Automating the Creation of Personalized Mobile Multimedia Services using Cellware

Kazuhisa Tanaka[+], Michael E. Kounavis[++] and Andrew T. Campbell[++]

kazu@tcd.hitachi.co.jp, {mk, campbell}@comet.columbia.edu

[+] Internet Systems Division Hitachi Ltd. 216 Totsuka, Yokohama 244-8567 Japan
[++] COMET Group, Center for Telecommunications Research, Columbia University, New York, NY 10027 USA

Abstract **–Personal mobility services are likely to be diverse and sophisticated in nature. We introduce 'Cellware', an architecture that automates the creation of personalized mobile services. Cellware attempts to capture the diversity of personal mobility services using a simple programmable abstraction called a 'cell'. Cells are event-driven middleware components that can be customized by scripts to perform personalized communication functions, such as call/session forwarding or event notification. Combining cells results in the dynamic creation of more complex personal mobility services. In this short paper, we provide an overview of our architecture and discuss some simple services that have been implemented using an initial implementation of cellware.**

1. Introduction

Recent trends toward IP telephony indicate that the Internet and PSTN will merge into a single information delivery system. It is likely that a variety of services will be offered in the future over many diverse transport environments. Services will include the web, electronic commerce, voice over IP, video conferencing and distributed games. Today, only service providers have the capability to program such services. Clients have limited choice to customize features of the services they are offered.

Another characteristic of multimedia services is that they are becoming more personalized and location-independent. The term 'personal mobility' refers to end-to-end communication services where the person, not the device, is the communication endpoint. In many systems that support personal mobility [1] [2] [3] call forwarding functions are driven by a single event type: receiver mobility. We believe that users should be able to specify their own types of events for driving the creation, transmission and consumption of multimedia content. Events should mirror physical world actions (e.g., mobility of people, scheduled appointments, accidents, and physical phenomena). Progress has been made to push computers and multimedia devices into the 'background' of everyday life [10]. However, ubiquitous computing while desirable is limited by the fact that the configuration, usage and maintenance of computing and multimedia devices is non-trivial.

We argue that a software model is needed for automating the creation of personalized mobile multimedia services in pervasive computing and communication environments. Such a software model should be capable of capturing the diversity of events that trigger the delivery of multimedia content, as well as the diversity of actions that take place when events occur. These events may include the forwarding of multimedia content to devices that best meet receiver preferences, the dynamic configuration of network computers in remote environments and the broadcasting of announcements.

It is likely that clients will play a key role in customizing multimedia services. Customization should be done in a seamless manner pushing computation into the background of people's lives instead of making it the focus of their attention. Customization should encompass registered events, multimedia content and user preferences. In addition, the behavior and mobility of people should be monitored taking into account social as well as geographical factors. Middleware infrastructure should be capable of capturing the many different social and geographical factors allowing the customization of these parameters on-demand.

The aim of our work is to manage the complexity associated with creating and customizing personalized mobile multimedia services. In this paper, we introduce the concept of a 'cell', a software component for service creation, which is event-driven and can be customized by client-defined scripts. Cells should be capable of executing on a wide variety of computing devices including laptop/hand held computers, microelectronic mechanical sensors, PDAs and PCs/workstations, adapting the environment to client preferences.

We envision networks having the capability to adapt to client preferences through the customization of a large number of cells running seamlessly in the areas where people work, study, and rest. This paper is structured as follows: In Section 2 and Section 3 we describe the Cellware architecture and our initial implementation, respectively. Following this, we discuss examples of service creation using cellware in Section 4 and Section

5. Finally, in Section 6 we provide some concluding remarks.

2. Architecture

2.1 Network Model

Our network model is shown in Figure 1. At the lowest layer, diverse transport technologies deliver digital information to end-systems. Transport technologies include the Public Switched Telephone Network (PSTN), wireless networks, Gigabit Ethernet networks and ATM networks. The IP protocol binds these technologies together into a single information delivery infrastructure. QOS extensions to IP forwarding (e.g., per-hop behaviors for differentiated services) are indicated as IP++ forwarding in Figure 1. On top of this transport environment, control and management planes support higher-level communication services. Network control functions include routing, resource reservation and call control. Each network has its own management system, which operates over long time scales.

