An Architecture for Location-based Service Mobility Using the SIP Event Model

Ron Shacham, Henning Schulzrinne
Department of Computer Science
Columbia University
New York, NY 10027
rs2194, hgs@cs.columbia.edu

Wolfgang Kellerer, Srisakul Thakolsri
Future Networking Lab
DoCoMo Communications Laboratories Europe
D-80687 Munich, Germany
kellerer, thakolsri@docomolab-euro.com

Abstract— **We present a location-based architecture, based on the Session Initiation Protocol (SIP) event mechanism that provides users with service portability between a wide variety of communication end-points. The architecture supports devices with different configuration formats, through the use of device controllers that manage all devices in an area. Several means of location discovery are used which, along with user scripts that specify desired device configuration scenarios, make configurations and personal data available as long as the user needs them.**

Keywords— **Mobility, Location-based Services, Ubiquitous Computing.**

I. Introduction

As Internet multimedia communication moves rapidly into the commercial realm, it is expected that the customized services that it offers on end systems will become available in many environments. For instance, IP telephones and video-conferencing equipment, already available in office settings, may find their way into hotel rooms and even replace traditional payphones. These devices will complement personal mobile communication devices such as networked PDAs and 3G wireless phones.

In this paper, we outline the use of standard Internet protocols to give users access to their personal configuration settings and services across the many devices that they use. This may include data that is needed for device connectivity and call processing, such as SIP proxy settings, end-system scripts (Call Processing Language (CPL) [10], Language for End System Services (LESS) [7]) and authentication information, as well as personal data such as address book entries, buddy-lists, speed-dial settings and ring tone audio files. Further, we describe the use of context-awareness, the interpretation of location and other user input to understand activity, for the automatic customization of devices. The framework supports a wide range of devices, configuring not only software endpoints, but also devices such as IP telephones, video displays and embedded devices which may not all support a single configuration format.

The following example illustrates the use of such an architecture. Charlie carries around his PDA which he uses for Internet media communication. He walks into the department conference room where his presence is detected and the public IP video phone is automatically configured with his settings since he is there for a scheduled teleconference with his clients. While using the phone, he makes a number of updates to his settings which are automatically sent to his PDA, synchronizing the devices without his request. He leaves the room, carrying his PDA, but takes no action to indicate his exit, such as swiping his identification card at the door. When he enters another room, his PDA sends a message to update his current location. His departure from the conference room is understood, and the video phone reverts to its default settings so that another user may not view his personal profile.

The example demonstrates several features of the architecture:

- Support of heterogeneous devices: Most devices, such as the video phone, cannot be programmed to use the presence of users to configure themselves. Moreover, they use varied configuration formats and transport protocols, and a single profile document cannot, by itself, be used to configure all devices. With full standardization and compliance many years away, our architecture makes use of Device Controllers (DCs) to manage a variety of devices in a given area, such as a floor in a building. A DC keeps track of user presence in rooms so that it knows when to configure a given device. It is authorized to access the roaming profiles of users and can translate the profile to a device-specific format and configure the device.
- Location-based configuration: When a user enters a room, the devices are immediately configured with his settings, and remain so as long as he needs them. The automatic configuration of stationary devices is achieved through a location-enhanced presence system to track the user's movements through the use of user tokens such as

iButtons [8], swipe cards and active badges. Private mobile devices may deconfigure themselves based on user location. For instance, once a mobile phone detects the user in another location based on updates that are made through the same user tokens, it may erase the user's configuration, which may contain sensitive data which could be used by an unauthorized user.

- Limiting configuration: In order to give users control over configuration, we define a mechanism for limiting the scope of devices that are to be configured. Owners of devices may define which users are authorized to configure and use them. At the same time, these authorized users may not always want to exercise this control capability. A person's movements over the course of a day are complex, and personal configuration that occurs indiscriminately can be annoying. Therefore, this system takes into account that the user desires to configure devices only in some locations and at certain times. For example, Bob may know that he only makes use of the conferencing equipment in the conference room during certain times when he has a meeting. In Section IV, we describe a language for expressing that the equipment should be configured at likely conference times, but not when he makes a quick dash into the room to fetch the notepad that he left behind during the conference.

