An architecture for context-aware mobile web browsing

J. Noltes (Johan)
j.noltes@student.utwente.nl

ABSTRACT
Context information is not widely used while browsing the internet on mobile devices. With an adapted website, the content can become more suitable to the user. Currently, most examples of context-aware browsing are based on context information gathered by telecom providers, but mobile devices can also gather context information. In this paper, we investigate which context information can be gathered by mobile phones. We describe an architecture for sending context information from a mobile web browser to a web server via an HTTP post request with PIDF extended with Geopriv, thereby enabling easier and richer context-aware mobile web browsing.

Keywords
Context-aware, mobile phone, web browsing, location based services, exchange context information, privacy control

1. INTRODUCTION
Currently, most mobile phones have the possibility to connect to the internet. With upcoming flat-fee data plans, mobile phones are increasingly used to check e-mail, synchronize agenda’s, and browse the web [1].

Mobile phones can, depending on the specific phone, gather context information [2], like camera images, calendar appointments, gsm cell-id, gps-coordinates, nearby Bluetooth and Wi-Fi devices and activity. We adopt Dey’s definition of ‘context information’ [3]: “Context is any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves.”

The content of a website can become more suitable to the user when using context information while browsing the web on a mobile phone. For example when checking the departure times of trains, the web server doesn’t need user-input to know which information likely should be showed, but can directly show the information gathered by the mobile phone, this may give a web server more, used while browsing the web, but (also) the context information gathered by telecom providers has a number of disadvantages:

First, web servers depend on telecom providers to get context information, which they can use to adapt the website to the situation of the user. The context information differs per telecom provider (different coverage), and separate business contracts and exchange formats might be needed. Developers are depending on telecom providers to be able to use this context information. This creates a barrier for developers, because they need to take extra effort to develop new services.

Second, the context information gathered by telecom operators is currently limited to positioning and the information is not provided with a high accuracy. For example, the positioning of a mobile phone based on TDOA (triangulation) has an accuracy of about 100 meters [4]. Some websites need more precise context information to work properly.

Third, users don’t know exactly which context information is gathered by telecom operators, and can’t control the disclosure of this context information. This might cause decreased acceptance of websites using context information [6].

On the other hand, mobile phones could also gather all kinds of information that could be used by web browsers as context information [2]. Because only context information gathered by telecom operators is used, not all available context information can be used by web servers. Research has been done on using telecom operator context information, but for exchanging context information between a mobile web browser and web servers no architectures exist at the moment. Context information gathered by mobile phones is hardly used, so there are a lot of opportunities to use this information.

2. PROBLEM STATEMENT
Currently, context-aware mobile websites only adapt based on location data provided by telecom operators. For example, Hyves1 uses the gsm cell-id to locate a blog post sent via SMS.

Using context information for mobile websites based solely on context information gathered by telecom providers has a number of disadvantages:

This results in the following research question:

“How to communicate context information gathered by a mobile phone from mobile web browsers to web servers?”

To be able to answer this research question, we have to answer the following sub-questions:

- Which context information can be gathered by mobile phones, and what is the quality of this context information?
- What kind of frameworks and protocols for context-aware mobile web browsing already exist, and what are their features?
- What are criteria for a protocol to exchange context information between a mobile web browser and a web server?
3. MOBILE PHONE CONTEXT-FEATURES

3.1 Operating systems

Modern mobile phones with internet access run a limited number of operating systems. Windows Mobile runs on most HTC devices. Symbian OS had an 65% market share on smart phones in 2007 [5]. Nokia and Samsung phones run Symbian OS with the S60 platform and Sony-Ericsson phones use the UIQ platform. Apple runs it own iPhone OS on iPhones. RIM runs BlackBerry OS on its phones. Google is developing Android.

Because the iPhone OS API is not public available yet, and no details about Android are available yet, these operating systems won’t be included in our research.

All of these operating systems support Java ME, which supports locations based services via the location based API (JSR 179). However, this API is optionally to implement in Java distributions and implementations also differ per operating system, so we won’t discuss Java ME in this paper.

