An End–End Approach to Wireless Web Access

Vladimir Korolev and Anupam Joshi
Department of Computer Science and Electrical Engineering
University of Maryland Baltimore County
Baltimore, MD 21250
{vkorol1,ajoshi}@cs.umbc.edu

Abstract

In this work we propose a lightweight scheme for negotiating client's capabilities in the context of end-end content adaptation for wireless web access. Our method is much less complex then W3C's proposed CC/PP framework. We suggest that for the purposes of content negotiation all (mobile) clients could be grouped into a few relatively large categories. With this assumption we simplify the CC/PP protocol and implement it as an Apache module. This paper describes the simple CC/PP protocol, issues related to its implementation, and performance measurements.

1. Introduction

With the advent of dynamic and executable content, integration of security mechanisms, and emerging metadata standards, it is clear that the Web is transforming into the basis of a globally distributed computing and information access system. Further, wireless access to the web from mobile/palmtop devices is increasingly attractive to many users, since it enables new applications in areas like m-commerce, public information services, education, telemedicine, battlefield awareness etc. For example business users with palmtop/laptop type devices (aka Road Warriors) constitute a large and growing segment of users. Their work typically involves accessing and modifying corporate information repositories with multimedia data over low bandwidth connections typified by wireless or phone-line access. More often than not, the applications they use are web enabled and use the browser as a "thin" client.

As was discovered fairly early on[JWM+96, JWH97, Kat94], the web (more specifically, the HTTP protocol) is designed to work in wired, high bandwidth environments, and does not operate particularly well when the access point is a mobile host. The reasons behind this such as low/variable bandwidth, disconnections, etc.[JWH97, FZ94] are quite well known. Further, the mobile host is typically resource poor. Even though the very high end laptop machines can now deliver performance comparable to low-end to moderate desktops, most "thin and light" subnotebooks and PDA/PCS type devices are constrained in terms of CPU, power, memory, disk, display capabilities etc. This creates a *capability mismatch* between the multimedia rich data that the web servers have on one hand, and the capability of the client to handle/display that data and the capability of the wireless network to deliver it in an acceptable time on the other.

These bandwidth and resource related problems are sometimes thought of as transient, since wireless network speeds as well as the resources available on mobile platforms are steadily increasing. It should be noted though that the speeds on wired networks and the resources (memory, CPU speed, etc.) on static hosts are increasing just as fast, if not faster. As has been amply demonstrated over the last decade, software catches up and uses all available hardware resources – Existing applications will evolve, and novel applications will be created, to use these enhanced capabilities. So while the absolute performance measures of mobile systems will undoubtedly improve, the *bandwidth gap* and *resource gap* will remain[Jos00].

There exists a large body of work which handles the problem of *capability mismatch* for multimedia content in wireless web access. The solution, typically, has been to use a client-proxy-server model. The proxy transcodes multimedia formats, most often images, according to some predefined rules, usually in some manner that trades quality for bandwidth. We present many such systems in the related work section. However, as PDA or thin-light notebooks become more popular, the proxy's functionality will increase. For example, some proxies now seek to deal with videos as well as images. This includes work done at Berkeley,

This research was supported in part by the DARPA DAML program under contract F30602-97-1-0215, by National Science foundation grants NSF IIS-9875433 and NSF CCR-0070802, by an IBM Faculty Development Award.

as well as our own recent work[BJA98]. We have also examined other questions that relate especially to PDA type mobile clients, such as what to do with active content and HTML fonts / styles which the PDAs typically cannot handle. There are also proxy approaches that re-render the HTML in a format appropriate for the PDA before transmitting. This added functionality of the proxy increases the computational resources it requires.

We have argued[Jos00] that a purely proxy based solution will become increasingly non scalable, especially with the number of users connecting wirelessly expected to grow. We note that limited functionality proxy systems have been developed recently that are quite scalable. A good example is Inktomi's traffic server, or the proxy developed by the Daedalus project at Berkeley for dial in connections. However, with the advent of palmtop type devices, the transformation needed by the client, and hence the computational resources needed by the proxy to affect it, increase significantly. While workstation clusters supporting proxies can possibly be deployed to provide computational resources, it is not clear that this *proxy only* approach provides the best solution to the problem. The role of proxies has been recently questioned elsewhere as well – there is some debate as to whether proxy based solutions are really needed to provide networking services to mobile clients. Moreover, the proxy based approach typically assumes that the data is being served by a host on the wired side. This means that a proxy can be run on some host with lots of MIPS on the wired side which is on the path from the server to the mobile client. Most often, this is at the mobile support station. Clearly, in ad-hoc networks that will be engendered by Bluetooth like devices, such an assumption would be fallacious.

