ABSTRACT
Nowadays Presence Enabled Services, like Instant Messaging or multipart games, are taking a growing importance. One characteristic of these services is that they need user’s presence information, understanding this presence in a wide sense, with many parameters. The problem of this information is that it generates a big amount of signaling messages, that increases exponentially with the number of users involved in the service. When users are connected through wireless links of cellular networks, this distribution can be a critical drawback. In this paper we present a distributed architecture of managing and distributing presence information, in contraposition with the standard one that is completely centralized. The system permits to use many optimization techniques in order to minimize at the maximum the amount of presence information to be transmitted to the final user that will use the cellular network as access network.

KEY WORDS
Presence optimization, instant messaging, SIMPLE, SIP, signaling, cellular networks, proxy.

1. Introduction
Presence is a well known concept in Internet and widely used in applications as Instant Messaging (IM) or Push to talk over Cellular (PoC). Presence enables users to know if a partner is present in the network and able to establish a new communication. A part of this basic model where users publish their state (online / offline / busy), the concept of presence is moving forward, towards a much more global and flexible idea that includes the user’s context useful for allowing to control and adapt communications in a personalized and efficient way. Consequently, presence information may contain geographic localization, activities that the user is doing at a moment, capabilities of the device or devices that can participate in the communication, ambient conditions and user’s communication preferences.

Some authors have already been talking about the concept of presence understood as a kind of context. In [1] Dey defines the context as “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”. Although some definitions may induce some confusion between context and presence, we consider the context as a wider concept than presence. User’s presence is a subgroup of user’s context that contains all the information related with user’s condition, capacity and preferences of communication using some kind of service, and with the characteristics of this service also. In this way, network entities can use this information to take intelligent decisions about beginning, course and ending of communications with that user.

Presence Enabled Services (PES) are those applications with an ubiquitous nature that adapt their behavior in function of user’s presence information. The most known of them are Instant Messaging and chats [2] but we can find others like PoC, online multipart gaming, voice and video conferencing, dating, ... [3].

Nowadays tendency is that PES, initially thought to be used in devices connected to fixed networks, are evolving to be used in wireless networks. The fact that the 3G cellular network Universal Mobile Telecommunication System (UMTS) is finally completely deployed and the future 4G, where all wireless networks will be totally integrated, made the usage of mobile devices very attractive to users. This situation has not been unnoticed for organizations like 3GPP (3rd Generation Partnership Project) or OMA (Open Mobile Alliance). The last one, with the purpose of standardize and homogenize PES has specified IMPS (Instant Messaging and Presence Service), and 3GPP is doing a big effort developing IMS (IP Multimedia Core Network Sub-System) in order to integrate multimedia IP-based services into UMTS network. One part of IMS specification is the management of presence enabled services in UMTS.

Apart of compatibility between different systems, distributing presence information over wireless networks has some more troubles. In group communications using
PES it is often necessary that presence messages with the same content is distributed to several receivers and even when users are not communicating, these messages keep being transmitted. These is because presence services send and receive periodic updates of user situation. This fact is not a serious issue in fixed networks but it can become critical in mobile environments, firstly because it consumes device’s batteries and secondly because in the majority of cellular networks the billing is in a per data transmitted basis instead of plain tariffs, and in most cases the user will not agree on paying money for data that he is not conscious that he has transmitted. And even more if we take into account that presence update messages increase exponentially with the increase of subscribers in any network [3]. There is also the problem of management nodes saturation like the case of the CSCF (Call State Control Function) of the IMS platform where presence traffic can represent up to the 50% of the total [4].

Therefore presence traffic optimization will be required if PES is willing to be widely accepted over wireless networks. This fact has already been studied in [5], and there are several authors that have proposed some optimization procedures like in [6], [7] or in [8].

In this paper we present a newer developed solution in order to optimize the amount of messages to be delivered to user’s devices for Presence Enabled Services. Our solution is based on a decentralized architecture where each user has a personal proxy which handles rules about the transmission of presence messages to devices where PES are running. This personal proxy is a software that should be installed in a computer owned by the user and will act as a presence message redistribution point between PES running in user’s devices and presence entities of presence systems. Although we focus the usage of the optimization capabilities of the Personal Proxy towards wireless devices it can be used to reduce the amount of presence messages delivered to any kind of device that runs PES.

