

A CC/PP aware Apache Web Server

Christos Papachristos, Evangelos Markatos¹

Institute of Computer Science

Foundation for Research and Technology

1. Introduction

While current Internet is looking for standards that it can be based and grown upon them in a stable manner, different protocols have been suggested by several consortiums and groups. The Internet of the future will likely contain not only the standard PCs and portable computers but many other devices as well. We have already seen the mobile phones that are trying to access the world of Internet despite their limited capabilities in power and size.

In order for such mobile devices to access the Internet a new protocol must be used. The weaknesses mentioned, about the low power and limited size that such devices may have forced us to treat them in a different way depending on their capabilities. This can be explained by the following example.

When a PC (a client) is requesting a web server for a web document the web server will return that document. When another device with some different capabilities (mobile) is requesting the same document as in the previous case the interaction between the client and the server should not be the same. Suppose that the web document contains large images. In the first case, the computer has a screen capable of presenting such images. In the latter case however, the mobile phone has a much smaller screen incapable of handling those large images. Such devices need to be integrated with the current Internet in a way that they receive the best result depending on their capabilities. For that reason, we need a protocol that will take care of the interaction between clients and servers and make servers 'recognize' the kind and the capabilities of the client of each incoming request.

Such protocol, that could provide the web servers with the information of what kind of client performs a request each time, is the new CC/PP (Composite Capabilities/Preferences Profiles) protocol. This protocol is suggested as a standard from the W3C group [WWW] (<http://www.w3c.org>).

2. Related Work & Possible Solutions

Currently there are not many implementations that can support the CC/PP protocol. A web server at the moment that is capable of handling HTTP requests containing CC/PP headers is the Jigsaw web server implemented by the W3C. Other implementations have been developed by the Keio University [KY] and the University of Wales [STL]. Researchers from Keio University have developed their own browser which supports the CC/PP protocol (called PANDA) and they make use of a CC/PP aware server proxy to communicate with the web server.

In the second case of University of Wales a server proxy and a browser proxy have been used in order to test the functionality of the CC/PP. The browser proxy was used to help current browsers and devices (without CC/PP) interact with the CC/PP compliant part of the system designed. The server proxy was in front of the web server containing the data requested, and it

¹ {cpapachr,markatos}@ics.forth.gr

was responsible of requesting the data with traditional HTTP headers without CC/PP. The communication with the HTTP+CC/PP headers is held between these two proxies.

There is also another application developed in Monash University that can serve a mobile device with content read from a file [LT]. The system takes as input a document in MS excel format. It then applies some constraints, that specify the layout of the output document. Depending on the CC/PP attributes read from .ccpp file it produces a file that can be shown in a PalmOS emulator. The format of the file containing the CC/PP attributes is conformant to the W3C suggestions about the format of the Profile header. In this approach the CC/PP information is not sent by the client using a known protocol like HTTP. HTTP protocol may be involved in a newer version.

Four different approaches can be used in order to provide CC/PP functionality to a client:

1. The first approach is implemented with two intermediary proxies, a client proxy and server proxy. In this architecture CC/PP information is only transferred between these two proxies and they are responsible to perform the protocol translation in order the end-to-end (client through server) communication to be achieved (Figure 1). Having these two proxies between the corresponding client and the server may make the communication slow. There is also the issue that adding more intervening parts in the system is like adding more failure points to the round-trip request-response scenario and becomes less flexible in possible future changes.

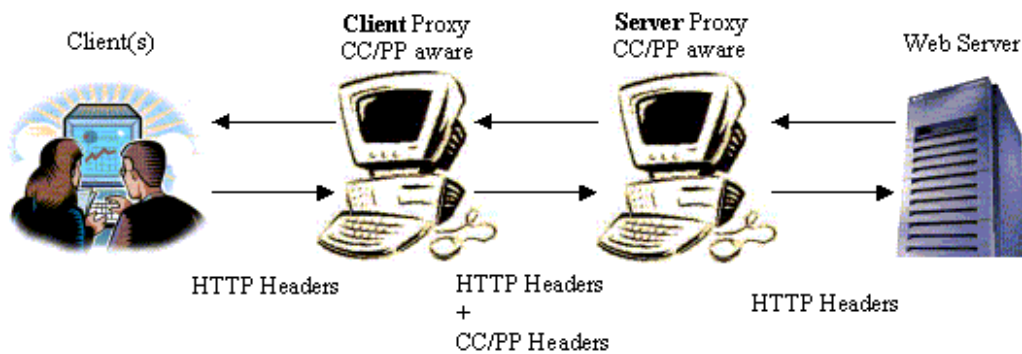


Figure 1: Architecture using two edge proxies supporting CC/PP communication between two hosts.

2. The second approach adopted from the Keio University includes one intermediary proxy, a server proxy. The PANDA browser implemented is CC/PP aware, so the client proxy for the translation between HTTP to HTTP+CC/PP and vice versa is not needed (Figure 2). In this approach the new browser implemented helped the researchers to test the functionality of the CC/PP protocol. Having only one proxy is a common case as this architecture is widely used in current Internet. The difference is that proxy's current usage is focused on serving clients with the cached documents so they are placed closer to the clients and away from the server in order to minimize the transfer delay of the documents. Locating a proxy away from the client may not have the same effects.

