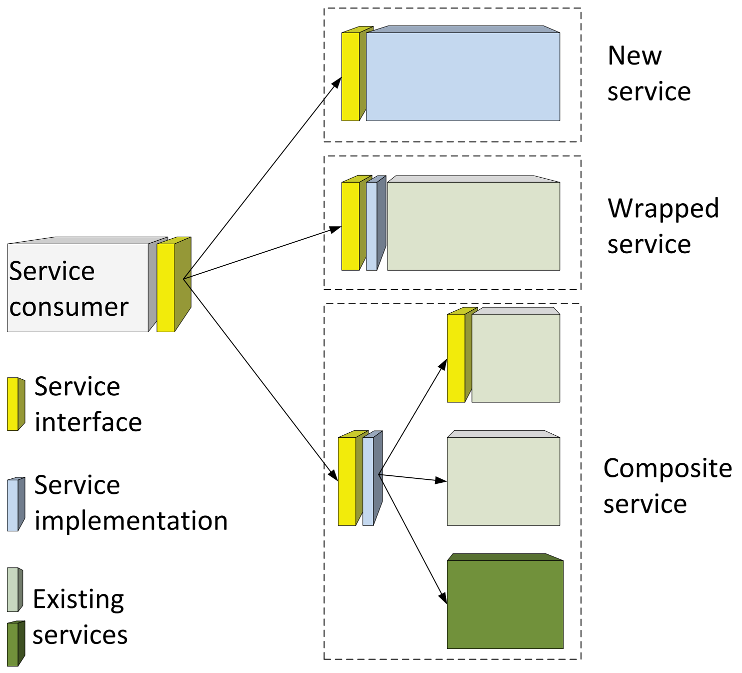


Figure 2: Realizing services

The NNEC Feasibility Study [2] presents a discussion of technology, focusing on the needs of future interoperable military communications. The information infrastructure has to allow for communication across system and national boundaries while at the same time taking legacy systems into account.

The need for inclusion of legacy systems is what makes SOA a suitable choice. As shown in Figure 2, the wrapped service-approach provides the possibility to include legacy systems in a SOA environment, and make them available as services.

3.2 Web services

Web services technology is becoming increasingly popular as a technology for realizing the SOA paradigm. Businesses use the technology not only as a middleware within their own corporate networks, but also across the Internet as a means of providing business-to-business services. The technology, being based on standards, makes it a simple and cost effective way to build loosely coupled systems. Since Web services are so successful in civil applications, it makes sense to attempt to employ this technology for military applications as well.

The NNEC Feasibility Study identifies Web services technology as the key enabler for realizing SOA in NNEC. The World Wide Web Consortium (W3C) has defined Web services as follows [3]:
A Web service is a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-

processable format (specifically WSDL). Other systems interact with the Web service in a manner prescribed by its description using SOAP-messages, typically conveyed using HTTP with an XML serialization in conjunction with other Web-related standards.

In Section 3.1 we described how SOA require service descriptions for all services. When using standard Web services, these service descriptions are formatted using the Web Services Description Language (WSDL). Furthermore, communication is done using the SOAP protocol, which usually (but not necessarily) runs on top of HTTP and TCP.

When Web services operate over military networks, it is important to consider that Web services messages are often relatively large compared to typical bandwidth capabilities and other network resource capacities of radio-based military networks. In [4] and [5], some of the challenges related to service invocation in military networks are addressed, more specifically which optimizations are needed for Web services to function in low capacity networks and how to achieve delay tolerance for cross-network invocations.

Within NATO, a number of common infrastructure services have been defined as core enterprise services. Having a common understanding of how these services are to be implemented and used is critical when attempting to achieve interoperability across national and systems boundaries. In particular, we sometimes see that the standard specifications are not always completely unambiguous. This can, in turn, lead to different implementations of the same service being incompatible. Therefore, an important part of the work towards achieving a common understanding is to utilize these services in experimentation, in which the candidate technologies are tested under conditions similar to those found in an operational network.

3.3 Service discovery

Service Discovery is the process of finding the services that are available in the network. NNEC CES has recommended the use of the registry based solution Universal Description, Discovery and Integration (UDDI) for this core service.

Basically, UDDI allows service providers to register their services and service consumers to discover these services. UDDI is defined as a Web service itself, meaning that the SOAP protocol must be used to interact with the registry. In principle, UDDI is centralized, but mechanisms for federating several registries have also been specified in version 3 of the specification. Having multiple registries, or letting a registry consist of several nodes that replicate data, increases the robustness of the discovery solution. In UDDI, replication between registry nodes must be configured manually. It is also possible to let several separate UDDI registries exist independently of each other, but information will not be replicated unless a custom scheme is designed. Additionally, a hierarchical model may be used, using a root registry and affiliate registries. The UDDI registry reflects the liveness of services (i.e., whether a given service is available or not) to the extent that the services informs the registry before shutting down. Services that go down unexpectedly, on the other hand, will remain visible in the registry indefinitely, since UDDI has no mechanism for checking the liveness of services.

All core entities in the UDDI information model are assigned unique keys. UDDI provides a flexible model in that specific service types can be registered with the registry and referenced by service instances that implement the service type. This is called a technical model, or tModel, in the UDDI information model. A tModel can be used for different purposes. For instance it can include pointers to further description of a service, for instance its WSDL description and bindings. It should be noted, though, that tModels can have arbitrary (including proprietary) formats, which in turn could result in incompatibilities between different entities using UDDI.

However, when operating in a wireless network environment where node mobility and shifting network conditions can cause network partitions and loss of network connections, it is vital to use a service discovery mechanism that does not rely on the availability of any given node. In other words, we need a fully distributed service discovery mechanism [6]. The only standardized Web service discovery protocol that currently fulfills this requirement by operating in a distributed mode is WS-Discovery [7], which was utilized during the SOA experiments in CoNSIS.

3.4 Publish/subscribe

Web services support both traditional request/response (“pull”) communications, as well as publish/subscribe (“push”) - a well-known pattern for event-driven, asynchronous communication. Simply speaking, publish/subscribe means that you will only receive the information that you have subscribed to. In other words, instead of a node having to repeatedly check if there is new information, the node simply sends a subscription request to the information provider, asking to be notified whenever new information is available (see Figure 3). 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. Using pure “pull” mechanisms introduces a compromise between wasted requests (when no new information is available) and liveness/latency requirements.

The publish/subscribe pattern is particularly well suited in situations where information is produced at irregular intervals, and its importance is demonstrated by the fact that it has been identified as one of the core services by the NATO CES working group [8].

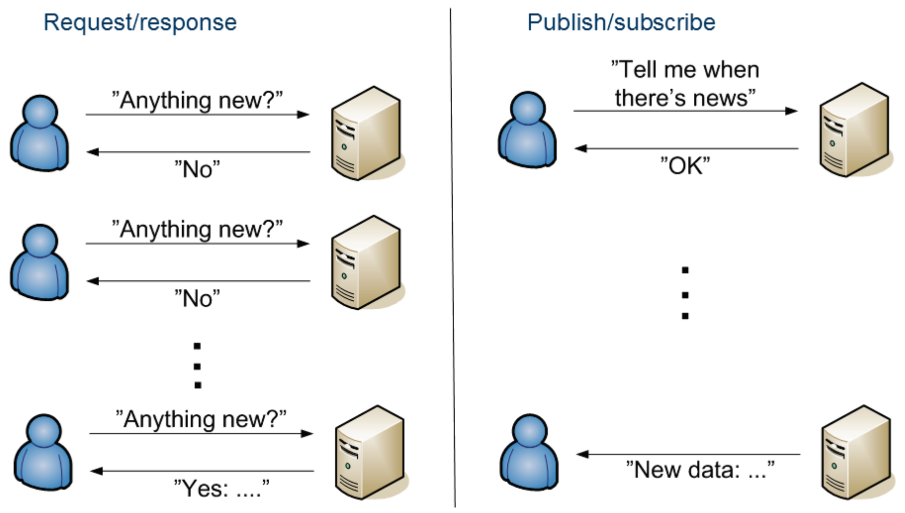


Figure 3: Request/response versus publish/subscribe

WS-Notification [9] is an OASIS standard and consists of a group of specifications that enable publish/subscribe-based communication between Web services. It comprises WS-BaseNotification, WS-BrokeredNotification and WS-Topics. While WS-BaseNotification defines which interfaces consumers (clients) and producers (servers) should expose, WS-BrokeredNotification introduces the concept of a message broker, an intermediary node which decouples consumers and publishers, and relieves producers from several tasks associated with publish/subscribe. NNEC CES specifies WS-Notification as the standard to be used for publish/subscribe in NATO.

3.5 Topics

The organization of information flows in WS-Notification is fundamentally different from the request/response paradigm: When looking for a request/response service, you are interested in a service with a particular interface. This is because that interface is the only aspect of the service that is known to the consumer, and thus represents the only interface that consumer is about to invoke. The service description for a service, the WSDL file, does not contain information about the actual content provided by the service.

For publish/subscribe, on the other hand, all services are equal with respect to the actual interface, and you need information about the content the service offers in order to distinguish between content providers. Consequently, when moving from request/response to publish/subscribe, traditional service discovery is no longer enough. This is because all publish/subscribe endpoints will appear as the same service type, generating a need for additional meta-information about services.

In WS-Notification, the actual notifications are normally always of the same type, independent of the actual information that is delivered (i.e., the payload of the notification). Consequently, all WS-Notification services have the same service type, and it is not possible to determine what information a given service provides, based on its service type. So when a client wants to subscribe to a specific type of data, it therefore expresses the type of information it is interested in by including a *topic* in the subscription request.

For WS-Notification, the WS-Topics standard specifies how such topics should be expressed. It also defines three topic expression dialects, to allow for topic expressions of different complexity. This use of topic dialects means that one can express a number of different topic structures within the same standard, including defining one's own dialect for handling topics. While this topic handling scheme is flexible, the added complexity using such topics means that one needs to agree not only on which topics to use, but also which dialect to use to express topics before communication is possible.

3.6 Enterprise Service Bus

The Enterprise Service Bus (ESB), sometimes also called Enterprise Integration Bus, is a central component meant to simplify the connection of services in complex SOA environments. Typically an ESB consists of a set of tools for reliable and assured message transfer, routing-mechanisms for message-distribution, pre-designed adaptors for the integration of different systems, management- and government tools and other components.

The usage of an ESB is recommended for the following reasons:

- Modularity
- Flexibility
- Fast integration of functionalities
- Standardized data streams by using WSDL
- Abstraction from physical conditions (security, transport)

Figure depicts a general consumer-/provider structure in a SOA environment. This figure is the basis for the considerations to follow and, despite its simplicity, it contains some important statements.

