A Service Oriented SIP Infrastructure for Adaptive and Context-Aware Wireless Services

Wei Li
Department of Computer and Systems Sciences
Royal Institute of Technology
liwei@dsv.su.se

Abstract

Due to the variety of widespread network communication technologies, a user has more possibilities to access various services. However, having alternative networks and services does not bring ease to the user immediately, but often results in increased burdens in terms of repetitive configuration and selection work, although the user is only interested in the actual use of the appropriate services. Our project tries to attack this problem, aiming to facilitate the user's (in particular the mobile user's) ability to make use of different services in an affordable way, based on adaptive services which exploit their awareness of the user's context. In this paper, we propose a service-oriented context infrastructure, to simplify the exchange of context among services, thus facilitating users and applications access of services in an efficient way. The Session Initiation Protocol (SIP) and its sibling protocols were adopted for transferring context information (encoded in XML) within our infrastructure.

Keywords: context-aware, service-oriented infrastructure, SIP

1 Introduction

The computer has to this date been with us for more than a half century. During this period, it has experienced a rapid development in many aspects; from increasingly powerful microprocessors to decreasing energy consumption, physical size and price. Affected by the Internet spreading almost all over the world, the computer today is becoming a crucial tool for accessing and exchanging information across the limit of physical boundaries. The promising goal "access information everywhere anytime" has further paved the way for the inhabitation of computers in our daily life by riding the tide of wireless network technologies (such as Wireless LAN, Bluetooth and GPRS), and has also shaped the computer itself into mobile and wearable fashions like laptops, Tablet PCs, personal digital assistants (PDA) and smart phones. As computers become ubiquitous, people find themselves immersed in a world full of computing power.

However, the availability of rich computing resources does not immediately mean usefulness. For example, a nearby resource in a Bluetooth network cannot be accessed directly through a Wireless LAN or GPRS network without a third party standing in between and acting as a proxy or gateway, since inherently they are not compatible networks. On the other hand, a user may in many cases have several personal devices (e.g., a laptop, a PDA and a smart phone) and there may be additional resources like a desktop computer, a projector and a printer available in the surrounding environment, such as a meeting room. It becomes a critical issue how to "tame" so many different devices to work together fluently without burdens of configuration overwhelming the user, especially when the user is new to the environment and has less knowledge about computers and other devices.

A service-oriented perspective can simplify the problem by presenting different resources (including devices) in the form of services. The user will only see a printing service and a projecting service available in the case above, thus hiding other complexities such as devices and networks. This service-oriented perspective also introduces an emerging computing paradigm - Service Oriented Computing, evolved from mobile and distributed computing, and the object-oriented and component-based computing. In this paradigm, the whole system is constructed by services (with levels) as the building blocks, which are autonomous, platform-independent computational elements and also have capabilities to communicate with each other across networks. Service-oriented computing brings more concrete hints on how to reuse previous work than component-oriented computing since a service normally embodies more complete application logic and often appears as a runtime entity. Along with the service-oriented computing paradigm, many efforts such as Web Services [W3C 2003], GRID computing [Ian 1998], as well as peer-to-peer network technologies work together to envision a future computing environment where computing resources in terms of services will be available to be used as conveniently as today's electricity.

Nevertheless, the term "service" has been used too extensively, and now suffers from a less concrete meaning. Today, we call many things services: network service, printing service, … to discovery service. As a consequence, it is difficult for the user to grasp the relationships between them. For instance, a printing service will not be available without a network service and the ability to discover the printer. Without a good arrangement and guidance for this multitude of services, users can easily get overwhelmed by the great number of services even though they are not interested in the distinctions and implications among them, but rather the immediate use of the interesting services.

Our work in Adaptive and Context Aware Services (ACAS) project tries to attack these problems, aiming to facilitate the user's (in particular mobile user's) ability to make use of different

services in an affordable way based on adaptive services – which exploit their awareness of the user's context. In this paper, we will first give an introduction to context-aware computing and why it is meaningful for mobile users, together with some problems of using context information. Then in the next section, we present our context stack (a layered structure) to help the understanding of context information acquisition. In section 4, we will talk about the mechanism of aggregating and distributing the acquired context information. We will also elaborate how we adopt the Session Initiation Protocol (SIP) and its sibling protocols to build our context delivery infrastructure conforming to a service-oriented architecture. Then in section 5, we will present our prototype application scenario, which exploits the benefits from our context delivery infrastructure. Finally, we will refer to some related work, and discuss some relevant issues and future work.

