16th ICCRTS

“Collective C2 in Multinational Civil-Military Operations”

Title of Paper

Adapting WS-Discovery for use in tactical networks

Topic(s)

Primary: Topic 9: Networks and Networking
Alternatives: Topic 8: Architectures, Technologies, and Tools, Topic 6: Experimentation, Metrics, and Analysis

Name of Author(s)

Frank T. Johnsen and Trude Hafsøe
Norwegian Defence Research Establishment (FFI)
P.O. Box 25
2027 Kjeller, Norway

Point of Contact

Frank T. Johnsen

Norwegian Defence Research Establishment (FFI)
P.O. Box 25

2027 Kjeller, Norway

frank-trethan.johnsen@ffi.no

Abstract

The NATO Network Enabled Capabilities (NNEC) feasibility study has identified Web services as a key enabling technology for NNEC. The technology is founded on a number of civil standards, ensuring interoperability across different operating systems and programming languages. This also makes the technology a natural choice for interoperability also in multinational civil-military operations, where a large number of heterogeneous systems need to exchange information. Web services provide loose coupling and late binding, which are desirable properties in such a setting. Discovering available services in an operation is essential, and the discovery process must leverage standards to ensure interoperable information exchange. WS-Discovery is a standard for Web services discovery suited for dynamic environments and civil networks, but has high overhead and is not so suitable for tactical networks. Like the other Web services standards, it uses XML for encoding messages. In civil networks bandwidth is abundant, but in tactical networks XML may incur unacceptable overhead. However, the W3C has created a specification for efficient XML interchange (EXI), which reduces XML overhead by defining a binary interchange format. This paper investigates the performance gains (in terms of reduced bandwidth) that can be achieved by combining the WS-Discovery standard with the EXI specification.

1 Introduction

The NATO Network Enabled Capabilities (NNEC) feasibility study [1] has identified Web services as a key enabling technology for NNEC. This is due to the NNEC demands for interoperability, and the fact that Web services are founded on standards. This means that the technology, which is in widespread use in civil networks today, is emerging in military systems in NATO countries and will therefore also be an important part of civil-military interoperability in the future.

Alberts and Hayes [2] describe a civil-military operation as a complex system which involves a large number of disparate entities such as military units, civil authorities, multinational and international organizations, non-governmental organizations, companies, and private volunteer organizations in dynamic environments operating with imperfect and incomplete information.

Due to the dynamic nature of such an operation, mobility and abrupt disappearance of services must be supported. This means that service discovery must be distributed and robust. The participants may use heterogeneous devices, requiring standardized service oriented middleware to achieve interoperability.

Discovery of Web services can be performed either by utilizing a service registry, or a decentralized non-registry based solution. There are three standards addressing Web services discovery, two registries and a non-registry solution. The registries, UDDI [3] and ebXML [4], suffer from liveness and availability problems in dynamic environments [5]. The third standard for Web services discovery is WS-Discovery [6]. It is better suited to dynamic networks than the registries in that it is a decentralized discovery mechanism, thus removing the single point of failure that a centralized registry constitutes.

We attempted to use WS-Discovery in a disadvantaged grid, and found that it was unsuitable for use there since it generated too much traffic in the network and flooded the modem buffers (see our paper [7]). If we can reduce the overhead of WS-Discovery, however, then it may be better suited for use in military networks as well. Recent work by the W3C regarding efficient XML interchange (EXI) can potentially make WS-Discovery suitable for both civil and military networks, and thus interesting as a solution for use in multinational civil-military operations. The contribution of this paper is an evaluation of the performance gains (in terms of reduced bandwidth) that can be achieved by combining the WS-Discovery standard with the EXI specification. For evaluation purposes we combine an open source implementation of WS-Discovery with an open source implementation of EXI.

The remainder of this paper is organized as follows: Section 2 discusses related work; standardized protocols and proprietary solutions. Section 3 introduces the initiative regarding XML compression by W3C's EXI Working Group. In section 4 we describe our proof-of-concept implementation: combining an open source WS-Discovery implementation with an open source EXI implementation, and the results of our evaluation. Section 5 concludes the paper.