3.2 Context information

The API’s and SDK’s of these operating systems show us that raw context information can be obtained. Table 1 shows which context information can be obtained by which operating system, based on the available information. Of course the context information available depends on the specific phones, the table shows which functionality the API’s of the operating systems offer.

Table 1. Context information available on operating systems

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Mobile phone number</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Device ID</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>GSM Cell-ID</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Location Area Code / Country Code</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Network provider id (current / home)</td>
<td>+</td>
<td>-</td>
<td>+</td>
</tr>
<tr>
<td>GPS coordinates</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
</tbody>
</table>

All this context information can be grouped into four categories: identity, activity, location and time [9]. Some of this context information can be processed on the mobile phone to higher level information. For example (locations of) calendar appointments can be interpreted to standardized locations like ‘home’ or ‘work’. The categorization is shown in Table 2.

Table 2. Categorization of context information

<table>
<thead>
<tr>
<th>Category</th>
<th>Identity</th>
<th>Location</th>
<th>Activity</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Phone number</td>
<td>+</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Device ID</td>
<td>+</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cell-ID</td>
<td>+</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GPS</td>
<td>+</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bluetooth / Wi-Fi</td>
<td>+</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Appointments</td>
<td>+</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Messenger status</td>
<td>+</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Time</td>
<td></td>
<td></td>
<td></td>
<td>+</td>
</tr>
</tbody>
</table>

Table 2 shows that mobile phones can gather context information in all context-categories, so information about all categories can be sent from the mobile web browser to the web server. We will discuss the categories in some more detail.

3.2.1 Identity

The phone number and device id indicate the identity of the user. To protect the user privacy, the identity will be discarded in our architecture.

3.2.2 Location

There are several ways to display the location of a mobile phone. We will discuss them briefly, including their ‘spatial resolution’ to describe the Quality of Context. Spatial resolution is defined as ‘the precision with which the physical area, to which an instance of context information is applicable, is expressed’ [10]. A high spatial resolution means we know a location of an object in more detail than with a low spatial resolution.

The GSM Cell-ID shows to which GSM tower the mobile phone is connected. The Cell-ID can be mapped to i.e. a GPS location[11]. With the average sending strength of the GSM signal, this defines an area where the mobile phone is located. This information has a low spatial resolution.

The Location Area Code (LAC) indicates the area code of the phone numbers of the location of the mobile phone. The LAC can be mapped to a city. For example, the LAC of Enschede is ‘053’. The LAC has a low spatial resolution.

The Country Code indicates an id of the country where a mobile phone is located. This has a very low spatial resolution.

The GPS-coordinates indicate the location of a mobile phone with a high spatial resolution. Disadvantage of using GPS is
that it is not always available (for example indoors), and consumes a lot of battery power.

Bluetooth and Wi-Fi can detect nearby devices. When some devices are on static locations, a fingerprint of them can be made, and linked to a specific location (beacon)[12]. These fingerprints can be stored in a database on the mobile phone and can be used to determine the location of the mobile phone. The spatial resolution depends on the quality of the fingerprints, but can be high.

Appointments retrieved from the agenda mostly contain a location. These locations are difficult to interpret, and can have a varying spatial resolution; from country to room number.

3.2.3 Activity
The activity of a user can be derived from his calendar appointments and messenger status. The appointments are highly privacy sensitive so they have to be interpreted on the mobile phone to activities like ‘in a meeting’. The messenger status (presence) can indicate statuses like ‘out to lunch’ and ‘on the phone’.

3.2.4 Time
The time indicates when the context information was gathered. This indicates how accurate the context information is.

4. EXISTING FRAMEWORKS
For our architecture, we try to use existing frameworks and protocols. In this section, we describe which frameworks are built already for context-aware applications, and which existing protocols can send context-information. In section 6, we choose which framework and/or protocol we will use in our architecture.

For building context-aware applications, a number of frameworks exists. The framework developed by Dey [13] is made for personal computers, and does not fit on mobile phones. Johnson [12] presents a framework which uses fingerprints of Wi-Fi, Bluetooth. These frameworks provide functionality for rapid development of some context-aware applications, but are not appropriate for mobile web browsing.

