UiO : Department of Informatics
University of Oslo

Improving the performance of Web Services in Disconnected, Intermittent and Limited Environments

Joakim Johanson Lindquister

Master's Thesis Spring 2016

Abstract

Lorem ipsum dolor sit amet, cu sed suas apeirian, decore iudicabit at per, pro ne lorem dicit dictas. Cu quo aeque maiorum gubergren, principes complectitur ei ius, numquam veritus minimum mel id. Ea ius vidit soleat. Mel timeam laoreet tractatos no.Pro an sadipscing efficiantur, esse ludus diceret nam in. Vis percipit probatus in. Est noster moderatius dissentiet te. Eirmod latine dissentias in sea, perfecto omittantur at duo, mea vide exerci ut. Nec euismod vocibus consectetuer eu.

Et fierent delectus sapientem eam, id eum dolore nullam. Cu his quod possit utamur, mel offendit copiosae forensibus ut, ius fabulas fierent sapientem an. Sed at vidit mentitum expetendis, utamur insolens ad cum, dicat dicta salutatus ei duo. Est te numquam explicari posidonium. Vim amet nostrud at, ea nam graece mediocritatem, cu fabulas maiorum nostrum vix. Ius id zril nullam aperiam, at sint corpora repudiandae eam.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Military units operate under conditions where the reliability of the network connection may be low. They can operate far from existing communication infrastructure and rely only on wireless communication. Such networks are often characterized by unreliable connections with low date rate and high error rates making data communication difficult. In a military scenario it is necessary for units at all levels to seamlessly exchange information across different types of communication systems. This ranges from remote combat units at tactical level, to commanding officers at operational level in a static headquarters packed with computer support. To the North Atlantic Treaty Organization (NATO), this concept is referred to as Network Enabled Capability (NEC). In a feasibility study, NATO identified the Service Oriented Architecture (SOA) paradigm and the Web Service technology as key enablers for information exchange in NATO[1].

Web service technology is well tested and in widespread use in civil applications where the network is stable and the data rate is abundant. However, certain military networks suffer from high error rates and very low date rate, which can leave Web services built for civilian use unusable. This thesis investigates how these challenges can be overcome by applying different optimization techniques. The main approach looks into how using alternative transport protocols other than HTTP/TCP may increase speed and reliability.

1.1 Background and Motivation

NATO is a military alliance consisting of 28 member countries [2] and which primary goal is to protect the freedom and security of its members through political and military means. In joint military operations the relatively large number of member countries can be a challenge when setting up machine-to-machine information exchange. Differences in communication systems and equipment attribute to making the integration of such systems more difficult. In order to address this issue, NATO has chosen the SOA concept, which when built using open standards facilitates interoperability[1].

1.1.1 Service Oriented Architecture

SOA is an architectural pattern where application components provide services to other components over a network. SOA is built on concepts such as object-orientation and distributed computing and aims to get a loose coupling between clients and services. In their reference model for SOA, the Organization for the Advancement of Structured Information Standards (OASIS) define SOA as [3]:

Service Oriented Architecture 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.

In SOA, business processes are divided into smaller chunks of business logic, referred to as *services*. A service can be business related, e.g a patient register service, or a infrastructure service used by other services and not by a user application. OASIS define a service as [3]:

A service is 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

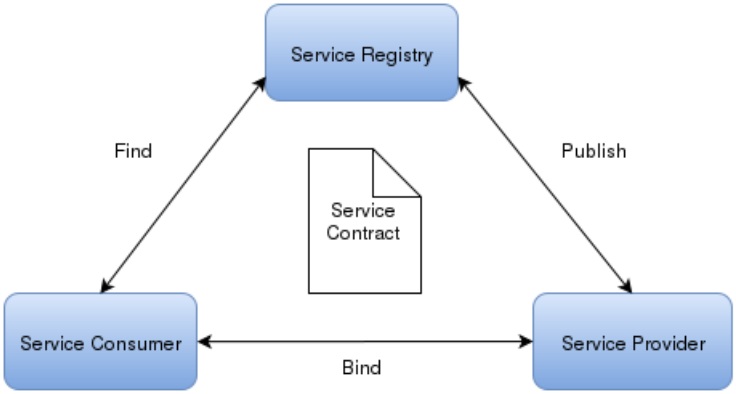

Figure 1.1: The three roles in SOA(from [4])

Services are provided by *service providers* and are consumed by *service consumers*. The service provider is responsible for creating a service description, making the service available to others and implementing the service according to the service description. Services are made available to service consumers through a form of *service discovery*. This can be a static configuration, or more dynamic with a central *service registry*, where service providers publish service descriptions. Service consumers find the services they need by contacting the service registry. The communication between services occur through the exchange of standardized messages.

Following the SOA principles dictates a very loose coupling between services and the consumers of those. This allows software systems to be more flexible, as new components can be integrated with minimal impact on the existing system. Another aspect of loose coupling is with regard to time, which enable services and its consumers to not be available at the same instance of time. This enables asynchronous communication. Loose coupling with regards to location allows the location of a service to be changed without needing to reprogram, reconfigure, or restart the service consumers. This is possible through the usage of runtime service discovery, which is dynamic retrieval of the new location of the service.

