Efficient SOAP Binding for Mobile Web Services

Kwong Yuen Lai Thi Khoi Anh Phan Zahir Tari
RMIT University, School of Computer Science and Information Technology
GPO Box 3476V, Melbourne, VIC 3001, Australia
{kwonlai, thikhoi, zahirt}@cs.rmit.edu.au

Abstract

Existing web services rely on HTTP and TCP as the underlying transport protocols for SOAP messaging. While these protocols provide a number of benefits, including being able to pass through firewalls and are universally supported across different platforms, they were designed for wired networks with high bandwidth, low latency and low error rate transmissions. Due to the variability of wireless channels however, these assumptions do not hold in wireless environments. In this paper, we investigate the performance of HTTP and TCP as transport protocols for SOAP in wireless environments. Through extensive testing, we show that SOAP-over-HTTP and SOAP-over-TCP are inefficient and lead to high latency and transmission overhead for wireless applications. To overcome these limitations, we study the use of UDP as a binding protocol for SOAP. The results obtained are promising and show that SOAP-over-UDP provides throughput that is ten times higher compared to SOAP-over-HTTP in a wireless setting. Furthermore, using UDP to transport SOAP messages reduces transmission overhead by more than 50% compared to SOAP-over-HTTP. Finally, to illustrate where UDP binding can be useful, example applications are also described in this paper.

1. Introduction

Web services are software components that can be accessed through standards-based protocols in a distributed network. Over the past few years, we have seen an increasing number of web services appearing on the Internet. The XML-based Simple Object Access Protocol (SOAP) has been a vital component that fuelled part of this success. It provides a standardized mechanism for heterogeneous information systems and applications to communicate with each other regardless of their implementation language and operating platform.

In order to send a SOAP message, a SOAP binding needs to be specified. A SOAP binding describes how an underlying protocol is used to transport SOAP messages. The W3C SOAP specification[1] does not restrict the binding of SOAP to any particular protocol. Most of today's web services use SOAP over HTTP. There are a number of reasons why HTTP is an attractive binding option. First, HTTP is already widely used on the Internet and universally supported by web servers. This provide a strong base for the adoption of the SOAP over HTTP binding. Second, most firewalls allow HTTP packets to pass through and processed securely, therefore removing the need and security risk of opening up a different port to accept SOAP messages. Third, HTTP uses TCP/IP as its underlying transport which ensures packets are reliably delivered.

Despite these benefits, SOAP-over-HTTP is not a universal solution that is appropriate for all applications and environments. In recent years, the success of web services and the advances in mobile computing technologies have led to the emergence of mobile web services. The limitations of TCP/IP (which is used by HTTP) in wireless environments are well documented [16]. The flow control mechanism used by TCP considers packet loss as an indication of congestion. When a packet loss is detected, TCP reduces the transmission rate of the sender to avoid congesting the network. While this mechanism works well in wired networks where communication channels are reliable (i.e. packet losses are mainly the result of congestion instead of transmission errors), the same does not apply in wireless environments where connectivity is highly volatile and susceptible to interference and limitations in network coverage.

Another problem of using HTTP and TCP as the underlying transport protocols for SOAP is the high overhead associated with managing connections. In order to send a SOAP message over HTTP, a TCP connection must be established through a three-way handshake. Once the SOAP message is sent, the connection is closed through another handshaking process. The establishing and closing of connections increases transmission cost and lead to longer delays. This overhead may not be justifiable for applications

1 http://www.w3.org/TR/soap

Proceedings of the IEEE Conference on Local Computer Networks 30th Anniversary (LCN'05)
0-7695-2421-4/05 $20.00 © 2005 IEEE

IEEE COMPUTER SOCIETY

that only need to send small amount of data at a time. This is especially the case in wireless environments where limited bandwidth is available and round trip time (RTT) is high.

Finally, the messaging model used by HTTP makes it easy for applications to correlate requests with responses. However, this request-response messaging model may not be appropriate for some applications. For example, a weather monitoring application that periodically sends observation data to a processing center may not need to receive any response from the server. Similarly, many applications do not require the guaranteed delivery provided by TCP. In these cases, an alternative transport protocol other than HTTP and TCP with lower overhead may be more suitable.