The alternative is to make the server itself provide data in a format that is most suited for mobile access. This represents an instance of an end-end approach. End-end approaches are well known in networking and systems literature. In the web context, dual versions (graphics heavy vs. text only) of web pages kept at servers represent an end-end approach. To the best of our knowledge, Seshan [SSK97] were one of the first to present the notion that the Web clients could use network performance parameters to download documents from a server at different "fidelities", and explicitly mentioned that this was something beyond text only pages. Implicit in their paper was the idea that the server would indeed have different fidelities to present to the client. In prior work[Jos00], we have shown how a variation of the HTTP/1.1 content negotiation could be used to create an end–end system for web access from mobile hosts. However, that approach is awkward since it requires the creation and proliferation of new mime subtypes. Moreover, it also assumes that content in different versions is always pre-created.

However, recent standards from the W3C, in particular Composite Capability / Preference Profile (CC/PP) and XSLT, provide us with an efficient and robust mechanism with which to build and end–end system for mobile web access. In this paper, we point out problems with the existing CC/PP approach in the context of mobile access, and present a variation called *Simple CC/PP*. We implement this approach as an Apache module and provide experimental results of its efficacy.

2. Background & Related Work

2.1. Proxy based Transcoding approaches

In the past a considerable amount of work has been done in the area of the web access from mobile clients. Due to space limitations, we present here some of the larger efforts in enabling web access from mobile computers. Other related work done earlier includes the TeleWeb system of Schilit , the notion of stream transducers advanced by Brooks [BMMM96], location specific personalization[ST93], IBM's WebExpress[Cor], and Rover[JdT$^+$95].

Significant work in this area has been done by the Daedalus group at Berkeley. In GloMop [KBA$^+$96, FB96], the proxy performs *distillation* of the document received from the server before sending it to the client. Distillation is defined here as a highly lossy, real-time, datatype-specific compression that preserves most of the semantic content of the document. For instance, GloMop performs transcoding of motion JPEG to sub-sampled H.261 for video data. A more formal model for proxy functionality (TACC), along with an overview of their system, is described in [BKC$^+$98]. More recently, this group has used a similar approach to create a split browser[FGG$^+$98] for the Palm pilot PDA. Note however that their approach is essentially proxy based.

The Mowgli system [LKR96] consists of two mediators located on the mobile host and the mobile-connection host which use the Mowgli HTTP protocol to communicate with each other, reducing the number of round-trips between the client and server. Mowgli reduces the data transfer over the wireless link in three ways: data compression, caching, and intelligent filtering.

The notion of web intermediaries to affect transcoding and personalization related functionalities is also the focus of IBM's WBI[BMK97] system.

In the work of Noble [ea], the proxy is developed in the context of what the authors term *agile, application aware adaptation*. Basically, they allow an application to register with the OS its expectations about a resource and the variability it can tolerate. The Odyssey system monitors resources, and informs the applications via upcalls when the

resource value strays outside the bounds decided by the application. The application can then adapt its behavior. For web browsing in particular, a module called Cellophane on the client transforms HTTP requests from Netscape into file operations on Odyssey web objects and selects fidelity levels for images which are forwarded to a distillation server. However, this approach is specific to the Odyssey file system and requires a modified version of the Net BSD kernel. This also requires the addition of a module on the client.

2.2. XSL and XSLT

A few years ago W3C proposed XML as a new method for information representation. XML stands for eXtensible Markup Language [BSM96]. The main features of XML is the ability for a user to define her own tags and the requirement that each XML document must be well formed in terms of XML grammar. The later requirement simplifies writing the programs that deal with marked-up documents. Although it is possible to write such programs for HTML documents as well, the task is much more complicated because of considerable number of non conforming documents on the web, which are usually fine for displaying in the browser, but are very problematic when parsed by computer programs. Typical examples of non conformity include overlapping tag groups and not closed tags. Using XML a web master can mark up her documents based on the nature of the information contained in the document. For example a page in a catalog will have tags for item's price, catalog number, item description and so on. Separately from documents the web master must create an XSL (XML Stylesheet language) style sheet. XSL style sheet specify how different tags should be rendered. Such approach creates a cleaner separation between the content of a document and its presentations. When a user requests a particular document one of the two things could happen. The document could be combined with the style sheet on the server side and resulting HTML file sent to the user, alternatively in case the user's browser is capable of rendering XML/XSL documents directly, both the requested XML document and the XSL style sheet are be sent to the user's browser and then combined and rendered at the user side. In the context of providing Web based services to the mobile devices, this allows us to have different style sheets for different kinds of devices a rich and colorful style with interactive content can be used for desktop devices and minimally adorned style can be used for mobile devices such as PDA.