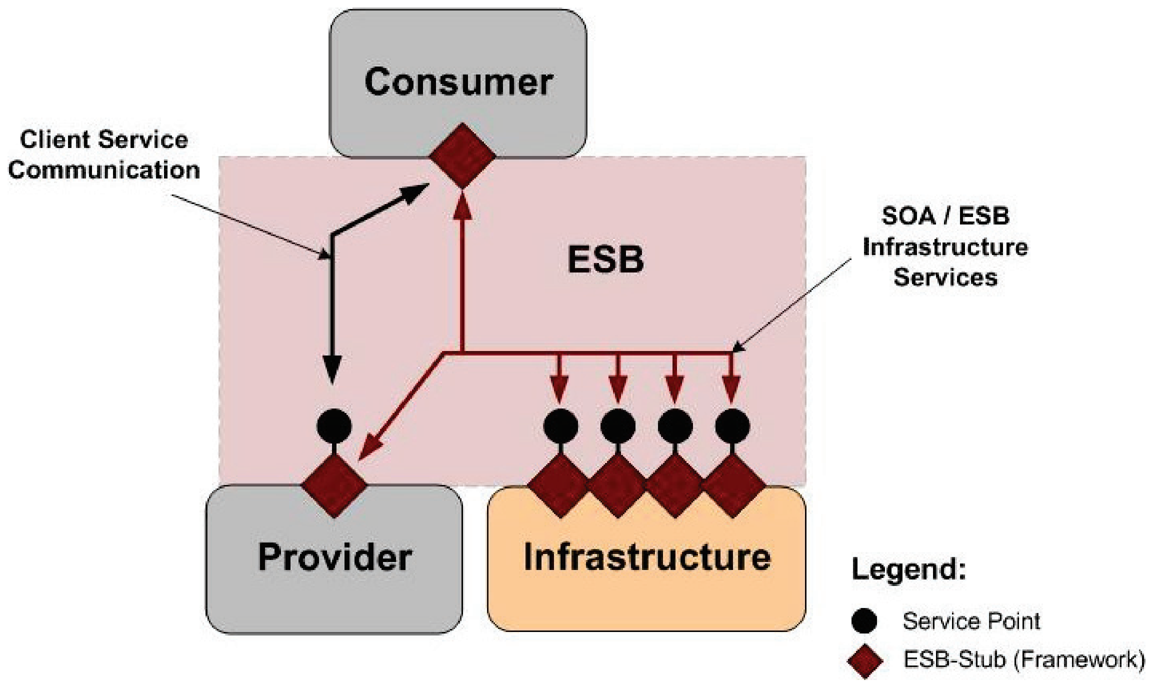


Figure 4: General provider / consumer structure in an ESB environment

The use of an ESB extends the general SOA model of provider, consumer and service discovery. Since an ESB will usually contain the discovery mechanism, we arrive at these four main components of an ESB-based SOA[1]:

- Provider: A provider makes a service available to one or more consumers.
- Consumer: A consumer is an application that uses a service of a provider. In turn, a consumer again may provide a service to other consumers.
- Enterprise Service Bus: An ESB forms a kind of middleware that mediates between a service provider and one or more users (consumers). As a minimum the ESB provides routing, messaging, transformation, mapping and logging capabilities.
- SOA (ESB) Infrastructure: The SOA (ESB) Infrastructure components are part of the ESB. They are usually located centrally and, provide basic services like service discovery, directory- and security services.

The ESB used in the German Referenzumgebung Dienste (RuDi) [10], actually consists of only one component – the ESB stub, through which any further component (e.g. provider, consumer) is connected. It is co-located with a service component and can be regarded as a library providing access to certain ubiquitous capabilities such as message transport and transformation, but also connection to the ESB infrastructure. A provider offers a service endpoint and a consumer uses the service of a provider

[1] Note that there is no single, agreed-upon, definition of an ESB. In general, an ESB is a toolbox, and which tools are available varies considerably between different ESBs. Thus, other ESBs may not necessarily have all the components described here

through the ESB stub. It will perform basic tasks such as obtaining the service description or public keys for encryption automatically by calling the infrastructure services as necessary. Indeed, the ESB infrastructure components contain the ESB stub as well. Through the connection of the ESB stubs, a virtual Enterprise Service Bus is formed.

3.7 Challenges when using SOA in tactical networks

SOA based on web services and even more so ESBs, are intended for high-bandwidth IP networks. This means that deploying a service-oriented system in a tactical, mobile setting is challenging, given the limited resources available in such settings. For instance, the bandwidth can be very low in the available transmission mediums, which in mobile military networks include VHF, UHF, TDL, and SATCOM. Using Web services and ESBs without modifications in a tactical environment would easily result in overloaded networks.

As technical limitations of network elements are a critical aspect of the tactical usage of SOA implementations, information about these limitations must be made available to services intended to be used across the network. Such limitations should be expressed with Quality of Service (QoS) or transmission attributes, and a standard for such attributes needs to be developed. It is also necessary to extend, e.g., service registries, to not only store the availability of services, but also the conditions under which these services can be used.

To make sure the sparse bandwidth of tactical mobile networks is used efficiently, standardized, efficient compression of XML structures is required (including the backbone connections). Moreover, in military scenarios it is often the case that the same information is distributed from one source to several receivers. In civil networks, dissemination of such information is usually done using one unicast connection per receiver, but multicast would in many cases be far more efficient (depending on network topology, routing algorithm, etc). This necessitates a change in the communication mode between services for tactical networks.

Additionally, in tactical networks, node availability and visibility is not always given. Nodes may be in a radio shadow, under radio silence, network connections may be established only on a temporary basis and with permanently changing conditions.

Services in a tactical domain will in many cases be only partially visible to partners (where the availability is limited by geographical, tactical or administrative limitations). In addition, standard military rules will often enforce the operation of critical services close to consumers - an operational process may require a set or sequence of services which are all necessary and mandatory; the execution of autonomous (self-sufficient) operations generally means that a standard service repository accessed remotely (a service cloud) cannot be accepted in a military domain. As a consequence, in tactical domains, the number of service providers must be increased and located close to the service consumers to ensure availability.

Another challenge is presented by coalition operations. The integration of coalition nodes in a common operation, which may be pre-planned or not, requires an architecture and policies that allows an ad-hoc organization, provision and usage of services across various domain boundaries. This may include using a service from a coalition partner to process data from another coalition partner. As a consequence, the coalition partners have to trust the implementation and provision of these services.

In coalition operations various information domains have to be protected on request of the local domain owners. This will be in conflict with cross domain operations, e.g. when generating a common operational picture. To avoid intensive pre-planning and configuration, a security model is required which allows a flexible adaptation of a single domain under changing coalition tasks. Domain related security policies for the operation of nodes and services must have a significant overlap with those of coalition partners, allowing an (automatic) synchronization in the case of common operations.

4 TASK 2 FOCUS AREAS

The main task within CoNSIS Task 2 was to verify whether SOA-based infrastructures can be efficiently used and interconnected in tactical domains during coalition operations. In particular, we wanted to verify if the CES specification provided by NATO was sufficient for interoperability for ad-hoc combined elements of coalition partners in various scenarios.

Our focus was on the conditions of military nodes and the additional aspects for coalition operations on lower command level (down from battalion level to small groups). Based on this, a practical experimentation was planned and executed at a military side to verify the usability of these models. This experimentation verified the principle technical feasibility of the approach and identified the still open areas to create a complete and usable model.

4.1 Discovery of services and resources

As described in Section 3.3, service discovery is a term used for any mechanism that allows a service provider to make its service known to potential service consumers. The traditional way to implement service discovery for Web services, and the way recommended by the CES working group, is in the form of stand-alone registries. However, it was clear that in the type of environment we had in the CoNSIS scenarios, a distributed solution was needed.

As described in the CoNSIS Task 1 and Task 3 final reports, the model of co-operation in coalition systems on tactical levels should be primarily ad-hoc oriented. The initial German approach was to use comparable mechanisms at various protocol levels to achieve a homogeneous behavior at communicating nodes. In a short summary, this model was based on the following elements:

- At network level, using a completely distributed IPsec discovery mechanism by pro-actively exchanging (over multicast) information on active crypto devices to automatically generate a security association on an ad-hoc basis (under certain security policy restrictions).
- The same pro-active Hello-mechanism was intended to be used as an information mechanism to inform remote nodes about the service availability from partners. This mechanism was called “Generic Advertisement of Applications (GAA)” [11], which was operated as a peer-to-peer mechanism and which periodically triggered a Synchronization Protocol (SyncD) [12] to exchange (over peer-to-peer) the necessary information on available services for partner usage.

Both protocols were peer-to-peer oriented, with the consequence that they have to be initiated for each peer of nodes in a common operation. This was chosen originally under the constraint that multicast communication was not available in tactical radio networks.

For the actual experiment, however, it was decided that WS-Discovery should be used as service discovery protocol. On the German side, WS-Discovery was integrated into the RuDi system, and connected to the internal service registry. This meant that information from WS-Discovery would be added to the service registry, which in turn meant that the announced service could be invoked from within RuDi. This was done by allowing RuDi to periodically probe for information, but at a low enough frequency to not cause overloading the network. On the Norwegian side, WS-Discovery was used as the only discovery mechanism. A self-contained WS-Discovery application was therefore used for announcing and searching for services, which made it possible to limit the amount of probes sent into the network by only probing when a new service was needed.

WS-Discovery is designed for use in one of two modes: managed and ad hoc. In managed mode all nodes communicate through a discovery proxy, an entity which performs the service discovery function of behalf of all the other nodes, and which communicates with the other nodes using unicast messages. This mechanism could be used to achieve interoperability between registry based service discovery mechanisms and WS-Discovery, but it does introduce a single point of failure, which could be problematic in tactical networks.

In ad hoc mode, on the other hand, communication is fully distributed. Requests for service information are sent using multicast to a known address, and each node is responsible for answering requests from others about its own services. The ad hoc mode is intended to be used for local communication only, and the standard recommends limiting the scope of multicast messages by setting the time-to-live (TTL) field of the IPv4 header to 1, or by using a link-local multicast address for IPv6.

The CoNSIS experiment consists of a number of ad hoc networks connected to each other using Multi-Topology Routers (MTRs) [13], forming an IPv6 based network-of-networks. The dynamic character of these networks implies that one cannot rely on a managed mode discovery proxy to remain available, meaning that the distributed ad hoc mode should be used. However, since this mode is limited to link local communication it will not provide the multi-network service discovery capability required in the CoNSIS experiments. In order to work around this issue, we decided to go against the recommendations in the standard, and allow the multicast discovery messages to travel across network boundaries by using a site-local IPv6 address, and increasing the Hop Limit in the IPv6 header. This solution works within a controlled network environment such as the one used during the CoNSIS experiments, but it is less than ideal for use in larger scale networks. That is because increasing the scope of the multicast messages might cause the messages to travel further than intended, and thus cause increased network load in networks where the messages are not needed.

WS-Discovery is a hybrid discovery protocol, meaning that it has both a proactive and a reactive element (see [6] for further details on the different types of discovery protocols). The proactive element consists of the HELLO and BYE messages nodes send out when they first make a new service available, and when they remove a service, respectively. Other nodes then store the information gathered from these proactive messages, allowing them to perform service discovery without having to actively query for information. This proactive mode works well under stable network conditions, since the likelihood of these messages reaching all other nodes is high. The CoNSIS network is however not stable, which means that many of these messages will be lost, rendering the proactive element of WS-Discovery unable to provide service information to all nodes. This means that one has to rely on the reactive element of WS-Discovery, the PROBE and PROBE MATCH messages.

In this reactive mode a node that requires the use of a service will ask for services matching its needs by sending a PROBE message. This message is sent using multicast, and with the extended scope of multicast messages described above, the probe will reach all other nodes that it currently has a network connection to. Nodes offering a matching service send a unicast PROBE MATCH message back to the probe sender. Note that this reactive mode should be used sparingly in low capacity networks as it generates some network traffic. Since we allowed the packets to flow across routers, a request sent by any one node in the network is received by all other nodes. If the message sent was a probe for available services, then all nodes that did offer a service matching the request would reply with a unicast message to the sender.

4.2 WS-Notification in tactical networks

In the CoNSIS Task 2 experiments, the majority of the information exchanged was distributed according to the publish/subscribe paradigm. Using publish/subscribe instead of the request/response paradigm has several advantages: The network traffic is reduced, since the client doesn’t have to send periodic requests; the server load is reduced, since there are fewer requests to process; and the client will potentially receive new data sooner, although this is dependent on the request frequency in a request/response setting (which in turn will affect network and server load).

However, one added complexity when using WS-Notification in a limited capacity network is that the standard is designed to use unicast message transmission only. That means that, even when multiple nodes in the same network want the same information, WS-Notification will send one unicast message to each recipient rather than send one multicast message that reaches all recipients. In radio-based networks, where the transmission medium is shared, there is a potential for a significant reduction in network load by switching from unicast to multicast. Note that making such as switch will require further functionality to be implemented into WS-Notification, namely the ability to manage multicast group memberships. In addition, the actual benefit of using multicast is dependent on the network in use (topology, routing algorithm, etc) and the number of subscribers.

Furthermore, the verbosity of XML introduces a lot message overhead, making information exchange particularly challenging in low bandwidth environments like disadvantaged grids. Compression techniques enable the use of Web services on low bandwidths, but the degree of compression varies greatly with the volumes of data being transferred within a single message. In general, large messages will achieve a better compression ratio than small messages. By aggregating several small messages before sending them, larger messages and thereby better compression ratios, can be achieved. This, however, means that the individual messages must be delayed until the aggregate message is large enough, which in turn means increased and highly variable latency. In situations where, e.g., up-to-date position information is critical, this may not be acceptable.

