TELKOMNIKA, Vol.9, No.3, December 2011, pp. 503~514
ISSN: 1693-6930
accredited by DGHE (DIKTI), Decree No: 51/Dikti/Kep/2010

Performance Comparison of Enterprise Applications on Mobile Operating Systems

Adrian Mullally[1], Nigel McKelvey[1], Kevin Curran*[2]
[1]Department of Computing, Letterkenny Institute of Technology, Ireland
[2]Faculty of Computing & Engineering, University of Ulster, Derry, Northern Ireland
e-mail: kj.curran@ulster.ac.uk

Abstract

Seiring kemajuan dalam teknologi ponsel, Smartphone memiliki kemampuan untuk mengakses layanan web dalam aplikasi pengguna berinteraksi untuk mengambil dan mengirim informasi dari dan ke layanan web. Karenanya, Smartphone semakin meningkat popularitasnya. Pemanfaatan layanan web baru dan layanan web yang telah ada juga akan tumbuh pesat. Makalah ini memberikan perbandingan dari dua jenis utama dari layanan web: SOAP dan REST. Kinerja penggunaaan SOAP dan REST pada smartphone juga dievaluasi. Pengujian yang dilakukan meliputi waktu yang dibutuhkan untuk melakukan panggilan layanan web, RAM digunakan membuat panggilan layanan web, dan data yang dikirim dan diterima ketika membuat panggilan layanan web.

Keywords: *layanan web, SOAP, REST, kinerja perangkat lunak, aplikasi enterprise*

Abstract

Due to the advances in mobile phone technology, Smartphones have the ability to access web services within applications the user interacts with to retrieve and send information from and to web services. As Smartphones have grown in popularity, the utilisation of both new web services and web services already in place will also grow. This paper provides a comparison of the two main types of web services, SOAP and REST. We evaluate the performance of using SOAP and REST on a smartphone. The tests performed include the time taken to perform a web service call, the RAM used making a web service call and the data transmitted and received while making a web service call.

Keywords: *enterprise applications, REST, SOAP, software performance, web service*

1. Introduction

The method of interaction between a user and the internet is comprised of more than HTML and JavaScript web pages. There are many background web services in use which provide a new level of interactivity between the user and a web site. These web services allow a business to operate highly functional applications, between different platforms, without the need for a server for each independent platform. There are many different types of web services, using any number of programming languages to perform their functions, which are accessible over the internet. Mobile operating systems have increased in complexity in recent years, growing smarter and faster. Now, mobile devices are closing the gap on PCs in terms of processing power, processing speed, RAM. They possess the ability to connect to the internet, and the operating systems are able to run apps. One such method of calling services remotely is through web services.

This paper focuses on consuming web services on mobile operating systems and evaluating and the viability of doing so. It compares the two main web service protocols, SOAP and REST. These comparisons were performed on two operating systems; Android and Windows Phone. These operating systems are popular mobile operating systems for smartphones. Windows Phone was released in September 2010 [1] and Android version 2.3 was released in December 2010 [2]. Testing both REST and SOAP on these limited operating systems can show how the operating systems can be improved in calling web services, and how well they can progress in the future, in terms of memory usage, time taken to complete the web service call and the amount of data sent between the phone and the server. The software artefact for the paper included creating a number of web services in which the mobile operating

Received June 17th, 2011; Revised July 25th, 2011; Accepted July 28th, 2011

systems consumed, and a client on each mobile OS which measured the amount of data transferred, RAM used and time taken to complete the request. The languages required were mainly Java and C#, but also knowledge of XML (Extensible Mark-up Language) is required for SOAP and REST.

There are some websites [3, 4, 5] which demonstrate how to create a web service client application for mobile operating systems such as Android and Windows Phone. This paper can therefore be used as a means of demonstrating how to create web service clients on phones, and comparing SOAP and REST on said mobile operating systems. The utilisation of web services is growing, and allowing customers to consume these services through a mobile device allows for greater utilisation of these web services. Therefore the observations outlined here can assist developers in deciding whether to use SOAP or REST web services on a mobile platform.

2. Web Services

Web services are expected to be the key technology in enabling the next instalment of the Web [6]. The evolution of web services has stemmed from distributed computing, which was the older method for information and service distribution overt a network, including the internet. CORBA (Common Object Request Broker Architecture) and RMI (Remote Method Invocation) were two such distributed computing methods. However, CORBA required an IDL (Interface Definition Language) to interface the server and the client and required a steep learning curve [7] and RMI was dependent on Java [8]. Firewalls also needed to be configured to allow specific ports through for both CORBA and RMI communication [9] and in some cases, entire port ranges or protocols must be allowed through for applications to work correctly. The introduction of SOAP and REST as service providers and consumers allowed for distributed computing to advance to a point where platform dependency or intermediary languages were unnecessary. The nature of SOAP and REST allow for multiple platforms, written in multiple languages, to serve and consume web services [10]. They allow a developer to program the client or the server in whichever language they are comfortable with that supports SOAP or REST (e.g. Java, .NET etc.), which allows for a wider consumption of a web service. Web services have become an integral part of both Web 2.0 and cloud computing [11], as they allow for complex data processing and data storage to be outsourced away from the client to a server on the internet, or “in the cloud”. With the sales of smartphones on the rise [12], web services have an important role in providing information to a smartphone user through applications they install. These applications allow a user to post and retrieve information to and from the cloud, without having to open a web browser and navigate to a web page on their smartphone, or on a PC. The tests carried out in this paper can help to decide in what situations SOAP or REST web services should be used [13].