We are building our framework around the Session Initiation Protocol (SIP) [1] event mechanism [3], with the support of other technologies such as SLP [4]. The SIP event notification is used to convey user presence and location, as well as configuration parameters. The Service Location Protocol (SLP) is used for discovering configurable devices based on their location.

The remainder of the paper is organized as follows: Section II discusses existing mechanisms for user profile mobility and data synchronization. Section III describes our architecture. Section IV discusses limiting device configuration through usage authorization, scenario specification and profile removal. We conclude the paper in Section V, and describe plans for future development.

II. Existing Service Mobility Mechanisms

There are alternatives to the event-based model. Subscriber Identification Modules (SIMs) [19], which are currently used to maintain profile information on GSM cellular phones, could be used to make the same preferences available on any device. This solution has several drawbacks. First of all, the concurrent use of multiple devices would require multiple SIM cards. Secondly, they do not provide a synchronization mechanism. Thirdly, SIM card readers are currently not available on most devices. Lastly, they are difficult to use on embedded devices such as ceiling-mounted projectors.

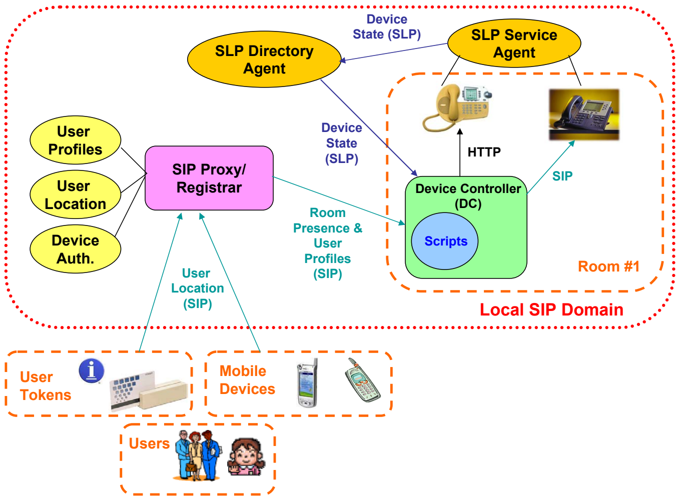

Fig. 1. *The architecture of the location-based service mobility framework.*

SyncML [6] is a recent attempt to standardize a format for synchronizing a wide range of user data so that it may be used simultaneously and updated on a number of mobile devices. The model used by SyncML does not fit into our architecture for a couple of reasons. First of all, SyncML uses a model of explicity "syncing" offline changes with a master version. There is no push mechanism available for online devices to receive incremental updates whenever they are made. Secondly, it is intended for the user's own devices and not to allow data retrieval by third parties. In our model, a visitor gives permission to a device configuration agent in the visited domain to retrieve his profile.

III. Architecture

Our architecture consists of four components: sources for location information, messaging for user profiles (uploads, downloads and change notification), configurable end devices and a device controller (DC). Figure 1 shows a diagram of the architecture. We describe each component below.

A. Sources of Location Information

The user's location may be updated by a number of different location sources, which we group into two categories: stationary and mobile. Stationary location sources are fixed hosts that identify users entering particular rooms and publish the information. We are currently using iButtons and swipe cards for user location monitoring with token readers installed at the entrances to rooms. We are also using IR/RF badges that receive the room's location through an infrared interface and forward the user's SIP

URI and his location to a central monitor through a radio frequency. Mobile location sources are handheld devices owned by users which receive their current location through Bluetooth beacons located in the room, a DHCP location option [20] or a Global Positioning System (GPS) monitor on the device. Once acquired, the location state information is sent to the SIP server through two different mechanisms. Individual user location is sent to the user's SIP URI using the SIP PUBLISH [2] method and is represented in the message body as a location object with civil or geodesic information [12]. This message updates the user's general presence state and is seen by his presence watchers.

