12th ICCRTS

"Adapting C2 to the 21st Century"

# "Adapting Web Service Publish/Subscribe Technologies for use in NEC C2 Systems"

Track 8: C2 Technologies and Systems, Track 7: Network-Centric Experimentation and Applications, Track 2: Networks and Networking

Trude Hafsøe, Frank T. Johnsen, Ketil Lund, Anders Eggen
Norwegian Defence Research Establishment (FFI)
P.O. Box 25
NO-2027 Kjeller
Norway

Point of contact:
Frank T. Johnsen
Norwegian Defence Research Establishment (FFI)
P.O. Box 25
NO-2027 Kjeller
Norway

+47 63 80 79 60
frank-trethan.johnsen@ffi.no

## Abstract:

Web Services (WS) are in widespread use on the Internet today, and COTS products are readily available. WS are being considered as an enabling technology for NEC, and seem well suited since they provide both push/pull communications as well as the asynchronous publish/subscribe paradigm. Push communication allows orders to be disseminated to troops; pull can be used by the troops to retrieve additional information on-demand. Publish/subscribe is suitable for sending updated track data and improving situational awareness.

The challenge lies in using WS over so-called "Disadvantaged Grids" (DG), i.e., networks with low available bandwidth and high error rates. Tactical communication systems can often be classified as DG, and are therefore usually not well suited for WS. In this paper we present our ideas for adapting WS to DG. Proxy servers can be used to minimize the number of messages traversing the links, and can thereby improve both push/pull and publish/subscribe communications. We will discuss push/pull briefly, but the main focus of this paper is publish/subscribe. Furthermore, we present techniques for reducing the publish/subscribe communications by adapting the representation of the information that is being transmitted.

# 1. Introduction

The NATO NEC feasibility study (NNEC FS) presents considerations for NEC, and contains a roadmap for its implementation. In this study, IP is presented as the universal communications protocol, and it is discussed how "EoIP", everything over IP, will facilitate interoperability between systems. Furthermore, the study mentions that IP should be used for all systems that can benefit from it. In Network Based Defence (NBD), the Norwegian equivalent of NEC, we anticipate the use of IP in all systems. Indeed, research has shown that using IP is feasible even over tactical communications systems such as the Harris HF radio [5].

The first step towards NEC/NBD is to integrate legacy strategic and tactical systems into a common network. For such integration the modular concept from Service Oriented Architectures (SOA) is essential. Each legacy system can be viewed as a separate module that needs to be interconnected with others. In order to get the different modules to cooperate one needs a common standardized means of communications between them. On the network layer the IP protocol has been chosen, but one needs to decide on a protocol for the application layer as well. In NBD we are investigating the possibility of using WS for this purpose. WS use the XML-based SOAP protocol for information exchange, and are in widespread use on the Internet today, with COTS products readily available. Thus, it makes sense to attempt to utilize this technology for military purposes. This is acknowledged not only in Norway for NBD, but seems to be a general trend in the industry as the Network Centric Operations Industry Consortium [1] supports the WS standards.

Previously, we have performed experiments with WS in a multi-national scenario at CWID 2006 [4]. In these experiments, we showed that WS, including publish/subscribe, could be used to exchange track data (with end-to-end security) between nations. We used the object-oriented XML-version of the Command and Control Information Exchange Data Model (C2IEDM) from the Multilateral Interoperability Programme (MIP), and exchanged XML-based messages using SOAP over HTTP. Using the WS-Notification standard [2], nations could subscribe to track updates from each other, and thereby establish a common operational picture. Our experiments showed that the utilization of WS in NEC is feasible, but it also revealed several challenges [3], some of which are addressed in this paper. Furthermore, in those experiments WS were used at the strategic level, where bandwidth is abundant. For full-fledged NEC/NBD we need to integrate tactical communications into the network as well. We discuss the challenges that arise in this context in this paper, along with our ideas for overcoming some of the obstacles.

The remainder of this paper is organized as follows: First, we discuss background and motivation in section 2. Second, we present relevant standards for publish/subscribe in WS in section 3. Then we proceed to discuss several concepts that can be used to improve the use of WS in DG in section 4. Section 5 concludes the paper.

## 2. Background and motivation

If we look at today's military solutions for information exchange, the information flow is typically defined by a set pattern from sender to recipient. It is pre-planned and pre-configured, and changes normally require manual assistance. The solutions used are often stove piped, tailored for certain applications within one military service and with no or minimal interoperability with other types of systems. If we look at the NEC ambitions of seamless information exchange between units, there is certainly a requirement for more flexible and dynamic concepts of information sharing.