The activity of a user can be derived from his calendar appointments and messenger status. The appointments are highly privacy sensitive so they have to be interpreted on the mobile phone to activities like ‘in a meeting’. The messenger status (presence) can indicate statuses like ‘out to lunch’ and ‘on the phone’.

HTTP Enabled Location Delivery (HELD) is a protocol using Geopriv that is used for retrieving location information from a server within an access network. The protocol includes options for retrieving location information either by-value or by-reference. The protocol is and used for example as Location server for IP telephony [16].

HTTP Enabled Location Delivery (HELD) is a protocol using Geopriv that is used for retrieving location information from a server within an access network. The protocol includes options for retrieving location information either by-value or by-reference. The protocol is and used for example as Location server for IP telephony [16].

5. CRITERIA
The architecture we develop will be used for a particular situation: it has to carry context information on a mobile connection. Therefore, it needs some special criteria.

5.1 Architecture
As mentioned in section three, a number of mobile operating systems support web browsing. Because the protocol is within the architecture, no specific operating system functions are needed for the protocol. The architecture should be able to communicate with these operating systems.

For exchanging context information (location, activity and time), a number of protocols has been developed, which we will discuss in more detail. We will use some of these protocols to send the context-information from the mobile web browser to the web server.

- Secure User Plane Location (OMA)
- Mobile Location Protocol (OMA)
- Presence Information Data Format (IETF)
- HTTP Enabled Location Delivery (HELD / Geopriv)

The Open Mobile Alliance (OMA) Secure User Plane Location protocol (SUPL) specifies how mobile phones and cellular networks can exchange location-related information to enable quicker positioning of a handset. It uses A-GPS and Network Based Positioning. [14]

The OMA Mobile Location Protocol (MLP) is an application-level protocol for obtaining the position of mobile stations (mobile phones, wireless personal digital assistants and so on) independent of underlying network technology. MLP serves as the interface between a Location Server and a Location Services Client. A Location Service Client continuously updates its location to the Location Server, so other party’s may retrieve the clients location from the Location Server.

Presence Information Data Format (PIDF) is a common presence data format for presence protocols, allowing presence information to be transferred across protocol boundaries without modification, with attendant benefits for security and performance. [15]

Geopriv is an IETF working group chartered to develop a way to express geographic location and permissions policy that allows users to state where they are and who can see that. A number of protocols have been specified. All these protocols use Geography Markup Language, a XML grammar defined by the Open Geospatial Consortium to express geographical locations.

HTTP Enabled Location Delivery (HELD) is a protocol using Geopriv that is used for retrieving location information from a server within an access network. The protocol includes options for retrieving location information either by-value or by-reference. The protocol is and used for example as Location server for IP telephony [16].

5.2 Protocol
In section three, we have seen that mobile phones can gather location, activity and time as context information. For location, the protocol should be able to exchange GSM Cell-ID, Location Area Code, GPS coordinates, and higher level location-information like ‘home’, ‘work’ and ‘car’[17]. For activity, presence information (messenger status) should also be includeable.

The context information is transported over a mobile connection like GPRS. These connections have a long round trip time[18] and a low bandwidth, with a maximum of 80 kbit/s for download and 20 kbit/s for upload. This means the number of (extra) packets and the packet size must be as low as possible, especially because the upload speed is even lower than the download speed. With upcoming use of HSDPA, higher speeds appear, but round trip times on mobile networks stay long.

Mobile browsing is a service, which is not used continuously. It shouldn’t be necessarily to frequently update the context information of a user because this costs extra data traffic and
battery power on an open connection. This means context information must only be sent while browsing.

5.3 Privacy
Sending context information gathered by a mobile phone to a web server has as a consequence that privacy-sensitive information is disclosed. Since privacy may be a factor in the acceptance of the architecture, this information must be handled with care. This will be a major task of an application, the protocol will only send the information.

The protocol will send context information about the users location and activity. The application must provide options to let users determine whether they want to disclose information or not. Context information should only be sent to websites, which are context aware, and allowed by the user. The protocol must be sent secured.

