Inferring Presence in a Context-Aware Instant Messaging System

Mikko Perttunen and Jukka Riekki

Department of Electrical and Information Engineering and Infotech Oulu,
P.O.BOX 4500, 90014 University of Oulu, Finland
{Mikko.Perttunen, Jukka.Riekki}@ee.oulu.fi

Abstract. The increasing volume of digital communication is raising new challenges in the management of the information flow. We discuss the usage of context to infer presence information automatically for instant messaging applications. This results in easy-to-use applications and more reliable presence information. We suggest a new model, context relation, for representing the contexts that are relevant for inferring presence. The key idea is to represent both the communication initiator's and the receiver's contexts. The model allows sophisticated control over presence information. We describe a fully functional prototype utilizing context relations.

1 Introduction

Instant messaging (IM) has proved its usefulness: its popularity is growing fast, and it has been introduced into corporate use as well. Due to the increasing importance of IM as a method of everyday interpersonal communication, the volume of messages and simultaneous messaging sessions will increase. This development may lead to excessive interruptions on the user's other tasks [1]. The problem can be solved by reducing the attention required from the user to manage the messaging.

IM has been described as "near-synchronous computer-based one-on-one communication" [2]. It is said to consist of two components: synchronicity and presence awareness. Synchronicity refers to real-time information transfer [3]. We define *presence* as the ability and willingness of a user to communicate with other users. This meaning has lately been designated as *availability* [4,5], but we prefer to speak about *presence* in a broad sense to cover the user's situation more extensively than just in terms of online status [6,7]. For example, presence information can define the user to be at her desk or to be currently typing a message [3]. Presence-awareness means that presence information is used to decide whether communication with a user can be initiated. This information can be used by the user about to initiate communication, or the IM application can deliver instant messages only to users with appropriate presence status.

In current IM applications, the users must update their presence information manually. For example, when a user enters a meeting room, she must remember to set her presence as *unavailable*. IM applications would be easier to use, if the updating of presence information were automatic. Such updating could be more reliable as well, since the users would not need to remember to manually set their presence. Further, more reliable presence information would eliminate unsuccessful attempts at communication, which unnecessarily interrupt the recipient and frustrate the initiator. Such automatic updating can be achieved by recognizing the user's context and inferring her presence from that. In the above example, a *meeting* context would be recognized, and presence would be automatically changed to *unavailable*.

This article shows how presence information can be updated automatically. Our main contribution is to use the contexts of both the recipient and the initiator in inferring the recipient's presence. In the work reported by others, only the context of the recipient has been used. We formalize the concept as a *context relation*. We describe a fully functional prototype IM system to demonstrate this approach.

We adopted the instant messaging terms presented in RFC 2778 [8]. In the general discussion, we map these terms in a simple IM system consisting of a separate IM application for each user and one server. *Presentity* (i.e. presence entity) and *watcher* are mapped into the IM application's roles. Hence, a presentity is an IM application providing its user's presence information to other IM applications. A watcher, in turn, is an IM application receiving other users' presence information. We postpone the discussion of the implementation of such functionality to the section presenting the prototype. Furthermore, the users are also called presentities and watchers according to their roles at the different phases of instant messaging.

The rest of the paper is organized as follows. In section 2 we list related work. In section 3 we describe context-aware instant messaging and the context relation. Section 4 presents our application prototype. It is followed by a discussion of the findings in section 5.

2 Related Work

In this chapter, we present the related work on IM and context. We adopted Dey's definitions of context and context-awareness [9]:

Context is any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves.

A system is context-aware if it uses context to provide relevant information and/or services to the user, where relevancy depends on the user's task.

Peddemors et al [7] presented a system which (among its other features) allows users to set rules that cause the system to update their presence to a certain value when they are at a certain location. They also state that presence can be interpreted as part of the context of a user. The statement is reasonable, because presence information indicates the user's current situation, sometimes also activity, which fits Dey's definition.

Ranganathan et al [10] developed ConChat, an IM system that automatically gives users cues about the context of their contacts. They also note the usability of context in avoiding semantic conflicts. ConChat, for example, automatically tags the currency of the sender's country when discussing money.

Tang and Begole [11] describe an IM system called Lilsys, which utilizes ambient sound, phone usage, and computer activity to infer the availability of a user. The Awarenex system [12] uses such parameters as location and current calendar events to give cues about a user's presence. Related to their work with the Awarenex prototype, the researchers also developed algorithms to detect and model patterns of availability and unavailability over time. They integrated the work with the Awarenex prototype and demonstrated that the system can predict, for example, the time when a person comes back from lunch [13].

When a user is typing on a computer, she may be performing some task requiring concentration. When she has not typed for a while, almost every instant messaging application updates the user's presence to *unavailable* by monitoring the keyboard and mouse activity. This is sometimes contradictory [11].