Furthermore SOA enables service implementation neutrality. The implementation of service is completely separated from the service description. This allows re-implementation and alteration of a service without affecting the service consumers. Thus this can attribute to keep development costs low and avoiding proprietary solutions and vendor lock-in. Another benefit with SOA is re-usability by dividing common business processes into services, which may help cost reduction and avoids duplication. SOA is only a pattern and the concepts can be realized by a range of technologies. The most common used approach is the Web service family of standards, using the SOAP messaging protocol.

To achieve interoperability between systems from different nations and vendors, NATO has chosen the Web service technology in order to realize the SOA principles[5]. This allows member nations to implement their own technology as long as they adhere to the standards. The Web service technology is discussed in detail in section 2.2. Another approach to realize SOA is Representational State Transfer (REST), an architecture style which uses HTTP over TCP. REST has gained a lot of traction in the civil industry and is discussed in section section 2.2.2.

However, employing Web service solutions directly into military use may not be straight forward. These technologies were not specifically designed to handle conditions found in certain military networks. In the following sections we discuss characteristics of such networks and the possible challenges of using Web services in them.

1.1.2 Military Networks

Military networks are complex and consist of many different heterogeneous network technologies. We can group them into layers, which have different characteristics as can been seen in fig. 1.2. At the highest level, there is fixed infrastructure and relatively static users, meaning that they seldom move around or disconnect. At the lower levels, there are fewer units, but they are much more dynamic. The lower level is called tactical networks, which is discussed in the next paragraph.

Tactical Networks

Tactical networks are characterized by that the units are deployed to operate in a battlefield, which means there is no existing communication in-

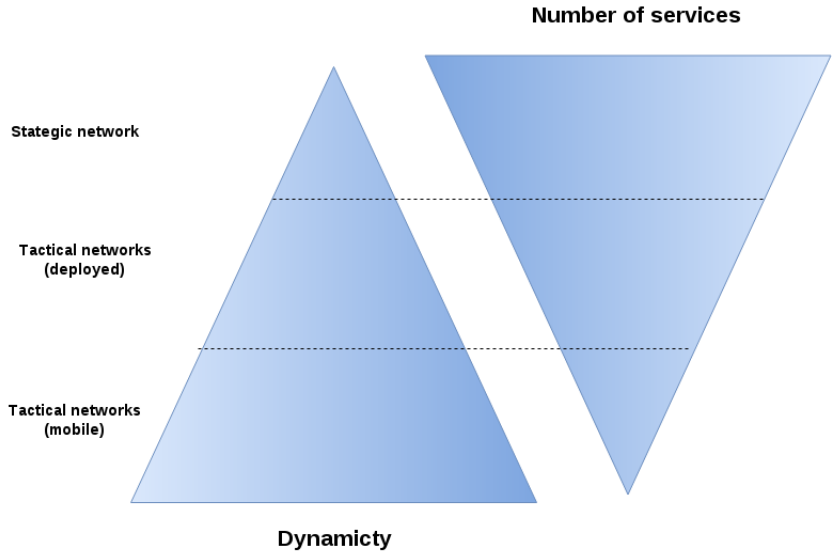

Figure 1.2: Complexity of military networks(from [6])

frastructure. They use tactical communication equipment, which includes technologies like VHF, UHF, HF, tactical broadband and satellites[4].

Examples of such units are mobile units like vehicles, foot soldiers and field headquarters. The tactical network connects deployed headquarters with mobile units. These types of networks are unpredictable and may have very low date rate, possibly high delay, high error rates and frequent disconnections. They are often called disadvantaged grids or Disconnected, Intermittent and Limited (DIL) environments, which is the term used in this thesis. DIL is discussed in section 1.1.3.

NATO studies[7] have identified such networks to have the following characteristics:

Disadvantaged grids are characterized by low bandwidth, variable throughput, unreliable connectivity, and energy constraints imposed by the wireless communications grid that link the nodes .

The characteristics of these networks and what challenges they impose are discussed in further detail in section 1.1.3.

1.1.3 Disconnected, Intermittent and Limited Networks

To improve the performance of Web services in limited military networks, we must understand what limitations we're dealing with. The DIL concept refers to three characteristics of a limited network. As we discussed in the introduction, military tactical networks may suffer from these constraints.

Disconnected Military units that participate in a tactical network are highly mobile and may disconnect from a network either voluntarily

or not. This causes topology changes. Unplanned loss of connectivity can be due to various reasons, such as loss of signal or equipment malfunction. The disconnected term refers to that nodes in the network may be disconnected for a long time, possibly for multiple days.

Intermittent Nodes in a DIL environment may lose connection temporarily before reconnecting. The duration range from seconds to minutes.

Limited The Data rate, how many bits that are sent per second, is limited in DIL networks. Various aspects that affects the date rate are discussed in the next section.

Other constraints

As well as being restricted by the communication link itself, military units may have other limitations as well. Consider that military foot patrols have limited battery capacity as they have to carry it with them in their backpacks. The transmission range of the communication equipment for mobile units may also be limited. Another factor that comes into play for military units is that in some cases they are required to enter radio silence in order to avoid being detected by the enemy. During such circumstances the soldiers may only receive data, but not send any.

1.2 Example scenario

Jeg tenkte her å introdusere et scenario som illustererer problemer og utfordringer med DIL nettverk.