Recently, a SOAP-over-UDP specification was proposed by developers from BEA, Lexmark, Microsoft and Ricoh[3]. UDP is a simple, low-overhead transport protocol. Unlike TCP, it does not provide any flow-control mechanism and only guarantees best-effort delivery of packets. Packets delivered by UDP may be duplicated, arrive out of sequence or even not reach their destination at all. However, due to its simplicity, UDP provides a number of benefits over TCP:

- UDP does not require a connection to be established to send a packet. Each UDP datagram carries its own destination address and is routed independently of other packets. This reduces the setup time associated with sending a message.
- UDP packets are smaller compared to TCP packets. The UDP header is only eight bytes in length, in comparison to the TCP header which is over 20 bytes in length. For wireless environments where bandwidth availability is low and users are charged by the amount of data they transmit and received, UDP provides a cheaper alternative to TCP.
- UDP supports multicasting. This opens up the opportunity to create push-based and publish/subscribe web services where SOAP messages or notifications are sent to multiple clients periodically or based on certain triggering event.

This paper aims to provide a detailed comparison between three different SOAP binding options, including SOAP over HTTP (with TCP as transport), SOAP over TCP and SOAP over UDP. Such a comparison is important because the results obtained will highlight the strengths and weaknesses of each binding option. With this knowledge, we are able to determine the most suitable binding to use for different types of usage scenarios.

The rest of this paper is organised as follows. Section 2 provides background information on SOAP and the SOAP binding framework. A survey of related work on SOAP performances is presented in Section 3. Section 4 describes the SOAP-over-UDP binding and compare this to SOAP-over-HTTP and SOAP-over-TCP. This is followed by a description of the experiment setup and a detailed discussion of the results collected in Section 5. In Section 6, we describe applications where each type of SOAP binding is most suitable. Finally, the paper concludes in Section 7 where future work are also identified.

2. Background

In this section, background information on SOAP, SOAP-over-HTTP, SOAP-over-TCP and the SOAP binding framework are presented.

2.1. SOAP

Web services are an emerging technology, allowing interoperable communication between applications on different platforms. The deployment of web services in mobile environment, which is called mobile web services, is also increasing rapidly. However, due to the nature of mobile wireless networks, existing middleware infrastructure cannot efficiently support web services on mobile devices.

Simple Object Access Protocol (SOAP) is a standard for web services transportation. SOAP is designed with the aim of replacing traditional remote communication methods such as DCOM, CORBA and RMI. The main benefit of SOAP is interoperability. It allows applications written in different languages and deployed on different platforms to communicate with each other over the network. Furthermore, SOAP can be carried on top of any underlying protocols (such as HTTP, TCP, UDP, BEEP and SMTP). Thus, SOAP creators have defined a binding framework for SOAP instead of a fixed binding. Specifically, the SOAP binding framework specification [9] provides a high level of flexibility in terms of how SOAP messages are transmitted.

2.2. SOAP Binding

A SOAP binding is the boundary between SOAP and the underlying protocol. Such binding has to specify the syntactic and semantic rules for passing SOAP messages between SOAP and the underlying protocol.

2.2.1. SOAP-over-HTTP Over the Internet, HTTP is the protocol that is most widely used for SOAP binding. SOAP-over-HTTP is also the only concrete binding specification defined in the SOAP binding framework proposal. This is because HTTP is one of the core protocols of the Internet and is universally supported by web servers. It also has the benefit of being able to pass through firewalls, which makes it a convenient candidate for transporting SOAP.

A SOAP message can be transported using HTTP by encapsulating a SOAP request into the message body of a

IEEE COMPUTER SOCIETY

```
POST /axis/servlet/AxisServlet HTTP/1.0
Content-Type" text/xml; charset=utf-8
Accept: application/soap+xml
User-Agent: Axis/1.2RC3
Host: http://www.weatherhost.com:8081
Cache-Control: no-cache
Pragma: no-cache
SOAPAction: ""
Content-Length: 438
Authorization: Basic dXN1cjE6cGFzczE=

<?xml version="1.0" encoding="UTF-8"?>
  <soapenv:Envelope: xmlns:soapenv="..."
                     xmlns:xsd="..."
                     xmlns:xsi="...">
    <soapenv:Body>
      <ns1:getTemperature
        soapenv:encodingStyle="..."
        xmlns:ns1="...">
        <symbol xsi:type="xsd:string">
          Melbourne
        </symbol>
      </ns1:getTemperature>
    </soapenv:Body>
  </soapenv:Envelope>
```

