Robust Web Services in Heterogeneous Military Networks

Ketil Lund, Espen Skjervold, Frank T. Johnsen, Trude Hafsøe, Anders Eggen
Norwegian Defence Research Establishment (FFI)
{ketil.lund,espen.skjervold,frank-trethan.johnsen,trude.hafsoe,anders.eggen}@ffi.no

Abstract

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.

In this paper, we explain how the communication protocols normally used in Web services are unsuited for disadvantaged and heterogeneous networks. We then present our prototype proxy, which enables the use of standard, unmodified Web services across all network types including tactical networks with low data rate and frequent disruptions. It is designed to work with existing security mechanisms, and also offers further optimizations in the form of optional plug-ins.

1. Introduction

NATO and the member nations are migrating towards NATO Network Enabled Capability (NNEC). The NATO policy is to adopt civil standards as far as possible in order to be able to procure commercial off-the-shelf (COTS) products. International standards are important for the members to agree on common solutions and for the industry to compete under equal conditions. Two important recommendations made in the NNEC Feasibility Study (NNEC FS) [1] are that the information infrastructure should be implemented as a service-oriented architecture (SOA), and that IP should be used as a common network protocol in all network types. A SOA is most commonly realized through Web services, using XML formatted documents. This technology is defined in a number of standards, and while this standardization work is still ongoing, Web services remain the most adopted SOA implementation technology. By providing a standardized way of describing interfaces and data formats, Web services enable loose coupling of information consumers and producers, which means that each nation is free to implement clients and services according to its own requirements and preferences. If the military requirements cannot be met by civil standards, specialized military solutions must be developed and issued as NATO standardized agreements (STANAGs).

It is not realistic to expect all NATO nations to adopt the same systems, and there will always be legacy systems that must be integrated and need to co-exist with

Dette er en postprint-versjon/This is a postprint version.
DOI til publisert versjon/DOI to published version: 10.1109/MCOM.2010.5594680

contemporary systems. Therefore, the NATO nations must instead agree on standardized descriptions of interfaces and data formats, and leave it to each nation to implement the interfaces according to the specifications. By implementing such clients and services using Web services, and according to the agreed standards and formats, interoperability between nations is ensured. In addition, the use of Web services means that using COTS software in many cases is a viable solution, contributing to reduced cost and development time.

However, Web services use XML, and the XML documents tend to be large, causing significant overhead. This represents a problem when trying to extend Web services into tactical networks. In addition, the transport protocol normally used in Web services implementations, TCP, is not suited for networks characterized by high delay and frequent disruptions.[1] This is of particular importance when considering users in the field who may only communicate with others over *disadvantaged grids*, i.e., tactical communication systems with low data rate, high delay, and frequent disruptions.

2. Web Services in Heterogeneous Networks

In Web services, all communication is based on sending XML-based SOAP messages.[2] A SOAP message is an "envelope" consisting of a header and a body. The header contains information related to the handling of the message, such as addressing and security information, while the body contains the application data. In regular Web services, SOAP messages are transmitted using the HTTP protocol, which in turn uses the TCP protocol for reliable transfer of the messages. This protocol set is not suited for use in disadvantaged grids, and the main question is how to enable the use of Web services in disadvantaged networks and across heterogeneous networks. Through our research on Web services in disadvantaged grids, we have found that this question can be broken down into three requirements that must be met:

1. Reduce the network traffic generated by Web services
2. Remove the dependency on end-to-end connections
3. Hide network heterogeneity

2.1 Addressing Web Services Overhead

The first problem is related to the amount of network traffic generated by Web services. It is necessary to reduce both the size of the individual messages, and the number of messages being transmitted. XML is a rather verbose language, and tends to produce much larger messages than binary formats do. Using techniques

[1] There exist optimizations for TCP, such as TCP Reno and TCP Vegas, that might alleviate the delay problem. However, these do not handle disruptions. For a comparison of TCP varieties, see [2].

[2] One exception to this is REST Web services, which do not rely on SOAP. However, SOAP is a better solution with respect to maintaining security, which is why we focus on SOAP Web services in this paper.

Dette er en postprint-versjon/This is a postprint version.
DOI til publisert versjon/DOI to published version: 10.1109/MCOM.2010.5594680

such as compression will reduce the size, and thus the bandwidth requirements of each individual message, but will not reduce the number of messages sent between nodes. In our work, we have looked at several ways of limiting the number of messages: 1) employing caching near the clients, which allows for reuse of older messages; 2) using the publish/subscribe paradigm, where clients subscribe to information instead of requesting it, allowing the same message to be sent to multiple clients; and 3) employing content filtering to ensure that only relevant data is transmitted.

2.2 End-To-End Connections

The second issue is that regular Web services depend on a direct, end-to-end connection between the client and the service. TCP is connection-oriented, and designed for wired networks, which means that the control mechanisms are designed for handling congestion, and much less for handling errors. In tactical networks with high error rates and high latencies, the congestion control of TCP will therefore cause sub-optimal utilization of the network due to frequent connection timeouts. When multiple networks are interconnected (see Figure 1), TCP's need for establishing an end-to-end connection increases this problem; each traversed network adds delay, increasing the risk of connection timeout. Similarly, HTTP is synchronous, which means that when a SOAP request is sent, the HTTP connection is kept open until the SOAP response is returned in the HTTP acknowledgement message. If the connection times out the SOAP response cannot be routed back to the service consumer.