2.3. CC/PP Protocol

Recently W3C consortium has proposed a new protocol which is supposed to solve a problem of delivering web based services to mobile clients. The new protocol is called *Composite Capability/ Preference Profiles*. CC/PP allows the client to specify its profile, and for the proxy *e.g. WAP gateway* to tailor the content based on it. The two parts of CC/PP solution are the *CC/PP Exchange Protocol* and the *CC/PP Description Framework*. CC/PP Exchange Protocol is used for delivering descriptions of client's capabilities to the server over standard HTTP protocol. CC/PP Description Framework is a way of describing these capabilities.

CC/PP Exchange Protocol [HH99] works on top of HTTP/1.1 protocol using standard HTTP Extensions protocol [NN00]. The use of HTTP Extensions protocol assures that there will be no interference with other possible HTTP extensions and that CC/PP data that was attached to the HTTP request will be delivered to the other end without any damage even if gateways and proxy servers exist on the way. An example of extended HTTP request that uses HTTPExt protocol is shown on Figure 3.

CC/PP Description framework [RHDS99] is an RDF [Las97] based document specification which allows to describe various capabilities of the client's hardware as well as user customization profiles. CC/PP allows composing of capabilities from multiple sources. Each capability description must be publicly accessible via a unique URI, which is used as a capability identifier. Capability descriptions are used to describe all software and hardware aspects of the client's device, such as CPU model and speed, amount of installed memory, screen resolution and depth, version of the operating system and the client's software, and user preferences such as whether sound is turned on or off. It is expected that a "standard" description of the client would be provided by the manufacturer.

CC/PP Exchange protocol specification describes an elaborate system of caches so that capability descriptions don't have to be transfered from the repository for every incoming HTTP request. There are provisions in the specification for a user to send only those parts of the description they altered therefore differ from the standard description that was provided by the manufacturer of the client's device.

However, there are certain problems that arise with the use of the proposed CC/PP solution. In a mobile environment the first problem is that *CC/PP Description Framework* tries to describe every possible configuration of the client machine including all little details. Such fine grained descriptiveness seems of the very limited use given that it complicates the development of the web services, because the web server will have to fetch all the necessary descriptions that are specified by the client, apply the required profile differences "DIFFS" and maintain profile caches in order to reduce the performance hit created by downloading of the descriptions. In addition to that the web server has to analyze the received descriptions and provide the suit-

able content. Given the large number of variables in the CC/PP descriptions the corresponding number of transformations of any particular document is very large. This not only makes the transformation process very complex, it also makes caching of the transformed document less attractive in terms of performance. On the server side most details of this descriptions would likely end up being ignored, or would lead to extremely complex and hard to maintain web sites.

The fine granularity of CC/PP descriptions also contributes to the complexity of the client's software, because the web browser must be aware of all possible configuration changes on the client, and create the necessary "DIFFS" to standard configuration which requires relatively expensive computation of MD5 signatures for the original profiles. Given that this protocol is supposed to be used on mobile devices that have very limited resources, such complexity is at least inconvenient.

Finally, having a separate description for every single model of client device even if those devices are essentially the same in terms of their capabilities and connectivity (*e.g. Dell's notebook vs. Compaq Notebook, 3COM's Palm Pilot vs. Handspring's Visor*) will generate a lot of extra traffic and will require a lot of storage in the caches for fetching and storing duplicate capability descriptions, which is not desirable in wireless networks. In addition to that there is a considerable processing overhead on the server associated with merging the standard capability description of the client device with user supplied differences.

3. Problems with CC/PP

In this work, we present an alternative implementation of the CC/PP description framework which addresses the aforementioned problems. In our framework each device is classified, without loss of generality, to belong in one of four different categories. These are: Desktop or high end notebook with the broadband connection, Notebook on the road with dial up/wireless connection, Hand held computer with CDPD modem, or WAP device. These categories cover a vast majority of possible client resource and connectivity combinations. However, adding a few more categories if needed, or using a different set of categories, still does not affect the underlying idea behind our framework. The case we make is that a small fixed number of categories can cover a vast majority of client resource and connectivity combinations.