2 Context-Awareness Computing

Context is a longstanding concept in the human-computer interaction area where it is thought implicit (e.g., a quiet corridor), as opposed to a user's explicit interaction with a computer, which conventionally consists of actions like moving a mouse or pushing a button. The idea behind context-aware computing is to support computer human-literate by feeding in various background information (so-called context) so that the computer can adapt to the user and her situation accordingly. For instance, a mobile phone with location awareness would switch to vibrating mode automatically when its carrier enters a concert. The main purpose of using context is to facilitate user's interactions with computers by replacing some explicit interactions with automatic system interactions, thus reducing attention distraction of the user. This kind of adaptability has great significance for mobile users who are usually willing to pay less attention to the management of a changing environment.

Context-aware computing has been a vital research area, although there are no agreements on the definition of context yet. Many researchers have proposed different interpretations, among which we prefer the most comprehensive one – "context is everything but the explicit input and output" to a system [Henry et al. 2002]. Fortunately, the notion of context has been commonly understood as the physical and social situation where the interaction takes place. It is often associated with primary questions like "When", "Where", "Who" and "What", or in other words: time, location, entity and activity. This also appears to include the most useful context information. In addition to these, we think computing status is also one type of important context, such as the running program and network status (there are actually measured by local processes).

Although there have been many context-aware systems and applications tested since last decade, most of them are still prototypes and only available in research labs or academic world. The common problems with context-aware system lie in the complexity of capturing, representing and processing the contextual data [Pascoe et al. 1997], because the adaptability of a context-aware system will only be appreciated if the context can be interpreted correctly. However, computer technology to date can only try to approach user's intention by "guessing" based on predefined rules applied to different context data and historical behaviors. Another critical issue, which we believe has affected the spreading of context aware systems, is the difficulty of distributing acquired context information broadly among massive context-aware applications across different networks. In a later section, we first present our context stack to help to solve the problem of context acquisition and processing, and after that, we elaborate our context distribution architecture based on SIP Presence framework [IETF 2003].

3 Context Information Acquisition

Sensor technologies have been widely used to capture context information in the context-aware computing arena. The most well-known ones include the early IR-based Active Badge [Want et al. 1992], RF tags in DUMMBO [Salber et al. 1999], the later GPS receiver in Cyberguide [Long et al. 1996], digital camera in StartleCam [Healey et al. 1998], and today's compound sensors: iButton [Dallas Semiconductor] can store any information (with a limited capacity) as context information, such as Identity; and the Mica Motes [Crossbow 2003] (a more advanced sensor board developed by UC Berkeley) can accommodate many different sensors on one board including thermometer, accelerometer, magnetic and photo sensors. The Mote also has the capability to communicate with other sensor boards to create a dynamic wireless sensor network.

The wealth of sensor technologies brings more possibilities for context-aware systems. On the other hand, it makes context information acquisition more complicated, especially when the same context information (in different forms) can be retrieved from different sensor data-sources. Just as most communication means based on radio signals transmitting (like GSM, Wireless LAN, Bluetooth), can be used for positioning and location information, the same sensor data can also be used to give different context information. For example, a GSM phone can simultaneously give presence, location and identity context, a camera can detect motion to tell the presence of people in the room, and can be used to identify users (with facial recognition technology) as well. In such cases, it becomes more puzzling for the system to retrieve the useful and correct context information because adding sensors also adds noise and introduces the possibility of apparent contradictions.

3.1 Logic and physical sensors

There have been many different categories for context information in the field to simplify the understanding and the use of context from different perspectives, such as active vs. passive context [G. Chen et al. 2000] according to whether an application will react to the context changes or not. One of the most accepted divisions of context is to distinguish context as sensed context (retrieved directly from sensors) or derived context which cannot be acquired directly from sensors, but is obtained through inference (also called context fusion, e.g., an activity context may be inferred from a fusion of context information based on people, time and location). The divisions are sometimes also called physical and logical sensors respectively. As one of the results of our survey in the context-aware computing area, we strongly feel the absence of a standard view to support discussion and analysis of context information thoroughly. This view should be more comprehensive than the two-category division into physical and logic sensors.

3.2 The Context Stack

We propose a layered Context Stack like figure 1, along with a service-oriented perspective to promote the understanding of context information acquisition. This view was inspired by the Open System Interconnect (OSI) seven-layer network model and the work of the Location Stack [Jeffrey et al. 2002] as well. We

also borrow some terms from the Location Stack, but recast with our definition to support broader context information.

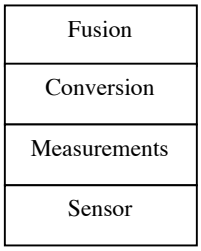

Figure 1. The Context Stack

- **Sensor layer:** Contains the physical hardware (sensor) and its corresponding driver (in some cases some embedded algorithms/firmwares as well). The sensor layer offers the upper layer access to "raw" sensor data, such as the NMEA sentences from a GPS receiver.