The reminder of this paper is organized as follows. Next section evaluates the different technological possibilities in order to develop the personal proxy. Section 3 describes the SIP/SIMPLE architecture which will be used in order to build the Personal Proxy. The architectural model of the system is explained in Section 4. The section 5 reviews software technologies used to develop the described proxy personal, and the most important findings of this work are emphasized in the conclusion.

2. Evaluation of presence distribution technologies

Nowadays there are different solutions to provide presence information that are usually tied to Instant Messaging services because this one is the most popular presence application. In order to develop the Personal Proxy we have evaluated the most extended open technologies, focusing our attention to the fact that should be used over wireless environments. In particular we have considered Jabber/XMPP, IMPS and SIP/SIMPLE.

The eXtensible Messaging and Presence Protocol (XMPP) was developed in conjunction with the Jabber platform to provide an open system of Instant Messaging and Presence (IMP) and later was adopted by the IETF (Internet Engineering Task Force) and described in RFCs 3920[9] and 3921[10]. XMPP is a protocol based on XML (eXtensible Markup Language) which makes it quite bulky to transmit and requires some computation capacity to parse and interpret messages that can be problematic in some low capabilities mobile devices.

IMPS (Instant Messaging and Presence Service) is a group of specifications defined by OMA and to be used over mobile devices. IMPS follows a client-server architecture with two basic protocols: SSP (Server-Server Protocol) and CSP (Client-Server Protocol). IMPS is a quite complex specification written in 16 documents that describe the architecture, requirements, usage cases, the protocols and the messages of presence information and messaging.

IETF developed SIMPLE (SIP for Instant Messaging and Presence Leveraging Extensions) as a group of extensions of SIP (Session Initiation Protocol) for supporting presence information and instant messaging. SIP is a signaling protocol, also developed by IETF, to initiate, modify and terminate multimedia sessions for interactive communications with two or more participants. It is an application protocol, text based, that follows a transactions model with request / reply messages. SIP was adopted by 3GPP to be the session initiation protocol and SIMPLE to be the presence signaling protocol for its IMS platform. Therefore IETF and 3GPP are actively collaborating to improve SIP/SIMPLE specifications for its usage on mobile environments.

In order to compare these protocols we have taken into account several aspects: their market implantation, their ability to adapt to mobile environments, their specification complexity, their functional and security possibilities and their interoperability with other presence protocols.

The main founders of IMPS, Ericsson, Motorola and Nokia, are the companies who support it. There are cellular phones of these different brands with IMPS facilities in the market, and nowadays, there are around twenty different companies that provide IMPS clients and servers.

SIP/SIMPLE is present in widely used solutions implemented by Microsoft (MSN Messenger), IBM and Yahoo, although there are much more companies with other freeware and private applications. SIP has been also chosen by 3GPP to be used as the session management
protocol and SIMPLE for Instant Messaging and Presence protocol for its IMS platform. We have to stress the implementation of these protocols in the Java platform (JSR 180) and the JAIN (Java API for Integrated Networks) initiative.

Jabber/XMPP is nowadays a little more extended than SIP/SIMPLE due to the fact that was the first standard to be adopted for Instant Messaging. Some companies that use it are Google (GoogleTalk), Hewlett-Packard, Intel Capital, ... Nevertheless, the tendency is that SIP/SIMPLE applications are growing faster than Jabber/XMPP ones, mainly due to the acceptance of SIP/SIMPLE by the Java community and the usage of SIP for other purposes like IP telephony.