2.1 SOAP

SOAP was created to allow distributed applications to be created easily. SOAP uses WSDL and UDDI, both of which are integral to SOAP, allowing developers to find available web services, and allow their web services to be discoverable over the internet through UDDI, and easily program web service clients using SOAP toolkits and compilers [14] through WSDL. It is these toolkits and compilers that have allowed SOAP to be widely used as a means of developing and consuming web services, as knowledge of the complexities of SOAP are not required for a developer to create a web service, or a web service client. This has led to the creation of enterprise size applications, which can be created using multiple programming languages on both client and server, allowing companies to link and integrate different systems together with relative ease, as well as allowing other users or companies to consume services and access information they can provide over the internet.

Although SOAP is inherently robust, as SOAP applications can be written in many programming languages, served on different operating systems [15], and is relatively easy to implement with IDE toolkits, SOAP messages contain excess information, included with the message payload. It was also originally designed for wired networks [16]. WSOAP (Wireless SOAP) enables the compression of SOAP messages [17]. This would aid in keeping message size to a minimum over a mobile wireless network (e.g. 3G). Although the proposal allowed for between 3x and 12x message compression, a separate server and data compression was required. If WSOAP was implemented, SOAP web services on mobile devices would become more complex in terms of development and resources, as two web services and two servers would be required, one of each for normal SOAP clients, and one of each for mobile SOAP clients. Despite this, the amount of data used by a smartphone client would be far less than the normal data usage for a SOAP request, which would be beneficial to a smartphone user as mobile phone data is billed by the kilobyte. Also, as the message data would be compressed, any encrypted data within the message would also be compressed, and decompressed on either side.

When SOAP was first released, there was no security mentioned within its specification [18], and the only way to secure the SOAP messages was through SSL using HTTP. However in April 2002, IBM, Microsoft, and VeriSign designed a specification for securing SOAP messages called WS-Security [19], which addressed the security issues of SOAP message exchange by defining a SOAP header element that carried security related data [20]. WS-Policy was added to help with security and when used correctly with other standards, XML rewriting attacks can be avoided [21]. With these security features built into SOAP, smartphone applications are able to secure their SOAP messages between client and server. This can allow smartphones to use secure transaction services, so long as the web services themselves are SOAP web services. SOAP is a W3C recommendation, which has gained in popularity, and as mobile phones have evolved into smartphones, SOAP web services have the ability to be utilised more as more applications are built consume SOAP web services. As development tools allow for the easy consumption of these web services to smartphones, SOAP can become an easy decision for smartphone developers who are designing web services or consuming them for applications. However smartphone applications may also consume REST web services.

2.2 REST

Representational State Transfer (REST) is a term coined by Roy Fielding. Representational State Transfer is intended to evoke an image of how a well-designed Web application behaves: a network of web pages (a virtual state-machine), where the user progresses through an application by selecting links (state transitions), resulting in the next page (representing the next state of the application) being transferred to the user and rendered for their use. The motivation for REST was to capture the characteristics of the Web which made the Web successful. Subsequently these characteristics are being used to guide the evolution of the Web [22].

When implementing web services, there is always a question of whether the web service should be SOAP based or RESTful based. Recommendations made in [23] state that RESTful web services should be implemented when the web services are stateless. It argues that if the return statements can be cached and if bandwidth is limited, then RESTful web services should be implemented, as they save time and data usage. Further conclusions made in [15] states that REST web services should be implemented for “tactical, ad hoc integration over the Web”, or so long as the enterprise level features of SOAP (e.g. transactions, reliability, message level security etc.) are not required or if a high level of flexibility and control is necessary. However using REST can also present greater development efforts and technical risk. SOAP provides better tool support and programming interface convenience however most require a dependency on open source projects and vendors [15]. The discovery of SOAP web services is accomplished though UDDI (Universal Description Discovery and Integration), which has the advantage of allowing businesses to provide access to their web services by posting the WDSL document for consumption. However, there is no such repository for REST. REST web services can be discovered though URIs and hyperlinks. The disadvantage of this is that RESTful web services are discoverable through many web sites, instead of a number of repositories, which can make them harder to find. The use of the WDSL document also makes creating the client easier. Within the decision of choosing SOAP or REST, there is confusion over the best practices for implementing RESTful web services. There are two implementations of REST called Hi-REST and Lo-REST. Hi-REST states that all four REST verbs should be used and recommends that “nice” URIs and suggests that Plain Old XML be used for formatting the context of messages. However Lo-REST recommends a minimalist approach to accessing RESTful web services. Instead of using all four verbs, only two verbs are used (POST and GET) [15]. Along with the confusion over the best practices of REST, understanding and implementing those principles have proven difficult [24]. It is due to these difficulties that many web systems are implemented without the promised benefits of REST, including caching or stateful interaction [24]. Compared to SOAP, REST does not have any security features built in, other than what is used by HTTP (i.e. SSL), nor does REST natively support encryption or QoS guarantees [25], which can rule out using REST for any type of secure transaction service (e.g. a payment system or any type of system which requires personal information) and would be better suited for SOAP, due to the high level security features and QoS guarantees available to it.

As REST lends itself to the same security methods and vulnerabilities as HTTP, securing REST web services with SSL has the same vulnerabilities as with HTTP. Although SSL can secure the transport layer, it has been shown that if a web service consumes another web service, which uses the data sent by the client, there is no guarantee that the back end web services are also secured [27]. A second known vulnerability is through a MITM (Man-In-The-Middle) attack, in which the end user unknowingly gives the MITM attacker their security credentials, and then the MITM attacker can spoof the victims' credentials to the original server [26]. One such tool capable of implementing this attack is called SSL Strip. The tool, based on the MITM attack, takes advantage of the method for redirecting people from the insecure to the secure version of a web page, allowing the attacker to compromise any information sent between the user and the supposedly secure webpage [27].

