Web Services Over Infrastructure-less Networks

Arvind M Bhusate, Dr Hermann De Meer

Department of Electronic & Electrical Engineering University College London, 2003

Abstract: *This paper discusses the possibility of the convergence between web services and infrastructure-less networks such as Ad-hoc and P2P. Convergence meaning the operation of web services over networks such as Ad-hoc and P2P. The paper highlights parts of the detailed analysis carried out and the conclusions formed from this in these areas. It also reviews the fundamental web services technology components, along with that of ad hoc and p2p networks.*

1. Introduction

Web services tend to be accessed at all times and at all places. Clients employ a wide range of devices including desktops, laptops, palm or hand held devices, and smart phones that are connected to the Internet using very different kinds of networks such as wireless LAN, cell phone network, broadband network, telephone network, or local area network. Frequent disconnections and unreliable bandwidth characterise some of these networks. The availability of Web services is thus a significant concern to consumers using mobile devices and working in different kinds of mobile networks either with or without an infrastructure. Therefore this research is necessary to be carried out to discover any models or analyse existing models/ architectures to enhance and diversify the use of web services (such as over infrastructure-less networks). This paper discusses possible ways that convergence maybe applicable, the issues involved in each case and also the problems that will be encountered with such proposals.

2. Web Services

Web Services are self-contained, modular applications that can be described, published, located, and invoked over a network. They are based on service-oriented architecture (SOA) and enable new and existing applications to be integrated through the use of XML as a data format over standard network protocols such as HTTP for transportation.

Web Services eliminate the interoperability issues of existing solutions, such as CORBA and DCOM, by using open Internet standards such as: WSDL, SOAP, UDDI. The WSDL (Web Services Description Language) is an XML-based format, which holds information such as the services available methods, parameter types and the actual SOAP end point for the service. The WSDL is like the "Users Manual" for a Web Service. The SOAP (Simple Object Access Protocol) is the XML-based protocol for sending requests and responses to and from Web Services. The UDDI (Universal Description, Discovery, and Integration), also known as the "Yellow Pages" for Web Services, is like the meeting place for web services. The UDDI registry stores descriptions about companies and the services they have to offer in a XML format.

There are three components of service-oriented architecture: 1) Service Provider: is responsible for creating and publishing the interfaces of services, providing the actual implementation of these services, and responding to any requests for the use thereof. 2) Service Broker: registers and categorises public services published by various service providers and offers services such as search. Service brokers act like a yellow pages for Web Services. 3) Service Requestor: This component is the actual user of Web Services. Service requestors discover Web Services and then invoke these services by communicating with the actual service providers [1].

The three most basic operations through which the SOA participants interact are: 1) Publish operation, using WSDL, allowing the service provider to publish its services and interface requirements with a service broker. 2) Find operation, using UDDI registry, allowing a service requestor to locate, search and discover the services published via a service broker that is offered by a specific service provider. 3) Bind operation, using SOAP based XML messages, enabling a service requestor to actually bind and use a service provided by a service provider [2].

3. Ad hoc Networks

Ad hoc networks are infrastructure-less networks and are basically made up of mobile nodes. The network is mobile and is also a form of wireless communication. Ad hoc networks are flexible, self-configuring and robust against disasters. There are minimal wiring difficulties and they are very speedy and easy to deploy. The problem with these networks however is that deployment costs are high, and there is a limited wireless spectrum (licensing and low bandwidth) [3]. Ad hoc networks are formed when two or more devices (or hosts) are in proximity of each other. More devices may join at any time and can also leave at any time and thus may become unreachable. Two ad hoc networks may

also merge to become one at anytime, and one Ad hoc network may partition into two. No base station is necessary, however a shared vision is.

Ad hoc networks require multi-hop routing [4] (using algorithms such as AODV, ad-hoc on demand distance vector routing algorithm) and each device acts as a router/switch in order for communication to take place. An altemative approach to this would be for the designated devices to act as routers, and the others simply as end nodes.

4. P2P Networks

P2P is a communication model in which each party has the same capabilities and either party can initiate a communication session. Users can use the Internet to exchange files with each other directly or through a mediating server. P2P's architecture allows for decentralised application design, moving from a centralised server model to a distributed model where each peer, independent of software and hardware platforms, can benefit from being connected to millions of other peers. Napster and Gnutella are examples of P2P using the Internet.

A P2P application should be able to discover other peers in the network. Since P2P networks cannot operate in an environment of stable connectivity and fixed IP addresses, the nodes have to operate outside the Domain Name Service (this converts names to IP addresses) system. This is fundamental to the nature of P2P systems, and is how they work. If they used stable IP addresses, they would be no different from a normal infrastructured network [4]. So, the method used by the peers to locate each other is achieved in a pure P2P-based architecture by network broadcasting. Once an application is able to locate other peers, it should be able to communicate with them using messages. Once the communication is established with other peers, the application should be able to receive and provide information, such as content.

5. The Convergence1

A good solution to convergence should be transparently deployable and generally applicable. Transparent deployment means that the solution must not require changes to the implementation of the Web services, either to the server and client side modules or to the communication protocol between them. The growth in the number of Web services has been huge and hence applying changes to existing Web services is an improbable proposition. For the same reason, the solution should also be scalable and general enough to apply to all the Web services.