1.3 A suggested solution

The Web service technology enables interoperability between systems, but also increase the information overhead, requiring higher data rate demands. Employing Web services developed for use in civilian networks directly into a DIL environment may not perform satisfactorily. To increase the performance we can apply different optimization techniques. The task-group(?) IST-090[4] investigated which improvements that could be made in order to get SOA applicable at the tactical level. They did not find a magic bullet that would solve all problems, but identified factors that would offer measurable improvements. The most important findings were:

- Foundation on open-standards.
- Ease of management and configuration.
- Transparency to the user.

- The Web services should optimized without the need to incorporate proprietary, ad hoc solutions that ensure tighter coupling between providers and consumers of services.

The last bullet point refers to the issue of that when we have identified optimization techniques, where do we place them? One approach is to modify the Web service application itself. However, this would mean that every application that is used in a tactical network would require modification. This would require a lot of resources and severely limit the flexibility of using Web services. Another solution is, by using proxies, we can apply the optimization there without altering the Web services themselves. The only thing required to do is to configure the application to send and receive data through the proxy. The proxy will take handle of the optimization for tactical networks. This approach is identified in IST-090 and is explored in this thesis.

1.3.1 Proxies

What is a proxy?

Possible proxy optimization. Compression, delay tolerance, overcome network disruptions.

1.4 Problem Statement

Most of the Web service solutions used today are aimed for civilian use and do not necessarily perform well in military environments. In contrast to civilian networks where the date rate is abundant, mobile tactical networks may suffer from high error rates and low date rate. Adapting Web service solutions meant for civil networks directly for military purposes may not be possible. Therefore, Web services needs to be adapted in order to handle network challenges. However, it can be very expensive to alter existing Web service technology and incorporate proprietary solutions. A NATO research task group has previously identified the foundation on open standards to avoid tighter coupling between service providers and consumers[4]. It is much better to use Commercial off-the-shelf (COTS) software. By placing the optimization in proxies, the Web services can remain unchanged.

The goal of this thesis is to investigate different optimization techniques that can be applied in order to improve Web service performance in DIL networks. In order for the clients and services to remain interoperable the optimization techniques will be placed in proxies. The Web services will communicate as normal, while all network traffic is tunneled through a proxy. The Web service itself does not need to pay attention to the bad connectivity, the proxy will choose the appropriate protocol and configuration.

1.5 Premises of thesis

The proxy developed as a part of this thesis shall support both REST and Web service communication between machines connected in a DIL network. In order to optimize Web services in DIL environments, the applications themselves should not be required to be customized, all optimization should be placed in proxies. This retains the interoperability with standardized solutions(COTS).

1.6 Scope and Limitations

The goal of this thesis is to investigate optimization techniques for Web services in DIL environments. Security is therefore not addressed in this thesis. However, applications that are to be used in military networks need to be approved by security authorities. If the application is too complex, e.g. it has a very large code base or use a lot of external frameworks, the approvement process will be very lengthy. It is therefore a important that the proxy is relatively simple. Furthermore, some security features such as IPSec will be enabled or disabled as part of the evaluation of the proxy.

When investigating optimization techniques, we limit it to techniques that can be applied at the application or the transport layer of the Internet protocol suite(see table 2.1). The reason for this is that NATO has previously decided "everything over IP", a statement describing that all data communication in NATO should occur with IP packets. We therefore limit our optimization possibilities to the mentioned layers.

Finally, the proxy implemented as a part of this thesis, only accepts Hypertext Transfer Protocol (HTTP) as input from the Web services. As we discuss later in section 2.2, most Web services use HTTP to communicate.

1.7 Research Methodology

Denning.

1.8 Contribution

The outcome of this thesis is a recommendation regarding which optimizations techniques can be used in DIL to enhance the performance of Web services. As well as a prototype implementation of a DIL proxy.

1.9 Outline

Hvordan er resten av oppgaven strukturert.

Chapter 2

Technical Background

In this chapter we present the technical background of the concepts and protocols that are central for this thesis. We first give a introduction to computer networks in general and how they are organized. Next, we discuss common Web services used for exchanging data in military systems. Then we look into a number of protocols that we can replace HTTP/TCP with in order to increase the performance of Web services. Finally, we present some available frameworks and solutions for creating a proxy.

2.1 Network layers

To reduce design complexity, networks are organized into layers, each one built upon the one below it. In the Internet Protocol Suite[8], networks is divided into 4 layers. As stated in the scope of this thesis, we only look into optimization techniques for the application and transport layer.

Application Layer
Transport Layer
Internet Layer
Link layer

Table 2.1: The layers of the Internet Protocol Suite

Link layer The lowest layer is the link layer, where link refers to the to the physical network component used to interconnect nodes in a network. Link layer protocols operate between adjacent network nodes. An example of a link layer protocol is Ethernet.

Internet Layer Where the link layer is only concerned of moving data over a wire to an adjacent node, the Internet layer is concerned of how to deliver data all the way from a source to a destination, possible passing through multiple nodes on its way. It does not guarantee delivery of data, since data can be lost on the way to the destination. Guaranteed delivery is usually handled on the higher levels of the Internet Protocol Suite.

The core protocol of the Internet layer is Internet Protocol (IP) and its routing function enables sending data over interconnected networks.

Transport layer In the Internet protocol suite model, the transport layer provides end-to-end communication services for applications. It builds on top on the network layer, and takes responsibility of sending data all the way from a process on a source machine to a process on the destination machine. The far most used transport protocol is the Transmission Control Protocol (TCP), which provides reliable transport of data to applications. With reliable transport we mean that if data in transmission is lost or received in the wrong order, this is all handled by the transport protocol. This provides an important abstraction for applications so that they don't need to deal with the characteristics of the physical network itself.

