Position Aware WAP Content Personalization

Sergio Ríos, School of Computer Science,
Pontifical University of Salamanca in Madrid

This paper proposes a state-of-the-art framework for complete WAP content personalization, using network-based mobile positioning techniques and CPI/UAProf (Capability and Preference Information/User Agent Profile) data processing at the origin server.

Within this framework, in the dynamic generation of the response to the WAP device, the origin server uses Java Servlet code to access a Location Server in order to determine the geographic position of the mobile subscriber and, with a proper management of the CPI conveyed with the incoming HTTP request (for selecting or customizing the content being delivered to the client), provide him with location-sensitive content and services.

Introduction

Today, one of the most important technologies being developed around WAP is the physical positioning of mobile terminals requesting services, because it really opens new frontiers in contents personalization.

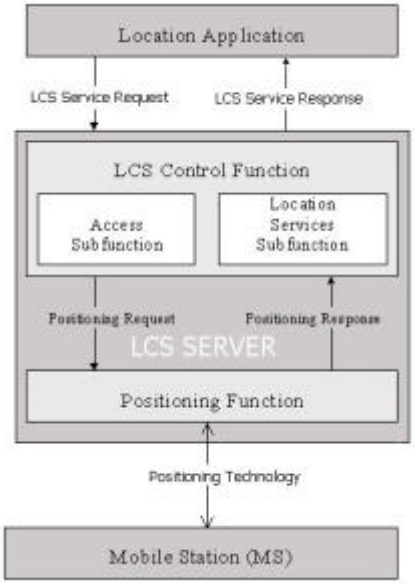
Here, I'll be presenting the fundamentals of Location Services and current location technologies. I'll go on to develop two real location-aware applications in Java, the first being a stand-alone location application for continuous position tracking of a mobile terminal, and the second a WAP-enabled Servlet for self and remote positioning.

As there aren't any standardized interfaces yet for Location Services within ETSI (European Telecommunications Standards Institute), the implementation of this framework uses Ericsson's HTTP-based Mobile Positioning Protocol (MPPv1.1) to request the position of mobile phones

Finally, in order to achieve a truly complete WAP contents personalization, I'll present UAProf, a framework that will allow us to know the particular characteristics preferred by the active user and the WAP mobile terminal.

LCS Logical Reference Model

In short, the logical reference model for LCS (LoCation Services) states that a LCS Client is enabled to request location information for one or more certain target Mobile Stations (MSs) from a LCS Server supported by a Public Land Mobile Network (PLMN). This is illustrated here.



The LCS Server, in turn, makes use of a particular positioning technology to obtain the location information and deliver the information to the LCS Client.

In terms of privacy, the particular LCS restrictions associated with each target MS are detailed in the Target MS Subscription Profile. So, the LCS server only responds to a location request from a properly authorized LCS client with location information for the target MSs specified by the LCS client if target MS privacy is fully satisfied.

Positioning Technologies: an Overview

Current physical mobile positioning technologies fall into terminal-based (handset-based) and network-based solutions. For mobile-assisted and mobile-based positioning methods, the target MS (the object to be positioned by the LCS Server) actively supports LCS, while for network based positioning methods, no support for LCS is required by the target MS.

A terminal-based positioning solution relates to positioning intelligence that is stored in the terminal or its SIM card. Examples of terminal-based solutions are the network-assisted global positioning system (A-GPS), enhanced observed time difference (E-OTD) and SIM Toolkit.

By contrast, network-based positioning solutions do not require positioning intelligence to be built into the handset (mobile terminal) so the location services can be deployed and marketed for existing handsets with no changes in hardware. Examples of network-based solutions include the cell global identity and timing advance (CGI+TA) and uplink time of arrival (UL-TOA) methods.

Before starting a description of these positioning technologies, it's necessary to address the question of privacy. It is pretty obvious that both positioning approaches can be used either severely compromising the users privacy, or, quite the opposite, providing assistance in those emergency situations where the user cannot request it.

