CONTEXT-AWARE WEB APPLICATIONS USING THE PINPOINT INFRASTRUCTURE

Jörg Roth
University of Hagen
Department for Computer Science
58084 Hagen, Germany
Joerg.Roth@Fernuni-hagen.de

ABSTRACT

Context-aware applications take into account the user's current context (e.g. the location). Many applications, e.g., tourist guides or office and meeting tools, use context information to adapt to a specific situation. Most of the actual systems use proprietary platforms and infrastructures to deal with context information, thus benefits are not directly available for a large community. The World Wide Web, on the other hand, is a well-established, widely available platform for a huge number of networked applications. In this paper, we present the PinPoint platform, which combines the benefits of the World Wide Web with context-aware computing. With PinPoint, a developer of context-aware web applications can concentrate on the application function and web contents and has not to deal with different client and server platforms or the capturing of contextual data. Context-aware web application can use existing web browsers and servers off the shelves, thus developers as well as end-users can use reliable and well-known software environments. PinPoint is especially designed for mobile applications that take into account the user's current location. Since applications can react on the device type (PC, PDA, mobile phone), web applications can produce output in an appropriate format. We demonstrate the strength of PinPoint with the help of a location-aware tourist guide, which runs on notebooks and PDAs.

KEYWORDS

Context-aware Computing, Web Applications, Mobile Computing, Tourist Guide

1. INTRODUCTION

Context-awareness is a key issue in the area of mobile and ubiquitous computing. Mobile users often require information depending on their current context, e.g. their location, their environment or available resources. E.g., a tourist using a tourist information system demands a map of the current location or descriptions of sights in the environment. There exists a huge variety of systems which take into account the current context. However, such systems are often built 'from scratch' and provide only a low degree of re-usable parts. Since most of these systems implement their own information infrastructure, it is difficult to transfer these platforms to a larger community using, e.g. different end-user devices.

The World Wide Web is often viewed as de facto standard for a huge class of client-server applications. There exists a number of server and browser environments for different operating systems. Especially the well-established formats and protocols make the web suitable for different usage scenarios. Convenient development tools and environments allow a developer to create complex applications such as electronic warehouses.

In the current version, the World Wide Web makes nearly no use of context information. Same requests passed to a server return the same information, regardless of the user's current location, browsing device or current time. As a result, a number of desirable services are not directly available and require modifications of browsers and servers. Examples for such services are location based services such as tourist guides or content services which adapts the page layout and format to the capabilities of the end-user's device and, e.g., downscale graphics for small displays.

In this paper, we present the PinPoint platform for context-aware web accesses. The PinPoint platform meets the following requirements:

- Arbitrary web browsers off the shelves cooperate with PinPoint. Neither browser modifications nor browser plug-ins are required. Only two processes have to be installed on the end-user device in order to collect contextual information. These processes run in the background and do not need the user's attention.
- Arbitrary web servers (e.g. apache) and additional server environments such as servlets or CGI scripts work together with PinPoint.
- Developers of context-aware web applications can use well-established tools, e.g., web editors, and design paradigms to create pages.
- Contextual information is embedded into web pages and URLs with the help of new tags.

As a main benefit, context-aware applications can be set up with low implementation and development costs, since we can use high-quality and inexpensive browser and server environments.

2. RELATED WORK

Tourist information systems are ideal examples for context-aware applications. GUIDE (Cheverst et al., 2000) and CYBERGUIDE (Abowd et al., 1997) are systems, which offer information to tourists, taking into account their current location. Both systems use modified or new browser applications to display maps and information about the tourists' environment.

Context-aware messaging systems trigger actions according to a specific context (Schilit et al., 1994). *ComMotion* (Marmasse & Schmandt, 2000) is a system which links personal information to locations and generates events (e.g. sound or message boxes), when a user moves to a certain location. *CybreMinder* (Dey & Abowd, 2000) allows the user to define more complex conditions under which a reminder will be generated (e.g. time is "9:00" and location is "office"). Conditions are stored in a database and linked to users. Whenever a condition is fulfilled, the system generates a message box.

