Performance of Web services on Mobile Phones

Sander Kikkert
Supervised by Prof. Dr. M. Aiello

February 2, 2010

1 Introduction

Smartphones are cell phones with more advanced capabilities than ordinary cell phones. They have usually more processing power and larger screens than normal phones and they offer PC-like functionality. Last year their prices dropped significantly which gave a boost to their market share.

A Web service is a way to call a procedure on a remote computer system with web-related standards like SOAP and HTTP. They are becoming a de facto standard for information exchange between different computer systems.

When one combined smartphones and Web services i.e. to run a Web service on a mobile device one can greatly increase the functionality of mobile devices to interact with its environment. This makes a wide range of new functionality possible. Especially in the field of ubiquitous computing.

Some of the key challenges of running a Web service on a mobile device includes: to host a service on a platform that is not designed to host services. For instance Symbian OS has an extensive HTTP framework to access HTTP based services but it is not possible to use that framework to host a service. Another challenge is that the mobile platforms are optimised to save energy as much as possible. In order to achieve that goal some platforms (mainly Symbian) use low-level coding techniques to make very efficient coding possible.

Running Web services on mobile devices opens a wide range of new possibilities and solves heterogeneity and interoperability issues [7]. However the performance characteristics are important to ensure the mobile device is able to handle multiple requests while the device is able to function normally. I study the performance characteristics of Web services running on the various mobile phone platforms.

The remainder of this paper is organized as follows: The application domain of mobile Web services and related work are discussed in section 2. Section 3 defines the concept of Web services performance testing and describes the performance metrics used in the performance analysis. The experimental setup and implementation are discussed in section 4. Section 5 contains the results whereas section 6 gives a evaluation of the results. Finally the last 2 sections discuss conclusions and further research.

Personal Web services make a new level of ubiquitous computing possible because external systems can exchange information with the user almost without user intervention.

2 Related work

Measuring the performance of Web services running on mobile devices (mobile or personal Web services) spans diverse research fields. Of particular interest are the new applications that mobile Web services opens up. Berger, McFaddin, Narayanaswami and Raghunath describe a fully automatic billing system as an application of mobile Web services[4]. When a customer goes to the check-out with a filled shopping cart the contents of the shopping cart is automatically determined with help of RF technology. After that the check-out establishes a connection with the personal Web service of the customer. The mobile device verifies the payment request and after customer approval completes the transaction. The article describes further the technical complications of personal Web services. The complications range from assigning IP addresses at different locations to security issues. Only trusted sources should be allowed to connect to your personal wallet service.

An approach to make a mobile device easy reachable is to make use of a home agent of the mobile device which keeps track of the current location of the device [9]. Whenever the mobile device moves to another location it requests a new temporary address and informs its home agents of the new address. In this way a mobile device is always reachable with the same fixed address.

Aiello[1] shows possibilities and the feasibility of Web services at home. According to his work Web services will have a central role in the home of the future as an infrastructure to ensure openness, scalability and heterogeneity (total interoperability) among home devices. The home of the future will have a hierarchy of devices, sensors and actuators all with different processing power and network capabilities. As a case study the article describes an elderly home that is capable of monitoring a fall of the inhabitant.

Work in the area of performance measuring of Web services can be found in an article by Lencevicius and Metz [5]. The work gives an overview about how to use performance assertions to validate performance requirements. It describes and resolves issues in the use of performance assertions and it describes in detail a framework that can be used to validate the performance of mobile devices.

Saddik [6] describes a way of measuring the performance of a Web service: one measures the response time and number of successful responses per second at different load levels (for instance 1, 5, 10, 25 agents/threads). First the throughput increases if you increase the number of agents. But at a certain load level the throughput will not increase any more. That throughput is the performance of the Web service according to this article.