Voida et al identified contradictory user requirements in IM. For example, people want IM to be both asynchronous, in order to be able to be involved in many communication threads simultaneously, and synchronous, to get a prompt reply. Furthermore, they studied user expectations about context. They noticed, among other things, that the initiator of communication, despite the availability of the recipient, still wanted to query the recipient's context in the first message [14].

3 Improving Presence Inference with Context Relations

In IM, a message can be sent when the recipient is logged in to the system and her presence is *available*. This is interpreted by the sender as willingness of the recipient to receive messages and to respond to them in a timely manner. To avoid disturbance, a user can manually select to be *unavailable*. Because presence gives such an implication to the sender, it is important that the presence information provides maximally reliable information of the recipient's situation. The correctness of presence reduces unsuccessful communication attempts.

To be easy to use, the application should be able to update the presence of the user without the user's intervention. This also increases the reliability of the presence information, providing that the user might sometimes forget to update it. Presence updating should be based on automatic context recognition and function according to rules accepted by the user.

Automatic updating of presence can be achieved in two ways. First, only the context of the presentity can be used in inferring her presence to the watchers. In this case, the watcher's context has an effect only if it is part of the context of the presentity. The second way of inferring presence will be discussed in more detail later in this chapter.

The first way of inferring presence is illustrated in Figure 1. User B is the presentity and User A is the watcher. User A has subscribed User B's presence information. User B's context change triggers presence inference, and User A is notified of the new presence of User B. Inferring is done using User B's context-based presence rules.

Figure 2 shows how presence is inferred from user preferences and context information. Boxes denote data and circles functions. User preferences define the user's important locations, e.g. home and office. The context information used in the inference should be designed, as for any context-aware application, to best suit the needs in modeling the user's situations. Here, we use activity and place as examples in our illustrations. For example, the user might select a rule "When my activity is *conference*, I'm *unavailable*".

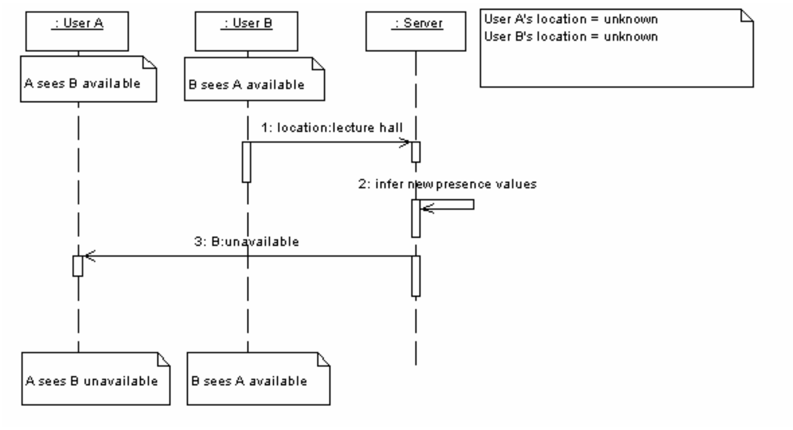


Fig. 1. Context-based inference of presence

Generally, the presence function in Figure 2 takes context as input and yields a presence value. Because both activity and place are derived from location, the rules may give conflicting values for presence. For example, a meeting could be scheduled to take place in the user's own office, in which case the activity could be *meeting* and the place *office*. This situation can be handled by defining activity as the higher-priority context. Thus, only when the user's activity is unknown is the place used to derive presence.

The above way of automatically updating presence easily leads to the following situation. User A and User B are both in the same lecture hall in a conference and find each other to be *unavailable*. However, for two friends, it would be natural be to be able to communicate and to exchange comments about the presentations. The desired scenario is illustrated in Figures 3 and 4: when User A enters the same lecture hall as User B, they are able to communicate. Such functionality can be achieved with the second way of inferring presence, context relation-based inference, which is discussed next.

In Figures 3 and 4, User A has User B on her contact list. User A has thus subscribed the presence changes of User B. Similarly, User B has User A on her contact list. At the beginning of the scenario illustrated in Figure 3, User B's context (location) changes, which causes the system to infer a new presence value for User B and to deliver it to the watcher, i.e. User A. This is straightforward for any context-aware IM system. The novelty of context relations suggested in this article is that the watcher's context has an effect on the presence of the presentity. Hence, a new presence value is inferred for User A as a reaction to the new context of User B. Figure 4 shows similar actions taken when User A's context changes. Consideration of the watcher's context enables the presence of User B to change when the context of User

A changes. At the end of the scenario shown in Figure 4, the users A and B see each other as *available* and are thus able to communicate.