Such categorization allows for straightforward web site maintenance. A small number of versions of tailored content need be kept on the web site, and very simple transcoding rules are sufficient for generated content.

Tailored, pre created, content should be used in cases when the content is very different for different categories of clients. For example, the broadband version of the site might use elaborate scripts or plug-ins that are either impossible to transform to lower fidelity representations using universal rules (e.g. scripts), or cannot be so transformed in near real time (e.g. selecting key frames from a video). Another example would be images that contain high quality three dimensional corporate logos. Such images usually look very poor when transformed to lower resolution or color depth using dithering or similar technique, and should better be redrawn in flat two dimensional versions. Other transforms, such as converting markup from XML to HTML or WML can be done on the fly. This content should be generated the first time a request comes for it and then cached.

Since the number of possible device capability descriptions is very small, those capability descriptions could be very well represented with a single URI. Note that URI itself describes the capabilities of the device not the data located at that URI, so there is no need for generating extra HTTP requests for fetching the capabilities and there is no need for complicated and resource consuming capability description caches. However for of being compatible with the W3C specifications the actual capability descriptions can be placed at the locations specified by those URIs.

We found that CC/PP Exchange Protocol is adequate for our purposes so we adopted it as is for our implementation. However, due to simplified capability description approach many of the features of Exchange Protocol such as handling of DIFFs to profiles, support for different versions of the CC/PP descriptions and profile cache infrastructures are not needed and therefore they were not implemented.

4. Implementation

We implemented "Simple CC/PP" as an add-in module for the popular Apache Web Server. We choose Apache because of its widespread use, high performance and the ease of implementing add-in modules. Add-in modules have a number of advantages over popular CGI approach. Unlike CGI scripts add-in modules are integral part of the Web server process, therefore there is no need for time consuming processes execution for every incoming HTTP request. Another advantage of add-in modules which is extensively used in our implementation is the ability to control the behavior of the Web server in all stages of HTTP request processing.

The "Simple CC/PP" module presented in this work was implemented in Perl language using Apache's mod_perl module [SM99] which provides the interface between internal workings of Apache and a perl interpreter which is embedded into mod_perl module. Perl was chosen for relatively high speed of execution, excellent abilities of pattern matching and automatic memory management.

We considered to use Java servlet for implementing the simple CC/PP module, but decided not to do so because of very poor execution speed of Java for applications that are very dependent on the I/O performance. The study done by Kernighan and Van Wyk [KW98] shows at on average a typical I/O bound Java application is about hundred times slower then C or Perl version of the same program. Such inefficiency is caused by the design of Java I/O libraries. In addition to that the typical string manipulation program in Java is also very poor. The same study done by Kernighan and Van Wyk shows that Java version of a typical string manipulation program is about ten times slower then Perl's counterpart. Since the performance of input/output and string manipulation operations is very important for this application we chose not to use Java.

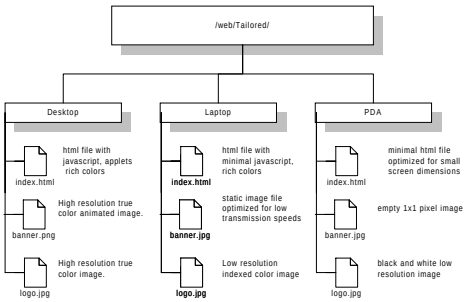

Figure 1. Example of Tailored content tree

4.1. Web site setup

To use the "Simple CC/PP" module the Web server has to be set up in a certain way. First the web master has to decide which content will be tailored and which content will be generated. A document should belong to tailored content section if there are separate pre-made representations available for each type of the client. For example a map could be drawn in full color for desk tops, grayscale for laptops to minimize the use of bandwidth, and made black and white and reduced in size and/or resolution for PDAs. If document does not need custom modifications then it should be placed into generated content section. A good example of a generated content document is a phone directory of a small group, which is made flashy and colorful for laptops and desk tops and simple plain table for PDAs. Such page could be encoded as XML document with two different XSL style sheets one for desktop and laptop clients and one for PDAs.

After it's clear which documents would go into tailored content category the web master should create a root directory for the tailored content documents and also create subdirectories for each type of tailored content. One possible tailored content tree is shown on Figure 1, this setup contains three different tailored content subdirectories one for Desktop computers, one for Laptops on a wireless link and the one for PDAs. The Desktop subdirectory contains interactive and multimedia rich content with javascript applets and true-color animated images, the subdirectory for laptops contains the content that is optimized for transferring over slow wireless links in particular the color depth and the resolution of the images is reduced, animated images are downgraded to simple static images most of the applets and heavyweight scripts are dropped. And finally the PDA directory contains very minimal HTML files that are convenient to see on the small PDA screens, all the images are either dropped or reduced to bare minimum black and white low resolution files. All of these three tailored content directories are placed under `/usr/web/tailored` directory which is the root directory of tailored content tree.