A number of indoor-positioning systems offer context-aware services: the *Active Badge System* (Want et al., 1992) and *WIPS* (Wireless Indoor Positioning System) (WIPS, 2000) use infrared beacons. *SpotON* (Hightower et al., 2000) uses signal strength of radio signals. *Active Bat* (Ward et al., 1997) uses ultrasonic, the *Cricket* system (Priyantha et al., 2000) uses a combination of ultrasonic and radio. Many indoor-positioning systems are based on the *cell-of-origin* paradigm. They do not offer latitude/longitude co-ordinates, but provide a semantic position such as "Office F08, 3rd floor", which is often more meaningful for a user. This information is suitable for a number of context-aware applications, e.g., finding resources or people inside a building.

Cooltown (Kindberg et al., 2000) is a collection of applications, tools and development environments. It is mainly based on the World Wide Web infrastructure. The *Cooltown museum*, e.g., offers a web page about a certain exhibit when a visitor is in front of it. The corresponding URLs are transported via infrared beacons. The most critical drawback of existing systems is the lack of a common and widely available information infrastructure. Even though Cooltown uses the web, mechanisms to integrate context information into the data flow are proprietary.

3. THE PINPOINT APPROACH

PinPoint uses a proxy process to extend the traditional web architecture with a web browser on the client computer and a web server. Web proxies between client and server act as relay station and offer functions such as caching or filtering. They usually run on special computers, often inside an intranet. Our idea is to move the proxy process to the client computer, which then enriches data streams in both directions with context information. Neither browsers nor servers have to be changed since the proxy protocol remains unmodified. Fig. 1 presents the PinPoint architecture.

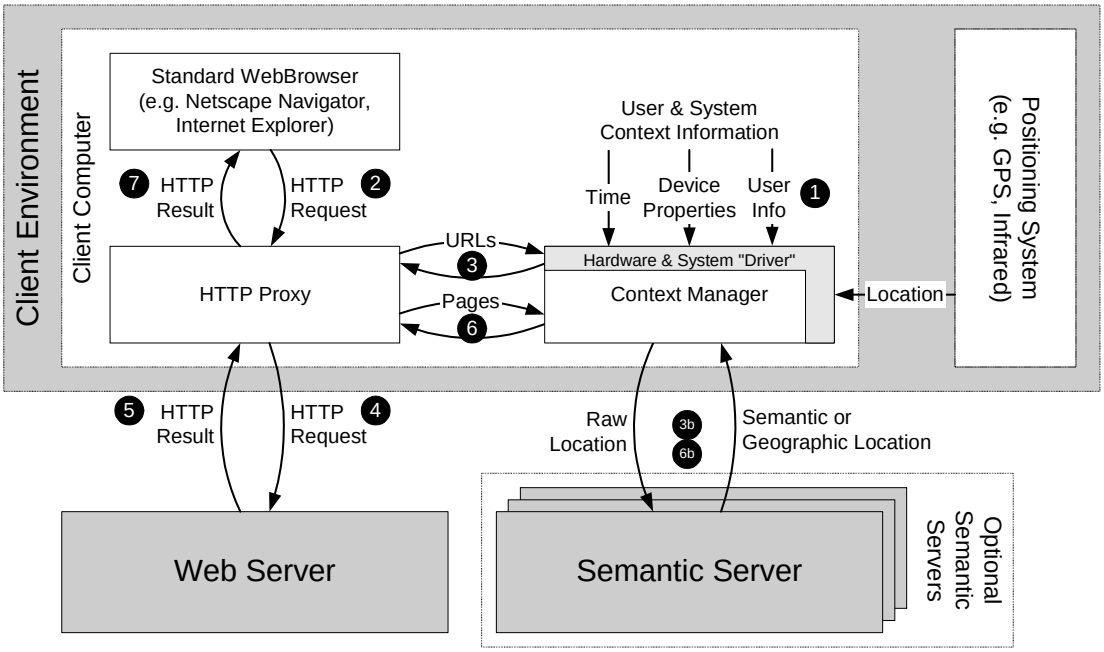

Fig. 1. The PinPoint architecture

The numbers indicate the steps performed by the system in chronological order. Besides the proxy process, a so-called *Context Manager* is hosted on the client computer. The Context Manager collects relevant context data such as user information or location. These data are collected either with the help of operating system functions or with additional devices such as GPS receivers or indoor positioning devices. The main function of the Context Manager is to decouple the web application from capturing context data. Device properties and user data are collected once (step 1), while location and time information is updated permanently (e.g. every 10 s).