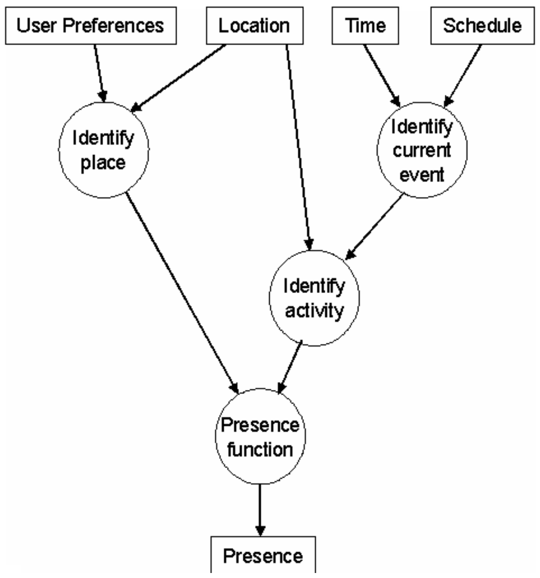


Fig. 2. Inferring presence from context and user preferences

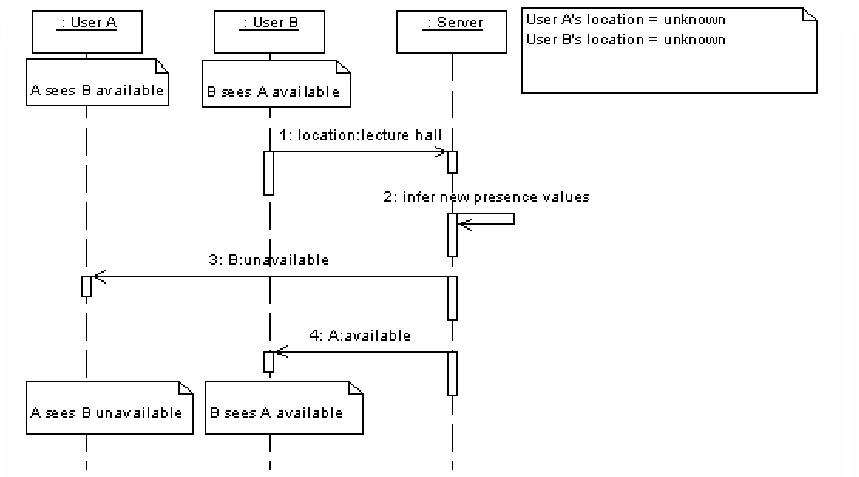


Fig. 3. Context relation-based inference of presence: User B goes into a lecture hall. User A sees User B as *unavailable* and User B sees User A as *available*

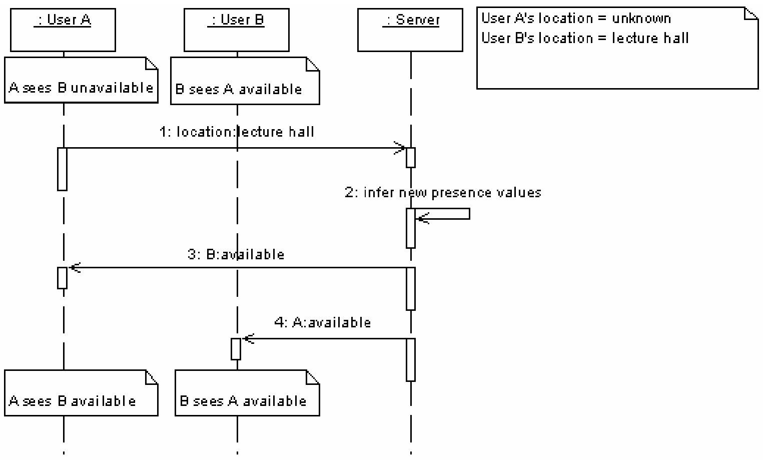


Fig. 4. Context relation-based inference of presence: User A goes into the same lecture hall where User B is already present. As a result, they see each other as *available*

We define the context pair (watcher, presentity) as a *context relation.* In IM, a context relation is established between two users when one adds the other on to her contact list. The contexts related by a context relation can be represented as:

$(context_w, context_p)$

where,

context = context value,
p = presentity = user,
w = watcher = user or a group.

In the above example, the user denotes an individual user of the IM system. The group denotes two or more users grouped together in the IM system. A wildcard *any* can be applied instead of a context value in the above expression. The number of context relations for a (watcher, presentity) pair is not constrained; there can be as many relations as are required to achieve the desired context-aware behavior. Furthermore, the watcher's context value refers to an individual user's context even when the watcher is a group. Groups are used to minimize the number of relations and rules needed to implement context-sensitive presence updating.

