Reasoning In CC/PP

Youyong Zou yzou1@cs.umbc.edu
Cassie Thomas cthoma10@gl.umbc.edu

Abstract:

This project is about W3C draft CC/PP protocol. It implemented a prolog based inference engine which can match and transform CC/PP profile.

1. Introduction

1.1 What is CC/PP?

Composite Capabilities/Preference Profiles **(CC/PP)** [1] is a working draft made available by the World Wide Web Consortium **(W3C)**, which specify how client devices (Web Browser) express their capabilities and preferences (the user agent profile) to the server that originates content (the origin server). The origin server uses the "user agent profile" to produce and deliver content appropriate to the client device. This allows for sophisticated content negotiation techniques between web servers and clients, to produce optimized XML-based markup for display and use on a wide variety of web user agents.

CC/PP is based on RDF[6] and designed to work with a wide variety of web-enabled devices, from PDAs to desktop machines to laptops to WAP phones to phone browsers to web television units to specialized browsers for users with disabilities.

1.2 How to use CC/PP

CC/PP is simply the language for describing what the user agent can do. This information would then be conveyed to the originating server as part of an HTTP request, and it's up to the server to decide how to use the user agent profile to best meet the needs of the user agent client. The two primary ways in which a profile might be used are **selection** and **transformation**. [7]

Selection is the process by which the originating server chooses an appropriate representation of requested web content from a finite set of existing representations.

Transformation, on the other hand, assumes that there is no finite set of representations, but rather than content is created on the fly, based on the properties expressed by the user agent profile. The content would be stored in an XML-compatible format and then transformed into an appropriate language (or modules thereof) that could be understood and optimized for the user agent, such as XHTML or WML.

CC/PP draft defined some use cases where CC/PP format can be used. Vladimir Korolev and Anupam Joshi described a "simple CC/PP" in the paper "An End–End Approach to Wireless Web Access", which implemented HTTP use case with repository (CC/PP: Requirement 2.1.3) and WAP use case (CC/PP: Requirement 2.2). Here, different devices will get HTML page with different size based on their profile.

1.3 CC/PP Need Reasoning

CC/PP is based on RDF, and RDF is for semantic web. So, the power of CC/PP is not in providing some profiles which machine can read, but providing the profiles which machine can understand. To understand, we need some inference engine which can read CC/PP profile, understand them.. RDF can be parsed into triples and loaded triples into Prolog. Prolog can THINK!. So, with some reasoning rules specified for CC/PP, we can have a “brain” for CC/PP. Selection and transformation of CC/PP can easily be done in this inference engine.

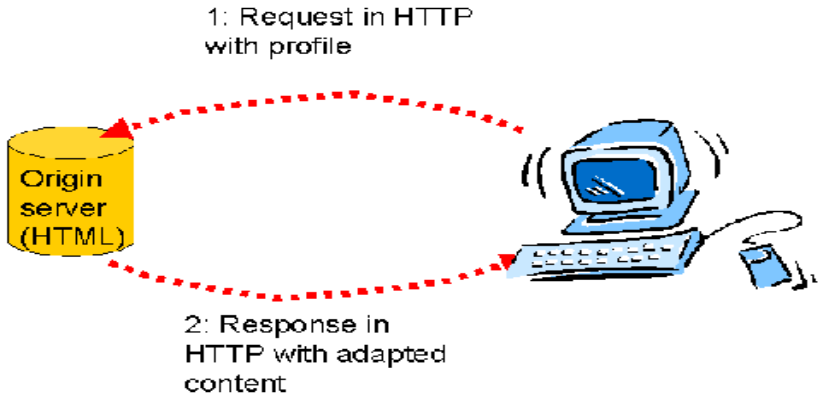
2. Design

We want to do the following implements of CC/PP :