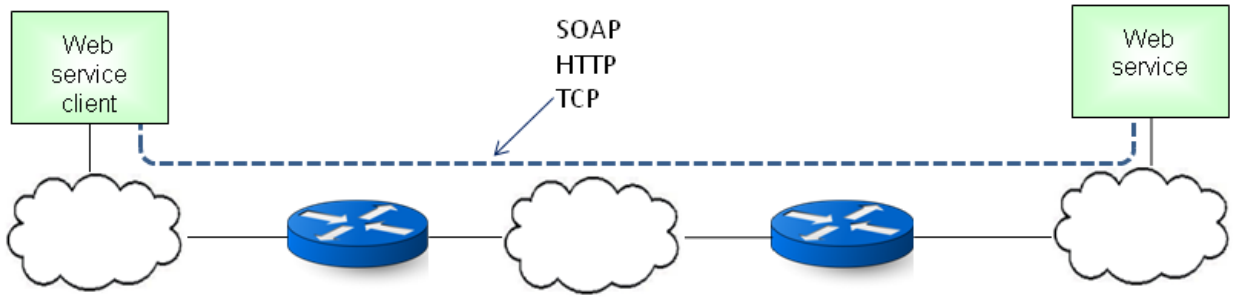

Figure 1: HTTP and TCP establish end-to-end connections

The obvious solution to this problem is to replace HTTP and TCP with other, more suitable protocols. However, this requires modifying the application software. Alternatively, an extra communication layer can be introduced. Within this layer, hidden from the applications, more suitable protocols can be used (e.g., tactical protocols such as STANAG 4406 Annex C & E), that are able to withstand long and variable round trip times, and have little communication overhead.

By implementing this extra communication layer in a *proxy* solution, standards compliance can be retained. A proxy is a node in the network between a client and a server through which the network traffic passes. A proxy can be used for several

Dette er en postprint-versjon/This is a postprint version.
DOI til publisert versjon/DOI to published version: 10.1109/MCOM.2010.5594680

purposes, such as caching, firewalling and content adaptation. For example, HTTP proxies have been popular on the Internet for years, since they lower response times when surfing the WWW. Web services proxies follow the same principle as HTTP proxies, in that they function as a "middle man" between the provider and the consumer of the service. However, they do not just understand the HTTP protocol, they must be able to recognize and process SOAP as well. We have developed an initial SOAP proxy prototype [3], that we extend further in this paper.

Introducing an extra communication layer means increased flexibility when it comes to selecting which transport mechanism(s) to use. Additionally, using this approach means that the end-to-end connection dependency is removed in favor of a per-hop-behavior. In this case, the application software can often be left unmodified. However, there is a possibility for information corruption along the route, which may not be detected without an end-to-end connection. Also, since packets are acknowledged on a per-hop basis, you do not get end-to-end reliability. These two issues can be mitigated if the client uses application level solutions for error control and reliability. For example, using XML signatures will ensure that any modifications to a SOAP message is detected by the receiver, despite the lack of end-to-end error control. Furthermore, using for example the WS-ReliableMessaging specification provides application level acknowledgements, which mean that you do not need an end-to-end connection on the transport layer to acknowledge delivery. Thus, client software must use the appropriate Web services specifications to add the desired level of resilience to their SOAP messages.

2.3 Network Heterogeneity

The third problem arises when heterogeneous networks are interconnected. In disadvantaged grids it is not uncommon to experience data rates of less than 1000 bits/s [4]. In particular, when several users are using the network simultaneously, the effective data rate can become very low because resources are shared. Connecting such networks to faster networks introduces a risk that the gateway between the networks has to drop packets due to its buffers filling up faster than the packets can be transmitted out onto the lower capacity network. This problem can be countered by introducing store-and-forward capabilities into the network. In addition, a store-and-forward capability can help alleviate the problems that arise from frequent communication disruptions, which can prevent a message from being delivered immediately. Having store-and-forward support can ensure that the message is not dropped and subsequently having to be retransmitted.

When traversing heterogeneous networks, different communication protocols may be required. This means that a message traversing several networks may have to use multiple different protocols on its way from sender to recipient. Therefore, it is necessary to add store-and-forward on the application layer, and not on the network layer.

3. Delay and Disruption Tolerant SOAP Proxy

In our previous work [5], we have focused on how to reduce network traffic through the use of techniques such as compression and content filtering. We now extend this work, by introducing response caching and publish/subscribe, to further reduce network traffic. We have implemented all these mechanisms, combined with the techniques discussed above, in a proxy prototype.

Our prototype middleware system, called the Delay and Disruption Tolerant SOAP Proxy (DSProxy), addresses many of the challenges associated with utilizing Web services in disadvantaged and heterogeneous networks. This proxy is an implementation of a wide array of principles and mechanisms that tackle different aspects of these challenges. The proxy software is designed to be modular, and its functionality can be divided into *core functionality* and optional *plug-ins*. The core functionality includes the basic optimizations that are required to make standard Web services work in tactical networks, while the optional plug-ins provide further optimizations such as caching and publish/subscribe support. A key difference between the two types of functionality is that the core functionality does not rely on inspecting the SOAP messages that pass through the proxy. The plug-ins, on the other hand, might require inspecting data, or rely on making small modifications to clients and Web services.

