

An Introduction to Web Services

Christopher A. Brooks
University of Saskatchewan
57 Campus Drive
Saskatoon, SK, S7N 5A9
(306) 966-4743
cab938@mail.usask.ca

ABSTRACT

Application frameworks for distributive computing over the Internet are generally made up of object models that are inseparable from messaging protocols. This forces developers to adopt an “all or nothing” stance when picking which distributed protocol to use. Conversely, the World Wide Web is a loosely designed distributive system which has seen rapid adoption by end users, but poor integration with applications due to the lack of well defined interfaces and workflow languages. Web services are the convergence of these two areas, and provide a platform and implementation-neutral way of doing distributive computing while leveraging lessons learned from the World Wide Web. This paper introduces web services and the specifications that deal with them, and provides an initial report on how web services are being used within the I-Help project.

General Terms

Design, Standardization.

Keywords

Web services, SOAP, WSDL, XML Schema, RDF, WSFL, XML-RPC, UDDI.

1. INTRODUCTION

A number of distributed programming protocols exist to connect applications over both local area and wide area networks. Three widely used protocols include the Common Object Request Broker Architecture (CORBA) [34] from the Object Management Group, the Distributed Component Object Model (DCOM) [40,30] from Microsoft, and the Remote Method Invocation (RMI) [19] protocol from Sun Microsystems. Each of these protocols exhibits at least one of the following problems:

- Non-interoperable with the other protocols
- Language or operating system dependant
- Vendor specific distributed object model
- Vendor specific serialization format for message passing

The rise in popularity of the World Wide Web has brought a new end-consumer driven method of distributed computing. Distributed calls are made through hyperlinks and encoded in HTTP messages as GET or POST requests. Responses generated by the server are sent in HTML over HTTP, and are rendered immediately by the web browser. Simple processing can be done

to the results of calls using JavaScript or a similar client-side scripting environment. Message formats tend to vary from server to server, and scripting implementations are different between each web browser. This method of distributed computing is simple and has given way to millions of services being offered and used around the world.

There is no standardized mechanism to interrogate the web at large for a list of services available, and no set of standard interfaces that describe how the services available should be used. The limited expressiveness and control afforded by HTML and HTTP means that only simple interoperability between web sites can be achieved. Cauldwell et al [13] provide a brief overview of the technologies and techniques that have been used to try and make HTMLbased web services support a higher degree of interoperability. In general, HTML and HTTP-based web services require complex processing on the client side to understand and interpret server messages, and are extremely inflexible.

The Extensible Markup Language (XML), widely heralded as the new *lingua franca* of the World Wide Web, helps to solve some of these problems. By providing a standardized, vendor extensible communication syntax, and a set of modular display primitives (XHTML), browser interoperability can be achieved.

But how can automated software entities discover and use these web services? How can data validity constraints, both on the input and output parameters of a web service, be enforced? Can a set of web services be linked together to support business-to-business transactions and determine higher level process flows? The answers to these questions have been the continued work of the World Wide Web Consortium (W3C) Web Services Activity [24], a set of three working groups devoted to specifying and supporting XML based web services. Each working group is made up of members of academia and the information technology industry, and contributes through the publication of joint specifications.

This paper is organized as follows. Section 2 gives a brief survey of the work going on with web services, and presents details on the major standards and specifications that have been released. Section 3 outlines the I-Help public discussion system and describes ongoing work on making this system web service compatible. Section 4 presents a preliminary analysis of the implementations, and gives reflections on the specifications involved. Finally, Section 5 gives a set of conclusions and recommended directions for future investigation.

2. WEB SERVICES

2.1 Overview

With the creation of a common and extensible syntax for describing services on the web (XML), a common vocabulary of how a web service is to be used is required. The W3C Web Services Architecture Working Group [14] is the principle investigator of this activity. The group is newly formed (January of 2002), and is expected to produce several high level and technical specifications by the year 2004.

While this suggests that there is currently no singly accepted architectural model for web services as a whole, a number of vendors have already begun work on defining how web services will be used with their products. Myerson [17] provides a comprehensive overview of the work done in this area, including submissions that have been made to the W3C Architecture Working Group. The work done by IBM and Microsoft for the W3C Workshop on Web Services [35] serves as a good list of requirements needed for web services. In general the web service support model is made up of at least three layers:

- Wire layer – defines the format of message components as they are passed between web service entities.
- Description layer – describes the format of information in the wire layer through meta data documents.
- Discovery layer – provides a framework for discovering the information in the wire and description layers.