2.1 Simple Web Case

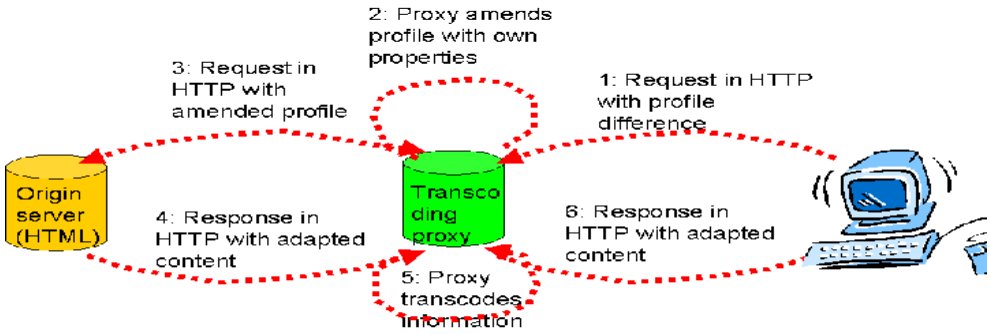
This use case in itself consists of two different use cases: The origin server receives the CC/PP profile directly from the client; the origin server retrieves the CC/PP profile from an intermediate repository.

The CC/PP Exchange Protocol creates a modified HTTP GET which carries the profile or the profile difference. The extended content negotiation with the CC/PP Exchange Protocol uses the CC/PP format to describe the device.



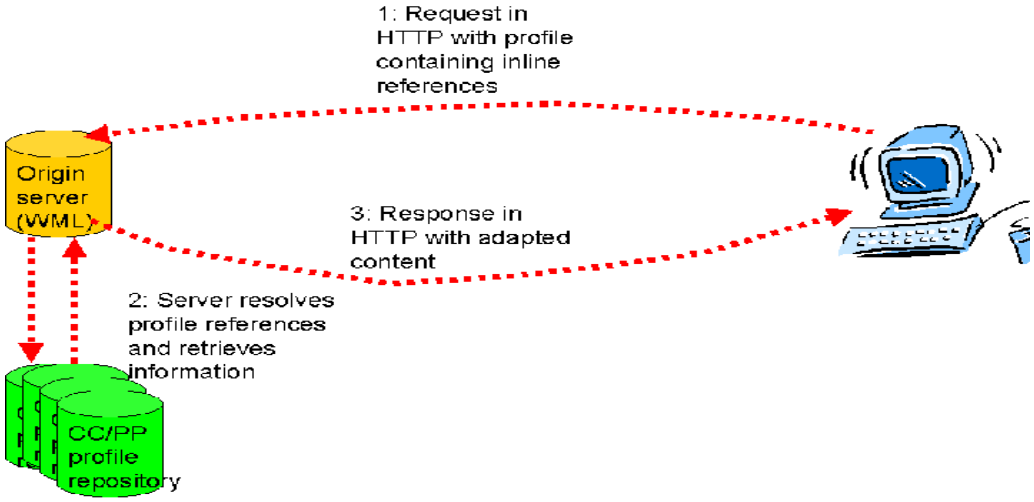
2.2 Intermediate Entity use case(CC/PP: Requirement 2.5 [1]):

Get request with profile difference, proxy amends profile with own properties and generate a amended profile. In this case, inference engine will be in proxy.



1. Match Two Profiles for Equality (CC/PP: Design Goal DG-6)

In this use case, web server will get HTTP request from one client with profile attached. Server can match the profile with the profiles already in database and find the best matching one. Origin server will then response with adapted content based on matching profile.



3. Implementation

3.1 Create CC/PP devices profile

Workstation (normal devices, high network bandwidth)

```
<Description about="http://www.example.com/MyProfile">
  <ccpp:component>
    <Description about="http://www.example.com/Hardware">
      <type resource="http://www.example.com/Schema#HardwarePlatform" />
       <uaprof:CPU>Sparc </uaprof:CPU>
       <uaprof:ScreenSize>1024*768</uaprof:ScreenSize>
    </Description>
  </ccpp:component>
```

```
<ccpp:component>
  <Description about="http://www.example.com/Software">
    <type resource="http://www.example.com/Schema#SoftwarePlatform" />
     <uaprof:OSName>Solaris </uaprof:OSName>
     <uaprof:OSVersion>2.6 </uaprof:OSVersion>
     <uaprof:OSVendor>SUN </uaprof:OSVendor>
   </Description>
</ccpp:component>
<ccpp:component>
  <Description about="http://www.example.com/Browser">
    <type resource="http://www.example.com/Schema#BrowserUA" />
     <uaprof:BrowserName>Netscape </uaprof:BrowserName>
     <uaprof:BrowserVersion>4.7</uaprof:BrowserVersion>
     <uaprof:CcppAccept>
       <Bag>
         <li>text/plain</li>
         <li>text/html </li>
       </Bag>
     </uaprof:CcppAccept>
   </Description>
</ccpp:component>
</Description>
```

