FFI-rapport 2012/02494

Quality aspects of Web services

Frank T. Johnsen, Trude H. Bloebaum, and Marianne R. Brannsten

Norwegian Defence Research Establishment (FFI)

14 December 2012

FFI-rapport 2012/02494

1176

P: ISBN 978-82-464-2199-5
E: ISBN 978-82-464-2200-8

Keywords

Tjenesteorientert arkitektur

Nettverksbasert Forsvar

Web services

Approved by

Rolf Rasmussen	Project Manager
Anders Eggen	Director

English summary

This report is a compilation of published papers which cover aspects related to Web services quality. Quality-of-Service (QoS) is often said to deal with the properties of a system that address non-functional requirements. This encompasses topics such as improving the users' experience of the systems, ensuring that the system uses its resources in the best available manner, ensuring that the systems functions correctly, and so on. This report discusses QoS support for Web services, and summarizes research in that area performed by FFI project 1176 by including relevant external publications in appendices.

Sammendrag

Denne rapporten er en artikkelsamling med introduksjon som dekker arbeid relatert til Web services og kvalitet. Tjenestekvalitet (QoS) kan sies å være de sidene av et system som det stilles krav til ut over den grunnleggende funksjonaliteten. Dette omfatter slike ting som å forbedre en brukers oppfattelse av bruken av systemet, å sikre at systemet bruker sine ressurser på best mulig måte, å sikre at systemet fungerer som det skal, osv. Denne rapporten tar for seg tjenestekvalitetsstøtte for Web services, og oppsummerer forskning som FFI-prosjekt 1176 har foretatt på dette området. Relevante eksterne publikasjoner er gjengitt som vedlegg for å utdype aspektene som diskuteres.

Contents

1 Introduction

The NATO organization is focusing on NATO Network Enabled Capabilities (NNEC) as a means to more efficiently utilizing available resources both within and across system- and national borders. The aim of NNEC is to increase mission effectiveness by networking military entities. Mission effectiveness does however greatly depend on the participants' ability to interact with each other in an efficient manner.

Having a common overview of a situation, and a common understanding of the task at hand, are key factors when it comes to efficient communication between people. There are of course many other factors that affect the interaction between people, but having the ability to share information in an efficient manner allows for creating a shared situational awareness that can reduce some of the challenges involved. Achieving technical interoperability, which is the interoperability between communication systems and application such as Command and Control (C2) systems, will require a common information infrastructure for NNEC.

Today, this interoperability between national systems usually takes place at the brigade level or higher. An information infrastructure designed to support NNEC operations must however fulfill all the communication requirements of the member nations' forces, including horizontal information exchange between systems at lower levels.

A common information infrastructure for NNEC will most likely consist of a federation of networks at different operational levels that will be required to work together. Not only will these systems have different network characteristics, such as data-rates, mobility, broadcast and error probability, but they will also be owned and managed by the individual coalition partners. The systems used within the different nations might not be directly compatible with each other, so achieving interoperability between the systems will require the use of interface points between them.

In order to achieve this cross-system interoperability, one needs to abstract away from the implementation details of the individual systems, and provide a loosely coupled framework that allows for communication through clearly defined interfaces. This allows the individual nations to choose and manage their own systems independently, while still allowing for an easy integration of systems.

Service Oriented Architecture (SOA) is an architectural paradigm that is based on this principle of loosely connected systems, which are integrated using standardized protocols through predefined interfaces. Web services, the most commonly used technology for implementation of the SOA principle, have been identified as a key enabling technology for the NNEC information infrastructure. Web services are standard based, but not all aspects of Web service implementation are covered by these standards. Also, the existing standards often exist in several versions, have optional elements or are lacking in detail. In order to ensure compatibility between the individual national systems, it is vital that all coalition partners utilize the standards in the same manner.

This requires the development of interoperability profiles, which must be agreed upon through for instance NATO standardization.

One example of such work is an ongoing NATO work group that defines a set of Core Enterprise Services, which may be used alone or as building blocks for more advanced services [9]. Employing SOA is particularly challenging in dynamic environments such as military tactical systems. Our previous research has shown that it is feasible to utilize Web services in military networks [37], provided you know their location. However, when the location is not known in advance, it is important to have service discovery solutions that are interoperable in order to be able to find and utilize services at different operational levels across heterogeneous networks. The differences between the systems used by coalition partners imply that it will not be possible to use a "one size fits all" solution for service discovery. Different solutions will have to be used for different types of networks in order to best utilize the network characteristics [29]. We have suggested a gateway approach to achieve pervasive service discovery, as discussed in [27], along with a series of optimization techniques to increase the usability of Web services in disadvantaged grids (e.g., using compression).

Quality-of-Service (QoS) is a term that can mean different things to different people. The most common usage of the term comes from networking, where it is used to describe the ability to reserve network resources, or to describe mechanisms that allow one to divide network traffic info classes which can receive differential treatment by the network.

When it comes to Web services, the term QoS has a much wider definition, which corresponds more closely to the QoS definitions used within multimedia systems. The following QoS definition was coined to describe such multimedia systems, but it can be argued that it covers the need for QoS in military networks as well, because there is also a need to disseminate multimedia (e.g., text, video, audio) in such networks. Thus, it is the QoS definition that we will be using throughout this document.

> QoS is the set of those quantitative and qualitative characteristics of a distributed multimedia system necessary to achieve the required functionality of an application [62].

One interesting aspect of this definition is that it uses the expression *functionality*; QoS is often said to deal with the aspects of a system that addresses requirements beyond the basic functionality a system offers. These aspects address topics such as improving the users' experience of the systems, ensuring that the system uses its resources in the best available manner, ensuring that the systems functions correctly, and so on.

1.1 Quality aspects of Web services

For Web services, the functional aspects of a service are described by the service description in the form of a WSDL document. The non-functional aspects, however, can be summarized as follows [39]:

- *Availability* is the quality aspect of whether the Web service is present or ready for immediate use. Availability represents the probability that a service is available. Larger values represent that the service is always ready to use while smaller values indicate unpredictability of whether the service will be available at a particular time. Also associated with availability is time-to-repair (TTR). TTR represents the time it takes to repair a service that has failed. Ideally smaller values of TTR are desirable.
- *Accessibility* is the quality aspect of a service that represents the degree it is capable of serving a Web service request. It may be expressed as a probability measure denoting the success rate or chance of a successful service instantiation at a point in time. There could be situations when a Web service is available but not accessible. High accessibility of Web services can be achieved by building highly scalable systems. Scalability refers to the ability to consistently serve the requests despite variations in the volume of requests.
- *Integrity* is the quality aspect of how the Web service maintains the correctness of the interaction in respect to the source. Proper execution of Web service transactions will provide the correctness of interaction. A transaction refers to a sequence of activities to be treated as a single unit of work. All the activities have to be completed to make the transaction successful. When a transaction does not complete, all the changes made are rolled back.
- *Performance* is the quality aspect of Web service, which is measured in terms of throughput and latency. Higher throughput and lower latency values represent good performance of a Web service. Throughput represents the number of Web service requests served at a given time period. Latency is the round-trip time between sending a request and receiving the response.
- *Reliability* is the quality aspect of a Web service that represents the degree of being capable of maintaining the service and service quality. The number of failures per month or year represents a measure of reliability of a Web service. In another sense, reliability refers to the assured and ordered delivery for messages being sent and received by service requester and service providers.
- *Regulatory* is the quality aspect of the Web service in conformance with the rules, the law, compliance with standards, and the established service level agreement. Web services use a lot of standards such as SOAP, UDDI, and WSDL. Strict adherence to correct versions of standards (for example, SOAP version 1.2) by service providers is necessary for proper invocation of Web services by service requester.
- *Security* is the quality aspect of the Web service of providing confidentiality and non-repudiation by authenticating the parties involved, encrypting messages, and providing access control. Security has added importance because Web service invocation occurs over the public Internet. The service provider can have different approaches and levels of providing security depending on the service requester.
- *Priority* is a means to distinguish between different types of traffic and ensure that each request is handled appropriately according to the business value it represents. Service providers can provide differentiated servicing by prioritizing different customers and service

types, and by ensuring appropriate QoS levels for different applications and customers. For example, one service might require good throughput, whereas another might require security and transactional support.

In addition to the aspects mentioned above, there are a number of additional aspects introduced by [53]:

- Service Time – What is the expected execution time of the service?
- Execution Price / Cost – How much does it cost to use the service?
- Transaction – Does the service support transactional properties?
- Capacity – The number of concurrent consumers a service can support
- Reputation – The reliability of as service, as seen from an end-user perspective

These aspects have their origin in civil systems, but most of them (perhaps all, with the exception of Execution price / Cost) should apply to military networks as well. Even the cost aspect may be of importance if "cost" is defined in terms of resource consumption rather than monetary costs.

1.2 Quality parameters

The quality parameters of a Web service are the quantitative aspects of a service that affects the quality the Web service is capable of offering. Each of these quantitative parameters can affect one or more of the quality aspects listed above. The Web service quality parameters can be divided into categories based on how likely the parameters are to change over time, and how these parameters can be measured.

Static parameters are determined when the Web services first becomes available, and have a very low probability for changing during the life time of the Web service. One example of such a parameter is the resolution and image formats supported by a camera service.

Dynamic parameters, on the other hand, change during the Web service life time. These parameters can be further divided into two groups, depending on how the value of the parameter can be measured. Single side parameters can be measured by only one entity, and does not depend on the network resources between two entities. Examples of such parameters are the availability and current load of a Web service. The other type of dynamic parameter can only be determined after a binding between two entities has been made, since they depend on the pair of entities involved in the communication. The majority of these parameters are network resource related, such as the available bandwidth between a service provider and a service consumer, or the expected response time of a service registry.

1.3 Pervasive Quality Handling

Dependencies between components means that no quality guarantees can be given unless quality handling is pervasive. By *pervasive quality handling* we mean that every component that performs

functionality that can affect the quality offered by a Web service is quality aware, and supports differential treatment of information based on the quality needs of that information. For Web services, this means that quality handling must be offered at every step of the Web service life cycle (see Section 1.7), and also at, and between, the levels in the network stack.

Softer quality guarantees, such as statistically calculated guarantees, and prioritization between information can be supported even if quality support is not pervasive, but pervasive support for quality handling will be beneficial even if only soft guarantees are supported. Having better knowledge about current resource availability and network conditions can be used to adapt traffic flow, do load balancing and limited admission to critical components. In addition, the ability to signal the priority, urgency and network resource requirements of a given piece of information can be used to ensure consistent treatment of information throughout the infrastructure.

1.4 Standardization of quality support

QoS, and what it means for Web services, has been addressed by many different sources, but standardization is still lacking in this area. In the last five or so years there are a few standards that have been ratified that will help build quality-aware services, but none of them provide a full quality framework for Web services:

- Security: WS-Security and related standards such as XML signature and XML encryption provide integrity and confidentiality for Web services. SAML/XACML provide role based authentication, authorization, and access control. Additional standards exist for e.g., digital rights management and other security-related aspects. For a more complete discussion of Web services and security please refer to [45].
- Reliability: The OASIS specification WS-ReliableMessaging defines and supports a number of delivery assurances for a single message or a group of messages:
  - AtLeastOnce – Each message will be delivered at least once. If a message cannot be delivered, an error must be raised. Duplicates are allowed, meaning that messages may be delivered more than once.
  - AtMostOnce – Each message will be delivered at most once, meaning that a message may not be delivered at all, but if it gets through then it will never be duplicated.
  - ExactlyOnce – Each message will be delivered exactly once. If a message cannot be delivered, an error must be raised. Duplicate messages shall not occur.
  - InOrder – Messages will be delivered in the order that they are sent. This assurance can be combined with any of the above assurances.
- Transaction management: WS-Transaction is a set of specifications, built on top of the WS-Coordination framework, that defines protocols for transaction support in Web services.

But other issues (e.g., prioritization, preemption) lack standardized solutions today.

1.5 Knowledge of the networking environment

In disadvantaged grids communication resources are scarce and variable. Thus, it is important that middleware and applications are able to adapt to the available capacity. This means that knowledge of the networking environment the C2 software is operating in is of great importance.

One approach to obtaining such network information is by employing monitoring. Monitoring of networks carrying data is important to improve the quality of service. There are two main types of monitoring: Monitoring in the planning phase, i.e., for example testing that the maximum achievable throughput is in accordance with the agreement. Monitoring during deployment, i.e., when the network is actually being used, has different goals. Here, it is important to know the current load (e.g., link utilization) to be able to shape and control data traffic in a coherent manner. There exist many solutions capable of doing this, but they are mostly geared towards use on the Internet and in corporate networks. Sending data to and from mobile units, like military vehicles moving in a combat zone gives challenges that may make some current tools unsuitable.

A recent Master's degree thesis supervised by FFI focused on freely available tools, and attempted to identify which tools were suitable for the planning phase and the deployment phase in disadvantaged grids.

In short, it was found that current tools may be employed in the planning phase, but none of the tools tested were suitable for use during deployment. Thus, there is a need to investigate other solutions in the future, as monitoring when the network is in actual use is important for supporting the advanced QoS features discussed in this report. For the full experiment details, see [55].

1.6 Interoperability challenges

There are a number of interoperability issues that must be addressed in order to achieve QoS across systems, which in turn allows for the implementation of cross domain Web services:

- Services must be described in a standardized manner, using a description format that is rich enough to ensure that end systems know enough about the service to perform automated interaction with services belonging to other systems. The current Web service standard for service descriptions, the Web Services Definition Language (WSDL) [10], offers only syntactical descriptions.
- A method for distribution of the QoS information is needed. It must be possible to distribute this across networks at different operational levels, and across system- and national boundaries.
- Provided we have rich service descriptions that we are able to distribute in an efficient manner, there is still a major challenge in ensuring that the understanding of these descriptions is common among nations. There is a need for both human and machine readable semantics to ensure that this is achievable.
- Due to the complexity of a coalition network, it is highly unlikely that one will be able to

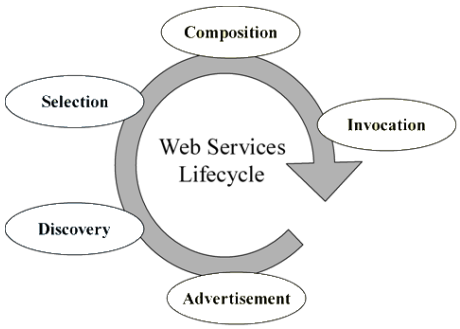


Figure 1.1 Web services lifecycle

find one mechanism for distribution of service descriptions that will work in all systems. This means that there will be a number of different service description distribution mechanisms that need to co-exist. These mechanisms must be made interoperable through the use of interoperability points, most likely in the form of gateways that are able to translate between the mechanisms.

1.7 Web services lifecycle

The Web services lifecycle is illustrated in Figure 1.1. It addresses the different stages encountered when developing, deploying, and employing Web services. Let us explore the lifecycle in further detail, while paying special attention to potential extensions necessary to support QoS:

1. The *Advertisement* is the process in which a service provider makes a service available (publishes it using a service discovery mechanism). Necessary extensions to support QoS:
   - Provide information about the different quality aspects and parameters supported by the service
   - Make this information available in addition to the standard service description
   - Quality handling requirements:
     - A common understanding/interpretation of quality
     - A standardized way of describing supported quality – and how to use the quality information
     - Service discovery mechanism must be able to handle additional metadata
2. *Discovery* is the process in which a service consumer queries the discovery mechanism, and finds suitable services. Necessary extensions to support QoS:
   - Return quality information along with the service description (simple approach), or
   - Allow the consumer to use quality parameters in their search. In this latter case the discovery mechanism must understand the quality parameters, an approach which enables more advanced matchmaking than the former.
   - Quality handling requirements (in addition to the above):
     - A way of describing the quality needs of the client

3. *Selection* is the process of selecting which services to invoke, based on the Discovery results. Necessary extensions to support QoS:
    - Enough information needed to make an informed choice
    - Need to choose which quality to ask for in addition to which service to invoke
    - Quality handling requirements (in addition to the above):
        - Quality aware matchmaking
4. *Composition* is fulfilling a requirement by combining independent services to provide new, value-added functionality. Necessary extensions to support QoS:
    - Handling combinations of quality parameters
    - Some parameters require run-time calculations
    - Quality handling requirements (in addition to the above):
        - Need a means to express the quality of the combined service
5. *Invocation* is the process in which a consumer calls a service, by sending a request message, and receiving a response. Necessary extensions to support QoS:
    - Messages must be sent using required network QoS
        - Reliability?
        - Transactional?
    - Requests must include information about required QoS, or be agreed upon up front (SLAs[1], policies).
    - Higher level quality should be signaled to lower levels
    - Application priority to network priority mapping
    - Adaption to current resource availability?
    - Quality handling requirements (in addition to the above):
        - Network knowledge (monitoring?)

1.8 Towards increased QoS support for Web services

Achieving QoS support for Web services is complex and features many different approaches. At FFI we have investigated some of the necessary building blocks that are needed for end-to-end QoS for Web services. Namely, we have investigated how *content filtering* can be used to reduce information overhead by altering the messages sent. This is orthogonal to employing compression, which does not alter the message content, but merely attempts to represent the exact same information using a more space-efficient representation. Our work on content filtering has been published externally at a peer-reviewed[2] conference. The paper is included in Appendix A.

[1] A service level agreement (SLA) is a contract between two parties, where one is a consumer and one a provider. It is a business level contract that has been negotiated, and provides a specification of promises/responsibilities. It does not detail how these promises are are implemented, however.

[2] Peer-review is used to determine a paper's suitability for publication at a certain conference, and is important to ensure that standards are maintained and to provide credibility. It should be noted that different conferences have different approaches to the peer-review process: IEEE conferences typically perform blind reviews of submitted papers, meaning

In order for a system to use resources available in other systems, finding services and determining how to use them must be done in a manner that requires as little manual configuration as possible. One issue with Web services standards is the inability to describe what the service is capable of doing. Semantic Web services (SWS) [41] extend the service descriptions of current Web services technology with rich, explicit semantics to improve service discovery, selection and invocation significantly. SWS are believed to facilitate run-time, capability-based discovery of services. This is essential for solving the problem of run-time service discovery in tactical networks while improving interoperability. We have performed preliminary experiments with SWS in conjunction with discovery of Web services and orchestration, with special focus on issues that can arise in mobile ad hoc networks. This effort has been documented through two externally published, peer-reviewed conference papers. The papers are included in Appendix B and Appendix C.

From a security viewpoint, role-based access control (RBAC) is an approach to restricting system access to authorized users. In a QoS setting the concept of *role* can be extended to include not only traditional security aspects regarding authentication and authorization, but also to signal the priority of messages traversing the network according to the role issuing said messages. We have investigated how role based admission control can be used in conjunction with QoS to ensure higher success rates for the most important Web services traffic. Cross-layer signaling of QoS data (i.e., networking information exposed from a router) was used to ensure efficient admission control and corresponding resource use. A prototype was implemented and tested as part of a student task at FFI, and the work was later published externally at a peer-reviewed conference. The paper is included in Appendix D. Continuing this work, a group of students implemented RBAC in an open source enterprise service bus (ESB)[3]. In this work the security standard SAML was employed to represent the roles, and cross-layer signaling (i.e., setting the appropriate DiffServ class in the network) was used to prioritize each message involved in the communication. This work has also been peer-reviewed and published externally at a conference, and is included here in Appendix E.

1.9 Conclusion and future work

Interoperability is crucial when attempting to realize NNEC, but achieving such interoperability is a considerable challenge given the large number and heterogeneity of the different systems currently being used, along with the limited communication resources available on the tactical level. Web services are the most common way of realizing a SOA today. Consequently, there is a strong focus both nationally and within NATO on employing Web services in military networks. However, while the core standards of Web services are mature, several aspects of QoS are still

that the reviewers are unknown to the persons submitting the paper. In recent years (thus applying to the papers included in this report) the ICCRTS has also adopted a peer-review process similar to that of the IEEE, except that the review is not blind. ACM conferences usually perform so-called double blind reviews, meaning that neither the submitters nor the reviewers know each others' identities.

[3] An ESB is a middleware product which implements an infrastructure upon which a SOA can be built. It is not SOA in itself, but provides important functionality such as message routing, and translation between message formats and different transport protocols. For interoperability reasons ESBs often leverage a subset of the different available Web services standards.

lacking standardization and non vendor-specific implementation.

In this report, we have looked into current research within the area of QoS for Web services. We have defined central terms, discussed which QoS properties are currently covered by ratified standards, and which properties are still lacking in standardization. We have included externally published research papers in appendices, which provide further details of how one can approach increasing the QoS support for Web services. It should be noted that these summarize research results, meaning that the solutions outlined are merely research prototypes – while promising in the scenarios they were tested, the solutions are still far from undergoing standardization or even being suitable as deployable proprietary products.

There are still many open issues regarding QoS for Web services, so further research is needed. One of the more pressing issues is extending service descriptions, and the mechanisms that handle these descriptions, to include support for the necessary QoS metadata. Many of the aspects discussed in this report cannot be implemented without such metadata support, as the metadata needs to be made available in the Advertisement and Discovery steps of the Web services lifecycle. This is the foundation of further research into QoS enabling Web services.

References

[1] J. Babiarz, K. Chan, and F. Baker. Configuration Guidelines for DiffServ Service Classes. RFC 4594 (Informational), Aug. 2006.

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

[3] T. Bilski. Trends in quality of service on long-distance connections. in proceedings of the Military Communications and Information Systems Conference (MCC 2009), Prague, Czech Republic, September 2009.

[4] D. Black, S. Brim, B. Carpenter, and F. L. Faucheur. Per Hop Behavior Identification Codes. RFC 3140 (Proposed Standard), June 2001.

[5] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss. An Architecture for Differentiated Service. RFC 2475 (Informational), Dec. 1998. Updated by RFC 3260.

[6] J. Carreno, G. Galdorisi, R. Goshorn, and A. Siordia. Maintaining situational awareness in large, complex organizations. 11th ICCRTS, Cambridge UK, 2006.

[7] H. Chen, T. Yu, and K.-J. Lin. Qcws: An implementation of qos-capable multimedia web services. IEEE Fifth International Symposium on Multimedia Software Engineering (ISMSE'03), 2003.

[8] L. Clement, A. Hately, C. von Riegen, and T. Rogers (eds.). UDDI Version 3.0.2. `http://uddi.org/pubs/uddi_v3.htm`, 2004.

[9] Consultation, Command and Control Board (C3B). CORE ENTERPRISE SERVICES STANDARDS RECOMMENDATIONS: THE SOA BASELINE PROFILE VERSION 1.7. Enclosure 1 to AC/322-N(2011)0205, NATO Unclassified releasable to EAPC/PFP, 11 November 2011.

[10] F. Curbera, M. Duftler, R. Khalaf, W. Nagy, N. Mukhi, and S. Weerawarana. Unraveling the Web services Web, An Introduction to SOAP, WSDL, and UDDI. In IEEE Internet Computing, vol. 6, no. 2, March/April 2002.

[11] B. Davie, A. Charny, J. Bennet, K. Benson, J. L. Boudec, W. Courtney, S. Davari, V. Firoiu, and D. Stiliadis. An Expedited Forwarding PHB (Per-Hop Behavior). RFC 3246 (Proposed Standard), Mar. 2002.

[12] Defence R&D Canada Valcartier. High capability tactical communications network-hctcn td. DRDC Valcartier Fact Sheet IS-226-A.

[13] D. Dragolov and A. Novák. Service oriented architecture - concept of services, infrastructure components, implementation practices and military environment specifics. Military Communications and Information Systems Conference (MCC 2009), Prague, Czech Republic, September 2009.

[14] S. Fuger, F. Najmi, and N. Stojanovic (eds.). ebXML registry information model version 3.0. `http://docs.oasis-open.org/regrep/v3.0/specs/regrep-rim-3.0-os.pdf`, 2005.