Although there is a threat of man-in-the-middle attacks to PC users, smartphone users are at a much higher risk, through Wi-Fi hotspots that provided freely to users in public places. Users can connect to these unsecure spots and their personal information can be transmitted over the public network, allowing malicious users on the network to possibly view private information. This can be a disadvantage to using REST on a smartphone. The use of both SOAP and REST in web sites and PC applications is not new, however they have started to gain popularity (28). Using SOAP and REST in Smartphones is relatively new, and allows a business to gain greater consumption of the information and services they provide. Deciding on which type of web service should be consumed on a smartphone is not easy, as many considerations must be made (e.g. security, ease of development). This paper analysed both SOAP and REST to determine how resources were used by each on a smartphone, helping to make a decision on which to use for different situations.

3. The Mobile Web

There are many cellular phones in the market with the title of smartphone, and this list is growing more with the advancement of technology in general. It is estimated that in 2009, 14% of mobile phone shipments were smartphones however in 2011; smartphone sales are expected to surpass desktop PCs [12]. In 2010, worldwide smartphone sales reached 302.6 million, up around 74% on 2009 and it is expected to reach 500 million in 2012 [29]. As smartphones and their operating systems becomes more main stream, web services have become useful. These smartphones have different methods of connectivity to the internet through both Wi-Fi and mobile data connections (e.g. 3G); making web services accessible from anywhere the smartphone has internet access. Where before, people were required to access a web browser on their PC, smartphones have taken over as a means of communication and information retrieval over the internet [30].

As web services give clients access to the Cloud, smartphones have the ability to take advantage of the services the Cloud can provide. However, there are issues that can hinder the usefulness of accessing cloud services on smartphones [31]. The first issue is the unpredictable connectivity that any GSM or CDMA mobile device has with the internet. As the phone travels from one cell tower to another, the speed and bandwidth can change from 105 Kbps to 7.2 Mbps and its IP address can change. As well as these, the phone can lose connection altogether, making the reliability of the application subject to both the Cloud web service as well as the network connection. The second issue identified is data privacy. Smartphones are normally dedicated devices for a single user [31] and their private data could be stored on the phone. Due to this the ease in which a cloud hosting company can monitor at will, lawfully or unlawfully, the communication and data stored between the user and the cloud as well as information about the phone they are using (e.g. IMEI, IMSI, usage statistics etc.) can be a concern.

3.1 Android

The Android operating system was produced and is supported by the Open Handset Alliance [32, 33]. Since its release in 2007, the Android OS has evolved many times [34]. Currently the latest version of Android is 2.3 (March 2011); with version 3.0 released for tablet devices. Android could become the standard of mobile computing, although it does have competition with iOS amongst others [35]. As the Open Handset Alliance creates and maintains the operating system, handset manufacturers (e.g. HTC, Samsung, LG etc.) are allowed to concentrate more on developing the hardware with little software development [36], and if required, customise the stock operating system (e.g. HTC Sense, Samsung Touch Wiz) to add features or make it more user friendly. Web services are accessible through the Android operating system. REST web services can be consumed through the “org.apache.http” API built into the Android stack. As well as this, there are numerous tutorials online explaining how to program the client on the Android phone to access these web services [37, 38]. However, SOAP is not natively supported by the Android stack. For Android applications which require a connection to a SOAP based web service, a third party library called “kSOAP” is used [35, 39].

Android is defined as a software stack for mobile devices that includes an operating system, middleware and key applications [40]. Android itself uses Linux version 2.6 at its core for services such as security, memory management, process management, network stack and the driver mode. The Dalvik Virtual Machine (VM) is used to run each application running on the Android OS. Although smartphones can consume a web service, [41] has proposed hosting a SOAP Server on an Android device. This can become a very interesting concept, as it can allow for a SOA implementation over mobile networks. One such use for having a smartphone as a web service server could be for location based service (i.e. using the GPS component on the phone to be able to find out where the phone is). There are some barriers need to be solved before smartphones can be used as web servers. One such issue is connectivity over the network, which can be lost at any time.

Android applications have the ability to access and consume web services, both through SOAP and REST. One of the selling points of using Android is allowing applications to make use of elements of other applications. For this reason, an Android application does not have one entry point (i.e. main()), instead it has components that the system and other applications can run as needed. The component the user directly interacts with is the Activities component, which can be compared to the user interface [42]. The Content component allows the initialisation of a program in the background to use a particular part of an application. Hypothetically, an application could have access to a number of web services, both SOAP and REST. If the content component is used, then other programs could have access to these web services, also. An example of this Content component in use could be a social networking application allowing other applications to post a message or a comment to the users’ profile. The Android operating system is an open source project, allowing outside developers to help with the development of the operating system as a whole. Windows phone however, is not open source which could hinder its overall success, as according to [43] “an open, free system (operating system) will normally win against a proprietary, closed one”. If Windows Phone is to be successful, it must provide a feature rich user experience and online services which can combat Android and iPhone and others.