The performance of various solutions for realizing Web services on embedded devices are studied by D. Schall, M. Aiello and S. Dustdar [7]. Two Web services frameworks for Symbian OS are compared: gSOAP (C++ based) and kSOAP (J2ME/Java based). They measured the performance with help of time stamped messages at various layers in the Web services stack. gSOAP has a better performance than kSOAP probably due to its C++ nature. In this work Web services are proposed to solve heterogeneity and interoperability issues when embedded devices wants to interact with each other in a continuously changing environment.

S. N. Srirama, M. Jarke, and W. Prinz[8] describes a similar set-up. They propose to use resource constrained smartphones to host Web services (called Mobile Hosts in the article) to host services in a true mobile peer-to-peer envi-

ronment. As an example, the work describes a guided parcel service, where a delivery-van equipped with a mobile host can send detailed location information to the client. They show the feasibility of Mobile Hosts with a prototype based on the kSOAP framework running on a Sony-Ericsson P800 Smart Phone and they perform a performance analysis. Their performance analysis uses timestamps and it turns out that total Web service processing time is a small fraction of the total request-response time ($< 10\%$) the rest being transmission delay.

Baranov[2] points the importance and the advantages of performance testing during the developing of a Web service. For instance performance-related bugs are easier to caught and to solve. Web services should be tested with invalid SOAP requests and valid requests with unusual or unexpected values as well as valid requests preferable in a real world mix. If a Web service is stateful then requests are dependable of each other i.e. in a ticket reservation system (first you login and then you make a reservation). In this context the virtual user emulation mode are recommended to test the performance. But in the case of a stateless Web service the request per second method is recommended to use. If a load test can sustain only the scheduled number of users, the effective requests injection rate may decrease dramatically if the server response time is long. The paper concludes with some tips about performance measuring. The response time of a request might depend heavily on the network environment. If the network is slow then the response time is not an accurate figure, it is better to monitor the server time as well. Take care that the PC running the test software has enough resources in order to get proper results. A last suggestion is that your network settings of Windows could affect the test results.

Tian, Voigt, Naumowicz, Ritter and Schiller[10] describes a way to increase the performance of mobile Web services by compression of the messages. According to this article the increase in performance is high whenever the connection is slow and the messages are big.

An extensive overview of the Android platform is done in an article by Speckmann [3]. In his Master's thesis he compares in detail Windows Mobile, Symbian OS and Android. Android is the best OS for mobile phones. He stated that Android is able to run unmodified Java code. That is incorrect; Android is not able to run neither Java SE nor Java ME but Android code does has a Java like syntax. A mistake which suggests that the writer did not take a critical look at Android.

A better comparison is found in a article by Wei et al.[12]. It turns out that differences between the platforms are relatively small.

3 Concept

Scalability is a capability of a Web service that describes how well a Web service performs under different load conditions. Scalability is closely tight to the performance of a Web service. A Web service that serves thousands of requests is not of much use if half of them are error responses [6].

In order to measure the performance and scalability of a Web service I need to identify some metrics that characterize Web service performance:

- Total number of Responses
- Number of succesfull responses and Succesfull responses per second

- Percentage of error responses
- Average execution time: The time from the last byte sent to the first byte received in milliseconds
- Average server time: The time from the first byte sent to the last byte received in milliseconds

In the following performance study the above metrics will be measured at 4 different load levels: 1, 5, 10 and 25 clients. The measure time will be set to 2 minutes. Each measurement will be repeated 3 times and averaged in order to lower the measurement error. At a certain point the throughput of the Web service will not increase and that is the maximum throughput (performance) of the Web service [6].

4 Experimental Setup and Implementation

There are multiple operating systems for mobile devices. The most popular operating system for smartphones is still Symbian OS. But Iphone OS and especially Android are growing fast. Microsoft has also its own operating system for mobile devices called Windows Mobile. However Iphone OS is not freely available thus limiting my performance study to Symbian OS, Windows Mobile and Android. In the following performance study I will compare the Web service runnning on a Windows Mobile 5.0 PDA with the emulated versions.