[15] T. Gagnes. Assessing dynamic service discovery in the network centric battlefield. IEEE Military Communications Conference (MILCOM 2007), vol., no., pp.1-7, 29-31 Oct. 2007.

[16] G. Ghinea, J. P. Thomas, and R. S. Fish. Multimedia, network protocols and users-bridging the gap. In Proceedings of the Seventh ACM international Conference on Multimedia (Part 1), Orlando, Florida, USA, pp. 473-476., October 30 - November 05, 1999.

[17] H. Gibb, A.and Fassbender, M. Schmeing, J. Michalak, and J. E. Wieselthier. Information management over disadvantaged grids. Final report of the RTO Information Systems Technology Panel, Task Group IST-030/RTG-012, RTO-TR-IST-030, 2007.

[18] C. Griwodz, M. Liepert, A. E. Saddik, G. On, M. Zink, and R. Steinmetz. Perceived consistency. In Proceedings of the ACS/IEEE international Conference on Computer Systems and Applications, Washington, DC, USA, June 25 - 29, 2001.

[19] D. Grossman. New Terminology and Clarifications for Diffserv. RFC 3260 (Informational), Apr. 2002.

[20] R. Haakseth, T. Gagnes, D. Hadzic, T. Hafsøe, F. Johnsen, K. Lund, and B. Reitan. Experiment report: "soa – cross domain and disadvantaged grids" – nato cwid 2007. FFI-Report 2007/02301.

[21] T. Hafsøe, F. Johnsen, K. Lund, and A. Eggen. Adapting web services to for limited bandwidth tactical networks. 12th ICCRTS, Newport, RI, USA, June 2007.

[22] T. Hafsøe, F. Johnsen, and M. Rustad. Automated qos-aware service selection and orchestration in disadvantaged grids. Military Communications and Information Systems Conference (MCC 2010), Wroclaw, Poland, September 2010.

[23] M. Hauge, J. Andersson, M. A. Brose, and J. Sander. Multi-topology routing for improved network resource utilization in mobile tactical networks. IEEE MILCOM, 2010.

[24] J. Heinanen, F. Baker, W. Weiss, and J. Wroclawski. Assured Forwarding PHB Group. RFC 2597 (Proposed Standard), June 1999. Updated by RFC 3260.

[25] J. D. J.E. Miller. Accessing and sharing: Facets of addressing information overload. 11th ICCRTS, Cambridge UK, 2006.

[26] F. T. Johnsen. An nffi-based web service discovery mechanism for tactical networks. Military Communications and Information Systems Conference (MCC 2009), Prague, Czech Republic, September 2009.

[27] F. T. Johnsen. Pervasive web services discovery and invocation in military networks. FFI report 2011/00257.

[28] F. T. Johnsen, T. Hafsøe, M. Hauge, and Ø. Kolbu. Cross-layer quality of service based admission control for web services. in proceedings of the 6th International Workshop on Heterogeneous, Multi-hop, Wireless and Mobile Networks (HeterWMN) 2011, Houston, TX, USA, 2011.

[29] F. T. Johnsen, T. Hafsøe, and M. Skjegstad. Web services and service discovery in military networks. International Command and Control Research and Technology Symposium (ICCRTS), June 2009.

[30] F. T. Johnsen, M. Rustad, T. Hafsøe, A. Eggen, and T. Gagnes. Semantic service discovery for interoperability in tactical military networks. The International C2 Journal, Special issue: Agility and Interoperability for 21st Century Command and Control, Volume 4, Number 1, 2010.

[31] D. Kleinman and D. Serfaty. Team performance assessment in distributed decision making. Interservice Networked Simulation for Training Conference, University of Central Florida, Orlando, FL, USA, 1989.

[32] Ø. Kolbu. QoS related admission control for web services. Master's thesis, University of Oslo, Norway, February 2011.

[33] K. Kritikos and D.Plexouakis. Owl-q for semantic qos-based web service description and discovery. First International Joint Workshop on Service Matchmaking and Resource Retrieval in the Semantic Web, Busan, South Korea, November 2007.

[34] J. Krygier, P. Lubkowski, and K. Maslanka. Simulation study of selected qos supporting components for nec compliant networks. in proceedings of the Military Communications and Information Systems Conference (MCC 2009), Prague, Czech Republic, September 2009.

[35] E. Kubera, J. Milewski, and M. Rozycki. Qos support in the special wireless networks. in proceedings of the Military Communications and Information Systems Conference (MCC 2009), Prague, Czech Republic, September 2009.

[36] R. Lausund and S. Martini. Norwegian modular network soldier (normans). FFI Facts, November 2006.

[37] K. Lund, A. Eggen, H. D., T. Hafsøe, and F. T. Johnsen. Using web services to realize service oriented architecture in military communication networks. IEEE Communications Magazine, Vol. 45, No. 10, October 2007.

[38] K. Lund, A. Eggen, D. Hadzic, T. Hafsøe, and F. T. Johnsen. Using web services to realize service-oriented architecture in military communication networks. *IEEE Communications Magazine, Special issue on Network-Centric Military Communications*, October 2007.

[39] A. Mani and A. Nagarajan. Understanding quality of service for Web services. `http://www.ibm.com/developerworks/webservices/library/ws-quality/index.html`.

[40] Matchmaker homepage. Owl-s/uddi matchmaker details. `http://www.daml.ri.cmu.edu/matchmaker`.

[41] S. A. McIlraith, T. C. Son, and H. Zeng. Semantic web services. In IEEE Intelligent Systems, vol. 16, no. 2, March/April 2001.

[42] V. Modi and D. Kemp (eds.). Web services dynamic discovery (ws-discovery) version 1.1. `http://docs.oasis-open.org/ws-dd/discovery/1.1/wsdd-discovery-1.1-spec.pdf`, OASIS standard, 2009.

[43] NATO Standardization Agency. IP QoS. draft 3 (work in progress), NATO/EAPC Unclassified, 2011.

[44] K. Nichols, S. Blake, F. Baker, and D. Black. Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers. RFC 2474 (Proposed Standard), Dec. 1998. Updated by RFCs 3168, 3260.

[45] N. A. Nordbotten. *XML and Web Services Security*. FFI report 2008/00413, 2008.

[46] OASIS. Security assertion markup language (saml) v2.0 technical overview, committee draft 02, 25 march 2008. `http://docs.oasis-open.org/security/saml/Post2.0/sstc-saml-tech-overview-2.0-cd-02.pdf`.

[47] OASIS. Ws-qdlv1.0. OASIS Web service Quality Model TC, `http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsqm`.

[48] G. On, J. Schmitt, and R. Steinmetz. On availability qos for replicated multimedia service and content. In LNCS 2515 (IDMS-PROMS'02), Portugal, pp. 313-326, November 2002.

[49] OpenSAML website. Home page. `https://wiki.shibboleth.net/confluence/display/OpenSAML/Home/`.

[50] Oracle. Glassfish home page. http://glassfish.java.net/.

[51] OWL-S API. Maryland information and network dynamics lab semantic web agents project. `http://www.mindswap.org/2004/owl-s/api/`.

[52] P. Psenak, S. Mirtorabi, A. Roy, L. Nguyen, and P. Pillay-Esnault. Multi-Topology (MT) Routing in OSPF. RFC 4915 (Proposed Standard), June 2007.

[53] W. Rahman and F. Meziane. Challenges to Describe QoS Requirements for Web Services Quality Prediction to Support Web Services Interoperability in Electronic Commerce. Communications of the IBIMA, Volume 4, 2008.

[54] S. Sakr. Xml compression techniques: A survey and comparison. *Journal of Computer and System Sciences*, 75(5):303 – 322, 2009.

[55] G. Salberg. Monitoring in disadvantaged grids. Master's degree thesis, Department of technology, Narvik University College, June 2012.

[56] M. Skjegstad. Mobiemu home page. `https://code.google.com/p/mobiemu/`.

[57] J. Sliwa and M. Amanowicz. A mediation service for web services provision in tactical disadvantaged environment. IEEE MILCOM, 2008.

[58] J. Sliwa and D. Duda. Adaptive web services supported by qos ip network. Military Communications and Information Systems Conference (MCC 2009), Prague, Czech Republic, September 2009.

[59] M. Tian, A. Gramm, H. Ritter, and J. Schiller. Efficient selection and monitoring of qos-aware web services with the ws-qos framework. IEEE International Conference on Web Intelligence, Beijing, China, September 2004.

[60] M. Tian, A. Gramm, H. Ritter, and J. Schiller. Efficient selection and monitoring of qos-aware web services with the ws-qos framework. IEEE/WIC/ACM International Conference on Web Intelligence (WI'04), pp. 152-158, 2004.

[61] V. Tran and H. Tsuji. A Survey and Analysis on Semantics in QoS for Web Services. International Conference on Advanced Information Networking and Applications (AINA '09), 2009.

[62] A. Vogel, B. Kerherve, G. von Bockmann, and J. Gecsei. Distributed multimedia and qos: A survey. IEEE Multimedia, Vol. 2, No. 2, pp. 10-19, 1995.

[63] W3C. OWL-S: Semantic Markup for Web Services.

[64] W3C. Semantic annotations for wsdl working group. `http://www.w3.org/2002/ws/sawsdl`.

[65] W3C. SWRL: A Semantic Web Rule Language Combining OWL and RuleML. `http://www.w3.org/Submission/SWRL`.

[66] W3C. Web Service Modeling Ontlogy (WSMO). `http://www.w3.org/Submission/WSMO`.

[67] X. Wang, T. Vitvar, M. Kerrigan, and I. Toma. A QoS-Aware Selection Model for Semantic Web Services. Service-Oriented Computing - ICSOC 2006, pages 390-401, 2006.

[68] WSO2. Wso2 home page. http://wso2.com/products/enterprise-service-bus.

[69] T. Yu, Y. Zhang, and K.-J. Lin. Efficient algorithms for web services selection with end-to-end qos constraints. ACM Transactions on the Web (TWEB), Volume 1 Issue 1, Article No. 6, May 2007.

Appendix A Reducing Network Load through Intelligent Content Filtering

This paper was written by Trude Hafsøe and Frank T. Johnsen. It was published at the 13th ICCRTS, Seattle, WA, USA, June 2008.

Abstract

Future international military operations will be more complex than traditional operations undertaken by just one nation; military units from different nations will have to cooperate with not only with each other but also with local governments and civil organizations in order to reach common goals and to ensure a shared understanding of each other's task and domain responsibility. One characteristic of such endeavors is that each organization brings with it its own information and communication systems. Interconnecting these communication systems will lead to an increase in the total amount of information available to users of these systems. One of the main challenges when building an information infrastructure to support such operations is to ensure information superiority; all users must get access to the information they need to perform efficiently, while at the same time avoiding that the user is flooded with irrelevant information. Making sure that only relevant information is transmitted is even more important in tactical systems, where communication resources are very limited. This paper describes the use of several different types of content filtering as a measure for reducing the network load, and presents the results of our experiments with content filtering in disadvantaged grids performed at NATO CWID 2007.

A.1 Introduction

International military operations, such as those performed by the NATO Response Forces, require that a number of participants from different nations and organizations work together towards a shared purpose. Mission effectiveness depends on the participants' ability to communicate both effectively and efficiently with all cooperating partners, thus a common information infrastructure is essential.

A common information infrastructure for NEC operations needs to be able to ensure that all users are supplied with information that is both sufficient and relevant enough for them to be able to make appropriate decisions at all times. Such an increase in the amount of information that is available to users can cause problems both at the cognitive level and at the network level. This paper discusses how content filtering can be used to reduce the impact increased volumes of information can have on the network. In particular, tactical links have low bandwidth available, and only relevant information should be sent over the network to limit the possibility of congestion due to irrelevant data. Maintaining information superiority, which is the capability to collect, process, and disseminate an uninterrupted flow of information while exploiting or denying an adversary's ability to do the same [25], means that proper information management is critical. Ensuring that only relevant information is transmitted over the network helps maintain information superiority,

in that irrelevant information is not allowed to disrupt the information flow by overflowing the network

Content filtering can be used to alleviate network congestion by removing information that is not relevant to the user. Several different types of content filtering exist, depending on the type of the data and how the data is used. This paper presents results of experiments performed at NATO CWID 2007. In these experiments we tested how content filtering can be used to ensure that only relevant information was supplied to tactical users, and thereby avoiding overloading the network and saving bandwidth resources at the same time.

The remainder of this paper is organized as follows: First we present the tactical system and the data format used in the experiments, followed by a description of different types of content filtering. Then we present the experiment setup and results.

A.2 Background and Motivation

Using content filtering is one of several measures that can be employed in order to increase the information infrastructures ability to adapt to changing network and battlefield conditions. [17] states that this kind of adaptability is one of the requirements next-generation military information systems need to fulfill. In order to test the usability of content filtering, we performed several technical trials at NATO CWID 2007, as part of a larger Service Oriented Architecture (SOA) experiment. We SOA-enabled an experimental tactical system called NORMANS Advanced, as this allowed us to test not only content filtering alone, but also confirm that the content filtering techniques we tested works in conjunction with Web services.

A.2.1 NORMANS Advanced

The NORMANS [36] concept includes a C4I system that is designed to take the roles of the different type of users into account. Dismounted soldiers will in the future act as sensors, effectors and decision makers, and their C4I equipment, both hardware and software, must reflect their main tasks. The NORMANS C4I concept has a modular approach based on a main navigation and communication module, named NORMANS-light, for all private soldiers in a section. A more advanced commander system (NORMANS advanced) uses digital maps, friendly force tracking and the ability to mark red force observations to help improve situational awareness for more advanced users. The NORMANS C4I concept is based on voice and data communication within the sections using a simplified data transmission protocol. In addition, voice as well as data can be sent between sections using IP and a tactical messaging system. At NATO CWID we used a slightly modified version of the NORMANS advanced software, as the use of a proprietary protocol complicates interoperability. Using standard based solutions makes the task of interconnecting systems easier, so we modified the software to communicate by inputting and outputting XML formatted data.

A.3 Filtering

To maintain information superiority in a coalition force it is paramount that all necessary and relevant information is disseminated throughout the network. Thus, it becomes important to identify the information that is indeed relevant, and transmit only that over the network. Which information that is relevant will vary from user to user depending on their role and what the information will be used for. In order to perform correct filtering, the system performing the filtering must be aware of the needs of its users, and filter accordingly. In our trials the filtering was done based on a profile that specified which information was most important to the user. How can we identify information as relevant? There are many factors to take into consideration. For example, some information may only be relevant within a certain area of operations, and thus it should be disseminated only in that geographical location or to users outside the area that specifically request that information. Some information may change frequently, for example position information, whereas other information can change less frequently or even be entirely static. In such cases messages containing status updates have different requirements as to how often they need to be transmitted. If network resources are scarce, then knowledge of the importance of the information can help the system prioritize by sending the most important information first, and delaying or perhaps even entirely discarding the less important information. Which information each unit needs, is first and foremost a question of which role the unit has. In some cases filtering an entire message or part of a message will save network resources while still ensuring that the recipient gets all the relevant information that it requires. Security issues, trust and clearance are also important aspects, and filtering should also be used to stop classified information from exiting a system. This latter aspect is currently subject to research and has a lot of open issues still which are beyond the scope of this paper. See [20] for an overview of some of the security related filtering experiments at NATO CWID 2007.

In summary, we have several aspects to consider when disseminating information. The most important aspect is that only necessary and relevant information should be received by the units. There are many factors that can be used for filtering; a non-exhaustive list is presented below:

- Geographical filtering
- Frequency based filtering
- Priority based filtering
- Role based filtering
- Security label filtering

Furthermore, the filtering can be of two types, in that one can filter

- Entire messages, or
- Parts of messages.

In our experiments at NATO CWID 2007 we used a combination of geographical and frequency

based filtering. The information we considered in the experiments was only tracking information, and thus we used filtering of parts of messages to ensure that only relevant tracks were delivered to the unit in the (simulated) field. In the following section we discuss the experiments and filter functionality in detail.

A.3.1 Implementation challenges

Having discussed some of the different issues of filtering above, we now turn our attention to the challenges that arise when one considers implementing a filtering scheme. I.e., we need to decide how and where the filtering should be done. The "how" of filtering is basically a matter of choosing which technique(s) to implement, for example a combination of geographical and frequency based filtering as we used for tracking information in our experiments. Which kind of filtering is best to use will depend on the kind of information the message contains. The challenge here is to identify the recipient's needs when designing the system, and performing filtering accordingly. How to describe these needs should be a matter of discussion and standardization within NATO, and a further discussion of these challenges are beyond the scope of this paper. After having decided which policy to employ, there is the issue of where the functionality should be implemented. The "where" of filtering is a matter of placing the filtering functionality in the NEC information infrastructure. The easiest way to filter information is in the receiving unit. That unit may know which data is relevant to present to its user, and can discard other information. This requires no state information in the network or in the information producers, thus leading to low system complexity. However, for each piece of unimportant information that is discarded in the end-system, a corresponding amount of bandwidth has been wasted in transmitting this information all the way from the producer to the receiver. Ideally, only information that is relevant according to the chosen policy should be injected into the network. If one can perform the filtering where the information is produced, then this is optimal in two ways; firstly, no bandwidth is wasted, and secondly, only relevant information is received by the end-system terminals. However, implementing the filtering policy in every potential information producer may be infeasible. Filtering requires some processing for the system to find out whether the information should be transmitted or not by inspecting the information and comparing it to the policy. As such, this will put higher requirements on the computational capabilities of these systems. In any case, proxies should be employed between networks to ensure better use of resources [21]. The proxies can for example function as security guards [20], something that will be needed on the way towards full-fledged NEC to secure the information flow. If we need proxies anyway, then perhaps one should just implement the filtering functionality there and reduce the complexity of the end-systems? System implementation complexity is reduced by centralizing the filtering functionality in proxies, but such a solution leads to an increase in bandwidth consumption between the producer and the proxy since the data transmitted between these are unfiltered. It should be noted that the proxies will need policy information for each kind of unit that is to receive information. Furthermore, the proxy must be able to recognize and process different kinds of message formats that can pass through it. This means that the proxies will become potential bottlenecks in the network due to the

computational complexity of message processing.

In short, we have discussed the three places to perform filtering:

- Filtering in the end-system terminals
  - Low complexity
  - Stateless
  - High bandwidth use
- Filtering in proxies
  - High complexity – must know all combinations of end-system terminal and message formats and the corresponding filter policy
  - Proxy may need to keep state (for example position information for each receiving unit in the case of geographical filtering)
  - Reduces bandwidth use across networks
- Filtering in the message producing system
  - Medium complexity – must know all end system-terminals and corresponding filter policy
  - May need to keep state
  - Best bandwidth utilization since no unnecessary information is injected into the network

Basically, filtering in the end system should be avoided since it wastes network resources, and especially on the tactical level bandwidth is scarce. Seemingly, filtering in the message producing system is the best option. However, proxies also have an important benefit over that of filtering in the producer: If the producer sends information to recipients with different capabilities, then it must filter once for each type of recipient. A proxy, on the other hand, will be closer to the recipient, and as such potentially have fewer types of recipient to filter for.

In our experiments we used the proxy filtering approach. The local HQ track store was on a high capacity LAN together with the proxy server which was connected to the (emulated) disadvantaged grid. We focused exclusively on disseminating tracking information, and as such we had a relatively simple proxy solution: Our proxy kept state about each recipient (last reported position). It received all the information from the track store at regular intervals, and would then perform geographical filtering of the tracks based on the state it kept before sending the regional tracks over the tactical network to the NORMANS unit. The proxy also performed frequency based filtering as one of the filter types used did not transmit all data at the same interval, but rather updated the most relevant data more frequent than other data.

A.4 Experiments and Evaluation

Defence R&D Canada have performed a series of technical trials related to the dissemination of operationally important information in congested tactical radio subnets using their Low Bandwidth Test Bed [12]. Among the experiments performed is a test of dynamic reduction of network load by using content filtering techniques, as described in [17]. The experiment involved using an information management rule to determine whether or not to suppress replication messages. These messages contained a unit's report of its own position, and the rule used was based on how far the unit had moved since it previously reported its own position. Each node would use this rule to make an autonomous decision to either broadcast or suppress its own position at given time intervals. This means that the type of content filtering done in this experiment was a type of frequency based filtering, but the information management rules used were geographically based. The decision whether to perform filtering or not was made locally, which means that the required state could be maintained by each unit independently.

In our experiments units reported their own position, and the position data was gathered by a central unit, and distributed to all units. Because the unit reporting its own position was in constant movement, we did not perform filtering of the unit's own position reports. We concentrated on filtering data that was being sent out to the units, as these messages could contain position data for all other units in the battlefield, and were thus significantly larger in size than the position reports transmitted by each unit individually. We performed this filtering as close to the source as possible in order to save bandwidth on the simulated tactical links in addition to avoiding flooding units with information they did not want according to their profile. Allowing an intermediate node to perform filtering made our experiments more complex, as the intermediate node had to have updated information about the location of each unit in order to correctly perform geographical filtering. The intermediary did this by intercepting the reports sent by each unit containing their location, and maintaining an overview of the last known location of each unit.

In our experiments at NATO CWID 2007 we looked into the use of content filtering for a blue force tracking application, using NFFI-formatted data. We performed two different kinds of filtering, namely geographically based filtering of complete tracks, and filtering of optional information within tracks. A simple form of geographical track filtering is using a fixed zone filter to remove all tracks that are outside the unit's area of operation. Geographical filtering can also be used in combination with a second content filter that reduces the frequency of track reports.

A.4.1 Geographical filtering - Fixed zone filter

The fixed zone filter consists of a simple filter mechanism on the server side which is performed on each track. "Relevance" in this filter is defined as tracks within a certain distance of the soldier. For each track the distance to the last known position of the unit is calculated. If the distance is greater than a certain number (fixed, but configurable in the filter) then the track information is not sent. All tracks closer to the unit than this are reported. Such filtering is important to ensure that no unnecessary information is sent. Figure A.1 shows unit placement on the battlefield

for a tactical user (left) and a local HQ user (right) respectively. The local HQ has a complete overview of the situation, while a fixed zone filter is used to limit the information sent to the tactical user. This means that the tactical user only gets notified of other units that are within its area of operation.

A.4.2 Geographical filtering - Zone ring filter

The zone ring filter is similar to the fixed zone filter in that it uses distance as its filter metric. The idea is to save bandwidth while at the same time providing the unit with an overview of a larger section of the battlefield. This technique can be used when bandwidth limitations makes it impossible to transmit information about all relevant units as frequently as needed. The zone ring filter is optimized to allow for more frequent updates of tracks that are closer to the client than those that are further away. While the fixed zone filter uses one ring as its zone (a track is either inside (report it) or outside (don't report it) the ring), the zone ring filter uses three rings. These rings are arranged in such a way that the inner ring is updated most frequently, followed by the second ring, and finally the third ring. Information about tracks outside the third ring will not be sent. This is comparable to the fixed zone filter, which actually only has the functionality of ring 3.

In Figure A.2 we give an example of the zone ring filter: The rectangle is the display of the NORMANS system. The display is always centered at the soldier's position. This position is reported back to the central service every 40 seconds. Based on this position, the service will send its filtered track information back at certain intervals (configurable), with different multiples of the interval for the different rings (also configurable). The way we used the filter at NATO CWID was to configure rings 1 and 2 so they fit inside the map view of the unit, and configure ring 3 so that it was just outside what was possible to visualize. That way, the unit would receive frequent updates for units positioned inside ring 1 and less frequently for units that are inside ring 2, but outside ring 1. Updates about units that fall between rings 2 or 3 were rarely sent, and no information outside ring 3 was ever sent. Note that the rings are adjacent and do not overlap: When track information inside ring 2 is updated the information inside ring 1 is not sent (unless ring 1 and 2 have the same update frequency configured). The same applies to information from within ring 3, which is also independent of ring 1 and 2.