2 Related work

There exist several standards and proprietary, experimental solutions that may be employed for Web services discovery. In this section we give an overview of the most prominent solutions that have emerged.

2.1 Web services discovery standards

There are three standards related to service discovery, all by OASIS: Universal Description, Discovery and Integration (UDDI), electronic business using XML (ebXML), and WS-Dynamic Discovery (WS-Discovery).

2.1 UDDI

UDDI [3] allows service providers to register their services and service consumers to discover these services both at design-time and run-time. In principle, UDDI is centralized, but mechanisms for federating several registries have also been specified. Having multiple registries, or letting a registry consist of several nodes that replicate data, increases the robustness of the discovery solution. In UDDI, replication between registry nodes must be configured manually. It is also possible to let several separate UDDI registries exist independently of each other, but information will not be replicated unless a custom scheme is designed. Additionally, a hierarchical model may be used, using a root registry and affiliate registries. The UDDI registry supports reconfiguration as long as services do not go down unexpectedly. If so, advertisements will be in the registry forever because there is no liveness information in the current versions of UDDI.

2.2 ebXML

Another service discovery standard is ebXML [4]. It is a collection of specifications for conducting business-to-business integration over the Web. It allows registering services in a similar way as UDDI according to its own registry specification. The ebXML registry also defines inter-registry interaction, or cooperation between registries, a so-called federation of registries. Note that an ebXML federation is different from that of a UDDI federation, because ebXML supports a non-hierarchical multiregistry topology. Here, each registry has the same role, and registries may join or leave a federation at any time. This allows flexible deployment. Federated queries are supported, enabling query forwarding to other registries without the need to replicate data first. The ebXML registry is meant to support both discovery and business collaboration, as opposed to UDDI, which mainly targets discovery. Just like UDDI, ebXML has issues with liveness, in that it supports reconfiguration as long as services do not go down unexpectedly.

2.3 WS-Discovery

WS-Discovery is the newest standardized Web services discovery mechanism. After being a draft since 2005, it became a standard in 2009 [6]. WS-Discovery is based on local-scoped multicast, using SOAP-over-UDP [16] as the advertisement transport protocol. Query messages are called probe messages. Services in the network evaluate probes, and respond if they can match them. To ease the burden on the network, WS-Discovery specifies a discovery proxy (DP) that can be used instead of multicast. This means that WS-Discovery can run in two modes, depending on whether there is a DP available or not. However, this DP is not well-defined in the standard. The standard fully describes the decentralized operation of WS-Discovery, but the functionality of (and

integration with) the DP is left to be implemented in a proprietary manner for now. We evaluate only the standardized parts of WS-Discovery in this paper, focusing on decentralized operation.

2.2 Other solutions

There exist several service discovery protocols (see [8] for a survey), but few that are tailored for discovery of Web services. Traditional discovery protocols such as SLP, Bluetooth SDP, and UPnP are geared towards device discovery [9], and do not possess the expressive power that is needed to support Web services discovery. To address the need for a Web services discovery protocol for dynamic environments, OASIS has created the WS-Discovery standard.

WS-Discovery is based on querying the network for services. Using multicast SOAP-over-UDP, so-called probe messages flood the network, and nodes that can match the query, respond with a unicast probe match message. In a broadcast environment, such behavior is not desirable, since it generates a lot of traffic. Even if your neighbor has just searched for services, you will have to repeat the search because you have not received the reply. This is a drawback that has been addressed by experimental service discovery solutions.

Service Advertisements in MANETs (SAM) is a service discovery protocol that we have designed and implemented for use in MANETs [10]. It addresses the high resource use of WS-Discovery, and uses periodic service advertisements. The advertisements are compressed to reduce overhead, and caching with timeouts is used to address the liveness issue. The protocol offers a fairly up to date view of available services (in the local cache), and may significantly reduce discovery communication overhead because there is no need to query the network. Instead, service advertisements are sent at fixed intervals using IP multicast to reach all nodes. This spreads out the service discovery traffic over time, and eliminates traffic bursts due to frequent searches for services.