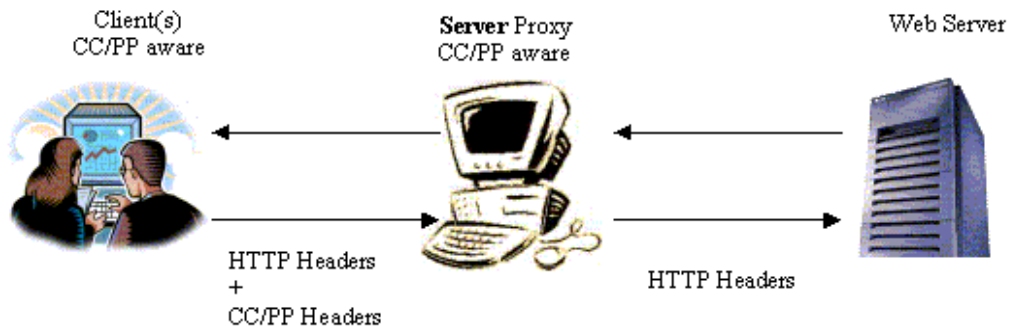


Figure 2: Communication between a client supporting CC/PP and a server that does not through a server proxy.

3. The third approach is to implement a CC/PP aware web server communicating with an intermediary client proxy, as seen in Figure 3. That means, that a web server must be capable of handling HTTP with CC/PP headers. The client proxy is used to translate the HTTP requests of each client to new HTTP requests including CC/PP information.

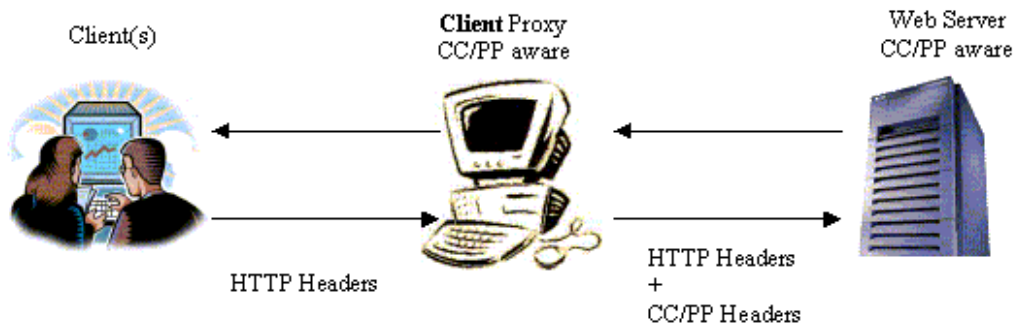


Figure 3: Communication between a web server supporting the CC/PP protocol with a client not supporting CC/PP. Usage of client proxy.

An alternative is to enhance the functionality of the client proxy in a client patch that will perform this translation as shown in Figure 4. This can give each client the opportunity to test if the CC/PP functionality provided by this new protocol offers any advantages in his/her daily navigation through the World Wide Web.

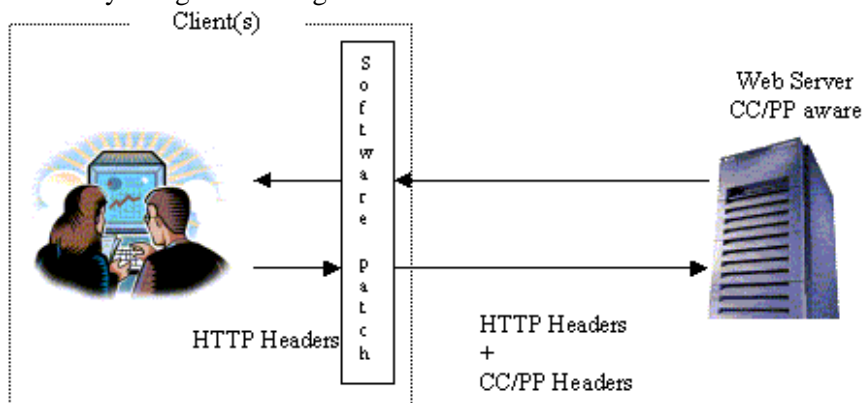


Figure 4: Communication between a configured Apache to support CC/PP and the usage of a software patch in the client side.

4. At last, there is the ideal approach of both server and client capable of sending and receiving CC/PP headers. Though is the best approach it is better to make one stable step each time and pass through some of the mentioned approaches (Figure 5).

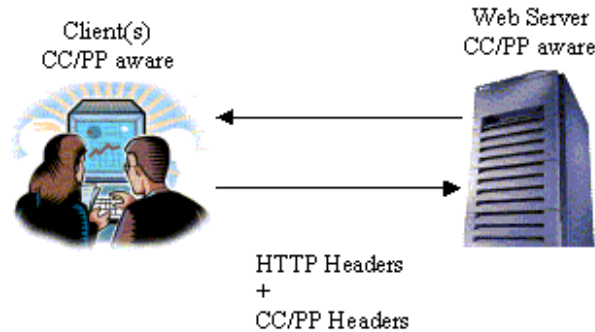


Figure 5: Ideal communication between two CC/PP compliant end hosts.