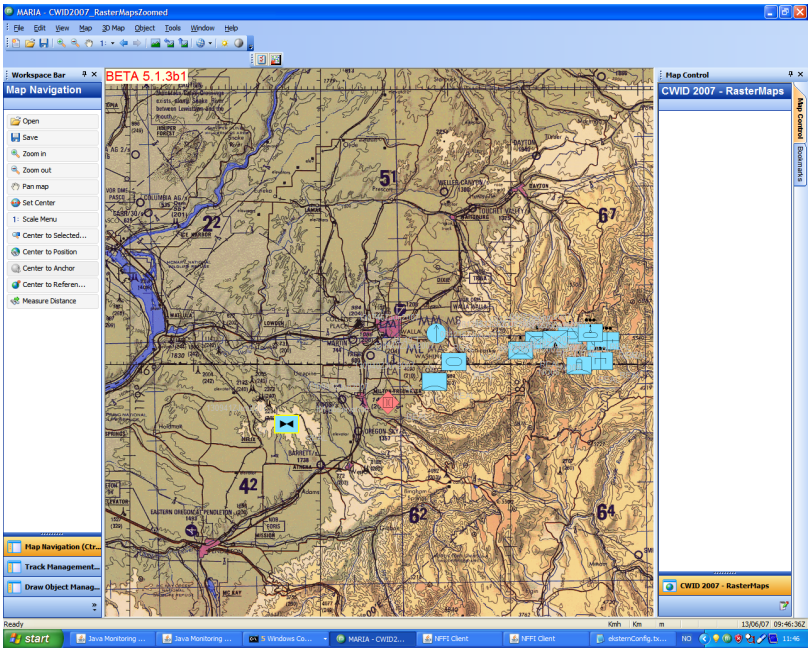
Figure A.3 shows an example of the map display for a tactical user on the left and the local HQ on the right. At first glance the two images seem to report the exact same information, but the local HQ has more frequently updated information for the units that are far away from the tactical user.

A.4.3 Filtering optional fields

We have now discussed two filters for tracks. We can also filter information within the tracks themselves, optional fields which may be of lesser importance to the soldier. This form of content filter is discussed below.

At CWID we transmitted all track information as NFFI, which has some mandatory and a lot of

(a) Tactical unit display



(b) Local HQ screenshot

Figure A.1 Fixed zone filter

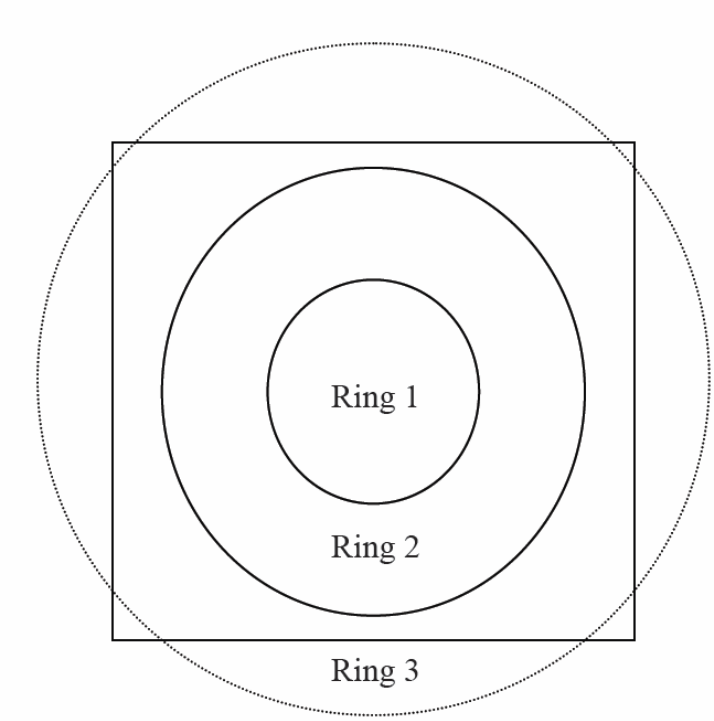


Figure A.2 Zone ring filter layout

optional fields. The NORMANS Advanced tactical system can only utilize a very small subset of the information that can be represented in an NFFI message. This means that if we transmit the full message, a large part of the information will be discarded by the recipient as not relevant. In order to save bandwidth we employed optional field filtering by removing all the irrelevant information at the server side before transmitting the data.

A.5 Results using filtering of optional fields combined with the zone ring filter

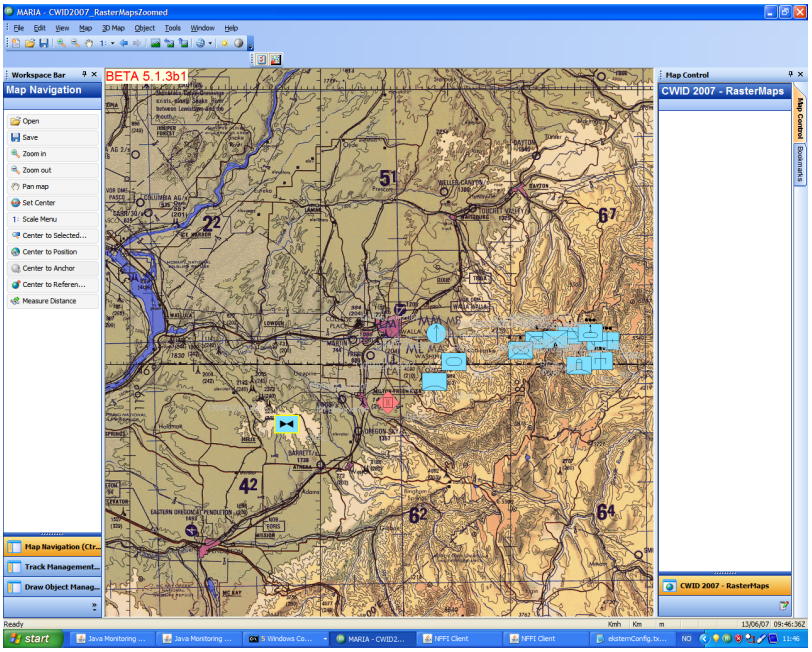
In our experiments we used the NIST Net[4] network emulator package for emulating the tactical link. Using this software, the link was limited to a bandwidth of 2.4 Kbit/s, which is representative of the bandwidth one can expect when using a radio network designed for speech traffic. The time it takes to transmit a package over this link will vary depending on other traffic that is using the same link, and the numbers given below are typical of the ones we observed during the experiment.

When transmitting track updates over a tactical network like the one used in the experiments, even a small number of tracks per message will quickly fill the link. The link usage can be reduced by either sending messages less frequent, or by reducing the size of the messages.

Our experiment with the fixed zone filter was aimed at determining what the maximum update frequency is when using a speech channel. Applying the fixed zone filter reduced the number of

[4]The NIST Net software is freely available: `http://www-x.antd.nist.gov/nistnet/`

(a) Tactical unit display



(b) Local HQ screenshot

Figure A.3 Zone ring filter example

Tracks in NFFI message	NFFI message size	Wire message size, i.e. with compression	Time to traverse a 2.4 Kbit/s link
13	10789 bytes	2246 bytes	10.0 s
7	5904 bytes	1707 bytes	8.0 s
5	4322 bytes	1509 bytes	7.2 s

Table A.1 Sending filtered NFFI messages over an emulated tactical network

tracks in the messages significantly enough to allow for an update frequency of 30 seconds. This update frequency used up most of the available bandwidth, but a somewhat lower update frequency that leaves more link capacity free will in many cases be sufficient. Further reduction of the bandwidth usage can only be achieved by applying a stricter filtering method or by reducing the update frequency. However, reducing the update frequency means that the risk of tracking information becoming outdated increases. We investigated how to better utilize the limited bandwidth available by applying the zone ring filter described above without having to compromise too heavily on accuracy.

Table A.1 shows some examples of the measured transmission time of NFFI track updates of varying size. When using the zone ring filter, the transmission of the NFFI tracks was split up so that the most frequently updated tracks, which in our case were 5 tracks, took about 7 seconds to transmit. This means that these tracks could be updated every 15 seconds, while at the same time leaving enough free bandwidth to allow tracks in the other two zones to be updated at least once per minute.

The exact update frequencies that can be used in an operational scenario will of course depend on the number of tracks that fall within the various zones of the filter, and the total bandwidth available. At NATO CWID we were operating in a controlled, experimental environment. The numbers given here should thus be considered as an example intended to illustrate the effects of the different types of filters.

A.6 The information overflow problem

As mentioned in the introduction, international military operations cause an increase of available information. This can cause problems not only on the network level, but also when it comes to the user's ability to process the information she receives, known as information overflow. As noted in [6], the predominant problems associated with overload of information is that there is more information available than can be absorbed and understood within a time span of any single individual. This can cause the recipient to overlook critical information.

Having some information overload is not necessarily bad: A skilled user, with the proper training, can learn to overcome information overload and in a team, such information can be shared. This corresponds to the thoughts in this older study [31], where it is pointed out that team members can perform better in high workload situations when there is a partial overlap in roles between them. So, with trained personnel, information overload may not be a major problem for a team to perform its tasks efficiently. However, for less trained personnel, a pre-processing of data prior to

dissemination can help guide them towards making the right decisions. It is crucial to understand the difference between recognizing and ignoring significant information that can result in either making an informed, strong decision or an ill-informed one [6]. The content filtering techniques described in this paper may also be used to alleviate the information overflow problem, but a full evaluation of the effects content filtering has on information overflow is beyond the scope of this paper.

A.7 Conclusion

Using content filtering is one of several measures that can be employed in order to increase the information infrastructure's ability to adapt to changing network and battlefield conditions.

To maintain information superiority in a coalition force it is paramount that all necessary and relevant information is disseminated throughout the network. Thus, it becomes important to identify the information that is indeed relevant, and transmit only that over the network. As a means to achieve this, we have discussed several different types of filtering: Geographical filtering, Frequency based filtering, Priority based filtering, and more. Furthermore, the filtering can be of two types, in that one can filter entire messages, or parts of messages.

Filtering in the end system should be avoided since it wastes network resources, especially on the tactical level where bandwidth is scarce. Filtering in the message producing system may be the best option. On the other hand, proxies have an important benefit over that of filtering in the producer: If the producer sends information to a many recipients, all with difference capabilities, then it must filter once for each type of recipient. Since a proxy will typically be placed at the edge of a network, it is much more likely that the recipients in this network will be of similar types. For instance, all users in a tactical network are likely to have similar limitations, such as low available bandwidth and limited power supply. The means that a proxy often has fewer different recipient types to filter for, which in turn mans that the proxy filtering implementation can be simpler.

As a proof-of-concept we have presented our experiments from NATO CWID 2007, where we successfully tested our implementation of a combined geographical and frequency based filter for track information in a proxy. The filter was based on a set of rings with different frequency assigned to each ring. We called this a *zone ring filter*.

Appendix B Semantically Enabled QoS Aware Service Discovery and Orchestration for MANETs

This paper was authored by Trude Hafsøe, Frank T. Johnsen, and Marianne Rustad. It was published at the 15th ICCRTS in Los Angeles, California, USA in 2010.

Abstract

In this paper we present our semantically enabled Quality of Service (QoS)-aware service discovery solution for dynamic environments such as Mobile Ad Hoc Networks (MANETs). Through the use of a decentralized service discovery mechanism coupled with liveness properties of services we get the needed resilience to changes that are required in a MANET.

Our novel solution includes an extended service description which, in addition to the standard Web service parameters, also allows for the description of both static and dynamic QoS parameters. The service descriptions have a supplementary description where a semantic layer adds meaning to the elements in the extended service descriptions. This is done using a Semantic Markup for Web Services (OWL-S) based service ontology, thus augmenting the solution with semantic capabilities.

Our experimental service discovery mechanism supports the distribution of extended service description parameters. These parameters are linked to an OWL-S based profile that can be used to perform more accurate service selection at the client side.

B.1 Introduction

The Service Oriented Architecture (SOA) concept, with loosely coupled entities and clearly defined interfaces is highly suitable for dynamic networks. In situations where heterogeneous nodes come together to perform a joint operation, such as search and rescue operations or internationally coordinated military operations, having the ability to quickly set up a functional information sharing capability is essential. SOA, implemented using Web services has, despite being designed for use in fixed infrastructure networks, shown promise when being extended to dynamic, low capacity networks such as military tactical networks.

There are, however, a number of challenges related to the use of Web services in disadvantaged grids, one of which is Web services discovery. In an operation the participating nodes will be mobile, and they will be likely to experience both network partitioning and/or loss of connectivity. A squad will need to access locally available services, even when connections to other networks or other partitions of the same network fail. In such a scenario, relying on a centrally located registry for Web services can lead to the loss of the discovery capability if the reach back link goes down. Nodes in the network will thus be prevented from finding local services even if these services are still available. In order to maintain the availability of the discovery capability, and thereby ensure that all currently available services can be located, we need a discovery mechanism that is suitable

for mobile ad-hoc networks (MANETs).

Operations vary greatly in complexity, making it difficult to accurately predict which units, how many units and what the capabilities of the units involved in the situation are ahead of time. The units involved are likely to be heterogeneous and their capabilities, both when it comes to connectivity and end system resources, may differ from one client to the next. These differences in capabilities mean that not all clients will be able to use the available services in the same way, even if they need to access the same information. One example of this could be that a client using a handheld device such as a rugged PDA would require images provided by a camera service to be transmitted at a different resolution or using a different file format than a client using a laptop computer. In order to enable each client to find the service that can best satisfy their requirements, the service discovery and selection mechanism must, in addition to the standard properties of a service, also take the Quality-of-Service (QoS) attributes of the services into account.

The benefit of including QoS properties is that it allows for more advanced service discovery and selection, but it introduces a requirement for an extended service description, and a means of using this new information to select which services to use.

Semantic Web services aim at enabling a more automatic and dynamic service environment. Services are described by their capabilities and properties rather than by syntactic WSDL descriptions of endpoints and data types. In QoS sensitive settings, such as tactical MANETs, non-functional properties need to be considered during service selection. Based on Semantic Markup for Web Services (OWL-S) we use a subset we call OWL-S Light with QoS, or OWL-S LiQ for short, for description, distribution and selection of services. The service ontology subset defines the minimum of conceptual elements needed to describe services and non-functional QoS concepts important in a QoS-aware environment. Individual services are described according to the ontology and the descriptions are distributed to clients and reasoned upon during service selection. Service selection has two reasoning parts, first based on service capabilities, then based on QoS parameter values. The semantically enabled QoS aware service discovery solution we suggest in this paper addresses all phases of service discovery and selection: We suggest a tailored service description, a mechanism for distributing service information, and discuss how QoS attributes can be integrated into the service selection phase.

B.2 Related work

The term QoS has been used to describe many different aspects of computer systems, with one of the more common usages being as a reference to network resource reservation schemes rather than the quality aspects of an application or a computer system. However, with the advent of multimedia applications, the QoS concept has been extended to cover not only network characteristics, but also those aspects of both middleware and applications that affect the end user's satisfaction with the service or application.

An established and more descriptive definition of QoS is the one given by Vogel et al. [62]:

"Quality-of-Service represents the set of those quantitative and qualitative characteristics of a distributed multimedia system necessary to achieve the required functionality of an application."

This definition, along with accompanying division of QoS parameters into five categories, was designed to describe multimedia systems, but many aspects of it also apply to other application types. The five categories of QoS defined in that paper are performance-, format-, synchronization-, cost-, and user-oriented parameters, all of which are equally significant in a Web service setting as they are for multimedia applications.

It is important to note that these parameter categories affect each other; for an end user, the user oriented parameters might seem most significant, but these parameters are in turn dependent on the performance-, format-, and synchronization-oriented parameters. For instance, to perceive the subjective image quality of a video as being satisfactory, the video must be coded in a format and resolution that allows a sufficiently high image quality. Next, the system must be able to provide a sufficiently high bandwidth to support the bandwidth requirement of the video, and the video and audio tracks need to be played back in a synchronized fashion.

There are currently no finished standards for describing and handling the quality aspects of Web services, but work is ongoing within OASIS. Their Web Service Quality Definition Language (WS-QDL) [47] is an XML based language for describing QoS attributes of services, but this effort is geared towards Internet and business use of Web services, and thus has a focus on more high level QoS parameters such as cost and billing. QoS support in MANETs requires a more flexible and light-weight solution for QoS handling.

Existing frameworks that support QoS for Web services are largely limited to experimental solutions. One such framework, WS-QoS [59] supports not only the description of QoS parameters for Web services, but also how to use these QoS descriptions in conjunction with QoS mechanisms in underlying networks. This work addresses not only the service discovery process, but also the invocation of services and monitoring of Web service quality aspects. Its service description and discovery are however closely linked to service registries, which makes it less suitable for use in limited resource networks such as MANETs.

OWL-S [63] is a Semantic Web services initiative that defines a service ontology for describing Web services. The ontology incorporates the possibility to add additional service parameters, but has no specific QoS concepts defined.

WSMO [66] is another initiative that defines a conceptual framework for Semantic Web services through a service ontology. As part of their service descriptions they represent network level QoS and other interesting non-functional aspects of services. WSMO can be quite complex, which is something we want to avoid in our solution. SAWSDL [64] (Semantic Annotations for WSDL) facilitates service descriptions with semantics, but does not have a service ontology specified.

OWL-Q [33] is as an alternative QoS focused ontology language. OWL-Q is a syntactical separated extension to OWL-S defined as a set of upper ontologies. The solution combines reasoning

with the Constraint Satisfaction Problem and SWRL [65] for QoS service description and discovery. The service description defined in that paper allows for the description of QoS parameters of services, but the work focuses mainly on the description and selection phases. Thus, it relies on existing mechanisms (i.e. registries) for making the service descriptions available to clients. Handling dynamic QoS parameters requires a distribution mechanism that is more flexible, without generating a need for sending full semantic service descriptions between nodes whenever an update of the service parameters occur.

B.3 QoS in mobile Web services

As previously mentioned, QoS parameters can be categorized according to which aspects of the described application or service a parameter affects. For the use of Web services in a mobile setting, we divide QoS parameters into categories according to how they need to be handled by the service discovery mechanism.

Static QoS parameters are parameters that are known at the time when a service is deployed, and do not change during the lifespan of the service. These parameters are entered into the service description before the service is deployed, and because they remain unchanged, it is sufficient to distribute these parameters once to each client. Examples of such parameters are the maximum available resolution of a camera connected to an imaging service, or the data formats the images can be provided as. The dynamic QoS attributes of a service instance need to be included in the same service description as the static parameters, but due to them changing over time they cannot be determined before the service is deployed, and will need to be updated whenever a change occurs. Because of the need for frequent updates, the current values of these parameters need to be included in the service announcements. Most dynamic QoS parameters, such as information about the availability of a service (for instance uptime and current server load) are known by the service, or the server hosting the service, and can thus be included in the service descriptions published by the server. Some QoS parameters, particularly those related to the availability of network resources, depend on the network connection between a given client and service, and must be calculated by the client side. These parameters, which include round trip times and available bandwidth, can be included in the same service description as the other parameters. However, the inclusion of such parameters will require the client node to be able to calculate the current values of the parameters and insert them into the service description themselves. Because of the complexity of handling such parameters, we have not included these in our current implementation.

Adding QoS awareness to Web services is complex, and requires QoS support to be integrated in everything from Service Level Agreements and service descriptions to the ability for a client to specify QoS needs during service invocation. Additionally, we need mappings from higher level QoS parameters down to simple networking parameters such a traffic priority and classification.

An important first step in implementing QoS awareness in Web services is to QoS enable the service discovery process, in which the clients gain knowledge about existing services and the QoS

these services support. In order to successfully implement such a QoS aware service discovery mechanism there are a number of issues that must be addressed:

First, an extended service description that includes information about the QoS attributes of the services must be generated. This extended service description must be designed in such a way that new QoS attributes can be added to the descriptions with little effort. The reason for this requirement is that which QoS attributes must be supported will vary from service to service, and may also change over time. It is important that this service description can handle both static and dynamic QoS parameters.

Second, a mechanism for distributing service information about available services to clients is needed. This mechanism must provide the client with all the information the client needs to make a well founded decision on which services that best fits the client's requirement. Last, the client system needs to be able to make the decision about which services can fulfill the client's needs based on the client's capabilities. This service selection can take on many forms, from simple parameter matching in low end clients, to advanced service orchestration by more capable clients. In order to enable these more advanced selection scenarios, the client needs to maintain a QoS profile that specifies both its preferences and abilities.

OWL-S is a Semantic Web services initiative that defines a service ontology through Web Ontology Language (OWL) constructs. OWL gives us the logical foundation we need to reason about services. OWL-S is an extensive service ontology, but in a MANET setting we try to keep the descriptions at a minimum as we have to consider possible limitations on client equipment and network capacity. Possible equipment and network limitations motivated us to use a subset of OWL-S with QoS parameters and in this way make the ontology as small as possible. OWL-S has the ability to express non-functional service parameters, and as QoS is such an important part in a MANET setting we introduce this as a full-fledged part of the service ontology while still keeping the descriptions compliant with other OWL-S descriptions. From OWL-S we have a service description with profile, process and grounding. The Profiles are for advertisements and discovery. The Process describes how to use the service and to define orchestrations. The Grounding describes the technical information needed for invoking a service. This OWL-S based ontology is extended in the profile with a QoS ontology parameter. The QoS ontology is connected to the service profile through the Service Parameter concept and the QoS concept. A first version of the OWL-S LiQ ontology (without the property names) is as shown in Figure B.1. It shows some top level elements defined in OWL-S and also the additional QoS element added in OWL-S LiQ. It defines the conceptual parts of a service. Advertisements are possible through the profile, the process and orchestration enabling process concept, and the physical invocation details listed in the WSDL defined in the grounding.

OWL-S LiQ describes the very simple advertisement setup using the profile part of the ontology. Based on limited processing and network resources often found in MANETs we decided to eliminate as much as possible. The new QoS concept is used in the profile to help find the most suitable service for the client's possibilities and the service abilities.

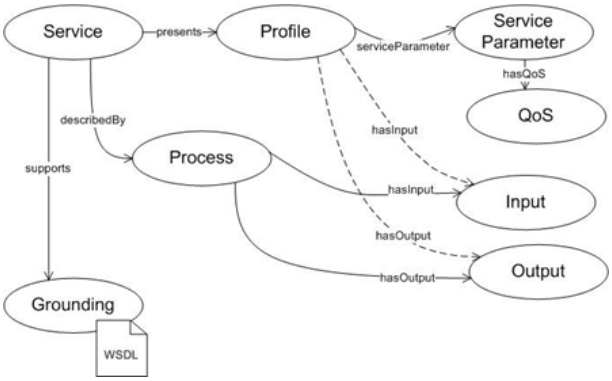


Figure B.1 OWL-S Lite with QoS ontology

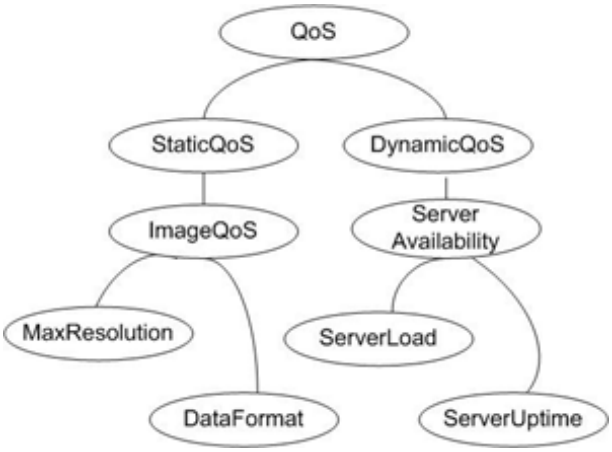


Figure B.2 OWL-S Light with QoS, the QoS ontology

The process part of the ontology describes how a service invocation flow is to be executed. Both CompositeProcess and AtomicProcess (not shown in the figure) are subtypes of the process concept and has by this input and output defined.