3.2 Windows Phone OS

Windows Phone was released for development in February 2010 and the first phones with the operating system were released in October 2010. Windows Phone is a complete remake of the older Windows Mobile operating system. It is similar to Android as it has access to cloud based services and it has an “App Store” where users can download 3rd party applications for the phone. Microsoft has taken a different approach to Android however. The UI is very different to that of Android or iOS, and they have seemingly aimed at both the corporate market and the gamer market. The Windows phone operating system has been built in such a way that enterprises can create in house applications that access their .NET systems in the same way their desktop applications would. It also uses the same UI and programming techniques as desktop applications, allowing developers to create applications without a steep learning curve. The phone is also aimed at gamers, as the XMA framework is also built into the operating system. As the XNA framework is used in the development of XBOX games, game developers would also be able to develop games without a steep learning curve. The operating system is also tightly integrated with the cloud services provided by Microsoft, allowing consumers to access their "Live" services anywhere they have an internet connection.

Windows Phone is built on the Windows Embedded Compact (also known as Windows CE) operating system, which is used on embedded systems such as phones, tablets, PDAs, industrial devices etc. Mobile phones which support the operating system must have a minimum specification which includes a capacitive touch screen, 256 MB of RAM, 8 GB of flash space and ARMv7 Cortex/Scorpion or better CPU (44). Visual Studio is used for development, which allows .NET developers to create applications without a steep learning curve. However, for those who do not use Visual Studio, or .NET, learning how to program for Windows Phone may be more difficult. There are 4 main components in the Windows Phone Architecture. Cloud services are an important component for information access which can be carried out in the background, without the user needing to interact with the smartphone. These cloud services are hosted on Windows Azure, allowing for a developer to integrate their applications with web services they have designed, or with web services already hosted on the Windows Azure platform with ease. It is also possible to consume services that are not developed in .NET; however Visual Studio does not have the same integration features that a .NET web service provides, especially for REST applications.

Programs for Windows Phone are created using C# and Silverlight or the XNA framework. The latter is used for creating games for Windows Phone, as well as for the Xbox and Windows, and the former for "event-driven" [45] applications. Unlike Android, the Windows Phone stack natively supports the consumption of both SOAP and REST clients. For consuming SOAP web services, a "SoapClient" class is required. For consuming REST web services, a "WebClient" class is required. Visual Studio and Expression Blend are used as development tools for creating Windows Phone applications. If a programmer is familiar with creating .NET applications in C# and Silverlight or WPF, then there is relatively no learning curve at all, as the creation process is the same for both desktop and mobile applications. The main difference arises from the different classes within the SDKs and the different UI modules available for use on the phone. The same methodology is used between desktop and phone applications when integrating with ASP.NET or WCF applications. For ASP.NET SOAP applications, the developer need only import the WSDL and Visual Studio will create the client code for accessing the web service. For REST web services, if the server was created using WCF in .NET, the client could be created by Visual Studio in the same way by importing a .svc file, instead of a WSDL file. If it was not created in .NET, the client could still be created using the "WebClient" class.

3. Evaluation

The implementation of this project was based on a client/server model. The server used the .NET framework, using ASP.NET to process the SOAP requests and WCF for REST requests. The distribution diagram in Figure 1 shows how the client/server system was configured. The diagram shows the two phone clients (Android and Windows Phone), which connect to the web server on a Windows Server 2008 R2 operating system. The web server then connects to a database which stores the test information recorded on the phone. Each phone has two classes, one for testing SOAP based web services (Test SOAP), and the other for REST based web services (Test REST). These were the web service clients which consume the web services found on the web server. Each phone consumed another web service which is available to them. This web service was used to record the test results, and was implemented using SOAP. Many clients can connect to the server, as shown by the many to one relationship (* - 1). This was because the test software could be implemented on many different clients. However, the web server could only connect to one database, as this information needed to be in one place for all the result information to be recorded and digested properly. This was shown through the one to one (1 - 1) relationship between the web server and the database.

The web server used was Microsoft's IIS 7, hosted on Windows Server 2008 R2, as this was the most up to date server operating system available. IIS 7 was used as it is also the most up to date web server for the .NET framework, allowing ASP.NET and WCF web services to be utilised. For the database, Microsoft SQL Server 2008 was used, due to its integration features with IIS. This database stored all the information gathered from the tests carried out on the clients such as Phone (string): Android or Windows; Type (string): SOAP call or REST call; Message (string): Text, Image, and Audio; Data (double): The amount of data sent/received;

Time(double): time taken in seconds from initial request to process response and Memory(int): Amount of memory used to perform request and process response. The server also stored an image file (.jpg) and an audio file (.mp3). These files were accessed by both the SOAP and REST web services. The server contained seven web services in total. There was three RESTful web services, each having access to a file type. There was another three SOAP based web services, which perform the same functions as the RESTful web services. The fourth SOAP based web service was used to record the performance information into the database. Once the client had finished processing the web service result, it recorded all the information required and sent it to the database using a SOAP web service. Communication between the web service which received the recorded data and the database was implemented using Linq [46]. There was also a separate program on the server which read all the results in the database. This allowed the user to quickly get averages for different types of results.