Regarding mobile environments adaptation, SIMPLE is based on a transactions model with short text messages, and can use TCP (Transport Control Protocol) or UDP (User Datagram Protocol) as transport protocol. The possibility of usage of UDP is a substantial advantage for mobile environments because is much lighter than TCP and can be used on multicast communications. On the other hand, XMPP is a XML streaming protocol, which makes messages much larger, and works only over TCP that requires more bandwidth. On the contrary of these two protocols intended initially to work over fixed networks, IMPS was designed specifically to run over wireless environments. However, the fact that IMPS platform was developed by the association of several manufacturers results in a complex recommendation to satisfy the interests of everybody, and sometimes it is not so light to run properly over mobile devices. This results in SIMPLE efficiency on mobile environments being better than IMPS’, regarding bandwidth requirements and delays.

At security level, SIMPLE and XMPP have a similar level of facilities while IMPS is a little below because it doesn’t give a consistent support to the message integrity and confidentiality.

At functional level, IMPS, SIMPLE and XMPP offer enough possibilities for a instant messaging and presence system. Nevertheless, SIMPLE and XMPP have several facilities not available in XMPP yet, and presence information over SIMPLE is the most complete one.

At complexity level, we can say that IMPS is the most complex, after that SIMPLE, and the simplest is XMPP.

All three protocols, are built according to the recommendations provided by the IETF working group IMPP (Instant Messaging and Presence Protocol). This fact facilitates the implementation of gateways for the interoperability between them.

The fact that SIMPLE has been chosen by the 3GPP for its IMS platform and the wide support it has in Java, causes to feel that it will be used by programmers to build numerous new generation services over cellular networks. This, plus its better performance at network and computational levels than XMPP and IMPS made that the authors selected SIMPLE in order to build the Personal Proxy presented in the current paper.

3. SIP/SIMPLE protocol architecture

SIP is a signaling protocol that provides a general framework for event notifications [11], whose purpose is allowing SIP nodes to receive notifications from remote nodes indicating that an event has occurred. The framework does not define the nature of the events causing notification. In order to upgrade SIP to support specificities of presence notifications, IETF developed SIMPLE, which main document is “A Presence Event Package for the Session Initiation Protocol (SIP)” [12].

The basic SIP network uses a model of client-server. The client endpoint is called the User Agent (UA) and the server side may include four different blocks:

- Register: it receives REGISTER requests from UA and records the address where the subscriber is located in the Location service database. When UA moves to different networks, the register always holds the current contact information.
- Proxy server: it processes INVITE requests from calling UA to called UA. An INVITE message is sent whenever a user wants to initiate a session with another. The proxy server either handles the request or forwards it to other servers, after having resolved the location of the called UA in the Location service database.
- Redirect server: it accepts INVITE requests from calling UA, queries the Location service database, and returns a new address to that UA. The calling UA will forward itself the INVITE request to this new address.
- Location service database: a database where registers update UA information and Proxy or Redirect servers retrieve it.

In order to add the Presence Service to the basic SIP network, new modules and messages are defined inside the SIMPLE framework. Also, Redirect and Proxy servers will deal with the new user’s SUBSCRIBE requests. A SUBSCRIBE request is sent whenever an entity wants to receive the presence of another one.

Firstly lets remember three generic definitions of the endpoints involved in Presence services:

- Presence Entity or Presentity: is an entity that provides Presence information to a Presence service.
- Presence Entity or Presentity: is an entity that provides Presence information to a Presence service.
- Watcher: is an entity that requests Presence information about a presentity.
- **Subscriber**: is a form of Watcher that has asked the Presence service to notify of changes in the Presence information of one or more Presentities.

Note that the presence service will only be provided to those entities that have explicitly requested it. Entities willing to receive presence information of a given presentity subscribe (with SUBSCRIBE message) to the “presence service” of that entity, so that they will be notified (with NOTIFY message) when a presence event related to that entity, e.g. user has registered from a new device, occurs.

Typically, each presentity is being observed by many watchers, each of which may see different degrees of detail for that presentity according to the level of priority that they have and that is managed by the Presence Agent. SIMPLE framework defines two logical entities to provide Presence service:

- **Presence User Agent (PUA)**: entity that manipulates the Presence information for a presentity. It is responsible to publish this information (with PUBLISH message) to the Presence Agent. One presentity can have different PUAs. For instance a presentity can be a user that uses an Instant Message Service through several devices: personal computer, laptop, PDA and cellular phone. There will be one PUA in each device, all of them providing information of the presentity.