The aim of NEC is to increase mission effectiveness by networking military entities, enhancing the sharing of information and situation awareness. The key prerequisite of shared situation awareness is increased access to and sharing of information. By using SOA as a foundation for the Information Infrastructure, military resources may be made available as services that may be published and utilized over a communication infrastructure. The service itself is defined using a well-defined interface that exposes the functionality and hides the underlying implementation details. Services may be aggregated, by either the service provider or service consumer, to create more advanced services. This modularization makes introduction of and dynamic reconfiguration of services easier.

Web Services are promising technologies for implementing SOA, allowing for dynamic information sharing between military units. Web Services provides loose coupling of functional entities that allows for the dynamic and flexibility required in NEC. In the Web Services functional model, Service Consumers do not need to know in advance where the Service Providers are located (only where the Service Registry is). The Service Providers do not need to know in advance where the Service Consumers are located (only where the Service Registry is). And the Service Registry does not need to know in advance where neither the Service Consumers nor Service Providers are located. This makes ad-hoc organization of entities easier since the requirement for preplanning is minimal.

In NEC there is an ambitious requirement for users at all operational levels to seamlessly exchange information. In order to achieve efficient information exchange between these users, the SOA solutions need to work with different types of information and communication systems. Systems and equipment used at the various levels are different, and the information exchange must be adapted to fit the capacity of the systems used. Data-rate constraints in tactical networks impose great challenges that have to be solved in order to fully deploy a SOA supporting NEC.

Using Web Services, information exchange may be done using either request/response interaction style or event driven models like publish/subscribe. As one of several means of reducing network traffic, the concept of publish/subscribe may be used. The publish/subscribe interaction style has its strength in more efficient information exchange, avoiding information overflow. Simply speaking, publish/subscribe means that you will only receive the information that you have subscribed to. This concept utilizes a combination of push and pull. As opposed to a general ”push” mechanism there are benefits in that you may select (subscribe) to the information sent to you (e.g. sensor tracks) , and when using pure “pull” principles you are not able to notify listeners when events occur.

The use of proxies and caching techniques are other measures to reduce the use of bandwidth in a SOA, which are described in this paper.

# 3. Publish/subscribe using WS

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

At present there are two standardization efforts regarding publish/subscribe for WS: OASIS finished its Web Services Notification (WSN) standard [2] late in 2006, whereas W3C has produced a draft version of a similar framework called Web Services Eventing (WS-Eventing) [8] which may eventually become a standard. The WS-Eventing specification defines a baseline set of operations that allow WS to provide asynchronous notifications to interested parties. WS-Eventing is available as a public W3C draft [8], but is not yet a standard. It provides basic publish/subscribe functionality. WS Eventing has similar features to that of WS-BaseNotification which we present below, so we will not go into the details of the draft. However, the techniques we present in later chapters should be possible to implement using either framework.

In this chapter we summarize the most important aspects from [2], [9], and [8]. We discuss event-based communications from a military perspective. For examples of civilian applications of event-based SOA, see [9].

## *WS-Notification*

WS-Notification is a Publish/Subscribe notification framework for Web services. There are three parts to the specification: WS-BaseNotification, WS-BrokeredNotification and WS-Topics.

- **WS-BaseNotification** [11] defines standard message exchanges that allow one service to subscribe and unsubscribe to another, and to receive notification messages from that service. The WS-Eventing specification provides similar functionality to that of WS-BaseNotification, but they are not compatible with each other.
- **WS-BrokeredNotification** [12] defines the interface for notification intermediaries. A Notification Broker is an intermediary that decouples the publishers of notification messages from the consumers of those messages; among other things, this allows publication of messages from entities that are not themselves Web service providers.
- **WS-Topics** [13] defines an XML model to organize and categorize classes of events into "Topics," enabling users of WS-BaseNotification or WS-BrokeredNotification to specify the types of events in which they are interested.

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

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

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

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

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

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

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

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

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

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

## WS-BaseNotification

The WS-BaseNotification specification unifies the principles and concepts of SOA with those of event-based programming.

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

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

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

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

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

- **Topic filters** provide a convenient way of categorizing kinds of notification. A topic is a concept used to categorize kinds of notification and their associated notification message types. A topic filter excludes all notifications which do not correspond to the specified topic or topics. Topics are standardized in WS-Topics, which defines topic trees and the use of namespaces. Further discussion of this standard is beyond the scope of this paper, refer to [13] for the complete overview. In a military system we foresee extensive use of this standard. For example, one can perform filtering on tracks about only particular vessel or vehicle types, weather reports, etc.