Application layer The top layer is the application layer and is where applications real user use reside. The other layers provide transport services to applications found in the application layer. When we talk about application layer protocols, we talk about protocols that applications use to communicate with other applications. Application layer protocols use the communication services the transport layer provides. Examples of application layer protocols is HTTP and File Transfer Protocol (FTP).

2.2 Web services

Web services are client and server applications that communicate over a network and can be used to implement a service-oriented architecture. Web services are critical in any data systems and are in widespread use in both civilian and military systems. It is a broad term and can be used to describe different types of services that are available over a network. The most common usage of the term refers to the World Wide Web Consortium (W3C) definition of SOAP-based Web services, but could also refer to more simple HTTP-based REST services.

In this thesis we investigate optimization techniques that should support both W3C Web services and RESTful web services.

2.2.1 W3C Web services

W3C has defined Web services as [9]:

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.

This definition points out a set of standards that enables machine-to-machine interactions. All communication is based on sending XML-based SOAP messages. It exists many definitions of Web services where the core principles are the same, but the finer details may vary. These standards are discussed in the following sections. The Web service technology is realization of the SOA principles, which provides loose coupling and ease integration between systems.

Figur her.

XML

The Extensible Markup Language (XML) is considered as the base standard for Web services. An XML document consist of data surrounded by tags and is designed to be both machine and user readable. Tags describe the data they enclose. The tags can be standardized, which allows exchange and understanding of data in a standardized, machine-readable way.

Service descriptions: WSDL

Web Services Description Language (WSDL) is an interface definition language that using XML describes functionality offered by a Web service. The interface describes available functions, data types for message requests and responses and binding information about the transport protocol, as well as address information for locating the service. This enables a formal, machine-readable description of Web service which clients can invoke.

SOAP

SOAP (SOAP) is an application level XML-based protocol specification for information exchange in the implementation of Web services. It is transport protocol agnostic and can be carried over various protocols. The far most used transport protocol is HTTP over TCP, but other protocols such as UDP and SMTP can be used as well. A SOAP message is an "envelope" consisting of an optional header and a required body. The header can contain information not directly related to the message such as routing information for the message and security information. The body contains the data being sent, known as the payload.

W3C Web service overhead

W3C Web services are associated with a considerable amount of overhead. Web Service technology is based on SOAP, which use XML-based messages. It is a textual data format and produce much larger messages than binary formats.

2.2.2 Representational State Transfer

There also exist other types of Web services which does not follow the previously discussed standards. REST is an architectural style which let

users manipulate data using a set of stateless operations. It is based on a client-server model where a client requests data from a server when needed. It is closely associated with HTTP and use HTTP verbs(e.g GET, POST, DELETE) to operate on resources on a server.

REST is easy to understand and has gained a lot of traction in the civil industry in the latest years. REST uses exclusively HTTP over TCP. However, TCP does not necessarily perform satisfactorily in DIL environments, which limits the usability in tactical networks(trenger kilde?). It also lack standardization, which may cause interoperability issues.

2.3 Hypertext Transfer Protocol

As we have seen in the previous section, both restful and w3c Web services utilizes the HTTP as their way to communicate with other services. The usage of HTTP is very widespread and it is the foundation of data communication for the World Wide Web since the early 90's. It's protocol specification is coordinated by Internet Engineering Task Force (IETF) and the W3C, and is defined as[10]:

The Hypertext Transfer Protocol (HTTP) is an application-level protocol for distributed, collaborative, hypermedia information systems. It is a generic, stateless, protocol which can be used for many tasks beyond its use for hypertext, such as name servers and distributed object management systems, through extension of its request methods, error codes and headers

HTTP started out as a simple protocol for raw data transfer across the Internet and has since been updated in HTTP/1.0, HTTP/1.1 and most recently a major update with HTTP/2.0. It is a request-reply protocol which means that all data exchanges is initiated with a client invoking a HTTP-request and then waits until a server responds with a HTTP response. A HTTP-request consist of the request method, URI, protocol version, client information, and a optional body. The server responds with a message containing a status line, protocol version, a code indicating the success or error of the request, and a optional body. Both HTTP requests and responses use a generic message format and can contain zero or more HTTP headers. Headers are used to provide information about the request/reply or about the message body, e.g information about the encoding and caching information.

HTTP, being an application level protocol, relies on a transport protocol to actually transfer data to an another machine. HTTP communication most often, but not necessarily, occurs over TCP/IP connections. The only requirement is that a reliable transport protocol is used.

HTTP methods

GET, HEAD, POST, PUT, DELETE.

2.4 Transmission Control Protocol

TCP is called the workhorse of the Internet because it is so critical for how the Internet works. It is the primary transport protocol of the Internet Protocol Suite[8] and provides reliable, in-sequence delivery of two-way traffic(full-duplex) data. In this subsection we present the characteristics of TCP and some of the issues we may encounter working with it.

TCP was defined in RFC 793[11] back in September 1981 and has since been improved in various RFC's. The main motivation behind TCP was to provide reliable end-to-end byte streams over unreliable networks.

Reliablility When transferring data over the Internet, the data may pass through various networks, routers and physical networks. Some of the routers may be not working correctly, a bit may be flipped when transferring data wirelessly, or some other factor may come in to play. For those reasons, we have to accept that some of the data will be damaged, lost, duplicated or delivered out of order.