One of the pitfalls [19] while developing context-aware applications is to let users configure too much, as they will simply ignore the configuration. This means the application should work without too much configuration. Users should also know that information is disclosed, and have control about what information is used exactly.

6. ARCHITECTURE
To send context information from a mobile web browser to a web server, we propose an architecture as shown in Figure 1. A plug-in will be installed to the mobile web browser (client) to gather context information and encapsulate it in a data format for a protocol to be sent to the web server. The web server also has a plug-in installed, which will receive the encapsulated information and translate it for the web server, whereby context-databases may be needed to interpret the information to a higher level of context information. The plug-in for the client and the server, and the protocol will be discusses in more detail.

6.1 Plug-in
6.1.1 Client
When visiting a context-aware website for the first time, the plug-in of the web browser (client) detects the context-awareness capability of the web server, because of the provided special header. The user will be notified that the visited website is context-aware, for example by showing a special button next to the address field. A user clicking this button will allow the plug-in to send the context information to the web server. Optionally, the user can indicate which context information may be sent (for example only city for location or messenger status for activity), and store these settings for later use. The next time the user visits a website, the plug-in knows that the website is context-aware before requesting the website, so these settings can be applied directly in the request and context-information can be send directly.

The plug-in gathers the context information using the information provided by the API’s of the operating systems. This information will be packed into the protocol data format, and sent to the web server via an HTTP post request, discussed in section 6.3.

The context information may be interpreted and manipulated before sent to the server. For example the GPS coordinates, Cell-ID or calendar appointments may be linked to favorite user locations, as ‘home’, ‘work’ and ‘cafe’. Activities derived from calendar appointments also need to be interpreted on the mobile phone, for example using categories assigned to the appointment. Only limited interpretation can be done at the mobile phone, because of the limited cpu power and capacity.
for databases. For example a database with Cell-ID’s linked to GPS-coordinates may be too large for a mobile phone, so this interpretation has to be done by the web server.

The context information is sent depending on the ‘send-context information-every’-setting given by the server. When this is ‘page view’, the context information must be send on every request to receive an adapted web site. When the setting is time-out, the client and server need to keep track of a session for that website, to be able to cache the context information. The client will generate a session-id for this reason. When the setting is ‘significant change’, the plug-in needs to remember the last sent context information, and on every request compare the actual information with the last sent information. When for example the Cell-ID or Messenger status has changed, the new context information must be send.

6.1.2 Server

The plug-in of the web server receives a HTTP post request with the context information described in PIDF, as discussed in section 6.3. It can interpret this information before sending it to the application. For example the Cell-ID can be translated to GPS coordinates, or GPS coordinates can be translated to a specific location. For this actions, some databases for translating context information are needed. The application running on the web server will receive the (interpreted) context information in PIDF, which can be used to adapt the website. The web server will send this adapted website to the mobile web browser.

As mentioned in 6.1, the server sets the ‘Send-context information-every:’ to indicate how often the client should send the context information. Values can be ‘page view’, ‘significant change’ and a time-out. Except with ‘page view’, the client doesn’t send the context information with every request for efficiency reasons. This means the server has to store the context information, and keep track of a session, with a session-id generated by the client. Because this plug-in is designed for context information of mobile phones, we assume a time-out of about one hour after the last page view must be enough to discard the information. After this hour, it is likely that the user is not visiting this website anymore and the information is outdated.

Figure 3 shows the three options for the options ‘page view’, ‘time-out’ and ‘significant change’. A dotted line indicates that context-information is sent.

Figure 3. Sending context-information with ‘page view’, ‘time-out’ and ‘significant change’

6.2 Protocol interactions

The protocol describes how to send the context information gathered by the mobile phone from the web browser (plug-in) to the web server (plug-in). There are different schema’s for a first time visit and a returning visit.

6.2.1 First time use

When a user visits a context-aware website for the first time, a normal request for a website will be done. When the result is received, the plug-in will notify the special ‘context-aware website’ header. The user will be notified that this website can be adapted by sending context information. When the user indicates to request the context-aware version of the website, the context information will be sent via a HTTP post request. The plug-in at the server will process this information, and send an adapted website back. The plug-in will store that the visited website is context-aware.