3.1 Core Functionality

The DSProxy is a proxy for Web services, and is designed to handle all types of information and traffic flows that are suited to be implemented as Web services. Some types of data, such as voice flows or other types of information with strict real time demands are likely to require other forms of optimizations beyond what can be supported by standard Web services.

The DSProxy system comprises multiple proxy instances deployed in a network, and forms what is known as an overlay[3] network (see Figure 2). The nodes that constitute the overlay network communicate with each other and exchange information about their state and the environment in which they operate, in order to maintain the overlay. The proxy is a lightweight solution that can be deployed locally on any node in the network. The largest benefit will be achieved if the proxies are deployed both on nodes that bridge networks, and in any node that communicates over a disadvantaged grid.

[3] An overlay network is a logical organization of nodes that form a virtual network on top of physical networks.

Dette er en postprint-versjon/This is a postprint version.
DOI til publisert versjon/DOI to published version: 10.1109/MCOM.2010.5594680

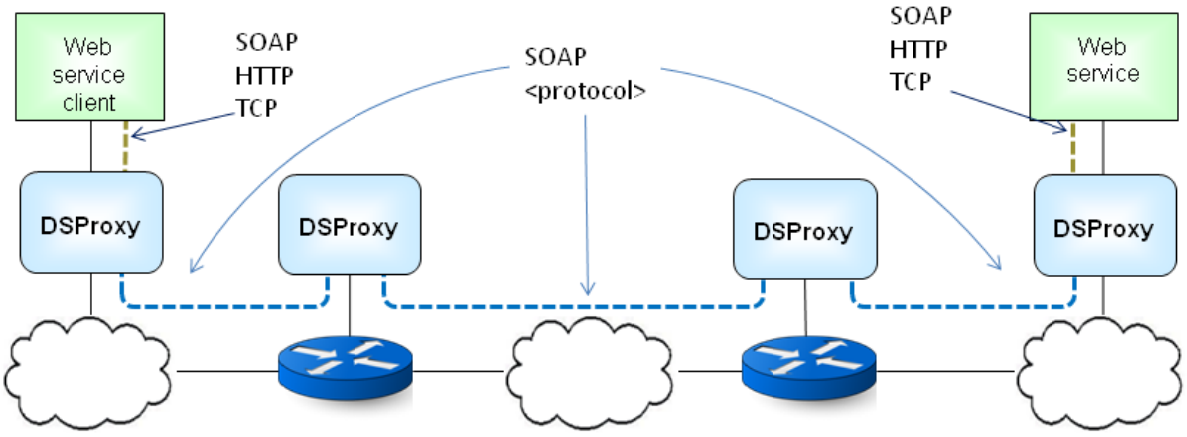

Figure 2: A DSProxy overlay network

The DSProxy overlay can be maintained in one of two ways: 1) Manual configuration by an administrator; or 2) Dynamic configuration by the proxies themselves. In the first case, there is a human in-the-loop responsible for creating a usable configuration. The second case warrants further discussion, because the proxies make autonomous decisions regarding overlay reconfiguring and routing. In the dynamic overlay, proxies advertise their presence by broadcasting messages, allowing them to be discovered by neighboring proxies. Topology information is distributed among the overlay members, which use the information to create the shortest routes through the overlay. If network conditions change, then the dynamic overlay reacts by reconfiguration to address these changes.

For these reasons, using an overlay adds resilience to a dynamic network. However, in cases where a multi-hop network is stable, then going through the proxy overlay has more latency compared to invoking a Web service directly. This is because there is a small processing penalty per overlay node hop: In each proxy, uncompressed SOAP messages add 3 ms processing time, whereas compressed SOAP messages add 11ms processing time (this is because they need to be decompressed in each hop so the SOAP header can be inspected). Additionally, some latency can be added due to the fact that the messages follow the shortest overlay path, which might be slightly longer than using the direct route. However, this effect will be mitigated by the overlay reconfiguration algorithm which ensures that the messages are sent via as few proxies as possible.

In order to enable end-to-end connections across multiple and heterogeneous networks, the DSProxy system offers store-and-forward capabilities at each instance in the network. Instead of relying on end-to-end connections between information consumers and producers as shown in Figure 1, the nodes relay the information between the parties by initiating single-hop connections between proxies, storing the information and retransmitting or re-routing when encountering failures. By storing information at intermediate nodes, added robustness is introduced into the

Dette er en postprint-versjon/This is a postprint version.
DOI til publisert versjon/DOI to published version: 10.1109/MCOM.2010.5594680

network. The messages are stored locally, either in memory or on a storage medium on the node. Once the message is successfully relayed to the next proxy, the message is removed from the storage.

By eliminating the need for end-to-end connections, one is no longer required to use the TCP-protocol for information exchange; one is free to choose more suitable transport protocols instead. As illustrated in Figure 2, SOAP is used as a pervasive message format, but the underlying protocols can be replaced on a per-hop basis depending on the network characteristics and the quality of specific links between the DSProxy nodes. This feature, together with the store-and-forward capabilities, enables efficient traversal of multiple heterogeneous networks, while hiding the underlying heterogeneity from the end users. For any particular Web services invocation, the request may traverse a series of different networks, dealing with different link qualities and utilizing different transport protocols. By utilizing compression, SOAP and XML-based information exchange is made possible even in networks with severely restricted data rates. Our prototype uses GZIP compression, but due to the modular design of the proxy, support for additional algorithms can be included later.