5.1 Web Services With P2P

There are quite a few common features of P2P and Web Services technologies, as they both aim to become a common stack for publishing and discovery across networks. Since P2P networks are based on a decentralised model and its primary focus is on supplying processing power, content, or applications to peers in a distributed manner, it is less focused on messaging formats and communication protocols. Web Services, on the other hand, is based on a centralised model and its primary focus is on standardising messaging formats and communication protocols. The convergence of these two systems would enable companies to use the processing power benefits of the P2P networks, through the standard formats for discovery and content exchange over the standard communication protocols of the Web Services world [5].

Web Services provide a very elegant way to handle: registration, discovery, and content lookup for P2P applications. As Web Services mature, their security standards would also ensure the integrity of data and services accessed by P2P software. Both user and application centered Web Services would play a role in P2P systems as the clients or nodes of a P2P system, ranging from a user, desktop, laptop, PDA, or Pocket PC to a server.

Some examples of how large hardware and software vendors are planning to support P2P and Web Services platforms together: 1) Intel is building P2P components and services on top of Microsoft's .NET platform. Microsoft's Visual Studio .NET and Intel's P2P services are intended to serve as infrastructure for vendors developing P2P services like instant messaging (IM), knowledge management, and collaboration. 2) The Project JXTA P2P platform, supported by Sun Microsystems, is being changed at the core to make peers interoperate with Web Services using protocols like SOAP and WSDL [6]. 3) The Microsoft .NET Framework provides a rich platform for building P2P applications [7].

Both P2P and Web Services technologies help to reduce some of the core complexities of each other. The communication between different peer systems should be based on open standards. This would allow companies to have different architectures and platforms for their P2P systems. Thus it is safe to

say that both P2P services and Web Services would rely on WSDL and SOAP for service descriptions and invocations [5].
Many design problems associated with directory services, communication protocols, and message formats are being addressed by P2P applications. UDDI is expected to reduce the complexity and amount of effort required to create P2P applications. As far as directory services are concerned, P2P technology has a lot to offer to Web Services technology as well. Using P2P, the Web Services implementation of using a single central UDDI registry, which contains the service description of Web Services, can be converted into a decentralised mode. It would be extremely difficult for a single registry provider to maintain that information and provide a high level of service. Since a P2P network is completely decentralised, the Web Services descriptions (the content of the UDDI registry) can be described and indexed locally to a given peer. The realistic vision is that the P2P systems would continue using the decentralised discovery model, while Web Services continue using the centralised discovery model of UDDI. Search engines would probably be the most widely used applications based on P2P and Web Services technologies, such as Google.

There are a few challenges of P2P and Web Services converging that arise. A critical factor of the success of P2P technology and its convergence with Web Services is the available network bandwidth. As the number of peers searching for content or communicating in the network increases, the resulting traffic can potentially block the whole network bandwidth. P2P applications eliminate central servers and create a loose, dynamic network of peers. Thus any content retrieval operation, such as a search for a specific item in a database, all the peers in the network are searched, using a lot of network bandwidth. As this network size increases and becomes more distributed, it may be affected by poor and slow connections [5]. The decentralised and distributed architecture of P2P is also one of its largest weaknesses. P2P applications are a major security threat to companies, as by definition they distribute data and computing resources over several peers. The architecture of P2P-based applications with or without Web Services integrated is extremely complex. It involves security issues and use of distributed resources in an optimal way, making decentralised and independent systems work as one. Maintenance of such applications is much more difficult since it is extremely difficult to identify, replicate, and fix application or network related problems.

The technologies enabling this convergence are some of the leading P2P protocols such as Jabber and JXTA. Jabber is an instant messaging and presence notification, open source XML-based protocol. It is platform neutral, thus Jabber is interoperable with messaging across both wireless and browser based messaging services. The real-time messages are exchanged as XML streams. Jabber's open XML protocol contains three top level XML elements, which in turn contain data through attributes and namespaces [8]. Project JXTA is an open network platform for P2P computing. It provides a framework and a set of protocols that standardise the way in which peers discover and advertise network resources, and communicate and cooperate with each other to form secure peer groups. Publishing and exchanging XML messages among participating peers perform all the functions supported by JXTA. JXTA is independent of programming language. Heterogeneous digital devices such as PCs, servers, PDA's, and appliances with completely different software stacks can interoperate with JXTA protocols. Moreover, it is independent of transport protocols such as TCP/IP and HTTP and can be implemented on top of them like SOAP [6].

5.2 Web Services With Ad Hoc Networks

A possible solution for web services working over this network could be the caching of web services. Since mobile nodes join and leave when they please there are times of disconnection from the Internet and as a result from the web service. Caching the web services to support this disconnected operation between nodes, which are connected to one another but not to the gateway providing the communication to the Internet seams to be a feasible solution for web services working over infrastructure-less networks. Service discovery would obviously have to be done when one the mobile nodes of the ad hoc network are connected to the Internet and using the AODV algorithm to route messages to the other nodes also trying to discover a web service and invoke operations on it.