The setup of generated content section is even easier then tailored content section. First the web master should mark up all the documents that belong to the generated content section in XML. Then she must create XSL style sheets for each type of the supported device. The style sheets should follow the same guidelines as the tailored content files: media and color rich for the desktops, bandwidth optimized for the laptops and screen size and bandwidth optimized for PDAs.

In order for the "Simple CC/PP" to function properly the Apache web server must have `mod_perl` module and `XSLT` content generation package [Cla99]. The `mod_perl` module provides embeds the PERL interpreter and provides all the necessary server API support for perl modules. The `XSLT` content generation package provides the support for handling XML/XSL documents and delivering them back to the client.

4.2. Module internals

The "Simple CC/PP" module is usually set-up to handle all requests that refer to the documents under "/CCPP" directory of the web server. All the documents under this directory are said to belong to the "virtual CC/PP space". All the work done by the "Simple CC/PP" module happens during content generation phase of the HTTP request processing. At the beginning of this phase the request headers have been successfully processed, all necessary authentication have been performed and the module that is supposed to generate a content is identified. So the work that's required to do by our module is to check for the presence of HTTP-Ext related headers and if such headers are present then try to extract "Simple CC/PP" related information from

them. If “Simple CC/PP” related information is present in the HTTP-Ext headers the module replaces the virtual CCPP portion of the incoming URI to refer to the document that is specific to this particular device. In the case when “Simple CC/PP” information is not present in the request headers, the module changes the URI to refer to the desktop variant of the document.

After URI has been changed to refer to the document that is appropriate to be displayed on the device that sent this particular request the “Simple CC/PP” module performs an internal redirection of the request. The internal redirection is similar to the standard HTTP redirection but it is performed entirely inside of the web server process so that no extra communication between client and a server takes place. Moreover the client is not even aware that redirection even took place. From the client’s perspective the URI of the documented returned by the server is the same as the URI of the requested document.

The “Simple CC/PP” module uses the following logic to select the proper URI for redirection. After receiving the URI and the type of the client’s device it replaces the portion of URI that refers to virtual CC/PP subdirectory with the subdirectory that is refers to tailored version of a document for this device and then checks if such version of the document does exist. If the tailored version of the document indeed exists the module selects the rewritten URI as a target for internal redirection. Otherwise it checks if it possible to generate the requested document from XML/XSL files. If it is possible then it construct the URI which refers to `XSLT` servlet and contains the references to document template and XSL style sheet file that is appropriate for the client’s device. The resulting URI is used as a target for internal redirection. After module finally decided on the appropriate target URI it instructs the web server to retrieve the document referred by the target URI, perform all necessary actions like running scripts, processing server side includes and executing proper servlets, and then return the document back to the user as if it was the located document in the virtual CCPP space. Note that the client is not aware of the true location of the requested document, this implies that all the absolute references to the documents in both tailored content sections and XML document sections must refer to the documents as if they were part of the virtual CCPP space. If neither tailored version of the document nor XML/XSL template exist then the module instructs the web server to generate “404 Not found” response.

5. Results

To test the “Simple CC/PP” module we used three different client programs: Netscape Communicator 4.5 with a custom proxy that inserts “Simple CC/PP” related headers into incoming request and forwards it to the appropriate server, an open source version of Netscape’s browser (Mozilla) modified to send “Simple CC/PP” profiles with each request and Netscape Communicator 4.5 talking directly to the web server for insuring a correct behavior of the module in the absence of “Simple CC/PP” related information.

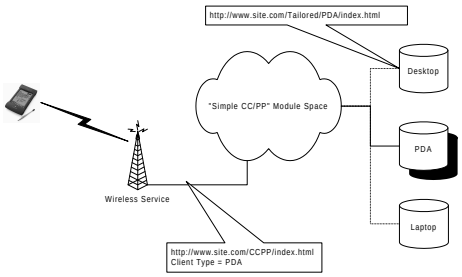

Figure 2. Example Request of tailored data for PDA

```
GET / HTTP/1.1
Content-Length: 421
Opt: "http://www.w3.org/1999/06/24-CCPPexchange"; ns=15
15-Profile="http://www.cs.umbc.edu/SCCPP/PDA"
.......
```

Figure 3. Example of Extended HTTP request with the PDA profile

For tailored content we used UMBC’s web site in three different versions. For resource and connectivity rich clients we used unmodified site with a lot high resolution images, and extensive interactive javascript content. For connectivity poor clients but resource rich clients such as laptops we replaced all color images on the web site with their grayscale counterparts. And for PDA’s which are resource and connectivity poor stripped all the images from the web site together with interactive content. The relative sizes of different versions of the web site and their relative transfer times are shown in the table 1.