- **Measurement Layer:** Contains algorithms to interpret raw data into "canonical" data types, which is more meaningful and practical to general applications. Taking the GPS receiver for example, the Latitude and Longitude should be deduced in this layer by parsing the raw NMEA sentences.

  As many researchers also pointed out that *history behaviors* should also be included as context in some cases, we think it is appropriate to have this kind of context in this layer, such as a data set to plot a curve of temperature changes in a specific time-span.

- **Conversion Layer:** The measurement layer normally offers sensor data in limited (canonical) forms. We think it is very important to have an abstract layer taking responsibility of converting the context data into different formats/units. For example, a software service can convert temperature from Kelvin to Celsius or Fahrenheit. The conversion layer can have a chain of services to support conversion between many different units. How to determine the optimal path from several possible service-conversion chains becomes an interesting issue per se.

- **Fusion Layer:** Contains a set of very abstract context information, such as location, time and activity. Each of them may be inferred from a group of sensor data but represents the same context information to different extents (maybe in different confidence, accuracy or granularity). For instance, the location fusion in figure 2 can be inferred from different data-sources with different characteristics, where GPS will only be available outdoors with an accuracy of 10-20 meters, and GSM gives less accuracy, roughly around 100 meters. The inference engine (also called context refiner) can also be backed with self-refinement [Anind et al. 2000] or even self-learning capability with the support of machine learning technologies.

Both the Conversion layer and Fusion layer can be further used repetitively to deduce higher-level context information. Like in figure 3, a context-aware application is interested in activity and people, wherein activity is a context information derived from a fusion based on time, people and location, wherein the location context information is derived from a fusion as well. We can impose a service-oriented view on this context structure by treating each supplier of the context information as a single context service, so every context service will only need to know other context services in which it is interested, without bothering to grasp the entire structure of services. Undoubtedly, some intermediary conversion services have to be available when one service cannot understand other required services. Considering when discovery is enabled among these context services, the context structure could be constructed on demand dynamically triggered by the user's request.

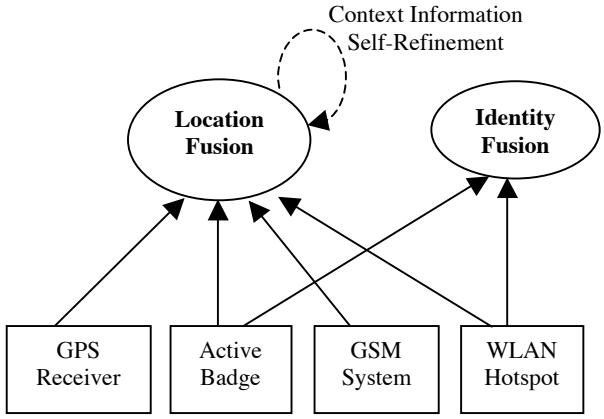

Figure 2. Location Context Fusion

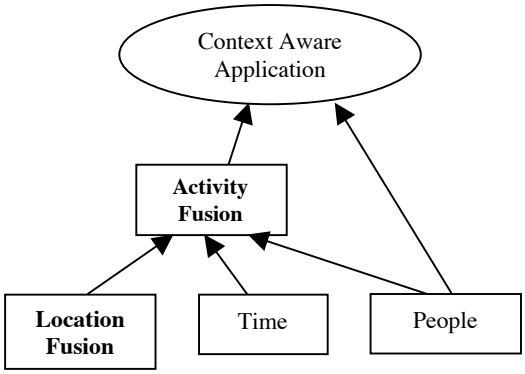

Figure 3. Context Service Tree

3.3 Computational Context

We believe computing status is also an important type of context (we call it *computational context*). It includes the running program and network status etc. Computational context helps to improve the deduction of the user's status and activities with more accuracy. For example, there may be a presentation taking place because user's computer is running Microsoft PowerPoint in a meeting room. And as an adaptation, all the mobile phones in the room turn to mute/vibrate mode automatically. Another preferable automation can be that the system assists users to fluently switch to a better interaction mode by transferring the communication session from PDA to desktop PC after arriving in the office. The examples require the system to have the capability of detecting the running programs status, e.g., to learn and remember the browsing web page in a running web browser application. However, computational context has not been fully used mainly because of the difficulty of acquiring the information of running processes, which normally can only be acquired from application program interfaces (APIs) coupled closely with the operating system.