A key concern is standards compliancy, and our prototype interoperates with COTS Web services software without modification. The system accepts regular incoming Web services invocation requests over HTTP and TCP, and is able to relay such requests to proxies, and finally to Web services endpoints, as shown in Figure 2.

The TCP connection from the client is terminated at the first DSProxy, meaning that the TCP request from the client is acknowledged immediately. This allows the TCP connection between the client and the first proxy to be kept open for the entire duration of the invocation process, and it will not time out, regardless of how long it takes for the service to respond. Once the response message is available, it is returned to the client using the same connection.

By placing the first proxy as close to the client as possible, preferably on the same physical machine/device, challenges associated with TCP and disadvantaged grids are avoided. This deployment scheme ensures store-and-forward capabilities throughout the network. Likewise, for the last communications hop, between the last proxy in the chain and the Web service, standard HTTP and TCP is used, allowing interoperability with existing Web services without requiring any modifications. By simply modifying the URLs used by clients to invoke Web services, the communication is routed through the overlay network. Any instructions to the proxies are specified as extra parameters added to the URL. Apart from the modified URL, both Web services clients and services are unaware of the presence of the DSProxy system.

3.2 Security

Security is a key concern in all military networks, and the increased information flow between systems that is central to the NNEC vision demands flexible security solutions. Current security policies tend to require IPsec or link layer cryptography,

Dette er en postprint-versjon/This is a postprint version.
DOI til publisert versjon/DOI to published version: 10.1109/MCOM.2010.5594680

while Web services security mechanisms can provide end-to-end application layer security.

The DSProxy core functionality is designed to work with existing security mechanisms, ensured by the core functionality of the proxy being content agnostic. The proxies will not parse, inspect or interpret the body parts of SOAP messages, which allows for end-to-end security through encryption. Because the information needed to route the messages is located in the SOAP header, there is no need to decrypt the messages being transported. If a Web service client relies on encrypting the entire SOAP message, a new SOAP envelope can be wrapped around the original message, placing routing information and DSProxy parameters within the unencrypted SOAP header. This is done by the first proxy in the chain, which gets this information from the modified invocation URL provided by the client. Because the SOAP header is unencrypted, the system remains compatible with the various Web services security related standards (see [6] for a survey of these). Also, because the DSProxy system does not require any special behavior by protocols part of the IP stack, it can also be used with lower layer security mechanisms, such as IPsec and link layer cryptography.

While the DSProxy core functionality works together with end-to-end security solutions, the optional advanced functionality described below, such as response data caching and the publish/subscribe mechanism, require all intended recipients to be able to decrypt the response message targeted towards one specific client. Because COTS Web services are unaware of these modes of operation, they will encrypt a given response message using the key designated for the client responsible for the actual invocation. In order to make this work with multiple clients, one should rely on group keys distributed to all intended recipients.

3.3 Optional Plug-ins

In many types of tactical scenarios it is a common situation that multiple information consumers require the same information, provided by the same information producers. This can be applications such as blue force tracking, weather forecasts or sensor data. In order to optimize bandwidth requirements for such scenarios, the DSProxy system supports response data caching, which offers a kind of economy of scale when serving multiple clients requesting the same information. By caching response data messages at multiple nodes in the network, clients may, depending on the degree of time-criticality of the information, suffice with cached response data. This means that, instead of having the chain of proxies relay the Web services invocation message all the way to the actual Web service and invoking it, a proxy along the way may determine that it has a valid cached response for the request, and return it to the client. Because caching behavior is specified by each client by appending a parameter to the Web service endpoint URL, the client is in control of if and when cached response data will suffice, and how old these messages may be. While a client may request response data caching on a per request basis its availability is not guaranteed, because the caching behavior is configured for each DSProxy. This configuration option is critical for proxies running

DOI til publisert versjon/DOI to published version: 10.1109/MCOM.2010.5594680

on limited devices, which may lack the system resources needed for extensive caching.

For contexts in which multiple consumers subscribe to services expected to produce response messages that change very little over time, great size and bandwidth reductions can be achieved by only sending the message differences between proxies. Once a DSProxy has received a response message from another instance, it is able to reconstruct the next response message using only the difference between the messages. However, this requires the proxies to keep state information, since they need a previous message to deduce the new message from using the differential information. An example of such a scenario is when multiple consumers subscribe to a blue force tracking service. While the messages produced by the Web service in response to invocations comprise HTTP headers, SOAP headers, and SOAP bodies, all that may have actually changed between two subsequent responses may be the units' location coordinates. Experiments have demonstrated that such messages containing hundreds of bytes of information can be reconstructed without loss using differences consisting of merely tens of bytes.

While standard Web services are based on the request/response paradigm, this is not always the best choice for disadvantaged grids. 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 sense to have the information producers push the information to the consumers as soon as it becomes available. The DSProxy system enables such publish/subscribe mechanisms to be used together with standard Web services software, requiring only minimal modifications to be made to the clients [7]. In order to subscribe to a Web service, a parameter is added to the URL before routing the request through the overlay network. This parameter instructs all proxies along the invocation path to treat the request as a subscription, and the last proxy in the chain initiates a polling cycle to the actual Web service (see Figure 3). This proxy sends an ordinary request to the Web service at regular intervals, and compares the response to the previous response, in order to see whether the data has changed.