- **Message filters** are Boolean expressions evaluated over the content of the notification message. This kind of filter can also be beneficial in military systems. For example, a squad on an operation needs only track information about the area it is operating in. Several squads operating out of the same local HQ should all get information relevant to them. Here a message filter using each squad's reported position can be employed to ensure that only tracks regarding their geographical location is sent.

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

## WS-BrokeredNotification

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

An implementation of a notification broker may provide additional added-value functions, for

example logging notification messages or transforming topics or notification message content. Logging is important in military systems for security and auditing purposes. We discuss other added-value functions in the chapter on proxies, where we suggest further functionality to implement in a notification broker.

There are cases where it is expensive for a publisher to detect a situation or create a notification message instance. A problem with the simple publisher approach is that even when there are no relevant subscriptions, publishers still have to do both of these things. As an optimization, a broker may offer support for demand-based publishing. If the broker detects that there are no relevant subscriptions, it can pause its subscription with the publisher, resuming it again when it acquires a relevant subscription.

# 4. Our ideas and suggestions

In DG it is important to reduce the amount of data traversing the network links. The available resources must be used optimally to ensure timely delivery of relevant information. Using WS over DG requires some adaptation to work.

In our previous research, we have experimented with various compression algorithms to reduce the inherent overhead in XML message exchange, and we concluded that data compression is necessary to make WS work over DG [14]. Our most important findings in that study were: The Zlib [15] algorithm gives the best overall performance when comparing compression ratio with compression/decompression speed. Specialized proprietary XML compression techniques such as Xmill [16] and EFX [17] can yield even better compression rates than Zlib on small XML documents, but they are slower than Zlib. In a DG the network is the limiting factor and not the processing capacities of the nodes, so compression is beneficial and should definitely be used. A key point for NBD is interoperability with other nations. The choice of compression algorithm should be standards based, so it is important to keep an eye on the developments in industry. Standardization efforts regarding binary XML are being undertaken by W3C at present [18].

Assuming some form of compression is used, we will now discuss two distinct ways to further improve WS communications over DG.

## *Proxy servers*

A proxy is a unit which functions as an intermediary between two communicating parties [7]. Such an intermediate node can perform several different functions in a network; it can reduce overhead, increase security, increase availability and reduce access latency. Ideally, one would want to place a proxy at the interconnection between different networks. A proxy can bring a lot of added-value functionality to a network which a regular gateway cannot perform, for example store-and-forward functionality. In a DG where nodes are highly mobile and network links are somewhat unstable, the use of store-and-forward mechanisms for messages between networks should be employed.

A proxy operates on the application layer; layer 7 in the OSI-model. This means that a proxy has access to more information than a router or a firewall that operates on lower layers of the OSI-model, enabling it to employ more advanced (content based) techniques for example when filtering.

Three of the most common proxy functions are:

- Connection management
- Caching
- Filtering and firewalling

Squid [6] is one example of a proxy that performs all these functions for web pages and is in widespread use on the Internet today. At present we are unaware of any proxy solutions that support WS. This chapter will discuss how proxy techniques currently employed for web pages on the Internet can be adapted for use with WS in NBD.

WS support two different communication paradigms; request/response and publish/subscribe. In a request/response service, the client will send an explicit request to a server, which in turn will process the request and respond with the requested data. In a publish/subscribe (notification) service, the client will subscribe to a service at a certain server, and the server will send new information to the client as soon as the data becomes available. Basically, request/response is traditional “pull” communications, whereas notifications uses a combination of “push” and “pull” communications. We will now describe how proxies can be employed to improve both “pull” and “push” communications in WS.

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

The following example illustrates how a proxy can be used to increase the efficiency of publish/subscribe communications by reducing network load. A notification service has a server which handles subscriptions and the dissemination of new information to its subscribers. When a client wishes to subscribe to information from such a service it needs to register its interest with the server. This is done by sending a "subscribe" message to the server, which then confirms that the message is received and that a subscription has been established. The illustrations below show how a proxy can be employed to optimize various aspects of the notification service. Figure 1 illustrates a subscription establishment phase in which two clients are connecting through the same proxy, wanting to subscribe to the same service. First, client A sends a "subscribe" message to the server S. The proxy intercepts the message, noting that client A wants to subscribe to a service at S. This is the first time P sees a request for a subscription to this particular service, so it starts a subscription for this service at S. The server acknowledges that the subscription has been established, and the proxy forwards the acknowledgement to A. At this point S knows that whenever new information is available, it must send a notification to P. P, upon receiving such a notification, knows that it should forward it to A. Other clients may want to subscribe to the same service through P. In this example, B sends a "subscribe" message which is intercepted by P. P notices that it already has established a subscription to this particular service, adds B to its local subscription list, and sends an acknowledge message to B. Assuming that the clients in this example are squad members, and that the squad leader's communications equipment has proxy functionality, then it is obvious that sending only one "subscribe" message over the low bandwidth link between squad leader and HQ avoids unnecessary messages and thus limits the use of the scarce bandwidth.