Figure 1. HTTP SOAP Request

```
HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Content-Type: text/xml;charset=utf-8
Date: Sat, 1 Jan 2005 00:00:00 GMT
Connection: close

<?xml version="1.0" encoding="UTF-8"?>
  <soapenv:Envelope: xmlns:soapenv="..."
                     xmlns:xsd="..."
                     xmlns:xsi="...">
    <soapenv:Body>
      <ns1:getTemperatureResponse
        soapenv:encodingStyle="..."
        xmlns:ns1="...">
        <getTemperatureReturn href="#id0"/>
      </ns1:getTemperatureResponse>

      <multiRef id="id0" soapenc:root="0"
        soapenv:encodingStyle="..."
        xsi:type="xsd:float"
        xmlns:soapenc="...">
        23.5
      </multiRef>
    </soapenv:Body>
  </soapenv:Envelope>
```

Figure 2. HTTP SOAP Response

HTTP GET or HTTP POST. Similarly, a SOAP response can be encapsulated into the body of a HTTP response. Figure 1 and Figure 2 illustrate a SOAP request using HTTP POST and the corresponding response.

2.2.2. SOAP-over-TCP Similar to SOAP-over-HTTP, in order to transport a SOAP message using TCP as a direct underlying protocol, a SOAP message must be contained into the data octets part of a TCP packet. There is not yet an official specification for SOAP binding with TCP, however, Apache Axis and Microsoft WSE (Web Service Enhancement) 2.0 already include APIs that enables the sending of SOAP messages via TCP channel [1].

2.2.3. SOAP-over-SMTP Apart from the HTTP binding, the SOAP-over-email binding is also presented in the W3C specification. However, unlike the HTTP binding, which forms part of the SOAP standard, the SOAP-over-email binding is only presented in the specification as an example to demonstrate the realisation of the SOAP binding framework. In the SOAP-over-email binding, SOAP messages are encapsulated in the bodies of emails. this allows asynchronous message exchange between web services.

3. Related work

This paper focuses on the performance of different SOAP bindings in wireless environment. In this section, existing work on SOAP performance and web services in wireless environment are discussed.

Earlier authors have looked at SOAP performance from different perspectives. A number of work compared SOAP with other traditional distributed technologies (e.g. [6, 11] and [12]) while others evaluated the performance of different SOAP implementations [14]. From these previous studies, it is noted that the current implementations of SOAP using HTTP as a transport protocol is slower than existing middleware technologies and other communication protocols such as CORBR and Java RMI.

In [4], the limitations of SOAP for scientific computing are investigated. Their experiments compared SOAP with Java RMI by sending large arrays of doubles; the results showed that SOAP is a factor of ten slower than Java RMI. Several techniques are proposed in this work to improve SOAP performance, specifically targeted at systems that require high performance for distributed scientific computing.

Gryazin and Seppala compared SOAP and CORBA for mobile devices and found that the performance of web services does not depend on the implementation of SOAP [12]. They also concluded that SOAP is more suitable for larger scale systems with non-critical architectures while CORBA is more robust and flexible for mobile applications.

An experimental evaluation of SOAP performance in business applications has been presented in [13]. In this work, SOAP is compared with Financial Information eXchange (FIX) protocol which also uses a text based wire representation as SOAP, and with CDR, a common binary wire format. The results demonstrated that it is possible for text-based protocols to achieve equivalent performance to binary protocols, but the text-based nature of XML is not sufficient to explain SOAPs efficiency.

Other projects have been working to improve SOAP per-

IEEE COMPUTER SOCIETY

formance by compression and caching mechanisms (e.g. [5, 6, 7] and [15]). It is shown in [15] that both clients and servers in poorly connected networks and clients with resource-constrained devices can benefit from compression. Devaram and Andresen [7] showed that an increase of 800% in performance can be achieved by caching SOAP payloads at the client side.

Interest in the usage of web services in mobile business applications is increasing. In a recent paper, Gebauer and Shaw suggested the deployment of web services in mobile environment will contribute to the success of mobile commerce [10]. Sun Microsystems has also released the specification for J2ME Web services to provide the capabilities of accessing remote XML-based Web services and parsing XML data to the J2ME platform. J2ME with kVM (kilo Virtual Machine) is a stripped down version of the Java Virtual Machine designed to have a small memory footprint between 40 and 80 kilobytes. The J2ME Web service specification provides standards and programming models for the Java community to develop web service applications for embedded devices.

kSOAP [8] is a SOAP API which provides an open source and small footprint implementation of XML aimed at developing applications for mobile devices using J2ME. In [2], the performance of Web services on J2ME and kSOAP clients is measured on a PDA and an IBM Thinkpad laptop. These clients access to a temperature service and a translation service which are available from the Internet. Their results showed that the main bottlenecks are the low available bandwidth causing large transmission time and the high cost associated with XML parsing.