- **Presence Agent (PA)**: entity in charge of managing presence information of a presentity sent by PUAs, processing SUBSCRIBE requests sent by watchers and notifying subscribers about presence changes in the presentity using NOTIFY messages. The PA may apply filters to NOTIFY requests, in this way, the presentity can define some rules on which subscribers will receive certain presence information.

It is also necessary to establish a new database with Presence information that will be accessed by the PA. Therefore the workflow of the whole system is the following:

- The PUA running in the device currently in use for the user registers itself in the Register. The Register stores the address of that presentity in the Location database and optionally notifies at the PA who will update the Presence database.

- When a watcher wants to be a subscriber and get the presence of a presentity, it sends a SUBSCRIBER message to his Proxy Server and that one forwards it to the PA of that presentity. The PA will authentify and authorize the subscription, and in case it’s positive it will send a NOTIFY message with presence information to the subscriber.

- Anytime the PUA detects a change in the Presence of the user, it sends a PUBLISH message with the new data to its Proxy Server and that one forwards it to the PA. The PA updates the Presence database and notifies the subscribers according to the preferences and restrictions set by the presentity.

The physical entity that manages SUBSCRIBER requests acting as a PA is called Presence Server (PS). Additionally, in the server side it also necessary to have a Register, a Proxy server and the Location and Presence databases.

### 4. A distributed architecture using a Personal Proxy

In the standard architecture, each Presence Service domain will have a centralized Presence Server, like in the IMS platform. Nevertheless, this centralized model has the drawback that when the system is used by many users the Presence Server may represent a bottleneck, moreover, presence updates will generate high volume of signaling traffic in the operator network and may represent additional monetary costs for the users. Another problem is the privacy of presence information, that will be stored in the service operator premises and users may not agree on having his presence data handled by others. More problems arise when the user’s device is connected through a wireless link, specially over public cellular networks like GPRS (General Packet Radio Service) or UMTS, that usually have bandwidth constrains. In these cases presence information over the wireless link should be optimized at maximum and this is very difficult to be done with a centralized server.

In this paper the authors present a new proposal that follows a totally distributed architecture that will reduce the problems stated in the previous paragraphs, introducing several algorithms to optimize the signaling traffic generated by presence information.

---

**Figure 1. Distributed architecture using Personal Proxies.**

Our solution is based in a Presence Personal Proxy that acts as an individual Presence Server for a certain presentity, in that case one user. Therefore, each user will have his own Presence Personal Proxy running in a computer of his own, and the different devices that he employs to run PES will use the Presence Personal Proxy...
to manage all presence information for that user (figure 1).

The system has been structured in two operational blocs. Firstly, the module that acts as Presence Personal Proxy and runs in the fixed computer, and secondly, the module called Proxy Client that runs in user’s devices, has been specially focused to be used in those devices with constricted capabilities, like cellular telephones or PDA, that are who will require a better optimization of presence signaling, and the proposed solution will have more impact.

The Presence Personal Proxy is an independent module that runs in a fixed computer and whose requirements of hardware and software are very low. The Proxy Client is presented as an API that programmers of PES may use to solve all tasks related with presence information, in a complete transparent way for the rest of the application. The Proxy Client performs very few operations and relays in the Presence Personal Proxy who will be in charge of nearly all intelligence of the system. Some examples of the provided API are functions like “publish my presence to all my resource list”, “subscribe me to the presence of all my contacts”, .... The whole system may be seen as a middleware that gives support of presence information to any presence enabled service.

4.1. Advantages over cellular networks

The benefits of the solution may be divided in different blocs. First of all, the presence service operator doesn’t require a huge computing center acting as a Presence Server and this possible bottleneck will disappear. Secondly, the privacy of users will be respected because all their presence data will be kept in computers of their own. And thirdly, our platform uses many ways to optimize the traffic between the Proxy Client and the Presence Personal Proxy that will reduce the presence signaling traffic in the operator’s network and, where is more critical, in the wireless link of portable devices.