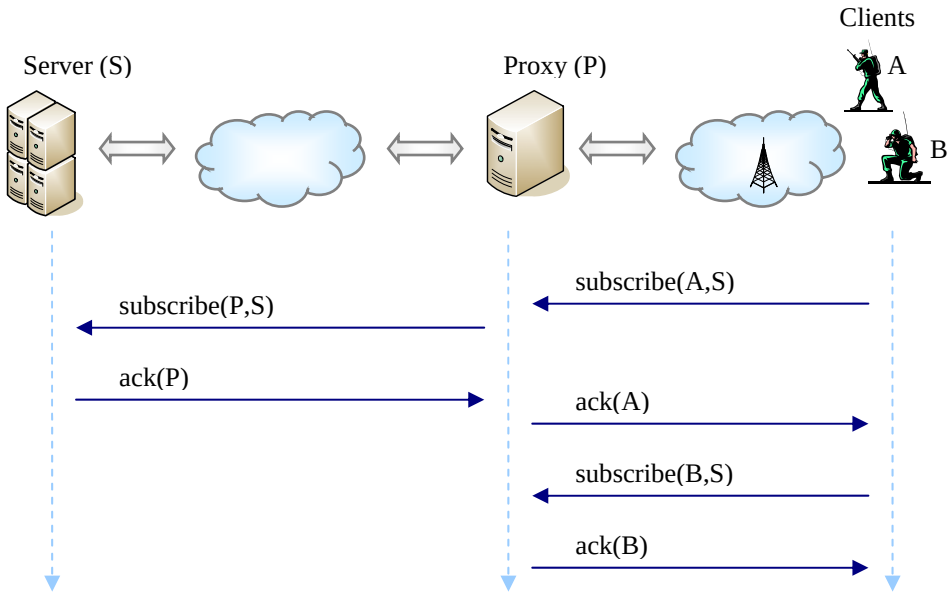


**Figure 1: Subscription by proxy**

When a subscription has been established further communications will be done asynchronously; the server publishes the content by sending a "notification" message to each and every subscriber. Figure

2 illustrates this: S has new information for its subscribers (which in this case is only P), and sends a "notification" message. P has established this subscription on behalf of clients A and B, and will forward this message to both those clients. How P forwards the message will depend on the transmission medium between P and the clients. If the network does not support multicast, then P will have to send one "notification" message point-to-point to each client. Unicast distribution of messages to several clients on the same subnetwork should be avoided if possible to limit bandwidth usage. Multicast should be used instead if it is available, as shown in Figure 3 where the proxy sends one multicast "notification" message to both the subscribing clients.