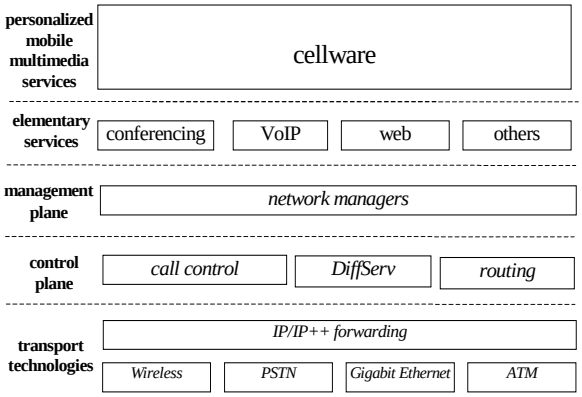

Figure 1: Network Model

Higher level services include the web, voice over IP, and video conferencing. These services deliver multimedia content, but they are neither location-independent nor personalized. Intelligence is added into the network via 'Cellware', an additional, middleware layer of cells. Cells provide value-added support for personalized mobile multimedia services without violating the integrity of the network infrastructure. Cells take advantage of the underlying network mechanisms to deliver audio, video, text or short messages, where and when needed.

2.2 Cells

As illustrated in Figure 2, a 'cell' is the fundamental building block for a personalized mobile multimedia service. Cellware consists of multiple cells. Cells are software components that implement a 'request-event-action' pattern [11]. Cells receive requests to support event-driven distributed multimedia services. Services are described using profiling scripts. Profiling scripts customize the operations of cells in order to support personalized content delivery. Cells respond to events that occur in the physical world and take actions when needed.

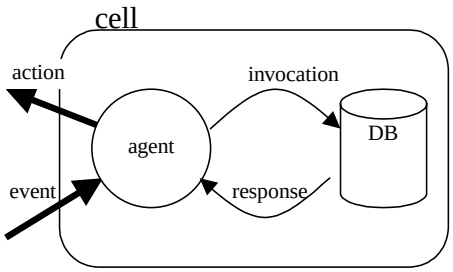

Figure 2: A Cell

Cells comprise an agent and a database. The agent receives events in the form of messages. Messages carry information about the location and type of the event, the time when the event occurs, and the entity that announces the event. Events are announced by other cells, network administrators, or network devices (e.g., microelectronic mechanical sensors). An agent invokes a query method on a database where service conditions are maintained. A search is made on the various service conditions stored in the database. If conditions, associated with the event are met then the database replies back providing the agent with an execution script. The agent executes the script, initiating a number of 'actions'. Actions can be other events, scripts that customize neighboring cells, or invocations on the transport network to deliver multimedia content.

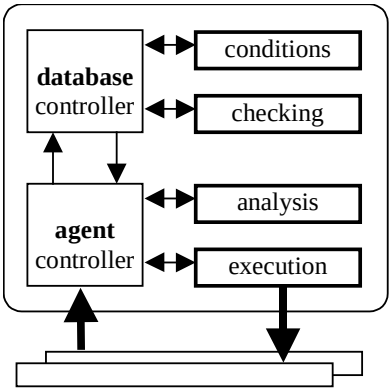

Figure 3: Functions of a Cell

Figure 3 illustrates the functions that are placed inside a cell. A database can be customized using scripts for 'checking' and 'conditions'. 'Checking' scripts specify the way the database should be searched. 'Condition'

scripts make decisions whether the cell should react to events that occur in the physical world. An agent can be customized using scripts for 'analysis' and 'execution'. By 'analysis' we mean instructions that specify where and how service-specific scripts should be executed. By 'execution' we mean actions that realize a personalized mobile multimedia service. A database controller module handles the 'condition' and 'checking' scripts. An agent controller handles the 'analysis' and 'execution' scripts.