2.2 Wire Layer

The wire layer is the lowest layer in the web services framework, and is made up of components to support the exchange of higher level services. Components in this layer can be thought of low level syntax descriptions required for message passing. This layer is responsible for message routing syntax, simple transaction support, digital signature and encryption support, quality of service contracts at the message level, and remote procedure call marshalling.

This layer has seen an extensive amount of development since XML was introduced, and has been the base for all web service development. While a number of protocols have been proposed, only two have seen widespread adoption. XML-RPC has been generally adopted for lightweight services that do not need complex routing or data typing mechanisms, while the Simple Object Access Protocol (SOAP) has been adopted for more heavy weight services. Each protocol defines an XML based message format, and abstains from enforcing any specific object model (such as distributed garbage collection). Both have a significant number of implementations and toolkits available, and are backed by major vendors such as IBM, Microsoft, and Sun Microsystems.

2.2.1 XML-RPC

The XML-RPC specification [23] was developed by UserLand Software Inc. in 1999 as a simple information exchange protocol for the Frontier web server. It is built on top of both HTTP 1.1 and XML 1.0, making it easily understandable for most web servers. Each XML-RPC request is sent as an HTTP-POST message, while responses are sent with an HTTP 200 response message, unless there are low level processing faults (typically this happens when the web server does not understand XML-RPC and rejects the request). All XML-RPC messages must have

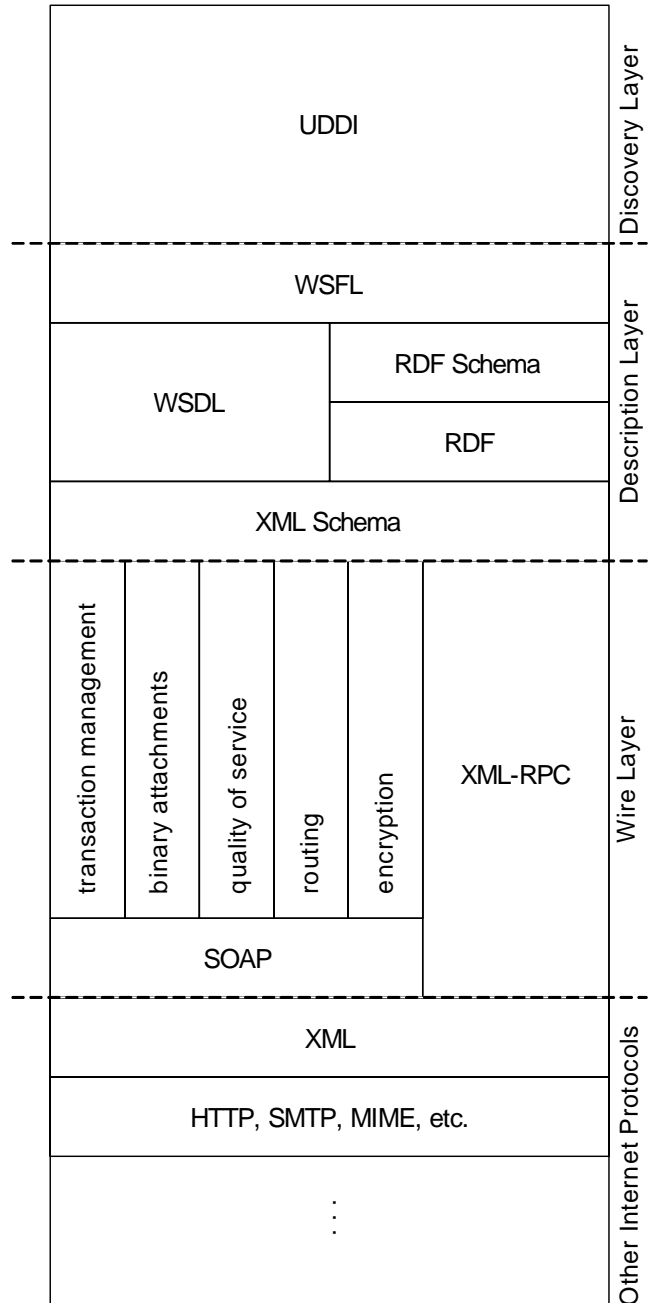


Figure 1. Web Services Technology Framework

HTTP headers that include the User-Agent, Host, Content-Type, and Content-Length fields as describe in section 14 of RFC 2616 [5]. The Content-Type must be "text/xml", and the Content-Length must be correct for the message.

Each XML-RPC request begins with a *methodCall* element containing one or more *methodName* elements. Each *methodName* element identifies the service that is being invoked – this is server dependant, and can include a URI to a script, a method name, a static object, a database table cell, or some other program resource. Parameters are each encoded within a *param* element as one of seven basic XML-RPC types. Each *param*

element is listed with order being enforced under a single *params* element.