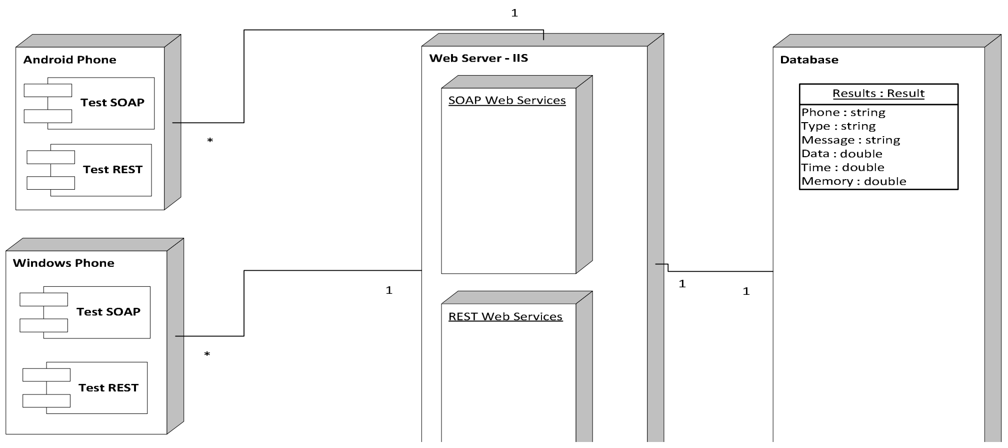

Figure 1. System Distribution Diagram

Twelve types of tests were performed to compare the impact SOAP and REST had on both mobile devices in terms of the amount of data sent to and received by the device while making a web service call, the amount of time taken to complete a web service call and the amount of memory used by the application making the web service call.

3.1 Android

Six types of tests were performed on the Android operating system. The results were recorded in the database and processed. Figure 2 shows the mean memory usage, measured in kilobytes, for the Android application for both the SOAP and REST calls. It measures when a small text message, an image file and an audio file was sent from the server to the client within the web service response. In terms of this test, term “memory” means the amount of RAM the application uses on the device. There are two different implementations of the memory test. This is because of the differences in the APIs between Android and Windows Phone. In Window Phone, the application can retrieve the “Application Peak Memory Usage” can be acquired by a function call. However in Android, a separate class must be created, and using a Linux shell command, retrieve the “VM Peak” memory usage for the process. The peak memory usage was measured as the memory usage of the application differs at different points of the application life cycle. In Windows Phone, the application retrieved the peak memory usage of the application directly after the stopwatch stopped, but before the time is recorded. In Android, a class needed to be created to access the peak memory usage. The class performed a Linux command to collect the memory statistics for the process. The application passes the process ID, which allows the command to identify the process to measure. In this case, the memory is probed after the stop time is recorded. Finally, the amount of data in bytes which is sent and received to and from the phone was measured.

The amount of data used by the applications must be measured also. This is to test if using larger data sizes can cause a larger data overhead between the two web service types, and the two operating systems. Both applications needed to measure the amount of data sent and received by the application. Due to classes not being available in the Windows Phone API, a program called Fiddler was used to track the amount of data send and received by both the Windows Phone and Android operating systems. The amount of data transmitted and received was read by Fiddler and then the test user inputted the total data exchanged between the client and server into the phone before the data was sent to the database web service.

When the memory usage of the application was probed, the REST web service calls used more memory than SOAP and its usage grew with the more data that was received from the server. The SOAP Audio test used on average less memory than both of the image tests, which was unexpected; however the other tests did perform as expected, using more memory the larger the returned value became. The median values for the memory usage tests were for Text SOAP/REST: 10028/13032; Image SOAP/REST: 10744/10782 and for Audio SOAP/REST: 10084/11092. These values are relatively close to the mean values of all the tests carried out and displayed in Figure 2. This means that the test values were spread out evenly above and below the median. This shows that the results are relatively accurate. Figure 3 shows the mean data usage of all the tests carried out on the Android device for both SOAP and REST while each web service was called. Each value within the mean was the total amount of data transmitted and received by the application. The data is measured in bytes.

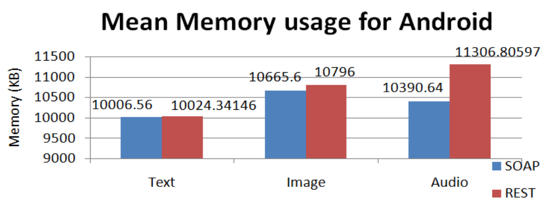

Figure 2. Mean Memory Usage for Android

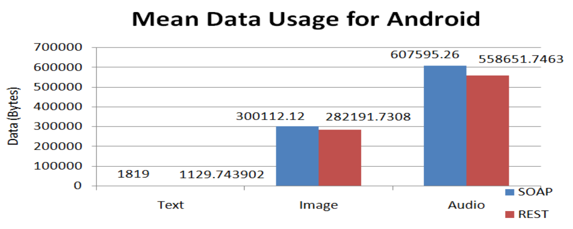

Figure 3. Mean Data Usage for Android

These results show that on average, the data usage for all three tests was higher in the SOAP messages than in the REST messages, which was expected. It is interesting to note that the larger the size of the response size, the larger the gap between the SOAP and REST data values became. The median values for the data tests were Text SOAP/REST: 1819/893, Image SOAP/REST: 299741/281842 and Audio SOAP/REST: 604060/558669. These values are close to the mean values of all the tests carried out and displayed in Figure 3. This means that the test values were spread out evenly above and below the median. This shows that the results are relatively accurate.

Figure 4 shows the mean time taken for all the tests carried out on the Android device for both SOAP and REST while each web service was called. Each value within the mean was the time taken for the web service call to return the response to the client. The time is measured in miliseconds. These results show that as the size of the response data increased, the time taken for the web service response to be received by the client increased also. This was expected. However, the difference in the time taken for the SOAP and REST messages for the image and audio tests was not expected. The median values for the time tests were Text SOAP/REST: 473/299.5, Image SOAP/REST: 4193.5/1534.5 and Audio SOAP/REST: 7288/2590. These values are close to the mean values of all the tests carried out and displayed in Figure 4. This means that the test values were spread out evenly above and below the median. However the SOAP audio mean and median values have a difference of roughly 2900. This can be interpreted as there are a number of tests which took an abnormally long time to complete.