4.1 Implementation

To test the performance of a mobile Web service a Web service needs to be constructed. The Web service needs to do some processing in order to try to increase the test result differences. An echo service simply replies the input without any processing and therefore the CPU is busy with making and breaking up of connections. To let the CPU do some actual work the test Web service determines if an IP address is inside the Netherlands. To speed up that search process the Web service makes use of the binary search algorithm.

All the three Web services function in the same way. First the Web service listens for incoming connections. The HTTP frameworks incorporated into the different APIs support outgoing connections only. To make an incoming connection possible you have to create a TCP socket yourself and start listening on it. After an incoming connection is made another socket is created to handle this connection. First the request is parsed and adequate action is taken. Five different responses are possible:

- Get-request for `geoip.wsdl`. Returns the WSDL-file. That is an XML description of the Web service.
- Get-request for any other file. Returns a HTTP 404 file not found error response.
- Valid post-request to invoke the Web service. Returns a SOAP message with either true or false.
- Invalid post-request. Returns a invalid SOAP request error response.

- Invalid HTTP request. Returns a HTTP 400 bad request error response.

In the case of a valid post request the IP address is extracted from the request. The IP address is in the dot-decimal notation *(xxx.xxx.xxx.xxx)* and needs to be converted to a 32 bit unsigned integer. After this conversion the unsigned integers is looked up in an array with the Dutch IP ranges. This array is so constructed that all even places (`array[2*t]`) corresponds to a start value and all odd places (`array[2*t+1]`) corresponds to an end value of a range. Because it is an ordered array we can use binary search to speed up the search process. If the integer value turns out to be inside a range true is returned otherwise false is returned.

4.2 Implementation differences

C# and the Java-like programming language of Android are high-level programming languages which use managed code, have garbage collection and have an extensive function library. Symbian OS on the other hand has a C++ like programming language which is more low-level, does not have a garbage collector and makes use of special techniques such as descriptors and the cleanup stack. In 1990s when Symbian is designed those techniques were necessary to get a reasonable performance on the then powerless hardware. Nowadays it is questionable if the low-level techniques of Symbian are still beneficial. It introduces extra complexity because programmers have to concentrate on low-level techniques instead of application specific logic. One of those very efficient techniques is Active Objects. An Active Objects is a very lightweight non-pre-emptive kind of thread which runs inside a thread. Because they are non-pre-emptive there is no need to use mutexes, semaphores, critical section or other kinds of synchronization to protect against the activities of other threads. An Active Object is about 100 times lighter than a thread in a normal language.

4.3 Experimental Setup

To test the performance of the Web services, Parasoft SOAtest 6.1 and Loadtest 6.1 will be used. In Loadtest 6.1 it is possible to define an amount of threads/-clients that concurrently sends requests to the Web service for a certain amount of time. *129.125.102.100* will be used as an input value for the Web service which is a RuG IP-address so *true* should be returned. When the tests are done the metrics mentioned in the concept section will be displayed.

The Symbian version of the Web service is targeted at Symbian OS 9.3 (S60 3rd edition FP2), currently the most used version of Symbian OS. Symbian support Java (J2ME) and a C++ dialect which runs faster because it can be compiled into native machine code instead of byte code. I will choose Symbian C++ for the Symbian Web service. The Android version is targeted to a Android 1.5 r3 device. Android supports one language: a Java-like language with an extensive function library, which is not compatible with J2SE or J2ME. Windows Mobile supports multiple programming languages ranging from C# to C++. The Windows Mobile Web service will be written in C#, the most common language to use for Windows Mobile.

The test software and emulators will run on a modern PC running Windows XP (Core 2 Duo E8400, Intel P43 Chipset, 4GB main memory) to ensure that

the PC is not a bottleneck. While the performance tests are running the CPU usage will be monitored to confirm that the host PC has enough computational power. The real Windows Mobile device is running Windows Mobile 5.0 and will be connected to the same PC by either a USB 2.0-cable or a Wi-Fi connection (IEEE 802.11b/g) to a wireless AP with a built-in switch which has a 100MBit connection with the PC.