In order to achieve favorable compression ratios on small messages, instead of aggregating many small messages from multiple sources over a period of time, one can create a virtual aggregate message, based on data previously exchanged with other nodes. By keeping track of previously exchanged messages (a local cache), and keeping track of which nodes they are exchanged with, a virtual message can be constructed by the sender, comprising one or more previously exchanged messages. Such a scheme was implemented in the DSProxys which were used on all radio connections on the Norwegian side, as well as all radio connections between Germany and Norway. The DSProxy is further described in Section 5.5.2

4.3 KadScribe

The dependency on central servers in mobile and distributed environments is a challenge. In favor of distributed and more reliable approaches an alternative to a broker-based messaging service was investigated. This work was performed within Task 2, but KadScribe was not used in the Greding experiment.

The SOA messaging component responsible for publish/subscribe functionality was chosen to be WS-Notification in Task 2. Publish/subscribe is a messaging scheme which allows a subscriber to subscribe to a certain kind of information (the topic) and enables it to receive updated topic information as soon they become available. It can also be described as a “push” communication rather than a “pull” communication

(Figure 5). Existing application of publish/subscribe are e.g. the use of social networks and instant messaging.

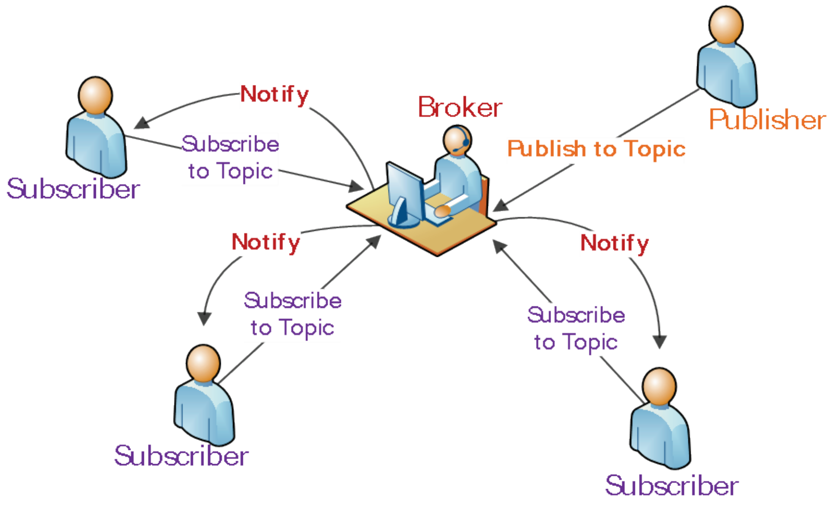


Figure 5: OASIS WS-BrokeredNotification

WS-BrokeredNotification scheme is an OASIS standard for SOA-based publish/subscribe functionality and has a comprehensive set of promising features. One disadvantage the scheme has: It relies on a central server. The central message broker may be a weakness when it comes to connectivity or availability problems at this computer. This may due to directed attacks on the server in a physical way or by cyber-attacks. Also other sources like power failure, atmospheric interferences, misconfiguration or maintenance outages may affect the server.

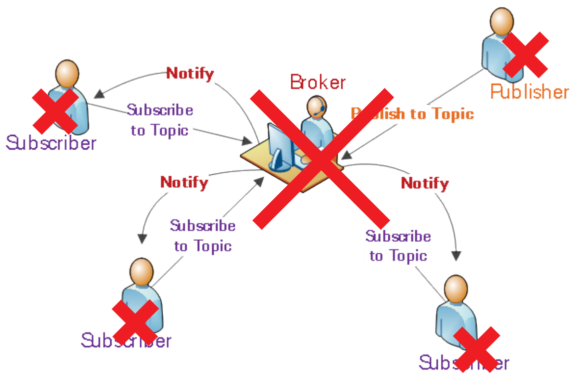


Figure 6: If the broker fails, the publish/subscribe system fails

The goal of this activity was to research how publish/subscribe functionality in SOA can be made more resilient against failures of components such as servers and network infrastructure.

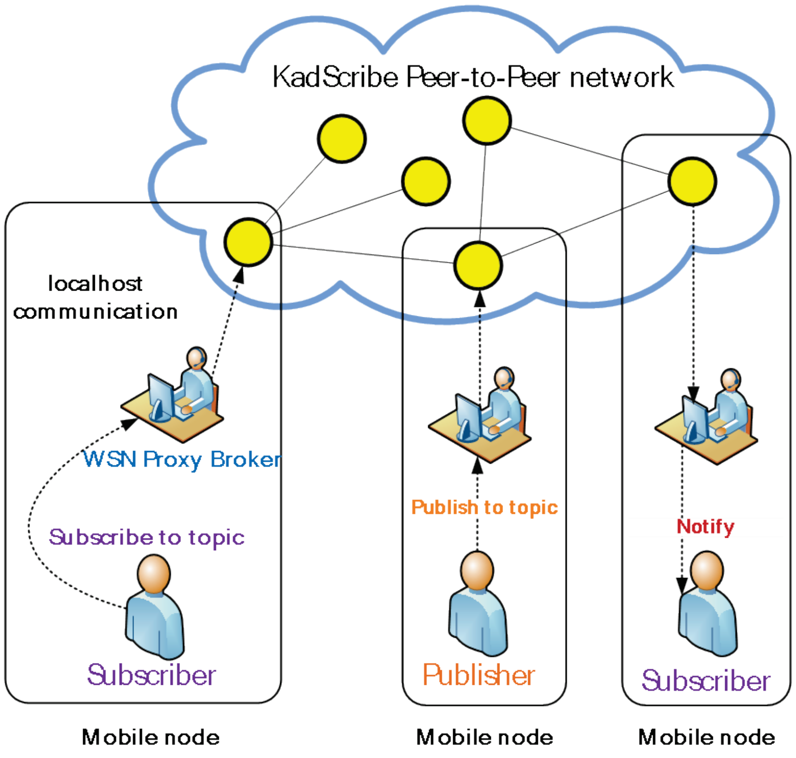


Figure 7: Peer-to-peer messaging with WS-Brokered Notification

Fraunhofer FKIE has implemented a method to circumvent the dependence on a single WS-Notification (WSN) broker server. In contrast to the original WS-BrokeredNotification the WSN broker (the proxy WSN broker) is at every machine. A Peer-to-Peer (P2P) network is used instead of a central broker to distribute messages (Figure 7). By doing so, failures of individual network components or computers do not affect publish/subscribe functionality as a whole (Figure 8). The functionality of a central WSN-Broker is taken over by the WSN Proxy Brokers, an additional mediation layer and the KadScribe P2P transport.

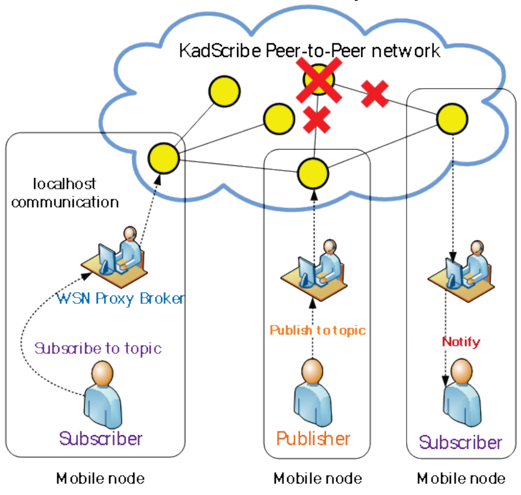


Figure 8 Failures of single nodes in the network do not affect overall publish/subscribe functionality

The implementation with a proxy approach results in compatibility of existing client services with the messaging. No client code change is necessary. The software necessary consists of a new CXF connection factory (`KademliaConnectionFactory`), a set of configuration changes to the WSN broker so it acts as a proxy broker, an intermediary component between the WSN proxy broker (KademliaDaemon) and the KadScribe library which supplies P2P network transport.

5 EXPERIMENT

For Task 2, the goal of the CoNSIS experimentation was to show that by using the Web service standards specified by the CES Working group as interoperability enablers, independent implementations are able to interoperate with only the service specifications as a common reference. The reasoning behind focusing on standards as a means of achieving interoperability was that it enables us to evaluate not only the implementations used, but also the standards themselves. In other words, we can assess how suitable the standards by the CES Working group are for use in tactical networks.

5.1 Venue

The CoNSIS experiments took place in the area of Greding, Germany, an area that was well suited for the purpose:

- Military infrastructure and support being provided by a German military installation (home base of WTD 81). This included both premises for the HQ, and a total of nine military vehicles.
- Large scale internal test area inside WTD 81 (former flight field with large line of sight radio connections without disturbance)
- Geographically interesting area with mountains/hill and deep valleys (for significant shadowing of radio connections)
- Sparsely populated area (allowing the operation of radio systems outside normal technical boundaries and usage of jammers without disturbing civilian installations)
- Ad-hoc operation of military activities (organization of convoys without involvement of civilian authorities)

5.2 Scenario

The scenario used in CoNSIS is described in the main report [14]. The following is a very brief summary of the scenario and use cases chosen for further work in CONSIS from a Task 2 point of view.

Phase 0: A natural disaster occurs in an area not controlled by the coalition. This area is far away from the nearest coalition headquarters. Additionally, in the area there are civilian organizations (non-governmental organizations (NGOs), etc.) that already help the displaced population. There is a danger that enemy forces may take advantage of the situation to infiltrate the area.

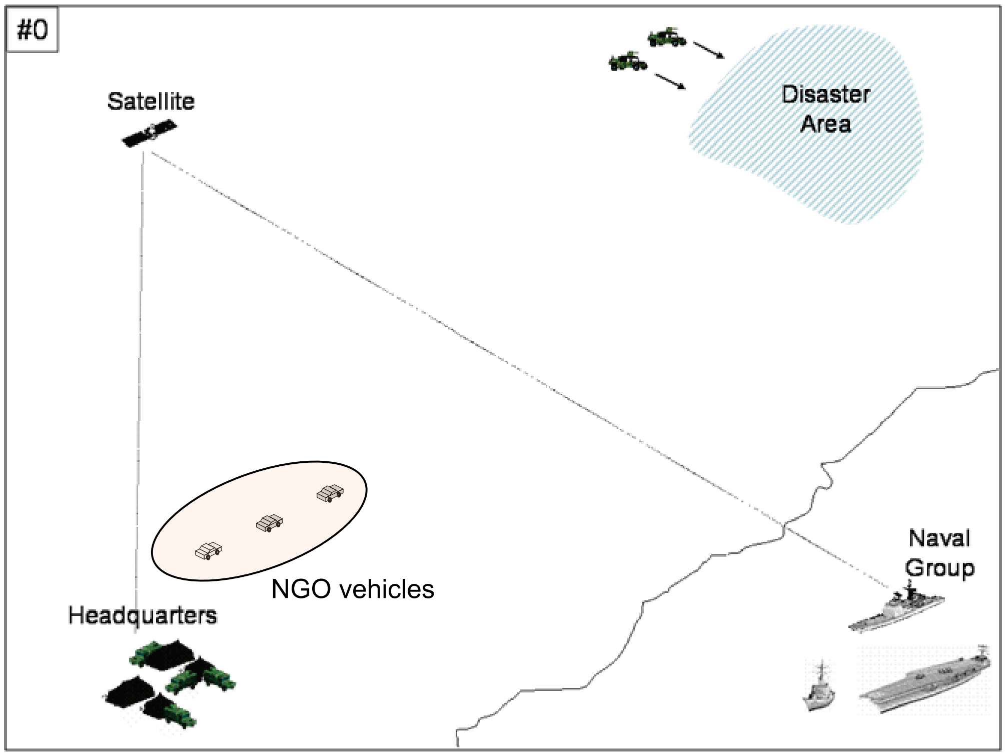


Figure 9: Initial situation of scenario

Phase 1a:
The coalition headquarters decides to escort an NGO convoy to the disaster area. A battalion of coalition forces has been assigned to protect this convoy, to set up a security perimeter to protect the NGOs when building temporary camps to host the displaced population, and to facilitate the NGO activities providing coordination and security. The convoy consists of NGO vehicles and coalition armored cars.
The convoy military vehicles are connected together in an ad hoc manner through a series of short hops. The convoy leaves the HQ.