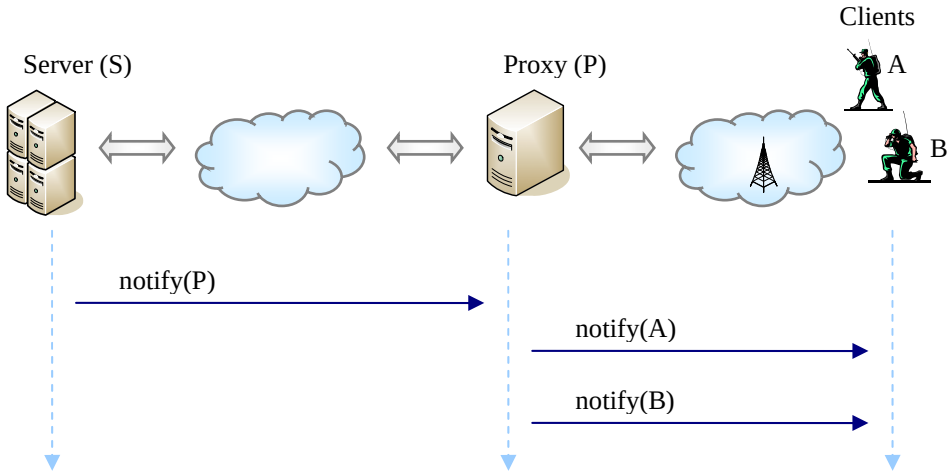


**Figure 2: Notification via proxy, unicast**

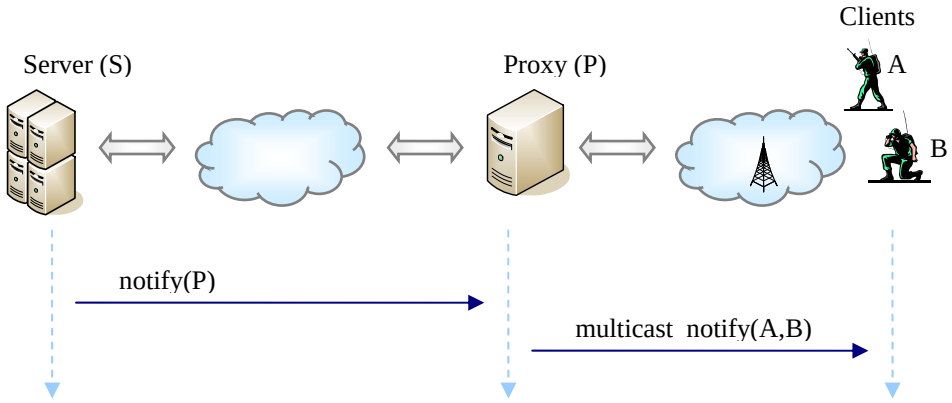


**Figure 3: Notification via proxy, multicast**

Using a proxy has both advantages and disadvantages: An important advantage is that bandwidth is saved between server and proxy. Furthermore, the proxy can adapt the mode of transmission to better utilize the transmission medium between itself and the clients by using multicast, for example. This can lead to a substantial reduction in bandwidth needs, since there will be only one message traversing the network per "notification" sent – no matter how many subscribers there are. This gives the system greater scalability. Further bandwidth savings can be achieved using filtering, which will be discussed below. Introducing a proxy also has some disadvantages. The proxy needs to keep state about subscriptions, and becomes a possible point of failure in the network. If a proxy crashes and loses its

information, then it will not know what to do with incoming notifications, i.e. where to forward them. It is necessary to investigate various solutions for subscription renewal to overcome this problem.

## Caching

Caching is a technique that is used to increase the response time for clients and reduce the load on the network and origin server. For example, it is in widespread use for accelerating web browsing on the Internet [10]. When someone accesses a web page, a local copy can be made in a proxy closer to the client. The next time someone requests the same page through that proxy, it can serve the client directly from its cache rather than having to fetch the page from the origin server. Caching is performed on the basis of observed client behaviour, and is a technique that is complementary to replication. Replication also leads to a copy of the original data being made, but it is controlled by the content owner. The origin server is responsible for making replicas and sending updates when content changes. Proxies are often used for caching, so called caching proxies. The use of caching proxies can be beneficial in DG, since resources are scarce. We will now discuss how the technique can be used with request/response driven Web Services. For simplicity, we will refer to a caching proxy as a proxy for the remainder of this paper.

Figure 4 shows clients in a network, where a proxy is used to connect the sub-network the clients are on to the rest of the world. All communications to and from that network will pass through the proxy. For example, the clients (A and B) can be seen as members of a squad. The proxy (P) is part of the squad leader's equipment since the squad leader can communicate with both the server (S), i.e. HQ and other squad members. If client A requests "object.x" from S, then this request will have to pass through P. P inspects the request, and discovers that "object.x" is not in its cache. The proxy forwards the request to S, which responds with the data. P can now store "object.x" in its local cache, and forward it to A. If another client later wishes to retrieve the same object, then it can be served directly from the cache, as is the case when B requests "object.x". This saves the time that would be spent retrieving the object from S, and leads to less communication between P and S, as well as less load on S since it will need to handle fewer requests.

Employing caching is especially useful if the server side network capacity is as shown in Figure 4, e.g. with the bottleneck between server and proxy. In such cases, all requests that P can handle without having to communicate with S will save bandwidth. Caching also increases the availability and robustness of the system; cached objects exist in the proxy even if the origin server becomes unavailable.