As multicast is not always available in MANETs, we have designed a delay tolerant, publish/subscribe mechanism that we call Mist. Using this protocol, we have implemented a Web services discovery solution, Mist-SD [11], which is suitable for use in large MANETs. Like SAM it is based on periodic updates, but instead of completely distributing all information to all nodes, it uses a combined broadcast and subscription scheme which ensures that only relevant information is propagated through the network. Mist-SD is capable of distributing WSDLs, as well as other types of application defined metadata.

Sailhan et al. [12] have created an experimental Web services discovery mechanism, Ariadne, that could potentially be usable in both large and small MANETs. It is based on having some nodes that function as registries, and allowing these nodes to form an overlay for service discovery.

Konark [13] is a fully decentralized service discovery mechanism for multi-hop MANETs. It is a complete middleware for service description, discovery and invocation. The framework is loosely based on Web services, in that XML is used for descriptions, and SOAP over HTTP is used for service invocation. However, the service descriptions are a proprietary twist on WSDL's, meaning that COTS Web services cannot be used with this solution. Konark uses periodic advertisements with a TTL to handle liveness. Konark multicasts a subset of its service knowledge to reduce bandwidth. This means that you may not be able to access all services in your network partition, since service advertisements are made in a random fashion.

The Service-Oriented P2P Architecture (SP2A) [14] supports service deployment with Web services. SP2A removes the loose coupling of Web services, demanding that the services are published, searched for, and invoked through the Peer-to-Peer (P2P) overlay.

Another experimental P2P based Web services framework which also integrates discovery and service invocation is WSPeer [15]. WSPeer is a component based solution that provides an API for hosting and invoking Web services. However, being a complete framework it too, like SP2A,

removes the loose coupling of Web services, tightly coupling service implementation, discovery, and invocation to their proprietary API.

3 Compression

Efficient XML (EFX) was one of the formats the W3C XML Binary Characterization Working Group [17] investigated during their work with requirements for a binary XML format. It was later adopted by the W3C Efficient XML Interchange Working Group (EXI) as the basis for the specification of the efficient XML format. The objective of the EXI Working Group is to develop a specification for an encoding format that allows efficient interchange of the XML Information Set, and to illustrate effective processor implementations of that encoding format. EFX was originally developed by Agile Delta and provides a very compact representation of XML information [18,19]. There also exists an open source Java implementation of the EXI specification called "EXIficient". It is available from "http://exificient.sourceforge.net/". In this paper we use the open source implementation of EXI, release 0.5.

We can apply EXI to SOAP-over-UDP in WS-Discovery, and still remain compliant to the standard, because encoding the entire SOAP message, headers and all, is allowed according to the SOAP Messaging Framework standard [20], section 4.2: “5. SOAP Message Construct provides that all SOAP envelopes are serializable using an XML 1.0 serialization, so XML 1.0 or later versions of XML MAY be used by bindings as the "on the wire" representation of the XML Infoset. However, the binding framework does not require that every binding use an XML serialization for transmission; compressed, encrypted, fragmented representations and so on can be used if appropriate.”

4 Implementation and evaluation

An open source implementation of WS-Discovery written in Java is available from “http://code.google.com/p/java-ws-discovery/”. The current release 0.2.0 is only draft compliant, but the version in the repository adheres to the standard. We downloaded the most recent WS-Discovery revision from the open source repository (which was revision 116 at the time we performed our experiments). The evaluation was performed in two iterations: First, we evaluated the WS-Discovery standard on its own. Second, we evaluated WS-Discovery with added EXI compression. For the evaluation of the standard we compiled the sources and used the software unmodified. To evaluate the standard with EXI compression, however, we had to make some modifications:

First, we modified parts of the SOAP-over-UDP library, where we added a new transport class that would apply EXI compression and de-compression to outgoing and incoming UDP packets, respectively. We enabled all the compatibility parameters for EXI (along with the parameter for maximum compression), thus ensuring that the lexical integrity of the XML documents was preserved. This was done to ensure that WS-Discovery functioned properly[1] upon de-serializing the data.