DOI til publisert versjon/DOI to published version: 10.1109/MCOM.2010.5594680

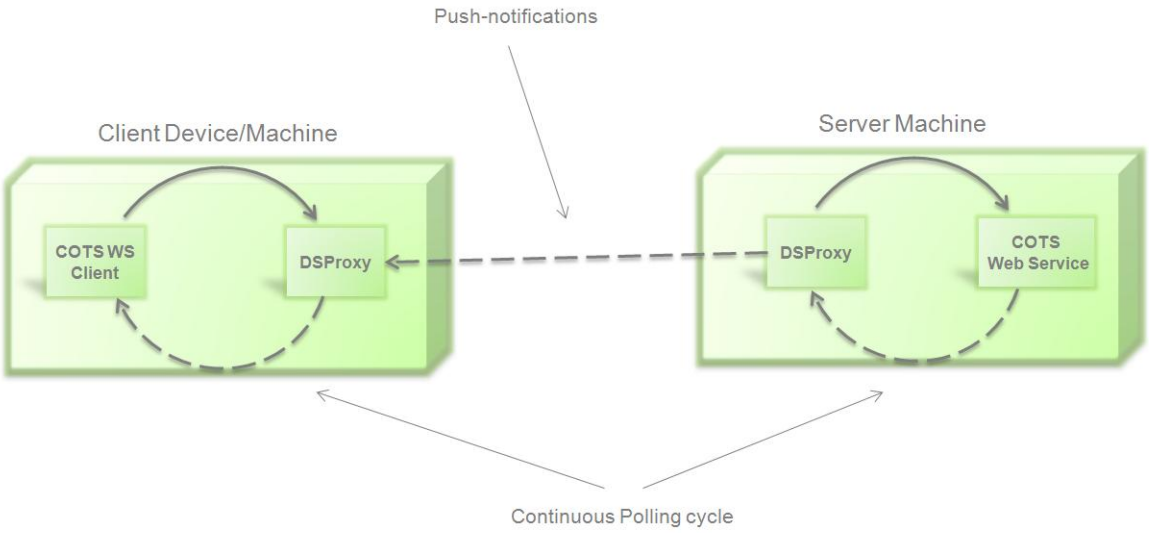

Figure 3: Standard, request/response-based Web services client and server engaging in publish/subscribe communication using the DSProxy overlay network

Placing the last proxy on the same physical machine as the Web service, means that polling only burdens the internal machine resources, and no traffic is transmitted across the network. Once this proxy discovers that new information is available (i.e., the response from the Web service has changed), it notifies its subscribing proxies by initiating an outgoing connection. Once the information arrives at the first proxy in the chain, it is made available for the client to retrieve. As standard Web services clients lack the ability to accept incoming connections, the client is required to poll the first proxy regularly, to obtain updated information. This typically involves wrapping the Web services invocation calls in a loop structure. This loop must also handle when the poll returns without new data. Again, placing the first DSProxy on the client machine avoids added network traffic. As with response data caching, economies of scale are achieved when several clients subscribe to the same information, since one proxy can subscribe on behalf of many clients, as well as on behalf of other proxies.

Publish/subscribe specifications like WS-Notification and WS-Eventing are emerging, but they are not yet in widespread use, compared to regular Web services. WS-Eventing is supported by the DSProxy, and support for WS-Notification is under development. However, the DSProxy publish/subscribe mechanism works with non-publish/subscribe Web services, allowing already existing software to utilize publish/subscribe.

3.4 Field Trials

Our proxy was tested in a series of live field trials at Combined Endeavor in September 2009, with promising results. During these experiments, the DSProxy system was deployed in a small combined setup consisting of four networks that were interconnected using IP on the network level and the proxies on the

Dette er en postprint-versjon/This is a postprint version.
DOI til publisert versjon/DOI to published version: 10.1109/MCOM.2010.5594680

application level. Figure 4 shows the network setup used during these experiments, where we interconnected the Norwegian and NATO C3 Agency (NC3A) networks. Interoperability between the nations was provided using the tactical communications standard (TACOMS) for federated networking, and the common Region C (RG C) Combined Endeavor backbone. The DSProxy was deployed in the gateways between the networks, and on each individual node within the Norwegian MANET. This deployment meant that not only did the proxy ensure successful information exchange between the networks, but it also enabled Web services communication within the MANET, where most of the tactical communication took place. This meant that the tactical users did not have to concern themselves with whether or not they were connected to the rest of the network when sending information.