For the purposes of performance testing we used several custom perl scripts that send a continuous stream of HTTP requests to the server, receive documents sent by the web server and then simply discard received documents. The documents were discarded without any processing because we are interested only in the delays caused by the

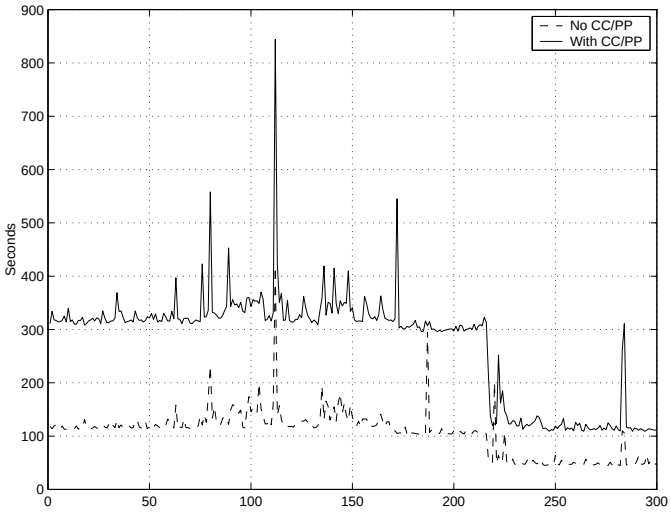

Figure 4. Request time with and without CC/PP (Plain HTML)

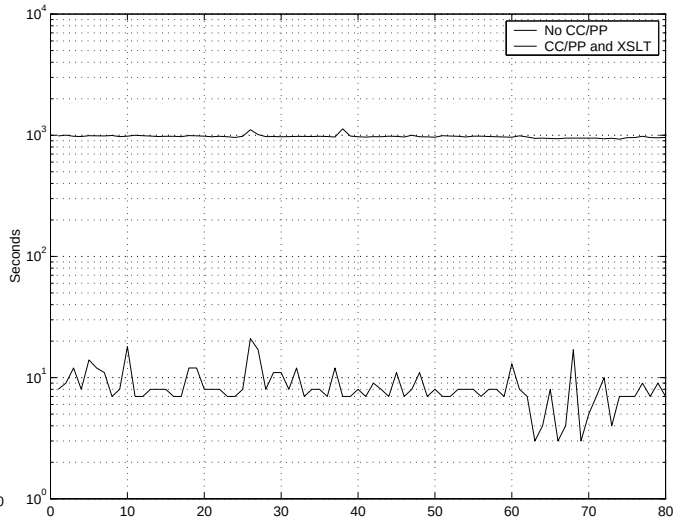

Figure 5. Request time with and without CC/PP (XML Documents)

network transmission and processing of the request by the web server. The transmission times shown in the table are given for the plain request, without CCPP module. These measures represent the case when both client and server machines are connected to the same local network.

Site	Size	Transfer time
Unaltered	1,956 kb	9.74 sec
Laptop	1,892 kb	8.50 sec
PDA	850 kb	3.34 sec

Table 1. Relative sizes and transfer times for different versions of test web site

To measure the performance overhead created by our "Simple CC/PP" module we used aforementioned perl script that sends out continuous stream of HTTP requests together with perl's `Benchmark` module. The `Benchmark` module provides mechanism to measure running time of the perl script. We performed two series of measurements. In the first series we measured the time it takes to download all the objects on the main page of our web site directly from the web server without the use of CC/PP protocol (CC/PP headers were present, but CC/PP module did not handle the request.) In the next series we performed the same experiment, but this time CC/PP module participated in handling of the request.