5 Results

5.1 Symbian OS (Emulator)

Load level	Total Number of Responses	Number of Successfull Responses	Percentage of Error Responses	Succesfull Responses p/s	Avg. Exec. Time (ms)	Avg. Server Time (ms)
1 Client	2277	2277	0.0%	18.9	2.0	2.0
2 Clients	3234	3234	0.0%	26.9	3.0	3.0
3 Clients	3820	3797	0.6%	31.6	4.0	4.0
4 Clients	5084	3868	23.9%	32.2	7.0	6.0
5 Clients	7804	3914	49.8%	32.6	10.0	8.0
10 Clients	9561	3998	58.2%	33.3	17.5	16.0
25 Clients	12483	4057	67.5%	33.8	54.7	92.0

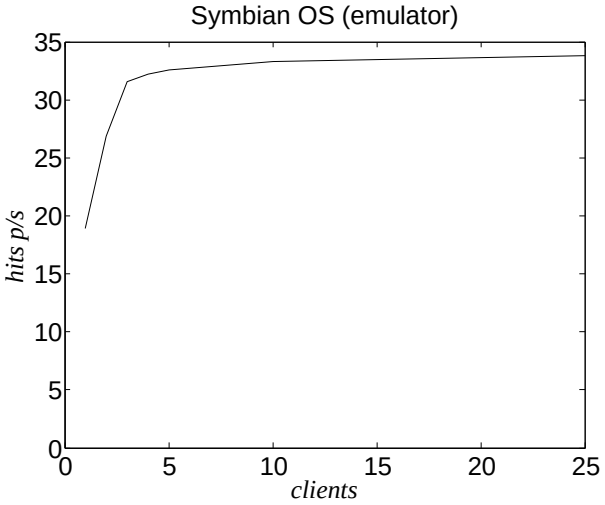

Figure 1: Performance Symbian OS (emulator)

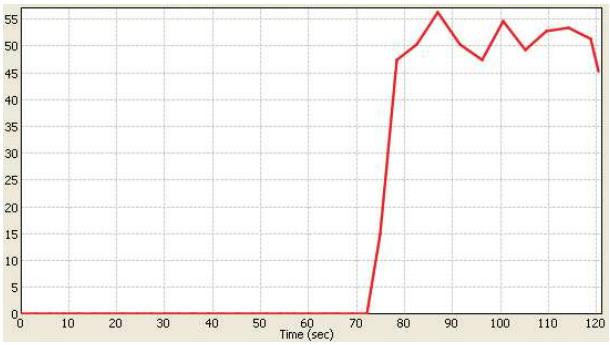

Figure 2: Error rate Symbian OS (emulator) (5 clients)

5.2 Android Web service (Emulator) - Single-threaded

Load level	Total Number of Responses	Number of Successfull Responses	Percentage of Error Responses	Succesfull Responses p/s	Avg. Exec. Time (ms)	Avg. Server Time (ms)
1 Client	111	111	0.0%	0.92	28	27
5 Clients	122	122	0.0%	0.98	24	24
10 Clients	124	124	0.0%	0.95	30	30
25Clients	141	141	0.0%	0.97	66	66