A cell is a 'smart' middleware component that supports event-driven and personalized mobile multimedia applications. Multiple cells interact with each other to dynamically combine and form personal services.

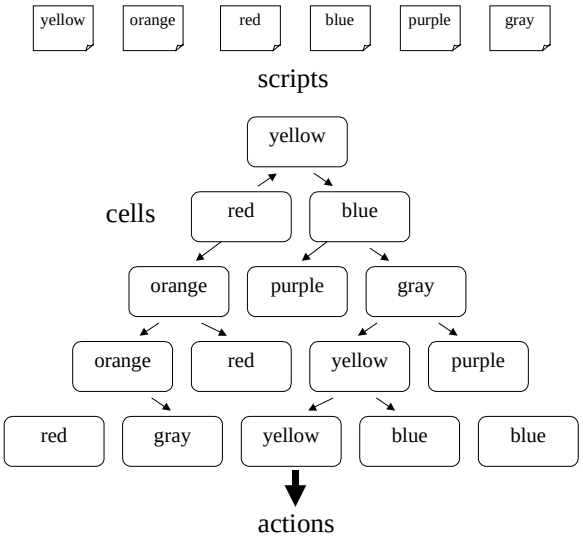

Figure 4: Service Creation Using Scripts

2.3 Service Creation

Scripts can customize cells of an environment to deliver client-specific services. Service creation using scripts is shown in Figure 4. Each color in the figure represents a different set of scripts for 'analysis', 'conditions', 'checking' and 'execution'. This separation is an important aspect of our methodology. Once a cell is 'colored' it can support a client-defined behavior. Colored cells interact with each other to 'reason' about events that occur in the physical world. In what follows we discuss three examples of service creation.

The first example concerns service mobility: Alice is printing a document at work. At some point Alice has to go to her manager's office. Alice wears an active badge [1][2][10], which allows sensing devices to track her movement in the building. Sensing devices trigger events on a cell, which controls Alice's personalized printing service. Upon receiving events, the cell reasons about the movement of Alice. In the end, the cell forwards the remaining pages of Alice's document to the printer located at her manager's office. This is a simple example of the interaction of service and location.

In this example the client who receives the service and the individual whose mobility is monitored are the same person (Other service mobility scenarios can be found in the literature [3]). This is not the case in the second example, where a different style of events makes cells react to changes in the physical world: Alice and Michael have a meeting in the conference room. Alice comes on time but Michael is late. At some point, Alice decides to go back to her office until Michael arrives. As soon as Alice leaves the conference room, the cell that manages Alice's appointments configures a sensing device to react to Michael's eventual presence in the conference room. When Michael arrives in the conference room, the sensing device triggers an event on the cell. Subsequently, the cell sends e-mail to Alice notifying her of Michael's arrival.

The first two examples involve a small number of cells. A more complex service is described below, where the environment dynamically adapts to multiple client preferences. Alice moves from her office in New York to a new office in San Francisco. Alice wishes to have the same working environment in San Francisco as she had in New York. In New York Alice receives a number of personalized services such as online information about the stock market, call transfer when she is out of her office, and the printing service discussed earlier. When Alice enters her new office, a sensing device detects her presence. The sensing device notifies a cell that manages personalized environments for employees. The cell identifies Alice as a new employee from New York. Subsequently, the San Francisco cell contacts a New York cell querying for scripts that describe Alice's personalized environment. New cells are spawned in San Francisco and are configured to support the personalized services Alice has been receiving in New York. In this way the San Francisco environment seamlessly adapts to Alice's preferences without her noticing the change.