Phase 1b:
Sometime after the convoy has left the HQ, it is joined by a second convoy of military vehicles. This second convoy is from a different nation and may use different radios internally. Communication between the convoys and between the convoys and the HQ is either done by satellite or by using radios compatible on the air. Some of the vehicles in the different convoys are equipped with these compatible radios, and the remaining vehicles communicate via the links set up by the compatible radios. These radios automatically discover each other.

Phase 2:
In order to protect the convoy, a UAV is launched from the HQ, in order to collect up-to-date intelligence information and to provide live surveillance. The UAV squadron has established a ground station that enables communication with the UAV, typically using CDL (Common Data Link). The ground station forwards the collected

surveillance/intelligence data to the HQ for analysis and storage. The ground station also forwards remote control data to the UAV. The HQ staff analyses collected source data and provides information products on intelligence and surveillance. The operation is supported by an AWACS aircraft that performs tactical air control.

Phase 3:
During the journey, the convoy radio communications are suddenly severely degraded due to jamming. The convoy informs the headquarters through the satellite link that is not affected by the jammer.

Phase 4:
The UAV detects and localizes the communication jammer. It gathers various types of information about the jammer (e.g. snapshots, geographic location) and sends them to the Headquarters. The headquarters send a request for air support to the naval group using an available network link/path. Two aircrafts are launched from the carrier. Their mission is to destroy the jammer located by the UAV. The jammer is destroyed.

Phase 5:
A short while after the jamming starts, the convoy is ambushed. The Coalition Forces convoy is stopped by the detonation of a remote-controlled IED. The blast hit the first armored vehicle. Simultaneously, insurgents equipped with fast moving vehicles open fire on the convoy. In the fire one NGO truck is destroyed and several NGO members are wounded. Coalition forces riposte but their situation is not favorable. The coalition forces inform the headquarters using an available network link/path. Air support from the Naval Group is requested by the HQ.

Phase 6:
The Headquarter requests additional air support to help the convoy. The aircrafts in-flight are re-assigned with a new target. Two more aircrafts are launched from the carrier. The Insurgents are defeated and escape; several of their vehicles are destroyed.

5.3 Experimentation plan

The experimentation architecture is a subset of the system architecture chosen for experimentation purposes. With respect to the scenario, the test and demonstration architecture involves phase 0, phase 1a, phase 1b, phase 3 and phase 4.

The overall challenge is to share information in order to build a COP for common situation awareness. Each of the enclaves (i.e., inside the convoys and the HQs) has services that the other enclaves need to use in order to maintain the COP. The convoys and the HQ are “played” by different nations taking part in the test and demonstration; the convoy consists of a German and a Norwegian part, and from a SOA perspective, also the multinational HQ is provided by Germany and Norway.

Phase 0:
All activity takes place in the HQ (and between HQs at different levels).

In this phase, it is important to check that all necessary services are available and visible to the relevant users before the convoy leaves the HQ. This is done once all systems are online and have registered in their respective service discovery mechanisms,

Demonstration issues - Core Enterprise Services:

- *Service Discovery*:
  - When each service is started, it should register at its local service discovery mechanism, indicating that it is available.
  - Since global multicast is used for announcing services, the two nations will also be able to discover each other's services. This is illustrated in Figure 10.
- *Messaging*: Only used for setting up subscriptions.
- *Publish/Subscribe*: Subscriptions are set up, and the services start to deliver notifications

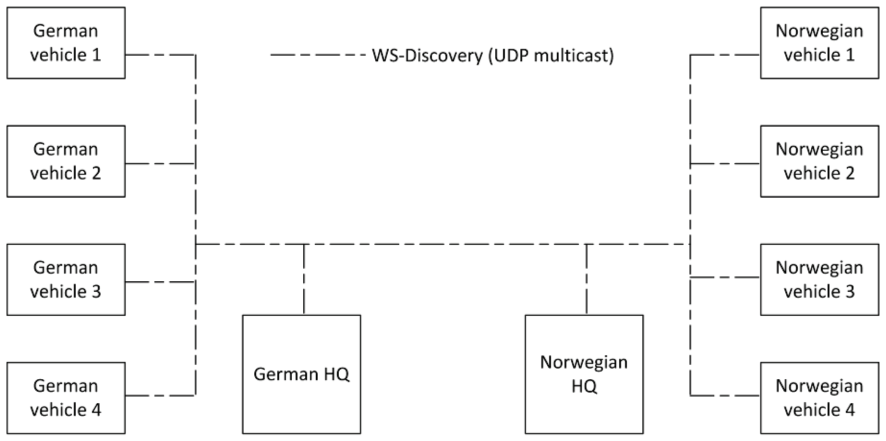


Figure 10: Flow of service discovery information

Phase 1a:

Demonstration issues - Functional Services

- *NFFI service*:
    - *Producers*: When started, these services must register themselves at their local service discovery mechanism. Once subscriptions are started, the services start delivering notifications
    - *Consumer*: Aggregation service that subscribes to NFFI producers. Uses service discovery mechanism to find producers, and then start subscribing to these.

The following subscriptions are now set up (this is also shown in Figure 11):

- Lead vehicle convoy 1 subscribes to position report (NFFI) from each vehicle in convoy 1 (used to produce a COP for convoy 1)
- Lead vehicle convoy 2 subscribes to position report (NFFI) from each convoy vehicle in convoy 2 (used to produce a COP for convoy 2)
- The two parts of the HQ subscribe to the convoy COP (NFFI) the lead vehicle of their respective convoy
- The two parts of the HQ subscribe to each other's convoy COP (NFFI)
- Lead vehicles in each convoy subscribe to full COP from HQ
- Each convoy vehicle subscribes to COP from lead vehicle (within the convoy)

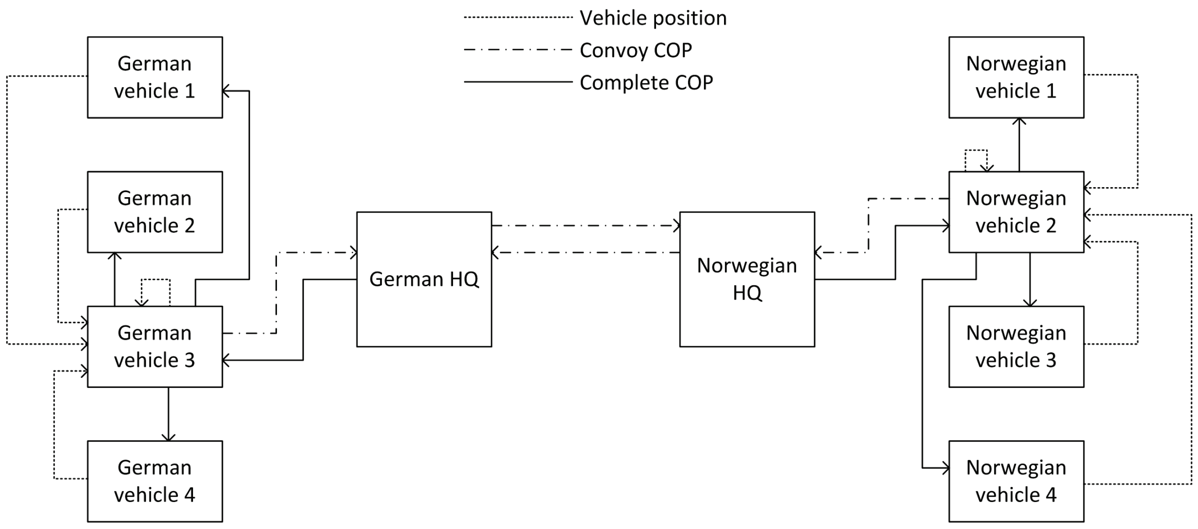


Figure 11: Subscriptions on position updates

Phase 1b:

Initially, the two convoys communicate via the HQ. When the two convoys come within radio range of each other, it is better to communicate directly, in order to reduce traffic over the radio networks. However, this requires that the subscriptions are changed.

Demonstration issues - Functional Services (see Figure 12):

- The COP provider in each of the convoys ceases to subscribe to the full COP from the HQ. Instead, the two COP providers subscribe to each other's convoy COPs and then build the full COP locally
- The two nations in the HQ continue to subscribe to the COP from their respective convoy. In addition, they subscribe to each other's convoy COP, and are thereby able to build full COPs

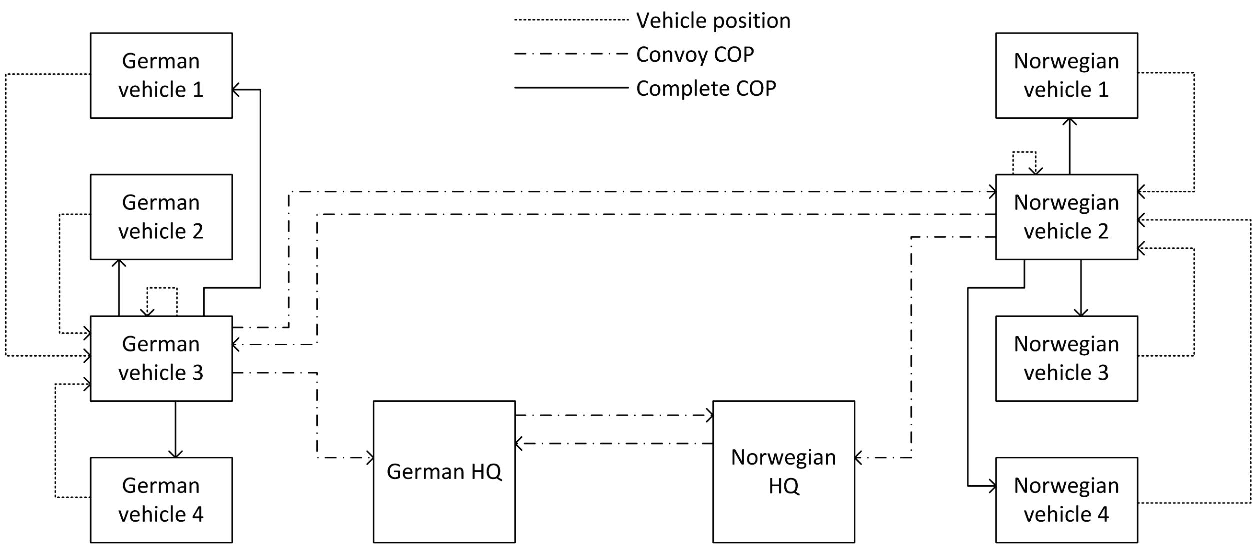


Figure 12: Subscriptions after reconfiguration

Phase 3:

When the jamming there will be a loss of communication in the time span from the jamming starts, and until the tactical routers have established an alternative connection.

We assume that there will be a layer of DSProxys between the network layer and the application layer. The DSProxys will hide the network disruptions from the application layer, and prevent loss of data.

The demonstration issue in this phase is therefore to show that the DSProxy is able to hold the traffic until the tactical router finds an alternative connection between the convoys, and forward it once connection is re-established.

All phases:
In parallel with the NFFI traffic, all nodes communicates using chat, and in addition, messages are sent using the Operational Message Service.

5.4 Networks

The CoNSIS reference model consists of a core network to which user domains are connected via IPsec crypto devices. The core network itself is composed of a number of interworking networks operated by different administrative authorities. The figure below shows the main elements of the CoNSIS architecture (for a more detailed description of the networks, we refer to the CoNSIS Task 1 final report).