Positioning devices may be very different. GPS receivers, e.g., may be connected to the mobile device in various manners (e.g. via a serial port or via a PCMCIA card). GPS receivers may use different protocols and data formats. A web application considering all these differences would be overloaded. For this, hardware- or device-dependent code resides in a driver layer. Web applications have not to struggle with issues related to the positioning device, thus the developer can concentrate on the actual application function.

For each request of the browser, the system performs steps 2 to 7. Steps 2, 4, 5 and 7 belong to the standard web data flow. Our system performs the additional steps 3 and 6, which integrate contextual data into the data packets with the help of specific tags (see section 3.2).

All client processes (i.e. proxy and Context Manager) run in the background. Both processes can be started automatically when the end-user device boots up.

Semantic Server

Positioning systems can be divided into two categories: one category provides geographical co-ordinates (e.g. GPS) and one, which provides semantic location information (e.g. indoor positioning systems based on infrared). Usually, a specific web application is only interested in one category. A navigation application, e.g., usually requests geographical co-ordinates. On the other hand, geographical co-ordinates are often meaningless for the user. Instead of providing the GPS co-ordinates `N51°22.579/ E007°29.615/169m`, it is more meaningful to use the term "University of Hagen, building IZ, back door". To convert geographical co-ordinates into semantic locations and vice versa we set up so-called *Semantic Servers*. Semantic Servers take location data provided by the positioning system and give back location information of the corresponding other category (steps 3b and 6b). The server maps e.g. GPS positions to semantic positions with the help of a local map. Conversely, semantic locations are mapped to GPS locations with certain accuracy. This allows a developer to create web applications without considering the character of a specific positioning system. In addition, the web application does not even notice, when a mobile user switches between two positioning systems when moving (e.g. between GPS outdoor and infrared indoor). Once the

Semantic Server returned a value for a specific location, the Context Manager keeps it inside a cache. The Context Manager can resolve subsequent queries for the same location internally.

Semantic Servers may exist in every local network. If a network covers a small area (e.g. a campus), a single server is sufficient. Large mobile networks (e.g., cell phone networks) may provide a number of Semantic Servers. A client can look up the appropriate Semantic Server via a service discovery protocol, via broadcast messages, or with the help of the DHCP protocol. We defined a specific DHCP record stored in the local DHCP server. This record points to the local Semantic Server. Note that Semantic Servers are optional. In principle, a web application could carry out semantic transformation without the help of the web server. However, Semantic Servers have essential advantages: a distributed network of Semantic Servers scales much better than a single central server. Since mapping information usually is available locally, semantic servers are easy to maintain. In addition, the Semantic Server follows a more modular and object-oriented approach since a web application can concentrate on the actual application function and can shift functions related to location transformations to Semantic Servers. If a web application had to realise semantic mappings individually, identical code parts would exist duplicated inside the different applications.

3.1 Context Information

Dey & Abowd (1999) mean by context "any information that can be used to characterize the situation of an entity", where entity can be a person, a place or a relevant object inside a situation. This meaning of context covers a wide area including social aspects. In our specific area however, we only use context information, which can be analysed by an application. PinPoint distinguishes three types of context data:

1. Time and space: This information is often viewed as the most important context information. Applications such as tourist guides or navigation systems heavily use this information. We acquire the geographic location in GPS co-ordinates, semantic location (if available), the precision as well as the age of the last measured location, time of the day and the current date.

2. Device properties: End-user devices have specific properties which rarely change over time. PinPoint captures the following data: device type (e.g. mobile phone, desktop computer), whether the device is mobile, the screen resolution, and the capability to display colours. These data allow a web application to provide user interfaces adequate for a specific end-user device. For, e.g., devices with low display capabilities, images with low resolution or few colours can be used. This reduces transfer times and allows a device to display images directly without re-computing size or colour tables. For devices with reduced graphical capabilities (e.g. i-Mode phones, terminals for blind persons), the web application can provide an alternative page layout.