6.2.2 Returning user

When the user later requests the website again, the plug-in has stored that this website is context-aware, so it will directly send a HTTP-post request with the context information. The web server will return the adapted website directly. This means no extra request is needed.

6.3 Protocol data format

The criteria state that at least possible connections should be needed. This means that protocols using a location server are not suitable for this purpose, because the mobile phone should continuously updates its location to the location server, which requires continuously connections and data traffic. Furthermore, the protocol should be able to send information about location, activity and time.

Table 3. How do the protocols answer the criteria

<table>
<thead>
<tr>
<th>Criteria</th>
<th>SUPL</th>
<th>MLP</th>
<th>Geopriv</th>
<th>HELD</th>
<th>PIDF</th>
</tr>
</thead>
<tbody>
<tr>
<td>Location</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>-</td>
</tr>
<tr>
<td>Activity</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>+</td>
</tr>
<tr>
<td>Time</td>
<td>-</td>
<td>-</td>
<td>+</td>
<td>-</td>
<td>+</td>
</tr>
<tr>
<td>No location</td>
<td>-</td>
<td>-</td>
<td>+</td>
<td>-</td>
<td>+</td>
</tr>
</tbody>
</table>
Most protocols mentioned in section 4 need a location server and continuously updates (MLP, HELD), or are designed for communication with a network operator (SUPL, MLP, HELD), so they do not meet the criteria. Geopriv and PIDF do meet the criteria, and will be used for our protocol.

6.3.1 Geopriv
The Geopriv Working Group designs protocols and formats to send all types of location-information. Geopriv protocols and formats use Geography Markup Language (GML) to represent location information. GML is very extensive, and can carry nearly all possible location information. For our purpose, we only need a few definitions like coordinates, and choose the simplest version of GML.

A typical GML-file looks like this:

```xml
<gml:Point gml:id="point1" rsName="epsg:4326">
</gml:Point>
```

Figure 6. Example of body GML-file

6.3.2 Presence Information Data Format (PIDF)
PIDF (RFC 3863 [15]) is a XML-based format, which can carry presence information, and can be easily extended with own types of information. PIDF contains three important elements: ‘contact’, ‘status’ and ‘timestamp’. For our purpose, the status-element is the most important one. It can contain presence and activity information. PIDF doesn’t prescribe the implementation of this element, only that it should be defined elsewhere. A PIDF-file is registered with the MIME content type ‘application/pidf+xml’.