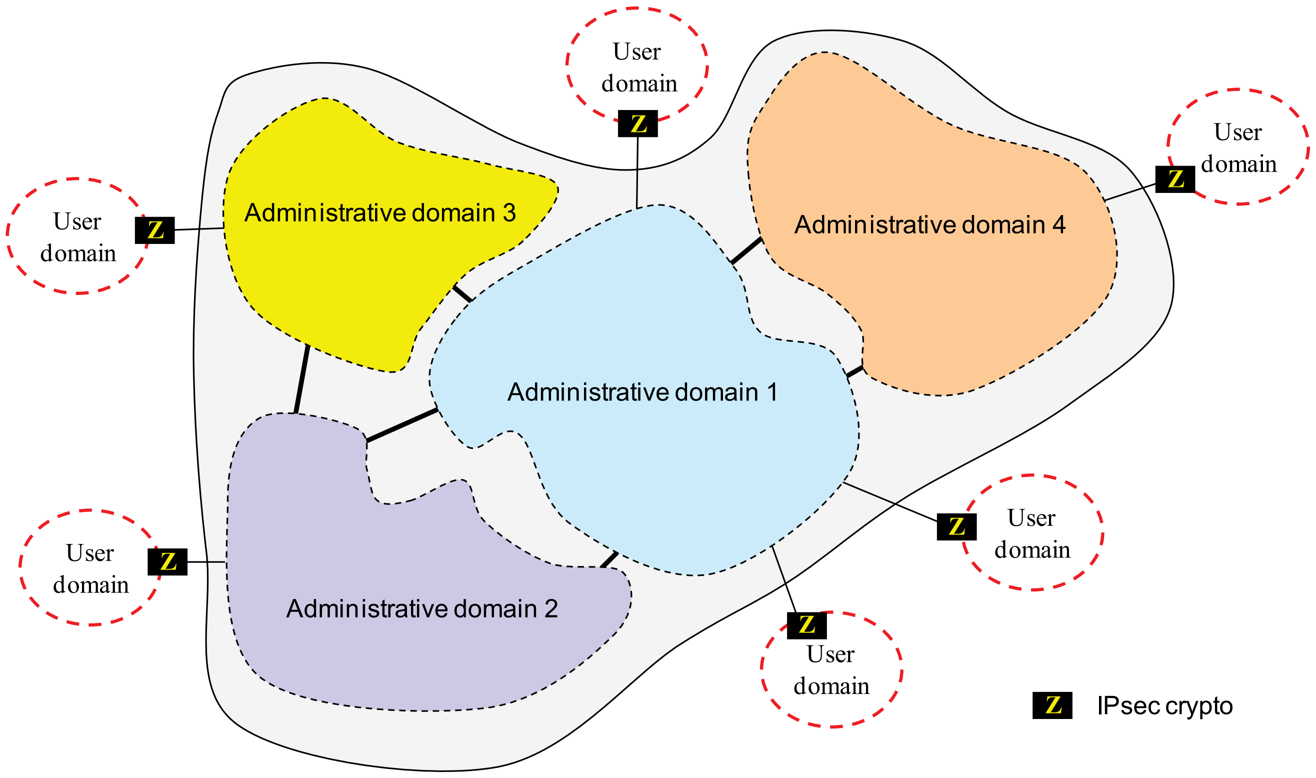


Figure 13: Administrative domains

This architecture is close to that of Protected Core Networking (PCN), in which *Coloured Clouds* (CCs) are connected to a *Protected Core* composed of *Protected Core Segments* (PCSs). However, the two reference models are not identical. In particular, CoNSIS administrative domains are not assumed to have exactly the same functions as PCSs regarding, e.g., security protection or the management of service-level agreements (SLA), and they interwork via interfaces which are not supposed to have the same features as the PCS-1 interface. Likewise, the generic interface between CoNSIS user domains and the core network is not necessarily compliant with the PCS-2 interface.

The CoNSIS land mobile part of the network consists of contributions from Germany and Norway. In addition to four German and four Norwegian mobile nodes located on military vehicles, there was also an NGO vehicle. However, this NGO vehicle was not used in the SOA experiments and will therefore not be further described in this paper. Figure 14 shows the parts of the CoNSIS network that was being using during the SOA experiments.

Two different radio types were used. This is partly because Germany and Norway do not have the same radio systems, but also because different wireless technologies, having different properties with regards to e.g., range, are needed. Together with the wired network at the HQ, the two radio links form a heterogeneous network. Within the German part of the convoy the IABG HiMoNN radio was used, whereas the Kongsberg WM600 radio was used within the Norwegian part of the convoy. To interconnect the two parts one German radio was placed on a Norwegian vehicle and one Norwegian radio was placed on a German vehicle.

In addition to the vehicle nodes, one mobile network node was physically co-located with the deployable HQ network. This node connected the mobile network to the rest of the CoNSIS network, and was connected to mobile nodes terrestrial radio links as indicated.

The eight vehicles formed two convoys, one German and one Norwegian. In the scenario, these two convoys were separated from the start, but at a later point in time, they merged into one large convoy. In addition, the network topology changed during the experiments, something that affected delay and available bandwidth. One such alternative topology is shown in Figure 15.

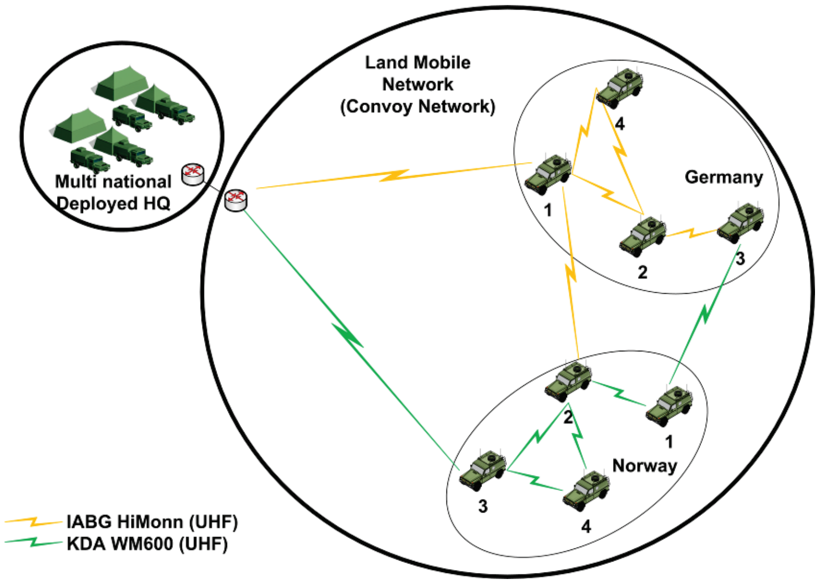

Land Mobile Network (Convoy Network)
Multi national Deployed HQ
Germany
Norway
1
2
3
4
IABG HiMonn (UHF)
KDA WM600 (UHF)

Figure 14: CoNSIS network, initial configuration.

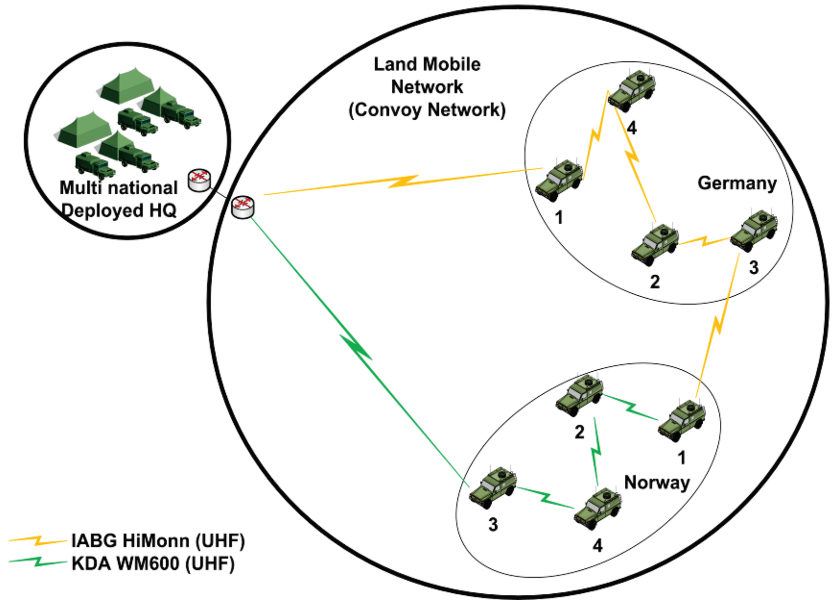

Multi national Deployed HQ
Land Mobile Network (Convoy Network)
Germany
1
2
3
4
Norway
1
2
3
4
IABG HiMonn (UHF)
KDA WM600 (UHF)

Figure 15: CONSIS network, alternative network topology

5.5 Infrastructure

The infrastructure being used within Task 2 relied on a fully transparent IPv6 network, including all mobile nodes and the fixed infrastructure inside and between headquarters. The transparency made the network independent from the real used physical entities like different radio systems and fixed, wired network types.

In all participating nations, the same messaging protocol, SOAP v1.1, was used. Normally, SOAP was operated on top of HTTP/TCP, at least within the fixed installations. Within the mobile domain, Norway used SOAP over HTTP/TCP between the client or service and the local DSProxy, and SOAP over UDP over the radio networks (between the DSProxys).

Germany was using SOAP directly over UDP for independent (national only) tests within the mobile domain, as this combination was considered to be more appropriate to fulfill specific mobility requirements in radio networks. To allow a better recovery from broken network connections, Germany was using a special variant of SOAP over UDP, SOAP over reliable UDP, to allow the usage of larger SOAP body types (beyond 32 kb) over radio networks.

For combined tests Norway/Germany, the DEU nodes were reconfigured to use DSProxys, in order to avoid a profile mix within the convoy.

To reduce the message sizes, especially within radio networks, compression of the SOAP message body was required. Compression is allowed for SOAP-messages in any case, as an available mechanism for SOAP body compression, GZIP was used (both for Germany and Norway, a better compression mechanism was foreseen, using

schema-aware compression algorithms, but the availability of those algorithms was not given during the Greding campaign).
All mobile nodes and the nodes within the deployed headquarter were equipped with WS-Notification brokers. The mobile WS-Notification brokers within the Norwegian domain did not publish their services to the German domain, only the broker at the Norwegian part of the deployed headquarter did. As a consequence, the positions of the Norwegian cars were reported to the German domain via this broker. But this operational behavior had no immediate consequences for the produced operational picture.

Germany also provided a full WS-Security implementation, but this protocol was not supported on the Norwegian side. Consequently, WS-Security was only used in internal German experiments. When operating in a coalition mode, this functionality was turned off.

Besides getting blue force information from the WS-Notification brokers (using a publish/subscribe mode), on the German side it was also possible to contact a single car directly via request /response, by using the local NFFI/SIP3 provider. Using this mode, it was necessary to periodically ask any of the mobile cars for the current position to produce an operational picture. This mode was found to be overloading the operators, as a permanent activity is required from them.

Germany and Norway each provided implementations of selected parts of the NNEC CES, using Web services technology as their foundation. The two implementations were technologically quite different, as the German implementation, Referenzumgebung Dienste (RuDi) [10], is based on an ESB, while the Norwegian solution consisted of multiple stand-alone components implementing the set of core services.

In the experiment, the core services were used to provide an infrastructure for functional services providing three types of information:

- *Vehicle positions*: Each vehicle reported its position (through a GPS-based positioning service) to the lead vehicle of its convoy. The lead vehicle then aggregated the positions of all convoy vehicles into a convoy Common Operational Picture (COP), which in turn was delivered to the HQ, where the two convoy COPs were aggregated into a full COP. This full COP was then distributed back to the vehicles.
- *Operational messages*: This is a service for sending messages to other users. The messages can be of the types Alert, Warning, Information, and Command. File attachments are also possible.
- *Chat*: This service provides ordinary chat functionality, based on chat rooms.

All information types were distributed using WS-Notification, which is the Web service standard for publish/subscribe that is specified by the CES Working group. Furthermore, all information types were distributed among all the vehicles as well as the HQ.

5.5.1 German Infrastructure

The German infrastructure used during the Greding experimentation was the full implementation of the Referenzumgebung Dienste (RuDi). The functional bandwidth of this implementation was broader than the Norwegian one with the consequence, that portions of this implementation were not used. The overall service environment was based on:

- RuDi service registry (a content-based management service)
- WS-Discovery (Norwegian implementation)
- WS-Notification 1.3 / WS-brokeredNotification 1.3 (no fully implementation, as still experimental)
- DSProxy (Norwegian implementation, used for SOAP messages to/from the Norwegian domain)
- WAPProxy (used only in the German infrastructure, as the Norwegian infrastructure used UDP via the DSProxy)
- Services not used:
- WS* - Security (authentication, security token service, encryption, signature, XKMS)
- SOA PKI (administration of exchange of service provider X.509-certificates)
- GenKey (Generation of key pairs and X.509-certificates directly used by service providers)
- DomCtrl (domain control service for supervision of service provider certificates)
- SyncD (ad-hoc Synchronization of service registries, operated peer-to-peer, based on TIBER implementation)

For disruption tolerant behavior, the German infrastructure was supporting two methods: DSProxy and WAPProxy. In the coalition experiments, it was decided to use only DSProxy, as WAPProxy was not available at the Norwegían nodes.