In network-based systems the network operator is dealing with the position of each individual cellular phone, so it has all the position information. So, when the user requests position-dependent personalized information, privacy may be compromised or at least not fully guaranteed. In the case of terminal-based systems, the operator cannot actively deal with position of each individual cellular phone, because the localization is done in the mobile system and only due to explicit user requests.

Anyway, it is important to understand that in both cases we are talking about potential privacy violations by means of active tracking of user position, but recall that in the end, when the user requests location-aware contents, an application running on a HTTP server or an integrated WAP Gateway gathers position data from a LCS Server. This data can be collected in a user's history repository, as done in most commercial web sites, and that may be considered a passive form of privacy compromise.

Terminal-Based Solutions

A-GPS

In the Assisted GPS solution, a GPS receiver unit incorporated in a mobile handset receives signals from at least four GPS satellites. Each signal contains a time stamp and a description of the position of the satellite. Location is calculated either in the handset or by a central computer.

The GSM network can provide assistance information that gives A-GPS terminals much better coverage than stand-alone GPS receivers in those situations where the satellite signal is very weak or not available (for example, inside buildings).

E-OTD

In the Enhanced Observed Time Difference (E-OTD) method the signals from at least three base transceiver stations (BTS) are received by a terminal, which measures the time difference between arrivals of bursts of nearby pairs of BTS and reports them to the network, where a central computer calculates the desired location data.

Network-Based Solutions

CGI+TA

This positioning technology relies on two parameters, the Cell Global Identity (CGI) and the Time Advance (TA).

With respect to CGI, the mobile network host base transceiver station cell area is used as the position of the mobile terminal. Of course, this is a wide estimation and the obtained accuracy depends heavily on cell size (at best 150m in a "pico cell" to over 30km in rural areas) and type (a cell can be a circular or triangular sector).

On the other hand, the TA relates to a network-determined time difference of the actual arrival of a signal from a handset and the allocated time, and it is measured when a communication is to be established between a phone and the chosen cell. Simply stated, inside the controlled area of a serving cell, far-away mobile terminals must transmit their data some time ahead (according TA parameter) to keep in synch with the BTS.

In practical terms, the TA parameter is used to estimate the distance from the mobile terminal to the base transceiver station. Usually the time advance increment is 550 meters, but recent developments achieve much better granularity, with TA resolutions up to 16 times higher (≈30 m). With these improved resolutions, the precision of a combined CGI+TA system is quite good, typically offering an error boundary of less than 10m.

In any case, the estimated location is usually reported in the form of a longitude, a latitude, and an uncertainty region within which the mobile terminal has been located.

UL-TOA

The Uplink Time Of Arrival technique works by measuring the exact time of arrival of a mobile terminal radio signal at three or more separate base transceiver stations, thanks to special devices named LMUs (Location Measurement Units), installed at BTS, as an overlay to existing operator's cell networks.

Using the differences in arrival time at pairs of BTSs, received at a Mobile Positioning Center, it is possible to apply conventional triangulation techniques and calculate an area (the intersection of several hyperbolas) that represents the estimated location of the transmitting mobile terminal.