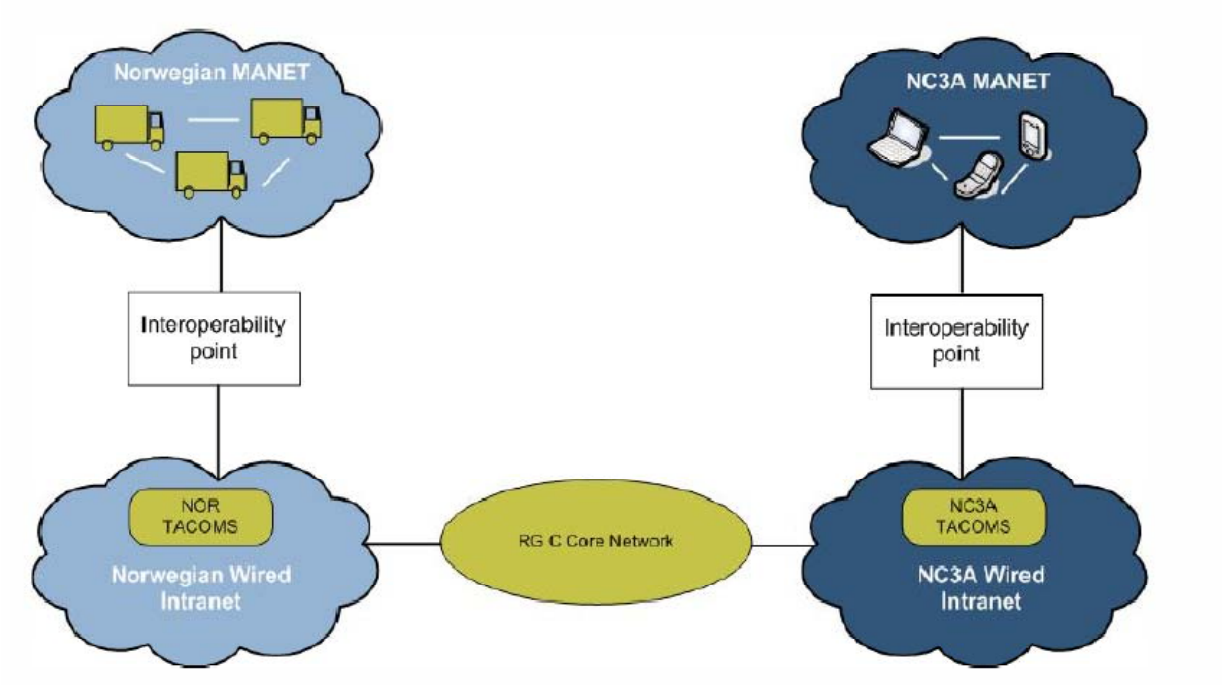

Figure 4: Combined Endeavor experiment setup

5. Related Work

In [8], a solution employing proxy servers for disruption tolerant networking (DTN) is presented. This work uses proxies to translate from applications' native use of end-to-end transport protocols to network level DTN messages. This solution allows standard TCP and UDP software to take advantage of DTN without modification. However, the solution relies on network level DTN support.

Faucher et al. [9] introduce an experimental Web services proxy aiming to overcome the challenges associated with disadvantaged grids. Their proxy handles SOAP sent over HTTP/TCP. The authors argue that XML compression will enable the proxy to perform better in tactical networks, but they have not implemented this.

Dette er en postprint-versjon/This is a postprint version.
DOI til publisert versjon/DOI to published version: 10.1109/MCOM.2010.5594680

Metzger et al. [10] have created a proxy adding delay tolerance to the XML-based Jabber protocol. Jabber is mainly used for chat, but a SOAP extension which allows Jabber to carry Web services traffic exists. Their prototype system does not support dynamic joining of group members, limiting its use to pre-configured static users.

6. Summary

In this paper we have discussed the importance of Web services for realizing NNEC. Web services, being based on standards, are central to the NNEC vision because the technology provides interoperability between applications - it enables the NATO nations to interconnect their command and control systems and share information in a functional and cost-efficient manner.

Further, we have highlighted issues related to adopting Web services technology in military networks, and identified ways of mitigating these issues. In particular, we have introduced a network of proxies that add delay tolerance to SOAP, in addition to employing other overhead-reducing measures. We have implemented a prototype solution, and shown that it makes it feasible to employ Web services in tactical networks in a field trial.

At Combined Endeavor 2009, we have shown that our prototype enables the use of unmodified Web services software across heterogeneous, tactical networks, and that it also can augment the functionality of such Web services by introducing support for subscription based services.

7. References

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

[2] S. Papanastasiou, “Investigating TCP Performance in Mobile Ad Hoc Networks,” VDM Verlag, May 2008.

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

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

[5] K. Lund, A. Eggen, D. Hadzic, T. Hafsøe, and F. T. Johnsen, “Using Web Services to Realize Service Oriented Architecture in Military Communication Networks,” IEEE Communications Magazine 46(10), October 2007, pp 47-53.

Dette er en postprint-versjon/This is a postprint version.
DOI til publisert versjon/DOI to published version: 10.1109/MCOM.2010.5594680

[6] N. A. Nordbotten, “XML and Web Services Security,” FFI-report 2008/00413, http://rapporter.ffi.no/rapporter/2008/00413.pdf

[7] E. Skjervold, T. Hafsøe, F. T. Johnsen, and K. Lund, “Enabling Publish/Subscribe with COTS Web Services across Heterogeneous Networks,” 4th International Workshop on Architectures, Concepts and Technologies for Service Oriented Computing (ACT4SOC 2010), Athens, Greece, July 2010.

[8] K. Scott, “Disruption tolerant networking proxies for on-the-move tactical networks,” In IEEE MILCOM 2005, Vol 5, October 2005, pp 3226 — 3231.

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

[10] R. Metzger and M.C. Chuah, “Opportunistic information distribution in challenged networks,” CHANTS '08: Proceedings of the third ACM workshop on Challenged networks, San Francisco, California, USA, 2008, pp 97 – 104.