5.5.2 Norwegian Infrastucture

The Norwegian infrastructure used during the CoNSIS experiments consists of a number of internally developed components. The core of the infrastructure is an implementation of the core features of WS-BaseNotification and WS-BrokeredNotification. This Java implementation has support for subscribing and unsubscribing, as well as for receiving and sending notifications. While not being a full implementation of the WS-Notification standard, this light-weight solution is well suited for use in test environments like the CoNSIS network. During the experiments all the Norwegian units were running a notification broker, and all clients and services on a node subscribed to, respectively delivered notifications to, its local broker. Between nodes, the brokers subscribed to each other.

Web services, including WS-Notification, normally relies on HTTP over TCP for message delivery, meaning that it is necessary to establish an end-to-end TCP connection between client and service. In the CoNSIS network this means that TCP connections in many cases would have to be established across several radio networks with unstable links and often very high delays. To enable standards-based Web services over such connections, we used the Delay and Disruption Tolerant SOAP Proxy (DSProxy) [16], developed at FFI.

While compression enables the use of XML and SOAP in disadvantaged grids, frequent delays and disruptions make end-to-end connections difficult to initiate and maintain. Connection-oriented transport protocols such as TCP break down when the delays get too long, or when disruptions occur. In practice, TCP is non-functional in many wireless tactical environments [15]. Standard Web services and publish/subscribe-based schemes such as WS-Notification rely on TCP-based end-to-end connections for communications between consumers, producers and brokers.

The DSProxy is designed to extend the reach of Web services from networks with infrastructure into the tactical domain [16]. It is a proxy-based network overlay enabling robust Web services across heterogeneous networks. By presenting a standard HTTP/TCP interface to the applications, COTS Web services as well as WS-notification brokers and consumers are able to send and receive SOAP messages through disadvantaged grids. Between two DSProxy nodes, several different protocols can be used, but in the CoNSIS experiments, we used UDP, with added functionality for reliable communication.

Even though the DSProxy already enables Web services and XML/SOAP-based publish/subscribe schemes in disadvantaged grids, bandwidth scarcity is still a factor and may become prohibitive when using a publish/subscribe system that produces many small notification messages frequently. To improve the compression ratios for small messages without compromising liveness/latency, the DSProxy has been devised with a diff-based compression scheme. Because DSProxy nodes can be deployed throughout the network, reliable exchange of diff-based messages can be performed between any two DSProxy nodes.

In order to achieve favorable compression ratios on small messages, instead of aggregating many small messages from multiple sources over a period of time, one can create a virtual aggregate message-based on data previously exchanged with other nodes. By keeping track of previously exchanged messages (a local cache), and keeping track of which nodes they are exchanged with, a virtual message can be constructed by the sender, comprising one or more previously exchanged messages.

5.6 Functional services

In addition to the infrastructure software, both Germany and Norway had functional services that utilized the infrastructure. Most of these were prototype services and applications that were developed for the CoNSIS experiments.

5.6.1 German software

Each car was equipped with a GPS sensor, periodically generating the position of the car. This information was used locally by a GPS Service, which was an input service for the final NFFI/SIP2 service. Each car had a notification broker, and notifications were exchanged between brokers, both within the single national domain and within the coalition domain. For the communication with the local NFFI service within cars, a local notification broker was requesting the NFFI messages directly.

The (local) consumption of NFFI messages was done by an own COP service per node. Within the German domain this service was independent from the equivalent service within the Norwegian domain, and it was commonly used both in mobile nodes as well within the German part of the deployed headquarter.

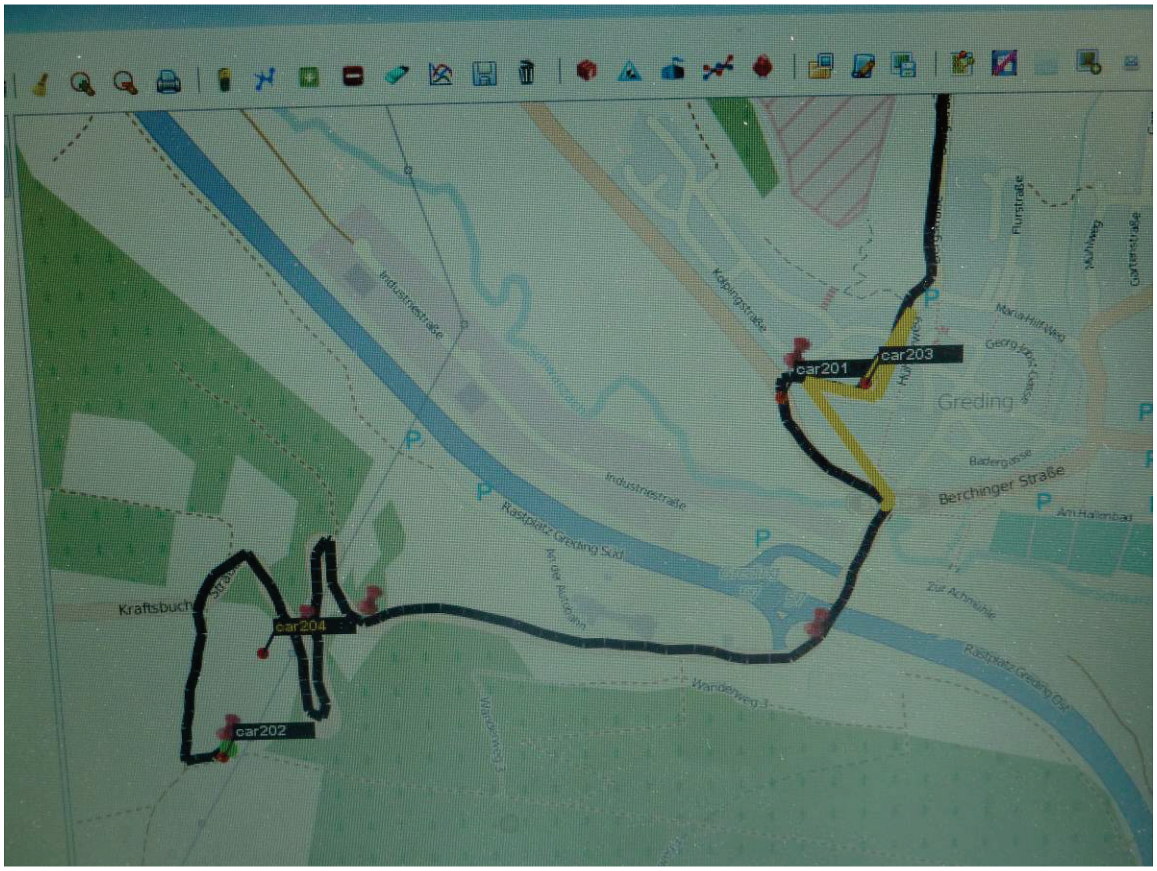


Figure 16: The German COP viewer

Within the German cars, the following application services were used:

- GPS service
- NFFI/SIP3 service
- Operational message service
- COP service (GeoDta Provider and GeoDta viewer)

5.6.2 Norwegian software

Each vehicle had a GPS component that reads the vehicles position from a GPS, creates an NFFI message, and delivers this as a notification to the local broker. Furthermore, there was a component for creating Operational Messages, and delivering these as notifications to the local broker. There was also a chat component, which both subscribed to, and delivered notifications to, the local broker. Next, there was an aggregator function that subscribed to the position of each vehicle, and then combined all vehicle positions into one NFFI message which was then delivered to the local broker as a notification (COP service). Finally, there was a viewer application, which subscribed to NFFI tracks and Operational Messages from its local broker and displayed these on a map (see Figure 17).

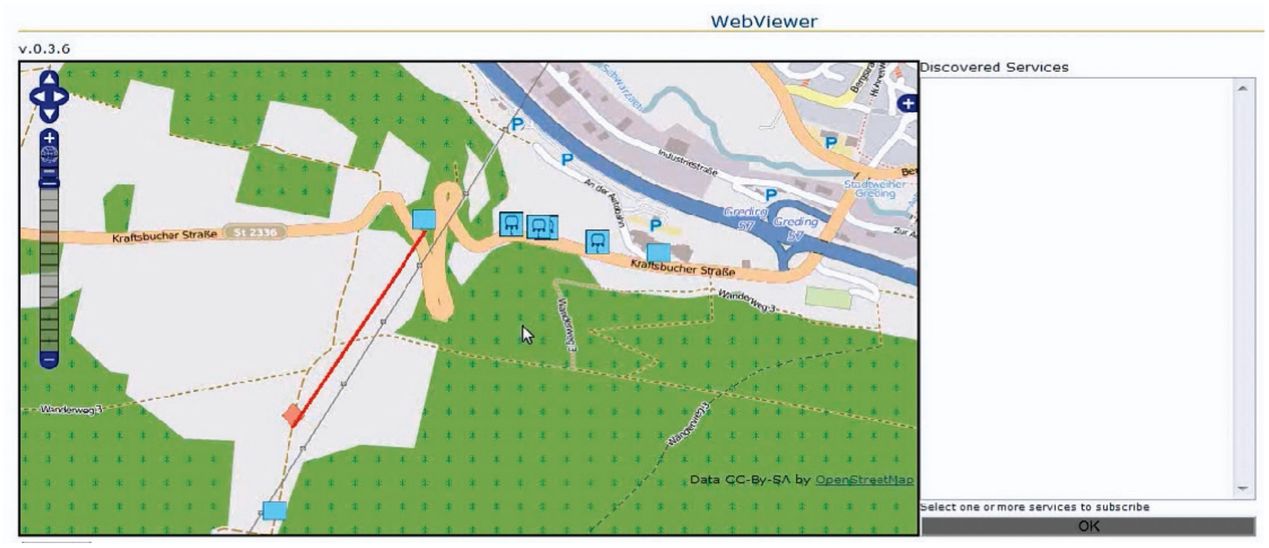


Figure 17: The viewer component

5.7 Experiments in cooperation with other tasks

Two experiments were carried out in cooperation with Task 4. One, MA-SOA, involved providing a SOA wrapper for the management framework PerfSONAR, the other, Jammer-Notification, using the Operational Message Service to alert HQ of the detection of a possible jammer attack. Details for both these experiments can be found in the Task 4 report [17], sections 3.2.1 and 3.2.2 respectively.

5.8 Execution of experiment

The operation of a SOA-based infrastructure was primarily depending on the availability of a transparent network service. Therefore tests in Task 2 depended on the network being made available by Task 1.

Initially, the tests were started in two separated national domains. These first separated tests were oriented towards generating a connectivity picture in all involved German and Norwegian cars.
Figure 18 shows the result of a network interruption (from the point of view of the deployed HQ), when a car was losing the network connectivity when driving through Greding (no line of sight connection between a single car and the HQ).