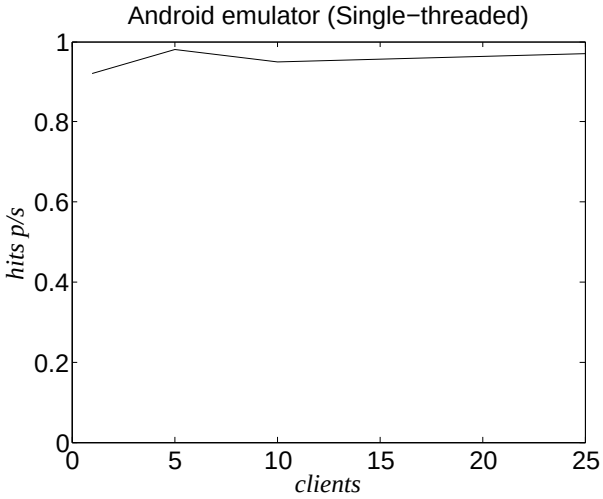

Figure 3: Android Web service (Emulator) - Single-threaded

5.3 Android Web service (Emulator) - Multi-threaded

Load level	Total Number of Responses	Number of Successfull Responses	Percentage of Error Responses	Succesfull Responses p/s	Avg. Exec. Time (ms)	Avg. Server Time (ms)
1 Client	107	107	0.0%	0.89	23	22
5 Clients	119	119	0.0%	0.95	26	26
10 Clients	127	127	0.0%	0.98	32	31
25 Clients	139	139	0.0%	0.96	69	75

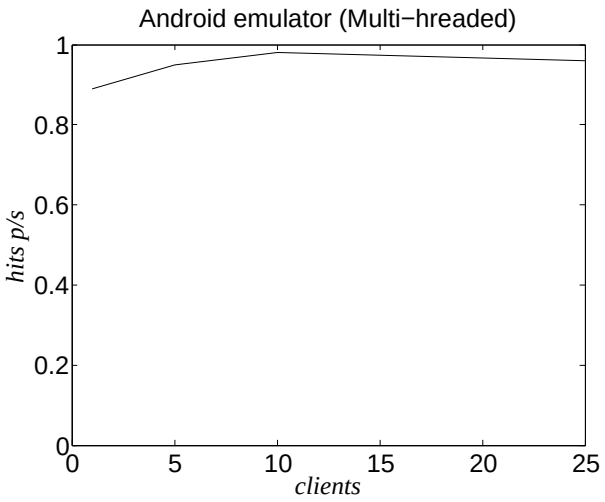

Figure 4: Android Web service (Emulator) - Multi-threaded

5.4 Windows Mobile Web service (Emulator)

Load level	Total Number of Responses	Number of Successfull Responses	Percentage of Error Responses	Succesfull Responses p/s	Avg. Exec. Time (ms)	Avg. Server Time (ms)
1 Client	554	554	0.0%	4.61	132	131
5 Clients	967	957	1.67%	7.97	416	382
10 Clients	1283	1141	11.9%	9.51	647	464
25 Clients	1601	1153	36.3%	9.6	1511	618

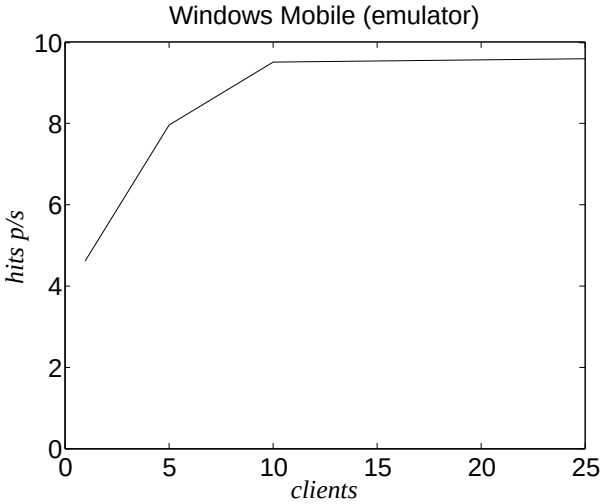

Figure 5: Windows Mobile Web service (Emulator)

5.5 Windows Mobile Web service - PDA (Cable connected)

Load level	Total Number of Responses	Number of Successfull Responses	Percentage of Error Responses	Succesfull Responses p/s	Avg. Exec. Time (ms)	Avg. Server Time (ms)
1 Client	1551	1550	0.06%	12.92	16	16
5 Clients	3516	3514	0.06%	29.28	84	83
10 Clients	3704	3558	4.13%	29.65	176	134
25 Clients	4236	3511	19.76%	29.26	421	156