TCP recovers from such faults by assigning sequence number to each packet being sent. It then requires an positive acknowledgement from the receiver that the data was actually received. If the acknowledgement is not received within a timeout interval, the data is transmitted again. For the receiver the sequence numbers are used to ensure that data is received in the correct order, as well as eliminating duplicates. Furthermore, to detect damaged data, TCP applies checksums to each segment transmitted. At the receiver the checksum is then checked and damaged segments are discarded.

Flow Control If a fast receiver sends data faster than a slow receiver is able to process, the receiver will be swamped with data and may experience serious performance reduction. Flow control is a mechanism to manage the rate of the data transmission to avoid overflowing a receiver. TCP provides this by using a window of acceptable sequence numbers that the receiver is willing to accept. With every acknowledgement sent back to the sender, the window is specified. This allows the receiver to control which segments, and how fast, the sender can send.

Connection TCP is connection-oriented, which means that a connection between a sender and the receiver must be established before any data can be transfered. A connection is specified by a pair of sockets identifying its two sides. Associated with each connection TCP initialize and maintains some status information for each connection. This includes window size, socket information and sequence numbers.

Protocol Computers supporting TCP has a piece of software, which manages TCP streams and interfaces to the IP layer. Most often this software is a part of the kernel[12]. It accepts data streams from local processes, and breaks them up into pieces, before sending them to the IP layer. The pieces are called TCP segments, which consist of a fixed 20 byte header, followed by zero or more data byte. The TCP software decide how big the segment should be, but for performance reasons they should not exceeds the Maximum Transfer Unit (MTU) of the link(the physical network). Each segment should be so small that they can be sent in a single, unfragmented package over the entire network. This usually limits the size of each segment to the MTU of the Ethernet, which is 1500 bytes.

When the TCP software receives data from applications, it is not necessarily sent imediately as it may be buffered before its sent. At the receiver, data is delivered to the TCP software, which reconstructs the original byte streams and deliver them to the target application.

2.5 Protocols of interest

In order to improve the performance of Web services in DIL environments we have investigated the usage of alternative transport protocol other than TCP can be utilized. In this thesis we're looking into protocols in the transport and application layer of the Internet Protocol Suite[8].

In the following sections we will give a short introduction to the protocols we're investigating in this thesis.

2.5.1 The Constrained Application Protocol

The Constrained Application Protocol (CoAP) is a specialized web transfer protocol designed for use with constrained nodes(low power) and constrained networks(lossy)[13]. It is designed for machine-to-machine applications, typically in the Internet of Things. CoAP was standardized in RFC-7252 as late as in June 2014.

It is based on the REST model, where the server makes resources available under a URL. Clients access these resources using the HTTP-verbs GET, PUT, POST and DELETE. Designed to use minimal resources, both on the device and on the network.

- Application level protocol.
- Can carry any data format.
- UDP on IP.
- Simple binary base header format

2.5.2 Advanced Message Queuing Protocol

Advanced Message Queuing Protocol (AMQP) is a messaging middleware that can utilize different transport protocols.

- Support both request/response and publish/subscribe communication paradigms.

- Reliable when facing network disruptions, since it employs a broker-based architecture with store-and-forward capabilities.

2.5.3 MQTT

MQTT is a client server publish/subscribe messaging transport protocol [14]. It is considered lightweight and is designed for use in networks where the bandwidth is limited. It is broker-based, where the broker is responsible for delivering messages to clients based on the topic of a message.

2.5.4 Stream Control Transmission Protocol

Stream Control Transmission Protocol (SCTP) is transport-layer protocol, which offers functionality from both User Datagram Protocol (UDP) and TCP.

- Message-oriented like UDP.

- Ensure reliable, in sequence transport of messages with congestion control like TCP.

- Multi-homing and multi-streaming.

2.5.5 UDP

WSReliability? For rest må det da bygges inn støtte i proxien. Reliable UDP. Implmentasjon av UDP som er reliable. En gammel protokoll?

- No mechanisms for flow control, packet ordering or integrity of messages.

2.6 Proxies

A proxy is an application which acts as an intermediary between an client and a server. Proxies are widely in use and their usage and type varies. Example of usage is load balancing, caching and security. Web proxies are proxies that forward HTTP requests, which is what we are investigating in this thesis.

In this section we will briefly present available popular web proxies.

2.6.1 Apache

2.6.2 Squid

2.6.3 Apache Camel

This does the work of converting the specific input/request (like an HTTP request) into something generic - a Camel Exchange - that can travel down a Route. - quoute. må skrives om.

2.6.4 Nginx

2.7 Performance testing

To determine which optimization techniques that have a positive effect on the performance of Web services in DIL environments, we can do performance testing.

2.7.1 Network metrics

Network metrics are used to describe various aspects of data transfer from a point to another.

Link throughput The link throughput is influenced by how large distance there is between the units communicating.

Link reliability How much of the arriving data that is correct. This is called *bit error rate* or *packet error rate*. With high error rates, more data to be transmitted again due to the data arriving being incorrect. This contributes to longer transmission time. In a military setting, an enemy may deliberate sabotage the network with jamming, causing higher error rates.

Link latency The communication technology in use influences how fast data transmission can be done. Long delay may cause that the application sending data timing out.

2.8 Summary

Summary of background chapter here.