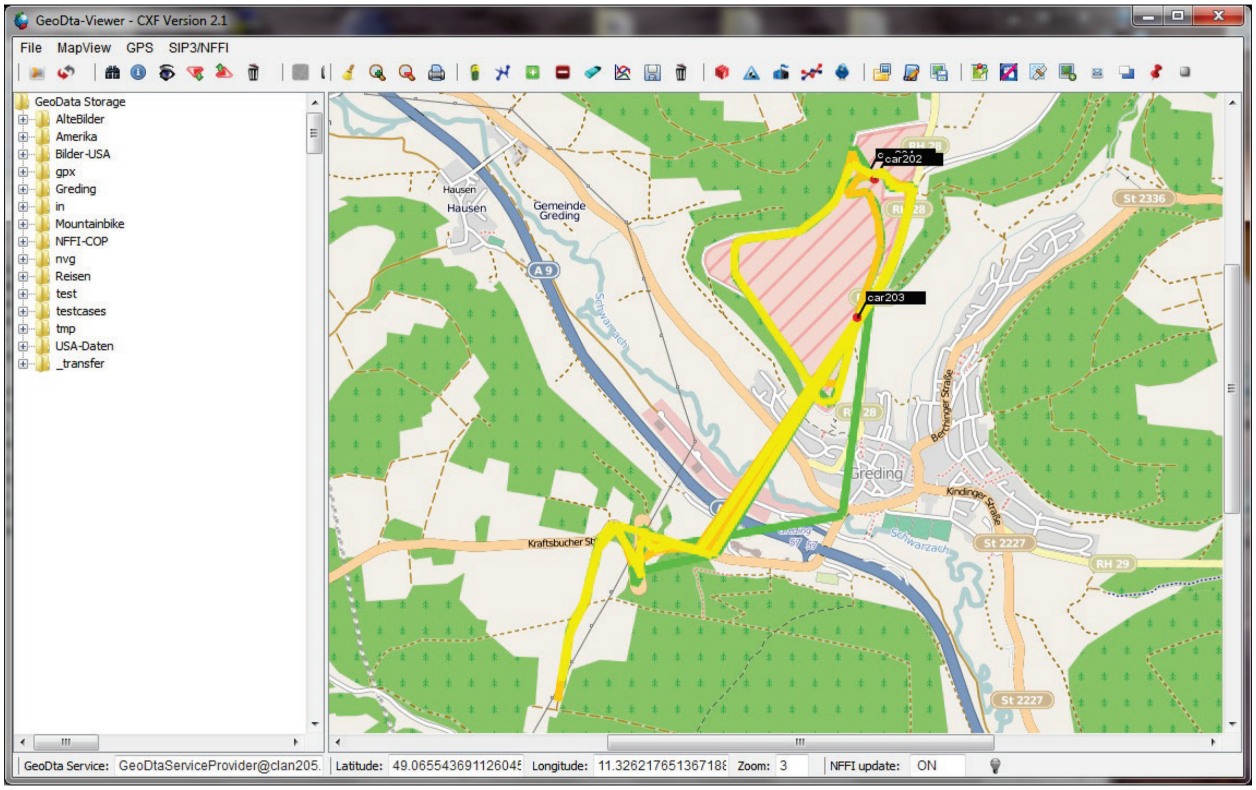


Figure 18: Difference between physical and logical tracks based on network disconnection

Both the yellow track and the green track shows that between the position of car203 (German car) and the next position (either south of Greding and on the opposite side of highway A9) no network connectivity was given.

To overcome the problem, the radio connectivity of other cars (e.g. within a convoy) was used, as shown in the following picture:

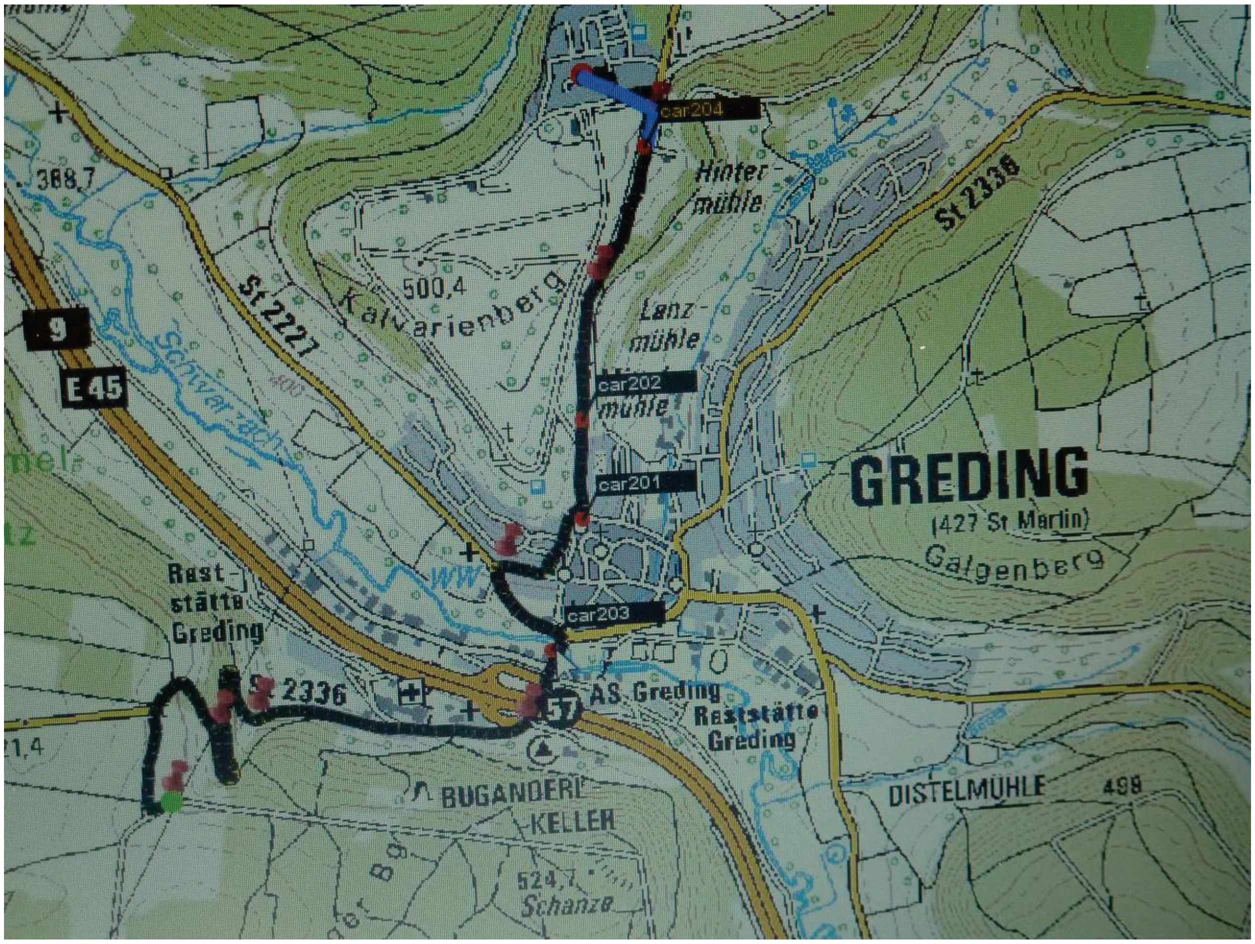


Figure 19: Used of node relays to report own positions

In Figure 19, car201 was using car202 and car204 as relays to report the own position back to the deployed headquarter.

After having the connectivity positively tested within the national convoy segment, the coalition convoy was tested. In this case, the idea was, that radios of the partner were used in case that the own radios have not supported the network connectivity.

The results were as expected, as this was only a network function, and generation and consumption of the NFFI messages was still national oriented.

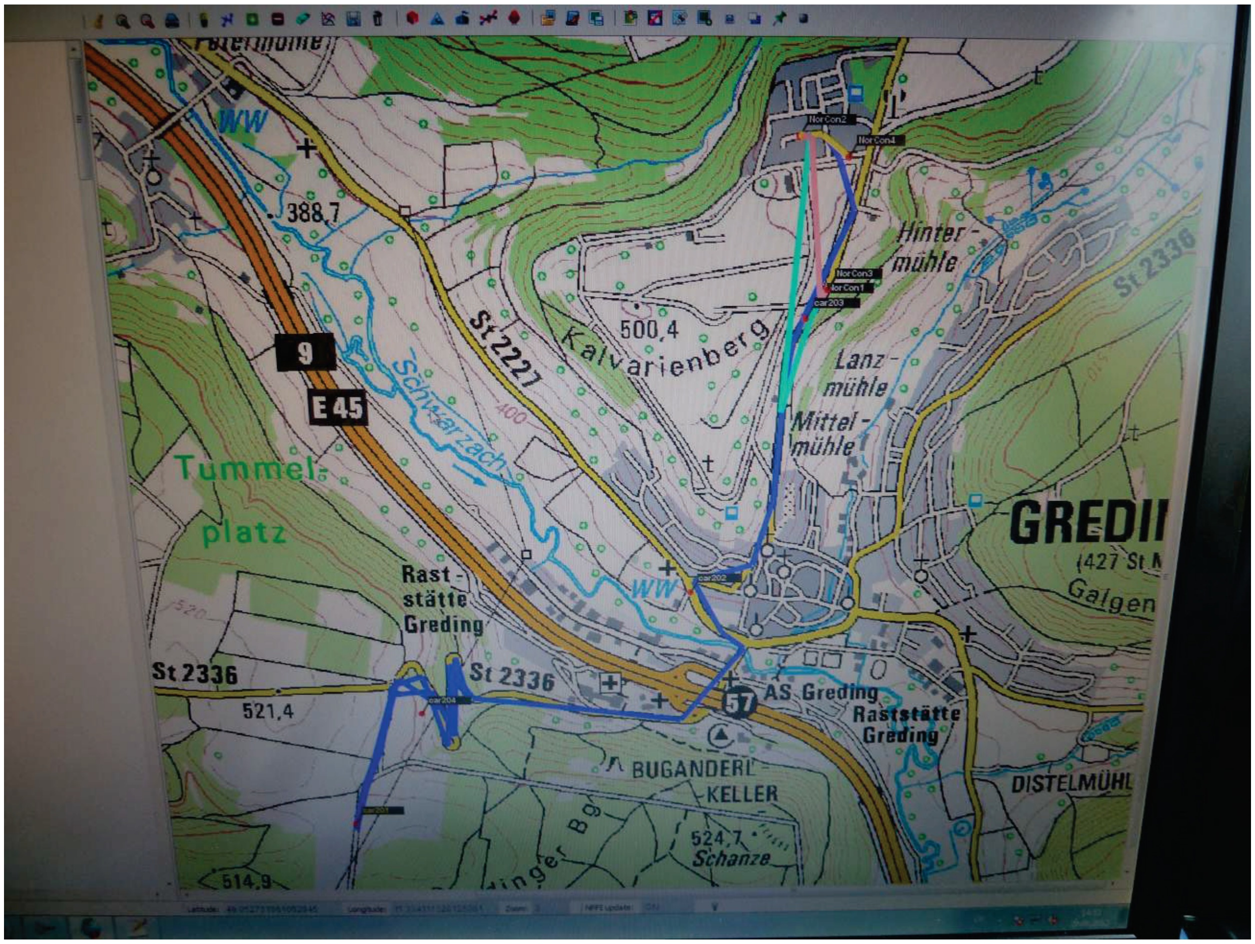


Figure 20: Combined convoy with better network connectivity

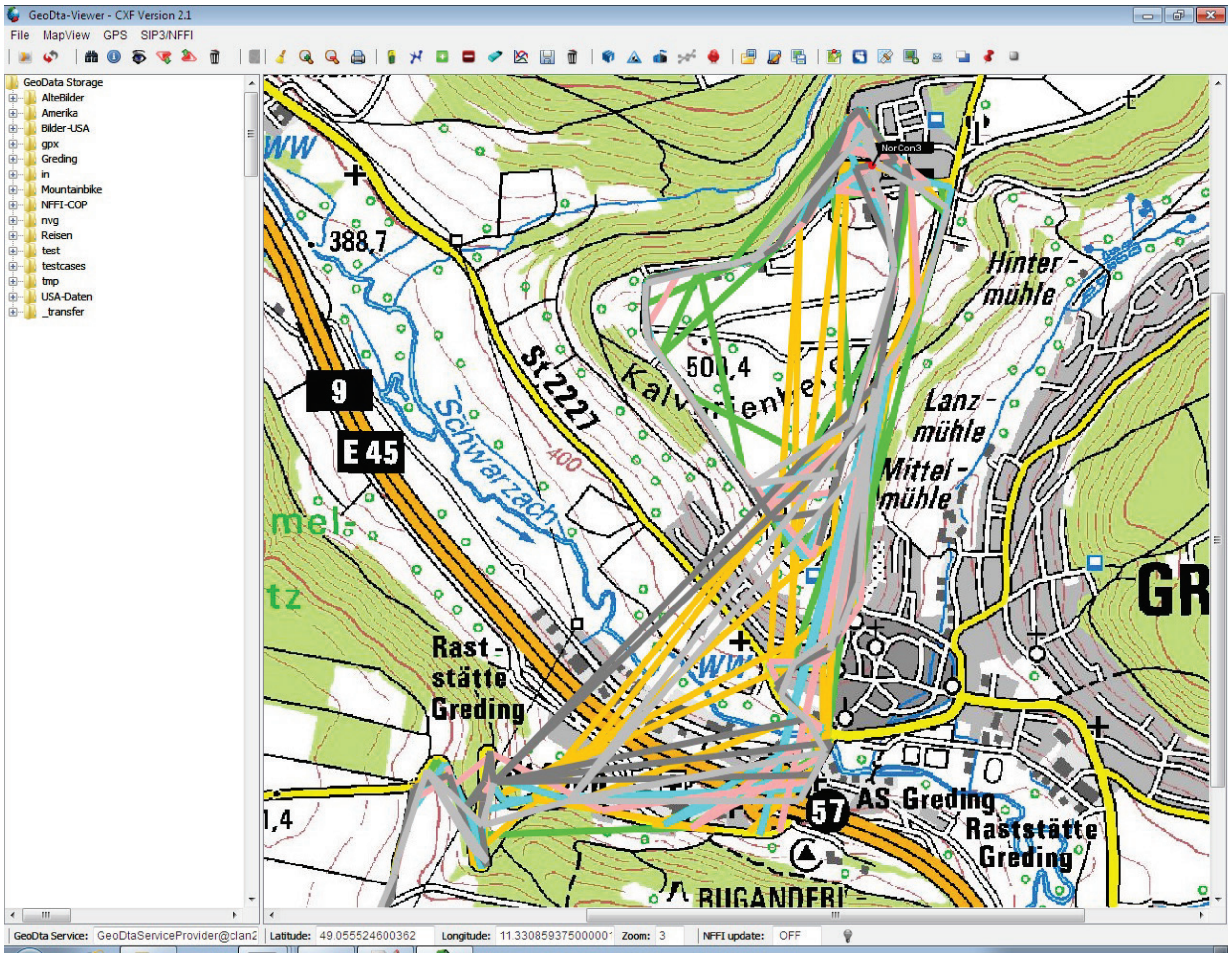
Figure 20 shows that in the combined convoy, based on 8 cars, the network connectivity to the HQ was supported by more than one car (in detail, car203 and Nor Con 1), which both reported (and received) their NFFI messages back to the HQ (both the German and the Norwegian part).