3. Cellware Implementation

We have implemented Cellware using XML and Java. XML is suitable for representing information structures and has been used for describing the physical world events and the conditions for reacting to events. Figure 5 illustrates our Cellware implementation. Java has been used for implementing the agent and database controllers of a cell, as well as customized programs for 'checking', 'analysis' and 'execution' (as illustrated in

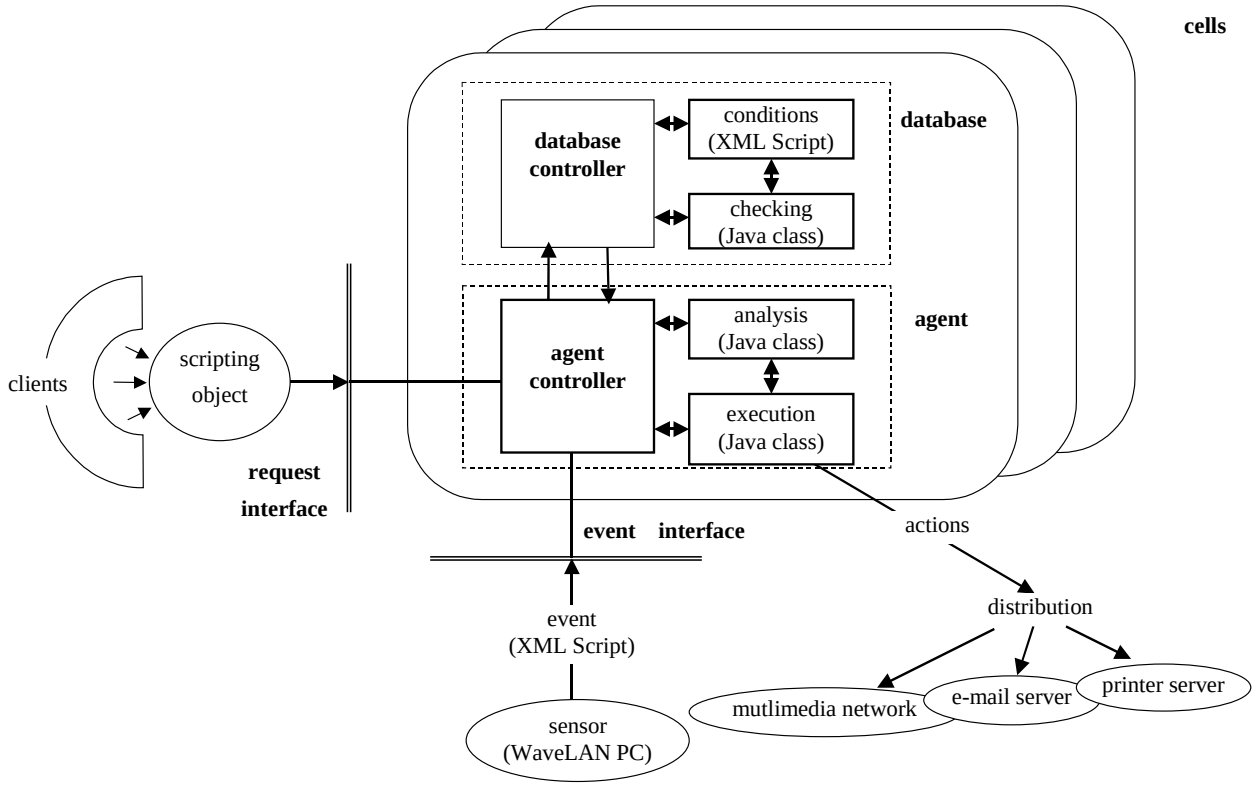

Figure 5: Cellware Implementation

Figure 5).