Protocol	Summary
HTTP over TCP	Widely used. Breaks down in DIL environments.
CoAP	Application level protocol designed for use in the Internet of Things.
AMQP	Messaging middleware with store-and-forward capabilities.
MQTT	Summary here
SCTP	Similar to UDP but also provide reliable, in sequence transport of messages like TCP.
TCP	The well-known transport protocol.
UDP	Lacks reliability, but frameworks exist that provides it.

Table 2.2: Protocols of Interest

Chapter 3

Related Work

In this chapter we will discuss earlier relevant work in the area of improving the performance of Web services in DIL environments. We identify results and recommendations that are applicable to this thesis. Furthermore, we discuss existing proxies and what they offer.

Quite an amount of research has been done in the area of compression. Also, improving the performance of Web services in DIL environments has been explored, but mostly for SOAP-based Web services.

In the report IST-090, a task group investigated solutions for making SOA applicable at the tactical level. Three key issues that needs to be addressed in order to apply Web services in tactical networks were identified[15, 4]:

End-to-end connections

Web services depend on a direct, end-to-end connection between the client and the service. Attempting to establish and maintaining connections in DIL environments can lead to increased communication overhead and possible complete breakdown of communication. Most Web services use TCP as the transport protocol, which is a connection-oriented protocol designed for wired networks. In DIL environments with high error rates and high latencies, the congestion control of TCP will cause sub-optimal utilization of the network due to frequent connection timeouts. Similar, HTTP, which is the application layer protocol most often used together with TCP, struggles in such environments. HTTP is a synchronous protocol, which means that the HTTP connection is kept open until a response is received. Long response times cause timeouts. IST-090 points out the obvious solution to replace HTTP and TCP with other, more suitable protocols.

The report[4] mentions two approaches to replace HTTP/TCP. The clients and services themselves can be modified to support other protocols, or proxies which support alternative protocols can be used. With employing a proxy solution, standards compliance can be retained.

IST-090 så ikke på rest i det hele tatt. SOAP-spesefikt.

Network heterogeneity

Another issue is when heterogeneous networks are interconnected. Different performance in networks may lead to buildup of data in buffers, risking loss of information. A proposed solution to this is to have store-and-forward support, which can support that messages are not dropped, but stored and forwarded when possible.

Web service overhead

Optimization approaches should seek to reduce the network traffic generated by Web services by using techniques as compression to reduce the size of messages. Another approach is to reduce the number of messages being sent, which was looked into in IST-090[4]. In their work they investigated three different ways to do this:

1. Employing caching near the client in order to reuse older messages.
2. Using publish/subscribe paradigm, which allows clients to subscribe to information instead of requesting it. This allows the same message to be sent to multiple clients.
3. Employing content filtering, which filters out unnecessary data.

3.1 Proxies

This section lists previous implementations of proxies designed to work in DIL environments.

Hva som er spesielt med min proxy. Den støtter både rest og soap.

3.1.1 DSProxy

DSProxy is a proxy solution developed by Norwegian Defence Research Establishment (FFI), which transports SOAP messages over DIL networks. It reduces bandwidth needs by employing different optimization techniques such as compression. It also provides delay tolerance, which allows COTS clients to function in DIL networks.

3.1.2 AFRO

Adaption Framework foR Web Services prOvision (AFRO) is an edge proxy which offers different levels of Quality of Service (QoS) to Web Services through performance monitoring and application of the context-aware service provision paradigm. It perform so called adaption actions, which modifies the SOAP XML messages by changing their encoding to more efficient data representation. It also cuts out information that is accepted to be removed by the service requester. However, since the proxy modifies the data being sent, the checksum of the data is also changed. In applications where we want to be sure that no one has tampered with the data before

arriving, checksums are often used. Therefore this solution would not work for such applications.

3.1.3 Suri

To be done.

3.2 Evaluation of Transport Protocols

Previous studies have investigated potential gains from replacing HTTP/TCP with alternative transport protocols [16]. They looked into how TCP, UDP, SCTP and AMQP for conveying Web services traffic under typical military networking conditions.

- SCTP has the highest success rate in military tactical communication. However, on the lower bandwidth links the protocols tends to generate more overhead than TCP. Due to SCTP has a more complex connection handshake procedure and in addition use heartbeat packets.

3.2.1 Configuration

IST-90: Configure HTTP on the application server or ESP to prevent time-outs. Anbefaler at hvis man trenger å gjøre propetiære optimaliseringer, så burde de plasseres i proxier.

3.3 Compression

Data compression is the technique of encoding information using fewer bits than the original representation. The goal is to reduce data transmission time or the storage requirements. We divide compression lossy and lossless compression. Lossy compression is used to compress data such as images and movies where the consequence of loosing some of the data is not critical. Lossless compression utilize repeating patterns in the data in order to represents the same data in a more efficient way.

XML is the data format used by Web services and has a significant overhead. Previous studies have evaluated different compressions algorithms for Web services. Previous experiments shows EFX has the best compression results with GZIP as the second best alternative[17].

3.4 Summary

As discussed in this chapter, there exist research and experiments targeting SOAP-based Web services in DIL environments. Proxies have been created but their are either limited to SOAP-based Web services or are inadequate to be used due to security reasons.