Figure B.2 shows the first edition of the new QoS ontology. The ontology is limited to our example but will be expanded when needed.

To sum up this service ontology we see that concepts are adopted from OWL-S in order to describe functional aspects of services, these concepts are used for reasoning about the functional aspects of the services enabling clients to find services based on the service capabilities and properties. The QoS ontology used in the profile enables clients to filter discovery results based on QoS parameters. The QoS ontology enables reasoning on QoS the same way as for functional service capabilities.

B.4 Representing QoS with XML

In order to be able to include QoS information in the service advertisements, we need to express the QoS parameters using XML. To do this we use a simple XML schema which defines the

metadata in terms of an URI identifying a semantic profile. In addition to this URI, a list of QoS parameters can be expressed. For example, a simple image service providing output in a certain resolution but with a choice of different image formats can be expressed as follows:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<QoS xmlns="http://xml.netbeans.org/schema/QoS_metadata">
  <ProfileURI>http://my.profile.org/semanticwonders</ProfileURI>
  <Parameter value="320" name="X"/>
  <Parameter value="240" name="Y"/>
  <Parameter value="gif" name="format"/>
  <Parameter value="jpg" name="format"/>
  <Parameter value="png" name="format"/>
  <Parameter value="raw" name="format"/>
</QoS>
```

B.5 Distribution

The service descriptions are pre-distributed together with the service ontology and the other domain specific ontologies. When a server gets ready to advertise its service, it has a service template, which is used to define the appropriate URI of the service along with a list of dynamic QoS parameters. When the advertisements arrive at the client they are stored in the repository of adverts. At selection time these adverts are used to fill instances of service templates to enable semantic matching based on service functionality and non-functional aspects that might have changed from the first advertisement was received. Further details of the distribution mechanism are discussed in the following section.

There are several challenges that arise when attempting to perform service discovery in MANETs. Below we summarize some of the most important challenges as identified by [15]:

- Dynamic environments may lead to frequent change in both service metadata (service descriptions) and the topology of the nodes that are part of the system. Frequent topology change means that both service nodes and registry nodes can come and go.
- A proper service discovery architecture for such an environment would reduce the amount of manual configuration, enable automatic discovery and selection of relevant services, and offer a complete and up-to-date picture of the services available at the given point in time.
- Service discovery should work in environments disconnected from the Internet. Moreover, it should be robust in terms of partial failure.
- The system should allow flexible resource utilization, since capacity (memory, CPU, storage) and connectivity can differ from node to node.
- The infrastructure should support different kinds of service description mechanisms, ranging from simple to rich (i.e. semantic) descriptions. Thus, both normal Web services as well as Semantic Web services should be able to use this infrastructure.

Considering these requirements, we look at each of the existing solutions for Web services in turn: Currently, there are three standards related to service discovery for Web services. All three

standards are under OASIS. The registry standards, UDDI [8] and ebXML [14], define central registries for use in wide area networks or local area networks. In order to find a service, the client needs to access the registry, search for an appropriate service, and choose a service from the query reply. However, this solution was developed for business use over the Internet, where connections are fixed, have a high bandwidth, high uptime, and nodes are immobile. In a MANET the network topology is unpredictable, and if the network is partitioned then only clients in the same partition as the registry will be able to discover services, whereas the clients in other partitions will not. This means that a client residing in the same partition as the service it needs may be unable to use it, just because it cannot access the registry to discover it. In contrast, a client that is able to access the registry may discover a service that it is unable to connect to, since the service resides in another network partition. These two aspects (shown in Figures Appendix B.3(a) and Appendix B.3(b)) are important drawbacks with registries if you attempt to use them in a dynamic environment such as a MANET.

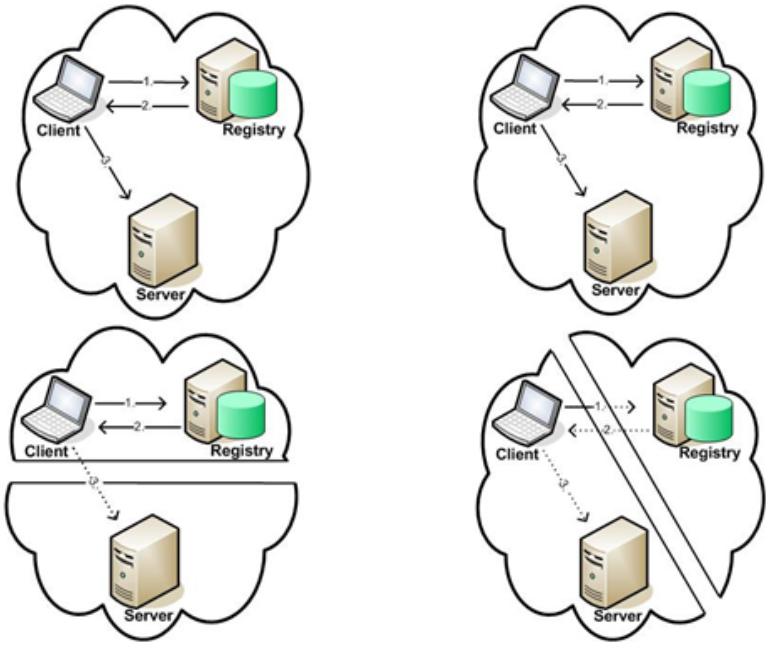
WS-Discovery [42] is a standard which attempts to remedy the drawbacks of registries. It is designed for use in local area networks, and is based on multicasting queries and responses. This means that by using WS-Discovery, you can discover services in your partition and use them. There is no single point of failure in WS-Discovery, and the query responses mirror the current network state.

All three discovery mechanisms support simple service descriptions. The registries have third party support for semantics as well. WS-Discovery in its current form has no support for semantic descriptions at all, and is thus not suitable for discovery of services beyond that provided by simple descriptions. As we can see, there is currently no standardized solution for Web services that supports semantic service discovery in MANETs, and there is a need for such a solution. Lacking a standard, we have implemented the necessary functionality in an experimental solution which performs distribution of Service Advertisements in MANETs (SAM). The differences between the existing standards and our experimental solution are described in Table B.1.

B.6 Implementation of SAM

Our experimental service distribution mechanism is tailored to the specific requirements for service discovery in MANETs. It can function disconnected from the Internet since it hosts all the necessary data and metadata in a local repository. It is resilient to partial failure since it is a fully decentralized solution. Thus, it will still function if the network becomes partitioned – the clients in each partition will have a fully operational service discovery mechanism at hand. The mechanism is based on periodic dissemination of service advertisements, and the advertisements are cached locally in each recipient. Whenever a new update is received the cache is refreshed, and any outdated services are removed. There is a limited time each service can exist in the cache before it is deleted, thus ensuring that a query in the local cache will give an up-to-date picture of the available services.

An advertisement contains a list of services being provided by the node that sent the advertisement.



(a) The *liveness* problem: Partitioning of the network leaves the server inaccessible

(b) The *availability* problem: Partitioning of the network leaves the registry inaccessible

Figure B.3 Issues in MANETs when network partitioning occurs.

	Registries	**WS-Discovery**	**SAM**
Can work disconnected from the Internet?	UDDI: No ebXML: Potentially, since it has a repository which can host important data	Potentially, as long as you only need information from the WSDL and no additional schemas	Yes
Resilient to partial failure?	No	Yes	Yes
Supports user defined metadata?	Yes	No	Yes
Supports semantic descriptions?	Yes, third party support	No	Yes
Gives a current and up-to-date picture of services?	No, requires services to actively deregister before they disappear	Yes	Yes
Distribution method	Unicast-based: Client connects to the registry and executes a query	Multicast-based: Client queries the network, and gets responses back	Multicast-based: Periodic dissemination of available services. Client queries local cache.
Gives the location of services?	Perhaps, could be supported through user added metadata	No	Yes
Suitable in MANETs?	No	Yes	Yes

Table B.1 Capabilities of Web services service discovery standards compared to our experimental SAM.

```
<?xml version="1.0" encoding="UTF-8"?>
<ns0:ServiceDiscovery  xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'
   xmlns:ns0='urn:no:ffi:servicead'
   xsi:schemaLocation='urn:no:ffi:servicead FServAd.xsd'>
    <ns0:service address="http://my.host.org/service1"
  hash="F82F0CFAD205B9A88907D24DF1ADFF94BF5BD70D"></ns0:service>
    <ns0:service address="http://my.host.org/service2"
  hash="47FC70C359DED9E80BE5A476C9E2DEEF0EB12638"></ns0:service>
</ns0:ServiceDiscovery>
```

Figure B.4 An example advertisement using only the mandatory fields.

This list contains entries uniquely identifying the service, and any metadata associated with the service. If the node chooses to report its position (e.g. if it is fitted with a GPS), then the positional data (e.g. latitude and longitude - the field uses the PositionDataType (PDT) from NFFI version 1.3 (STANAG 5527)) is sent along with the list of services. ? Basically, an advertisement contains the following data:

- Position (optional)
- Service list
  - Unique service ID (required), endpoint URL (required), metadata (optional)
  - (Possibly more service entries...)

The endpoint URL is the invocation address of the service being identified by the Unique service ID at that particular node. In other words, this is the endpoint address which can be found in the service definition of the WSDL. This URL is the only dynamic aspect of a Web service description, as the rest of the WSDL can remain static from deployment to deployment. Based on this observation, we found that by removing the endpoint URL from the WSDL, we would get a WSDL which could uniquely describe all services of the type defined therein. A WSDL is a large XML document, so to reduce the bandwidth requirements of the advertisement distribution mechanism we chose to use a hash over the WSDL without endpoint to uniquely identify a service. Metadata associated with the service is optional, but for the purposes of semantic service discovery, this field must be used. If it is left empty, then no semantic reasoning over QoS can be done by the client (see Figure B.4 for an example with only the mandatory fields of the advertisement being used). The metadata field is thus a flexible measure provided by the solution, in that it can be used by regular Web services for discovery (i.e. just ignore the field) or by Semantic Web services for added value and reasoning capabilities (see Figure B.5 for an example of all fields of the advertisement in use).

The idea is to pre-distribute service and domain ontologies to the clients with a set of service instance templates. A service instance template is a definition of a specific kind of service, but without the values of the dynamic QoS parameters. The grounding points to a WSDL without a specified endpoint. In addition to the slots for WSDL hash and endpoint, the advertisements also have the metadata slot which we use for a semantic description of the service and a list of dynamic