The strongest point of the platform are the different ways that it uses in order to reduce the amount of presence traffic that will travel between the Personal Proxy and the Proxy Client. These are:

- The Personal Proxy has the intelligence of the system and will take all decisions about user’s presence information, which includes from which devices the user connects and which services he is using. The Proxy Client only manages the information that is totally dependent of the user in that moment (e.g. where the user is, which mood he has, ...). Once the user registers in the Personal Proxy, this one launches a procedure to detect automatically the maximum of user’s presence, for example this procedure detects the hardware capabilities of the device where the Proxy Client runs and which PES the user is able to use. Lately the Personal Proxy uses this information to notify subscribers with a complete presence of the user.

- The management of the user’s resource lists are responsibility of the Personal Proxy. When a user connects to one PES, the Proxy Client will send a message to the Personal Proxy and that one will subscribe to the presence of all members of the resource list, saving in this way the transmission of many messages from the Proxy Client to the Personal Proxy.

- Watchers authorization policies: In the standard model, presentities should authorize any SUBSCRIBER message from watchers that arrives at the PA. Our system provides a HTTP (Hyper Text Transfer Protocol) user interface in the PA that enables users to establish certain authorization policies that will apply immediately when a SUBSCRIBER message arrives, and in this way the traffic from the Personal Proxy to the Proxy Client is reduced. Obviously, the option of asking to the user it is also possible.

- Partial publication/notification of presence information: By default, presence delivered using SIMPLE is represented in the Presence Information Data Format (PINF). A PINF document contains a set of elements, each representing a different aspect of the presence being reported. When any subset of the elements change, even just a single element, a new document containing the full set of elements is delivered. The Proxy Client and Personal Proxy implement the IETF draft draft-ietf-simple-partial-publish-05 [14] which gives a mechanism with which the PUA is able to publish to the PA only the presence changes made since the last publish. In this way, the amount of bytes transmitted on the wireless link between the Proxy Client and the Personal Proxy is reduced considerably. In the other hand, the Personal Proxy follows IETF draft draft-ietf-simple-partial-notify-08 [13] that proposes an extension allowing the PA delivers to the watchers only the presence data that has actually changed and in this way reducing considerably the document size exchanged between the Personal Proxy and watchers.

- Source throttling: This procedure consists in the limitation of the number of PUBLISH messages that the Proxy Client may send, by establishing a minimum time interval between two of these messages. This is useful when changes in the presence of the Proxy Client occur very often. This technique is used in combination with the presence aggregation, meaning that if two changes occur inside one of these time intervals, both changes will be published using the same message.

- Event throttling: This mechanism is similar to the previous one but to be used on the Resource List Meta-Information (RLMI). RLMI is a document that describes the state of the virtual subscriptions associated with a list subscription. In this way, although the Personal Proxy gets NOTIFY messages from other presentities, it
will not communicate them to the Proxy Client until the timer expires, aggregating in a single message all notifications received during the period. Though the user loses a certain degree of precision on presence information due to the added delays, it will reduce substantially the amount of data to be transmitted.

- Partial notification of resource list: the Personal Proxy only sends partial RLMI documents to the Proxy Client. A partial RLMI document is a partial resource list with only those resources that have really changed since the last RLMI document notification.

- Event notification filtering: With this procedure the user can indicate to the Personal Proxy which kind of presence information from his resources wishes to watch. Therefore, the Personal Proxy will only notify about these events, filtering the rest of information that arrives from user’s resources, and reducing the amount of traffic.

- Efficient codification of information: In the standard model, SIMPLE messages are in text format and presence information is transferred using XML documents attached to these messages. In the presented platform all XML documents are codified using WAP Binary XML (WBXML), that reduces considerably its size, and was defined by the W3C (World Wide Web Consortium) as a part of the WAP (Wireless Application Protocol) protocol architecture.

4.2. Architecture

The middleware has a logical architecture divided in three logical layers as shown in figure 2.

The application layer contains the interfaces to the service and is available from the Proxy Client. The Presence API will provide all those functions related with presence. The HTTP API offers a user interface to configure the different parameters of the Personal Proxy: resource list management, event notification filtering policies, user’s preferences, ...