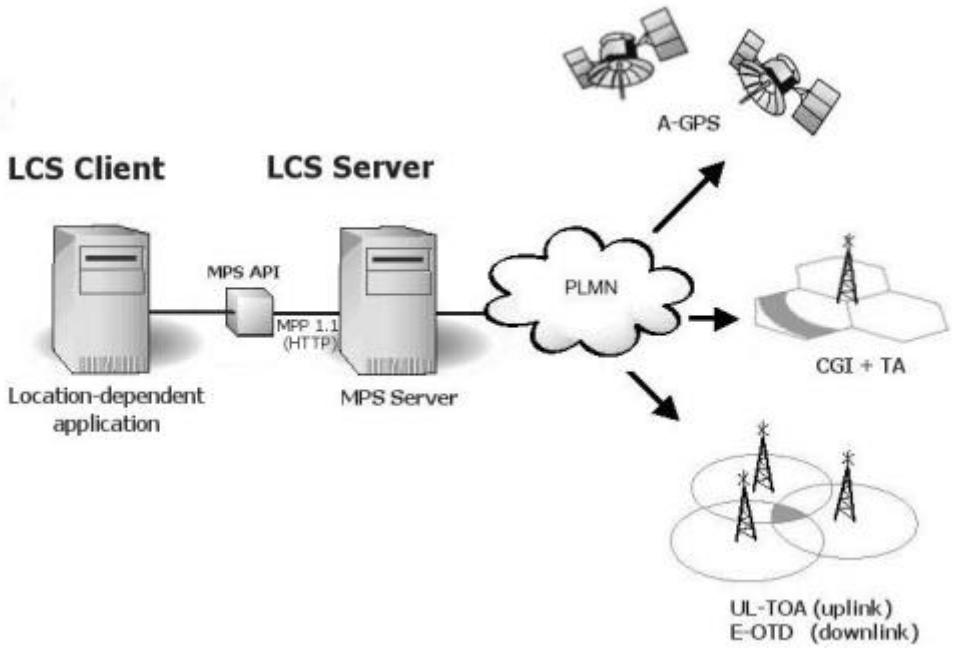
AOA

The Angle Of Arrival technique determines the direction of arrival of a handset's signal at the BTS, and it is based on LMUs, which measure the phase difference of the signal on elements of a calibrated antenna array mounted at the cell site. The intersection of the angles from two or more sites provides the location.

Mobile Positioning Protocol

The Mobile Positioning Protocol (MPP), currently in version 1.1, is an Internet-based protocol used by location-dependent applications to interface a Mobile Positioning Server. Through this protocol, it is possible to request the position of mobile terminals.

The Mobile Positioning System (MPS) is the particular Ericsson solution for providing Location Services (LCS) and, as part of this system, the Mobile Positioning Center (MPC) is the gateway between the mobile network and location-dependent applications, playing the role of the LCS Server component in the aforementioned LCS Reference Model. As expected, the MPC calculates the position of a mobile terminal based on information from the network, and delivers it to the application:



The important point to note is that the MPP offers a carefully designed generic interface towards the MPC, thus making the applications independent of the underlying physical positioning technology that is used. So, when new positioning technologies are developed, the location-dependent applications will immediately take advantage of them (for now, the MPC makes use of Cell Identification and Time Advance (TA) technologies for positioning).

Also, the MPP is fully based on HTTP, thus making the MPC available from any platform with TCP/IP capabilities.

Please note that the MPP is currently Ericsson proprietary, but as the standards for Location Services within ETSI will become firmer, this protocol will be changed to meet the standardized interface.

At this point, we'll look at the practical details of the Mobile Positioning Protocol so that we can develop true positioning applications in Java.

Requests and Responses

The Mobile Positioning Protocol defines an URL that location-dependent applications can use to request the position of a mobile station. As a response to the position request, the MPC delivers an answer telling the client positioning application the estimation of the mobile terminal location.

As this location data may be sensitive in terms of privacy, the MPC supports HTTPS (HTTP over SSL v2.0/3.0), thus making it possible to send secure positioning requests as well as receiving secure answers.

One important thing to note about requests, clearly remarked in the MPP specifications, is that a position request will not return until it has been executed, meaning that the HTTP connection will not be closed down until the answer is delivered. Also, a position request can be used to get the position of one or more mobile terminals at a time in only one HTTP transaction.

The MPP specifications define a set of parameters for a positioning request (in the usual name=value pairs form), and the mandatory ones are shown in the table below:

Parameter	**Description**
USERNAME	Name of the positioning user.
PASSWORD	Password of the positioning user.
POSITION_ITEM	The MSISDN of the mobile phone(s) to be positioned. The number must be complete, (consist of the country code, national destination code, and subscriber number). If more than one MS shall be positioned, the MSISDNs must be comma separated.
POSITION_TIME	Position requests only accepts (time+now) for this parameter, meaning that the positioning will be performed immediately.

According to the MPP specification, a response is successful if the position request has the correct syntax and passes the security validation process. However, under some circumstances, the positioning process may fail due to problems in the PLMN. In this case, the positioning fails, but the response to the request is still considered to be successful, since the request has correct syntax and has passed the security check.