Initially, a cell comprises only the agent and database controllers. The agent controller supports a request interface and an event interface. The request interface is used for instantiating new services within a cell. A service is instantiated through the uploading of Java classes that implement 'checking', 'analysis' and 'execution' modules. These modules are specific to the service that is being instantiated. In addition, the agent controller receives a script that describes conditions for reacting to events of the physical world. Conditions are specified as a sequence of XML tags. Conditions are stored in the cell's database via the database controller. To keep the cell implementation simple, a cell supports only a single service at a time.

Figure 6 illustrates the messages, which are exchanged between cell components when an event is received. A cell receives an event from the environment (e.g., a sensor) through the event interface (as indicated by e-1, in Figure 6). Events are described as sequences of XML tags. Subsequently, the checking module compares the event against the conditions stored in the database (e-2). If conditions that match with the event are found then these conditions are selected and passed back to the agent controller. Otherwise the cell does not react to the event.

The agent controller reports the event and its associated conditions to the analysis module (e-3). The analysis module decides which actions should be invoked and their order of execution (e-4). Finally the analysis module interacts with the execution module (e-5) to distribute method invocations to other elements of the environment (e.g., a multimedia transport network, printer servers, e-mail servers etc.). Cellware separates the event decision process from the event execution process. In this case Cellware offers a programming environment that can be easily customized to meet client needs.

We have implemented the simple printing and appointment services described in Section 2. Our experimental environment is illustrated in Figure 7. The Cellware testbed delivers the printing and appointment services over a picocellular mobile-computing environment. Our testbed comprises Sun Ultra-Sparc workstations (sirius and orion, as illustrated in the figure), multi-homed 300 MHz Pentium PCs (accordion, harp, cymbal and voice) and HP Laser-Jet printers (rainbow and radio). PCs, workstations and printers are connected to a 100 Mbps Ethernet LAN. The Cellware testbed connects the room 8LE5 at the CEPSR building

at Columbia University, where the COMET group is located with room 810.

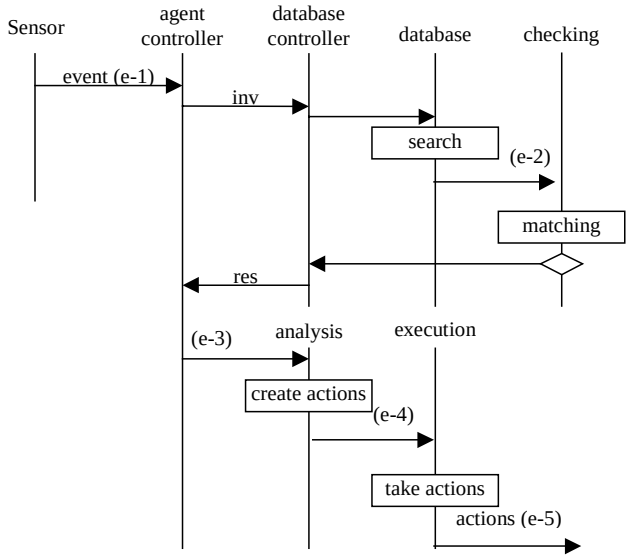

Figure 6: Event Processing

Our radios are based on WaveLAN operating in the 2.4-2.8 GHz ISM band. We use 2 Mbps WaveLAN cards, for which we have a low-level radio utility API for programming beacons, and getting signal-strength measurements. Toshiba Libretto hand-helds equipped with WaveLAN PCMCIA cards have been used as active badges. Multi-homed PCs (cymbal and voice) equipped with WaveLAN ISA have been used as sensing devices. In what follows, we provide an overview of the simple printing and appointment services.