Our evaluation study differs from others in the way that we provide a benchmark for various SOAP binding options in a wireless environment. Our results will be useful for other researchers to examine the effectiveness in performance of these transport protocols to the development of web service in wireless networks. From the results, we argue that SOAP-over-UDP provides certain performance benefits over the traditional SOAP-over-HTTP binding.

4. SOAP-over-UDP

Although HTTP is widely adopted as the de-facto binding option for SOAP messaging in existing web services, it is inefficient when supporting web services in wireless environments. HTTP uses TCP as the underlying transport, which performs poorly in wireless environments due to a number of limitations. In particular:

- The congestion avoidance mechanism in TCP assumes packet loss are always due to congestion. However, in a wireless network, packet losses due to disconnections and transmission errors are common.
- TCP requires a connection to be established before any useful data can be transmitted. Furthermore, as data is received, acknowledgement packets are sent. These lead to additional overhead which may not be justifiable where bandwidth and client energy are limited and reliable transmission of packets is not required.
- SOAP messages that carry only a small amount of data can finish transmitting while the TCP connection is still in its slow start phase. This results in poor utilisation of the available bandwidth. The problem is particularly severe in wireless environments due to high round trip time.

To deal with these problems, we investigate SOAP-over-UDP as an alternative binding for SOAP messaging. UDP is light-weight, unreliable and connectionless. Due to its simplicity, UDP requires little overhead and offers fast packet delivery. These characteristics mean that UDP is suitable for applications where high transfer rate is valued over reliability (e.g. real-time multimedia streaming). UDP is also well suited for quick exchange of information that are small in size. An example of this is the domain name service (DNS), which uses UDP datagrams to transmit domain name lookup requests.

The recently released SOAP-over-UDP specification defines that a SOAP message has to be encapsulated into the data octets part of a single UDP packet (as shown from Figure 1). The SOAP message must be small enough to fit within a single datagram which is less than 65,536 (2^{16}) bytes in size [3]. The specification supports four messaging patterns as follows:

- Unicast one-way,
- Multicast one-way,
- Unicast request, unicast response and
- Multicast request, unicast response.

5. Performance Evaluation

This section presents the experiment setup and the results obtained. The aim of this study is to understand the performance tradeoffs of selecting different transport protocols for SOAP. Three SOAP bindings are studied, these are SOAP over HTTP, SOAP over TCP and SOAP over UDP.

To evaluate the performance of the various bindings, we have designed a weather application that provides weather information based on requests. The application is hosted on a server as a web service and clients access the service via SOAP requests.

Clients can send two types of requests to the server:

- **Simple Request** (SRqt): A request that provides the server with a location and expects the server to reply with the temperature of that particular location.

IEEE COMPUTER SOCIETY

- **Detailed Request** (DRqt): A request that provides the server with a location and expects the server to reply with a full weather report for that location.

The server response with the following messages:

- **Simple Response** (SRsp): A reply to a simple request. Contains a single float value representing the temperature of the specified location.
- **Detailed Response** (DRsp): A reply to a detailed request. Contains a textual description (string) of the weather condition.

Two laptops are used for the experiments. The first (laptop 1) is a Toshiba laptop with a 1.5GHz Pentium-M processor and 512Mb of RAM. The second (laptop 2) is a LG laptop with a 1.6GHz Pentium-M processor and 768Mb of RAM. Windows XP professional is used as their operating systems. Both machines are equipped with 802.11b Wi-Fi interfaces and are connected to each other through a Netgear DG834G wireless router. The experiments are tested under two modes:

- **Loopback**: The server and the client both run on laptop 1. Connection is via the loopback (localhost) interface, therefore no packet loss or bandwidth limits apply to this mode.
- **WLAN**: Laptop 1 hosts the server and laptop 2 runs the client application that accesses the web service. The two laptops are connected via the wireless router. They are the only machines connected to the router at the time of the test. A series of pings from laptop 2 shows the average round trip time of 27ms to laptop 1.