Mobile devices(low network bandwidth):

```
<Description about="http://www.example.com/MyProfile">
  <ccpp:component>
    <Description about="http://www.example.com/Hardware">
      <type resource="http://www.example.com/Schema#HardwarePlatform" />
      <ccpp:Defaults rdf:resource="http://www.nokia.com/profiles/2000k" />
       <uaprof:ScreenSize>640x400</uaprof:ScreenSize>
     </Description>
  </ccpp:component>

  <ccpp:component>
    <Description about="http://www.example.com/Software">
      <type resource="http://www.example.com/Schema#SoftwarePlatform" />
      <ccpp:Defaults rdf:resource="http://www.Symbian.com/profiles/EPOC" />
     </Description>
  </ccpp:component>

  <ccpp:component>
    <Description about="http://www.example.com/Browser">
    <ccpp:Defaults rdf:resource="http://www.nokia.com/profiles/Mozilla" />
      <uaprof:CcppAccept>
```

```
            <Bag>
              <li>text/plain</li>
              <li>text/vnd.wap.wml</li>
              <li>text/html</li>
            </Bag>
          </uaprof:CcppAccept>
        </Description>
      </ccpp:component>
    </Description>
```

3.2 Web server and Proxy Server

Function of web server: accept HTTP request with profile attached, get profile from profile repository, response adapted content;

Function of proxy server : accept HTTP request, add CC/PP profile and send to origin server, get reply from web server, forward to client;

We plan to use Jigsaw [8] as web server. The Jigsaw software provides the Consortium's Java-based Web server. With a modular architecture and full HTTP/1.1 compliance, the Jigsaw server is a premier experimental platform for W3C and the Internet community.

3.3 Define Prolog rule for CC/PP profile match

We need to define some prolog rules about profile matching.

3.4 CC/PP Inference Engine

Generate triples from CC/PP profiles, output triples into Prolog. There are two options for inference engine:

- Use SiLRI [9] as CC/PP parser and reasoning engine: SiLRI is a main-memory logic-based inference engine implemented in Java. It implements a major part of Frame-Logic and has support for RDF.
- Use SirPAC as CC/PP parser, use Prolog as reasoning engine: SirPAC read CC/PP and translate into RDF triple. Prolog read in the triple and assert as term.

What ever option we use in our project, the Inference engine will run as a Resource in JigSaw.

4. Testing

We can test and compare of result requesting same HTTP from different devices (with different profile).

Prolog can be in Web Server or in Proxy, we can compare the difference in performance.

We can also analysis the difference possible implementation of inference engine: define Resource in Jigsaw; run as servlet in Web Server; Put into apache as module.

5. Timeline (Assume 8 weeks available)

Research and Design: 3 week

Coding: 2 week

Debug and Testing: 2 week

Writing paper, Preparing presentation: 1 week

6. Future

The CC/PP Working Group is watching the XHTML modularization process and the XHTML Working Group's document profiles.

Reference:

[1] CC/PP http://www.w3.org/Mobile/CCPP/

[2] W3C Mobile Mail list Archive http://lists.w3.org/Archives/Public/www-mobile/

[3] CSS Mobile Profile 1.0 http://www.w3.org/TR/2001/WD-css-mobile-20010129/

[4] Device Independence Working Group http://www.w3.org/2001/di/Group/

[5] "An End–End Approach to Wireless Web Access" by Vladimir Korolev and Anupam Joshi

[6] RDF http://www.w3c.org/RDF

[7] CC/PP http://www.ccpp.org

[8] Jigsaw http://jigsaw.w3.org/

[9] SiLRI http://www.aifb.uni-karlsruhe.de/~sde/rdf/