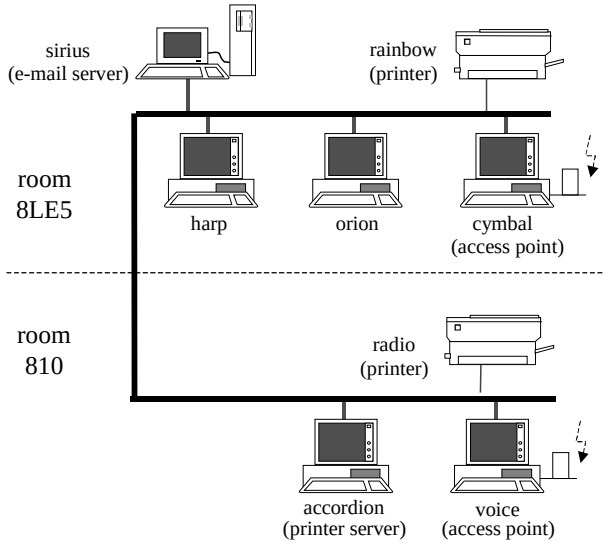

Figure 7: Cellware Testbed

Clearly many new services can be considered and we only present a small but interesting set in this work. The printing and appointment services use the same environment for service creation and this indicates that our approach is more generally applicable to the introduction of new personalized mobile multimedia services.

4. Simple Printing Service

4.1 Service model

This service simply redirects a printing task to the printer that is closest to the location of a client. We have built a printer server in Java that can send a document to different printers on-demand even on a page-by-page basis if necessary. Figure 8 illustrates a model for the printing service. In this model, two types of cells control the printing service: a 'location manager' and a 'printing manager'. The location manager determines the position of a client inside the area controlled by Cellware. The printing manager maintains a list of client preferences expressed as location-printer pairs. The printing manager contacts the printer server to redirect an ongoing task when it is requested.

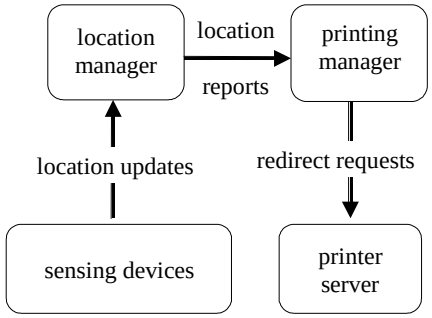

Figure 8: Printing Service

A printing task is initiated by a client using a 'scripting object' (a Java applet) as illustrated in Figure 5. The scripting object raises an event to the printing manager announcing the new task. When a new printing task is announced, the printing manager requests from the location manager to report client location updates. Sensing devices (i.e., wireless access points) measure the signal strength of beacons transmitted by the client's hand-held device. These measurements are collected and compared by Cellware. In this manner Cellware fully determines the position of a client inside the working environment. A printing task is redirected to a new printer when the printing manager receives an event announcing a new client location. The printing manager compares this information with each client preference and decides whether the printing task should be

redirected. Finally the printing manager contacts the printer server to set the destination printer accordingly.

```
<?xml version=”1.0”?>
<database>
<condition>
  <type>owner</type>
  <location></location>
  <time></time>
  <data>
    <name>kazu</name>
    <ip>128.59.69.15</ip>
  </data>
</condition>
<condition>
  <type>task</type>
  <location>radio</location>
  <time>16:11</time>
  <data>
    <status>ongoing</status>
    <file>”http://comet/~kazu/cellware.ps”</file>
  </data>
</condition>
<condition>
  <type>redirect</type>
  <location>room 8LE5</location>
  <time></time>
  <data>
    <dest>rainbow</dest>
    <user>kazu</user>
  </data>
</condition>
<condition>
  <type>redirect</type>
  <location>default</location>
  <time></time>
  <data>
    <dest>radio</dest>
    <user>kazu</user>
  </data>
</condition>
</database>

<?xml version="1.0"?>
<event>
  <type>report</type>
    <location>room 8LE5</location>
    <time>16:12</time>
    <data>
      <user>kazu</user>
      <ip>128.59.69.15</ip>
    </data>
</event>
```