We have also introduced the concept of a "room state," defined by parameters that include a list of all users who currently occupy it. Rooms are represented by SIP URIs such as "sip:conf_room1@example.com," and REGISTER messages are sent upon entrance of a user, listing him as a contact of the room. This update can be done easily with iButtons that have memory which can be used to record the user's SIP URI. However, some user tokens contain only other types of identification such as a student ID number that is stored on a magnetic swipe card. For these tokens, a mapping must be done before sending the REGISTER. To avoid having every token reader access a central database, we are using SIP messages to retrieve the mapping. A REGISTER message is sent to a dedicated mapping server that looks up the SIP URI of the user who owns the given token, and returns it in the Contact header of the response. The token reader then sends the REGISTER message described above, with this URI listed in the Contact header.

Whereas location publishers such as iButton readers, which are trusted parts of the organizational framework, can easily be authorized to update room presence, allowing a visitor's device to do this gives him the ability to later manipulate the room presence from outside. To avoid this, a DHCP server or Bluetooth beacon could send a time-based secret to allow the device a short window during which it may add the user to the room's state.

B. User Profile Access and Update

Seamlessness across devices is made possible by having a roaming user profile, represented as a set of XML documents such as a general user profile document, an address book and a buddy list. The profile must be accessed and updated by any of the user's devices and by third party entities, such as Device Controllers, discussed in Section III-C. The uploading of profile information, both as whole documents and as incremental updates, has been specified in XCAP [14] using the standard HTTP PUT method. The documents of all users are represented in a tree on the

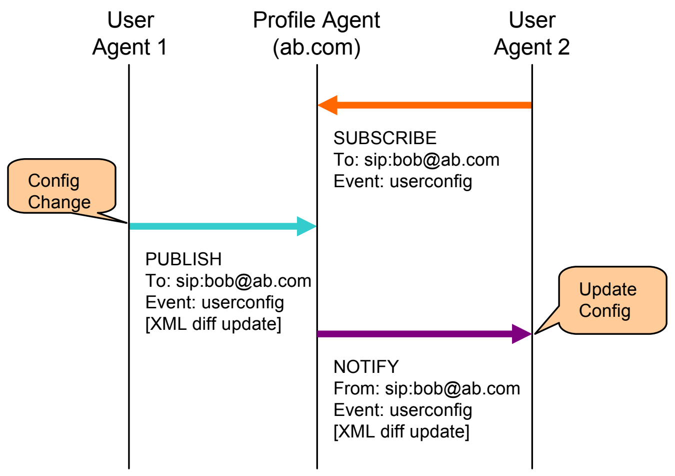

Fig. 2. *Protocol flow for a profile update to synchronize user's active SIP user agents.*

XCAP server, and the destination URI is used to specify both the document and a location within it, using the Xpath [15] format.

An alternative approach is to view this operation as a change to the user's state, like his presence, rather than an update of a document in the organization's hierarchy. This implies the use of a method such as PUBLISH, addressed to the user's SIP URI, to send an update request with an XML "difference" syntax, using Xpath to specify which element within the document to update or add. In this model, the SIP Event header is used to specify the type of configuration document with standard names such as "userconfig," for general user agent configuration or "addressbook." The two options are semantically the same and we have so far implemented updates using PUBLISH, so we use that model in our examples.

The SIP event framework, which uses the SUBSCRIBE and NOTIFY methods for the update of user state is also appropriate for updates of user configuration parameters, and is specified by [13] and [16]. In XCAP events, only a URI for the location of the update is sent in the NOTIFY body, prompting the receiver to make a GET request to retrieve the updated elements. In our model, the actual updated data is returned in the message body, with a separate URI returned when the data is too large for a single NOTIFY message, as in the case of a user's addressbook.

Many devices offer interfaces for the user to update his personal settings and, in our architecture, the changes are propagated immediately to update the profile, with any device he is using being synchronized without his explicit request. Even if the user does not concurrently use several devices, he may find the rather cumbersome interface of his mobile phone unsuitable for making a large number of changes and, therefore, may prefer to use his PC for changing his personal settings or policies. Also, since it may not be possible to propagate updates that the user makes on certain hardware devices, a different device would be used to make changes that should be propagated, with these changes being reflected on the hardware device as well. PUBLISH messages update the user state and subsequent NOTIFY messages are sent to all subscribing devices, informing them of the changes so that they can update the data locally. We demonstrate this protocol flow in Figure 2.