3. Functionality of proxies and software patch.

3.1 Client Proxy and Software Patch.

A client proxy as shown in the figures 1 and 3 communicates with the web client with the standard HTTP protocol. The functionality of the client proxy is to convert the standard HTTP headers sent by the web client to extended HTTP headers that contain CC/PP headers (HTTP headers + CC/PP headers). The new headers are then sent to the appropriate web server. If the web server is CC/PP aware then the headers are sent directly to that server. Otherwise the headers will be sent to a server proxy to make the appropriate translation.

3.2 Server Proxy.

A server proxy is used in implementations where the web server has not been transformed to support the CC/PP protocol. Its functionality is to:

1. perform any necessary parsing in the headers (the CC/PP part)
2. send the parsed headers to the appropriate web server (only HTTP headers)
3. convert the returned results from the web server according to the CC/PP headers that were previously parsed
4. send back the reply to the client. The client could be the browser or a client proxy.

A client proxy and a server proxy have similar functionality except that they communicate with different end-parts, the client and the server respectively.

4. Our approach

Our approach is based on the usage of a client proxy and is focused on integrating the server's proxy functionality inside a web server as a module. The whole system includes a client proxy, a web client (e.g. iexplorer netscape) and the transformed server with the new module (Figure 6).

A well-known web server that is open source and is widely used is the Apache web server. These features made our choice easier between other existing web servers. The API (Application Programming Interface) provided from the Apache web server enables the implementation of new modules. We chose not to consider the issue of the client part as we wanted to focus on the server-side. Instead of trying to extend a well-known client with the CC/PP protocol, we decided to use a simple client proxy that could do the appropriate work of the transformation between original HTTP and extended HTTP (CC/PP headers included).

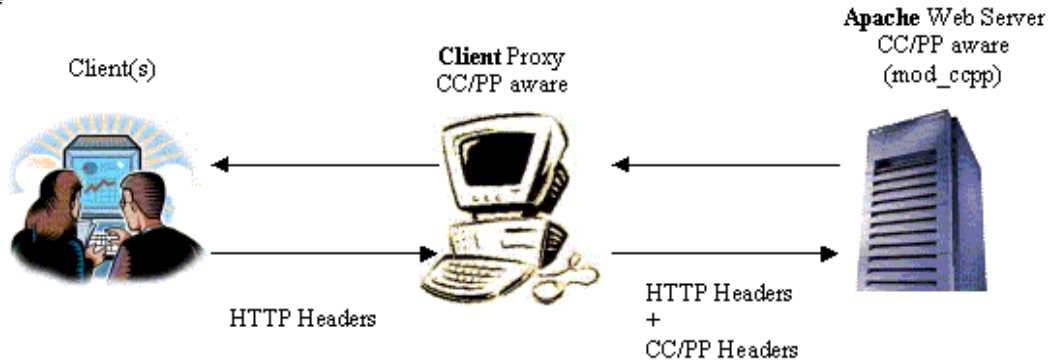


Figure 6: The communication between a CC/PP aware Apache server with a client proxy

The full functionality of a server proxy can be incorporated into the Apache web server. With regard to the client side, we developed a proxy that mediates between the Apache server and the web browser. In a future version we may try to transform the client proxy into a software patch (Figure 4).

Till now we have implemented a module for the Apache server that handles the CC/PP headers. This module is called `mod_ccpp` (`mod_ccpp.c`). It can be imported to the Apache like all the other extra modules with the usage of `'--activate-module'` directive. This can be done in the initial configuration of the server. In order to test our module we have linked it with the [ImageMagick](#) (an image processing tool) library which can be used to transform images. ImageMagick library provides an API to transform images between formats and sizes. The generated web page is now dependant on the client's device characteristics sent in the CC/PP header.

Including that library inside the module makes the deployment of the module little more complicated. One automatic generated Makefile had to be altered in order to link the module with the ImageMagick library. Having installed the module, we were able to make requests to the Apache with various values for the CC/PP header fields. The server responded as expected in all cases.

As future work we may try to develop a client patch that could enable a browser (Mozilla) to sent CC/PP headers and avoid the usage of the client proxy. Some measurements will be also held to determine the overhead of the CC/PP headers processing and the image transformation.

5. References

1. [LT] Lawrence Teo, "Constraint-Based Document Layout for Mobile Computers", http://www.csse.monash.edu.au/projects/MobileComponents/projects/pda_doc_layout/report.html

2. [WWW] World Wide Web Consortium, Composite Capabilities/Preference Profiles Working Group, <http://www.w3.org/Mobile/CCPP/>

3. [KY] Kinuko Yasuda, "Implementation and Evaluation of Keio CC/PP Implementation", <http://yax.tom.sfc.keio.ac.jp/panda/slidemaker/0011ccpp/Overview.html>

4. [STL] Stuart David Lewis, "Content Negotiation for Varying Web Enabled Devices", <http://www.ccpp.co.uk/>