Then, we made two changes to the WS-Discovery library, where we added our new EXI capable transport under available transport types, and finally set this transport to be the default to be used.

Finally, we compiled the libraries and repeated the tests made with the standard implementation.

4.1 Evaluation framework

We evaluated WS-Discovery using WSDLs for services such as finance, news, weather services,

[1] Enabling these EXI compatibility parameters mean that compression rate is slightly reduced, but it ensures that all namespaces and other metadata are preserved. This is especially important if one wants to employ security measures, as changes to the document will break cryptographic signatures.

etc. These WSDLs were fetched from “http://www.webservicex.net/” and “http://www.webservicelist.com/”, which provide lists of freely available Web services. Also, the WSDLs from Google and Amazon’s search services were included, yielding a set of 100 WSDLs in total. This provided us with a representative set of interfaces (see Table 1) for a wide array of Web services which we could use in our evaluation.

Minimum size	Maximum size	Average	Standard deviation	Median
1643	149342	13830	19202	8514

Table 1: Sizes (in bytes) for our 100 WSDL files.

We used Wireshark version 1.2.1 for Windows from “http://www.wireshark.org/” to capture data traffic in a small network with two nodes. This enabled us to capture actual WS-Discovery traffic, and examine the packet payload sizes, thus giving a foundation for further analytical study.

4.2 WS-Discovery network usage evaluation

The standardized WS-Discovery behavior is a decentralized discovery protocol. Services are required to send UDP multicast HELLO messages that advertise when they become available. Also, services should send UDP multicast BYE messages when they go away. If services are able to do this, then each node will have an up-to-date view of the available services. In a dynamic network we cannot rely on receiving all such messages. It is also possible to actively query the network by sending PROBE messages. In order to accurately mirror the current network state of a dynamic network probing must be used, in which case each node replies with UDP unicast PROBE MATCH messages. This generates a lot of data traffic, but is required to ensure an up-to-date view of the available services. The standard requires all multicast packets (i.e., HELLO, PROBE, and BYE messages) to be sent twice, and the unicast PROBE MATCH messages to be sent once. Since we are concerned with WS-Discovery in dynamic environments, we focus on the HELLO, PROBE, and PROBE MATCH messages in this paper.

WS-Discovery is based on a query-response model, where a multicast query (probe) triggers unicast responses (probe match). The load incurred on the network by the number of querying nodes (q) in a network with a total number of n nodes can be calculated using the formula below. If all nodes should have an up-to-date view of the currently available services, then q = n, conversely, if only one node is querying, then q = 1.

$$LOAD = (sizeof(probe) + sizeof(probe\ match) * (n - 1)) * q$$

In our tests the HELLO messages yielded different sizes (see Table 2) depending on the different WSDLs that were published (two HELLO messages generated per WSDL).

An uncompressed PROBE message was always 581 bytes (using a generic probe querying for all available services with no scope limitations). An EXI compressed PROBE message varied between 272 and 274 bytes (compression varying with varying UUID and time stamp in message; for simplicity we assume a compressed size of 273 in our calculations below as this is the average over time). According to the standard the message had to be transmitted twice, meaning that sizeof(probe) = $2 * 581$ bytes for uncompressed traffic (EXI compressed sizeof(probe) = 2 * 273 bytes).

Compression	Minimum size	Maximum size	Average	Standard deviation	Median
Uncompressed	807	887	834	17,04	830
EXI	373	420	390	9,22	388

Table 2: HELLO message payload statistics (in bytes), calculated from HELLO messages corresponding to the Web services described by our 100 test WSDL files. For each Web service that is published, WS-Discovery sends two identical such messages.