C. End Devices and Device Controllers

Configurable communication endpoints fall into two categories: programmable and non-programmable. Programmable endpoints are software entities that can be extended to support subscriptions to SIP event packages, and handle events when they are received. They can also publish updates that are made on their user interface. They are, therefore, capable of configuring themselves based on a roaming profile, independent of a separate controller.

Many devices are non-programmable, and a separate controller is needed to configure them. The Device Controller (DC) determines the need to configure devices based on room updates described above, and subscribes to user profiles, which are used to perform the configuration. To configure a device, the DC translates between the XML format of the user profiles and the device's format, and transmits it using the device specific protocol. Current high-end IP phones receive configuration either through uploads to their built-in HTTP server as in the case of the Pingtel Xpressa or downloading from a tftp server, which is the method used by Cisco 7940/60 phones. More recently, Pingtel phones that receive SIP NOTIFY messages specified by [13] have become available. Through the device controller, any configurable device can be supported.

In order to automatically configure endpoints, the device manager must have an accurate picture of device location. Since the organization may be very large, and devices may be frequently added and moved, manual maintenance is unlikely to keep device location information up-to-date. One option for dynamic, decentralized maintenance of device location information is the Service Location Protocol (SLP) with location-based queries [17]. We are defining an SLP Service Template which includes common attributes of communication devices, such as the vendor and supported media, as well as location parameters. A Service Registration is sent by either the device itself, acting as an SLP Service Agent, or by a dedicated SLP Service Agent acting on behalf of the device. The DC, acting as an SLP User Agent, sends a Service Request to the Directory Agent, asking for devices whose locations are on a given floor which it serves. SLP also provides a mechanism that pushes service update events to subscribers [5]. This may be used to update the DC about the device topology without requiring it to poll. The DC sends an Attribute Request for each device in order to ascertain its room location for location based configuration, and the device model for proper mapping between the roaming profile format and the device format. Using a standard protocol such as SLP has the benefit of allowing other applications to make use of the device topology. For instance, a user who walks into the room in search of devices to which he may transfer his call can query for this using the same Service Request message used by the DC.

IV. Configuration Policies

We have included in our architecture three ways to limit device configuration in order to offer flexibility of use. First, owners of devices may limit access to certain users and users may give permission to the device controller to receive their profile (Section IV-A). Secondly, users may specify which devices should use their configuration and when (Section IV-B). Thirdly, after being set on the device, the profile is removed when it is determined that the user no longer requires it (Section IV-C).

A. Authorization

A device is associated with an owner or a group of owners who may give permission to others to configure it. The DC must find out who the owner is, and who he permits to configure the device. Our architecture already uses SLP for the DC to query for devices based on their location. The template described above could also contain the SIP URI of the device owner, which may represent an individual ("sip:bob@example.com") or a group ("sip:managers@example.com"). This URI is then used to access the owner's device authorization document, whose format would be similar to the one currently defined for granting authorization to watchers of presence [18]. The document is maintained as part of the device owner's profile data, like his basic configuration profile, buddy lists and scripts, so either of the two models described above in Section III-B may be used by the DC to access it. It is essentially a list of tuples of (device, user/group, priority) where users are listed individually or referenced as groups such as "sales" or "engineering." The priority may be used to resolve conflicts between multiple users who try to configure the same device.