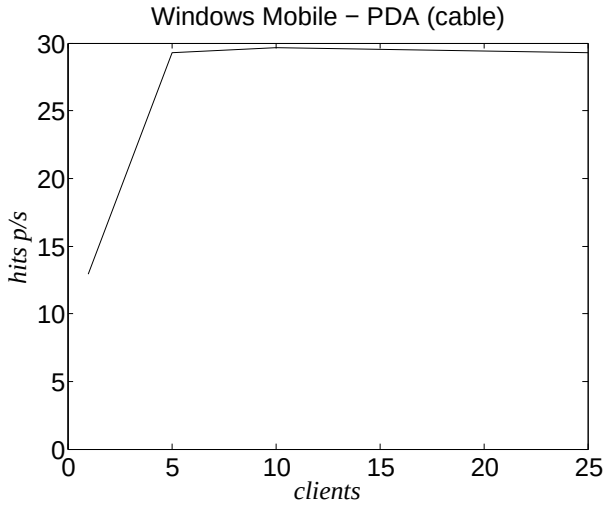

Figure 6: Windows Mobile - PDA (cable)

5.6 Windows Mobile Web service - PDA (Wi-Fi)

Load level	Total Number of Responses	Number of Successfull Responses	Percentage of Error Responses	Succesfull Responses p/s	Avg. Exec. Time (ms)	Avg. Server Time (ms)
1 Client	1670	1670	0.02%	14.1	16	15
5 Clients	5657	4968	13.8%	41.4	35	29
10 Clients	6349	5409	17.5%	45.1	59	43
25 Clients	6373	5490	15.7%	45.7	156	53

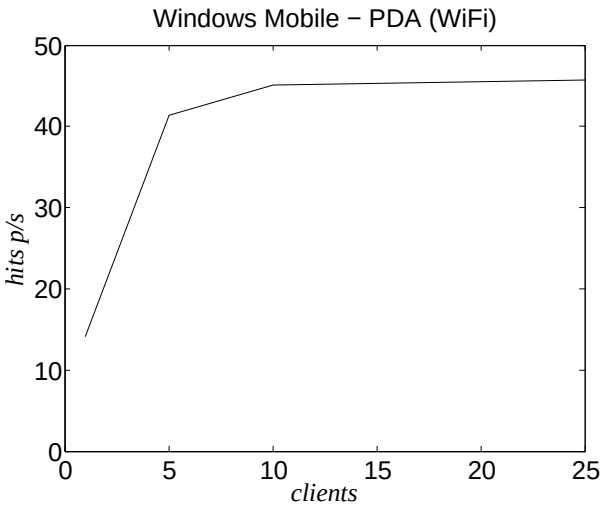

Figure 7: Windows Mobile - PDA (Wi-Fi)

5.7 Platform comparison

Platform	Performance (Hits p/s)	Percentage of Error Responses (25 Clients)	Avg. Exec. Time (25 Clients)	Avg. Server Time (25 Clients)
Symbian OS	33	67.5%	54.7	92
Android ST	0.95	0%	66	66
Android MT	0.95	0%	69	75
WM emulator	9.6	36.3%	1511	618
WM PDA - cable	29	19.70%	421	156
WM PDA - WiFi	45	15.7%	156	53

6 Discussion

6.1 Symbian OS (Emulator)

Figure 1 shows us that the throughput of the Web service running on a Symbian device emulator is at most about 33 requests per seconds. This is a very reasonable performance (for an emulator). The scalability however is disappointing. The loss of requests is enormously by 5 clients (49.8%) consequently I repeated the test with 2, 3 and 4 clients. It turns out that up to 3 clients there is some processing power left. But from 4 clients and onwards there are more requests coming in than the Web service can handle. In that case the failure count increases dramatically and the Web service becomes useless in a normal (production) environment. The bottleneck of the Web service is not the host PC. The Web service running on the emulator uses 6 seconds CPU power (5 threads) which is just a fraction of the processing power of the host PC (5%). The process list of the Symbian OS emulator reports 1 second CPU time. Making and destroying connections must be very time consuming in comparing to the actual work. That is the most probable explanation for the (relatively) low performance of this Web service.