Context relations are used to form context relation-based rules that are controlled by the presentity. A rule for inferring presence from a context relation can be represented as:

$(context_w, context_p) \rightarrow presence_p.$

A wildcard *any* can be applied instead of a watcher or a context value in the above expression. A rule for *any* watcher is used when there are no specific rules for the watcher in question. Further, *any* context matches all contexts. It can be used, for example, to specify that, for a specific context of a presentity, a certain presence is to be inferred no matter what the watcher's context is.

The presence of a user is inferred separately for each watcher as follows: First, those rules for this (watcher, presentity) pair are triggered where the context values match the presentity's and the individual watcher's contexts. A rule for a watcher group is triggered if the individual watcher belongs to that group. Second, one rule of the triggered rules is selected (based on the rule priorities) and fired, resulting in a new presence value. The rules are owned by the presentities. Hence, the rules are used to control what presence the watchers see.

When the context of a presentity changes, the presence must be updated in two different ways:

1. *For each watcher, infer the new presence of the presentity according to the presentity's context relation rules.*
2. *For each contact, infer the new presence of the contact according to the contact's context relation rules.*

Context relations and rules can be illustrated by the example shown in Figure 5. James has the users Peter, Diana, and David on his contact list, which means that he is watching these three users. Sarah and Diana are watching James' presence (i.e. they have James on their contact lists). Now, James might have the following rules:

$(office_{Sarah}, office_{James}) \rightarrow available_{James}$

$(any_{all}, office_{James}) \rightarrow unavailable_{James}$

Let's assume that James' place changes to *office*. The system performs the steps 1 and 2 defined above. In step 1, James' rules are applied to infer James' presence for the watchers Diana and Sarah. For Sarah (assuming her context is *office*), both rules are triggered, but the first is fired as it has higher priority. There is no rule for Diana, so the rule for the group *all* is triggered and fired, because Diana is a member of this group. Further, the context *any* in the rule matches Diana's recognized context (even if the context is unknown). As a result, Sarah can communicate with James, but Diana cannot.

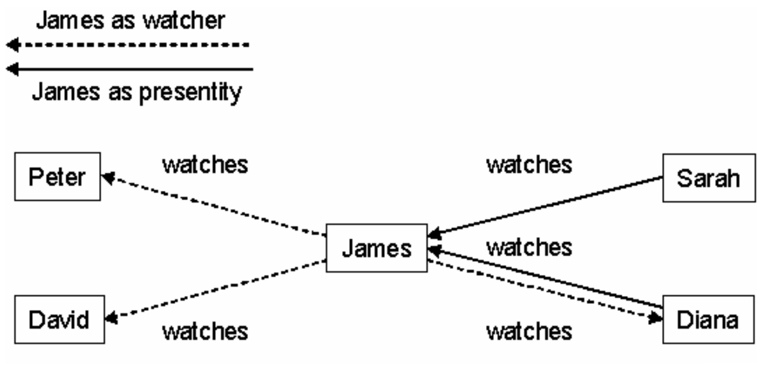


Fig. 5. Users as presentities and watchers

In step 2, we go through the contacts on James' contact list. Taking Peter as an example, the system would now find all the rules (not shown) for the presentity Peter. Then, a specific rule for the watcher James (or a group to which James belongs) would be searched and fired. This example illustrates how groups can be used to specify presence for the individual watchers that have no specific rules. The groups can be general, i.e. seen by many users, or a user can create groups for herself and specify the rules for those groups.

To further illustrate these issues, the steps 1 and 2 above are shown as a simple algorithm in Figure 6. In the figure, the watcher list is a list of watchers of a presentity. Furthermore, the Tables 1 and 2 show an example of how the contexts of a watcher and a presentity connected by a context relation can be represented. Each cell contains the resulting presence for a pair of context values (watcher, presentity), and each cell thus specifies a rule. The tables have a row for each context value of the watcher and a column for each context value of the presentity. Separate tables or rows are required for each separate watcher for which rules are defined.

Table 1. Presence rule table for context type activity for inferring presence from a context relation

		PRESENTITY'S ACTIVITY	
		CONFERENCE	**UNKNOWN**
WATCHER'S ACTIVITY	**CONFERENCE**	AVAILABLE	AVAILABLE
	UNKNOWN	UNAVAILABLE	AVAILABLE

Table 2. Presence rule table for context type place for inferring presence from a context relation

		PRESENTITY'S PLACE		
		OFFICE	**HOME**	**UNKNOWN**
WATCHER'S PLACE	**OFFICE**	AVAILABLE	UNAVAILABLE	UNAVAILABLE
	HOME	UNAVAILABLE	AVAILABLE	AVAILABLE
	UNKNOWN	AVAILABLE	AVAILABLE	AVAILABLE

In these tables, the value *unknown* may denote that the data is not available because of access control issues. It could also be interpreted as a place or activity that the system cannot recognize. The rules in Table 2 specify, for example, that when the user is at home, watchers at the office see her as *unavailable*, while watchers at home see her as *available*.