In addition to unnamed basic types, XML-RPC also contains the notion of a named basic type through the *struct* element. A *struct* is made up of one or more members, each of which contain a *name* element which distinguishes the resource name, and a *value* element which distinguishes both the type and the value of a resource. Note that an individual *struct* does not have a name, but could be added as a named resource to another *struct*. Finally, XML-RPC also includes a way of representing a set of multi-typed unnamed resources using the *array* element. The array contains a single data element with a set of value elements, each containing a type resource. Arrays can be recursive, and can include not only the basic types but *structs* as well. A complete mapping of XML-RPC types to Java types is given in table 1.

Table 1. XML-RPC to Java Type Mappings

XML-RPC Data type	Java Data Type Mapping
<i4 />	java.lang.Integer
<int />	java.lang.Integer
<boolean />	java.lang.Boolean
<string />	java.lang.String
<double />	java.lang.Double
<dateTime.iso8601 />	java.util.Date
<struct />	java.util.Hashtable
<array />	java.util.Vector
<base64 />	byte[]
None	java.lang.String

Message responses must contain a single *methodResponse* element. If the remote call succeeded, the *methodResponse* contains a single *params* element with a single *param* element encoded as described previously. If the remote call failed, a single fault element is returned. Each *fault* element contains a *faultCode* element that holds an integer, and a *faultString* element that holds a user readable string. The *fault* element can contain no other elements.

2.2.2 Simple Object Access Protocol (SOAP)

The Simple Object Access Protocol version 1.0 was started in 1998 by members of Microsoft, UserLand Software Inc., and DeveopMentor. This was submitted to the Internet Engineering Task Force as an Internet Draft for XML based communication over HTTP. Members of IBM Corporation, and Lotus Development joined in the effort, and version 1.2 (the current version as of writing) was submitted to the World Wide Web Consortium as a W3C Note. Just like XML-RPC, SOAP is information exchange protocol for typed data. Unlike XML-RPC, SOAP version 1.1 is deliberately decoupled from HTTP. This allows for developers to implement remote procedure calls in SOAP over different wire protocols such as smtp/pop3. The only transport syntax supplied in the specification, however, is HTTP.

The SOAP specification is made up of three parts: an envelope, a set of encoding rules, and an RPC convention. The envelope sits in its own namespace and is made up of a top level *envelope*

element, a *header* element and a *body* element. Headers are a mechanism for extending the message on the fly. Elements added under the *header* are collectively known as "headers", and must be namespace qualified. They may include attributes that set the style of encoding, set the intended recipient of the element, or require that a SOAP processor must understand a given header processing. The *body* element holds all data intended for the final recipient of the message. All elements contained in the body must be namespace qualified.

While XML-RPC maintains a set of strict and simple encoded data types, SOAP provides a large set of data types as well as providing a method for developers to define their own encoding styles. Both simple and complex types are defined by the XML Schema Recommendation (described in section 2.3.1), which are extended to include an invariant type (data without a given type), arrays, and compound types. SOAP also allows the developer to distinguish between values and references, within the scope of the document. That is, one parameter may just be a reference to another parameter within a given SOAP message, but a parameter cannot be a reference to a value that is not included with the SOAP message.

Invariants are represented as elements containing no type attribute, and arrays are represented using an *arrayType* attribute which is namespace qualified in the encoding namespace. The *arrayType* attribute contains both the type of the data in the array, and the size of the array – SOAP arrays cannot contain mixed type elements. The SOAP specification also includes mechanisms for identifying a partially transmitted array, and a sparsely populated array (an array containing some null elements). Finally, generic compound types are valid and are implemented by untyped elements containing typed elements.

The SOAP specification defines a set of guidelines to follow when doing RPC:

- method calls are modeled as structs which are identically typed to the method signature
- parameters appear in the same order as supplied in the method signature
- responses to method calls are modelled as structs, which contain the return value followed by values for any parameters passed by reference

If a fault occurs, it must be the only element in the body of the return message. The *fault* element must contain a *faultcode* element and *faultstring* element similar to those in XML-RPC. Further, the *fault* element may contain a *faultactor* element which describes which processor caused the fault, and a *details* element which contains a list of human readable detail elements (such as a stack trace).

2.3 Description Layer

The description layer contains formal semantics for describing the messages that a web service can understand, the restrictions applied to the data within those messages, the categories or ontologies that those messages fall into, and the way in which web services can be combined with one another to support business workflows.