To solve the problem of retrieving computational context information, we suggest a database solution. The running processes use database to store and retrieve computing status. Such a solution requires the processes to have extra knowledge about the database interface and data schema for accessing computational context. However, comparing with the complexity of massive low-level system APIs, it gives a uniform structure to simplify the computational context exchange. Practically, each device can have a computational context agent monitoring running processes, and report process changes to the context database. Then on the other side, another context agent observing the context database for changes will present itself as a context service to fit into our context stack (on the fusion layer). Connectivity is a good example of computational context which is actually measured by local process. It is very significant for many applications to determine the optimal communication means.

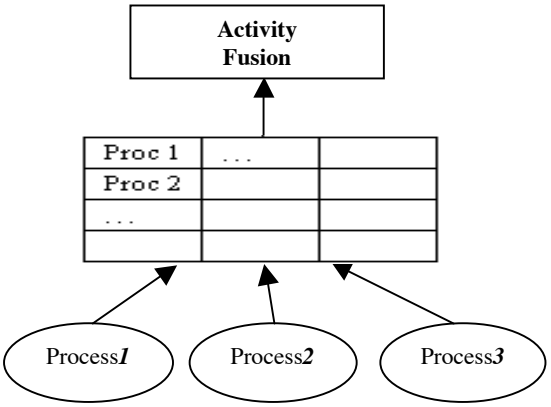

Figure 4. Computational Context Based on Database

3.4 Context Representation (Meta-data)

It is also important to show the semantics of the context acquired by using context meta-data, which is useful to identify facts such as how and from where the context is acquired. For better interoperability, XML [W3C 2000] was chosen as the encoding format due to the fact that XML has become a de facto data standard. So far, we represent context information in an atomic fashion like the example in Table 1, which can be read as an acquisition of a set of context, consisting of the temperature from wireless sensor board MICA2 and the location from an RF tag. Such context information may come from the output of *context refiners* producing refined context information through the processing of acquired context (e.g., MICA Mote does not report temperature in any standard unit directly) taking place locally or remotely. However, we decided to make our context representation neutral to the means they are acquired. We did define a *source* element which currently has only one attribute *uri* exposing the sensor entity (or device) to give the clue how this context was acquired. And we left room in the content of source element for further extension (so far it is simply the entity ID). To identify our context information and be able to co-exist with other workers, we also defined an XML namespace. We have successfully applied this context schema with SIP based Presence framework in our context information distribution infrastructure.

Table 1. XML-encoded Context Example

```
<acas:node xmlns:acas="urn:acas:">
  <acas:contextelement>
    <acas:value datatype="integer"
      unit="temperature/kelvin">300</acas:value>
    <acas:time>Fri Sep 26 16:42:25 CEST 2003</acas:time>
    <acas:source
      uri="http://MICA2_001.fuse.k2lab.dsv.su.se">
      MICA2_001 </acas:source>
  </acas:contextelement>
  <acas:contextelement>
    <acas:value datatype="string" unit="location/floor">
      Forum Floor 7 </acas:value>
    <acas:time>2003-9-17 12:00 am</acas:time>
    <acas:source
      uri="http://RFID001.fuse.k2lab.dsv.su.se">
      RFID001</acas:source>
  </acas:contextelement>
</acas:node>
```

4 Context Delivery

Most of the accessible context-aware systems are only available in research labs and the academic world. They are constrained within a specific space such as a room and a building, or more precisely by the coverage of specific networks which prevents context information to be shared among different context-aware applications.

Today, with the emerging network technologies of GPRS and 3G, context-aware systems can be deployed in a much wider area. However, considering their high cost and limited bandwidth compared to a conventional local area network (LAN), as well as the popularity of Wireless LAN and the other short-range network technologies such as Bluetooth, the optimal hybrid use of such heterogeneous networks for context delivery becomes an issue of significant value.

The optimal use of networks requires a smooth-switch capability, e.g., context information is delivered through an available Wireless LAN, and then switched to GPRS when moving out of the coverage of the Wireless LAN hotspot. However, the frequent network switch (handover) cannot be smoothly handled by the conventional network technologies like TCP/IP, Help exists from some technologies targeting this purpose, such as MobileIP [IETF 2002] which pushes the responsibility to the network infrastructure. Here, we propose another solution based on the Session Initiation Protocol (SIP) [Rosenberg et al. 2002], an application-layer control (signaling) protocol for creating, modifying, and terminating sessions. This application layer protocol creates an overlay network for communication without modification of existing network infrastructures.