Although the context relation-based rules are applied automatically, the user can explicitly specify her presence (possibly for each watcher separately) at any given moment. The system may provide this option in two ways: First, the user can select her presence, as in an ordinary IM application, from a menu. Second, the user may be able to specify a rule "In the current context I'm *unavailable* for all users". This is semi-automatic context-based presence: the user manually selects the rule, and thus her presence, for the time being, but the presence is automatically updated according to changes in context. In the example, the user-defined rule determines the presence for the time the context remains the same. When the context changes, that rule is deactivated and automatic presence updating continues.

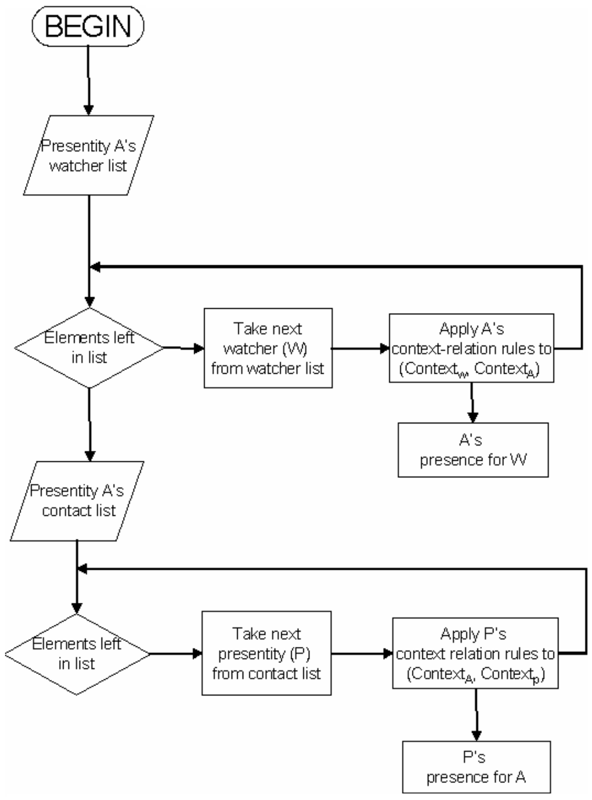

BEGIN
Presentity A's watcher list
Elements left in list
Take next watcher (W) from watcher list
Apply A's context-relation rules to (Context$_W$, Context$_A$)
A's presence for W
Presentity A's contact list
Elements left in list
Take next presentity (P) from contact list
Apply P's context relation rules to (Context$_A$, Context$_P$)
P's presence for A

Fig. 6. Algorithm for inferring presence when a presentity's context changes

4 Application

4.1 Capnet System

The Capnet (Context-Aware and Pervasive Networks) program focuses on context-aware mobile technologies for ubiquitous computing. At the highest level, the Capnet architecture is decomposed into Capnet Engines, see Figure 7. Each device that belongs to the Capnet universe contains an engine. An engine may be in a powerful server without any user interface or in a mobile device with many application user interfaces (UIs). As Capnet is a component-based software architecture, the basic building blocks of the engines are component instances, each specialized for producing the functionality of a certain domain area, such as service discovery, user interface, context recognition, media processing, connectivity management, component management, database access, or any service added to the system by developers. The Capnet universe is a distributed environment, which means that the engines can use component instances running on the local device as well as remote instances running in engines somewhere in the Capnet universe to provide the required functionality for the applications.

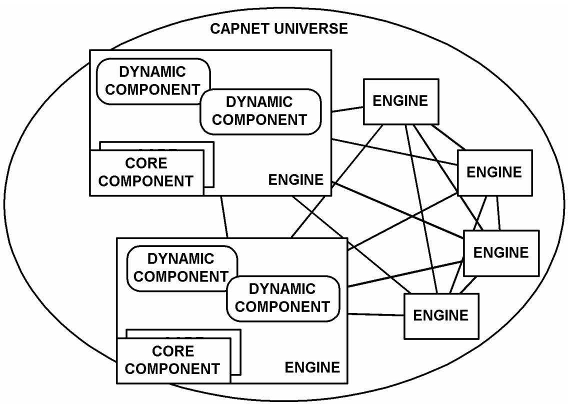


Fig. 7. Capnet universe

Considering the design of an IM system, the distribution shown in Figure 7 gives a natural starting point. As the users of IM are distributed in the network (using mobile devices or many different desktops), each engine in Figure 7 can be seen as a device hosting the IM application of a user. Considering the natural connectivity of the engines, the Capnet architecture seems well suited for developing an IM application.

The context components provide abstraction of context information for its consumers in the Capnet universe. Context components receive sensor data from a number of sensors and use the data to infer higher-level context information that is utilizable by the other Capnet components.