The RC3 release of Apache's AXIS 1.2 (Java) is used for the SOAP implementation. Tomcat 5.5 is used to host the web service. Both client and server are written in Java using JDK 5.0.

5.1. Transmission Overhead

Figure 3 shows the size of the different messages in bytes. We can see that for a simple request (SRqt), SOAP-over-HTTP requires more than twice the number of bytes compared to SOAP-over-UDP. In a simple request, the client only need to provide the server with a location parameter (type String), however the HTTP and TCP headers add significant amount of overhead on top of the SOAP request. UDP also has some header overhead, however this is restricted to 8 bytes. Similar trends can be seen for simple responses (SRsp), detailed requests (DRqt) and detailed responses (DRsp). The size of detailed responses are significant larger than the other message types because it contains the weather report payload generated by the server which must be communicated to the client. For messages carrying only a small number of parameters (e.g. SRqt and DRqt), using UDP can reduce message size by over 50% compared to HTTP. This is significant because if a mobile client is to repeatedly send simple SOAP requests to a server, such reduction in message size leads to huge saving in transmission cost and client energy consumption.

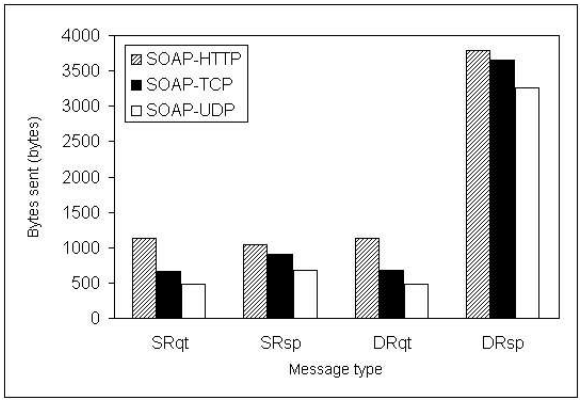

Figure 3. Bytes sent for different message types

Figure 4 shows a breakdown of the message overhead of using HTTP, TCP and UDP binding for the different message types. The packet overhead represents the SOAP payload and the protocol header. For example, SOAP-over-HTTP messages contains a HTTP header, a TCP header and the actual SOAP message. It can be seen that TCP and UDP have very similar packet overheads, however the connection overhead of using SOAP-over-TCP makes TCP more expensive than UDP. In the case of request messages (SRqt and DRqt), the connection overhead is a result of the connection establishment handshake consisting of three separate packets (SYN, SYN-ACK and ACK). Once the server has responded to the requests, four more packets are sent to tear down the connection. These packets are the cause of the connection overhead of SRsp and DRsp. The connection overhead of DRsp is further increased as the SOAP response for a detailed request does not fit into a single TCP packet. As a result, it is split over multiple packets, which introduces TCP acknowledgement packets into the message exchange, thus additional overheads.

5.2. Experiments performed using loopback connection

Next, we study the impact of using different binding options on execution time and throughput. Figure 5 shows the time it takes for client to execute 100 requests over the loopback connection. For both simple requests and detailed requests, SOAP-over-UDP is significantly faster than SOAP-

IEEE COMPUTER SOCIETY

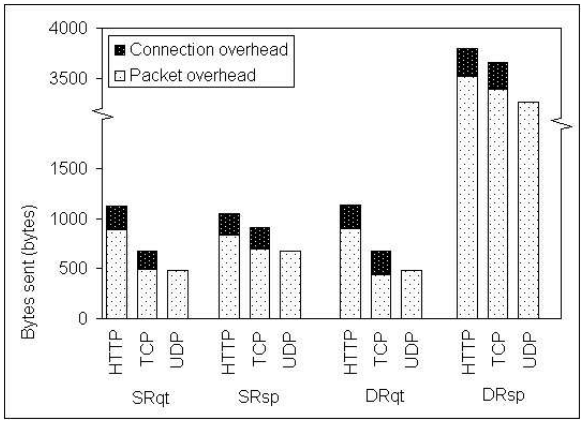

Figure 4. Connection overhead vs. Packet overhead for different binding

over-TCP and SOAP-over-HTTP. The poor performance of TCP and HTTP are mainly due to the additional cost in encoding and decoding the HTTP and TCP headers and also time spent on connection establishment. SOAP-over-TCP is around 17% faster than SOAP-over-HTTP, while SOAP-over-UDP is roughly 30% faster than SOAP-over-HTTP.