An experiment conducted as part of research at Microsoft in which a caching proxy was placed between Microsoft's .NET My Services web service and the sample clients that ship with these services. The .NET My Services were chosen for this experiment because they were the only example of publicly available, non-trivial XML Web services that support both query and update operations [9]. An HTTP proxy server was built and deployed on the client device. All HTTP messages originating at the client, including those generated by Web clients and Internet browsers were made to pass through the proxy server. The proxy server acts as a simple tunnel for all HTTP packets that are not SOAP messages. A cache for storing SOAP messages was added to the proxy server. This cache stores SOAP messages received in response to SOAP requests. Whenever the network is connected, the SOAP request received is sent to the server. The received SOAP response is stored in the cache associated

with the request, replacing the old response if it exists. In the case where the cache contained a previous response for the same request, the new SOAP response is compared to the previous response and the result of the comparison is recorded in a log file. These comparisons provide valuable insight into what kind of operations can be cached and what other operations affect the validity of the cached responses. If the network is disconnected, the SOAP response stored in the cache is returned to the client and the SOAP request is stored in a write back queue. All requests are stored in the write back queue for later replay since the cache manager cannot determine which requests modify the service state and which are simply non-update queries. Whenever the connection to the Web service is restored, the queued up SOAP requests are played back to the server and the SOAP responses are stored in the cache [10].

The experiment was able to demonstrate that web services such as the one examined can be used during disconnections with little awareness of the disconnection. It also proved that caching everything works well for the large class of Web services that only provide query operations on fairly static data, such as map services, currency conversion services and yellow page services. However, the experiment also exposed a number of issues that need to be handled in order achieve a significant improvement in the consistency and availability of offline access to Web services.

When operating in disconnected mode, a cache manager cannot provide strong consistency (maintenance of correctness and validity of data) guarantees since it does not have access to updates performed by other users. In order to maintain correctness, the earlier response in the cache would have to be either correctly modified or deleted from the cache [9]. Ideally, a user should obtain the same experience when disconnected as when connected. However, achieving the ideal goal is not practically feasible, especially if the Web client is unaware of the existence of the cache.

To eliminate some of the issues arising from the standard caching architecture an enhanced Web services cache architecture has been developed. This adjusts its behavior for different Web services and is based on WSDL annotations to improve availability of Web services [10]. The Web service cache resides in a HTTP proxy server on the client device as before. A Custom Cache Manager is a new addition to enhanced web services cache that is generated automatically for each Web service that has an annotated WSDL specification. The custom cache manager controls the behavior of the cache according to the annotations described in the WSDL document. For regularly used Web services, annotating the services WSDL documents is recommended so that custom cache managers can be used. The default cache manager can be useful in those cases where annotations are not available.

6. Conclusions

It seems the integration of web services with infrastructure-less networks is a very useful concept and should definitely be deployed seamlessly in the future when the web service technology it self has matured along with the P2P and Ad hoc without a doubt. Even now it is possible but with many limitations in place and also without maximum transparency or full scalability. The models discussed in section 5 require further evaluation and research before they can be widely adopted for everyday use. Further work for this project includes implementing simple web services over infrastructured networks (including wireless networks) to show what does work and to show the diversity of web services. In addition to this, further expanding on the models and technologies reviewed in this paper to complete the detailed analysis.

References

[1] Web Services Architecture: Direction & Position Paper, Donald F. Ferguson, dff@us.ibm.com IBM Corporation Paper for W3C Web Services Workshop, April 11-12, 2001

[2] The Java Web Servcies Tutorial, Eric Armstrong, Stephanie Bodoff, Debbie Carson, Maydene Fisher, Dale Green, Kim Haase, August 1, 2002, http://java.sun.com/webservices/tutorial.html

[3] Ad hoc networking, C. E Perkins, Addison Wesley, 2001

[4] Distributed Systems Concepts and Design, George Coulouris, Jean Dollimore, Tim Kindberg, Third Edition, Addison Wesley, ISBN: 0201-61918-0, 2001

[5] Web Services and P2P Computing, Gunjan Samtani, Dimple Sadhwani, Tect, 2002, 29 South LaSalle St. Suite 520 Chicago Illinois 60603, USA, http://www.webservicesarchitect.com

[6] Project JXTA v2.0: Java Programmer's Guide, Sun Microsystems, May 2003, http://www.jxta.org/docs/JxtaProgGuide_v2.pdf

[7] Support WebCast: Microsoft .NET: Introduction to Web Services, published under Q324881, 2003, http://support.microsoft.com/default.aspx?scid=kb;en-us;324881&gssnb=1

[8] Jabber Technology Overview, Jabber Software Foundation, 2002, http://www.jabber.org/

[9] Caching of XML Web Services for Disconnected Operation, Venugopalan Ramasubramanian and Douglas B. Terry, Microsoft Research, 1065 La Avenida, Mountain View, CA 94043, 2002

[10] Caching XML web services for mobility, DOUGLAS B. TERRY AND VENUGOPALAN RAMASUBRAMANIAN, 2002, http://www.cs.cornell.edu/people/ramasv