A device user must grant permission to the device controller to access his profile. A local user must do this only once and has no need to revoke the authorization, so the permission may be set, for example, during the creation of the user's SIP account. A visitor, however, must authorize the DC when entering the new domain, and only for the duration of his visit. Several methods of granting such on-demand authorization to user-specific events are described in [21]. One is to use an authorization document in XCAP [14] like the one currently defined for presence. The document would then be updated through XCAP operations to grant and revoke authorization. However, this requires the user to explicitly revoke the authorization. An alternative solution is the use of a cryptographic ticket, which the user requests from a special "ticket server," and sends to the subscriber. The ticket identifies the authorized subscriber and the event. It can also include other details such as a time window for the authorization, which would make revocation unnecessary. The subscriber includes the ticket in the request for event subscription. Either of these methods could be used to grant authorization to the DC to subscribe to user profile data.

```
<LESS:LESS xmlns:LESS …>
  <Presence:presence-switch>
    <LESS:address-switch field="origin">
      <address is="sip:conf_room@example.com">
        <loc:attribute-switch field="location-source">
          <type is="bluetooth">
            <sub ref="configure" device-list="video-phone">
          </type>
          <type is="voluntary">
            <sub ref="configure" device-list="video-phone">
          </type>
        </loc:attribute-switch>
      </address>
    </LESS:address-switch>
  </Presence:presence-switch>
<LESS:LESS>
```

Fig. 3. *Example LESS script with extensions for use in customizing configuration behavior.*

B. Customizing Configuration Behavior

We also propose a way for the user to create policies that determine which devices should be automatically configured and when. The most basic policy would specify which devices in a room should be configured upon the user's entrance. A more sophisticated policy would attempt to model the user's activity patterns based on a number of parameters. There may be certain times when a user enters a public room for the purpose of communication, and only then should his entrance serve as a signal to customize the room's devices. The user could mention these times explicitly, or he could specify that they should be inferred from his calendar information. The manner in which a user's location is published may also have semantic meaning that may be utilized in modeling configuration behavior. A user's location may be conveyed through passive means, such as an IR badge or a handheld device, or actively (requiring user action) through a user token such as an iButton. The active location update could serve as an explicit request for room configuration if the user were to specify this. Other methods of location update could implicitly identify usage scenarios. For example, a user may sometimes enter a conference room at work in order to use the video phone to continue a session that he began on his PDA while in the car. Since the user's room location is published by the PDA based on information gathered through DHCP or Bluetooth, a rule could specify that this device should be configured whenever this type of update is sent. This information could be expressed by an extension to PIDF [11], such as a "location source" element for a given tuple, and could distinguish between "voluntary" (such as an iButton) and "involuntary" (PDA publishing location from a Bluetooth source) location publishing, as well as using specific tokens, such as "bluetooth."

One way to give the user this control is through a policy language that may be used to specify which devices are to be configured, and under what circumstances. We propose an extension to LESS [7], an XML-based language for specifying end-system signaling-related services. Whereas the normal usage of LESS is in a user's personal end system and allows users to specify events that trigger actions, these scripts are run by the device controller and are only called when a location update indicates that the user has entered a particular room. LESS has subroutines for specific device configurations, which are implemented on the device controller and called from the script. Figure 3 shows an example script for specifying how devices are to be configured for a user.

The user specifies that when he is found to be in the room with SIP URI "sip:confroom@example.com" the configuration behavior in the room will depend on how his location was determined. When location is determined through Bluetooth, he describes the scenario where he is walking into the conference room with his Bluetooth-enabled PDA, and wishes to configure the video phone before transferring his call. The subroutine called "configure" refers to a procedure executed by the device controller on specific devices, referred to by device-type tokens such as "ip-phone" and is not written by the script writer. In all other cases, the user only wishes the phone to be configured if he voluntarily updates his location.

C. *Profile Removal*

A good policy for removal of a user's settings from a device is necessary, particularly when dealing with settings that may be personal, such as a user's address book or buddy list. Both the device controller and the user's private mobile devices can use heuristics to interpret the location updates that they receive as part of the user's presence. An explicit way for the user to make his exit known is by activating a user token as he leaves. If he does not do so, other methods may be used to determine that the user has left. For instance, if the user has recently published his location with a user token or a mobile device in another place, he may be assumed to have left the room he was in earlier. A cruder safety net exists in the form of expiration of location updates, which would cause the device controller to assume that the user is no longer in the room, unless there is sign of activity. Finally, the user can use the rule-based language to specify his expected duration of device use, causing his settings to time out after that time.