The web server was running on a Linux machine with Pentium 200MHZ CPU and 128 of ram. We choose such low power machine for the server so that performance overhead associated with extra processing could be observed more easily. The client script was running on the Silicon Graphics O_2 machine with 200MHZ R5000 CPU and 128MB of RAM. Each test in the series was made of three hundred chunks. Each chunk contained hundred requests for the main page of the website. We run both series of tests at the same time by running one chunk at a time for each series alternatively. This was done in order to minimize inaccuracies caused by spurious network traffic and system CPU activity. Also before we run these tests, we executed several chunks of each test without recording the time. This was done in order to reduce inaccuracy created by various caches like file system cache, DNS cache, ARP cache etc. We chose not to record this data because it is very hard to insure that all the caches are empty, unless all the tests are executed in tightly controlled environment. As an extra precaution we disabled the child spawning behavior of the Apache web server in order to minimize time measurement error created by the overhead of spawning extra process.

The results of the test are shown on Figure 4. The X axis of the graph represents the chunk number of the test. And the Y axis of the graph represents the time in seconds it took for the chunk to complete. The graph shows that it take approximately three times longer for CCPP request to complete compared to the request without CCPP headers. This is so because each CCPP request is essentially two requests one request is the request from the client to the web server that ends up in "Simple CC/PP" module. And the other request is the request send from "Simple CC/PP" module to the web server which generates the actual document that is sent to the client. Since it took about 31 hours for the whole test to complete and the testing was done on

the shared Ethernet segment, all the tests were suspectible to errors caused by various factors such as network traffic created by other users and CPU activity created by the network traffic etc. Since the testing started on Friday afternoon, by the time $2/3$ of the tests was completed it was a Saturday night and most of the system activity has dropped severely. This could be seen on the graph that the running times of the test has severely. Also it should be noted that activity drop created more severe impact on the CC/PP requests then on the then on the non CC/PP request. This can be explained by the fact that handling of CC/PP request requires more CPU time then handling of plain requests and that activity affected the available CPU cycles on the web server more then the network traffic.

In addition to measuring time overhead created by the handling of CC/PP request, we performed measurements of time overhead created by the processing of XML documents. Since we did not have access to an actual website that uses XML/XSL we used a small number of non web related XML files from the Centaurus project, as well as a simple XSL transform stylesheet from that formats any given XML document using and performs some color highlighting of different elements of XML documents. This stylesheet comes as a part of `XSLT` package.

As in the first set of tests we had two series of request. For the first series of tests we requested two XML documents using straightforward HTTP without CCPP protocol. For the second series of tests we requested the same documents using extended HTTP protocol with CC/PP headers. All the requests in the second series were handled by the "Simple CC/PP" module and `XSLT` servlet. The same perl scripts that were used for measuring the time overhead created by "Simple CC/PP" module were also used for in this set of tests as well. The machines for the client and the server and the measuring techniques were also identical to the those in the first series of tests. Because in this series of tests we were interested only in the time overhead created by the XML to HTML transformation, all the caching behavior of the "Simple CC/PP" module was disabled. The results of the tests are shown on the Figure 5. Because of the significant overhead created by the XML to HTML transformation the logarithmic scale was used. Also due to the fact that it takes so much longer for the request that involves XML to HTML transformation to complete we limited the number of tests in the series to 80 instead of three hundred as it was in the first set.

The low performance of the XML to HTML transformation is caused by several factors. One of them is the performance of Java programs in general, which was discussed in section 4. Second factor is the method of XML to HTML transformation which requires the all XML files and stylesheets to be loaded into memory before transformation can begin.

Although such low performance might seem as not acceptable from the first sight, it should be noted that the all these tests were done on the very low performance web server. Moreover, this extra overhead is a tradeoff which is more than compensated by the gain in time taken to transfer the smaller document on the slow wireless networks (especially in CDPD and WAP), and the reduced consumption of battery power by the client in receiving and processing the simplified data. Of course, in certain cases described above if XML to HTML or WML transformation were not done, the client wouldn't be able to display the document at all.

References

[BJA98] Harini Bharadvaj, A. Joshi, and Sansanee Auephanwiriyakyl. An active transcoding proxy to support mobile web access. In *Proc. IEEE Sumposium on Reliable Distributed Systems*, October 1998.

[BKC+98] E.A. Brewer, R.H. Katz, Y. Chawathe, A. Fox, S.D. Gribble, T. Hodes, G. Nguyen, T. Henderson, E. Amir, H. Balakrishnan, A. Fox, V. Padmanabhan, and S. Seshan. A network architecture for heterogeneous mobile computing. *IEEE Personal Communications Magazine*, 5(5):8–24, 1998.

[BMK97] R. Barrett, P. Maglio, and D. Kellem. A confederation of agents that personalize the web. In *First Intl. Conf. on Autonomous Agents*, Marina Del Ray, CA, 1997.