Data typing is handled by XML Schema, while semantics for web service interfaces are handled by the Web Service Definition Language (WSDL). The Resource Description Framework

(RDF), and the Resource Description Framework Schema (RDFS) allow for creating ontologies by which to categorize messages and services. The specification most lacking in the web service framework is that for process and business flow, however, the IBM backed Web Services Flow Language (WSFL) appears to be the most popular and widely used syntax at the moment.

2.3.1 XML Schema

XML Schema [22,21] is a complex data typing mechanism for XML documents. It allows for both the definition of new data types, and the redefinition or extension of other data types, and does not force the user to adopt any specific programming methodology. XML Schema relieves the ambiguities that can be brought up while using Document Type Definitions (DTDs), the default data typing mechanism for XML documents. XML Schema has been widely adopted as the new *de facto* data typing mechanism for XML documents.

XML Schemas support vendor defined extensions through the use of XML Namespaces [11], and provide a mechanism to uniquely identify individual schema instances. Vendors can define both simple string based datatypes, or more complex XML document fragment datatypes. Datatypes can be extended or restricted through the use of twelve data type facets, and forty-four default simple types.

2.3.2 Web Services Definition Language (WSDL)

The WSDL is analogous to Java interfaces, C++ header files, or the CORBA Interface Definition Language (IDL). It defines a grammar for describing the endpoints, inputs, and outputs of a web service. Typically WSDL files are given to developers when applications that use a specific web service are created. A web service can also be queried for its' WSDL descriptions, facilitating run time inspection.

Messages between web services are broken up into *part* elements which are represented by named document fragments. Operations are named entry points into a web services and are described as *portTypes* elements. WSDL defines four different types of transmission primitives:

- One-way, client initiated with no response
- Request/Response, client initiated with server response
- Solicit/Response, server initiated with client response
- Notification, server initiated with no response

XML Schema is used as the default data typing mechanism, but other data typing languages can be substituted. Services can be bound to different messaging protocols as well, such as SOAP messages or HTTP GET URLs.

Box 1 provides an example of a Request/Response WSDL definition that defines a method *GetMessage* that takes a single *int* as an input parameter and returns a single *string* as the element *TextMessage*.¹

Box 1. WSDL Definition Example

```
<?xml version="1.0" encoding="UTF-8"?>
<definitions name="ForumMessages" ... >
  <!-- Define data types -->
  <types>
    <elements name="message" type="string" />
    <elements name="messageNumber" type="int" />
  </types>

  <!-- Define input parameters -->
  <message name="GetMessageInput">
    <part name="body" element="messageNumber"/>
  </message>

  <!-- Define output parameters -->
  <message name="GetMessageOutput">
    <part name="TextMessage" element="message"/>
  </message>

  <!-- Define ports (methods) -->
  <portType name="GetMessagePort">
    <operation name="GetMessage">
      <input message="GetMessageInput" />
      <output message="GetMessageOutput" />
    </operation>
  </portType>

  ...
</definitions>
```

2.3.3 Resource Description Framework (RDF)

The Resource Description Framework (RDF) [38] is a specification produced by the W3C Semantic Web Activity, and is meant to define a grammar that can be used to semantically mark up web resources. Its role within the web services framework is fuzzy, but it has been suggested for use in both the description and discovery layers. An RDF document describes any resource that is identified by a URI (such as a web service endpoint) by creating a directed acyclic graph where edges are predicates and nodes are either subjects or objects. This allows for RDF statements to be viewed as an entity-relationship diagram. Figure 2 shows an example definition of a web service endpoint done in RDF.

RDF documents are seen as key to providing useful metadata on the web, and as essential to the semantic web movement [39]. With RDF Schema [2], a type declaration mechanism for RDF

¹ XML syntax can be extremely verbose, especially when using namespaces. Examples provided in this paper are meant to help the reader understand the basic issues but will not always be well-formed or fully specified.

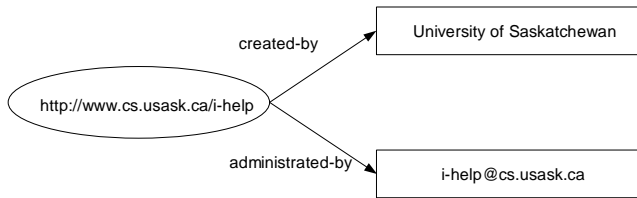


Figure 2. RDF Statement Example

documents, web services can be marked up as to which ontology they belong to. RDF statements can have multiple syntaxes including the graphical syntax shown here, and a more common XML based syntax.

2.3.4 Web Services Flow Language (WSFL)

The Web Services Flow Language (WSFL) [28] is a specification created and supported primarily by IBM. It defines both the interaction of web services to one another, and the mapping between this interaction and business processes models. It builds upon WSDL by connecting entry points into a directed graph and relating them to business processes. This allows for a web service to describe a set of messages and the ordering among them to define a transaction. Workflow documents typically encompass several different web services.