A typical PIDF-file is illustrated in Figure 7. In this example, the format is extended with ‘myex’ as an enumeration of ‘home’, ‘office’ and ‘car’. These are only simple examples, also complex (XML-)elements can be added.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<presence xmlns="urn:ietf:params:xml:ns:pidf"
  xmlns:myex="http://id.example.com/presence/
  xmlns:im="http://prowl.org/ns/im/
  xmlns:myex:location="http://example.com/presence/"
  entity="pressoneone@example.com"/>
<tuple id="bs35r9">
  <status>
    <basic>open</basic>
    <im:busy/im:"
  <myex:location>home</myex:location>
</status>
<contact priority="0.8">
  <mailto:someone@mobilecarrier.net</contact>
</tuple>
</presence>
```

Figure 7. Example of PIDF-file

6.3.3 Combination
We have chosen to use a combination of Geopriv and PIDF to communicate location and presence information. As mentioned, PIDF can be extended with new types of information. RFC 4119 [20] shows us that PIDF can be extended to carry Geopriv information. With this extension PIDF is optimal for sending context information gathered by a mobile phone.

In RFC 4119, PIDF is extended with a new complex type called ‘geopriv’. There are four sub elements that are encapsulated within the geopriv-element: ‘location-info’, ‘usage-rules’, ‘method’ and ‘provided-by’.

The ‘location-info’ element stores GML-information with location. The ‘usage-rules’ tell us how to handle the location-information (refresh-time etc.), this is outside the scope of this research.

The optional ‘method’ element describes the way that the location information was derived or discovered. The values are limited to the ones that have been registered:

- A-GPS: GPS with assistance
- Cell: location of the cellular radio antenna
- DHCP: provided by DHCP
- GPS: Global Positioning System
- Manual: entered manually by an operator or user, e.g., based on subscriber billing or service location information
- Triangulation: triangulated from time-of-arrival, signal strength, or similar measurements
- 802.11: 802.11 access point (used for DHCP-based provisioning over wireless access networks)

The optional ‘provided-by’ element describes the entity or organization that supplied this location information.

This RFC only shows that it is possible to use Geopriv in PIDF, but it leaves the implementation up to the developers. Because we have a specific purpose: developing for mobile web browsing, we can indicate which elements we need. For Geopriv location-info, we need GPS coordinates, GSM Cell-id, and enumerations of higher level locations. For activity-status, we need to store the messenger status and standardized calendar appointments.

Mobile phones and operating systems may add new context information in the future. This information can be included in the data format by adding a new simple or complex type in PIDF.

The securing of the information carried in PIDF is described in section 4 of RFC 4119. To secure the whole XML-file, it is possible to use S/MIME, as described in RFC 3851 [21].

In the PIDF example in Figure 7, we saw the entity-tag with the messenger-address of a user. To respect the users-privacy, this entity-tag won’t contain contact-information of a user, but a session-id for the server. A unique session-id is used for every website and can be changed regularly to guarantee the privacy of a user. Which context information is exactly included, will be determined by the plug-in, based on the user preferences and privacy-options. By default, only information with a low or medium spatial resolution is sent.

6.3.4 Encapsulation
There are several options to send the PIDF-file from the web browser plug-in to the web browser plug-in. Because we are dealing with web content, HTTP seems to be the most logic option. Within HTTP there are several options to send extra information. First, the PIDF-file can be included in an extra header in the HTTP get request, when requesting the web site. Secondly, the PIDF-file can be sent in the body of a HTTP post request. Thirdly, the PIDF-file can be included via MIME in HTTP get.
We choose to send the PIDF-file in the body of the HTTP post request. It is not possible to send the information via an extra header in HTTP get request, the PIDF-file has a lot of data, and consists of more lines. There is no formal limit to the size of the HTTP-header, but in practice the header should be on one line with a maximum of 4kB.

Figure 6 shows schematically how context information is encapsulated in the HTTP post request, Figure 9 shows the request itself. When the client also wants to send other HTTP post data, an extra MIME-header can be used to separate the different contents.

Figure 8. Protocol message with location and activity encapsulated in PIDF and HTTP-message.

8. RELATED WORK

Frameworks like Johnson [12] provide functionality for rapid development of context-aware applications on mobile phones. However, these frameworks don’t provide support for mobile web browsers, and have no protocols to communicate context information from a web browser to a server.

The data format in RFC 4119 is developed to carry presence and location information. The RFC however does not give an implementation, and leaves this up to developers. This paper selects specific functionality for mobile web browsing from an existing framework, and puts it in an architecture for exchanging this information, thereby enabling easier and richer context-aware mobile web browsing.

The Location Aware project has the same purpose as this research, but has a different perspective. It tries to make a JavaScript API for mobile web browsers, so context-information is only locally available, and no protocol is needed. The Location Based Web project also develops a plug-in for Windows Mobile Internet Explorer, but only uses location-information, sent via an HTTP-header. Our architecture uses location but also activity-information and sends information via a HTTP post request with context information formatted in extended PIDF.