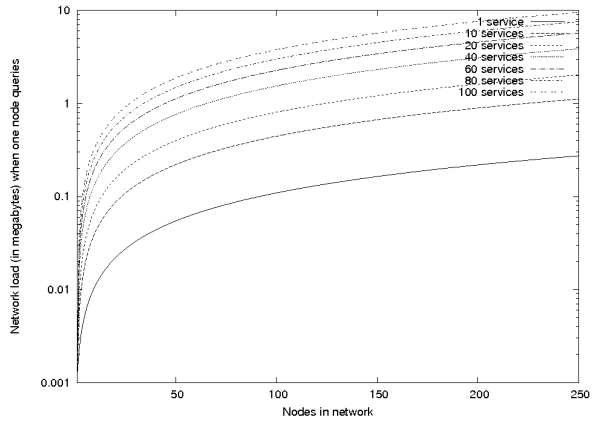
The PROBE MATCH varied in size with the number of services published, since it contained all the services published by a node. Table 3 shows the different sizes of PROBE MATCH messages sent by a node with 1 to 100 services published. We see that publishing just one service incurs a lot of overhead (1092 bytes to disseminate information about it), whereas for a larger number of services this overhead is reduced (more actual Web services porttype information in the response compared to SOAP headers, etc). Calculating the average when publishing multiple services (i.e., the average of the message sizes divided by the number of services) yields 497 bytes for uncompressed WS-Discovery, and 130 bytes for EXI compression. UDP can carry a payload of 65507 bytes, meaning that WS-Discovery has a theoretical upper limit of publishing approximately $65507/497 \approx 131$ Web services per node when considering results from our 100 WSDLs. Conversely, with EXI compression we may publish around $65507/130 \approx 503$ Web services per node. Naturally, the number is approximate, because in practice varying namespace lengths in different WSDLs can affect the PROBE MATCH size. We can also see that an increase in the number of services in a PROBE MATCH leads to an increased compression rate, because of recurring patterns in the XML encoding of the service information. For just one service, the compression rate is $511/1092 \approx 0.47$, whereas for a 100 services the compression rate has increased, yielding $5009/38200 \approx 0.13$.

Number of services	Uncompressed PROBE MATCH	EXI compressed PROBE MATCH
100	38200	5009
80	30292	4000
60	22902	3146
40	15531	2339
20	8122	1552
10	4476	1072
1	1092	511

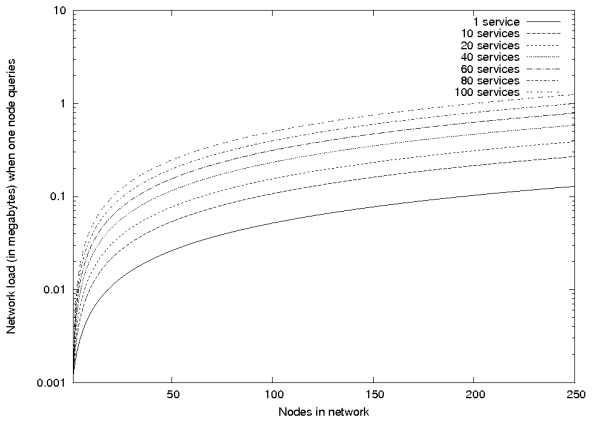
Table 3: The size (in bytes) of the unicast PROBE MATCH message sent by a node publishing a certain number of Web services with WS-Discovery.

Using the LOAD equation, we fill in values for sizeof(probe match) using values from Table 3, as well as the above mentioned sizeof(probe). The number of nodes in the network, n, is varied from 1 to 250. First, we set q = 1, meaning that only one node is querying. Figure 3 illustrates WS-Discovery's resource use (in megabytes) in this case when one node is querying in networks with up to 250 nodes. This means that in such a network, for every query issued, we get the resource use indicated by the graph. Next, we set q = n, so that in a network of a given size, all nodes query. In both cases, this means that the querying node(s) send PROBEs and receive(s) n PROBE MATCHES.