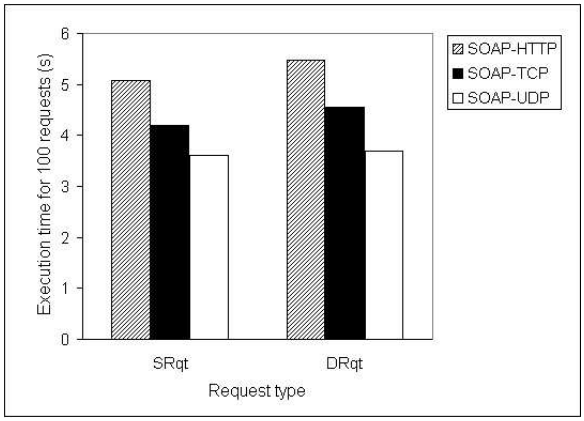

Figure 5. Execution time for 100 requests (loopback)

Another interesting performance metric is throughput. It shows the scalability of the binding options tested. Figure 6 plots the throughput (number of messages processed per second) under the loopback connection scenario. It is found that SOAP-over-HTTP and SOAP-over-TCP have roughly the same throughput, 28.6 requests per second for HTTP and 30 requests per second for TCP. The throughput of SOAP-over-UDP is significantly higher (over 40 requests per second). Using the UDP binding, SOAP messages can be sent without setting up a connection, thus minimal system resources are held up. Furthermore, the simplicity of the UDP protocol means the server can process UDP headers faster than TCP and HTTP headers.

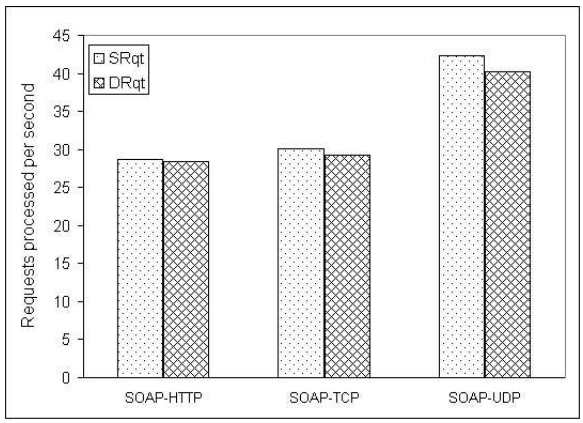

Figure 6. Throughput (loopback)

5.3. Experiments performed over Wi-Fi connection

The execution time and throughput experiments are repeated over a 802.11b Wi-Fi connection using laptop 1 and laptop 2. The laptops are connected to the router at 11Mbps.

The execution time of 100 requests is plotted in Figure 7. We found a significant difference between the result in this graph compared to those collected from the loopback experiments. For SOAP-over-HTTP and SOAP-over-TCP, the total execution time over the wireless network is over 100 times higher than over loopback. The difference in the UDP results is less, where total execution time over wireless is around 15 times higher than over loopback. The results show that SOAP-over-UDP is much faster compared to HTTP and TCP. This is due to a number of reasons. First, connection establishment, packet acknowledgement and connection tear-down means a higher number of packets are sent when the HTTP and TCP bindings are used. Due to the high round trip time in the wireless network, the high number of packets exchanged lead to an increase in execution time. Second, unlike the loopback connection which is reliable, the wireless links are susceptible to packet losses. When packet loss occurs, the retransmission mechanism in TCP is triggered, leading to an increase in execution time. Furthermore, when an acknowledgement packet is lost, the TCP client must wait out the TCP retransmission timeout (RTO), this lead to further increase in execution time. On the other hand, UDP does

IEEE COMPUTER SOCIETY

not spend any time establishing and tearing down connections, therefore a large number of packets can be transmitted quickly. Of course, due to the unreliability of the wireless links, some packet losses occurred. However, for applications with high performance and low reliability requirements, SOAP-over-UDP provides significantly better performance than SOAP-over-HTTP and SOAP-over-TCP. The results show that SOAP-over-UDP is approximately 10 times faster than HTTP, while TCP achieved similar performance to HTTP.