4.1 SIP Protocols

The Session Initiation Protocol (SIP) is a general-purpose communication protocol supporting interactive session establishment across the Internet. It defines a complete process mechanism for establishing a distant communication session, which is independent of the underlying transport protocol and without dependency on the type of session to be established. It is a text-encoded protocol using Uniform Resource Identifiers (URI) [Berners-Lee et al. 1998] as the addressing mechanism, which resembles the normal e-mail address like “sip:user@domain”. SIP works on a client-server transaction model akin to HTTP. A SIP client generates a SIP request to the network, and waits for the response from other SIP servers, in order to establish a communication session. Every SIP message includes enough routing and session status information, so that each single

message can be delivered to its destination accordingly. In addition, SIP supports personal mobility by discovering users and locating devices, as well as the negotiation among session participants with different capabilities to determine an agreed communication session.

SIP was originally designed for IP telephony. However, due to its neutral session establishment characteristics, many other features were added with a series of complementary specifications to improve the functionality for other usages, such as supporting asynchronous information transfer.

The SIP-Specific Event Notification specification [Roach et al. 2002] addresses the issue of asynchronous notification of events through a SUBSCRIBE/NOTIFY mechanism. An application (called a Subscriber in the specification) subscribes to certain events, and will then receive notifications when the corresponding changes occur. This mechanism is especially suitable for distributed context information delivery. And the further issues pertinent to subscription duration, event packages, re-subscribe and un-subscribe etc have already been addressed in detail in the specification.

Meanwhile, SIP for Instant Messaging and Presence Leveraging Extensions (SIMPLE [IETF 2003] - another SIP-related IETF working group) focuses on introducing the Instant Messaging and Presence (IMP) service extensions to SIP network. In SIMPLE Presence Event draft [Rosenberg, J. 2003], two new notions are introduced: the *Presence Agent* (PA), a representative of a SIP Presence entity, and the *Watcher*, a consumer of a PA for status information. A new method *Publish* is proposed in the draft [Campbell et al. 2003] for sending information to any SIP Presence entity without establishing a session beforehand. Then there is also a mechanism of *Presence Aggregation* proposed in a later draft [Roach et al. 2003] to enable the aggregation of a group of presence information into a single presence entity for subscription. This mechanism can easily be adapted to present a context entity (such as a user) with a collection of different context information related.

Due to the fact that SIP and its sibling protocols have many advanced features satisfying the criteria of delivering context information in a distributed heterogeneous network environment, we chose to build our context delivery infrastructure on SIP, it is also because we believe SIP will be commonly accepted as a standard communication protocol in the near future with the 3G technology coming into force.

4.2 Context Information Distribution Infrastructure

Despite the advantages of SIP for context information transfer, the adoption of SIP is not easy due to the complexity of its transaction centric process mechanism for communication, which does not match with the principle of traditional network application programming, thus demanding developers a long study curve to get familiar with the details of the SIP protocols. We decided to bridge this gap by taking a service and component-oriented design principle to define a set of software components taking charge of SIP-based communication such as SUBSCRIBE/NOTIFY mechanisms, hence hiding the complexity of SIP network communication from context service development. With these components, context services can be easily constructed and connected to build a context network infrastructure in a *plug-and-play* style.

As SIP Presence framework is very appropriate to be extended to delivery context information, and two basic SIP entities in the framework are also explored: the Presence Agent and the Watcher, which can be implemented as reusable building blocks to construct context-aware systems. Our design components are shown in figure 5.

- **SIP Stack Wrap:** Takes the responsibility of setting up communication parameters such as network protocol (TCP or UDP) and port number, which any SIP entity has to have. The network changes such as switching IP addresses when roaming between networks will be handled by this layer without the awareness of upper layers.

- **Presence Agent (PA):** Takes charge of handling the subscription of presence and other context information, and also offers a simple interface for context services (such as fusion) to report and update detected context changes. Notifications are organized (coded in XML format) and sent to the registered Watchers automatically. The PA also handles the issue of subscriptions expiring in an autonomic way.

- **Watcher:** The Watcher is a context consumer who first registers to the PA with an indication of interesting context, and then waits for the notifications when the corresponding context updates are noticed by PA. Besides the basic subscribe, Watcher component also gives developers with interfaces for un-subscribe and re-subscribe etc.