A number of other web service flow grammars are available, including the Business Process Modeling Language (BPML) [9] from the Business Process Management Initiative, and XFLOW [27] from Microsoft.

2.4 Discovery Layer

The discovery layer encompasses processes and protocols by which web services can be discovered either by searching through their metadata (discovery), or by determining the metadata of a web service that is already defined (introspection). This area of web services is the least defined, and is aimed primarily at business-to-business transactions.

While there are no specifications available for introspection, the Universal Description, Discovery, and Integration (UDDI) specification [15] is appropriate for supporting the discovery mechanism. Essentially, the UDDI service is a centralized repository of description layer specifications built much like the Domain Name System (DNS). UDDI registrations can either be on a global scale or deployed within local intranets. UDDI entries include web service entry points and associated metadata published in one or more taxonomies. Currently standardized classification schemes, such as the Standard Industrial Classification (SIC) or North American Industrial Classification System (NAICS), are accepted, as well as vendor specific classifications such as Microsoft's geographical classification system GeoWeb [18]. Data is stored in three different forms; white pages for human readable descriptions of the services, yellow pages for services indexed by classification, and green pages for technical information about the services. Currently four companies provide public UDDI naming services including Hewlett-Packard, International Business Machines, Microsoft, and SAP America.

3. I-HELP

An ongoing research project in the Advanced Research for Intelligent Educational Systems (ARIES) laboratory at the University of Saskatchewan is the use of online peer help systems to facilitate learning. This system, called I-Help is made up of two parts, a private discussions agent-based system and a public discussions web based message board environment. Motivations and a full discussion of the I-Help system are given in [25,20].

Currently the I-Help public discussions system has the interface, content, and service definitions all included in one web site. Users access a set of HTML pages in a four framed interface, and can view and post messages with the aid of JavaScript functions provided. Client requests are sent to the web server where an Oracle 8i PLSQL script controls transactions and composes appropriate HTML responses.

This implementation has shown itself to require high maintenance, as there is no logical separation of the components used, forcing each support staff member to know about presentation, database, transaction, and overall flow details to ensure a working product. In addition to the overhead on staff, various implementations of client side scripting have lead to inconsistent and sometimes crippled user interfaces on popular web browsers such as Mozilla and Internet Explorer [3,37]. Since complex web pages are browser dependant, every change to one layer of presentation must be tested for compliance on each browser.

A number of reasons have motivated the investigation of redeploying I-Help as a web service. These include desires to:

- Decouple content, workflow, and presentation
- Allow ubiquitous access through a variety of devices (Desktop computer, PDAs, etc.)
- Facilitate access to public discussion messages by intelligent agents, which currently requires direct database access, or complex HTML parsing algorithms to decouple the content from the human readable presentation
- Allow for other institutions to extend services as needed for both research and support services, without affecting other institutions are users

4. I-HELP AS A WEB SERVICE

4.1 Description of Implementation

Initial work has been done on redeploying the I-Help public discussions as a web service, with the goal of providing access to the public discussions to users running the Windows CE platform. Specifically, a web service client for the Compaq iPAQ device was implemented using the PersonalJava programming language.

The focus with this client implementation was on the wire layer. Both the XML-RPC and SOAP protocols were explored, and were compared to the Java RMI protocol for performance.

The XML-RPC protocol was implemented with the use of the Helma XML-RPC libraries, now part of the Apache XML project [7]. These libraries allow for automatic conversions of XML encoded objects into corresponding Java instances. There is no built in mechanism to pass complex types between the caller and the web service. Instead, the programmer must handle reserialization of the object from data files passed from the library

as a *Hashtable*. Both synchronous and asynchronous message passing is available.

The SOAP protocol was implemented with the use of the Apache SOAP 2.2 libraries, also part of the Apache project [32]. These libraries have seen constant growth since the initial SOAP specification was released, and allow for a number of time saving measures for programmers. The libraries contain automatic conversions for all of the built in Java primitives and a number of other core classes. In addition, a mechanism for automatically converting any object that conforms to the Bean framework standard [31] is included, so developers spend less time writing conversion utilities. Finally, the Apache SOAP libraries support the SOAP with attachments specification, which allows for the passing of binary data with SOAP messages.

The server component of the implementation was made by using the XML-RPC webservice packaged with the Helma libraries, and by setting up the SOAP library as a servlet running on Apache Tomcat.