V. Conclusions

We have presented an architecture for context-based device configuration that allows a user to make use of many different devices in his environment. The implementation of the system is within SIP, with its event mechanism being used to send location-enhanced presence information and user profiles. Device Controllers support service mobility across a heterogeneous set of devices. Sources of location information make the mobility automatic for a user in the proximity of a device. We are extending LESS in order to give users control over device configuration, and are introducing a format for specifying device authorization.

Our current implementation senses the location of users through iButtons and configures the Cisco 7940/60 and Pingtel IP telephones in the room based on their roaming profile. In the future, we plan to implement authorization, scripts for control of configuration and SLP location-based device discovery, while supporting a larger number of devices.

References

[1] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, R. Sparks, M. Handley and E. Schooler, **SIP: Session Initiation Protocol**. RFC 3261, Internet Engineering Task Force, June 2002.

[2] Niemi, A., **Session Initiation Protocol (SIP) Extension for Event State Publication**. Internet Draft, Internet Engineering Task Force, January 2004. Work In Progress.

[3] Roach, A.B., **Session Initiation Protocol (SIP)-Specific Event Notification**. RFC 3265, Internet Engineering Task Force, June 2002.

[4] Veizades, J., Guttman, E., Perkins, C. and Kaplan, S., **Service Location Protocol**. RFC 2165, Internet Engineering Task Force, June 1997.

[5] Kempf, J. and Goldschmidt, J., **Notification and Subscription for SLP**. RFC 3082, Internet Engineering Task Force, March 2001.

[6] http://www.openmobilealliance.org/syncml/

[7] Wu, X. and Schulzrinne, H., **Programmable End System Services Using SIP**. ICC2003 May 2003

[8] Dallas Semiconductor. ibutton. http://www.ibutton.com

[9] X. Wu, **Columbia University SIP user agent (sipc)**. http://www.cs.columbia.edu/IRT/sipc

[10] Lennox, J. and Schulzrinne, H., **Call Processing Language**. RFC 2824, Internet Engineering Task Force, May 2000.

[11] Sugano, H., Fujimoto, S., Klyne, G., Bateman, A., Carr, W., Peterson, J., **Presence Information Data Format (PIDF)**. Internet Draft, Internet Engineering Task Force, May 2003. Work In Progress.

[12] Peterson, J., **A Presence-based GEOPRIV Location Object Format**. Internet Draft, Internet Engineering Task Force, January 2004. Work In Progress.

[13] Petrie, D., **A Framework for SIP User Agent Profile Delivery**. Internet Draft, Internet Engineering Task Force, October 2003. Work In Progress.

[14] J.Rosenberg, **The Extensible Markup Language (XML) Configuration Access Protocol (XCAP)**. Internet Draft, Internet Engineering Task Force, October 2003. Work In Progress.

[15] http://www.wc3.org/TR/xpath

[16] Rosenberg, J., **A Session Initiation Protocol (SIP) Event Package for Modification Events for the Extensible Markup Language (XML) Configuration Access Protocol (XCAP) Managed Documents**. Internet Draft, Internet Engineering Task Force, June 2003. Work In Progress.

[17] Berger, S., Schulzrinne H., Sidiroglou, S., Wu, X., **Ubiquitous Computing Using SIP**, ACM NOSSDAV 2003, June 2003

[18] Rosenberg, J., **Extensible Markup Language (XML) Configuration Access Protocol (XCAP) Usages for Setting Presence Authorization**. Internet Draft, Internet Engineering Task Force, October 2003. Work In Progress.

[19] http://www.etsi.org (European Telecommunications Standard Institute)

[20] Schulzrinne, H., **DHCP Option for Civil Location**. Internet Draft, Internet Engineering Task Force, June 2003. Work In Progress.

[21] Trossen, D., Schulzrinne, H., **On-Demand Access Authorization for SIP Event Subscriptions**. Internet Draft, Internet Engineering Task Force, October 2003. Work In Progress.