3.2 Windows Phone

Figure 5 shows the mean memory usage, measured in kilobytes, for the Windows Phone application for both the SOAP and REST calls. It measured when a small text message, an image file and an audio file was sent from the server to the client within the web service response. Compared to the Android memory tests, the SOAP messages used more memory than the REST messages. However a familiar pattern occurs. The memory usage of the REST web service calls grew in an almost linear fashion, like the Android tests. The SOAP web service calls peaked with the image tests; however the audio tests still used more than the text test, which is also similar to the Android tests. The median values for the memory usage tests were Text SOAP/REST: 13032/7644, Image SOAP/REST: 18232/11372 and Audio SOAP/REST: 16176/13808. These median values are close to their corresponding mean values, except for the SOAP audio test. The median value is almost 1000 higher than the mean, meaning that some of the results were very low.

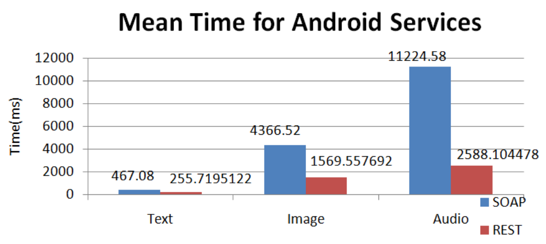

Figure 4. Mean Time for Android Services

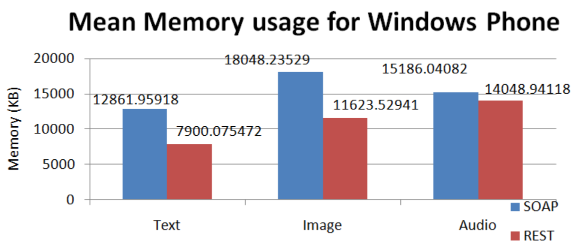

Figure 5. Mean Memory usage for Windows Phone

Figure 6 shows the mean data usage of all the tests carried out on the Windows Phone device for both SOAP and REST while each web service was called. Each value within the mean was the total amount of data transmitted and received by the application, measured in bytes. These results differ from the results of the Android tests. In the image test, the REST web service calls required more data on average. The text test results have the same difference in data sizes (approximately 700 bytes), and the audio test results have the same pattern as the Android results, however the difference between the amount of data used in SOAP and REST is much less. The median values for the data usage tests were Text SOAP/REST: 1285/570, Image SOAP/REST: 226122/265414 and Audio SOAP/REST: 528574/527866. These median results are very close to the mean results. This shows that there was continuity in the amount of data sent and received by the Windows Phone application. This shows the size of the REST messages were all the same throughout the tests, and the size of the SOAP messages differed slightly over the tests carried out.

Figure 7 shows the mean time taken for all the tests carried out on the Windows Phone device for both SOAP and REST while each web service was called. Each value within the mean was the time taken for the web service call to return the response to the client. The time is measured in miliseconds. These results show a greater improvement over the mean time results of the Android tests. The SOAP audio web service call lasted on average 21 times longer on the Android device than on the Windows Phone device. In the case of the text test, the REST web service calls took longer than the SOAP web service calls, however the REST image and audio web service calls took less time than their SOAP counterparts, which coincides with the Android test, even though the Android SOAP web service calls took much longer to complete. The median values for the time tests were Text SOAP/REST: 75/135, Image SOAP/REST: 226/215 and Audio SOAP/REST: 465/439. These median results are mostly similar to the mean results; however the SOAP text result is a quarter lower than its corresponding mean. This shows that there were a few results within the SOAP text raw result data with high results, showing that some of the SOAP text web service calls took an abnormally long length of time.

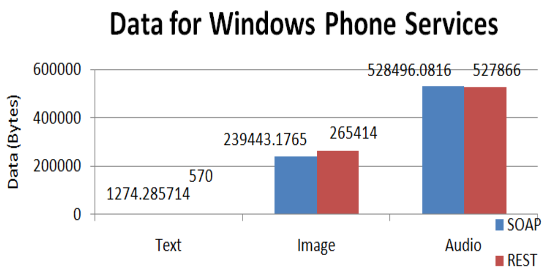

Figure 6. Mean Data usage for Windows Phone Services

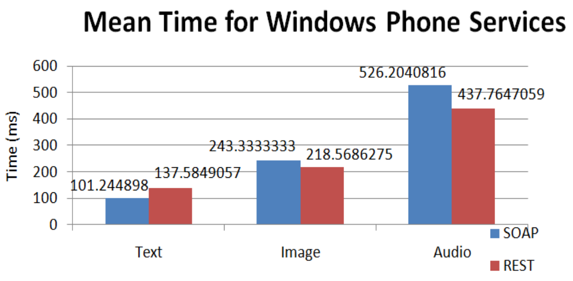

Figure 7. Mean Time used for Windows Phone Services

5. Conclusion

We sought here to examine how both SOAP and REST web services impact mobile devices. The results were split into different categories; by operating system, and then by the type of data recorded (i.e. Time, Memory and Data). The results were statistically analysed, calculating the mean and the median for each set of results and the means of each set was displayed on a graph, comparing the text, image and audio over SOAP and REST. The memory usage of both SOAP and REST must be considered when choosing which web service type to use, if the web application is designed to connect to a phone. If small text based messages are sent to the client, then a REST based web service connected to a Windows Phone client appeared to be the most efficient in terms of memory usage. However for sending larger size responses to the client, Android SOAP messages should be used, as it is able to process large size responses with the least amount of memory used. The data usage of SOAP and REST on mobile devices is similar to the data usage of SOAP and REST on personal computers. On average, REST uses less data than SOAP when sending the same amount of data from the server to the client. This test was used to find out how much data it would use on a mobile device, as mobile phone networks charge customers for every kilobyte sent or received. This makes the decision of which web service to employ even greater. This makes REST a cheaper option for the consumer. However, if using Windows Phone, sending an image seems to use more data, however the difference is minimal. It is interesting to note that the more data was used by SOAP on Android, compared to REST, the higher the size of the web service response became.