[BMMM96] C. Brooks, M. S. Mazer, S. Meeks, and J. Miller. Application-specific proxy servers as http stream transducers. In *Proc. WWW-4, Boston, http://www.w3.org/pub/Conferences/WWW4/Papers/56Application-Specific*, May 1996.

[BSM96] Tim Bray and C.M. Sperberg-McQueen. Extensible Markup Language (XML). (W3C Note, 14 November), 1996.

[Cla99] James Clark. XSL Transformations (XSLT) Version 1.0. W3C Recommendation 16 November, 1999.

[Cor] IBM Corporation. Ringing in wireless services: Web access without wires. http://www.ibm.com/Stories/1997/08/wireless4.html.

[ea] B. D. Noble et. al. Agile application-aware adaptation for mobility. In *Proceedings of the 16th ACM Symposium on Operating System Principles*.

[FB96] A. Fox and E. A. Brewer. Reducing www latency and bandwidth requirements by real-time distillations. In *Proc. Fifth International World Wide Web Conference*, May 1996.

[FGG+98] A. Fox, I. Goldberg, S.D. Gribble, D.C. Lee, A. Polito, and E.A. Brewer. Experience with top gun wingman: A proxy-based graphical web browser for the usr palmpilot. In *Proc. IFIP International Conference on Distributed Systems Platforms and Open Distributed Processing (Middleware '98)*, 1998.

[FZ94] G. Forman and J. Zahorjan. The challenges of mobile computing. *IEEE Computer*, 27:38–47, April 1994.

[HH99] O Hidetaka and J Hjelm. CC/PP exhange protocol based on HTTP Extension Framework. (W3C Note, 27 July), 1999.

[JdT+95] A. D. Joseph, A. F. deLespinasse, J. A. Tauber, D. K. Gifford, and M. F. Kaashoek. Rover: A Toolkit for Mobile Information Access. In *Proc. 15th Symposium on Operating Systems Principles*. ACM, December 1995.

[Jos00] Anupam Joshi. On proxy agents, mobility and web access. *ACM/Baltzer Journal of Mobile Networks and Applications*, 5(4), 2000.

[JWH97] A. Joshi, S. Weerawarna, and E. N. Houstis. On disconnected browsing of distributed information. In *Proceedings of the seventh International workshop on Research Issues on Data Engineering*, pages 101–107. IEEE Press, 1997.

[JWM+96] A. Joshi, R. Weerasinghe, S. P. McDermott, B. K. Tan, G. Benhardt, and S.Weerawarna. Mowser: Mobile platforms and web browsers. *Bulletin of the IEEE Technical Committee on Operating Systems and Application Environments*, 8(1), 1996.

[Kat94] R. Katz. Adaptation and Mobility in Wireless Information Systems. *IEEE Personal Communications*, 1(1):6–17, 1994.

[KBA+96] R. H. Katz, E. A. Brewer, E. Amir, H. Balakrishnan, A. Fox, S. Gribble, T. Hodes, D. Jiang, G. T. Nguyen, V. Padmanabhan, and M. Stemm. The bay area research wireless access network (barwan). In *Proceedings Spring COMPCON Conference*, 1996.

[KW98] B. W. Kernighan and C. J. Van Wyk. Timing trials, or the trials of timing: Experiments with scripting and user-interface languages. *Software Practice and Experience*, 28(8):819–843, July 1998.

[Las97] Ora Lassila. Introduction to RDF Metadata. (W3C Note, 13 November), 1997.

[LKR96] M. Liljeberg, M. Kojo, and K. Raatikainen. Enhanced services for world-wide web in mobile wan environment. http://www.cs.Helsinki.FI/research/mowgli/mowgli-papers.html, 1996.

[NN00] S. Lawrence N. Nielsen, P. Leach. HTTP Extension Framework. (RFC2774), 2000.

[RHDS99] F Reynolds, J Hjelm, S Dawkins, and S Singhal. Composite Capability/Preference Profiles (CC/PP): A user side framework for content negotiantion. (W3C Note, 24 June), 1999.

[SM99] Lincoln Stein and Doug MacEachern. *Writing Apache Modules with Perl and C*. O'Reilly and associates, Sebastopol, CA, USA, 1999.

[SSK97] S. Seshan, M. Stemm, and R. Katz. Spand: Shared passive network performance discovery. In *Proc. 1st Usenix Symposium on Internet Technologies and Systems (USITS '97*, 1997.

[ST93] M. Spritzer and M. Theimer. Scalable, secure, mobile computing with location information. *Comm. ACM*, 36:27–27, 1993.