3. User identity: The PinPoint system identifies the current user with the help of the login name passed to the operating system. If the operating system accepts unregistered users (e.g. Windows 98), a user name is taken from a configuration file. Since the login name is not unique outside a local network, we use additional information (e.g. the email address). It depends on the web application to request further authentication such as a password from the user.

3.2 New Tags

The Context Manager adds context data to the data streams between browser and server with the help of embedded tags. We defined a set of new tags, which a developer integrates into HTML pages or URLs. Fig. 2 illustrates the concept.

The proxy replaces tags such as `__Gusername__` and `__Gpos__` by the corresponding context information (i.e. the current user name and the current position). Since the new tags can occur at arbitrary places inside a page (e.g. even *inside* an HTML tag or inside a path), we decided not to use the HTML or XML style tags (i.e. `<TAG...>`).

For each new tag, we provide a *get* and a *put* variation (denoted as `__Gtag__` or `__Ptag__`). Get tags are replaced when a page is transferred from server to client (fig. 1, step 6), whereas put tags are replaced inside a URL request of a client (fig. 1, step 3). In most cases, get tags are sufficient. In rare cases however, a web application needs context data at exactly the time, a user requests a new page. In these cases, a get tag would have been replaced to early (when the page was loaded). The Context Manager transforms put tags whenever a user follows a link and request a new page.

a) An HTML source

```
<HTML>
<HEAD><TITLE>PinPoint test page</TITLE></HEAD>
<BODY bgcolor=white>
 <DIV align=center><H1>PinPoint</H1></DIV>
 <P>
 Welcome <A href="http://pi2.fernuni-
 hagen.de/~__Pusername__">__Gusername__</A>
 <TABLE border=1>
  <TR><TD>GPS Position:</TD>
      <TD>__Gpos__</TD></TR>
  <TR><TD>Semantic Position:</TD>
      <TD>__Gsempos__</TD></TR>
  <TR><TD>Date:</TD>
      <TD>__Gdate__</TD></TR>
  <TR><TD>Local Time:</TD>
      <TD>__Gloctime__</TD></TR>
  <TR><TD>Screen Resolution:</TD>
      <TD>__Gscreenx__ * __Gscreeny__
          Pixel</TD></TR>
  <TR><TD>Resolution Quality:</TD>
      <TD>__Gsemscreen__</TD></TR>
  <TR><TD>Colors:</TD>
      <TD>__Gcolbits__ Bit(__Gcols__)</TD></TR>
  <TR><TD>OS:</TD>
      <TD>__Gos__ Version __Gosver__</TD></TR>
  <TR><TD>Device Type:</TD>
      <TD>__Gdevice__</TD></TR>
 </TABLE>
</BODY>
</HTML>
```

b) The output

Fig. 2. A simple PinPoint page

3.3 Security Issues

Context information often has a confidential characteristic. Usually a mobile user is not willing to, e.g., publish her or his current location to any server. The World Wide Web on the other hand is an open infrastructure, e.g., any web server can participate without permission of a trusted organisation. This leads to the problem of security and privacy. We have to answer two questions:

- How can a user be sure that only trusted web servers read context information?
- How can a user be sure that third parties cannot spy out context information when it is transferred to the server?

To ensure privacy during the transfer process, a user can rely on security and encryption mechanisms used with HTTP. We can divide the entire communication into the two segments *browser – proxy* and *proxy – server*, where only the latter segment requires a security protocol. Note that the browser – proxy communication runs locally, i.e. is usually not vulnerable against attacks. For the proxy – server communication, protocols such as *Secure Socket Layer* or *Transport Layer Security* can be used. In order to detect context tags inside the data streams, the proxy must be able to read all communication between browser and server in plain text. Protocols, which forward *encrypted* data via the proxy (e.g. *tunnelling SSL*), cannot be used.

The first question is more crucial. On one hand, the user explicitly wants to transfer context information to some servers in order to get a specific service. On the other hand, the system should protect users against servers which want to spy out context information. To address this problem, a PinPoint user configures two lists: a *black list* contains all web servers which are untrusted, and a *white list* contains all web servers which are viewed as safe. In addition, a user configures three profiles: one for black list servers, one for white list servers and one which applies for unlisted servers. Each profile defines for all context data, whether a specific entry will be transferred or not. If the user does not define an explicit rule, the Context Manager asks the user before contacting a server.