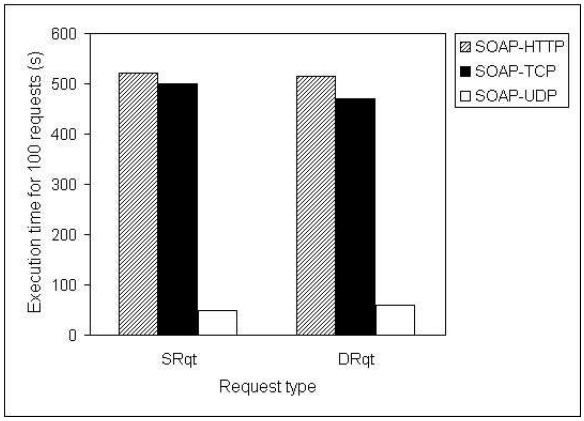

Figure 7. Execution time for 100 requests (WLAN)

The throughput of the different binding over the Wi-Fi network is shown in Figure 8. As expected from the execution time results, SOAP-over-HTTP and SOAP-over-TCP provide significantly lower throughput compared to SOAP-over-UDP. For simple requests, SOAP-over-UDP achieves throughput nearly 11 times higher than HTTP, while for detailed requests, the throughput under the UDP binding is 8.5 times higher. The performance difference is less for detailed requests because TCP and HTTP performs better when transmitting larger amount of data. The reason for this is that for large data transfers, the connection management overhead of TCP and HTTP become less significant compared to the transmission cost of the actual payload.

6. Applications

This section gives examples of different applications that are suitable for the various SOAP binding.

6.1. SOAP-over-HTTP

The main advantage of the HTTP binding is that it allows SOAP messages to pass through firewalls and be processed by web servers. However, due to the size of the HTTP header and the use of TCP as the underlying transport protocol, SOAP-over-HTTP incurs relatively high overhead when it is used to send simple SOAP messages. Some examples of where SOAP-over-HTTP is suitable include:

- *Translation service* - where clients provide text in one language and the web service performs translation to another language and send it back to the client.
- *Payment service* - where the web service collects sensitive credit card or banking information from clients, process the transaction and reply with a receipt/confirmation. HTTPS can be used instead of HTTP to provide security.

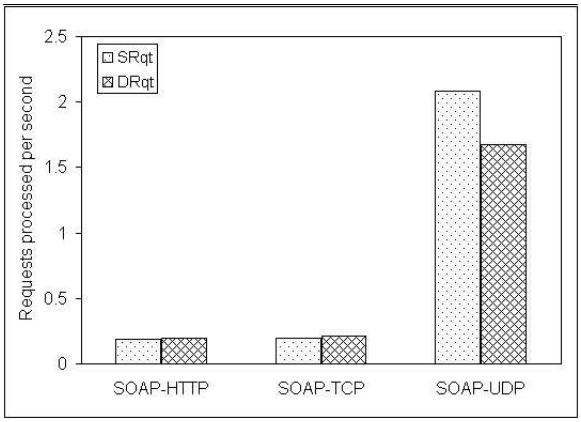

Figure 8. Throughput (WLAN)

6.2. SOAP-over-TCP

Our results show that SOAP-over-TCP does not provide much benefits over SOAP-over-HTTP in terms of throughput and execution time. However, the packet size of a simple SOAP request with one parameter using TCP is around 40% smaller than the equivalent request using the HTTP binding. As a result, TCP can be used for applications where clients send short requests to the server to request large amount of data, and the data needs to be transported reliably. An example of such an application may be:

- *Scientific data transfer* - where clients sent requests to retrieve large scientific data collections (e.g. planetary observations or genebank DNA sequences).

6.3. SOAP-over-UDP

The results obtained in Section 5 show that SOAP-over-UDP provides significantly higher throughput compared to SOAP-over-HTTP and SOAP-over-TCP. Additionally, it reduces the amount of bytes required to deliver SOAP messages. As a result of this, SOAP-over-UDP is most suitable

IEEE COMPUTER SOCIETY

for applications where short SOAP messages are sent frequently and reliability is not of concern. In this type of applications, the small amount of data to be delivered does not justify the overhead of establishing a connection, thus making the simple UDP transport appropriate. Secondly, as UDP supports multicasting and broadcasting, it can be used in push-based web services. Some example applications where SOAP-over-UDP is suitable include:

- *Postcode lookup* - where clients provide a location and the web service reply with the corresponding postcode or vice versa.
- *Stock quote push* - where the server broadcast stock prices to a large number of mobile clients. The frequency of such push could vary from very high to very low as it depends on the movement of stock value.