4.2 Design and Implementation

The context-aware instant messaging system was developed on the existing Capnet system. Below, we will describe the Capnet IM system briefly, without going into details of implementation.

The IM system is composed of the components shown in Figure 8. In Capnet, the schedule information comes from a calendar application, while the location information is provided by a location sensor component. The context components receive sensor data and infer contexts from it.

In our current implementation, automatic presence inference is performed by the IM component, which receives context information from the context component. In the next version, we plan to update the context component to provide presence information as well. The context types *activity* and *place*, as shown in the Tables 1 and 2, are used in inferring presence. The inference is done using hard-coded rules common for all users; explicit representation of these rules is one challenge for further research. There are rules for the groups *co-workers*, *family*, and *other*. The group used for presence inference is defined by the group of the watcher on the contact list of the presentity. If the watcher is not on the contact list of the presentity, then the group *other* is used for presence inference. A better way would be to show the watcher list for the presentity and to allow her to group the watchers.

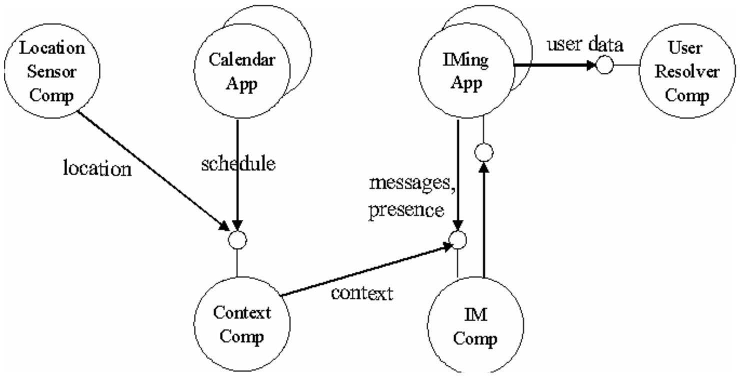


Fig. 8. Software components of the prototype IM system

If a user sets her presence manually, the application instance reports the value to the IM component, which delivers the information to the watchers. The messages travel through the IM component from application to application. The user resolver component is used to resolve users by their name and to fetch user data by the user ID. Unique email-like identifiers identify users, and an identity is associated with a user when she logs into the Capnet system.

Only the UI component is required to be executing in the thin terminal, e.g. a PDA, while the other component instances may be run on any device running a Capnet engine, i.e. on a network on some server.

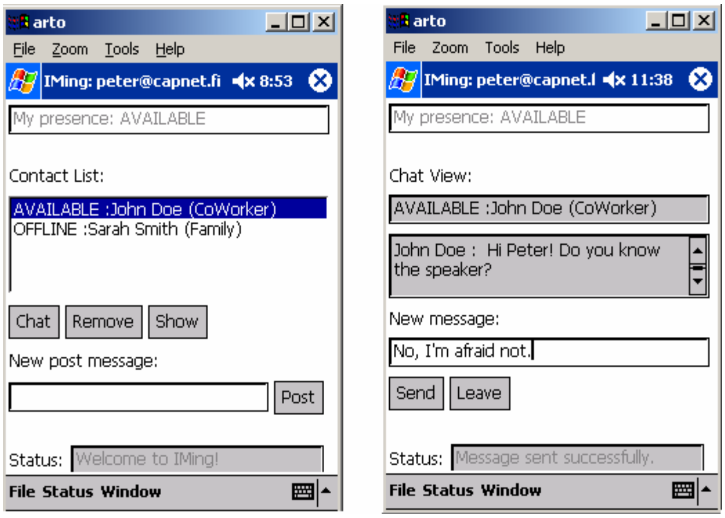
4.3 Test Environment and Testing

The mobile device, Compaq iPAQ PDA, is equipped with WLAN cards and IBM's J9 virtual machine. All mobile device components are implemented according to the PersonalJava 1.2a specification. Devices are located with the Ekahau positioning engine that utilizes WLAN signal strengths measured in the devices. The university's premises covered by WLAN are used as the test environment. All inter-device communications utilize XML-RPC; the open-source Marquee XML-RPC library is used.

We demonstrated the system with two users according to the scenarios in the Figures 3 and 4. The users were using the PDAs described above. A smart living room environment was configured to be the home for the users, a meeting room to be the lecture hall in the scenarios and an office in our premises to be the office for the user. Also, we added an event occurring in the meeting room into the calendars of both users.

When a user entered the meeting room, the Capnet system recognized the context change, allowing the IM system to infer a new presence for the user. In the case of two co-workers present in the same meeting room, the system successfully utilized context relation to infer them as *available* for each other. Overall, the system functioned as shown in the scenario in Figures 3 and 4.