The default profiles provided by PinPoint are:

- *black list*: transfer no context information,
- *white list*: transfer all context information,
- *unlisted servers*: always ask the user.

With profiles, a user can define the security behaviour in a very fine-grained manner.

Until now, security issues are discussed from the end-user's viewpoint. There is however another important question: How can a server trust context data transferred from a client? This question may be relevant for a number of applications, e.g., mobile payment or fleet management systems. Currently, this is an open issue, since all context data are generated by a possibly untrusted client. Authenticity of context data can only be proofed, if additional services, e.g., tracking systems or certification authorities outside the platforms, are used.

4. A TOURIST GUIDE WITH PINPOINT

To verify our approach, we developed a web-based tourist guide on top of PinPoint (fig. 3a). The tourist gets a map, which presents her or his actual environment. The current location is marked with an arrow. The user can switch between different maps, i.e. can zoom in and out. For a specific location, the application lists interesting sights in the area.

a) Notebook guide b) Palm guide

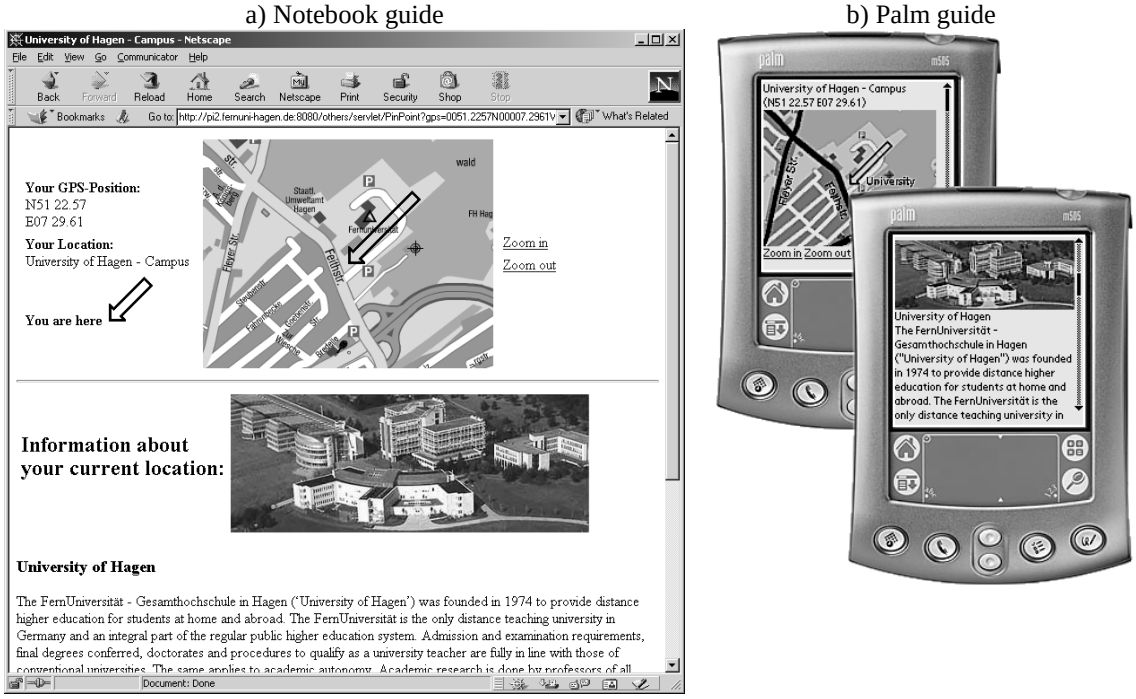

Fig. 3. The PinPoint tourist guide

Since the web application can query the display capabilities of the end-user's device, it can generate appropriate maps. The page layout for notebooks (fig. 3a) provides very detailed information, the pages for palm devices (fig. 3b) are reduced to the necessary elements. The map for palm devices, e.g., is not just downscaled from the original map, but a different map, which only provide most important entries.