Another important aspect is the privacy control. Context information is only sent when a user explicitly enables it. This also ensures data traffic is only used when a website is context-aware. The plug-in can enable a user to exactly specify which context information is send, but this requires extra configuration. By default, only context information with a medium or low spatial resolution is disclosed.

This architecture can also be used in AJAX-websites: a XMLHttpRequest posts the context information to a server, and the server will reply with the adapted part of the website. We think AJAX won’t be used much for mobile websites, because they are small and mostly consist of content and little markup, so when new data arrives, nearly the whole website will be replaced. There will be little difference in bandwidth usage when reloading the whole website, or only a part of it.

A small investigation of Dutch mobile websites reachable via the T-Mobile portal shows that these websites usually have a size between 1 and 10 kB. The HTTP-post with context information uses about 1 kB. In the worst case, only once visiting a context-aware website of 1 kB, the HTTP post results in 100% extra data traffic. An example of visiting a website of 5 kB three times with only one HTTP post with context information results in 1 / (3*5) = 7% overhead. In general, one extra data packet is needed to send the context information for the first time, and 1 kB is needed every time the context information is sent. But because the website is adapted, some user input screens aren’t needed, and decrease the number of user requests. The packet size can be decreased by using another type of data format, but the round trip time of the packet is more significant.

7. DISCUSSION / EVALUATION

We have designed an architecture for sending context information (location and activity) from a mobile web browser to a web server. In section 3, a number of criteria for the architecture are stated. In this section, we will check the architecture against these criteria.

The goal of this architecture is to need less user input while browsing the web on a mobile phone. We have seen that a user explicitly needs to enable the context-awareness of a website. This needs an extra user event once. This can be compensated by the adapting website, which results in immediately showing the requested information, in case the website is adapted correctly according to the context information.

Another important aspect is the privacy control. Context information is only sent when a user explicitly enables it. This also ensures data traffic is only used when a website is context-aware. The plug-in can enable a user to exactly specify which context information is send, but this requires extra configuration. By default, only context information with a medium or low spatial resolution is disclosed.

This architecture can also be used in AJAX-websites: a XMLHttpRequest posts the context information to a server, and the server will reply with the adapted part of the website. We think AJAX won’t be used much for mobile websites, because they are small and mostly consist of content and little markup, so when new data arrives, nearly the whole website will be replaced. There will be little difference in bandwidth usage when reloading the whole website, or only a part of it.

A small investigation of Dutch mobile websites reachable via the T-Mobile portal shows that these websites usually have a size between 1 and 10 kB. The HTTP-post with context information uses about 1 kB. In the worst case, only once visiting a context-aware website of 1 kB, the HTTP post results in 100% extra data traffic. An example of visiting a website of 5 kB three times with only one HTTP post with context information results in 1 / (3*5) = 7% overhead. In general, one extra data packet is needed to send the context information for the first time, and 1 kB is needed every time the context information is sent. But because the website is adapted, some user input screens aren’t needed, and decrease the number of user requests. The packet size can be decreased by using another type of data format, but the round trip time of the packet is more significant.
also gathers information from a mobile phone, but does not use it for mobile web browsing.

9. CONCLUSIONS AND FUTURE WORK

We have created an architecture for communicating context information gathered by a mobile phone from a mobile web browser to a web server. This architecture consists of a plug-in for a mobile web browser, a protocol for exchanging the context information and a plug-in for the web server. The protocol consists of a HTTP post request, with information enclosed in PIDF. Thanks to RFC 4119, PIDF can carry activity / presence information and Geopriv location information. The plug-in gathers context information from the operating system, possibly interprets it, and puts it in the protocol to be sent to the web server. The plug-in also takes care of user privacy, by enabling the user to control which information is sent to specific websites.

It seems that the extra packets and data traffic are compensated because the website is adapted, so some user input screens can be skipped. The architecture doesn’t decrease the bandwidth or data traffic, but offers the user a faster navigation and adapted websites once context-awareness is enabled.

The proposed architecture hasn’t been tested yet. To test the functionality and performance on the number of user clicks, privacy and used data traffic, we advise to prototype the architecture. Windows phones support plug-ins for their web browsers, and would be a good start for the prototype.

This paper doesn’t pre-define categories or complex PIDF types for location (and activity. Research can be done on pre-defined categories. Research can also be done on interpretation context information gathered by the mobile phone, before sending it to the web server. For example converting a room number in an appointment to the category 'work'.

ACKNOWLEDGMENTS

I would like to thank my supervisors, Tom Broens, Martin van Sinderen and Maarten Wegdam for their input and advice while writing this paper. I would also like to thank the students from the track group for their peer-reviews.

REFERENCES