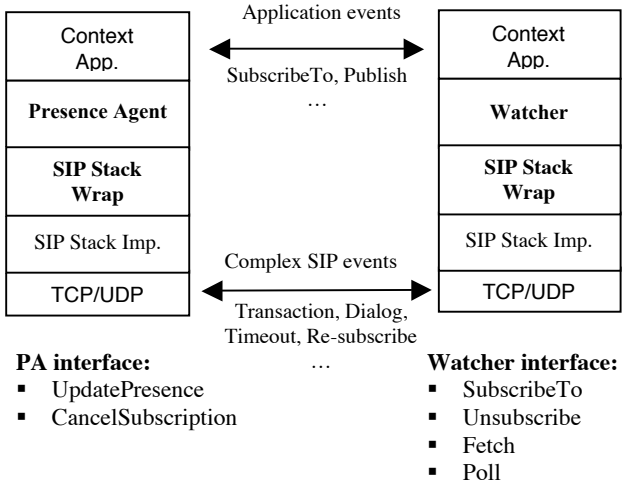

Figure 5. Presence Agent and Watcher

Our SIP components are built on top of NIST-SIP 1.2 [NIST 2003], an open source SIP stack implementation in Java, compliant to the JAIN-SIP-1.1 API standard [JCP 2002]. With the encapsulation of software components, developers can easily build context-aware systems upon SIP network to enable wide-area delivery of context information. In the following section, we will elaborate on an application scenario with which we built our context-aware system prototype based on these software components.

5 Prototype

We defined a general application scenario where a project group needs to keep live contact. We assume that project members are always online, so that they can be aware of each other's status, as

well as the progress of the project including documents and meeting schedules. To achieve this, we constructed an application infrastructure using our SIP components to support mobile ICQ-like applications. Additionally, such infrastructure provides rich context information and consequent implications rather than merely user presence information. This characteristic offers abundant possibilities for further context-aware applications development.

In figure 6, each group member subscribes to a Group Facilitator (GF) for receiving the status information of other members and the project as well. The GF also subscribes to every member's Presence Agent for acquiring information from every group member, and then aggregates the individual status into a group view shared by the whole group. The Presence Agent can be located in a stationary office desktop or in a mobile device such as Sony Ericsson P800 mobile phone.

We are testing to equip our mobile users with P800 mobile phone or iPAQ, Mica Mote and GPS receiver to acquire and report different context information including location, speed, noise-level, temperature and the people in the vicinity, as well as the available services discovered, which all together give rich implication for user and system to determine the optimal communication and interaction means. We have also connected this context infrastructure to our internal SIP telephony network with the first context-aware service – a context-aware call controller, which determines whether the incoming calls will get through, be blocked or redirected to be voice mail, according to user's current state (inferred from context) and the preference.

The infrastructure is built on top of our reusable SIP components. The GF is actually implemented as a combination of a PA and several Watchers, since GF on the one hand receives context information from the PAs (each represents a group member), while on the other hand presents an aggregated group view to subscribers (group members). The client application is based on a Watcher component with a graphical user interface (GUI) showing the status of the project resources and group members. This infrastructure will be connected to our interactive Lab – iLounge, (an interactive room with various sensors) to evaluate the potential influence on mobile users with access to rich context information from the environment. The SIP telephony network is constructed on an open source SIP server implementation – Vocal v1.5.0 [Vovida 2003], acting as a SIP proxy for delivering various SIP messages, such as Voice over IP (VoIP) call request. Upon which we developed a call-control service (also based on Watcher component) that makes call routing decision based on user's status inferred from the presence and context information retrieved through GF, and dynamically generates the corresponding Call Processing Language (CPL) [Lennox 2003] script to control the Vocal server. We have tested caller applications with Microsoft Windows Messenger (v5.0) and some other free SIP softphones successfully, the multimedia calls, including instant message text, audio and video, got across fluently. Since our context distribution is also based on SIP, we can tie our infrastructures seamlessly with any other SIP-based networks. This brings great advantages on system scalability and extensibility.