Fig. 4 shows the underlying communication infrastructure. GPS positions are read from a mobile GPS receiver. If no GPS signal is available, the co-ordinates may be derived from an alternative positioning system, if necessary, with the help of a Semantic Server. The mobile device communicates to a server via a wireless connection. Small areas, e.g. a campus, may be equipped with WLAN base stations. For a larger area, we have to use mobile phone links to communicate with the server.

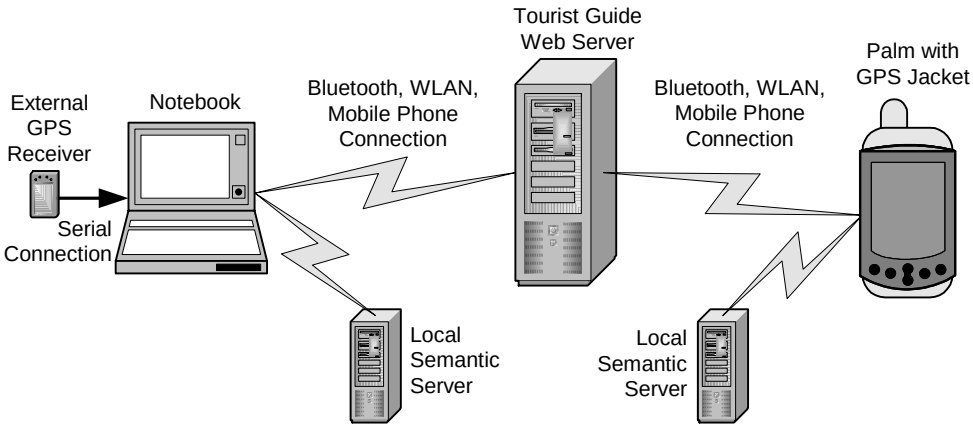

Fig. 4. The tourist guide system

The tourist application passes context data to the server with the help of tags embedded into a requesting URL. Each time the browser requests or refreshes a map, the following URL is sent to the server:

```
http://carmen.fernuni-hagen.de:8080/servlet/PinPoint ?
        serve=page & gps=__Ppos__ & age=__Pposage__ &
        quality=__Pscreenx__x__Pscreeny__
```

`Carmen` is our tourist guide server, running a servlet which maintains the maps and computes the user's position inside the maps. The PinPoint system replaces the tags by the actual context information. The servlet reads the actual values from the current parameter set given to the servlet. In order to be up to date when moving, a small JavaScript procedure embedded into the main page requests a new map every 10 seconds.

For each map image, a co-ordination function is stored, which transforms GPS co-ordinates into pixel co-ordinates. In addition, the co-ordinates of interesting sights and the URLs of the corresponding web pages are stored inside the server. With this information, the system can provide a list of interesting sights in the neighbourhood.

5. IMPLEMENTATION DETAILS

We realised the entire PinPoint system (i.e. proxy, Context Manager and Semantic Server) in Java. The proxy process is a modified *Muffin* web proxy. The source code of Muffin is publicly available (`http://muffin.doit.org/`), thus, it was easy to extend the proxy to communicate with the Context Manager (steps 3 and 6 in fig. 1).

The Context Manager is designed as a background process, however, it provides a rudimentary user interface. For testing and debugging purposes, the user can put the Context Manager into a *simulation mode*, in which the user can enter context data directly. With the help of the simulation mode, the developer can, e.g., easily test the tourist guide without moving around. Our Context Manager prototype contains drivers to read system and user data. In addition, we developed a driver for a GPS receiver based on the established *NMEA* (*National Marine Electronics Association*) protocol.

The Context Manager communicates with a Semantic Server process via two well-known ports. A UDP port is used to search for Semantic Servers in a network. For this, a context manager can broadcast discovery packets to the network and can wait for replies. We use this discovery mechanism, if the DHCP search fails. The second port, a TCP port, is used for the actual communication (i.e. to perform location requests).

Our prototype implementation currently runs only on operating systems which support full Java (Standard Edition). Small devices such as PDAs or mobile phones often only support Java Micro Edition. In our testing environment, we run the PinPoint processes on another computer in such cases. This however conflicts with our original approach and raises a number of new issues (concerning, e.g., security). Our next goal is to transfer the PinPoint system to very small systems.