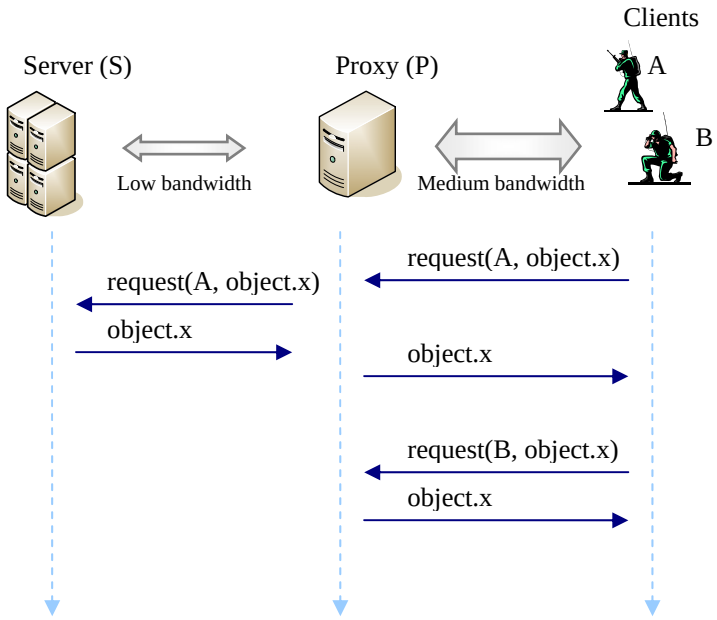


**Figure 4: Caching**

Implementing a caching proxy for WS is a challenging task. The proxy must have an understanding of the objects it caches in order to handle them in an appropriate manner. For example, if an object has a limited time that it remains valid, the proxy must know this. If a client requests the object after it has expired, then the proxy must fetch a new version of the object from the origin server. Caching is not efficient if objects change frequently.

Furthermore, a proxy does not have unlimited storage space. If its cache becomes full, it will have to choose which objects to discard from the cache and which objects it should keep. Proxies often use simple statistical cache replacement algorithms [10] for this purpose.

## Filtering

In a DG it is important to avoid flooding the network with more information than it can handle. On the other hand, it is important that important and relevant information is sent to the receiver. To achieve this it is necessary to prioritize and filter information between networks. Filtering can be divided into three main types:

1. Filtering at the connection level, i.e. granting or denying the establishment of a connection. This can be used in various security measures, but also to prioritize resource use.
2. Filtering of whole messages based on various criteria such as message priority, relevance and classification. Messages with high priority should be dispatched before messages with low priority.
3. Filtering of parts of messages based on the message content.

A proxy can perform all three types of filtering. Type 1 and, to a certain extent, type 2 filtering can be performed at the network layer by a router, whereas more advanced forms of filtering need application layer solutions.

## *Information model optimization*

When using WS, for instance for building a common operational picture, large amounts of information must be transferred over the available communication infrastructure. One method that can be used to reduce the size of the data that is transferred is optimizing the information model. This means a reduction in data size prior to compression.

The way information is represented and exchanged must be taken into consideration when adapting WS to DG. Some principles and mechanisms for information representation are better suited than others for distribution limited capacity networks. Our previous work has been centred around two information models developed by MIP, namely C2IEDM and JC3IEDM, and these two models are used as a basis for the discussion given here. C2IEDM and JC3IEDM support three different mechanisms for information exchange:

- Message based communication
- Replication based communication
- Query based communication

These principles will be described in the context of MIP, but they are independent from the actual data model used for information exchange. As long as all objects can be uniquely identified these principles can be used in conjunction with any information exchange model. This allows us to use these general principles when discussing information exchange in low capacity networks.

We are focusing on networks with low bandwidth and high delay (including turn times). As shown in [5] the achieved throughput varies with packet size. Packets need to be at least 2-5kB before an acceptable throughput (more than 50% of the bandwidth) is reached. The maximum bandwidth available in these networks is low, often below 2kbit/s. This means that while we on the one hand want to send as little information as possible, we also need to avoid sending packets that are too small as this gives low bandwidth utilization. On the other hand larger packets are more vulnerable to bit errors, and more data will have to be retransmitted. This forces the use of smaller packets in networks with high error rates. In addition to these concerns, changes of sending direction should be kept to a minimum as this can incur large turn time penalties.

## Message based information exchange

Message based information exchange means that the XML-documents that are exchanged are referentially complete, which means that a document only contains references to objects which are included in the message. This method of communication was used during the demonstration at CWID in 2006. The actual exchange of messages can be pull or push based. Pull based means that the recipient explicitly requests new messages from the sender, while in push based message exchange the sender automatically transmits notifications. The push based communication mechanism generates less network traffic in total, which makes it the mechanism of choice for transmission over limited capacity networks.