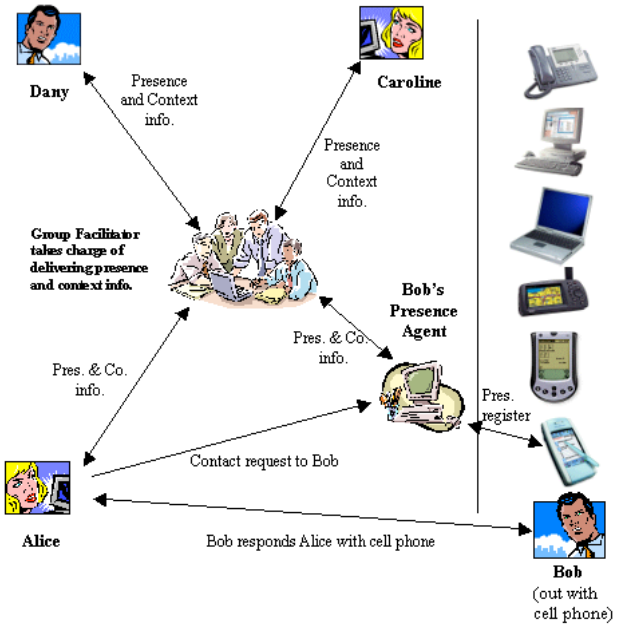

Figure 2. Group Facilitator

6 Related Work

There have been many studies on context acquisition and distribution. Here, we list some most relevant work. Schilit's infrastructure for ubiquitous computing in 1995 is probably the first context framework, which concentrates mostly on location with an active badge infrastructure [William 1995]. Then the Context Toolkit [Anind 2000] in GeorgiaTech defines a distributed architecture supporting context fusion and delivery with three notions of widget, server and interpreter based on an object-oriented architecture. Technology for Enabling Awareness (TEA) project [A. Schmidt 1999] also models the context data in a layered structure from time-stamped raw sensor data layer, Cue layer for data interpretation, to the "Context" layer. The Location Stack [Jeffrey et al. 2002] gives seven layers structure similar to OSI network model for acquisition of mainly location context.

On the other hand, the earlier work of Roychowdhury and Moyer [Arjun et al. 2001] proposed the use of presence information for remote awareness and instant messaging over SIP for remote control. Shim et al presented Spontaneous Enterprise Communications (SEC) [Hyong et al. 2001], a system designed for ad-hoc conferencing within an organization. SEC also relies on SIP for the infrastructure, using presence and availability management, conference control and text messaging.

7 Conclusions and Future Work

In this paper, we have proposed a layered context stack to help to understand the collection, interpretation, aggregation and analysis (fusion) of sensor data to deduce useful high level context information, and to facilitate the communication between researchers working on context. We suggested a service-oriented view on context information processing by presenting each process as a service, thus making these processes reusable and the context information (in different forms/levels) sharable. In addition, with this service-oriented view, the technologies applicable for services, such as service description and service discovery, can be applied to context processing as well, and the context processing may also benefit from other services which are already available in the network.

We have also explored the use of SIP and its sibling protocols for context distribution over networks, due to its advantages on flexible establishment and modification of general communication sessions. We think this advanced feature will simplify the communication between applications (and services), especially in a frequent changing environment, such as delivering the context of a mobile user.

However, there are many issues not addressed in this paper. For example, how to support dynamically adding new sensors into the infrastructure in a flexible style like plug-and-play? How to model and describe context information in a complex ubiquitous computing environment, then how to add context discovery support upon the model? ... So far, we have chosen a very simple prototype scenario, and our current model is fairly preliminary as well. It only consists of two entity types – the user and the group. The relationships between them are also rather static – the GF has the priori knowledge about the address of each group member, and so does every member about the address of GF.

We will look into such issues in detail in our future work with more complex context-aware applications, complying with the service and component-oriented architecture. One practical way is to compose more context-aware software components and small-granularity services by wrapping the existing applications we have in everyday use with context-awareness to promote the development of adaptive services. Furthermore we will also refer to the development in other research areas such as ontology and semantic web for relevant support.

Acknowledgements. This research is led and organized by Wireless@KTH, and partially funded by the Swedish Foundation for Strategic Research within the "Internet & Mobility" program.

References

W3C, Web Services Description Language (WSDL 1.2), http://www.w3.org/TR/wsdl12/

IAN FOSTER, 1998, *The Grid: Blueprint for a New Computing Infrastructure*

HENRY LIEBERMAN,TED SELKER, 2002. Out of Context: Computer Systems That Adapt To, and Learn From, Context, MIT Media Laboratory http://lieber.www.media.mit.edu/people/lieber/Teaching/Context/Out-of-Context-Paper/Out-of-Context.html as of August 7, 2002

PASCOE, J., 1997. The Stick-e Note Architecture: Extending the Interface Beyond the User, International Conference on Intelligent User Interfaces, Orlando, Florida, USA. ACM.

WANT, R., HOPPER, A., FALCAO, V., GIBBONS, J. 1992. The Active Badge Location System, *ACM Transactions on Information Systems 10*(1) pp. 91-102.

SALBER, D., DEY A.K., ABOWD, G.D. 1999. The Context Toolkit: Aiding the Development of Context-Enabled Applications, CHI'99.

LONG, S., ET AL. 1996. Rapid Prototyping of Mobile Context-aware Applications: The Cyberguide Case Study. *2nd ACM International Conference on Mobile Computing and Networking (MobiCom'96)* November 10-12, 1996.

HEALEY, J.; PICARD, R.W. 1998. Startlecam: A Cybernetic Wearable Camera. *2nd. International Symposium on Wearable Computers*, Pittsburgh, Pennsylvania, 19-20 October, 1998, pp.42-49.

Dallas Semiconductor. iButton Home Page. http://www.ibutton.com/.

Crossbow, http://www.xbow.com/Products/Wireless_Sensor_Networks.htm as of June 2003

G. CHEN AND D. KOTZ. 2000. A survey of context-aware mobile computing research. Technical Report TR2000-381, Dartmouth College, Computer Science, Hanover, NH, Nov 2000.