6. CONCLUSION

PinPoint is a platform to develop context-aware web applications. The platform consists of a client extension and optional Semantic Servers. Web servers and browsers remain unmodified, thus web applications can use widely available and reliable software components. We especially avoid system-specific browser plug-ins in our system. Context-specific processes and devices are strongly separated from the web application, thus the developer can concentrate on the application function and has not to struggle with positioning systems and device protocols.

PinPoint provides three types of context information: *space and time*, *device properties* and *user's identity*. Web applications can use location data without considering the used positioning system. Semantic Servers can mediate between geographic and semantic locations and vice versa. Thus, even switching between two kinds of positioning systems when moving is transparent to the web application.

Currently, there exist a huge number of mobile web-enabled end-user devices with different display capabilities, e.g. notebooks, sub-notebooks, palmtops or cell phones (with, e.g. i-Mode). An ideal web application does not rely on the browser to generate an appropriate user interface. With the help of a number of PinPoint tags, the web application can ask the device for its screen resolution, capability to display colour etc., and generate an optimised page layout.

A security mechanism integrated into the platform prevents a user from giving away context data to untrusted servers. We demonstrated the strength of PinPoint with the help of a tourist guide application. It uses location information as well as device properties to produce an appropriate output.

REFERENCES

Abowd, G. D.; Atkeson, C. G.; Hong, J.; Long, S.; Kooper, R.; Pinkerton, M, 1997: *Cyberguide: A mobile context-aware tour guide*. ACM Wireless Networks, 3: 421-433

Cheverst, K.; Davies, N.; Mitchell, K.; Friday, A.; Efstratiou, C., 2000: *Developing a Context-aware Electronic Tourist Guide: Some Issues and Experiences*, in Proceedings of CHI'00 (Netherlands, 2000), ACM Press

Dey, A., K.; Abowd, G., D., 1999: *Towards a better understanding of context and context-awareness*, GVU Technical Report GIT-GVU-99-22, Georgia Institute of Technology,

Dey, A., K.; Abowd, G., D., 2000: *CybreMinder: A Context-aware System for Supporting Reminders*, Second International Symposion on Handheld and Ubiquitous Computing 2000 (HUC2K), Bristol (UK), Sept. 25-27, 2000, LNCS 1927, Springer, 187-199

Hightower, J.; Boriello, G.; Want, R., 2000: *SpotON: An Indoor 3D Location Sensing Technology based on RF Signal Strength*, Technical Report #2000-02-02, University of Washington, Feb. 2000

Kindberg, T.; Barton, J.; Morgan, J.; Becker G.; Caswell, D.; Debaty, P.; Gopal, G.; Frid, M.; Krishnan, V.; Morris, H.; Schettino, J.; Serra, B.; Spasojevic, M., 2000: *People, Places, Things: Web Presence for the Real World*, Proc. 3 rd Annual Wireless and Mobile Computer Systems and Applications, Monterey CA, USA, Dec. 2000. p. 19

Marmasse, N.; Schmandt, C., 2000: *Location-aware Information Delivery with ComMotion*, Second International Symposion on Handheld and Ubiquitous Computing 2000 (HUC2K), Bristol (UK), Sept. 25-27, 2000, LNCS 1927, Springer, 157-171

Priyantha, N., B.; Chakraborty, A.; Balakrishnan, H., 2000: *The Cricket Location-Support System*, Proc. of the 6. Annual Internat. Conf. on Mobile Computing and Networking, Aug. 6.-11., 2000 Boston

Schilit, B.; Adams, N.; Want, R., 1994: *Context-Aware Computing Applications*, Workshop on Mobile Computing Systems and Applications, Santa Cruz, CA, USA, 1994

Want, R.; Hopper, A.; Falcao, V.; Gibbson, J., 1992: *The Active Badge Location System*, ACM Transactions on Information Systems, Vol. 10, No. 1, Jan. 1992, 91-102

Ward, A.; Jones, A.; Hopper, A., 1997: *A New Location Technique for the Active Office*, IEEE Personal Communications, Vol. 4, No. 5, Oct. 1997, 42-47

WIPS, 2000: *WIPS Technical Documentation*, Royal Institute of Technology, Schweden, http://2g1319.ssvl.kth.se/2000/group12/technical.html