Message based exchange mechanisms can also be classified according to how references to objects within the message are handled:

Inline referencing means that when an object references another object, the referenced object is encapsulated into the object containing the reference. One consequence of this approach is that objects that are referenced more than once is replicated within the message (once for each referencing object), adding a considerable amount of redundancy to the message. One feature of using inline referencing however, is that the message has a structure that is easy for humans to get an overview over as all referenced objects are gathered together. The actual benefit of this feature is debated, as the structures are often too complex to get a good overview over,

When using the by reference approach to handle references between objects the message structure is flattened by using object identifiers (OIDs). Objects will only appear once in each message and

redundancy is avoided.  OIDs must be mapped directly from the core of an OO-system in order to ensure unique and meaningful OIDs. In addition, one must also consider to which degree sub-objects need OIDs of their own.

Message based information exchange, with push based communication, was used in [3]. This form of communication is inefficient because it leads to two forms of redundancy in the message exchange:

- Information about all objects is transmitted every time, without taking into consideration if the object has been changed since the previous update.
- If inline referencing is used, several copies of the same information will be included in the information sent.

Despite the redundancy issues, there are several advantages to using message based information exchange in tactical networks. By using data compression to reduce the size of the messages, the impact of the redundancy is reduced. Experiences from [3] have shown that messages with inline references can easily reach sizes of roughly 500 kB. However, tests preformed in [14] shows that XML based formats are compression friendly, and a message of 500kB can be reduced to 10-15kB by using publicly available compression tools such as GZIP.

In networks with high error rates the redundancy in the messages improves reliability. The impact of losing an update is reduced since the next update will also include the information from the lost update. This reduces the need for retransmissions of lost data, which in turn means less directional changes as the receiver does not have to send requests for missing data. The load on the receiver is on the other hand slightly increased as each receiver must check each object in the data set to see if there has been a change and update the object if a change has occurred.

It is not clear if it is better to use by reference or inline object references. Inline referencing increases the redundancy in the message even more, but that problem can be reduced by using a more efficient XML representation. When by reference is used the redundancy is smaller, but it requires all objects to have a unique OID, and OID assignment and management must be handled accordingly.

Another benefit of message based information exchange is that the receivers can be stateless. All information is sent to all receivers, allowing each receiver to use and handle the information based on the needs of that receiver. That fact that the same information is sent to all receivers whenever a change occurs allows for easy and widespread use of multicast. An additional optimization would be use of proxies for message filtering and possibly to keep state for receivers.

The main downside of message based information exchange is that the large amount of data that is transmitted means that the transmission frequency must be fairly low. This low frequency avoids network overload, but it introduces a delay before receivers are notified of a change that has occurred. In situations with high activity, implying very frequent object updates, replication based information exchange might be preferable. The same applies to situations where the delay in update deliveries must be limited as far as possible.

## Replication based information exchange

When using a replication based mechanism for information exchange the involved parties first exchange an initial set of information in order for them to be more or less synchronized. After this initial communication has taken place, only updates are exchanged. This ensures that only changes in the information about an object (such as a geographical position), is sent in addition to the OID. This form of information exchange does not require messages to be referentially complete. Push based communication is almost always used to send updates, and the sender can either send one message every time there is a change in the information set, or changes can be gathered up and several changes can be transmitted together in periodical update messages.

Using replication based information exchange to replication relational data bases was the initial idea behind JC3IEDM. The fact that the internal structure of the relational data base is revealed in the message layout is the main problem with OO-XML. Exposing the data base structure in this way works well is a purely relational system, since it allows for each replication of the entire data base. An object oriented system on the other hand uses a completely different structure, often with objects encapsulated in other objects. Including the entire information history of an object should also be avoided, as the purpose of the communication is to update all parties on the current situation. Whether to store historic data for each object should be left to the end systems.

Replication based information exchange is far from optimal, particularly when using event-driven communication in tactical networks. The combination of events and replication based information exchange means that a small message will be generated each time an object is updated. This leads to poor bandwidth utilization. In order to improve the bandwidth utilization it is possible to gather up several updates and send them out together as one large message. This approach introduces a small delay in the information exchange, and can only be used as long as such a delay is acceptable.

Using replication based information exchange in networks with high error rates introduces another potential problem. If the update about a given object is lost, and that object updates rarely, there is an increased chance that a receiver will end up with outdated object information. One example of such a situation is when the classification of an observed object changes, by changing its type from neutral to hostile. If this update is lost the entity that was supposed to get that update will not be aware of the change in type until a new change occurs in the same object.