```
<?xml version="1.0" encoding="UTF-8"?>
<ns0:ServiceDiscovery  xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'
   xmlns:ns1='urn:nato:fft:protocols:nffi13'
   xmlns:ns0='urn:no:ffi:servicead'
   xsi:schemaLocation='urn:no:ffi:servicead FServAd.xsd'>
    <ns0:position >
        <ns1:dateTime>20090327111334</ns1:dateTime>
        <ns1:coordinates>
            <ns1:latitude >35.5310533333333</ns1:latitude>
            <ns1:longitude >67.4055644444445</ns1:longitude>
        </ns1:coordinates>
    </ns0:position>
    <ns0:service address="http://my.host.org/service1"
  hash="F82F0CFAD205B9A88907D24DF1ADFF94BF5BD70D" metadata=myMetadata></ns0:service>
    <ns0:service address=" http://my.host.org/service2"
  hash="47FC70C359DED9E80BE5A476C9E2DEEF0EB12638" metadata=myMetadata2></ns0:service>
</ns0:ServiceDiscovery>
```

Figure B.5 An example advertisement with position and metadata for each service.

QoS parameters defined in a list with names and values (e.g. the metadata presented in Section 4.2).

See [30] for further details about SAM.

B.7 Service selection

Clients receiving these announcements handle the advertisements as previously described, ready to populate service instances during selection. If a new announcement arrives from the same provider about the same service the old announcement is overwritten.

Service selection is the process where the client discovers, orchestrates and invokes services. Non-semantic Web Service technology has limitations in its static nature. WSDLs define where the service is located and the data type of the messages. BPEL orchestrations are defined before runtime. Adding semantics to Web services could enable dynamicity at runtime, where services are found based on their ability and orchestrations are done on-the-fly to satisfy client requests.

The pre-distributed ontologies are used by the client to formulate a service request. Requests are matched against service profiles and by this the request has the same format as the service templates. The client defines the desired service based on conceptual descriptions of input, output and QoS parameters. Requests are matched and the service advertised to the client.

Figure B.6 shows how a vehicle can have a service repository and a list of received advertisements ready to be searched if services are needed. To define the difference of equipment in a MANET, Figure B.6 also shows how a soldier with a PDA and a camera can be a client as well as a provider.

Figure B.7 shows a camera ontology. The model shows that Camera subsumes WebCamera. This means that the WebCamera concept has inherited all its properties from the Camera concept. In addition, the WebCamera also has its own specific properties. This means that a WebCamera is in

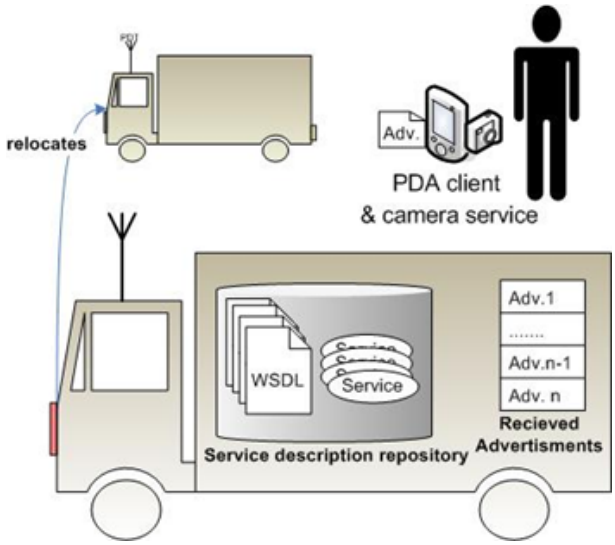


Figure B.6 Mobile service provider and consumer

fact a camera, and a Camera could be a web camera.

In our solution we define service matching as a stepwise process. The idea is to divide the matching into two general steps. First we match based on functional aspects of the service through input and output matching, then we match based on QoS parameters.

In our solution the second step is to match on the QoS parameters. QoS matching is based on the same principles as input and output matching. Based on the result of both matching processes there could be the need for orchestration of two or more services. For example, a service delivers observation data in raw image format. The service takes position as input and that matches the client's requested input parameter, but the output of the service does not match as the client application uses JPEG format. A conversion service available takes raw image data as input and produces JPEG data as output. By orchestration of these services the client gets the requested service. An example of reasoning is if there are other available services that would satisfy the client, but this is not obvious without reasoning. Reasoning on a conceptual level could lead the client to an alternative service. If the client wants surveillance footage from a specific location it could be that there are no available services delivering video from that location, but there are several soldiers near the location with cameras, based on the fact that pictures also could be surveillance footage this service is proposed.

The next step is to find the service advertisements with the same profile hash value from the list of service advertisements received. Instances of the template are populated with QoS parameter values, and a matching of client request QoS parameters and service profile QoS parameters is performed. If there is a service with satisfying QoS the service can be invoked. If there are no QoS matches a new service discovery is launched in order to try to find a service that can be used in an orchestration to deliver a result to the client.

Figure B.8 shows the idea of a conceptual matching process as described earlier. The different

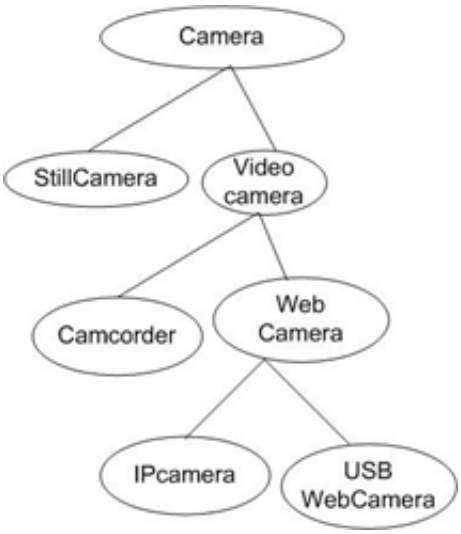


Figure B.7 Camera ontology

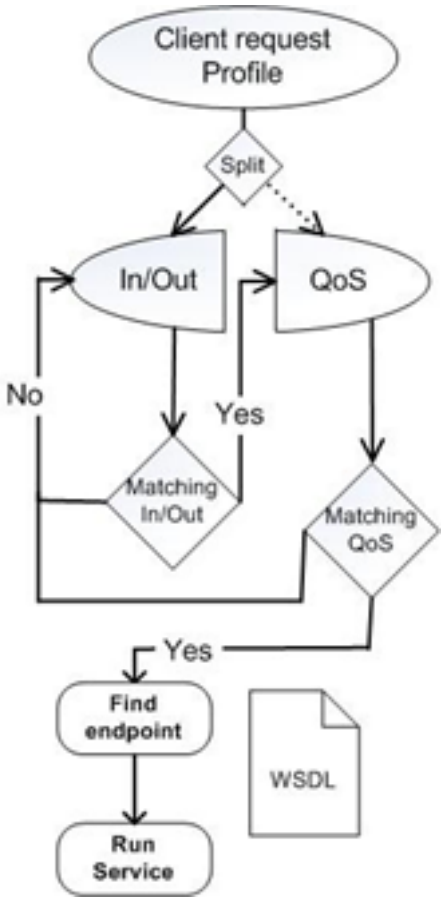


Figure B.8 Conceptual service matching process

services available are instantiated and we are to ready search for services. A client's request is split in two halves. One part has the input and output parameters and the other has the QoS parameters. Before matching on QoS we search using the functional input and output parameters. If we get a match, then we match the QoS of the service to the user's QoS needs. If there is no match, an attempt to orchestrate is initiated. Orchestration uses the client request input as input to the orchestration matching, and if there are any matches the orchestration process tries to match services with the first orchestration service output as input and the clients service request output as output. If this results in a match the QoS has to be. Let's take a few steps back and assume that the first match was OK, and that a QoS matching now is initiated. If this matching does not return any satisfying results, we need to start again and try to orchestrate services based on QoS parameters. For example, a client requests a service with location as input and picture as output with the QoS resolution to the value pair (X, Y) = (320, 240). This corresponds to QVGA, which is a common resolution for handheld devices such as PDAs. In our template we find a service match, but in the QoS matching we find that the service has a QoS resolution set to (1024, 768). This corresponds to XGA resolution, which is common in larger devices such as laptops. This means that a new service search is started, but this time for a service that has picture as input, picture as output and QoS resolution parameter set to (320, 240). By performing an orchestration with the originally found picture service and this rescaling service, an execution chain consisting of the picture service and the rescaling service would yield the desired output. It would match both service type and the QoS requirements. By orchestration of these services we now have a solution for the client request and the services can be invoked.

To invoke the services we need the physical address of the service. The semantic service descriptions have a grounding instance. When the individual service instances are populated the WSDL in the advertisement are also populated with the appropriate endpoint and the grounding is set to point to this WSDL.

B.8 Implementation

As a first step towards QoS aware Semantic Web Services in MANETs we have implemented a demonstrator based on the OWL-S API [51] that selects, orchestrates and invokes services based on standard Semantic Web Services input and output.

Figure B.9 displays the demonstrator system selection pane. For searching and selecting services a request is defined with a definition of input and output. The user requests surveillance footage of a given area and the wanted service is by this described to take position as input and JPG picture format as output.

Figure B.10 shows the services found during search for services. Performing the search the system does not find a single service that match the user's requirement. Performing a search for an alternative the system finds that an orchestration of the Picture service and the Conversion service would fulfill the user's requirements. The picture service takes a position as input and delivers GIF formatted pictures and, as the conversion service accepts GIF as input and returns a

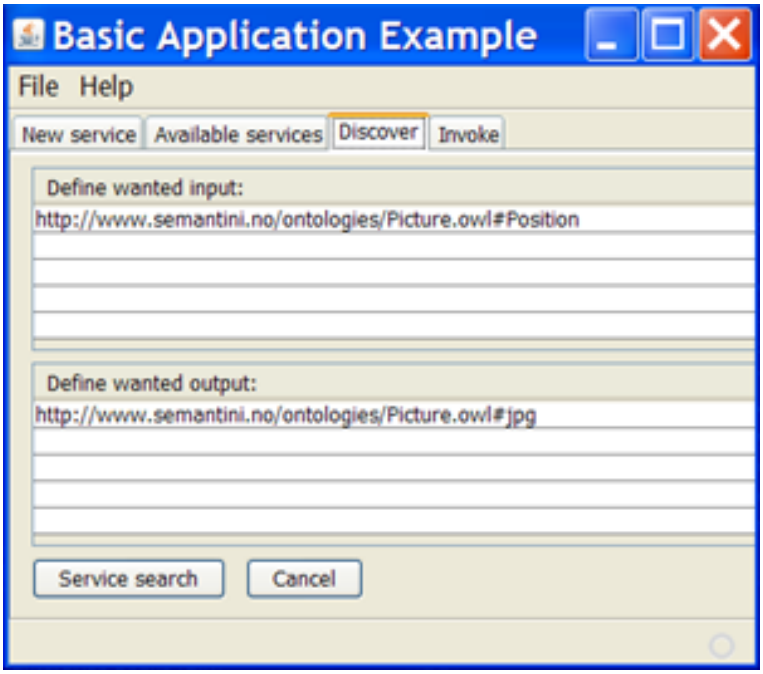


Figure B.9 Service selection

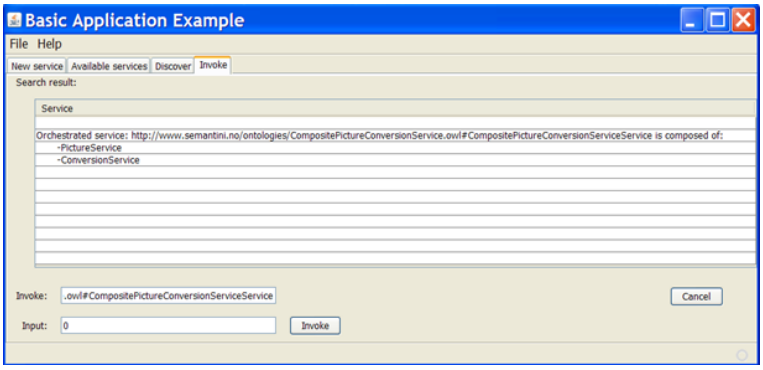


Figure B.10 Service search result

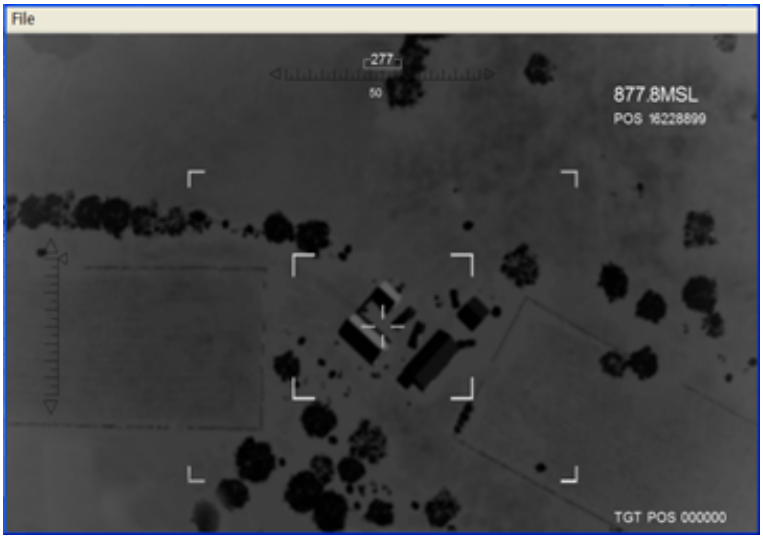


Figure B.11 Result picture

converted picture in the JPG format, a combination of the two would serve the user. A new service description is defined automatically and is ready for invocation.

Figure B.11 shows the end result of the service invocation. After invoking the new service description the user does not participate in the steps of invoking the two services as the system uses the new description to do this automatically. The orchestrated service returns to the user the wanted picture in the correct format.

Figure B.12 illustrates the improvement of using Semantic Web services in contrast to just Web services technology for applications depending on available services. The user interfaces a simple example application, getting surveillance footage by simply pressing the “Get image” button. If plain Web services are used in the background the service lookup would be unsuccessful, as there are no services able to respond successfully to the request. If, on the other hand, Semantic Web services are working in the background we get the runtime semantically orchestrated “Composite Picture Conversion Service” (see Figure B.10) which returns a picture with correct formatting shown to the user. Using Semantic Web services in contrast to plain Web services would in such cases have a binary outcome, meaning that plain Web services do not complete the task while Semantic Web services complete the task using runtime orchestration. The user of the surveillance footage application is none the wiser of the complex actions carried out behind the scenes, happy to get the surveillance footage requested. For further details regarding the expressive power of Semantic Web services vs Web services, and the benefit of using semantic technologies in military networks, see our paper [30].

B.9 Conclusion and future work

In this paper, we have presented our experimental prototype which brings QoS awareness to Semantic Web services discovery and orchestration. Our contribution is twofold: First, we have

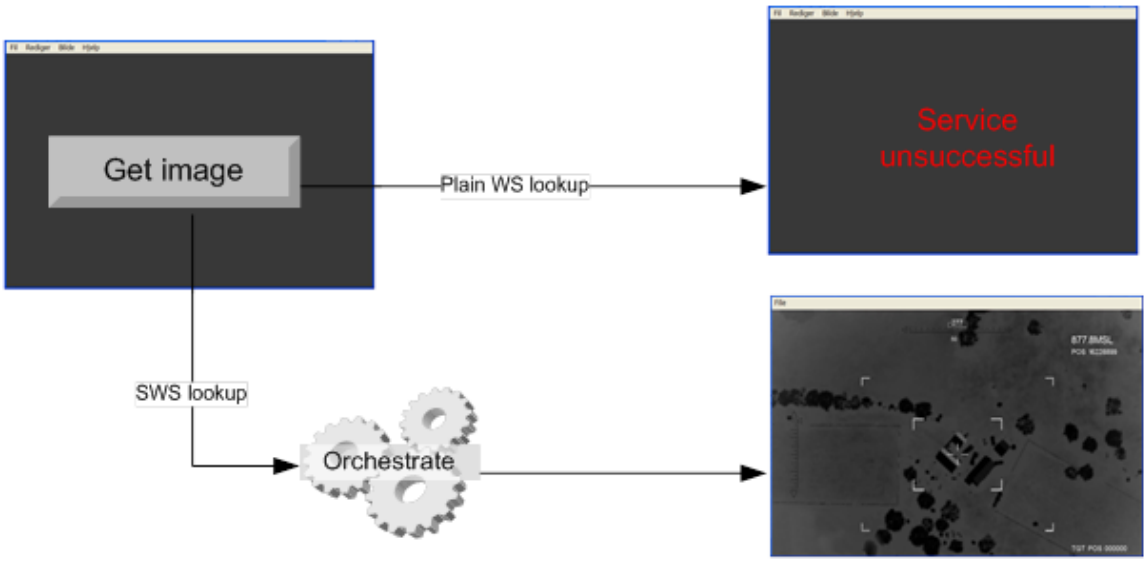


Figure B.12 Service execution

implemented a novel mechanism, SAM, for distributing service advertisements in dynamic networks. Second, we have defined an OWL-S based service ontology which we call OWL-S LiQ. By using SAM to provide an up-to-date view of available services and corresponding QoS parameters in a MANET, we were able to employ OWL-S LiQ to perform selection and orchestration of the available Web services based on both service types and specific QoS requirements. The system is capable of having both high end clients such as laptops and low end clients such as PDAs present, and accommodates their specific needs based on the service description, service QoS parameters, and the QoS profile of the specific device. Such a system is valuable in a military operation, where devices with different capabilities take part in the operation and need to access various services and tailor them to their available resources. Semantic Web services can provide the necessary agility and interoperability to heterogeneous systems that need to be interconnected in joint and/or combined operations.

Future work consists of expanding the QoS support of our OWL-S LiQ ontology, so that it can describe a wider array of QoS parameters. Also, the device profile and matching capabilities need to be expanded from the simplistic high/low end client profile of today to a more flexible and configurable type that can support any kind of mobile device.

In addition, our current service matching is done based on a simple fail/success criterion. Due to the hierarchical nature of ontologies, several different degrees of matching [40] can be defined, and we will extend our service matching algorithm with support for this.

Appendix C Automated QoS-aware Service Selection and Orchestration in Disadvantaged Grids

This paper was written by Trude Hafsøe, Frank T. Johnsen, and Marianne Rustad. It was published at the MCC 2010, Wroclaw, Poland.

Abstract

Web services technology has been identified as a key enabler for NATO Network Enabled Capabilities (NNEC). However, this technology does not provide a standardized means of supporting Quality-of-Service (QoS). QoS support is a must in military networks, especially in the disadvantaged grids where resources are very limited. In this paper we discuss how Web services technology can be extended with the necessary QoS support by leveraging semantic technologies. By using a decentralized service discovery mechanism, we enable a robust means of discovering Web services and the associated QoS metadata. The contribution of this paper is our novel two-step algorithm for QoS-aware semantic matchmaking, which enables automated service selection and orchestration based on QoS criteria.

C.1 Introduction

The NATO NEC feasibility study [2] has provided a set of suggestions for achieving interoperability between NATO nations: First, a Service-Oriented Architecture (SOA) approach should be adopted, since the loose coupling provided by this paradigm is well suited for interconnecting different military entities. Further, the study suggests that Web services technology can be used to implement such a SOA. Web services are in widespread use in civil systems today, and thus commercial products are readily available. However, in civil systems Web services are usually used in static networks. In military networks, particularly tactical networks, there is a need for dynamic solutions [13]. This means that we must be able to advertise, discover, and select among available Web services at run-time. In previous work, we have shown that it is feasible to discover Web services in disadvantaged grids [26]. However, we were still left with manual service selection. In this paper we address this issue, and show how to achieve automated service selection and orchestration by using Semantic Web services.

Previously, we have addressed the service advertisement and discovery steps of the Semantic Web services lifecycle [26] shown in Figure C.1. This paper focuses on the selection and orchestration steps. To achieve this automatically, we need machine-processable semantic descriptions of the services. From Web services we have the WSDL, which defines the interface of the service. By grounding this WSDL in an OWL-S ontology [40], we get the desired rich service descriptions featuring properties that enable us to let a computer reason about which services are suitable for invocation.

The intended service matching in this paper is based on the matchmaker described in [40]. In our solution we define service matching as a stepwise process. The idea is to divide the matching into

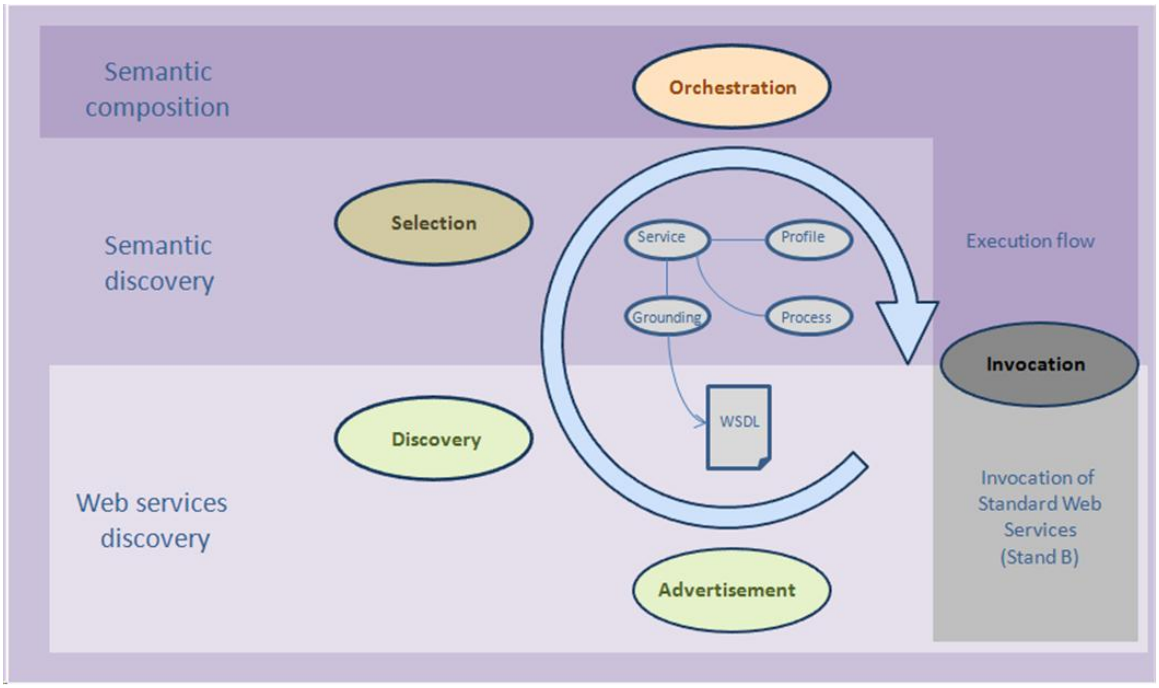


Figure C.1 Semantic Web services lifecycle

two general steps: First, we match based on functional aspects of the service through input and output matching.

The second step is to match on the Quality-of-Service (QoS) parameters. QoS matching is based on the same principles as input and output matching. Based on the result of both matching processes there could be the need for orchestration of two or more services. QoS is defined by both static and dynamic aspects, and both types need to be considered to be able to choose the "best" service. In this paper we focus on how to choose the "best" service utilizing stepwise matchmaking and orchestration where needed. This allows us to choose a service with functional aspects and QoS which fall within the desired profile of the client, while at the same time limiting the bandwidth use, so that the scheme can be utilized in a disadvantaged grid.

Semantic Web services technology is well known from use in civil systems. On the Internet, this technology is gaining widespread use. However, there is much overhead in this technology. In this paper, we show how we can address this overhead by using our efficient service discovery mechanism [26], and further discuss how the automated reasoning capabilities of machine-processable semantics are beneficial in military systems. The technology is particularly suitable for use in tactical networks. The technology can help provide interoperability between service descriptions from different nations, which is important in e.g., a multi-national NATO response force.

C.2 Related work

Sliwa and Duda [58] have identified four sub-problems regarding QoS in a Web services environment:

1. How to state and process the QoS requirements of the Web services client?
2. How to trigger the QoS mechanisms in the network layers?
3. How to decide if the service can be realized with the desired QoS?
4. How to adapt Web services operation when network resources do not allow sending information in a standard way?

Our work addresses sub-problems 1 and 3 in particular, providing a more agile and interoperable means for QoS expression and selection than what is currently available. We will now discuss some of the most prominent existing solutions (it should be noted that all of them are experimental, since there is currently no standards for QoS related to Web services). We do not cover sub-problem 2 in this paper, since that is handled sufficiently by other, complementary work which we present below. Sub-problem 4 is related to service invocation, and thus beyond the scope of this paper where we discuss an algorithm for use during service selection. However, this problem can be overcome by using a mediation service (see details in [58] and [57]).

Usually, one refers to network level QoS when talking about service quality. At the network level, you can measure available resources, and can prioritize traffic in traffic classes according to different criteria such as delay, jitter, bandwidth, etc. At the network level, resources are quantifiable, such as delay or percent packet loss [3]. At this level, the Differentiated Services (DiffServ)[5] mechanism can be used to ensure that some traffic is prioritized, a task that it has been shown to perform quite well through actual use and simulations, see experiment details in [34], [3], and [35]. This work is complementary to ours, in that we focus on application level QoS. Network level QoS is important for any system, and can be employed for all traffic. Application level QoS, on the other hand, is application specific, and needs to be determined for each and every application by itself.

Our work focuses on using semantics to express QoS requirements (i.e., sub-problem 1) and perform automated step-wise QoS matching and service selection (i.e., sub-problem 3). These issues are handled at the application layer. To achieve end-to-end QoS in a network, it is important that QoS requirements can be mapped to network layer QoS mechanisms, so that the network can enforce and prioritize the traffic accordingly (i.e., sub-problem 2). This is done in the work

[5] DiffServ relies on tagging the IP header's type-of-service field with a bit pattern corresponding to a certain traffic class. This enables DiffServ-enabled routers to prioritize the IP packets accordingly with a deterministic per-hop-behavior (PHB). DiffServ is covered by several RFC's: RFC 2474 (Definition of the Differentiated Services Field in the IPv4 and IPv6 Headers), RFC 2475 (An Architecture for Differentiated Services), RFC 2597 (Assured Forwarding PHB Group), RFC 3140 (Per Hop Behavior Identification Codes), RFC 3246 (An Expedited Forwarding PHB), RFC 3260 (New Terminology and Clarifications for DiffServ), and RFC 4594 (Configuration Guidelines for DiffServ Service Classes).

by Tian et al. [60], where Web services QoS demands are mapped to DiffServ classes to ensure differentiated services in the IP network. In [58], Sliwa and Duda discuss a framework for adaptive Web services supported by QoS in the network. They use SIP to signal an application's QoS requirements, and use DiffServ to handle network level QoS. Further, they discuss how a mediation service can be used to adapt Web services traffic to better utilize the available resources and meet the required QoS demands, i.e. the QoS profile given by the client (i.e., sub-problem 4). This work uses a non-Web services technology (SIP) for signaling QoS. In our work, we use ontologies and semantic Web services for this purpose instead. In fact, we use semantic technologies to express and reason about all our QoS parameters and profiles, whereas the work by Sliwa and Duda seemingly use ontologies only for filtering requirements, and not the rest of the QoS attributes. This makes our work complementary to theirs, in that our novel QoS expression and reasoning algorithm provided in this paper could be used as a replacement for those specific parts of their suggested solution, that is using our algorithm for sub-problems 1 and 3, and their solution for sub-problems 2 and 4. This would yield a complete and interoperable solution for QoS for Web services.

Wang et al. [67] create a QoS-aware selection model for semantic Web services. They create a simple QoS ontology for expressing qualitative metrics and values. These metrics and values are arranged in a matrix, which is used for matching provided QoS with the client's QoS profile. The QoS expression and matching is limited to numerically quantifiable QoS aspects, making this solution best suited for exposing network parameters such as delay, packet loss, etc. to an application, thus providing a simplistic solution to sub-problems 1 and 3 above. Our work is more general than this, allowing also for non-numeric QoS attributes. Any parameter (provided it has been defined in the ontology) can be reasoned about as part of a QoS selection process.

See the study by Tran and Tsuji [61] for a complete survey of how to express QoS using semantic technologies for Web services. Their work classifies and discusses the expressive power of existing languages for expressing QoS parameters using semantic technologies, i.e. algorithms that can be used to address sub-problem 1 above. This work is complementary to ours, in that basically any of these description types can be used with our solution. The important part is that QoS values and demands are expressed in a uniform and machine-processable fashion, so that the algorithm we provide in this paper can perform its function.

C.3 Service Advertisements

There exist three service discovery standards for Web services; the UDDI and ebXML registries, and the decentralized WS-Discovery mechanism. The registries can store QoS information along with service information, but they are not suitable for use in dynamic environments such as mobile tactical networks. In such networks a registry constitutes a single point of failure, and more robust mechanisms should be used instead. WS-Discovery is decentralized and thus more robust, but it cannot carry QoS information. This means that there is a need for a new service discovery mechanism for use in military mobile tactical networks. SAM is an experimental service discovery

protocol [26] that we have developed to address this need. It can disseminate information about Web services, position information, and semantic metadata (e.g., QoS data) in an efficient manner in small, highly mobile networks, providing each node with an up-to-date view of available services and corresponding QoS information.

SAM can be used as a service discovery library, where client software can query for a list of one specific service or just an exhaustive list of all currently available services. The matchmaker discussed in subsequent sections in this paper utilizes the latter of the two query modes, and obtains a complete list of services and attached QoS information when an invocation request is made. This enables the matchmaker to make relevant choices based on both service types and QoS parameters.

C.4 Service Matching

The service matching algorithm described here is intended for use in limited capability networks, where almost all service discovery and orchestration will be done to meet the run-time discovery need of clients. Typical clients will be disadvantaged users that do not have the time or ability to spend time doing manual service selection. If for instance a service that a user is using fails, it is important that the system is able to look for a replacement service without requiring the user to manually intervene and request and select a new information source. Because of this, our algorithm is designed to support automated selection and orchestration based on not only the functional aspects of a service, but also the QoS requirements of the users.

When doing matching based on QoS parameters we need information about both the QoS offered by the service and the QoS requirements of the service user. The information about the QoS offered by the service can be found as a part of the service advertisements, and will be readily available to the matching software. The requirements of the clients can be given in one of two possible ways; either it will be supplied by the client as a part of the service request, or it will be available as a part of a client profile.

Neither service nor client is obligated to provide this QoS information, as it cannot be assumed that all services and clients will be QoS enabled. In the case that QoS requirements of the client is not available, a simple matching on input and output parameters is performed. In the case when the client has provided QoS information, we need to match these requirements with the QoS the service offers. This is done according to the matching algorithm described below. Matches are then ranked based on how well they fulfill the client's requirements, with services that meet both required and optional QoS parameters being ranked highest, followed by services that fulfill the minimum requirements of the client. It is also possible that we do not know if a service is able to fulfill the QoS requirements of a client (for instance if QoS information about that service is not available), and these services are ranked next, just above the services that we know cannot fulfill the client's QoS requirements.

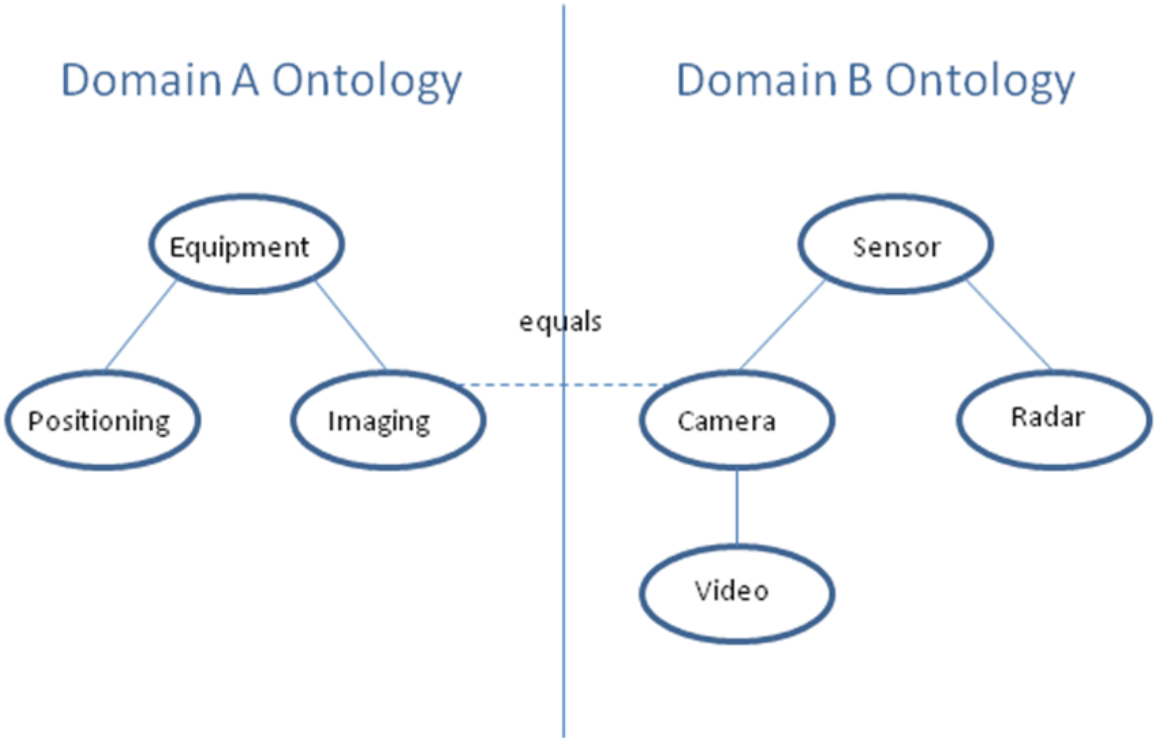


Figure C.2 Ontologies

C.5 Degrees of Match

A central concept when discussing semantically enabled service matching is the ontology, which is a formal representation of the concepts within a domain, including the relationships between concepts. When using such an ontology to perform service matching, the different concepts (which in this case represent different service types) are often arranged in a graph structure. Figure C.2 shows a simple example of service types that are hierarchically organized in two separate ontologies, but with an extra relationship being defined between one concept from ontology A and one from ontology B.

The structure of these service ontologies means that we can get matches of different degrees, depending on whether the matches found are of the exact service type we are looking for, or of a related service type. Most existing definitions for degrees of matching seem to be geared towards manual service selection. In these cases the goal of the matchmaking process seems to be to provide the end-user with an extensive list of services that match the request, and the main concern is to avoid eliminating services that might be able to meet the needs of the user.

In our case, we consider automated service selection. When performing automated service selection the goal of the selection process is not to find all the services that match a specific set of criteria, but rather being able to finding the one service that best meets the user's needs without requiring any manual input from the said user. Because of this, we have created our own method for ranking matches. It is based on the concepts from [40], but ranks matches in a different order.

The highest degree of matches are those that are defined as an exact match, which we define to be a match of the same service type as the request was for, or a match of a service type that has been defined to be equal with the service type that was requested. This means that a request for a Camera service in domain B will yield both domain B's Camera type and domain A's Imaging type as exact matches.

The second degree of matching is when a requested service type is a superclass of the matched type. A request for a Camera service will thus yield a service of the type Video as a second degree match.

The third degree of match are matches where the requested service type is a subclass of the offered service type, meaning that a request for a Camera would yield a service of the Sensor type as a match This type of match is considered the lowest degree of match (other than a complete fail), as there is no guarantee that the service found will support the functionality requested. Because of this, these types of service matches should not be considered a successful match for an automated service selection algorithm. It is however useful to also classify these types of matches in case the user wishes to perform a manual selection.

C.6 The Matchmaking Process

The matchmaking process is a step-wise algorithm, and during this process we maintain a ranked list of possible services matches. Services, both simple and orchestrated, that match the client's request are added to the list when they are found, and ranked both according to the type of match they provide and on number of services involved (for orchestrated services). When this process is completed, the client is provided with the highest ranked service.

As we are providing automated service selection, the client should only be provided with one service that will be automatically invocated. It is however important to keep in mind that automated processes like these can fail and having a back-up mechanism is vital to ensure that an end user can intervene and perform a manual selection if needed. Using our solution, this can be done by providing the end user with access to the ranked list of services should such a need arise. In the case that no matching services can be found, or the matching services do not provide a sufficient level of QoS, we have the option of performing an orchestration of services. This way we can create a new service that better fulfills the needs of the client. These orchestrated services will require multiple service invocations, in most cases leading to more network traffic being generated. Due to this issue, orchestrated services are ranked below basic/simple services that provide the same degree of match. Orchestrated services that are otherwise identical are then ranked on the number of services that are a part of the orchestration.

C.7 The step-wise service matching algorithm

Our matching algorithm, shown in Figure C.3, is based on a two step approach; the first step being a simple functional matching based on input and output parameters, while the second step