A possible explanation for enormous error rate from 4 clients and onwards is that Symbian OS buffers too much incoming connections. When the Web service performs at its maximum it does not have time to process all those requests. The error rate confirms that assumption. In the first part of the graph of figure 2 the error rate is zero because many requests have been sent and they could be returned. However after about a minute the time-out timer expires of all outstanding requests and at that moment the error rate explodes.

6.2 Android Web service (Emulator) - Single-threaded

The normal - single threaded - version of the Android Web service has very low performance. Although the performance is bad with one thread, increasing the number of threads does not improve the performance. Increasing the number of threads only increases the average execution and server time. The scalability is practically non-existent because the performance does not enlarge with the number of threads. However all request are being processed; the failure count is - independent of the load - 0.

6.3 Android Web service (Emulator) - Multi-threaded

In order to improve the performance the single threaded version has been rewritten to make use of multiple-threads. Normally making a program multi-threaded improves the performance. But it does not make a significant difference on the performance. It is as low as the single threaded version. The performance of both version is about 1 request per second.

I could not find a plausible explanation for low performance of this Web service. The emulator uses at most 6 seconds CPU time of the host PC (CPU usage: 5%) so that is not a limit in any way. It takes the emulator one second to process a single request which is long that you would expect that the program is waiting most of the time.

6.4 Windows Mobile Web service (Emulator)

The performance of the Windows Mobile emulator is better than the Android emulator but still low (about 10 requests per second). The performance scales disappointing with the number of threads. A loss of 36.3% is unacceptable high. The emulator does not draw a noticable amount of system resources of the host PC. But let us see how the Web service performs on a real device.

6.5 Windows Mobile Web service - PDA (Cable connected)

The performance of the Web service running on a real device is pretty decent (30 requests per second). It processes more requests per second than the Windows Mobile emulator. However the failure count is too high in the case of 25 threads.

Another interesting notion is that the device performed twice as much when it was connected to a power outlet instead of running on battery power. The clock of the CPU is probably clocked down when running on battery power so we may conclude that the CPU is fully utilised.

6.6 Windows Mobile Web service - PDA (Wi-Fi)

It is interesting to see if the connection method of the device to the test machine has an influence on the performance. In the first test the device was connected with help of a standard USB-cable. In this case the PDA is connected through Wi-Fi. And the figures are higher than by the cable method. The bandwidth of the USB-cable is not the explanation for this difference. Sending 30 request per second takes about 20KB/s which does not come close to the maximum transfer rate of USB 2.0 (480 MBit). The cable connection is probably largely driven by the CPU of the PDA while the Wi-Fi is probably driven by a separate microprocessor.

6.7 Platform comparison

The most obvious difference is the performance of the emulators: The Symbian OS emulator is by far the fastest and about three times faster than the Windows Mobile emulator and about thirty times faster than the Android emulator. This difference in performance could be explained by the efficient low-level nature of Symbian.

A similarity between the three emulators is that they all use very little processing power of the host PC. So processing power is not the bottleneck but opening/closing connections are probably the bottleneck.

The scalability differs enormously. The Android version has a zero error rate at a load of 25 clients probably caused by the low performance. The real Windows Mobile device handles 25 clients quite well with an error rate of about 15% till 20% despite of the limited hardware. The Windows Mobile emulator however losses way too much requests at a load of 25 clients and Symbian OS emulator has a very bad scalability (67.5% clients).

The real Windows Mobile PDA is performing much better than the emulators. The emulators seem to be not fully optimised to take full power of the host PC.

6.8 Results comparison - Van der Hoorn

H. van der Hoorn did a comparable research as his Master graduation project[11]. He compares 5 mobile platforms (Android, BlackBerry, iPhone OS, Symbian and Windows Mobile) and he performs 3 performance tests: AES encryption, AES decryption and XML parsing. The Symbian emulator is significant faster than the Windows Mobile emulator which in turn is significant faster than the Android emulator in all 3 tests. That is conform my research.