Below, we present two screen shots of the Pocket PC screen captured via Microsoft's Remote Display Control for Windows CE. Figure 9 shows the contact list view and a view of an ongoing chat of the IM application prototype. On the contact list, the user sees the presence of her contacts and the group that she has selected for each contact when entering it.



My presence: AVAILABLE

Contact List:

AVAILABLE :John Doe (CoWorker)
OFFLINE :Sarah Smith (Family)

New post message:

Status: Welcome to IMing!

My presence: AVAILABLE

Chat View:

AVAILABLE :John Doe (CoWorker)

John Doe : Hi Peter! Do you know the speaker?

New message:

No, I'm afraid not.

Status: Message sent successfully.

Fig. 9. Views from the IM application prototype. Left: Contact list view. Right: Chat view

From the screen shown on the left in Figure 9, one-to-one messaging can be started by selecting a user from the contact list and clicking the ‘Chat’ button. A single one-to-many message can be sent by selecting users from the contact list and clicking the ’Post’ button. In both cases, the selected users must be *available*. In ongoing chats, the presence of the participants may change to *unavailable* at any time, but the communication can still continue. When inferring presence, the system also generates a textual description of the reason for resulting presence value specific to each context relation. Clicking the ‘Show’ button displays the description of the selected user.

5 Discussion

We introduced context relation as a new concept in context-aware instant messaging. The key idea is to utilize both the communication initiator’s and the receiver’s contexts in inferring presence. We presented an IM prototype that utilizes context relations to update users’ presence information automatically. Automatic updating enables user-friendly IM applications with reliable presence updating. It reduces disruptions by minimizing the number of communication attempts by others when the user is not willing to communicate. Furthermore, utilization of context relations in automatic updating allows sophisticated control over the presence information delivered to others.

Being a new concept, context relation requires further research. Preliminary experiments show that the use of context relations in inferring presence may improve the user experience of IM applications, but this should be studied in more detail. Furthermore, a clear way to visualize the context relations and the rules would facilitate the creation and modification of personal relations and rules. Moreover, introducing more versatile contexts may require modifications of the context relation model we presented. Context relations could be used in other awareness applications as well, as awareness generally means knowing the activities of others [15].

Although we allow groups as watchers in the rules, the resulting context relations are one-to-one relations between two users. However, it would be possible to extend the relations to one-to-many. For example, the user could set a rule: "When my *co-workers* are having a *coffee break* and I am *working*, remind me about the coffee break". A separate threshold could specify the number of group members that are required to have a coffee break before the group context is set to *coffee break*. Similarly, we could allow presentity groups. For example, a system with the predefined groups *patients*, *nurses,* and *doctors* might set the presence of a nurse to *available* for patients when the nurse is at her desk and no doctor is near her (i.e. discussing with her). In this example, it is assumed that all patients and doctors are watchers of the nurses.

As automatic context recognition is such a challenging topic, systems providing automatic presence updating still need to support manual presence selection as well. Furthermore, the system could learn the manual selections – when a user selects the same presence in the same context repeatedly, the system could suggest automatic setting of presence. The routine learning methods presented in [16] could be used in learning the presence rules. It would be important that the user could control the learning, deciding which rules would be accepted and which rejected. Even when there is no learning, the user should be able to specify whether she wants to confirm the automatic presence changes, or if the new presence is applied without confirmation.

A context-aware instant messaging system should utilize the contexts of its users by all reasonable means. The other prototypes described in chapter 2 have demonstrated the feasibility of utilizing context in IM in various ways. The ConChat [10] prototype does well in enriching IM communication with context information. Awarenex provides awareness information on whether the contacts are having phone calls, are engaged in other IM discussions, or have a scheduled calendar event going on [12]. The Capnet IM system did not aim to enrich communications or to provide more awareness information, although we consider them equally important as reliable presence inferring for an IM system. As an addition to the features of Awarenex, Begole et al also demonstrated the prediction of availability from presence history data [13]. The routine learning methods discussed above could be used to predict presence as well. Predicted presence could be used in automatic presence inference when, for example, there is not enough information to infer presence from context.

Lilsys [11] infers user's availability from sensor data, such as sound, phone usage, and computer activity. Using inferred availability, the system gives cues to other users as 'neutral', 'possibly unavailable', or 'probably unavailable'. The system utilizes a wider range of sensor data than our prototype. The Capnet IM prototype aims for better presence inference by utilizing the context relation. The conceptual representation of the context relation enables the application of sophisticated rules to presence inference.

Dourish and Bellotti point out a problem in awareness: the appropriateness of information about a person's activity at a given time for a receiver depends on the receiver's needs [15]. This problem could be alleviated in a context-aware IM system by choosing the exposed context information about a user to a receiver based on the receiver's context. For example, more context information could be exposed about the users involved in the currently most active instant messaging session than about passive users on the contact list.