```
STEP ONE: Input/output matching
  a) create a list of all available services where input and output match the client's needs
  b) if the list is empty: perform an orchestration
       - if the list is still empty: failure
       - if the list is not empty, goto 1c
  c) if we do not have QoS information about the client, return first list item,
      if we have the client's QoS requirements, provide list as input for step 2)

STEP TWO: QoS matching
  a) sort the services list according to the client's QoS demands
  b) if the top list item does not fulfill the full QoS requirements, perform step 1b
     - if result from 1b is the same as the list just evaluated in 2a: failure
     - if new matches were added to the list returned from 1b, goto 2a
  c) choose the service (or orchestrated service) at the top of the list, and let the client invoke it
```

Figure C.3 Step-wise service matching algorithm

considers the QoS needs of the client and matches it with the QoS offered by the services.

The simple parameter matching performed in step one allows for a quick elimination of services that do not fit the request at all. Once this is completed we will have a list of available services ranked according to match type, exact matches being ranked first. We then consider the QoS matching:

In order to be able to compare the QoS requirements of a user to the QoS offered by a service, a minimum amount of knowledge about the QoS parameters is required. The parameters need to be agreed upon up front, so that both user and service know which parameters will be understood by the other party, how to interpret the parameters, and how to use the parameters in an orchestration. This means that a standardized way of describing QoS parameters will be needed before large-scale QoS support can be integrated into an operational network.

For the algorithm we are proposing in this paper, we have not created a full classification scheme for parameters, but rather concentrated on the type of information we need about the parameters, and applied this to some example parameters. Common to them all is that we, in addition to knowing the parameter name, range of allowed values and the parameter's actual value, need to know the type of the parameter, how to evaluate the parameter and how to use that parameter in a composition. Table C.1 gives an overview of the parameter types we considered when evaluating our algorithm, along with the possible match criteria and composition calculations that we identified for our example parameters. This list is in no way a complete listing of all possible parameter types and evaluation criteria, but serves to illustrate the type of information that is required to be able to perform automatic selection and matching based on QoS.

After having done this QoS matching, we determine if the highest ranked service match found is sufficient to meet the client's requirements. In our algorithm only exact matches that fulfill both the optional and required QoS of the client are considered a sufficient match, but it is possible to extend this to include other types of matches as well. If we can determine that a good enough match has been found, the best match found is returned. Otherwise, we enter the orchestration

Type Match	Criteria	Composition
Range	Within, Within or same as, Outside	Smallest common range, Added ranges, etc.
Set	Full set match, Element in set, etc.	Union of sets, Intersection of sets, etc.
Value	Same as, Smaller than, Larger than, etc.	Addition, Subtraction, Maximum Value, Minimum Value, etc.
Boolean	Exists, Does not Exist	Required for all services, Required for at least one service, etc.

Table C.1 QoS matching

phase, where we find new service matches by combining the simple services into more complex ones. After having populated our match list with these "new" matches, we perform the two steps of our service matching algorithm a second time to ensure that the orchestrated services are ranked correctly.

C.8 Conclusion and future work

In this paper we have presented our algorithm for automated QoS-aware Web services selection and orchestration. We have discussed how we can match a client's QoS demands with an appropriate available Web service in a highly dynamic environment such as a military mobile tactical network. By using a specially tailored mechanism for service advertisement distribution, we are able to provide each network node with an up-to-date view of available Web services and corresponding QoS data. Using these data as input in our novel algorithm, we are able to leverage semantic technologies to automatically choose between the available Web services based on input and output matches while at the same time satisfying QoS demands. In our experiments we have expressed QoS in a proprietary language, in order to test a proof-of-concept implementation. We have shown that it is feasible to employ semantic technology to achieve automated QoS-aware service selection. When single services do not match the requirements, performing an orchestration of services in an attempt to create a new service invocation flow that, when considering the aggregated QoS values, satisfies the client's demands.

In the future, our work could be expanded in a number of ways. We are planning experiments in an actual tactical environment, now that our solution has shown promise in the lab. Furthermore, a standardized way of expressing QoS for Web services needs to be developed. The suggestions we make in this paper cannot be adopted by NATO unless one has a common understanding of the QoS parameters and the meaning of their values. Also, one must look into such issues

as admission control, which should preferably be solved using a standard, e.g. WS-Security. These and other issues must be addressed before one is able to create a complete, secure and interoperable system. The contribution of this paper is to provide one specific building block of such a system, i.e. the algorithm needed to match information about services and QoS to a client's specific needs, and automatically select the "best" service according to a set of pre-defined selection criteria.

Appendix D Cross-layer Quality of Service Based Admission Control for Web Services

This paper summarizes the Master thesis work of Øyvind Kolbu, who was supervised by FFI. The paper was written by Frank T. Johnsen, Trude Hafsøe, Mariann Hauge, and Øyvind Kolbu, and was published at IEEE HeterWMN 2011, Houston, TX, USA, 9 December 2011.

Abstract

Web services are in widespread use today. This paper discusses Quality of Service (QoS) concepts which are not covered by existing Web services standards, and focuses on application level solutions that will be important building blocks in the future. We implemented a QoS based admission control mechanism, which provides priority based access to the network, while at the same time avoiding overloading the limited network capacity that is available. We use cross-layer mechanisms to combine QoS mechanisms at the network layer with our Web services admission control broker. The preliminary experiments we have performed with Web services over emulated wireless links are discussed.

D.1 Introduction

Quality of Service (QoS) plays a very important part in allowing the limited resources available in multi-hop heterogeneous wireless network environment to be utilized so that they best serve the needs of the system users. In this paper we focus on QoS for machine-to-machine interaction using Web services. Current Web services standards cover aspects such as implementing single sign-on and security measures, but they do not cover QoS adequately, so experimental solutions are called for.

QoS is a central issue when discussing all types of systems design, ranging from multimedia entertainment systems built on Internet technology to tactical military communications systems.

Category	*Example parameters*
Performance-oriented	End-to-end delay and bit-rate (bit-rate)
Format-oriented	Video resolution, frame rate, storage format, and compression scheme
Synchronization-oriented	Skew between the beginning of audio and video sequences
Cost-oriented	Connection and data transmission charges, and copyright fees
User-oriented	Subjective video and audio quality

Table D.1 The five categories of QoS parameters.

The concept of QoS was first introduced in the area of data communication, in order to describe technical characteristics, such as delay and error rate. With the advent of multimedia applications, the QoS concept has been extended to cover end-systems as well, to enable an end-to-end control of the quality level. Consequently, QoS concerns all parts of a distributed system: client system, network, and server. According to [62], there are five categories of QoS parameters, and these are shown in Table D.1.

The performance-oriented QoS-parameters are typically associated with the lowest layers (i.e., closest to the physical resources) of a system, and therefore have an impact on parameters in most other categories. In other words, mechanisms employed by lower layers affect the services offered to higher layers. Typically, the Differentiated Services (DiffServ) framework [5, 19] has been identified as a good solution for network level QoS.

In addition to the network aspects, we also have to consider the needs of users and the fact that some system components may fail and become unavailable. These issues refer to user level QoS, so called Quality of Perception (QoP) [16], and Quality of Availability (QoA) [48]. QoA is a measure for availability; replicating services and data in order to maximize availability and minimize system downtime. QoP, on the other hand, is a qualitative measure, and not quantitative like network level QoS and QoA. QoP is not fulfilled if the user is not satisfied with the system's performance. Thus, the users' needs must be analyzed and quantified, so that the QoP requirements can be mapped to physical parameters [18].

When using Web services in radio-based networks with scarce resources, such as military tactical networks, one of the most limiting factors is the bit-rate that is available to the applications. In this paper we address this issue by providing a broker based QoS admission control mechanism, which aims to provide priority based access to the network, while at the same time avoiding overloading the limited network capacity that is available. Even though we focus on military networks, we base our solutions on civil standards and middleware. Thus, our findings should be of interest also for civil network applications.

Our Web service broker interacts with QoS mechanisms on the network layer, and dynamically adjusts the number of admitted requests based on information about available resources received from the network layer. This solution allows for a detailed and efficient management of Web services requests while still being in line with the QoS-mechanism and admission control elements that must be present on the network layer for the total network traffic.

The remainder of the paper is organized as follows: In Section D.2 we discuss related work. Our design and implementation of a prototype QoS based admission control is covered in Section D.3. The evaluation of our preliminary experiments is presented in Section D.4. Section D.5 concludes the paper.

D.2 Related work

Sliwa and Duda [58] have identified four sub-problems regarding QoS in a Web services environment:

1. How to state and process the QoS requirements of the Web services client?
2. How to trigger the QoS mechanisms in the network layers?
3. How to decide if the service can be realized with the desired QoS?
4. How to adapt Web services operation when network resources do not allow sending information in a standard way?

Our work addresses sub-problems 1, 2, and 3, providing an experimental means of QoS expression and selection for Web services. Previously, we have created a means of Automated QoS-aware Service Selection and Orchestration in Disadvantaged Grids [22] using semantic technologies. The work in this paper solves another, related aspect of QoS support in that it implements a QoS admission control scheme for Web services. We do not create a new technique to solve sub-problem 2 in this paper, since that is handled sufficiently by other, complementary work, e.g., [60]. Rather, we adopt their suggestion of mapping application level QoS to the underlying network mechanisms. Sub-problem 4 can be overcome by using a mediation service (see details in [58] and [57]), a solution that could be combined with the prototype we evaluate in this paper.

Chen et al. [7] have addressed the issue of QoS for Web services using a broker approach, in which the broker functions as a mediator between service and client. Their experiments have verified the broker approach as viable for Web service related QoS support, as it allows for better scalability and flexibility that a tighter, direct service to client connection. However, the tasks performed by the broker serve a different purpose than the work presented in this paper: The work by Chen et al. focuses on matching the requirements of a client to a service that can meet those requirements based on both negotiations with services and client feedback about services.

Tian et al. [60] have built an infrastructure for handling prioritization of Web service requests, with a focus on mapping application level QoS descriptions to priorities on the network layer. This includes traversal of two different network technologies, namely a wired network using DiffServ, and UTMS, with translation of network priority between these technologies. This work focuses on the signaling of priorities during transmission, and the findings from this work can be used in conjunction with an admission control mechanism such as the one presented in this paper.

D.3 Design and implementation

Building a complete QoS infrastructure is a complex task involving a number of different aspects, such as network usage, user satisfaction and resource availability. When introducing QoS support for Web services in a tactical military network one of the major challenges is adapting Web services to the limited resources available at the network layer. Applying simple modifications such as the use of data compression allow for more efficient usage of the available network

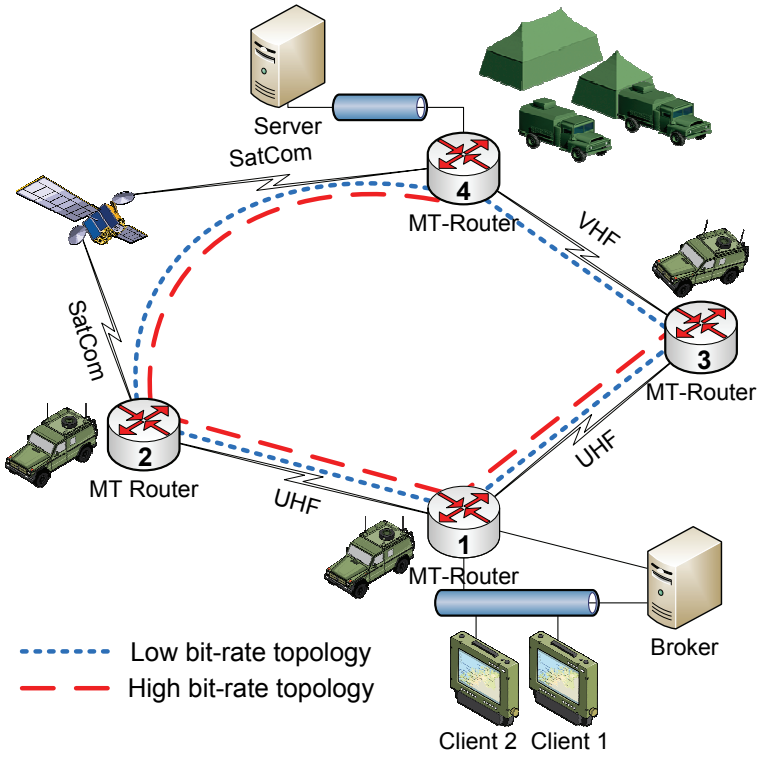


Figure D.1 Example network featuring Multi Topology (MT)-Routers

resources [38], but even with such techniques network resources, and bit-rate in particular, is a key limiting factor.

In order to prevent Web service traffic from overloading the network, while at the same time ensuring that important data gets the network resources it requires, a QoS based mechanism that provides admission control to the network is needed.

D.3.1 Network-layer QoS Scenario

The experiment described in this paper is set in a tactical network scenario as shown in Figure D.1. In this scenario we envision the following situation; requests for Web services from clients need to traverse a multi-hop heterogeneous network path to reach the server. The network path may include many different radio networks, and links based on different transmission technologies on the way from source to destination. This means that the network characteristics (e.g., bit-rate, jitter, delay) can vary substantially depending on the topology (and mobility) in the network.

We focus on the bit-rate characteristics of the network for this work. We want to deploy a broker-/admission control element for Web services on each platform-LAN. Due to the heterogeneity and mobility of this network, the broker must dynamically adjust the number of admitted requests to suit the present bit-rate capacity on the client - server path. Approximate information about the current maximum bit-rate on the path is made available to the broker by the network-layer QoS mechanism.

SBC/QoS-class	Traffic	High bit-rate topology	Low bit-rate topology	Base topology
NETR	Routing	N/A	N/A	X
VOICE	VoIP		X	
STREAM	Video, sensor	X		
BULK	Web services	X	X	
NORM	Best-effort	X	X	

Table D.2 Subset of QoS-classes (from [43])

In the heterogeneous network we use Multi Topology (MT) routing as one of our QoS mechanisms. This allows us to build several overlay topologies on the network. Each topology maintains its own routing table. For the work described here we build two topologies in the network; one topology that only allows high bit-rate links to participate in the path calculations, and another topology that involves alls links, but that can only guarantee low bit-rate paths. If the clients in Figure D.1 request a service from the server, the sum of the bit-rate requirements of these requests can equal the bit-rate allowed on the high bit-rate topology if the route via router 2 is available. If only the path via router 3 is available, the service requests sum must be lower than the allowed bit-rate on the low bit-rate topology.

We assume that all network traffic is tagged to belong to a specific Service-Based Class (SBC) as described in[43]. Table D.2 shows a subset of the proposed SBCs. Web services traffic can be present in several of these SBCs but the majority of Web services will naturally fit in the BULK class. The BULK class represents prioritized elastic traffic without rigid time constraints.