Approach	**Summary**
Build from scratch	Time consuming, complex.
Apache Camel	Supports some of the protocols
Squid	

Table 3.1: Possible proxy approches

Chapter 4

Requirement Analysis

In this chapter we discuss the requirements for our proxy. Since we're creating a proxy aimed at use in a military context, military operational requirements are part of the requirements for this proxy. As discussed earlier this proxy must provide robustness and reliable delivery of data in DIL environments.

The first requirements are the premises we discussed in the introduction. The proxy should be able to receive and forward HTTP-requests.

Next, the proxy must be able to handle the difficult network conditions of dil. It must handle very low data rates and handle brief disconnects. It should also support disconnects over a longer period of time by having store- and-forward capabilities.

Payload agnostic. The data type doesn't matter(JSON, XML etc.).

In order to perform compression the proxy must be able to modify the payload of the message. Due to security mechanisms that detect changes to the payload(checksums), the payload must be restored back to its original form before being forwarded to the final receiver.

As discussed in chapter 3, the dependency on end-to-end connections needs to be removed. This can be done by adding proxies to the network. Mobile units have to carry batteries with them and the capacity is therefore limited. Advanced compression techniques may reduce the overhead, but also requires more battery. This trade-off needs to be considered.

Requirement	**Priority**
Receive and forward HTTP requests	1
Allow modifications on the payload	1
Retain the checksum of the payload	1
Allow configuration of HTTP timeouts	1
Keep HTTP-connection alive	1
Support protocol X and y	2
Handle very low data rate	1
Have store-and-forward capabilities	1
Handle frequent disconnects	1

Table 4.1: Summary of proxy requirements

4.1 Optimization techniques

As we saw in the previous chapter there has been quite amount of research into optimization techniques in DIL environments. In this section we summarize which techniques we support in our proxy.

Protocol Stack	**optimization possibilities**
The application	Optimize the application
Web service messaging: SOAP	Optimize SOAP, e.g XML compression
HTTP/TCP, UDP or other transport protocols	SOAP is transport agnostic. Other protocols can be used.
IP	NATO NEC feasibility study states that all protocols should be over IP.
Lower layers	Not in the scope of this thesis.

Table 4.2: Optimization possibilities.

4.1.1 Compressing the payload

The first optimization techniques deals with the optimization of the encoding. By compressing the Web service payload, we can reduce the amount of data that need to be sent. This optimization technique addresses one of the many challenges of tactical networks, namely the bandwidth consumption due to large message sizes.

- GZIP
- EFX(Efficient XML). XML spesifikt, får ikke brukt på REST når vi har andre payloads..

4.1.2 Reducing overhead of SOAP

HTTP/TCP is the most used transport protocol for SOAP messages, but since SOAP is transport protocol agnostic different protocols can be used. Experiments show that this is possible.

4.2 Summary

In this chapter we have discussed the requirements for our proxy. See table xx for a summary. Next we discuss the design and implementation of our proxy.

Chapter 5

Design and Implementation

In this chapter we will introduce the design and implementation details of our proxy solution for improving the performance of Web services in DIL environments. We will first look into the overall design before we dive into the details.

5.1 Overall Design

The proposed design in this thesis includes a proxy pair.

5.2 Proxy

5.2.1 Apache Camel

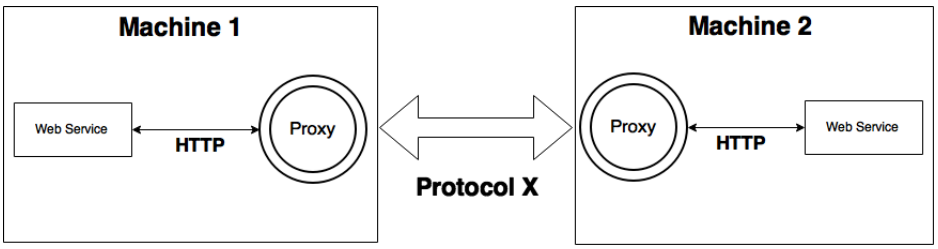

Figure 5.1: Architectural overview of proposed design

5.3 Summary

Chapter 6

Testing and Evaluation

To answer our research question of how to improve the performance of Web services in DIL environments, we developed some test scenarios. In this chapter we present how the testing was performed and then present the results we obtained.

6.1 Test Scenarios

1. REST-client and server sending each other GET requests.
2. W3c Web services.

Both the REST and Web service applications were developed with Java.

6.1.1 Using proxies

In order to enable the applications to tunnel all their HTTP traffic through our proxy, we needed a way to setup a proxy without altering the applications themselves. Fortunately, Java provide mechanisms to deal with proxies[18]. We configured the Java Virtual Machine (JVM) to get the applications to tunnel all HTTP traffic through our proxy. This is done by setting properties to the JVM:

Listing 6.1: "Setting a proxy on the JVM"

```
java -Dhttp.proxyHost=localhost \
-Dhttp.proxyPort=3001 \
-Dhttp.nonProxyHosts= \
-jar target/client.jar
```

In listing 6.1 the application **client.jar** is started and all HTTP traffic will go through the proxy server at localhost on port 3001.

6.2 Network Emulator

In order to simulate DIL environments we need some way to control the properties of the network traffic. Fortunately, the Linux kernel offers a rich

set of tools for managing and manipulating the transmission of packets.

Network Emulator (NetEm) is an enchancement of the traffic control facilities that allows us to control delay, packet loss and other characteristics to packets outgoing from a selected network interface.

6.2.1 NetEm emulating

tc(traffic control) is a linux program to configure and control the linux kernels Network scheduler.