4.2 Analysis of Implementation

Several metrics are of interest when examining the implementation. First, the speed at which a message can be sent and responded to is paramount. Three different kinds of remote method call messages that were tested:

- No parameter, simple return
- Simple parameter, simple return
- Complex parameter, complex return

Initial reports on the experiment are in [29], and are summarized in the figure 3. Analysis of variance (ANOVA) within a 95% confidence on the collected data suggests that each protocol speed is significantly different from the others. As an average, the RMI protocol is approximately twice as fast as the XML-RPC protocol, and almost twenty-one times faster than SOAP.

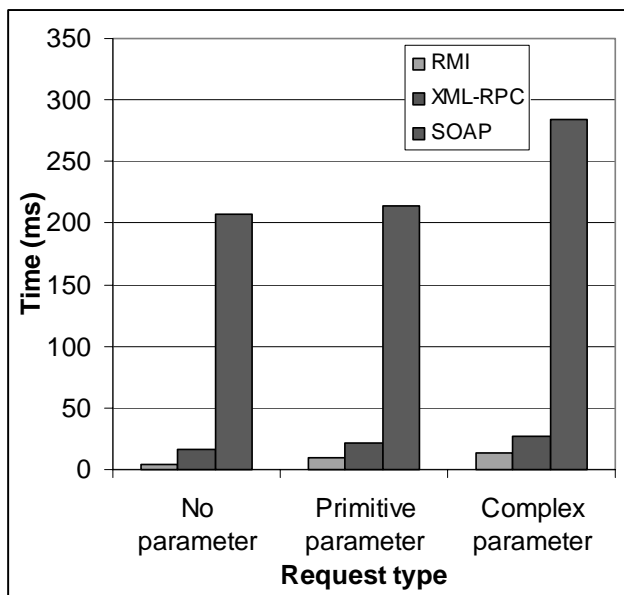


Figure 3. Protocol timing test averages

In addition to repeated requests, it is useful to note that first run startup times also vary tremendously between the three tested

protocols as well. While RMI has no measurable startup speed, the first XML-RPC request took well over 500 milliseconds, and the first SOAP request took over 12,000 milliseconds. These numbers are limited to the default deployment used, and the Apache SOAP libraries allow for more advanced mechanisms of controlling when objects should be loaded.

Some programmers will likely want to minimize overhead by implementing their own library to handle XML-RPC and SOAP requests. SOAP messages tend to be both larger in size and more complex. A typical I-Help Public Discussions request message is approximately three times the size in SOAP over XML-RPC. In addition, the SOAP message uses namespaces and attributes while the XML-RPC message only uses elements. This allows custom library developers to optimize XML-RPC parsers by including only partial XML parsers, ignoring attributes and namespaces.

Finally, two simple comparisons can be made between the two library frameworks that were used. The Helma XML-RPC libraries were a light 64 kilobytes, including a small parser suitable for servicing XML-RPC requests. The Apache SOAP 2.2 libraries, however, require over 200 kilobytes of core files as well as over 500 kilobytes of support files (including the XML parser, bean activation framework, etc.).

4.3 Implementation Conclusions

Most distributed protocols, such as RMI and CORBA, compress messages by using a binary format that is determined by the specification. To fit into the web paradigm, XML was designed to be human readable and writable, and was not optimized for transport. It was also designed to be a non-application specific method of marking up data. This makes XML based messages both bigger to transmit and slower to parse than most other distributed computing message formats.

The absolute poor performance by the SOAP protocol makes one question why it is the most popular wire protocol for web services. While XML-RPC is good for simple distributed calls, the SOAP protocol is more extensible and allows for greater expression. Through the use of namespaces, SOAP allows developers to easily import new data types in excess of those already supported by the XML Schema recommendation. XML-RPC only allows developers the use of the nine built-in types, and all other types must be combinations of those. SOAP also allows for the attachment of binary data, while such data must be character encoded in XML-RPC adding significant size. Finally, XML-RPC does not allow the calling of distributed methods with no return values (one way), which SOAP is specifically targeted at both messaging and remote procedure call frameworks.

Limited work has been done on re-implementing the SOAP version of this client using Microsoft Embedded Visual C++ and the PocketSOAP [26] COM libraries. Initial results suggest that implementation and operating system interoperability be achieved easily, as long as interfaces are well defined (i.e.: have appropriate WSDL pages). While no quantifiable comparisons have been done with this implementation yet, it is unlikely speed gains will be in excess of those provided by competing binary protocols.

5. DISCUSSION AND FUTURE WORK

The web services framework is a rapidly evolving set of specifications and protocols meant to facilitate easy platform and implementation agnostic communication. The three layers

described make up the core of the activity in the area, and are backed by a variety of vendors including Microsoft, IBM, Sun Microsystems, Hewlett-Packard, etc.