On the network layer the Multi Topology (MT)-routers associate one or more QoS-classes with one or more topologies (routing tables). In Table D.2 we have mapped the chosen SBCs with the two example topologies. In this table e.g., the STREAM QoS-class is only allowed on the high bit-rate topology while the BULK QoS-class are allowed on both topologies but different bit-rates will be available for the BULK class whether the low bit-rate topology or the high bit-rate topology is used. For more information on how MT-routing can be used as a QoS mechanism in a DiffServ-like environment for scenarios like this, see [23].

The MT-router also shapes the traffic for each link to a predefined value. The minimum of the predefined values for each link allowed in a topology then defines the maximum bit-rate available on the topology. This bit-rate is next divided between the different QoS-classes. The bit-rate capacity made available for the BULK class for each topology represented the share of the network capacity that the Web services broker manages. Other mechanism must be available at the network layer to handle admission control of the total network traffic.

D.3.2 Broker

Our mechanism for network admission control for Web services works in conjunction with QoS mechanisms at the network layer, and relies on an underlying priority handling mechanism to

ensure that important data gets priority during transmission.

For the experiment described here, we chose to use the BULK SBC for the Web services. The traffic is labeled with the BULK tag by the broker. Web services that bypass the broker will be treated as NORM (Best-effort) traffic and not be given the guarantees associated with the BULK class. The admission control performed by the Web services broker does thus only handle the BULK traffic.

Our admission control mechanism is broker based, and is designed to as far as possible utilize existing standards for Web services. A broker-based approach was selected to retain the loose coupling integral to Web services, which allows the admission control mechanism to be designed and implemented independently of the actual Web service deployment in the network being considered.

The admission control broker uses a role-based QoS scheme to make its decisions. That means that each user is assigned one or more roles, which represent the user's requirements for priority and timeliness. These requirements can vary per service the user accesses, which means that the priority a user should receive is determined by a combination of their role and the service they are attempting to access. This allows for prioritization based on what the user is going to use the information for, rather than simply who the user is. This priority scheme allows the broker to differentiate between the different demands for priority and timeliness of the different information flows.

For the admission control mechanism to function efficiently it needs to know not only how to prioritize between users, but it must also know the amount of resources it has available to distribute among the clients. In our mechanism, this aspect is handled by cross-layer mechanisms where the broker is interacting with QoS mechanisms in the MT-router [23]. The broker examines the MT-router's topology routing tables and can thus learn if the server in question is available via a high bit-rate path or only via a low bit-rate path.

The broker is implemented as a stand-alone entity which makes decisions on whether a client can be granted access to the network or not. The broker itself does not enforce this decision, and it is assumed that this enforcement functionality will be handled by the same mechanism that enforces security policies.

A client that wants to access the QoS benefits offered by the broker will have to contact the broker prior to sending its Web service request. Figure D.2 shows the communication steps between the involved entities:

First the client contacts the broker, asking for prioritized access to the network. If the broker grants the client access to the network, the broker performs the second step, in which it notifies any existing enforcement mechanism about the access rights the client has been granted. After the client has been notified of the fact that it has been granted access (step 3 in Figure D.2), the client sends its Web service request to the service, and receives a reply.

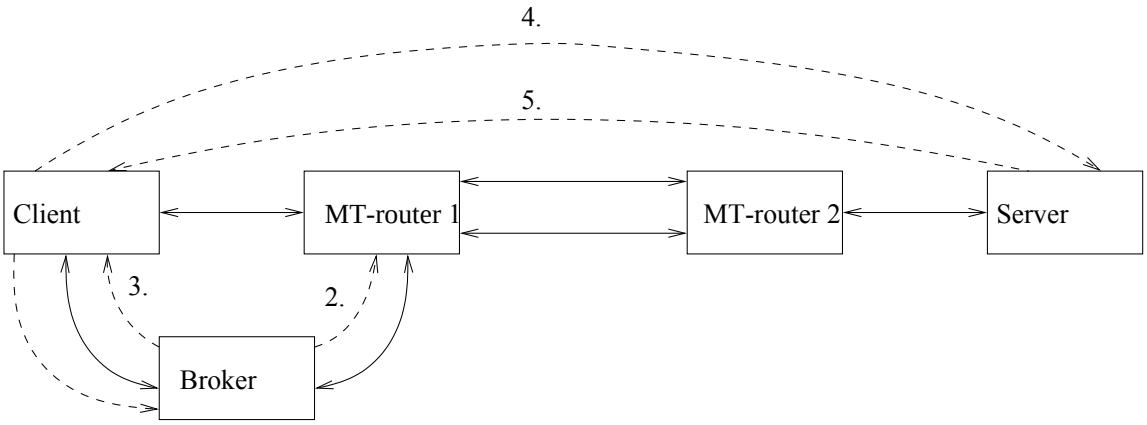


Figure D.2 This figure shows the network configuration for the test.

In order to make correct decisions, the broker needs to have information about both the actual bit-rate available to it in the network, and also know how much of those resources are in use. The broker does this by keeping track of the request it has already granted, including information about the size of the request and the client's demand for timeliness. For more information about how this information is gathered, see [32].

D.3.3 Emulated networks

The focus in this experiment was on the efficiency and fairness of different mechanisms in the broker for Web services, in a situation where the available bit-rate between the client and the server varied between two different capacities; a high bit-rate scenario (300 kb/s) and a low bit-rate scenario (16 kb/s). Thus to simplify the testbed we represented the two paths between client and server in Figure D.1 with two links as shown in Figure D.2. We used Linux traffic control(tc) in the router to shape the traffic marked with the BULK QoS-class to 300 kb/s for the one link and 16 kb/s for the other link. We did not find it necessary to emulate any other radio characteristics, (e.g., delay or packet loss) on the links, since bit-rate was the focus of our work.

The MT-router runs Open Shortest Path First - Multi Topology (OSPF-MT)[6] [52] to build the overlay topologies. OSPF uses by default a 10s interval between Hello packets. In this experiment we considered a mobile tactical scenario and thus wanted quick routing responses to topology changes, thus the Hello interval was set to 1s and the router dead interval was set to 3s.

It was assumed that the MT-router's treatment of NETR QoS-tag (Table D.2) for routing traffic worked satisfactory, thus in the experiment the routing traffic was ideal (lossless).

The broker learns which path is available to the server at any time by polling the routing table of the high bit-rate topology and the low bit-rate topology. The IP address (or network address) of the server must be present in the routing table for the router to have a path to the server on the specified topology. The broker polled the routing tables every 0.05s.

[6]The MT routing algorithm was implemented by Thales Norway AS.

Five different network scenarios were defined, in order to test the broker's ability to adapt to network changes. In the remainder of this paper, paths from the high bit-rate topology are referred to as the primary connection whereas paths from the low bit-rate topology are referred to as the backup connection. The following network scenarios were tested:

- «PrimaryOnly» – Always use the primary 300 kb/s link.
- «BackupOnly» – Always use the backup 16 kb/s link.
- «PrimaryThenBackup» – Start with the primary link active, after 30 seconds disable it and wait for the backup link to be active.
- «BackupThenPrimary» – Start with the primary link disabled, and then after 30 seconds enable it and thus provide more bit-rate.
- «VaryingNetwork» – Simulates a mobile unit going up and down on hills, where the primary link is only available near and at the top. The test starts with the primary link enabled. Then it enters a loop where it disables the primary link after 20 seconds, and then after 20 seconds more re-enables it.

The emulated network was stopped when all requests were either finished or preempted, which depends on the runtime for each test set.

D.3.4 Queuing options

Policies for queuing can vary from network to network. The implementation had five options which could either be enabled and disabled:

1. «Delayed start» – Support a delayed start. If a client is running and a new client can be run after the running client is finished, and still keep its time limit, then reserve bit-rate after the running client and inform the client that it must delay its start. If the new client has higher priority and it cannot wait until the current running is finished, it will preempt it.
2. «Always highest» – Always preempt lower priority clients. The negative aspect is that the network resources have already been spent, and that the lower priority client will probably try again later. Positive aspect, for the high priority clients at least, swifter responses.
3. «Token bucket» – Enables a token bucket mechanism. The largest reply size supporting the Token Bucket was set to 1 KB, or slightly more than the GPS Web service. Replenish would happen every 1 second, where the primary link gains 1 token at each tick, up to a maximum of 5 tokens. The backup link was limited to 1 token, and gaining 0.3 token each tick. First both links used the primary link's values, but the result was way too intrusive on the backup link. Recall that a token corresponds to a request up to 1 KB, which is half of the backup link's capacity for one second.
4. «Enforce timeslots» – If a client's ID should be revoked after the reservation has expired from the ring buffer. This might be long before the client's actual time limit has expired, and the option does not consider if any more requests are in the ring buffer. Thus potentially a client might be revoked and leave the link idle.

Number	Delayed start	Always highest	Token bucket	Enforce timeslots	Slow start adjust
1	False	False	False	False	False
2	False	False	False	False	True
3	False	False	False	True	False
4	False	False	False	True	True
5	False	False	True	False	False
6	False	False	True	False	True
7	False	False	True	True	False
8	False	False	True	True	True
9	False	True	False	False	False
10	False	True	False	False	True
11	False	True	False	True	False
12	False	True	False	True	True
13	False	True	True	False	False
14	False	True	True	False	True
15	False	True	True	True	False
16	False	True	True	True	True
17	True	False	False	False	False
18	True	False	False	False	True
19	True	False	False	True	False
20	True	False	False	True	True
21	True	False	True	False	False
22	True	False	True	False	True
23	True	False	True	True	False
24	True	False	True	True	True
25	True	True	False	False	False
26	True	True	False	False	True
27	True	True	False	True	False
28	True	True	False	True	True
29	True	True	True	False	False
30	True	True	True	False	True
31	True	True	True	True	False
32	True	True	True	True	True

Table D.3 Queuing option permutations (and their corresponding numbers on the x-axis of the evaluation figures)

5. «Slow start adjust» – Estimating transfer time is more complicated than dividing transfer size by speed per second. This is mainly due to TCP not utilizing the full capacity of the link at all times. To compensate for slow start, the requests were slightly increased by the formula `min(KBpts * 0.4, size*2)`, where *KBpts* is KB per time slot and *size* is the request's original size. The formula will mostly help when invoking Web Services with small payloads, such as a GPS, because otherwise the reservation would almost have passed before the client has invoked the Web service.

Options 1 and 2 are the most intrusive ones. For example, leaving both off will disable queuing and always allow the current client to finish. This may lead to priority inversion, as a best effort client would cause any other clients, even higher priority ones, to be rejected.

All permutations were tested, a total of $2^5 = 32$ combinations. In the evaluation these permutations are referred to both by their representation, e.g, True-True-False-True-False, and the permutation's number. The list of representations and their number can be seen in Table D.3.

D.4 Evaluation

In Figure D.3 one can see that queuing is helping when more requests with different priority are taken into consideration. Still 9, 10, 13 and 14 perform best by far, though the queuing half is catching in. This figure is the only one where all options are better than running without the broker.

Figure D.4 shows the trend that queuing is better for a larger set of priorities, and it is the first figure where all but one of the queuing option sets perform better than the non-queuing option sets.

In Figure D.5 one can see that when all four levels of priorities are considered, the queuing option sets are vastly superior to the non-queuing option sets. It could seem that not using the broker will perform better than almost all non-queuing option sets, and clearly better than 11, 12, 15, and 16. This is not true, as looking at the «no broker» column more closely one will see that the PrimaryOnly network covers a huge fraction of the successful clients, and that most non-queuing option sets perform better in all the other networks.

Also as in Figure D.3 and D.4, the time slot enforcing enabled in option sets 27, 28, 31, and 32, are clearly limiting the client satisfaction success rate.

D.5 Conclusion

Since Web services lack the necessary QoS standards, we have implemented a prototype QoS admission control broker. When using a broker dedicated for Web services, it is possible to tailor the decisions made by the admission control element to best suit the different classes of Web services requests and the requests' role priorities.

Five different queuing options have been tested under both high and low load, and for several

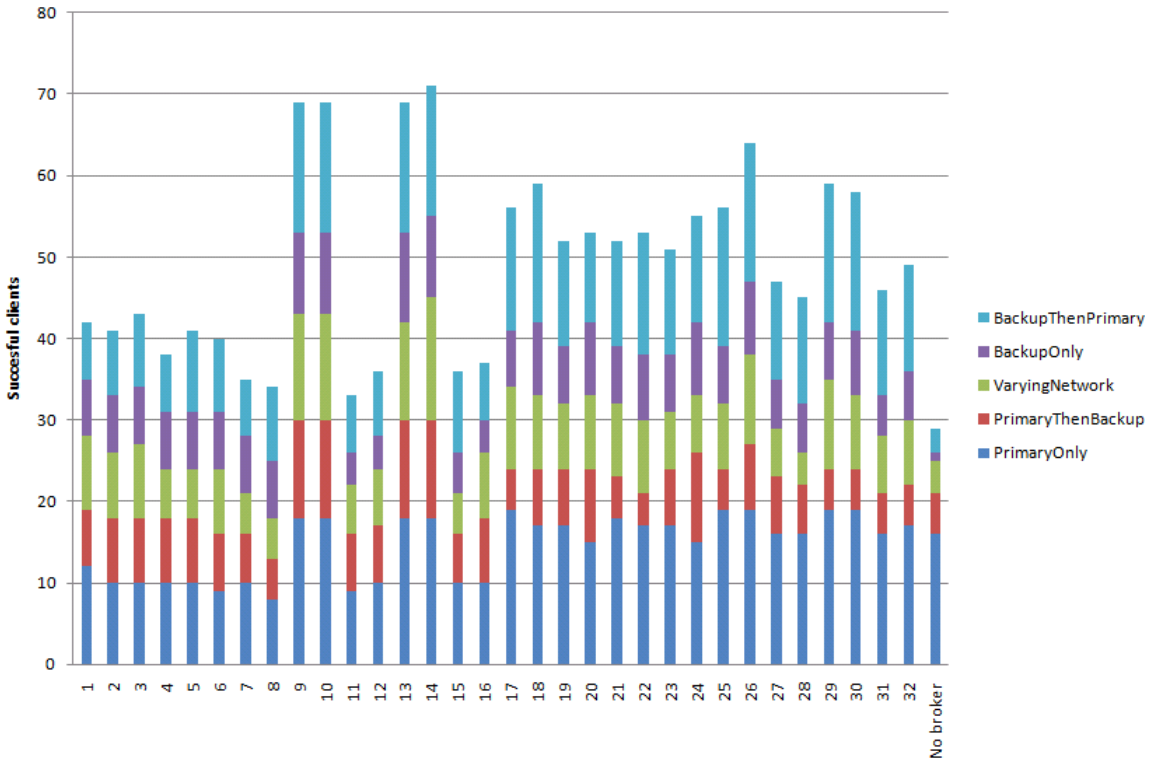


Figure D.3 Successful clients with high and medium priority. Max is 20 per emulated network, 100 in total.

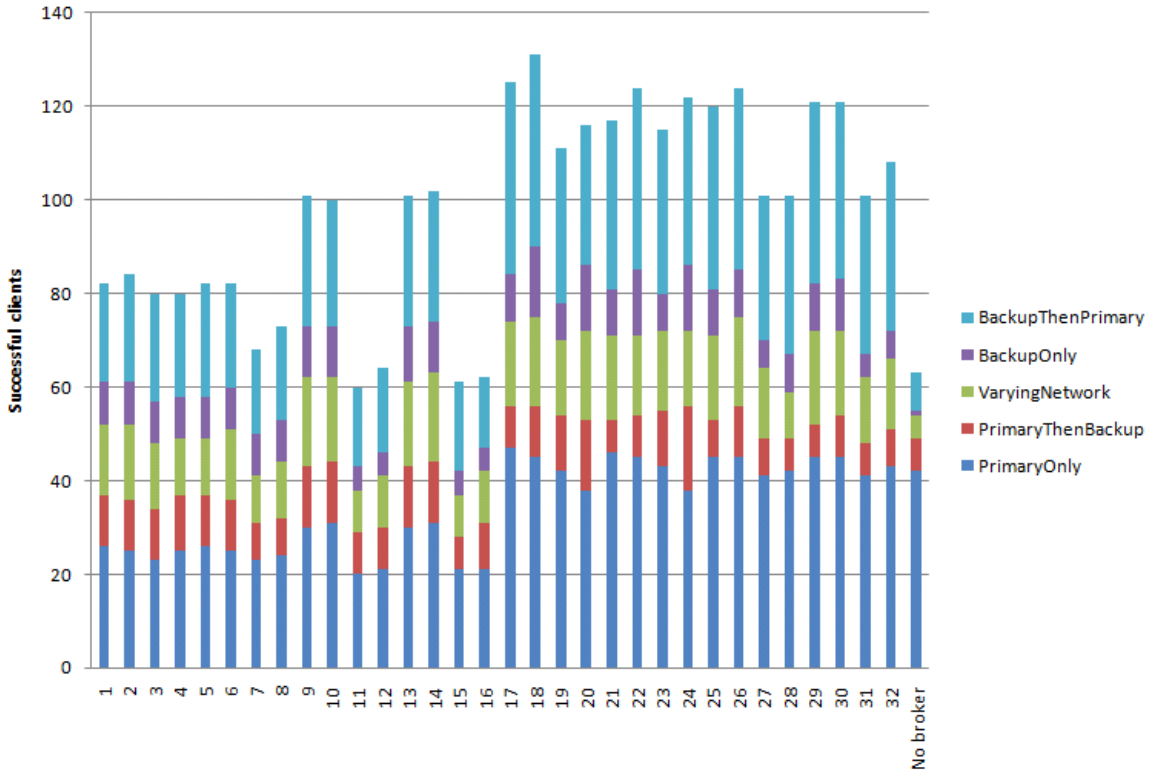


Figure D.4 Successful clients with high, medium and low priority. Max is 50 per emulated network, 250 in total.

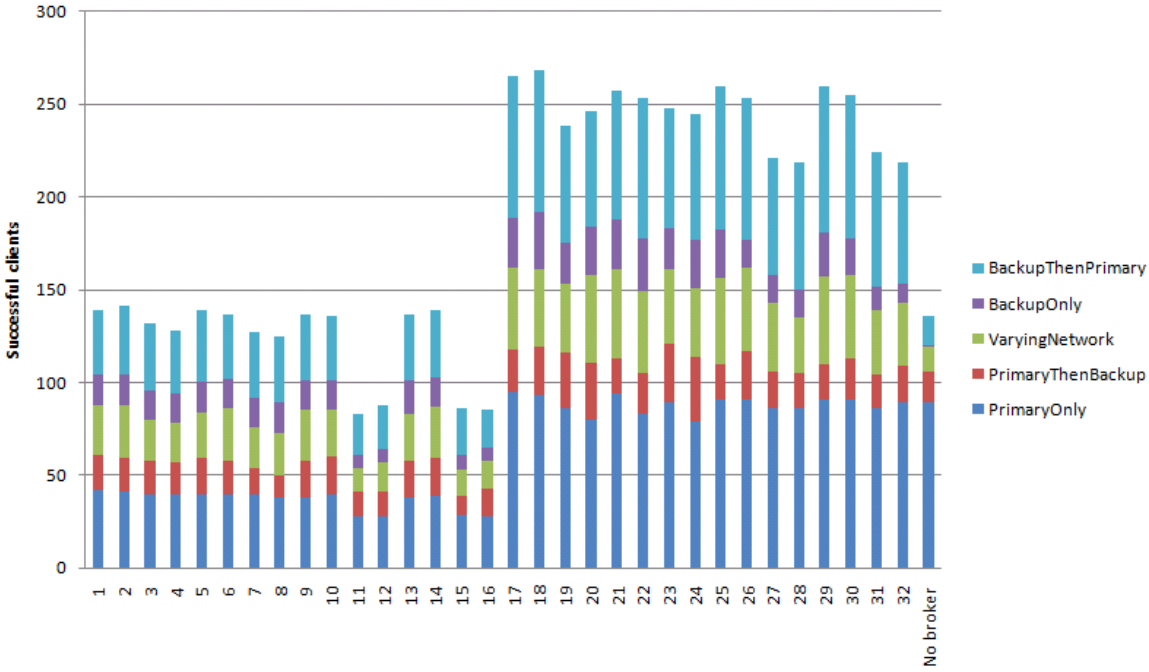


Figure D.5 Successful clients with any priority. Max is 100 per emulated network, 500 in total.

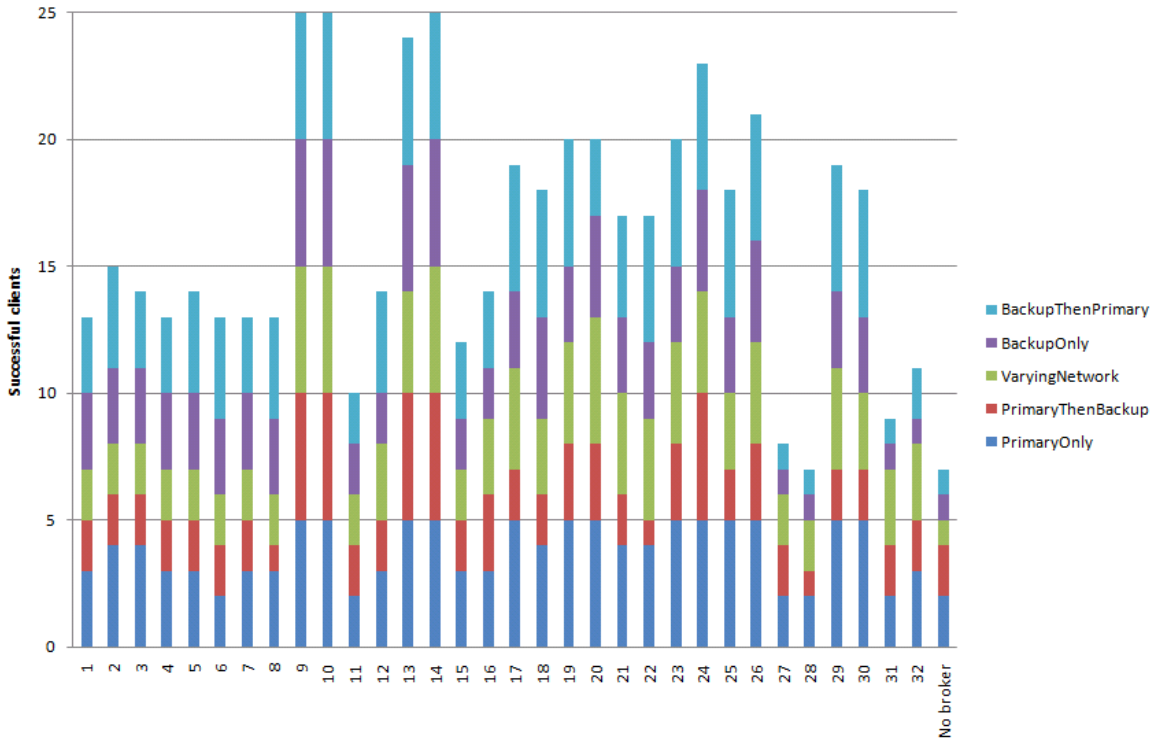


Figure D.6 Successful clients with high priority. Max is 5 per emulated network, 25 in total.

different network behaviors. For real-life deployments, it depends on policy whether the different queuing options should be used or not:

- If policy dictates that high client satisfaction for high priority clients is the most important metric, then queuing should be turned off, always preempt lower priority clients turned on, and time slot enforcing turned off. This corresponds to option sets 9, 10, 13 and 14, and as seen in Figure D.6 these options will give high satisfaction for high priority clients.
- On the other hand, if optimizing for highest overall client satisfaction, then option set 18 should be selected. Options set 17 was almost as good as 18, but as it did not have Token Bucket enabled for small requests it became the second best.

The major difference between maximizing for overall- and high-priority client satisfaction is that overall satisfaction requires queuing to be enabled, as seen in Figure D.5, where 1 to 16 do not have queuing, while 17 to 32 have queuing.

Appendix E Role-based Quality of Service for Web Services

This paper summarizes the work of a group of bachelor degree students from NTNU which was supervised by FFI. The paper was written by the supervisers (Johnsen and Bloebaum), and the team of students (Jørgen H. Nordmoen, Jan A. S. Bremnes, Stig Tore Johanessen, Magnus L. Kirø, Ola Martin T. Støvneng, and Håvard Tørresen), and published at IEEE HeterWMN 2012, Anaheim, CA, USA, 3 December 2012.

Abstract

We have designed and implemented a prototype system providing role based Quality of Service (QoS) for Web services in heterogeneous networks. We leverage industry standards to the fullest extent, in an attempt to bring role based QoS support to standard Web services. We have extended an existing enterprise service bus to accommodate the changes necessary for prioritization on the server side, and created a custom client library to ensure prioritization in both the request and the response of the Web services message exchange. Finally, roles are defined using Security Markup Assertion Language (SAML) tokens. Our framework has been released as open source.

Our evaluation shows that the concept is viable, and that prioritization on the application level of the OSI model, combined with network level prioritization as provided by DiffServ, is beneficial in networks with low bandwidth.

The Service-Oriented Architecture (SOA) principle is becoming a common approach to use when designing and building large distributed software systems. SOA allows for loose coupling of systems, and is thus ideally suited for building large-scale systems-of-systems. The most mature and most common technology used for implementing SOAs is Web services, a middleware technology which is based on standards, and which is capable of implementing a system based on SOA principles.

So far Web services have mostly been used on the Internet and within business networks, and the most commonly used protocol bindings are best suited for these types of networks. However, because of the flexibility offered by Web services, it is beginning to be more used also in wireless networks, ranging from IEEE 802.11 networks to specialized military radio communication networks.

This shift from mainly high capacity wired networks to more widespread use has led to an increased focus on the network resource consumption of Web services. Specific compression mechanisms have been developed for the eXtensible Markup Language (XML) format used by Web services [54], and other protocol bindings have been developed as an alternative to the standard HTTP-over-TCP binding used in most systems.