Although we did not discuss or implement any security or privacy issues, we want to mention that context relation also has implications for these design issues. Basically, the user must have control over who can access her presence, or context, information. Privacy in presence awareness systems has been studied widely, for example in [6].

In addition to security and privacy, instant messaging standards [17] need to be considered in future work. Another future topic might be representation of the context relation-based presence rules by RDF. As a conclusion, we have demonstrated the advantages of context relations and the rules applied to them. We will continue to develop the concept.

Acknowledgments

This work was funded by the National Technology Agency of Finland. The authors would like to thank all the personnel in the Capnet program, and the participating companies.

References

1. Cutrell E., Czerwinski M. and Horvitz E. (2001) Notification, disruption, and memory: Effects of messaging interruptions on memory and performance. INTERACT 2001 Conference Proceedings. IOS Press, IFIP, 263-269.
2. Nardi, B., Whittaker, S., & Bradner, E. (2000) Interaction and Outeraction: Instant messaging in action. Proceedings of the ACM Conference on Computer Supported Cooperative Work (CSCW 2000), New York, USA, 2000, 79-88.
3. Bradbury, D. (2001) Pigeon post for the 21st century. Computer Weekly, 2001 October 18, 1–6.
4. Fogarty, J., Lai, J., and Christensen, J. (in press) Presence versus Availability: The Design and Evaluation of a Context-Aware Communication Client. To appear in International Journal of Human-Computer Studies (IJHCS).
5. Greene, D., O'Mahony, D. (2004) Instant Messaging & Presence Management in Mobile Ad-Hoc Networks. In Proc. of the IEEE Annual Conference on Pervasive Computing and Communications Workshops (PERCOMW'04). Orlando, USA, March 14-17, 2004.
6. Godefroid, P., Herbsleb, J.D., Jagadeesan, L.J., Du Li. (2000) Ensuring Privacy in Presence Awareness Systems: An Automated Verification Approach. In Proc. of the ACM Conference on Computer Supported Cooperative Work (CSCW 2000). Philadelphia, USA, Dec. 2000, 59-68.

7. Peddemors, A.J.H., Lankhorst, M.M., de Heer, J. (2002) Combining presence, location and instant messaging in a context-aware mobile application framework. GigaMobile/D2.8 (TI/RS/2002/068) Telematica Instituut Enschede. https://doc.telin.nl/dscgi/ds.py/Get/File-21982/PLIM_d28.pdf
8. Day, M., Rosenberg, J., Sugano, H. (2000) A Model for Presence and Instant Messaging. RFC 2778, IETF.
9. Dey, A.K., Abowd, G.D. (1999) Towards a Better Understanding of Context and Context-Awareness, College of Computing, Georgia Institute of Technology, Atlanta GA USA, 1999, Technical Report GIT-GVU-99-22.
10. Ranganathan, A., Campbell, R.H., Ravi, A., Mahajan, A. (2002) ConChat: A Context-Aware Chat Program. IEEE Pervasive computing, Volume: 1 , Issue: 3.
11. Tang, J.C., Begole, J. (2003) Beyond Instant Messaging. ACM Queue 2003 November, Volume 1, Issue 8.
12. Tang, J., Yankelovich, N., Begole, J., Van Kleek, M., Li, F., Bhalodia, J. (2001) ConNexus to Awarenex: Extending awareness to mobile users. In Proc SIGCHI Conference on Human Factors in Computing Systems (CHI 2001). Seattle, USA, 2001, 221-228.
13. Begole, J. B., Tang, J. C. and Hill, R. (2003) Rhythm Modeling, Visualizations, and Applications. Proceedings of the ACM Symposium on User Interface Software and Technology (UIST 2003). Vancouver, Canada, 2003, 11-20.
14. Voida, A., Newstetter, W. C., Mynatt, E. D. (2002) When Conventions Collide: The Tensions of Instant Messaging Attributed. Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI 2002) Minneapolis, USA, 2002, 187-194.
15. Dourish, P. and Bellotti, V. (1992) Awareness and Coordination in Shared Workspaces. Proceedings of the ACM Conference on Computer Supported Cooperative Work (CSCW'92). Toronto, Ontario, Canada, 1992, 107-114.
16. Pirttikangas, S., Riekki, J., Porspakka, S., Röning, J. (2004) Know Your Whereabouts. 2004 Communication Networks and Distributed Systems Modeling and Simulation Conference (CNDS'04), San Diego, California, USA, 19-22 January 2004.
17. McCleaa, M., Yena, D.C., Huang, A. (in press). An analytical study towards the development of a standardized IM application. Computer Standards & Interfaces, Volume: 26, Issue: 4, August 2004, 343-355.