The tests are repeated on 3 real devices: an Android, Iphone OS and Symbian OS device. The Android device has more or less the same performance as the emulator while the Iphone OS and Symbian OS device has significant lower performance than the emulators. In my research however the Windows Mobile device is faster than the emulator. This difference can be explained by the fact that the test Web service handles a lot of connections while Van der Hoorn is mainly testing computional power. We both conclude that device emulators are not a suitable tool for performance measurements.

An interesting notion of Van der Hoorn is that the bad performance of the Android emulator could be explained by the lack of a JIT compiler. Which

means that no runtime optimizations are done resulting in bad performance.

7 Summary

It is certainly possible to run a Web service on a mobile device although they are not designed for running services. The performance of the emulators is much lower than expected and the differences in performance between the 3 platforms are huge. The emulators use just a fraction of the processing power of the host pc, so making/destroying an emulated connection must take a lot of time. The Web service running on a real device performs quite well and far better than the emulators.

8 Future work

Some ideas for future research includes: find out the bottleneck of the emulators and try to investigate what can be done to improve the performance. In the far future is probably possible to run multiple mobile operating systems on the same mobile device. And then it is possible to reveal the real difference between the different mobile platforms.

But before taking advantage of Web services on mobile devices a lot of work has to be done. First major security issues have to be solved and then the public has to be convinced to use Web services on their mobile devices. If that succeeds then a whole new level of ubiquitous computing is possible.

References

[1] Marco Aiello. The role of web services at home. In *AICT-ICIW '06: Proceedings of the Advanced Int'l Conference on Telecommunications and Int'l Conference on Internet and Web Applications and Services*, page 164, Washington, DC, USA, 2006. IEEE Computer Society.

[2] Sergei Baranov. Performance testing web services. WebSphere Journal (http://websphere.sys-con.com/node/46513), September 2004.

[3] S. Benjamin. The android mobile platform. Master's thesis, Eastern Michigan University, April 2008.

[4] Stefan Berger, Scott McFaddin, Chandra Narayanaswami, and Mandayam Raghunath. Web services on mobile devices - implementation and experience. *Mobile Computing Systems and Applications, IEEE Workshop on*, 0:100, 2003.

[5] Raimondas Lencevicius and Edu Metz. Performance assertions for mobile devices. In *ISSTA '06: Proceedings of the 2006 international symposium on Software testing and analysis*, pages 225–232, New York, NY, USA, 2006. ACM.

[6] Abdulmotaleb Saddik. Performance measurements of web services-based applications. *IEEE Transactions on Instrumentation & Measurement*, 55(5):1, 2006. Physical description: p. 5chart, 3diag.

[7] Daniel Schall, Marco Aiello, and Schahram Dustdar. Web services on embedded devices. *International Journal of Web Information Systems*, 2(1):43–48, 2006.

[8] Satish Narayana Srirama, Matthias Jarke, and Wolfgang Prinz. Mobile web service provisioning. In *AICT-ICIW '06: Proceedings of the Advanced Int'l Conference on Telecommunications and Int'l Conference on Internet and Web Applications and Services*, page 120, Washington, DC, USA, 2006. IEEE Computer Society.

[9] Andrew S. Tanenbaum and Maarten van Steen. *Distributed Systems: Principles and Paradigms (2nd Edition)*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 2006.

[10] M. Tian, T. Voigt, T. Naumowicz, H. Ritter, and J. Schiller. Performance considerations for mobile web services. *Computer Communications*, 27:1097, 2004.

[11] Hielko van der Hoorn. A survey of mobile platforms for pervasive computing. Master's thesis, Rijksuniversiteit Groningen, 2010.

[12] M. Wei, A. Chandran, H. Chang, J Chang, and Nichols C. Comprehensive analysis of smartphone os capabilities and performance. Technical report, University of Southern California, April 2009.