Due to the interoperability benefits offered by Web services, such as loose coupling and a strong standards foundation, NATO has decided to focus on this technology as a basis for interoperability between coalition members. Additional benefits, such as the ability to reuse services to build more

complex services, and the flexibility of Web service enabling already existing systems, has led to further adoption of this technology among partner nations.

Quality of Service (QoS) is an important aspect of any network, but it is of particular importance for systems that need to be able to function both in infrastructure networks and in radio based communication networks. On the Internet and in corporate networks the desired QoS can often be achieved by overprovisioning of resources in the networks. In wireless networks this is not possible to the same extent, because the maximum available bandwidth in such networks is a function of radio capabilities, distance between the communicating nodes, interference, node mobility, the number of network hops, and so on. This means that there is a need for QoS frameworks supporting such networks. In this paper, we address the issue of role based QoS support for using Web services technology across heterogeneous, multi-hop wireless networks.

The remainder of this paper is organized as follows: In Section E.1 we present related work. Section E.2 discusses the design and implementation of our prototype software, whereas Section E.4 covers the evaluation. Section E.5 concludes the paper.

E.1 Related work

Previously, we have implemented a QoS based admission control mechanism, which provides priority based access to the network, while at the same time avoiding overloading the limited network capacity that is available [28]. In this paper we expand on those ideas by building a role based QoS framework founded on industry standards for Web services clients and services.

Hauge et al. have shown how *Multi-Topology routing* [52] can be leveraged in heterogeneous tactical mobile ad hoc networks to improve the network resource utilization [23]. They suggest using a QoS model with a routing protocol which maintains several distinctive network topologies, where each topology is tailored to support either a single or multiple QoS classes. We assume the presence of such Multi-Topology routing for the work we present in this paper.

The Differentiated Services (DiffServ) mechanism can be used to ensure that some traffic is prioritized, a task that it performs quite well through actual use and simulations, see experiment details in [34], [60], [3], and [35]. DiffServ relies on tagging the type-of-service (TOS) field in the IP header with a bit pattern corresponding to a certain traffic class. This allows DiffServ-enabled routers to prioritize the IP packets with a deterministic per-hop-behavior (PHB). DiffServ is covered by several RFCs:

- Definition of the Differentiated Services Field in the IPv4 and IPv6 Headers [44],
- An Architecture for Differentiated Services [5],
- Assured Forwarding PHB Group [24],
- Per Hop Behavior Identification Codes [4],
- An Expedited Forwarding PHB [11],
- New Terminology and Clarifications for DiffServ [19], and

- Configuration Guidelines for DiffServ Service Classes [1].

In this paper, we rely on DiffServ for enforcing network level QoS.

Yu et al. [69] have investigated issues regarding service selection with multiple QoS constraints and proposed several algorithms. Their work is orthogonal to ours, in that their approach can be leveraged when you have multiple available services. In this paper our services reside in an enterprise service bus (ESB), thus yielding a single service source. In future versions of our prototype we may add support for these algorithms as well for even further gains when there are multiple copies of the available services in the network.

E.2 Design and implementation

Essential to Network Based Defense (NBD) is the concept of end-to-end QoS, which in turn requires employing cross-layer QoS signaling. This means that QoS must be considered at all layers of the OSI model, and that QoS information must traverse these layers. To achieve the end-to-end QoS needed in NBD, QoS metadata must also be allowed to cross both network and national boundaries. There are QoS mechanisms that can be used on the transport layer and below, and thus we focus our research efforts on the application layer and issues regarding cross-layer QoS signaling. Having IP as a common protocol and assuming DiffServ as the network level QoS framework, we focus on the application level solutions in this paper (i.e., the Web services middleware). Our goal is to provide prioritized access to Web services based on the client's role, and enforce this at the network level by mapping the demands to the TOS field in the IP header, enabling cross-layer QoS signaling. DiffServ provides coarse traffic shaping, so it is desirable to have finer grained control on the application level by taking user needs (represented by a role) and available resources (the current network resource state) into account.

In this section we discuss the design and implementation of the prototype system. First we present the different components used in our solution, then we cover the details regarding the server and client side implementation.

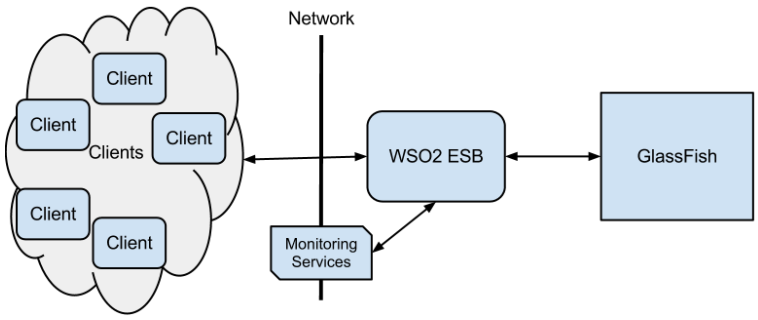


Figure E.1 Prototype overview

E.3 Components

We use the following components as part of our prototype design:

- Multi-Topology router with exposed monitoring service
- WSO2 ESB
- GlassFish application server
- Security Assertion Markup Language (SAML)

The monitoring service exposes the active routing table from the router, thus allowing our software to glean information about the current maximum available bandwidth. The router implements and performs Multi-Topology routing [23]. We employ the router as described in our previous work [28], in this paper we expand on the QoS support given by the Web services layer as described below.

The WSO2 Enterprise Service Bus (ESB) supports industry standards and is an open source platform for deploying Web services. It is in widespread use today, being used by several large companies (e-bay being a prominent example) [68].

We employ the GlassFish Server Open Source Edition for hosting Web services, since it is a robust and free community-supported application server featuring full Java EE 6 platform support [50].

SAML is an XML-based framework for request/response exchanges of authentication and authorization information [46]. SAML assertions describe the results of authentication actions that occurred previously. In our framework, SAML tokens are used to identify the client's role.

These components are used in our prototype, as Fig: E.1 illustrates. There, a cloud of clients (which can be in the same local network or in different networks) access Web services hosted through the WSO2 ESB. The pair <*role, service*> is used by the client library as well as the QoS functionality we implemented in the ESB to determine which QoS class a request and response is handled according to. The ESB mediates the client request messages into GlassFish where the Web services are deployed, and the requests are processed. In order for the ESB to know how much bandwidth each client can utilize, it is dependent on the monitoring service which relays that sort of network information. The monitoring service resides in a Multi-Topology router functioning as a gateway to another network. The Web service response is mediated through the ESB and sent back to the client. The information flows on the server and client sides are discussed in detail below.

E.3.1 Server side architecture

The server side architecture consists of the WSO2 ESB and the GlassFish application server. Furthermore, the server side needs access to one or more monitoring services as provided by a Multi-Topology router. All of these components were already available (see [68, 50, 28]), so what we needed to make were custom *mediators* in the ESB. A mediator is a component in WSO2 ESB

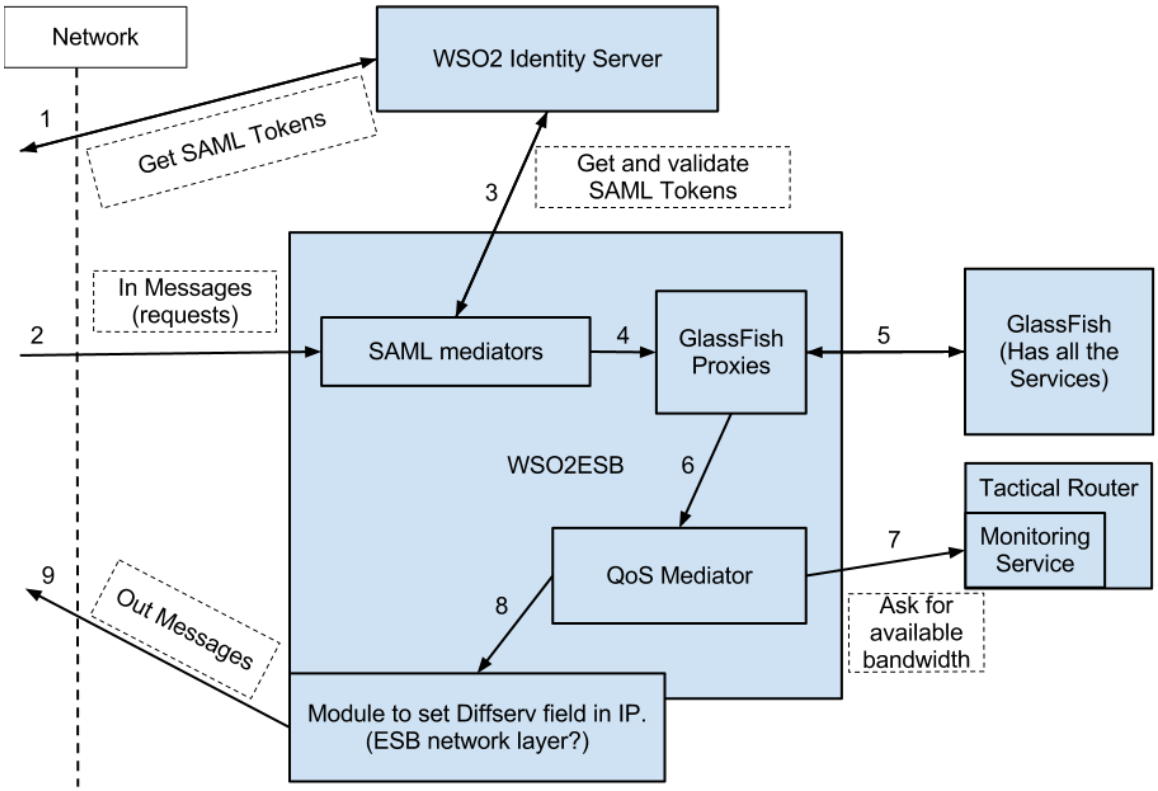


Figure E.2 The Server side Architecture

which can be used to work on incoming or outgoing messages that pass through the ESB.

Before the client can access a Web service it has to have a SAML token for identification. To get an ID-token it has to contact the Identity Server using the ESB as a *proxy* (i.e., an intermediary between clients and servers) (Fig: E.2-1). Then the client can access a Web service from the ESB. Several things then happen in the ESB: First the request message is sent to the SAML mediator (Fig: E.2-2), this mediator contacts the Identity Server to validate the client's ID-token (Fig: E.2-3). If the token is validated, then the client is granted access to the requested service, and the message is passed on to the GlassFish proxies (Fig: E.2-4), otherwise the message is dropped. The ESB then sends the request along to the corresponding service on the GlassFish server (Fig: E.2-5).

When the request is received by GlassFish, it processes the request and issues a reply message. This reply message is also passed through the ESB on its way back to the client. First the message is sent to the QoS mediator (Fig: E.2-6). This mediator will first look at the role of the client as well as the service requested, and use this information to assign a priority to the reply. If enabled, it will also perform throttling using our Throttle mediator: The Throttle mediator is used to ensure that high priority messages are sent first. To determine what to disrupt and what to hold back, and for how long, several properties are used; the priority of the message, the available bandwidth, the IP address of the Multi-Topology router, and the real time demand of the request. In order to do this, the mediator must keep a list of sending messages and where those messages are going.

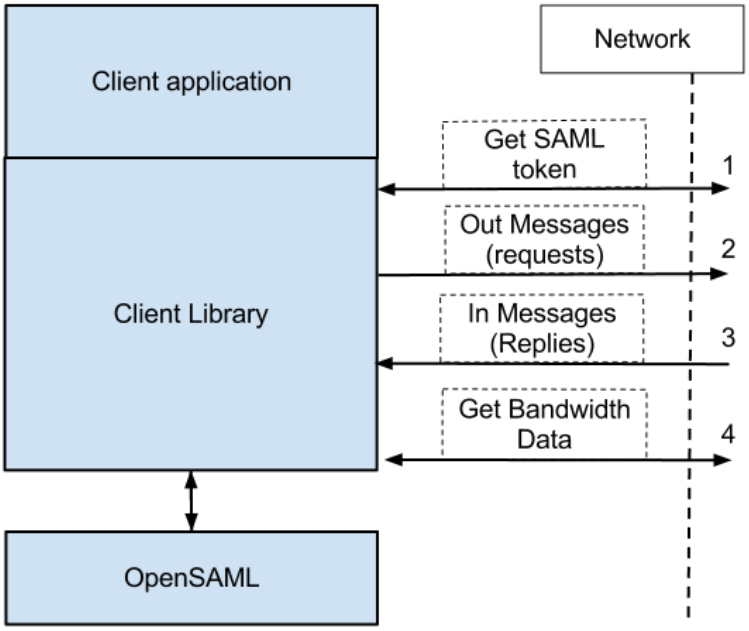


Figure E.3 The Client side Architecture

Then the monitoring service is contacted for bandwidth information (Fig: E.2-7), which is used together with the priority to determine whether the message should be sent right away or held back until some higher priority message is finished sending. Finally, the ToS field in the IP header is set (Fig: E.2-8) before the message is sent to the client (Fig: E.2-9). This field is used by the routers in the network to prioritize packet sending.

E.3.2 Client side architecture

The client side architecture is composed of our role based QoS-enabling client library, utilizing OpenSAML [49] for SAML support. Our library can be used by existing client applications, one only needs to replace the existing Web service call with a call to our library to gain QoS support.

Before the client library can ask for the data the client needs to get a SAML authentication token (Fig: E.3-1). The client library then sends the request from the client to the server (Fig: E.3-2), appending the SAML token to the message as well as adding some metadata in the SOAP header related to the client role and setting the TOS field of the IP packets to the corresponding value.

The reply from the server is examined by our client library for the metadata the server has added to the SOAP header. Relevant metadata is stored for future communication and the message is passed to the client application (Fig: E.3-3).

When new communication is initiated after this first connection is made the client should, if everything went as expected, have the necessary information to prioritize new messages. This means that the client can now make an informed decision about how it should prioritize the messages it sends. In order to do this it can also take into consideration the available bandwidth

(Fig: E.3-4) as provided by the monitoring service. This information may then be leveraged for admission control as we described in [28].

Client	Role	Interval	Number of requests	Delay
2 and 3	Low priority	1000	100	10
4	High priority	3000	30	15

Table E.1 Client parameters

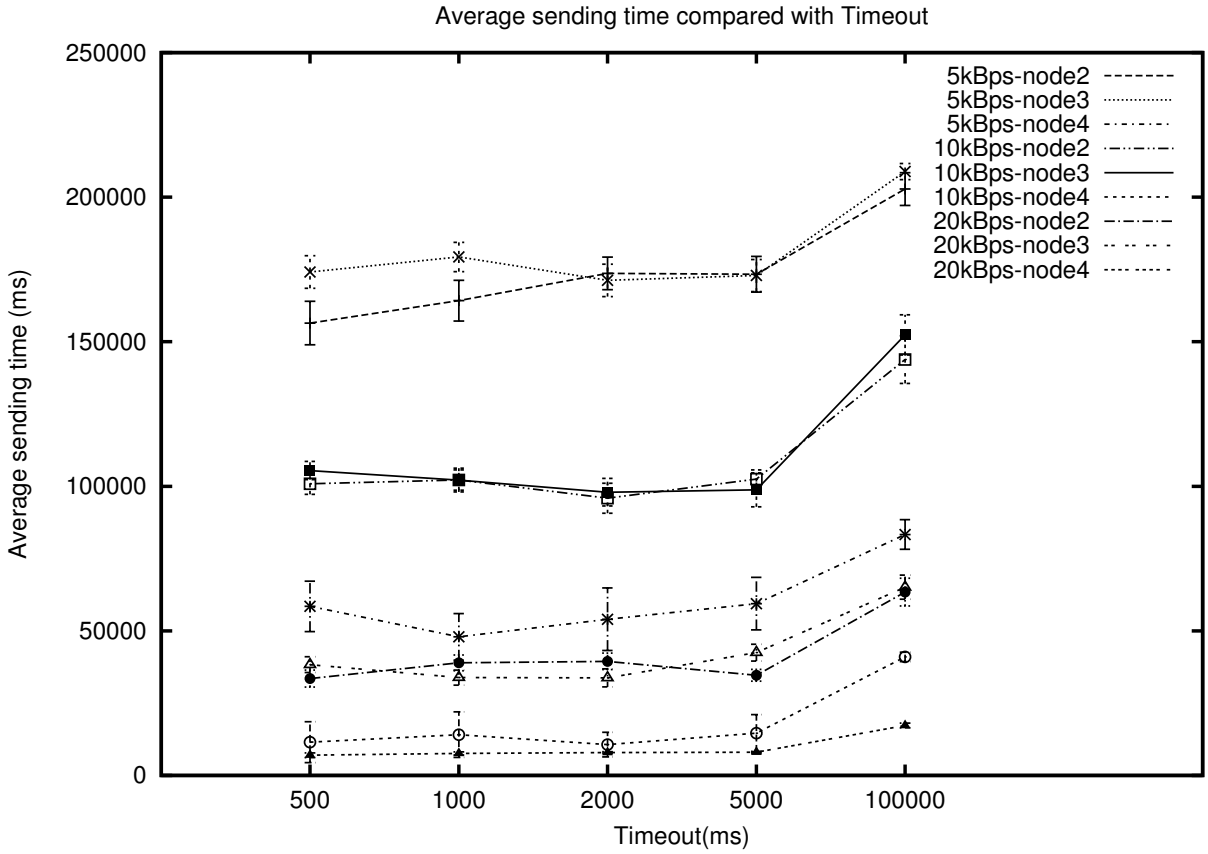


Figure E.4

E.4 Evaluation

In this section we first present the test framework employed, before we discuss the test results.

E.4.1 Test framework

Our tests were performed in *mobiemu* — a freely available open source framework for emulating mobile ad-hoc networks with Linux containers (LXC) and ns-3 [56]. The mobiemu framework operates by creating LXC and connecting these to so-called *tap bridges* created by ns-3. This then emulates any network possible to create in ns-3. From the point of view of programs running

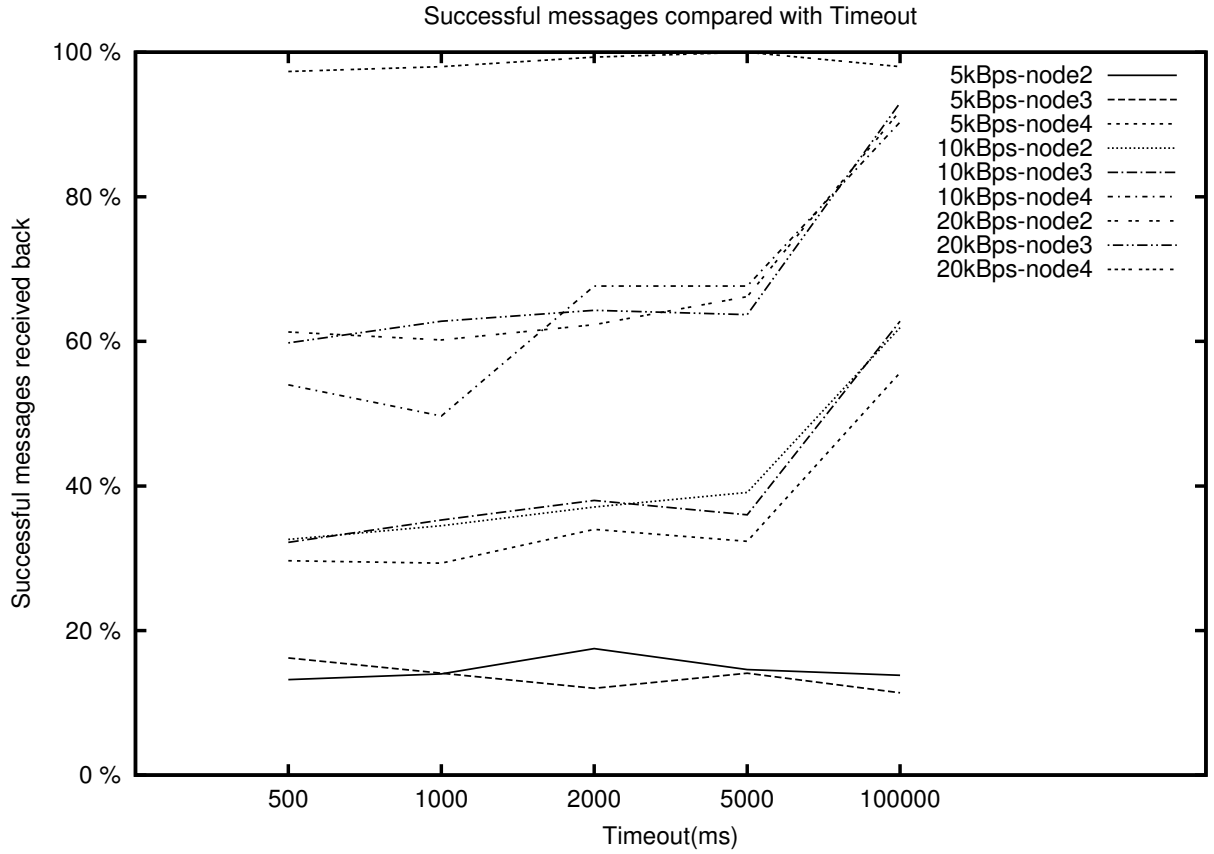


Figure E.5

inside the LXC, they are full Linux machines connected to a real network. This means that any program able to run on Linux should run properly inside the LXC and any messages they send out are sent through ns-3. This means that we have full control over how our network behaves, allowing us to emulate different scenarios.

When mobiemu starts up it creates a number of LXC, it then starts up ns-3 and connects all LXC to a corresponding tap bridge. Inside each of the LXC it then starts the experiment script and waits for the experiment to finish before cleaning up. Before each run mobiemu stores all files and folders in one experiment folder in order to enable easy experiment repetition. When the experiment is done the result files are moved into the result folder.

Our tests were set up as follows: We have three clients, two clients with low priority and one client with high priority. They communicate with the ESB as described above in Sections E.3.1 and E.3.2.

The tests were simple and contained four nodes in mobiemu, functioning as a proof-of-concept of our prototype: Node 1 — ESB, GlassFish, and monitoring service. Nodes 2-4 — Clients: 2-3 have low priority roles, 4 has a high priority role. The clients were configured as shown in Table E.1. All nodes issue the same size request and receive the same size payloads. The request is a SOAP

message with one parameter, whereas the reply is a SOAP message with 10 KB payload.

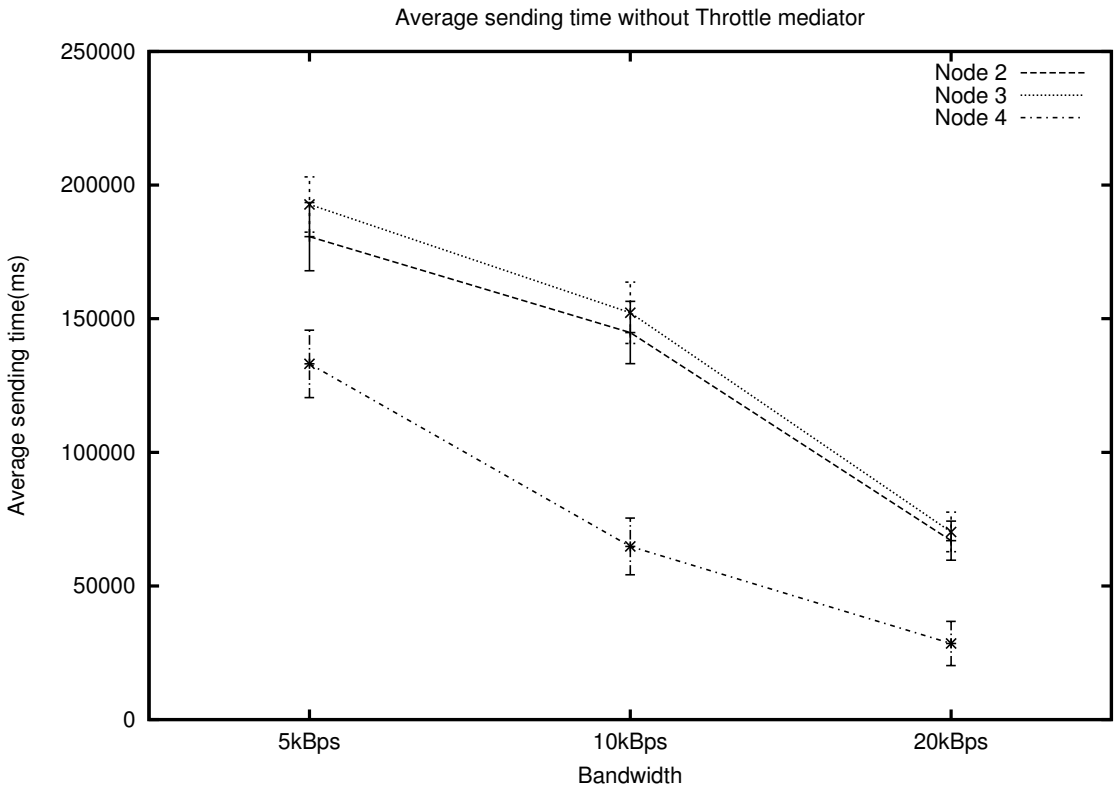


Figure E.6

E.4.2 Results

In Figure E.4 we have compared the average sending time with the timeout used in our Throttle mediator and added all the different bandwidths into one graph. From the graph we can see that increasing the timeout dramatically also has an effect on the time needed to send a message. We can also see that a higher bandwidth lowers the sending time, but the high priority node, Node 4 in the graph, generally does quite well and has a substantially lower sending time compared to the other nodes. If we compare the high priority node at a speed of 5 kBps we can see that the node fares better than the low priority node even though they have twice the bandwidth.

In Figure E.5 this next graph we have compared the percentage of successful messages received back with the timeout used in the Throttle mediator. The first thing we can see is that when it comes to messages a higher timeout is better. The reason behind this is simple as the timeout is the time before the Throttle mediator preempts messages because they have been sending for too long. We can also see that with enough bandwidth in the network the timeout has little effect as the message is sent before we need to preempt. As with the time graph we can see that the higher priority node does considerably better than the lower priority ones.

In Figure E.6 we have compared the average sending time to the bandwidth in the network without

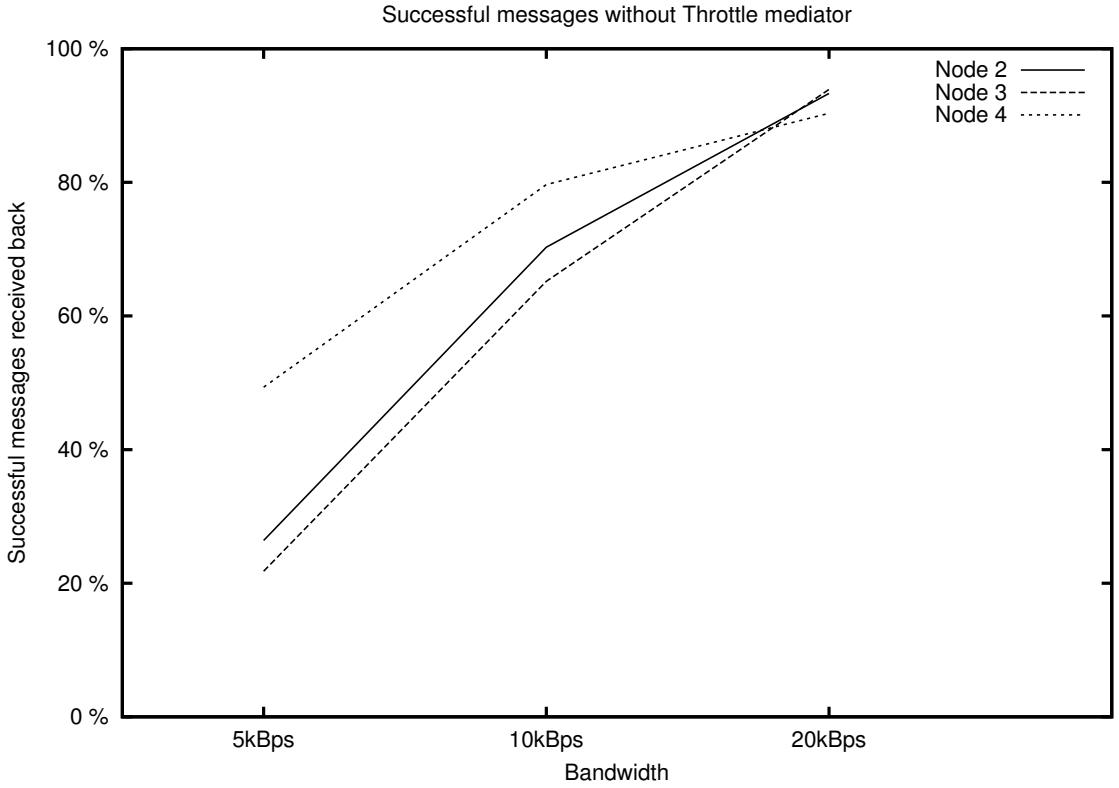


Figure E.7

the Throttle mediator. As one would expect the higher the bandwidth the lower the sending time is. Note that Node 4 has a substantially lower average sending time than the two other nodes. The reason behind this is twofold: First, since Node 4 sends fewer messages with more time between them it has a smaller sample than the two other nodes. Second, the ESB is still prioritizing messages, even though there is no Throttle mediator present to make sure that there is enough bandwidth available for the higher priority messages.

In Figure E.7 we have compared the successful messages with the bandwidth in the network. If we compare this graph to the graph above which had the Throttle mediator we can see that the lower priority nodes perform better, but at the cost of the higher priority node.

E.5 Conclusion and future work

We have designed and implemented a prototype system providing role based QoS for Web services in heterogeneous networks. We have extended an existing ESB to accommodate the changes necessary for prioritization on the server side, and created a custom client library to ensure prioritization in both the request and the response of the Web services message exchange.

Our evaluation shows that the concept is viable, and that prioritization on the application level of the OSI model, combined with network level prioritization as provided by DiffServ, is beneficial in networks with low bandwidth. Our prototype software was developed using the Java EE 6

platform, and is available as open source at `https://github.com/magnuskiro/it2901`.

Future work involves addressing aspects regarding accessing multiple simultaneous service deployments across heterogeneous networks, along with further developing the monitoring service. The current service only yields information about maximum bandwidth, but others providing additional metrics may enable even more fine-grained QoS control. Finally, we plan on employing our prototype in a real-life experiment in the future, where heterogeneous communications equipment will take part.