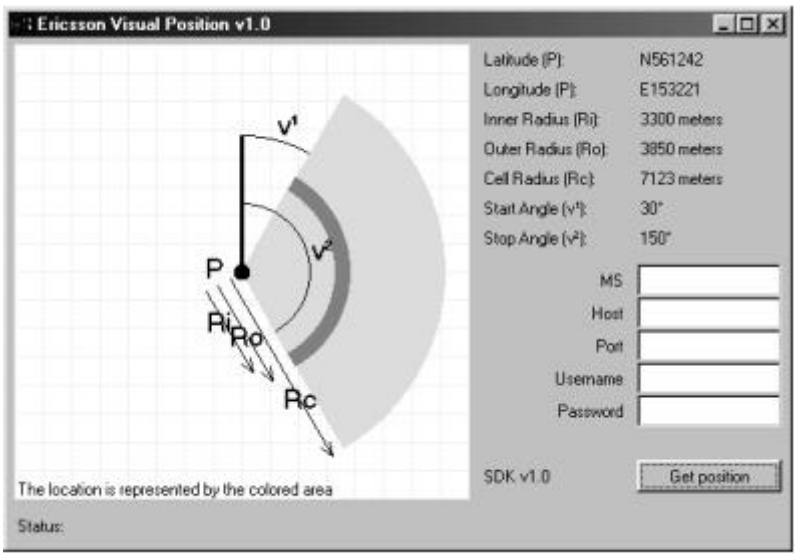
A position request may contain more than one mobile terminal to position, and therefore successful positioning may be mixed with unsuccessful positioning in the response.

The MPP response to a position request describes an area in which the mobile terminal has been located. As an example, a response may describe an arc — one of the best types of results in MPP in precision terms — using the coordinates (latitude, longitude) of a reference point, an inner and outer radius, and start/stop angles.

Here is an example of the plain text delivered by the MPC over HTTP in a successful answer to a position request:

```
<Head RequestID=2.873191662.1 AnswerID=1>
<MS=3452547198
RequestedTime=20001219173520+0200
Error=0
GeodeticDatum=WGS-84
HeightDatum=NotAvailable
CoordinateSystem=LL
PositionFormat=IDMS0
   <PositionData
    <PositionArea
     Time=20001219173520+0200
      <Area=Arc
      <Area=Point
       Latitude=N561238
       Longitude=E153755
       >
       InnerRadius=9900
        OuterRadius=10450
        StartAngle=30
        StopAngle=150
       >
       >
>
>
<Tail RequestID=2.873191662.1>
```

What follows is a snapshot of an application included in the MPP Software Development Kit (SDK). This application depicts in an easily understandable manner the position area data received in the former MPP response example:



Java Classes

The MPS SDK version 1.0 offers a powerful set of Java classes that provide a high-level abstraction of the Mobile Positioning Protocol, hiding the HTTP transmission of position requests sent and responses, and dealing with all necessary data parsing.

So, by using of this class library (`mppsdk` package), it is possible to develop positioning applications with little or no knowledge about the Mobile Positioning Protocol. Also, as an added value, the use of these classes protects the applications from changes in the syntax of MPP requests and responses.

The table below offers a description of the four classes involved:

Class	Description
`Coordinate`	Describes a point in which the positioning data is relative to
`MPSException`	Exception class thrown by `mpssdk` when errors occur
`PositionRequest`	Used to get the position of one or more mobile stations
`PositionResponse`	Contains the positioning data of the requested position item generated by `send()` method in class `PositionRequest`

First Code Example: A Stand-Alone Positioning Application

In this first code example I'll develop a stand-alone application, which performs a continuous tracking of the position of a mobile terminal.

Conceptually speaking, this application fits quite well in the role of the LCS client in the architecture described above. This means that the application must be executed in a Java Virtual Machine running on a system with any kind of direct HTTP connection to the MPC.

This code is fairly self-explanatory, and demonstrates the power and, at the same time, the simplicity of the `mppsdk` package. A `PositionRequest` object is used to represent the position request data, to send the request to the MPC and, finally, to store the response. Then, a `PositionResult` object is used to extract the response information from the `Positionrequest` object.