Figure 9: Printing Manager Conditions and Event

4.2 Printing Manager

The printing manager controls the output of a printing task. Figure 9 shows a conditions script stored in the database of a printing manager. In the example shown in Figure 9, the printing manager serves a user named ‘kazu’ who is printing a document located at the URL ‘http://comet/~kazu/cellware.ps’. The ‘owner’ condition specifies the client whom the printing manager serves. In our current testbed a client is specified by a name string and an IP address. The IP address identifies the wireless interface of the hand-held device, which the client carries inside the Cellware testbed. The ‘task’ condition contains information about the current printing task, which the printing manager controls. The ‘location’ tag declares the name of the printer associated with the task. The ‘time’ tag specifies the instance when the printing task started. The ‘status’ tag specifies whether the task is ongoing or has terminated. Finally, the ‘file’ tag declares the location of the file, which is being printed. The ‘task’ condition is added into the database when the client initiates the printing task. ‘Redirect’ conditions express client preferences as location-printer pairs.

An example of an event received by the printing manager is shown in Figure 9: At some point the printing manager receives an event from the Cellware testbed announcing that the user ‘kazu’ is inside the ‘room 8LE5’ at ‘16:12’. The printing manager compares this information with each one of the ‘redirect’ conditions (also shown in Figure 9). The printing manager determines that the printing task of the client should be directed to the printer named ‘rainbow’, which is located at ‘room 8LE5’. Finally, the printer server sets the destination printer to ‘rainbow’.

The checking module of the printing manager searches the cell’s database in a way that is independent of the action the cell executes (i.e., redirecting a printing task). In fact, the same checking module can be used for programming a variety of personalized services. These services have the same checking module but different analysis and execution modules. The separation of the event decision process from event execution process enables the development of flexible and re-usable service components. We have tested the printing service for documents of various sizes and for different mobility patterns in our working environment as proof of concept.

5. Appointment service

5.1 Service Model

Like the printing service, the appointment service takes advantage of the capability of the Cellware testbed to monitor the mobility of individuals. A model characterizing the appointment service is illustrated in Figure 10. Separate cells (i.e., location managers) track the movement of the clients that participate in the appointment service. An ‘appointment manager’ cell controls the service.

A client announces a new appointment using a scripting object. The scripting object raises an event to the appointment manager announcing the new task. An

appointment manager is configured with information about the meeting place, time and participants.

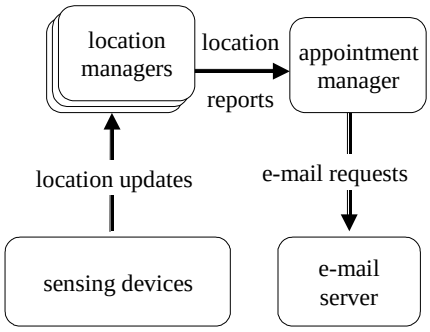

Figure 10: Appointment Service

```
<?xml version="1.0"?>
<database>
  <condition>
    <type>meetinginfo</type>
    <location>room 8LE5</location>
    <time>18:00</time>
    <data>
      <participants>
        <participant>
          <name>kazu</name>
          <ip>128.59.69.15</ip>
        </participant>
        <participant>
          <name>mk</name>
          <ip>128.59.69.17</ip>
        </participant>
      </participants>
    </data>
  </condition>
  <condition>
    <type>participantinfo</type>
    <location>room 8LE3</location>
    <time>17:52</time>
    <data>
      <name>kazu</name>
      <ip>128.59.69.15</ip>
      <state>notarrived</state>
      <email>kazu@comet.columbia.edu</email>
    </data>
  </condition>
  <condition>
    <type>participantinfo</type>
    <location>room 810</location>
    <time>17:55</time>
    <data>
      <name>mk</name>
      <ip>128.59.69.17</ip>
      <state>arrived</state>
      <email>mk@comet.columbia.edu</email>
    </data>
  </condition>
</database>
```

Figure 11: Appointment Manager Conditions

When the new appointment task is announced, the appointment manager requests the Cellware environment to report participant location updates. The appointment manager ‘wakes up’ before the appointment time and sends reminder to all the participants via e-mail. In addition, the appointment manager tracks the movement of every participant inside the Cellware environment. The appointment manager maintains a status flag for each participant, so that it can distinguish between participants who have arrived on time and participants who have not. After the meeting time passes, the appointment manager informs (via e-mail) all participants who are not present at the meeting place. Some of the participants may want to go back to their offices to continue their work until everybody arrived. In this case, the appointment manager sends a reminder via e-mail to all these participants when the last participant arrives at the meeting place.