In terms of the amount of time taken for the client on the mobile phone to receive the web service response, REST performed faster than SOAP in all the tests, except the text tests on windows phone. They showed to be faster than REST on average. However the difference was on average 36 milliseconds. This difference is negligible. These tests also gave unexpected results for the Android operating system. For the SOAP image web service, the response took 2.78 times longer to complete. For the SOAP audio web service, the response was 4.34 times longer. A possibility for this may be due to the kSOAP2 library. It may only be programmed to handle small sized responses of less than 2-3 KB with relative efficiency.

If a programmer is deciding on whether to create a web service in SOAP or REST, which would be consumed by mobile clients, be it their own application or an application programmed by a third party, a number of different aspects must be examined. If a developer is designing a web application as well as the mobile client, REST would provide the most efficient means of sending and receiving the web service request and response. However, if security is required, then this may cause issues, as the request information is shown in the URI, and a security protocol would need to be created for the application, which can add overhead to the application. Whereas SOAP has security features built into it, through WS-Security. SSL is an option for securing REST services; however man in the middle attacks can occur when using Wi-Fi, which can decrypt SSL encryption, allowing a hacker to view the information being sent between the client and the web service. If the developer is only creating the web application to be used by both PC and mobile clients, and is not creating the client, SOAP may be the better choice; so long as the web service does not send any large responses to the mobile client (e.g. data sizes over 10KB). This is because the developers creating the clients may not know what exactly the server is returning, or needs to program for multiple objects (e.g. if the server returns an object with a mixture of strings, integers etc.). It is also worth noting that the concept of REST is relatively simple to understand, however, implementing a REST based web service is not. Implementing REST requires a programmer to possess extensive knowledge of the result messages before he or she can create a client for the web service. These include the variables sent and received by the web service and the format in which the data is returned (XML, JSON, plain text). These issues do not occur when programming SOAP based clients, as the WSDL document provides the information required, and there are tools available which can read and understand WSDL files.

References

[1] Bright P. *Windows Phone 7 nears the finish line with SDK release date*, ars Technica, Sep 2010. http://arstechnica.com/microsoft/news/2010/08/windows-phone-7-nears-the-finish-line-with-sdk-release-date.ars