Figure 4 shows WS-Discovery's resource use in the case where q = n. In both graphs, all nodes are equal and publish the same number of services. Please note that the graphs have a logarithmic Y-axis to ease comparison between uncompressed and compressed results. We see that with an increasing number of nodes and published services, the overall resource use increases substantially.

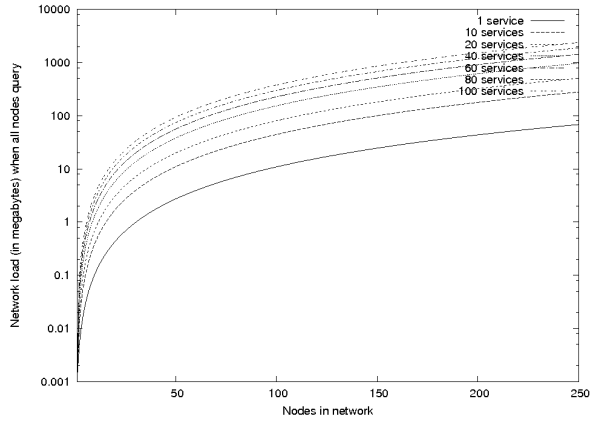


Uncompressed WS-Discovery

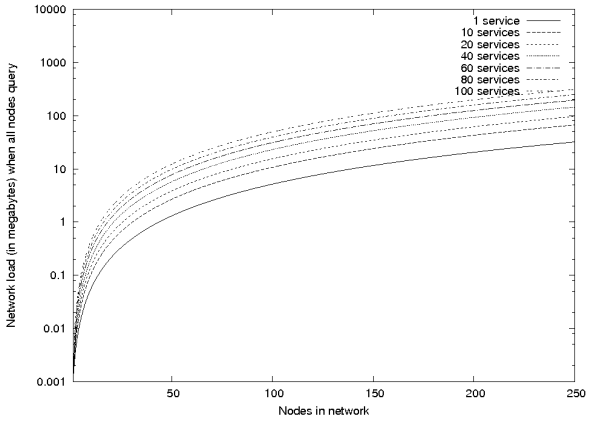


EXI compressed WS-Discovery

Figure 3: WS-Discovery's resource use when one node queries.



Uncompressed WS-Discovery



EXI compressed WS-Discovery

Figure 4: WS-Discovery's resource use when *all* nodes query.

5. Conclusion

Standards are preferable to proprietary solutions because they ease interoperability and reduce the chances of vendor lock-in. Though WS-Discovery in our previous research has proven itself to be less than optimal for use in tactical networks, its resource use is significantly reduced when coupled with EXI as our results in this paper show. Thus, WS-Discovery could well be of value in a civil-military operation, where it could provide a standardized Web services discovery capability for both the civil and the military dynamic networks.

References

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

[2] David S. Alberts and Richard E. Hayes, Planning: Complex Endeavors**,** 2007, http://www.dodccrp.org/files/Alberts_Planning.pdf

[3] OASIS. UDDI Version 3.0.2, UDDI Spec Technical Committee Draft. Luc Clement, Andrew Hately, Claus von Riegen, and Tony Rogers (eds.), http://uddi.org/pubs/uddi_v3.htm, 2004.

[4] OASIS. ebXML registry information model version 3.0 OASIS standard, 2 may, 2005. Sally Fuger, Farrukh Najmi, Nikola Stojanovic (eds.), http://docs.oasis-open.org/regrep/v3.0/specs/regrep-rim-3.0-os.pdf, accessed 2010-01-17.

[5] F. T. Johnsen, J. Flathagen, T. Hafsøe, M. Skjegstad, and N. Kol. Interoperable service discovery: Experiments at combined endeavor 2009. FFI report 2009/01934, http://rapporter.ffi.no/rapporter/2009/01934.pdf, 2009.

[6] OASIS. Web services dynamic discovery (ws-discovery) version 1.1 OASIS standard 1 july 2009. Vipul Modi, and Devon Kemp (eds.), http://docs.oasis-open.org/ws-dd/discovery/1.1/wsdd-discovery-1.1-spec.pdf, accessed 2009-12-28.