The layer that has seen the most activity is the wire layer, with two popular protocols. Over fifty implementations of the XML-RPC protocol exist [8], while more than eighty implementations of the SOAP protocol exist [1]. The World Wide Web Consortium XML Protocol Working Group is critical in continuing the development of SOAP as the XML Protocol, and have released a number of preliminary of requirements documents [10,16,6]. In addition, there are groups looking at defining mappings from typical distributed computing protocols to both SOAP and XML-RPC, such as the Java Community Process group JSR 101 [12].

Web service metadata is key to supporting interoperability between vendors. While technologies such as XML Schema and WSDL define the application programming interfaces for individual web services, workflow languages such as WSFL describe how web services can be linked together. Finally, ontology categorization to help facilitate the Semantic Web effort is handled by using RDF and RDF Schema. Initial efforts at supporting the searching and discovery of web services have been facilitated through centralized UDDI servers. UDDI is not provided through the W3C as most of the web service definitions are. It is expected that as the semantic web effort grows, the W3C will more clearly define how RDF data stores can be used with UDDI.

The web service implementation discussions provided in this paper are a few of the initial observations. The focus to date has been almost exclusively at the wire layer of the web service framework. Once in place, investigations on how RDF data stores and semantic web agents can access these services

6. ACKNOWLEDGEMENTS

Thanks to Dr. John Cooke and Dr. Ralph Deters for discussions, comments, and the equipment required for the implementations.

7. REFERENCES

- [1] Apache Software Foundation. (2001). Apache SOAP. Retrieved December 13, 2001, from <http://xml.apache.org/soap/index.html>
- [2] Apache Software Foundation. (2002). Apache XML-RPC. Retrieved April 2, 2002, from <http://xml.apache.org/xmlrpc/>
- [3] Apparao, V., et al [Eds.]. (2001). XML Protocol (XMLP) Requirements. W3C Working Draft. Retrieved March 23, 2002, from <http://www.w3.org/TR/2001/WD-xmlp-reqs-20010319/>
- [4] Arkin, A. Business Process Modeling Language (BPML) Specification. Working Draft, Version 0.4. March 2001, Business Process Management Initiative. Available online at <http://www.bpmi.org/bpmi-downloads/WD-BPML-20010308.pdf>
- [5] Biron, P., Malhotra, A. [Eds.]. (2001). XML Schema Part 2: Datatypes. Retrieved December 13, 2001, from <http://www.w3c.org/TR/xmlschema-2/>
- [6] Bowes, J. (1999, September 27). Menu Performs Slowly on Internet Explorer. Id 1561. Message posted to <http://www.cs.usask.ca/i-help/>
- [7] Bray, T., Hollander, D., Layman, A. [Eds.] (1999). Namespaces in XML. Retrieved March 23, 2002, from <http://www.w3.org/TR/1999/REC-xml-names-19990114/>
- [8] Brickley, D., Guha, R. V. [Eds.]. (2002) Resource Description Framework (RDF) Schema Specification 1.0. W3C Candidate Recommendation. Retrieved March 23, 2002, from <http://www.w3.org/TR/2000/CR-rdf-schema-20000327>
- [9] Brooks, C. (2002). A discussion of XML based middleware for web services. Retrieved April 2, 2002, from <http://www.cs.usask.ca/~cab938/papers/898finalpaper.doc>
- [10] Bull, S., Greer, J., McCalla, M., Kettel, L. (2001). Help-Seeking in an Asynchronous Help Forum. In Proceedings of Workshop on Help Provision and Help Seeking in Interactive Learning Environments, International Conference on Artificial Intelligence in Education.
- [11] Chopra, V., et al. Professional XML Web Services. Wrox Press, 2001. Page 15.
- [12] Chopra, V., et al. Professional XML Web Services. Wrox Press, 2001. Page 185.
- [13] Ehnebuske, D., Rogers, D., von Riegen, C. [Eds.]. UDDI Version 2.0 Data Structure Reference. June 2001, UDDI.org.
- [14] Fielding, R., et al [Eds.]. (1999). Hypertext Transfer Protocol - HTTP/1.1. Retrieved December 13, 2001, from <http://www.ietf.org/rfc/rfc2616.txt>
- [15] Gudgin, M., Hadley, M., Moreau, J., Nielsen, H. [Eds.]. (2001). SOAP Version 1.2 Part 1: Messaging Framework. Retrieved March 23, 2002, from <http://www.w3.org/TR/2001/WD-soap12-part1-20011217/>
- [16] Gudgin, M., Hadley, M., Moreau, J., Nielsen, H. [Eds.]. (2001). SOAP Version 1.2 Part 2: Adjuncts. Retrieved March 23, 2002, from <http://www.w3.org/TR/2001/WD-soap12-part2-20011217/>
- [17] Horstmann, M., Kirtland, M. (1997). DCOM Architecture. Retrieved March 23, 2002, from http://msdn.microsoft.com/library/en-us/dndcom/html/msdn_dcomarch.asp