Delays

NetEm can emulate delays on packets on a specific link.

Listing 6.2: "Emulating delay"

```
tc qdisc add dev etho root netem delay 100ms
```

In this example we add a fixed delay on 100 ms to all packets going out of local Ethernet.

6.3 Testing environment

The majority of testing was performed on machines located at the FFI-lab at Kjeller. Although testing on regular machines gives us a indication, to get as realistic results as possible we also performed tests on military communication equipment.

6.3.1 Testing on computers

Describe the test setup.

6.3.2 Testing on real hardware

Kongsberg radio.

6.4 Evaluation Tools

6.5 Test results

In this section the results from the tests are presented. These results lead up to the discussion and conclusion in the next chapter.

Chapter 7

Conclusion and Future Work

7.1 Conclusion

7.2 Future Work

Bibliography

[1] P. Bartolomasi et al. *NATO network enabled capability feasibility study*. 2005.

[2] NATO. *NATO - Member Countries*. http://www.nato.int/cps/en/natohq/nato_countries.htm. Accessed: 2015-05-04.

[3] OASIS et al. *Reference Model for Service Oriented Architecture 1.0 OASIS standard*. http://docs.oasis-open.org/soa-rm/v1.0/soa-rm.pdf. Accessed: 06. 10. 2015. Oct. 2006.

[4] F. Annunziata et al. *IST-090 SOA challenges for disadvantaged grids*. https://www.cso.nato.int/pubs/rdp.asp?RDP=STO-TR-IST-090. Apr. 2014.

[5] NATO C3 Board. *Core Enterprise Services Standards Recommendations - The SOA Baseline Profile*. 1.7. 2011.

[6] Frank T. Johnsen. "Pervasive Web Services Discovery and Invocation in Military Networks". In: *FFI-rapport 2011/00257* (2011).

[7] A. Gibb et al. "Information Management over Disadvantaged Grids". In: *Task Group IST-030/ RTG-012, RTO-TR-IST-030* (2007). Final report of the RTO Information Systems Technology Panel.

[8] R. Braden. *RFC 1122 – Requirements for Internet Hosts – Communication Layers*. https://tools.ietf.org/html/rfc1122. Accessed: 06. 01. 2016. Oct. 1989.

[9] Hugo Haas and Allen Brown. *Web Services Glossary*. http://www.w3.org/TR/ws-gloss/\#webservice. Accessed: 2015-05-06.

[10] R. Fielding et al. *RFC 2616 – Hypertext Transfer Protocol – HTTP/1.1*. https://tools.ietf.org/html/rfc2616. Accessed: 10. 02. 2016. June 1999.

[11] Information Sciences Institute - University of Southern California. *RFC 793 – Transmission Control Protocol*. https://tools.ietf.org/html/rfc793. Accessed: 10. 02. 2016. Sept. 1981.

[12] David J. Wetherall Andrew S. Tanenbaum. *Computer Networks*. Fifth Edition. Pearson New International Edition.

[13] Z. Shelby et al. *RFC 7252 – The Constrained Application Protocol (CoAP)*. https://tools.ietf.org/html/rfc7252. Accessed: 10. 02. 2016. June 2014.

[14] OASIS, Andrew Banks, and Rahul Gupta. *MQTT Version 3.1.1 Specification*. http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/mqtt-v3.1.1.html. Accessed: 06. 01. 2016. Oct. 2014.

[15] Frank T. Johnsen et al. "IST-118 - SOA recommendations for Disadvantaged Grids in the Tactical Domain". In: *18th ICCRTS* (2013).

[16] Frank T. Johnsen et al. "Evaluation of Transport Protocols for Web Services". In: *MCC 2013* (2013).

[17] Frank T. Johnsen and Trude Hafsøe. "Using NFFI Web Services on the tactical level: An evaluation of compression techniques". In: 13th International Command and Control Research and Technology Symposium (ICCRTS). Seatle, WA, USA, 2008.

[18] Oracle. *Java Networking and Proxies*. https://docs.oracle.com/javase/8/docs/technotes/guides/net/proxies.html.

Acronyms

AFRO Adaption Framework foR Web Services prOvision. 30

AMQP Advanced Message Queuing Protocol. 25

CoAP The Constrained Application Protocol. 24

COTS Commercial off-the-shelf. 16, 30

DIL Disconnected, Intermittent and Limited. 14–17, 22, 24, 29–31, 33, 34, 37

FFI Norwegian Defence Research Establishment. 30

FTP File Transfer Protocol. 20

HTTP Hypertext Transfer Protocol. 17, 20, 22

IETF Internet Engineering Task Force. 22

IP Internet Protocol. 20

JVM Java Virtual Machine. 37

MTU Maximum Transfer Unit. 24

NATO North Atlantic Treaty Organization. 11

NEC Network Enabled Capability. 11

NetEm Network Emulator. 38

OASIS Organization for the Advancement of Structured Information Standards. 12

QoS Quality of Service. 30

REST Representational State Transfer. 13, 20, 22

SCTP Stream Control Transmission Protocol. 25, 31

SOA Service Oriented Architecture. 11–13, 21, 29

SOAP SOAP. 21

TCP Transmission Control Protocol. 20, 23, 25

UDP User Datagram Protocol. 25

W3C World Wide Web Consortium. 20, 22

WSDL Web Services Description Language. 21

XML Extensible Markup Language. 21, 31