## Query based information exchange

When using replication based information exchange the sender is the active part in the communication. Query based information exchange uses pull based communication, making the receiver the active party. The receiver can for instance request information about all objects within a certain geographical area, and then request further information about one or more of these objects if needed. This is done by using the OID as a reference to the object(s) in question. Messages sent between the parties need not be referentially complete. This allows the level of detail in the queries to vary, making it possible for the receiver to request exactly the information it needs. The receiver can for instance ask for the complete information set, or specify one or more objects it wants information about.

This information exchange mechanism is not ideal in tactical networks, particularly if the receiver repeatedly asks for information about just one or a few objects as this will lead to small messages and many directional changes. One potential use of query based information exchange in tactical networks would be to piggyback a request for the complete information set onto a subscribe message. This would give the receiver a full overview of the information set as a response to starting a subscription.

## Combining different mechanisms for information exchange

Adapting the information representation for use in DG might benefit from a combination of information exchange mechanisms. One such combination is using a hybrid between message based and replication based information exchange. This would mean using message based exchange most of the time, but without requiring messages to be referentially complete. This means that messages can contain only a subset of the objects in the information set, and that not all information about the objects must be included. This allows messages to for instance only contain information about objects that have changed or objects that are of high importance (like enemy forces) even when these have not changed. Determining if the benefit of such a combination is large enough to warrant the increased complexity requires further studies.

If the available communication infrastructure is made up of networks with a higher bandwidth and low delay it would also be possible to combine replication based information exchange with query based

exchange. When using this combination a new receiver would issue a query with a request for an initial copy of the information set, and then rely on replication based exchange for updates. It would also be possible for the receiver to use query based exchange to request further information about an object.

## 5. Conclusions

This paper focuses on how Web services can be used also over so-called Disadvantaged Grids, i.e., networks with low available bandwidth and high error rate. Given the widespread use of WS on the Internet today, together with the standardization efforts that this technology represents, WS is well-suited for implementing NEC/NBD. However, being based on XML and SOAP, WS is not a very efficient means of communication, and can therefore not be used in a DG context without some adaptation.

Our approach to this problem consists of three different measures: We have earlier shown how XML is very compression-friendly, and that SOAP messages can typically be reduced to less than 5% of their original size [14]. In this paper, we have then presented the use of proxy servers and information model optimization, in order to further improve the efficiency of WS.

We have described how proxy servers can be used as a means for reducing network traffic in a publish/subscribe context, by subscribing on behalf of clients. Furthermore, we have discussed how we can use proxy servers for caching and filtering of information, in order to further reduce the amount of data sent over a network.

The idea behind information model optimization is to reduce the original amount of data to be transmitted, prior to compression. In our earlier work, we have used message-based information exchange, which implies sending a full operational picture each time. In this paper, we have therefore presented alternative mechanisms for information exchange, such as replication-based and query-based mechanisms.

We are currently implementing the concepts described in this paper, in order to demonstrate their feasibility in a DG.

## 6. References


[1] Network Centric Operations Industry Consortium (NCOIC)
http://www.ncoic.org/home

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

[3] Haakseth, R., Hadzic D., Lund, K., Eggen, A., Rasmussen, R. E., Experiences from implementing dynamic and secure Web services, 11th Coalition Command and Control in the Networked Era (ICCRTS), Cambridge, UK, September 2006

[4] Coalition Warrior Interoperability Demonstration (CWID) 2006 home page
http://www.cwid.js.mil/

[5] Performance Testing of STANAG 4406 (Military Messaging) Using IP over HF
Vivianne Jodalen, Bjørn Solberg, Ove Grønerud, Anton Leere
FFI/RAPPORT-2005/01183

[6] Squid Web Proxy Cache
http://www.squid-cache.org/

[7] Proxy Servers Tutorial – About Proxy Servers
http://compnetworking.about.com/cs/proxyservers/a/proxyservers.htm

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

[9] P. Niblett and S. Graham, Events and service-oriented architecture: The OASIS Web Services Notification Specifications, IBM Systems Journal, volume 44, no 4, 2005

[10] WEB Caching and Replication
Rabinovich and Spatscheck, Addison Wesley 2002, ISBN 0-201-61570-3

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

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

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

[14] Web Services in networks with limited data rate (in Norwegian)
Dinko Hadzic, Trude Hafsøe, Frank T. Johnsen, Ketil Lund, Kjell Rose
FFI/RAPPORT-2006/03886

[15] Zlib
http://www.zlib.net/

[16] Xmill
http://sourceforge.net/projects/xmill

[17] Efficient XML (EFX)
http://www.idealliance.org/proceedings/xml05/slides/schneider_exp.pdf

[18] Efficient XML Interchange Working Group
http://www.w3.org/XML/EXI/