- [18] Ibbotson, J. [Ed.]. (2001). XML Protocol Usage Scenarios. W3C Working Draft. Retrieved March 23, 2002, from <http://www.w3.org/TR/2001/WD-xmlp-scenarios-20011217/>
- [19] International Business Machines Corporation, Microsoft Corporation. Web Services Framework, in Proceedings of W3C Workshop on Web Services (San Jose, CA, April 2001). Available online at <http://www.w3.org/2001/03/WSWS-popa/paper51>
- [20] Kulchenko, P. (2001). Implementations. Retrieved December 13, 2001, from <http://www.software.org/directory/4/implementations>
- [21] Lassila, O. [Ed.]. (1999). Resource Description Framework (RDF) Model and Syntax Specification. Retrieved March 23, 2002, from <http://www.w3.org/TR/1999/REC-rdf-syntax-19990222>
- [22] Leymann, F. Web Services Flow Language (WSFL 1.0). May 2001, IBM. Available online at <http://www-4.ibm.com/software/solutions/webservices/pdf/WSFL.pdf>
- [23] Meier, J. (2001, September 28). Mozilla doesn't seem to cache tree view icons? Id 19500. Message posted to <http://www.cs.usask.ca/i-help/>
- [24] Microsoft Corporation. (1998). Distributed Component Object Model Protocol version 1.0. Retrieved March 23, 2002, from <http://www.microsoft.com/library/specs/distributedcomponentobjectmodelprotocolcom10.htm>
- [25] Mitra, N. [Ed.]. (2001). SOAP Version 1.2 Part 0: Primer. W3C Working Draft. Retrieved April 2, 2002, from <http://www.w3.org/TR/2001/WD-soap12-part0-20011217/>
- [26] Myerson, J. M. Web Service Architectures. (Chicago, IL, 2001), Tect publishing. Available online at <http://www.webservicesarchitect.com/content/articles/webservicesarchitectures.pdf>
- [27] Object Management Group. (2001). CORBA/IIOP Specification version 2.6. Retrieved March 23, 2002, from http://www.omg.org/technology/documents/formal/corba_iiop.htm
- [28] PocketSOAP COM component, version 1.23. (2002). Retrieved April 2, 2002, from <http://www.pocketsoap.com/pocketsoap/>
- Sharma, R. (2001). Java APIS for XML based RPC. Retrieved December 13, 2001, from <http://jcp.org/jsr/detail/101.jsp>
- [29] Sun Microsystems Inc. (1997). Java Remote Method Invocation Specification. Retrieved March 23, 2002, from <ftp://ftp.javasoft.com/docs/jdk1.1/rmi-spec.pdf>
- [30] Sun Microsystems. (2001). Javabeans Activation Framework. Retrieved December 13, 2001, from <http://java.sun.com/products/javabeans/glasgow/jaf.html>
- [31] Thatte, S. XLANG: Web Services for Business Process Design. 2001, Microsoft Corporation. Available online at http://www.gotdotnet.com/team/xml_wsspecs/xlang-c/default.htm
- [32] Thompson, H., et al [Eds.]. (2001). XML Schema Part 1: Structures. Retrieved December 13, 2001, from <http://www.w3c.org/TR/xmlschema-1/>
- [33] University of Saskatchewan Advanced Research in Intelligent Educational Systems I-Help Project. Retrieved March 28, 2002, from http://www.cs.usask.ca/research/research_groups/aries/iHelp2.html
- [34] Williams, S., Jones, M. [Ed.]. (2001). XML Protocol Abstract Model. W3C Working Draft. Retrieved March 23, 2002, from <http://www.w3.org/TR/2001/WD-xmlp-am-20010709/>
- [35] Winer, D. (2001). Implementations. Retrieved December 13, 2001, from <http://www.xmlrpc.com/directory/1568/implementations>
- [36] Winer, Dave. (1999). XML-RPC Specification. Retrieved December 13, 2001, from <http://www.xmlrpc.com/spec>
- [37] World Wide Web Consortium Semantic Web Activity. (2000). <http://www.w3.org/2001/sw/>
- [38] World Wide Web Consortium Web Services Activity. (2000). <http://www.w3c.org/2002/ws/>
- [39] World Wide Web Consortium Web Services Architecture Group. (2002). <http://www.w3c.org/2002/ws/arch/>