[2] Tarnowsk N. *Google Releases Android 2.3 SDK for Developers*. http://www.blogsdna.com. Dec 2010 http://www.blogsdna.com/14495/google-releases-android-2-3-sdk-for-developers.htm
[3] Gamulin, N. *How to Consume WCF Service with Android*. StackOverflow.com. http://stackoverflow.com/questions/669764/how-to-consume-wcf-service-with-android.
[4] Hoffman K. *Accessing Web Services from Windows Phone 7*. Kotancode.com. July 29th 2010, http://www.kotancode.com/2010/07/29/accessing-web-services-from-windows-phone-7/
[5] Heuer T. *How to Consume WCF and ASP.NET Web Services in Silverlight*. http://www.silverlight.net/learn/videos/all/how-to-consume-wcf-and-aspnet-web-services-in-silverlight/.
[6] Yu Q, Liu X, Bouguettaya A, Medjahed B. Deploying and managing Web services: issues, solutions, and directions. New York: Springer-Verlag New York, Inc., 2008; 17: 537-572.
[7] Nandigam J, Gudivada V, Kalavala M. Semantic Web services. *J. Comput. Small Coll*. 2005; 21(1): 50-63.
[8] Lerner RM. At the Forge: Introducing SOAP. *Linux Journal*. Mar 2001.
[9] Albrecht C. How clean is the future of SOAP? *Communication of the ACM*. 2004; 47(2): 66-68.
[10] Castagna G, Gesbert N, Padovani L. A theory of contracts for Web services. *ACM Transactions on Programming Languages and Systems*. 2009; 31(5): 1-59.
[11] Christensen J. *Using RESTful web-services and cloud computing to create next generation mobile applications*. Proceeding of the 24th ACM SIGPLAN conference companion on Object oriented programming systems languages and applications. Orlando. 2009: 627-633.
[12] Falaki H, Mahajan R, Kandula S, Lymberopoulos D, Govindan R, Estrin D. *Diversity in Smartphone Usage*. San Francisco. Proceedings of the 8th international conference on Mobile systems, applications, and services. California. 2010: 179-194.
[13] Box D. A Brief History of SOAP. http://www.xml.com/pub/a/ws/2001/04/04/soap.html
[14] Muelhen M, Nikerson J, Swenson K. Developing Web Services Choreography Standards. *Decision Support Systems*. Special issue: Web services and process management. 2005; 40(1): 9-29.
[15] Pautasso C, Zimmermann O, Leymann F. *RESTful Web Services vs. "Big" Web Services: Making the Right Architectural Decision*. International World Wide Web Conference Committee (IW3C2). BeijIng, China. 2008: 1-10.
[16] Anh Phan K, Tari Z, Bertok P. *A benchmark on soap's transport protocols performance for mobile applications*. Proceedings of the ACM symposium on Applied computing. New York, USA. 2006: 1139-1144.
[17] Apte N, Deutsch K, Jain R. *Wireless SOAP: Optimizations for Mobile Wireless Web Services*. In WWW (Special interest tracks and posters). Chiba, Japan. 2005: 1178-1179.
[18] Powell M. Real SOAP Security. MSDN.microsoft.com. 2010. http://msdn.microsoft.com/en-us/library/aa480522.aspx.
[19] Thompson S. Implementing WS-Security. IBM Developerworks. 2009. http://www.ibm.com/developerworks/webservices/library/ws-security.html.
[20] Seely S. Understanding WS-Security. Microsoft MSDN. Microsoft Corporation. 2002. http://msdn.microsoft.com/en-us/library/ms977327.aspx.
[21] Rahaman M, Schaad A, Rits M. *Towards secure SOAP message exchange in a SOA.* Proceedings of the 3rd ACM workshop on Secure web services. 2006: 77-84.
[22] Costello R. Building Web Services the REST Way. xFront. http://www.xfront.com/REST-Web-Services.html.
[23] Tyagi S. RESTful Web Services. Oracle. http://www.oracle.com/technetwork/articles/javase/index-137171.html
[24] Erenkrantz J, Gorlick M, Suryanarayana G, Taylor R. *From representations to computations: the evolution of web architectures*. Proceedings of the the 6th joint meeting of the European software engineering conference and the ACM SIGSOFT symposium on The foundations of software engineering. New York, USA. 2007: 255-264.
[25] Elkstein M. *Learn REST: A Tutorial*. Elkstein.org. 2010. http://rest.elkstein.org/
[26] Oppliger R, Hauser R, Basin D. SSL/TLS session-aware user authentication - Or how to effectively thwart the man-in-the-middle. *Computer Communications*. 2006; 29(12): 2238-2246.
[27] Sindark. The ‘SSL strip’ exploit. Sindark.com. http://www.sindark.com/2009/02/21/the-ssl-strip-exploit/
[28] Singh R, Mishra S, Kushwaha D. An efficient asynchronous mobile web service framework. *ACM SIGSOFT Software Engineering Notes*. 2009; 34(6): 1-7.
[29] Brownlow M. Smartphone statistics and market share. Email Marketing Reports. http://www.email-marketing-reports.com/wireless-mobile/smartphone-statistics.htm
[30] Cassavoy L. What is a Smartphone. cellphones.about.com. http://cellphones.about.com/od/glossary/g/smart_defined.htm.
[31] Dong Y, Zhu H, Peng J, Wang F, Mesnier M, Wang D, Chan S.. RFS: a network file system for mobile devices and the cloud. *ACM SIGOPS Operating Systems Review.* 2001; 45(1): 101-111.
[32] Meier R. Professional Android Application Development. Indianapolis: Wrox, 2009.

[33] Open Handset Alliance. FAQ Open Handset Alliance. 2010. http://www.openhandsetalliance.com/oha_faq.html.

[34] Grønli T, Hansen J, Ghinea G. *Android vs Windows Mobile vs Java ME: a comparative study of mobile development environments*. Proceedings of the 3rd International Conference on PErvasive Technologies Related to Assistive Environments. Arlington, USA. 2010: 451-458.

[35] Zanuz L, Filippetto A, Barcelos G, Crespo S, Pinto C. *SOA engine: services compositions execution in ubiquitous environments*. Proceedings of the XIV Brazilian Symposium on Multimedia and the Web. Vila Velha, ES. 2008: 25-28.

[36] Hall S, Anderson E. Operating systems for mobile computing. *Journal of Computing Sciences in Colleges*. 2009; 25(2): 64-71.

[37] Dobjanschi V. Developing Android REST client applications. Google I/O 2010. http://www.google.com/events/io/2010/sessions/developing-RESTful-android-apps.html.

[38] Cansin A. How-to: Android as a RESTful Client. Praeda & Co. ceng.metu.edu. http://senior.ceng.metu.edu.tr/2009/praeda/2009/01/11/a-simple-restful-client-at-android/

[39] Zanuz L, Filippetto A, Barcelos G, Crespo S, Pinto C. *U-SOA: towards a ubiquitous platform based on service-oriented architecture*. Vila Velha/ES: ACM. 2008: 105-108.

[40] Android Developers. What is Android. Android Developers. http://developer.android.com/guide/basics/what-is-android.html.

[41] Zanuz L, Filippetto A, Barcelos G, Crespo S, Pinto C. *SOA engine: services compositions execution in ubiquitous environments*. Proceedings of the XIV Brazilian Symposium on Multimedia and the Web. New York. 2008: 25-28.

[42] Application Fundamentals. http://developer.android.com/guide/topics/fundamentals.html

[43] Hall S, Anderson E. Operating systems for mobile computing. Consortium for Computing Sciences in Colleges. 2009: 64-73.

[44] Suror. Windows Phone 7 minimum specs revealed. wmpoweruser.com. WMPower User, March 16th 2010 http://wmpoweruser.com/windows-phone-7-minimum-specs-revealed/

[45] MDSN Microsoft. Application Platform Overview for Windows Phone. MSDN. http://msdn.microsoft.com/en-us/library/ff402531(v=VS.92).aspx.

[46] Box D, Hejlsberg A. LINQ: .NET Language-Integrated Query. MSDN. Microsoft. 2011. http://msdn.microsoft.com/en-us/library/bb308959.aspx.