Here's the code:

```
import mpssdk.*;

public class Position
{
 public static void main(String args[]){

   String ms    = "46777100009";
   String host   = "195.58.110.200";
   int  port   = 4000;
   String username = "epk";
   String password = "epkepk";

   try{
   System.out.println("Contacting positioning server");

   while(true){

   // Create an instance of a PositionRequest object
      PositionRequest posRequest =
            new PositionRequest(host, port, username, password);

      // The 'send' method connects to the MPC and gets a response
      System.out.println("Host contacted, waiting for reply...");
      posRequest.send(ms);

      PositionResult posResult;

      while ((posResult = posRequest.getPositionResult()) != null)
      {
       Coordinate coord = posResult.getCoord();
       System.out.println("Estimated Coordinates = (" +
                 coord.getLatitude()+","+
                 coord.getLongitude()+")");

       System.out.println("Radial distance to BTS = " +
                "["+ posResult.getInnerRadius() +
                ","+ posResult.getOuterRadius()+"]m.");
```

```
       System.out.println("Serving Cell Radius  = "+
                 posResult.getCellRadius()+"m.");

       System.out.println("Start and Stop Angles = ["+
         String.valueOf(posResult.getStartAngle())+"," +
         String.valueOf(posResult.getStopAngle() ) +"]\n");
      }

     }

   }catch (MPSException e){
   System.out.println(e);
   System.out.println(e.errMsg()+ " " + e.errCode());
   }catch (NumberFormatException e){
   System.out.println(e);
   }

 } //end of main


} //end of class
```

And here is an example of the output of this code when performing a position tracking of a mobile terminal that moves, as clearly stated in the sequence of radial distance ranges:

```
Host contacted, waiting for reply...
Estimated Coordinates = (N561238,E153755)
Radial distance to BTS = [4950,5500]m
Serving Cell Radius  = 15342m.
Start and Stop Angles = [345,105]

Host contacted, waiting for reply...
Estimated Coordinates = (N561238,E153755)
Radial distance to BTS = [8217,8767]m
Serving Cell Radius  = 15342m.
Start and Stop Angles = [225,345]

Host contacted, waiting for reply...
Estimated Coordinates = (N561238,E153755)
Radial distance to BTS = [11000,11550]m
Serving Cell Radius  = 15342m.
Start and Stop Angles = [345, 105]
```

Second Code Example: WAP-enabled Servlet for Remote/Self Positioning

In this second code example, I'll develop a Java Servlet that obtains the position of a mobile terminal using MPP and delivers it in WML format to a WAP-enabled phone through a WAP gateway.

Please note that the conceptual model of this kind of applications is quite different to the one described in the former example. In this case, the application will be running on a Servlet-capable HTTP server (the origin server) and the end clients are WAP mobile terminals.

As you can see in the code, this is a standard Servlet that generates responses in WML, with the usual concerns about response identification with the appropriate MIME and document type. The processing for obtaining location data is the same as in the former example:

```
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.PrintWriter;
import java.io.IOException;
import mpssdk.*;

public class LocationServlet extends HttpServlet
{
 String ms    = "46777100009";
 String host   = "195.58.110.200";
 int  port   = 4000;
```

```
 String username = "epk";
 String password = "epkepk";

 public void doGet(HttpServletRequest request,
          HttpServletResponse response)
         throws ServletException, IOException
 {
  //MIME type for WML documents
  response.setContentType("text/vnd.wap.wml");

  //get output stream handler
  PrintWriter out = response.getWriter();

  //generate contents
  out.println("<?xml version=\"1.0\"?>");
  out.println("<!DOCTYPE wml PUBLIC \"-//WAPFORUM//DTD WML 1.1//EN\"");
  out.println("\"http://www.wapforum.org/DTD/wml_1.1.xml\">");
  out.println("<wml>");
  out.println("<card id=\"card1\" title=\"Mobile Loc.\">");

  try{
   PositionRequest posRequest =
       new PositionRequest(host, port, username, password);
   posRequest.send(ms);
   PositionResult posResult;

   while ((posResult = posRequest.getPositionResult()) != null)
   {
     Coordinate coord = posResult.getCoord();
     out.println("<p><b>Your Position:</b><br/>");
     out.println("Lat ="+ coord.getLatitude()+"<br/>" );
     out.println("Long="+ coord.getLongitude()+"<br/>");
     out.println("InnerR="+ posResult.getInnerRadius()+"m<br/>");
     out.println("OuterR="+ posResult.getOuterRadius()+"m<br/>");
     out.println("Cell R="+ posResult.getCellRadius() +"m<br/>");
     out.println("Start Angle="
            + String.valueOf(posResult.getStartAngle())
            + "\u00BA<br/>");
     out.println("Stop Angle="
           + String.valueOf(posResult.getStopAngle())
           +"\u00BA</p>");
    }

  }catch (MPSException e){
   out.println("Positioning Error");
   out.println(e.errMsg()+ " " + e.errCode());
 }

  out.println("</card>");
  out.println("</wml>");
  out.close();
 }
}
```