5.9 Experiment results

In the previous section, the basic tests with exchanging primarily NFFI/SIP3 messages between some or all cars and the HQ were described. For the communication between all nodes and the HQ both chat and voice was used, to control the movement of the convoy. Unfortunately, the behavior of some radios influenced the display and control of the convoy tracks.

While Germany was using a radio type without a significant buffer (HiMoNN), Norway was using a buffered behavior, based on two different mechanisms inside WM600 and the DSProxy. The Norwegian consideration was to store available messages up to a positive transmission to the next node. As network disruptions started occurring (several seconds, up to 1 minute), the number of temporarily buffered messages was growing.

After re-initialization of the network connection the buffers tried to forward the stored messages with the result, that a great number of messages were exchanged at the same time, not displaying the current position of cars. The following figure shows the effect of late incoming NFFI messages on the German viewer.



GeoDta Service: GeoDtaServiceProvider@clan2 | Latitude: 49.055524600362 | Longitude: 11.33085937500001 | Zoom: 3 | NFFI update: OFF

Figure 21: Effect of delayed NFFI messages at the German HQ

Figure 21 shows the display on the German viewer, resulting from severely delayed messages. This is a natural effect of pro-actively produced NFFI messages and the reactive behavior of buffers within the complete system.

In coalitions, were different equipment configurations like these may occur, a strategy is necessary to avoid the above behavior. The approach used in Greding, was to delete those messages that had a considerable difference between actual time and the time the message was generated (the time stamp of the message). This can be used in principle in any scenario (at least when the information is based on NFFI), as GPS time always is correct.

Another approach would be to synchronize the elements at the network level to avoid that pro-active network elements (like routing) are colliding with re-active elements (like buffers).
In addition, if buffering is requested as an urgent mean, an active buffering service (e.g. notification cache) is required, which collect messages temporarily at brokers and forwards them only on request of CCOP consumers.

Yet another alternative is to let the consumer inspect each track when receiving it. If a received track already exists in the track store of the consumer (that is, it has the same Track ID as the track received) the time stamp of the newly arrived track is compared to the time stamp of the track that is already displayed. The position of the displayed track is then only updated if the newly arrived update has a newer time stamp. This is the approach that was used in the Norwegian viewer.

It should also be noted that how this problem is solved is dependent on the usage of the tracks. In the Norwegian viewer, it was acceptable to discard older positions because we were only interested in the last known positions of the car. In other applications, however, it may be necessary to store all received track updates, for instance for logging purposes. In such cases, each track can have its updates sorted based on time stamps, in order to reconstruct the route.

6 CONCLUSIONS

Web services are designed to be highly interoperable, even across implementations, platforms, programming languages and system boundaries. However, even if the standards are designed to enable interoperability, achieving this in practice requires all participants to have a common understanding of which standards to use for what task, and how to implement, configure and utilize the various implementations of these standards.

In CoNSIS Task 2, we have looked at how to achieve interoperability in practice, by addressing challenges that arise due to limited functionality in standards, ambiguities in standards, and resource limitations in radio-based networks. We first investigated the interoperability challenges of Web services, and determined which standards, standard options and configurations to utilize. In addition, we identified which overhead-reducing optimizations were needed in order to make Web services perform satisfactorily in the type of radio-based networks we were planning to use during our experiments.

The experiments themselves proved that Web services can be utilized as an enabler for interoperability in radio networks, but it also allowed us to identify a number of technical lessons learned, and also some challenges which should be addressed in greater detail in the future. Below we present an overview of the major technical challenges identified during the work in CoNSIS Task 2

6.1 WS-Notification

Using an information distribution mechanism based on the publish/subscribe paradigm can be a basis for improved bandwidth usage, and lower communication overhead. This is because it reduces the number of messages exchanged between consumer and provider, and it also enables further optimizations because identical information is distributed to multiple recipients at the same time.

WS-Notification, the leading standard for publish/subscribe Web services, does limit the amount of messages exchanged between information providers and consumers. It does however not optimize the information distribution of notifications when these are sent to multiple concurrent receivers. By adding support for multicast distribution to WS-Notification, one can further reduce the network resource consumption caused by notification dissemination. However, this will require further work on how to handle reliability and fragmentation issues.

In addition to the multicast optimization described above, we identified a need for some additional functionality in WS-Notification, namely notification caching and subscription status checking. Both these functionality would help limit the impact of potential network disruptions, and are thus particularly useful in radio-based networks.

Notification caching would enable any existing notification brokers to temporarily store a local copy of the notifications it distributes to consumers. By having this local copy, previously distributed information can be made available to late arriving consumers through a specifically designed interface. In addition, consumers that fail to receive certain messages can use this interface to request messages they may have lost.

WS-Notification brokers (or providers) keep a list of the consumers interested in the information they provide in the form of subscriptions. These subscriptions determine which consumers receive which messages, and proper management of these are thus vital to ensure proper functionality of the publish/subscribe framework. Consumers can make, update and delete subscriptions, but they have no way of querying the subscription manager in order to find out if they still have a valid subscription. The consumer can of course just assume that it does not have a subscription, and make one when needed. The downside of doing this is that one risks getting duplicate subscriptions, which would lead to duplicate notifications being sent to the consumer. In order to avoid this, having the ability to query the notification source to determine if a subscription exists would be useful.

6.2 Topics

In [18] it is described how WS-Discovery can be used to distribute information about which topics a service covers, while at the same time remaining backwards compatible with the WS-Discovery standard. As a preparation for the CoNSIS experiment, initial testing with topic discovery was performed at the Coalition Warrior Interoperability Experiment (CWIX), where WS-Discovery with topic support was tested by multiple partners. During this initial testing, as well as during the CoNSIS experiment execution, we discovered that while this approach provides enough information for nodes to be able to distinguish between the content offered by the different providers, certain extra functionality is desirable.

In particular for notification brokers (as described in the previous section), which can serve many nodes and offer information on many different topics, it would be very useful to be able to query the broker itself about topics: For instance, which topics a broker currently provides notifications on, which topics it knows about (i.e., has seen at some point), and if and when it has last seen notifications on a given topic.

One challenge when working with topic based information exchange is that it requires all the involved parties to have prior knowledge about how topics are organized. In order for an information consumer to get the information it desires, it needs to know in advance which topic to request from the broker. In the CoNSIS experiment we were working with two partners, making a priori distribution of topic information possible. We decided that a client normally needs information following a given schema, and we therefore chose to have a 1:1 relationship between root topic and the XML Schema of the information in question. Thus, we had the root topics “nffi”, “OpMsg” and “Chat”. However, in other contexts, other classifications may be better suited.

In general, for larger scale implementations of topics, it is necessary to utilize a common information model that describes how topics are organized. This means that NATO should be the driving force behind such a model, which would then be used by all member nations.

6.3 Service Discovery

As mentioned above, allowing the multicast packets from WS-Discovery to traverse routers is not an ideal solution. An alternative is to combine the managed and ad hoc modes in one deployment. When a WS-Discovery proxy announces its presence, all other nodes are asked to enter managed mode, relying on the proxy for service discovery. However, the WS-Discovery specification does not require the nodes to change to managed mode, and by allowing the majority of nodes to remain in ad hoc mode and at the same time keep a link local message scope, one can secure local service discovery without the risk of generating unneeded network traffic in other networks. Combined with discovery proxies that function as relays between the networks, cross-network discovery can be achieved as well.

Note that, even though the WS-Discovery specification does allow nodes to choose not to enter managed mode when receiving a message telling it to do so, it does not clearly state what the expected behavior of nodes is, once the network consists of nodes in both modes simultaneously. This combination of modes is desirable when working with multiple interconnected mobile networks, and therefore a profile of how to use the WS-Discovery standard in this context should be developed by NATO for interoperability between nations.

References

[1] "Reference Model for Service Oriented Architecture 1.0", OASIS, Oct. 2006, http://docs.oasis-open.org/soa-rm/v1.0/soa-rm.pdf

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

[3] "Web services glossary W3C working group note 11 February 2004", Hugo Haas and Allen Brown (eds.), http://www.w3.org/TR/

[4] "Using Web Services to Realize Service Oriented Architecture in Military Communication Networks", K. Lund, A. Eggen, D. Hadzic, T. Hafsøe, and F. T. Johnsen, In IEEE Communications Magazine, October 2007

[5] "Delay and disruption tolerant web services for heterogeneous networks," E. Skjervold, T. Hafsøe, F. T. Johnsen, and K. Lund, In IEEE Military Communications Conference (MILCOM 2009), Boston, MA, USA, 2009

[6] "Adapting WS-Discovery for use in tactical networks", F.T. Johnsen, T. Hafsøe-Bloebaum, ICCRTS, Québec, Canada, June 2011

[7] "Web Services Dynamic Discovery (WS-Discovery)", OASIS, July 2009, http://docs.oasis-open.org/ws-dd/ns/discovery/2009/01

[8] "Core Enterprise Services Standards Recommendations: The Service Oriented Architecture (SOA) Baseline Profile Version 1.7", NATO Consultation, Command and Control Board (C3B), 11 November 2011

[9] "OASIS Web Services Notification (WSN), OASIS", Oct. 2006, https://www.oasis-open.org/committees/wsn/

[10] "R&D Service Reference Environment (SRE), High Level Concept, Version 400", Franke. M., Seifert, H., IT-AmtBw, 20.09.2010 (English title for Grobkonzept Referenzumgebung Dienste (RuDi)

[11] "Protocol for the Generic Advertisement of Applications in Coalition Networks", M. Fey, CoNSIS/DEU/Task2/DL/002 rev.1, 24.03.10

[12] "Synchronization Service (SyncD)", M. Steeg, M. Fey, CoNSIS/DEU/Task2/DL/001, 27.05.10

[13] "Multi-Topology Routing – QoS Functionality and Results from Field Experiment," Document CoNSIS/NOR/Task1/DU/002, November 2012

[14] CoNSIS main report, Nov. 2012

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

[16] “Robust Web Services in Heterogeneous Military Networks”, Lund, K., Skjervold, E., Johnsen, F. T., Hafsøe, T., Eggen, A., IEEE Communications Magazine, Vol. 48, No. 10, October 2010, pp. 78-83

[17] “CoNSIS Final Report – Task 4,” Document CoNSIS/Task4/D/003, November 2012

[18] “Topic Discovery for Publish/Subscribe Web Services”, F.T. Johnsen, T. Hafsøe-Bloebaum, Wireless Communications and Mobile Computing Conference (IWCMC), Cyprus, Aug. 2012