DOI til publisert versjon/DOI to published version: 10.1109/MCOM.2010.5594680

	KETIL LUND (ketil.lund@ffi.no) is a scientist at the Norwegian Defence Research Establishment (FFI), where he has been working since 2006. His research interests include Service Oriented Architectures, Web Services, Quality of Service, and middleware. At FFI he is currently working within the area of secure pervasive SOA. He received his Ph.D. in informatics from the University of Oslo.
	ESPEN SKJERVOLD (espen.skjervold@ffi.no) is a research scientist at the Norwegian Defence Research Establishment (FFI), engaged in theoretical research and practical development in areas such as distributed systems and Service Oriented Architecture. Skjervold is also a part of FFI's team of professionals aiming to support and refine FFI's common systems design and development practices. Skjervold holds a master's degree in Distributed Systems and Computing (2007, MSc DISC) from Brunel University of London.
	FRANK T. JOHNSEN (frank-trethan.johnsen@ffi.no) received his Cand.scient. degree from the University of Oslo (UiO) in 2002. Following this he has been working with networks and distributed systems at UiO, until he started work as a scientist at the Norwegian Defence Research Establishment (FFI) in 2006. At FFI he is currently working within the area of secure pervasive SOA. His research interests include Web Services, Quality of Service, and middleware.
	TRUDE HAFSØE (trude.hafsoe@ffi.no) is a scientist at the Norwegian Defence Research Establishment (FFI), where she has been working since 2006. Before coming to FFI she worked with content distribution systems at the University of Oslo (UiO). She received her Cand.scient. degree from UiO. Her research interests are Web Services, Quality of Service, and network protocols.
	ANDERS EGGEN (anders.eggen@ffi.no) received his M.Sc. degree in computer science from the University of Oslo in 1991. Since 1992 he has been employed by the Norwegian Defence Research Establishment (FFI), where he is currently a principal scientist. He has worked on many aspects of communication and application protocol evaluation and design, as well as security and SOA solutions for military systems. He has contributed significantly in technical standardizations activities, and has been editor to both IETF and NATO standards.

Dette er en postprint-versjon/This is a postprint version.
DOI til publisert versjon/DOI to published version: 10.1109/MCOM.2010.5594680

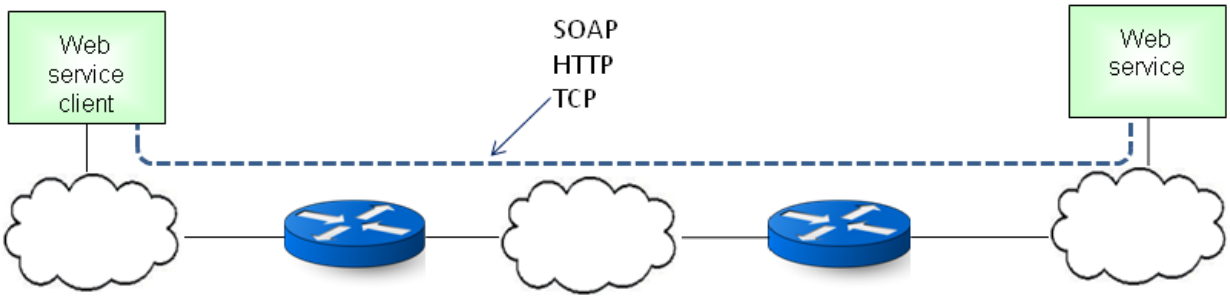

Figure 1: HTTP and TCP establish end-to-end connections

Dette er en postprint-versjon/This is a postprint version.
DOI til publisert versjon/DOI to published version: 10.1109/MCOM.2010.5594680

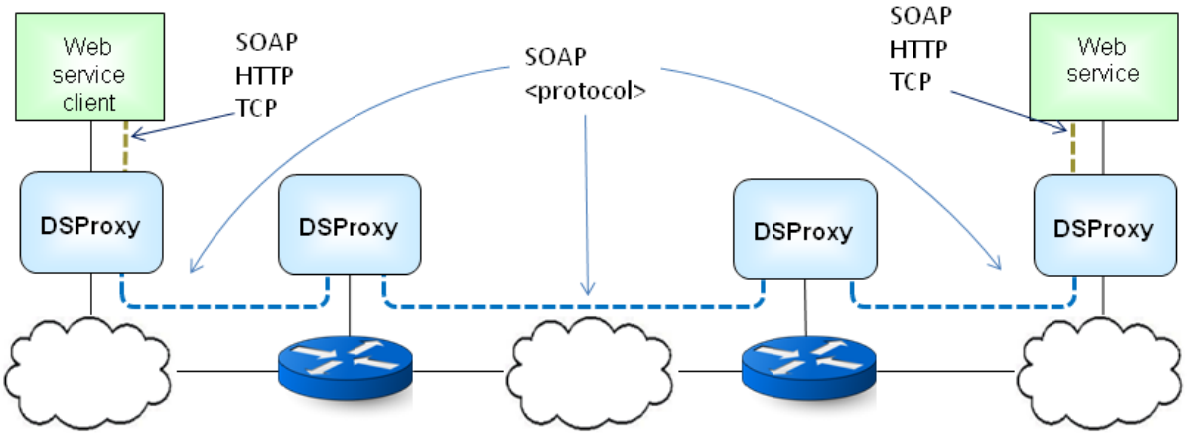

Figure 2: A DSProxy overlay network

Dette er en postprint-versjon/This is a postprint version.
DOI til publisert versjon/DOI to published version: 10.1109/MCOM.2010.5594680