7. Conclusion

Advances in wireless technologies and the increasing popularity of web services have lead to the emergence of mobile web services. As the main communication protocol for web services, SOAP plays a very important role in this context. Existing work have studied the performance of different SOAP implementations under different environments. However, the effect of using different SOAP binding have not been studied. The SOAP specification provides an open binding framework that allows SOAP messages to be transported using any underlying protocol. In this paper, we studied the performance of SOAP-over-HTTP, SOAP-over-TCP and SOAP-over-UDP. We argue that while HTTP and TCP have many benefits including the ability to pass through firewalls and reliability, these features may not be needed by all applications. HTTP and TCP also exhibit many limitations and performs poorly in wireless environments due to the slow start, connection establishment and packet acknowledgement. Through an extensive set of experiments, we show that using HTTP or TCP binding for SOAP lead to significantly higher overhead compared to SOAP-over-UDP. Furthermore, the throughput of SOAP-over-UDP is approximate 10 times higher than SOAP-over-HTTP. To illustrate where the various bindings are useful, we also describe a number of applications suitable for each binding. As part of our further work, we intend to study the use of reliable UDP to transport SOAP messages, and compare its performance to the unreliable UDP binding.

Acknowledgment

This project is support by the ARC (Australian Research Council - under the Linkage project scheme, no. LP0218853) and SUN Microsystems grant no.7832-030217-AIS.

References

[1] Apache Software Foundation. Web services - Axis. *http://ws.apache.org/axis/*, 2005.

[2] V. Bansal and A. Dalton. A performance analysis of web services on wireless PDA. *http://www.cs.duke.edu/ vkb/advnw/ project/PDAWebServices.pdf*, 2002.

[3] BEA Systems Inc., Lexmark, Microsoft Corporation Inc., and Ricoh. SOAP-over-UDP specification. *http://ftpna2. bea.com/pub/downloads/SOAP-over-UDP.pdf*, 2004.

[4] L. Chen and R. Nath. A framework for mobile business applications. *International Journal of Mobile Communications,*, 2(4):368–381, 2004.

[5] K. Chiu, M. Govindaraju, and R. Bramley. Investigating the limits of SOAP performance for scientific computing. In *Proceedings of the 11th IEEE International Symposium on High Performance Distributed Computing (HPDC)*, pages 246–254, 2002.

[6] D. Davis and M. Parashar. Latency performance of SOAP implementations. In *Proceedings of the 2nd IEEE/ACM International Symposium on Cluster Computing and the Grid*, pages 407–412, 2002.

[7] K. Devaram and D. Andresen. SOAP optimization via client-side caching. In *Proceedings of the First International Conference on Web Services (ICWS'03)*, pages 520–524, 2002.

[8] Enhydra: An open source SOAP for the kVM. *http://ksoap.objectweb.org*, 2003.

[9] C. Ferris and S. Williams. SOAP underlying protocol binding. *http://www.w3.org/2000/xp/Group/1/10/12/ Binding_Framework_Proposal*, 2001.

[10] J. Gebauer and M. Shaw. Success factors and impacts of mobile business applications : Results from a mobile e-procurement study. *International Journal of Electronic Commerce*, pages 19–41, Spring 2004.

[11] G. Gehlen and R. Bergs. Performance of mobile web service access using the wireless application protocol (WAP). In *Proceedings of the World Wireless Congress*, 2004.

[12] E. Gryazin and O. Seppala. SOAP and CORBA productivity comparison for resource-limited mobile devices. In *Proceedings of the IASTED International Conference Software Engineering*, pages 707–712, 2004.

[13] C. Kohlhoff and R. Steele. Evaluating SOAP of high performance business applications: Real-time trading systems. In *Proceedings of the Twelveth International World Wide Web Conference (WWW'03)*, pages 872–874, 2003.

[14] A. Ng, S. Chen, and P. Greenfield. An evaluation of contemporary commercial SOAP implementations. In *Technical Report, Department of Computing, Macquarie University, NSW, Australia*, 2004.

[15] M. Tian, T. Voigt, T. Naumowicz, H. Ritter, and J. Schiller. Performance considerations for mobile web services. *Elsevier Computer Communications Journal*, 27(11):1097–1105, July 2004.

[16] G. Xylomenos, G. Polyzos, P. Mahonen, and M. Saaranen. TCP performance issues over wireless links. *IEEE Communications Magazine*, 39(4):52–58, 2001.

IEEE COMPUTER SOCIETY