[7] T. Hafsøe, F. T. Johnsen, N. A. Nordbotten, E. Skjervold. Using Web Services and XML Security to Increase Agility in an Operational Experiment featuring Cooperative ESM Operations. 14th International Command and Control Research and Technology Symposium (ICCRTS), Washington DC, USA, June 2009.

[8] A. N. Mian, R. Baldoni, and R. Beraldi. A survey of service discovery protocols in multihop mobile ad hoc networks. In IEEE Pervasive computing, January-March 2009, pp. 66–74.

[9] G. G. Richard III, Service and Device Discovery: Protocols and Programming. Mc Graw Hill, 2002.

[10] F. T. Johnsen and T. Hafsøe. Service Advertisements in MANETs (SAM): A decentralized web services discovery protocol. Workshop on Ubiquitous Computing and Networks (UbiCoNet), Dec. 2010.

[11] M. Skjegstad, F. T. Johnsen, T. Hafsøe, and K. Lund. Robust and efficient service discovery in highly mobile radio networks using the Mist protocol. IEEE Military Communications Conference (MILCOM 2010), Oct. 2010.

[12] F. Sailhan and V. Issarny. Scalable service discovery for MANETs. In Third IEEE International Conference on Pervasive Computing and Communications (PERCOM). IEEE Computer Society, March 2005, pp. 235–244, ISBN 0-7695-2299-8.

[13] S. Helal, N. Desai, V. Verma, and C. Lee. Konark - a service discovery and delivery protocol for ad-hoc networks. In proceedings of the Third IEEE Conference on Wireless Communication Networks (WCNC), New Orleans, 2003.

[14] M. Amoretti, F. Zanichelli, and G. Conte. SP2A: a service-oriented framework for P2P-based grids. In proceedings of the 3rd International Workshop on Middleware for Grid Computing (MGC05), Grenoble, France, 2005.

[15] A. Harrison and I. Taylor. WSPeer — An Interface to Web Service Hosting and Invocation. In HIPS Joint Workshop on High-Performance Grid Computing and High-Level Parallel Programming Models, IPDPS, Denver, Colorado, USA, 2005.

[16] OASIS. SOAP-over-UDP version 1.1 OASIS standard 1 July 2009. Ram Jeyaraman (ed.), http://docs.oasis-open.org/ws-dd/soapoverudp/1.1/os/wsdd-soapoverudp-1.1-spec-os.pdf, accessed 2010-02-23.

[17] XML Binary Characterization Working Group Public Page. http://www.w3.org/XML/Binary/

[18] John Schneider (AgileDelta, Inc.), Efficient XML: Taking Net-Centric Operations to the Edge. 13th International Command and Control Research and Technology Symposium (ICCRTS), Seattle,WA, USA, June 2008.

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

[20] M. Gudgin et al. SOAP Version 1.2 Part 1: Messaging Framework (Second Edition). W3C Recommendation 27 April 2007, http://www.w3.org/TR/2007/REC-soap12-part1-20070427/#bindfw

Abbreviations

Bluetooth SDP	Bluetooth Service Discovery Protocol
DP	Discovery Proxy
ebXML	electronic business using XML
EFX	Efficient XML
EXI	Efficient XML Interchange
MANET	Mobile ad-hoc network
NNEC	NATO Network Enabled Capabilities
OASIS	Organization for the Advancement of Structured Information Standards
SAM	Service Advertisements in MANETs
SLP	Service Location Protocol
SOAP	Called the "simple object access protocol" up to and including its release as a W3C version 1.1 note, but this name did not describe exactly what SOAP was, and so it was later dropped. In its current version, the W3C version 1.2 recommendation, the protocol is just called "SOAP".
SP2A	Service-Oriented P2P Architecture
UDDI	Universal Description, Discovery and Integration
UPnP	Universal Plug and Play
UUID	Universally unique identifier
W3C	World Wide Web Consortium
WS-Discovery	Web Services Dynamic Discovery
WSDL	Web Services Description Language
XML	Extensible Markup Language