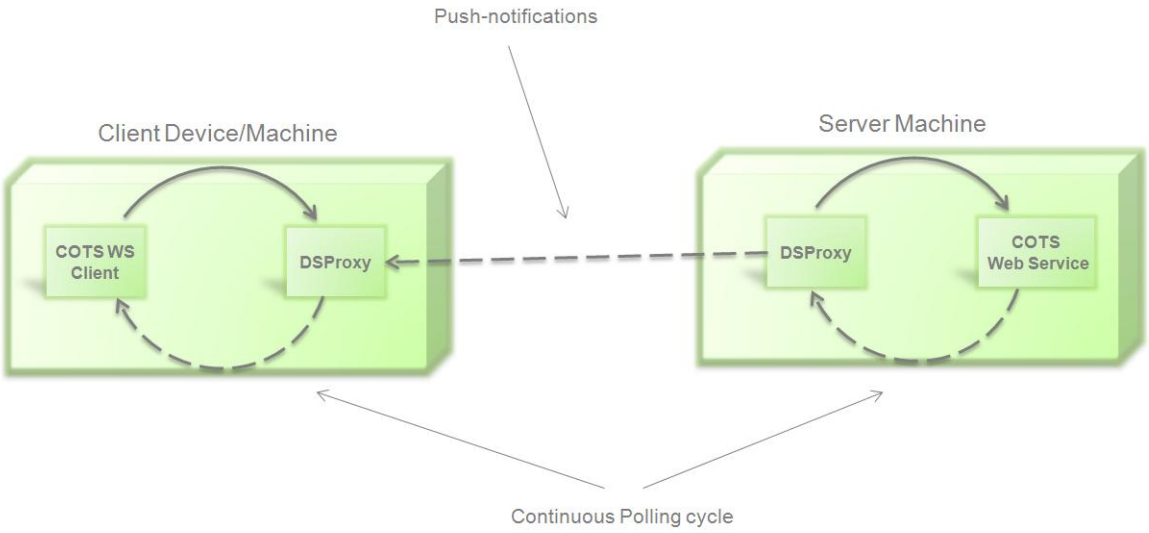

Figure 3: Standard, request/response-based Web services client and server engaging in publish/subscribe communication using the DSProxy overlay network

Dette er en postprint-versjon/This is a postprint version.
DOI til publisert versjon/DOI to published version: 10.1109/MCOM.2010.5594680

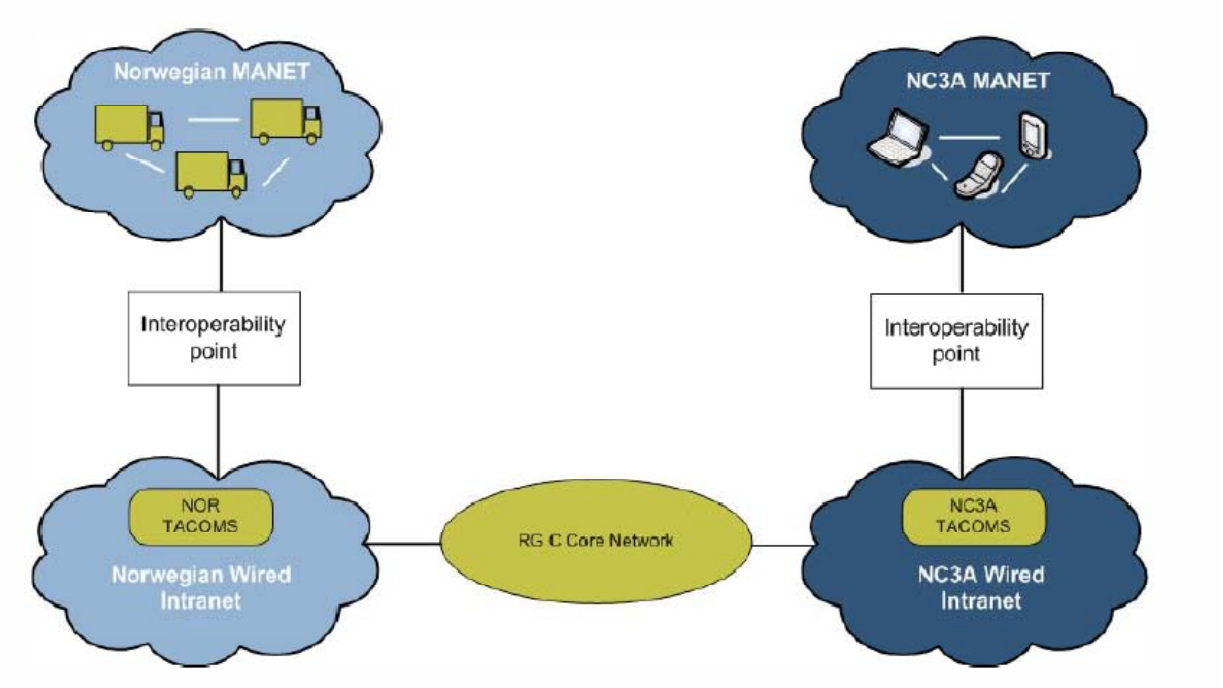

Figure 4: Combined Endeavor experiment setup

Dette er en postprint-versjon/This is a postprint version.
DOI til publisert versjon/DOI to published version: 10.1109/MCOM.2010.5594680