JEFFREY HIGHTOWER, BARRY BRUMITT, AND GAETANO BORRIELLO. 2002. The Location Stack: A Layered Model for Location in Ubiquitous Computing," in *Proceedings of the 4th IEEE Workshop on Mobile Computing Systems & Applications (WMCSA 2002)*, (Callicoon, NY), pp. 22-28, June 2002.

ANIND K. DEY, JEN MANKOFF AND GREGORY D. ABOWD. 2000. Distributed Mediation of Imperfectly Sensed Context in Aware Environments GVU Technical Report GIT-GVU-00-14. September 2000.

W3C. 2000. Extensible Markup Language (XML) 1.0 (Second Edition), http://www.w3.org/TR/2000/REC-xml-20001006, 6 October 2000.

KAREN HENRICKSEN, JADWIGA INDULSKA, ANDRY RAKOTONIRAINY. 2002. Modeling Context Information in Pervasive Computing Systems. 167-180, in *proceedings of Pervasive 2002*: Zurich, Switzerland.

MARTIN JONSSON. 2003. *Supporting Context Awareness in Ubiquitous Service Environments*, Licentiate Thesis, Royal Institute of Technology/Stockholm University.

IETF, 2002. IP Mobility Support for IPv4, RFC 3344, August 2002, http://www.ietf.org/rfc/rfc3344.txt

ROSENBERG, J., SCHULZRINNE, CAMARILLO, JOHNSTON, PETERSON, SPARKS, HANDLEY AND SCHOOLER. 2002. SIP: Session Initiation Protocol, RFC 3261, June 2002.

BERNERS-LEE, T., FIELDING, R. AND L. MASINTER, 1998. Uniform Resource Identifiers (URI): Generic Syntax, RFC 2396, August 1998.

ROACH, A., 2002. Session Initiation Protocol (SIP)-Specific Event Notification, RFC 3265, June 2002.

IETF, 2003. SIP for Instant Messaging and Presence Leveraging Extensions (SIMPLE) http://www.ietf.cnri.reston.va.us/html.charters/simple-charter.html as of June 2003.

ROSENBERG, J. 2003. A Presence Event Package for the Session Initiation Protocol (SIP)", IETF draft-ietf-simple-presence-10 (work in progress), Jan. 2003.

CAMPBELL, B., OLSON, S., PETERSON, J., ROSENBERG, J. AND B. STUCKER. 2003. SIP Presence Publication Mechanism Requirements, IETF draft-ietf-simple-publish-reqs-00 (work in progress), February 2003.

ROACH, A., ROSENBERG, J. ET AL, 2003. A Session Initiation Protocol (SIP) Event Notification Extension for Resource Lists, IETF draft-ietf-simple-event-list-04, (work in progress), June 2003.

NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY (NIST), 2003. http://snad.ncsl.nist.gov/proj/iptel/ as of June 2003.

JAVA COMMUNITY PROCESS (JCP), JAIN SIP API Specification, Aug. 2002.

VOVIDA. 2003 http://www.vovida.org/vocal as of Aug. 2003

WILLIAM NOAH SCHILIT.1995. *A system architecturefor context-aware mobile computing*. PhD thesis,Columbia University, May 1995.

ANIND K. DEY. 2000. *Providing Architectural Support for Building Context-Aware Applications*, PhD thesis, College of Computing, Georgia Institute of Technology, December 2000.

A. SCHMIDT, K.A. AIDOO, A. TAKALUOMA, U. TUOMELA, K. VA N LAERHOVEN, AND W. VAN DE VELDE. 1999. Advanced Interaction in Context. In H. Gellersen (Ed.) *Handheld and Ubiquitous Computing*, *Lecture Notes in Computer Science* No. 1707, ISBN 3-540-66550-1, Springer-Verlag Heidelberg: 1999, p. 89-101.

ARJUN ROYCHOWDHURY AND STAN MOYER. 2001. Instant messaging and presence for sip enabled networked appliances. http://www.iptel.org/2001/pg/final program/22.pdf, April 2001.

HYONG SOP SHIM, CHIT CHUNG, MICHAEL LONG, GARDNER PATTON, AND SIDDHARTA DALAL. 2001. An example of using presence and availability in an enterprise for spontaneous,multiparty, multimedia communications. http://www.iptel.org/2001/pg/final program/13.pdf, April 2001.

LENNOX, WU, AND SCHULZRINNE. 2003. CPL: A Language for User Control of Internet Telephony Services", IETF draft-ietf-iptel-cpl-08.txt (work in progress), Aug. 2003