One interesting point of this application refers to the mobile terminal being positioned. This example allows the remote tracking of a mobile terminal that is different from the one being used to display the positioning data, but it is also applicable for self-positioning, if the server knows the unique MSISDN of the requesting mobile phone.

The real question is how to obtain this MSISDN for use in a generic self-positioning Servlet. The telephone number of a WAP cell phone will most likely be unavailable as HTTP environment variables due to regulations regarding privacy, although it is technically possible for the operator to support this feature.

One solution to this problem would imply assistance of the WAP gateway being used. Basically, the gateway may pass along to the HTTP server a unique ID string for each subscriber, instead of his/her true MSISDN.

You should note that it's always possible for the user to explicitly pass to the WAP gateway and thus to the HTTP server the MSISDN as a parameter in the request URL.

Finally, as an example of which can be achieved with this code, this Servlet could contact a GIS server to generate contents describing a location in terms of near by elements of interest (pharmacies, banks, restaurants, etc).

An example of this application in action is shown here:

Getting Device Capabilities and User Preferences: UAProf

In order to achieve a truly complete WAP contents personalization, we still need to know the particular characteristics of the WAP mobile terminal and the active user preferences.

To help in this purpose, the World-Wide Web Consortium (W3C) is currently defining mechanisms for describing and transmitting information about the capabilities of WAP mobile terminals and the display preferences of WAP users.

The User Agent Profile (UAProf) specification extends WAP 1.1 to enable the end-to-end flow of a User Agent Profile (UAProf), also referred to as Capability and Preference Information (CPI), between the WAP client, the intermediate network points, and the origin server.

This specification uses the Composite Capabilities/Preferences Profile (CC/PP) model to define a robust, extensible framework for describing and transmitting CPI about the client, user, and network that will be processing the content contained in a WSP response. The CC/PP specification, in turn, defines a high-level structured framework for describing this information using the Resource Description Framework (RDF).

A new protocol, the CC/PP Exchange Protocol over HTTP, enables CC/PP profiles to be transported over the HTTP 1.1 protocol with the HTTP Extension Framework

The CPI may include:

- Hardware characteristics: screen size, image capabilities, manufacturer and model etc
- Software characteristics: operating system vendor and version, audio and video encoders etc
- Application/user preferences: browser manufacturer and version, markup languages and versions supported, scripting languages supported, etc
- WAP characteristics: examples include WML script libraries, WAP version, WML deck size
- Network characteristics: bearer characteristics such as latency and reliability

Summary

In this paper has been presented the practical use of Location Services (using the Mobile Positioning Protocol version 1.1), applied to the generation of position-aware contents to a WAP mobile terminal. This knowledge can be used along with the UAProf framework to achieve full personalization of the contents delivered to the end client. In this case, we are able to dynamically generate location-dependent, device-dependent and user preference-aware contents.

Resources

The European Telecommunications Standards Institute:
`http://www.etsi.org`

Ericsson:
`http://www.ericsson.com`

MPP and MPP SDK:
`http://www.ericsson.com/developerszone`

CC/PP:
`http://www.w3.org/Mobile/CCPP/`

UAProf:
`http://www.wapforum.org/what/technical.htm`