The management layer contains the procedures to process, aggregate, storage, generate and control presence information. It is composed by the following modules:

- HTTP dispatcher: It receives HTTP requests and distributes them to the modules in charge of processing them.

- Presence manager: It controls and aggregates the user’s presence information. Also it is in charge of generating the documents to be sent to subscribers.

- Resource list manager: It controls all the information associated to the resource list of the user. It processes and aggregates notifications of resources to which the user is subscribed.

- Watcher information manager: It contains all the information about user watchers, and will control new subscriptions

- Request admission police: It administers what communication types the user is willing to accept, depending on several parameters like content types of the communication or caller users. In this way the user can restrict communication requests that other users may send to him. Restriction rules for this module will be set by the user through the HTTP API.

- Presence publication police: It administers which presence information is to be send to the subscribers. Some subscribers only see part of the user’s presence. Restriction rules for this module will be set by the user through the HTTP API.

-Optimization Handler: It controls and applies the optimization policies set by the user in order to reduce the amount of presence information that the Personal Proxy sends to the Proxy Client and other Personal Proxies.

The SIP/SIMPLE layer is in charge of implementing these two protocols and transmitting and receiving their messages. It is composed by the following modules:

- Register: This module will store the localization of the user when he registers in the Personal Proxy by a REGISTER message.

- Resource list server: It will send and receive all SIMPLE messages related to the resource list management. It will send SUBSCRIBE messages to the resources of the list, and will send the resource list presence to the Proxy Client by NOTIFY messages.

Figure 2. Logical architecture of the Personal Proxy.
Presence server: It manages the SUBSCRIBE/NOTIFY messages related to user’s presence publications and also to user presence notifications to user’s watchers.

- Watcher info server: This module notifies the user about new subscribers, requiring positive or negative authorization.

- Proxy: It forwards user service requests (not related to presence) towards other users. Two service examples are instant messaging or session management.

- Presence dispatcher: This module concentrates all SIP/SIMPLE requests from the user and forwards them to the appropriate module.

5. Software Platform
The developed middleware is based on the Java Platform, the Personal Proxy has been programmed using Java Platform Standard Edition (J2SE) and it has been developed on top of a SIP/Method Application Server that is JSR-116 and JSR-289 (SIP Servlet APIs) compliant SIP Application Server runtime engine. The SIP Servlet API is Java extension for SIP servers, which is similar in spirit to HTTP servlet API, and offers many benefits like performance, platform independence, high level abstractions, and so on. The Proxy Client has been implemented using Java Platform Micro Edition (J2ME) with CLCD (Connected Limited Device Configuration) and MIDP (Mobile Information Device Profile). A new Java API, JSR 180 (SIP API for J2ME) has been used to implement the SIP communication between client mobile devices and personal proxies. It is a generic library that implements partially the RFC 3261 (SIP: Session Initiation Protocol). Concerning presence information, the presence documents are encoded by XML format, and this format is based on plain text. For this reason, usually the XML documents have great size and require bandwidth and CPU time. As the Personal Proxy is designed to constrained environments, the bandwidth and processing time on mobile devices are critical, then we write presence documents by Wap Binary encoded XML (WBXml), a specification of OMA, that allows us to encode XML files by a binary way and so to save bandwidth. In addition, a XML Pull Parser is used, which improves the parsing performance on mobile devices.

6. Conclusion
In the standard centralized architecture for managing and distributing presence information for Presence Services is difficult to optimize the number of messages to be send to the final user. If the user is connected through a wireless cellular network this can be a critical drawback for the bottleneck that may represent and for the additional monetary costs for the users that it can generate. In this paper we have presented a new distributed architecture that permits to use many optimization techniques in order to minimize at the maximum the amount of presence information to be transmitted to the final user and that will be more useful when users access through cellular networks. Some of these optimization techniques are based on the new Internet drafts to send partial messages, or aggregating and filtering specific messages.

Acknowledgements
This research has been funded by the Spanish Government through CICYT project TIC2006-04504.

References