5.2 Appointment Manager

Figure 11 illustrates the ‘conditions’ script stored in the database of the appointment manager. The ‘meeting info’ condition contains information about the venue, time, and participants of an appointment. The ‘participant info’ condition maintains information abut the name, identification, status, and e-mail address of each participant. Location updates are announced as events to the appointment manager. When receiving such events, the appointment manager updates that status and location of each participant inside the ‘conditions’ database. In this manner the appointment manager can seamlessly enhance a collaborative environment with new capabilities.

6. Conclusion

In this paper we presented the cellware architecture and showed that our framework is flexible enough to support various styles of personalized multimedia services. A number of simple services have been prototyped as proof of concept. As future work we plan to investigate security issues, as well as scripting languages that reduce the complexity associated with profiling and customizing personalized multimedia services.

References

[1] K. R. Wood, T.Richardson, F.Bennet, A.Harter, A.Hopper, “Global Teleporting with Java: Toward Ubiquitous Personalized Computing”, *Computer Magazine*, Volume 30, Number 2, IEEE. February 1997.

[2] G. Appenzeller, K.Lai, P.Maniatis, M.Roussopoulos, E.Swierk, X.Zhao, M.Baker, "The Mobile People Architecture." *Technical Report CSL-TR-99-777, Stanford University,*

January 1999

[3] H .J. Wang, B. Raman, C. Chuah, R. Biswas, R. Gummadi, B. Hohlt, X. Hong, E. Kiciman, Z. Mao, J. S. Shih, L. Subramanian, B. Y. Zhao, A. D. Joseph, R. H. Katz, "ICEBERG: An Internet-core Network Architecture for Integrated Communications", *IEEE Personal Communications Magazine*, 2000

[4] T. Hodes, R. H. Katz, "Enabling . Smart Spaces: Entity Description and Client Interface Generation for a Heterogeneous Component based Distributed System", *DARPA/NIST Smart Spaces Workshop*, Gaithersburg, Maryland, July 1998.

[5] T. Hodes, R. H. Katz, "Composable Ad hoc Location-based Services for Heterogeneous Mobile Clients", *ACM Wireless Networks Journal, Special issue on mobile computing: Selected papers from MobiCom '97* Vol.5, No.5, October 1999.

[6] C. Perkins, "IP Mobility Support", *rfc2002.txt*, IETF, 1996

[7] M. Handley, V. Jacobson, SDP: Session Description Protocol, *RFC2327.txt*, IETF 1998

[8] M. Handley, C. Perkins, E. Whelan, SAP: Session Announcement Protocol, *work in progress, draft-ietf-mmusic-sap-v2-06.txt*, IETF, 2000

[9] M. Handley, H. Schulzrinne, E. Schooler, J. Rosenberg, SIP: Session Initiation Protocol, *RFC2543.txt*, IETF, 1999

[10] M. Weiser, "The Computer for the 21st Century", *Scientific Am erican*, Vol 256. No 3, September 1991.

[11] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, "Design Patterns, Elements of Reusable, Object-Oriented Software", *Addison-Wesley Professional Computing Series*, 1994.

[12] A. T. Campbell, M. E. Kounavis, and R. R.-F. Liao, "Programmable Mobile Networks", *Computer Networks*, Vol. 31, No. 7, pg. 741-765, April 1999.