Instant Messaging and Presence: Research and Challenges

Jaakko Kangasharju and Marko Saaresto

***Abstract*— Various instant messaging and presence (IMP) systems have gained much popularity in recent years. However, existing systems are typically designed to the current computing environment and do not take into account the future. This paper reviews some research done in IMP systems for the future computing environment. In addition, we summarize challenges faced by IMP systems, and provide a roadmap for future research and development.**

***Index Terms*— Instant messaging, presence**

I. Introduction

Instant messaging has gained tremendous popularity in recent years and may soon be as common place a service as email and the WWW. So far the developement of IM technologies has been service driven, resulting in non-interoperable communities and market fragmentation [1]. At the same time instant messaging and presence services can be seen as enablers for richer and more user-centric communication of the future, being manifested as complementary features in business, productivity and communication applications. [2] Before indulging in prototypes, let us summarize some basic concepts on these technologies.

A. Concepts of Services

As has been established [3], [4], instant messaging and presence services, althouhg often seen as co-existing features of the same service, are conceptually and fundamentally different. They do complement one another when coupled but have some distinctly different properties when considered e.g. security, privacy or interoperability wise.

a) Instant messaging: engaging in a dialogue (or notifying another party) by message passing, using a network of devices. Notable features are the timeliness of the message delivery and the disjointed dialogue pattern. While a message can be delivered to the recipient terminal almost instantly, it is not uncommon for messages being queued until the recipient is reachable (logged in) or the user to actually read the message a while later at her convenience. Instant messaging should not be confused with streaming media communication like voice and video which, unlike IM, have real-time requirements.

At the very basic level IM is delivering a short message to a recipient – the person, not the inbox – as soon as possible. Features like group chat, pictograms, device selection etc. are just extensions of this.

b) Presence service: a relatively more complex technology. Earliest implementations provided other users with knowledge about whether a given user is logged in to the service or not, and possibly when she was last seen. Later services have begun to enrichen the information being related about the state of users. A user can be marked as being logged into a system but away from the device she uses, or she might have some other flag providing cues for contacting her. The *presence in the service* can be further extended to *presence in general* by disclosing further data like current activity, location or other contextual information retrievable.

Thus basicly presence service gathers information about the state of a user and make this available for other users. Features like presence semantics or context inference (II-F) are just extensions of this.

B. Related concepts

Instant messaging and presence services are pervasive technologies: message notifications are often made interruptive, prompting of user attention, and many client implementations track the user for example through keyboard activity. This pervasiveness raises some questions that should be considered when considering and developing IMP technologies.

c) Identity: a split goal. On one hand it is imperative that a user can prove her identity so that engaging in sensitive discussion is feasible. Many IM services maintain a user registry and try to verify each account using credentials (password). Although some may require providing identifying data like name and email address the means to verify these may be limited.

On the other hand it is important to be able to communicate anonymously or ad hoc: in these cases the service should either mask the identity of users or not require them to provide any identifying information. Delegating identification and using anonymized identifiers may provide a good a service level without unnecessary risks on credentials or identity. [1]

d) Privacy: on this topic messaging and presence differ significantly. On messaging it means the ability to control the influx of communication: to decide who, when and how can intrude your privacy. Failures at this mean unsolicited messages, unnecessary interruptions and possibly offensive content. A common solutions is to only accept messages from whitelisted users.

On presence service a user categorically forfeits her privacy on presence information the moment the data is published. The less privacy is retained, the more efficiently can other users adapt their communication and behaviour. As more information is inferred from a user (II-F) it becomes important to disclose only the minimun amount of data necessary to meet the needs. Also, the user may want to alter the maximum level

[1] http://www.projectliberty.org/

of presence details that are being provided. And to make the matter more complex, the presence data – in particular the reachability data (busy, free etc.) – may be different depending on who is asking.

e) Security: in particular confidentiality of messages and the service reliability. From an organizational point of view this also includes securing the internal communication from both external and internal threats. For some the ability to intercept and track instant messaging and presence traffic is a security issue.

f) Trust: the users of IMP systems need to trust the service like none before. When the services begin to go mobile (III-D) the potential impact on privacy can be tremendous. The ability for a single entity to tap into casual discussions, social and business relations, and perchance user location is as lucrative as it is terrifying. At some point users, organizations and legistlators will have to consider the possibilities of and countermeasures to abuse, and where to place their trust.

II. Prototypes and examples

This section reviews interesting prototypes of past. There is a lot of research going on that relates more or less to the concepts of instant messaging and presence, among others in the area of context-awareness. We have selected the following prototypes as interesting and good examples of the research in the field.

A. Active Badge

Active Badge [5] was one of the earliest reseach prototypes in presence and context awareness. The goal was to improve the reachability of employees by tracking their whereabouts and redirecting phone calls to the nearest device at the time mobile phones were not yet commonplace. The experiment that was conducted both demonstrated the applicability of context-awareness and raised some important questions about such systems.

The Active Badge was an infra-red transmitter [2] that could be attached to clothing of a person. Each badge was personal and broadcasted a different, identifying signal. Networked IR detectors were installed in all staff premises to report to a server as they detect the signals. The call center operator, upon handling an incoming telephone call, could then check from a terminal where the requested employee was last detected and connect the call to some phone in that room.

As it was physically possible for the person and her badge to be at different locations this caused some challenges for the reliability. For example if a user left a badge on her table it would fallibly report the person to be in office. These could be countered to some degree by infering an overnight static position as false. The infra-red signal could also be easily obstructed by objects or clothing, resulting in changes in location not being updated. The timestamp of the latest recorded sighting can be used for infering the reliability of the data.

The badges could be turned off to save battery when not at the office. Turning the badge off or slipping it into a pocket was also an easy way to regain privacy from the system. The power to unconditionally control the privacy of one self was considered an important feature by the users. The overly ease of hiding was seen as a problem by the researchers. Another privacy issue identified was unrestricted access – and potential abuse — by the staff to the location information.

[2]http://koo.corpus.cam.ac.uk/projects/badges/

B. Hubbub

The researches at AT&T who created and worked with the Hubbub [6] instant messenger prototype can be said to have made two major contributions. First of all they experimented with a novel user interface modality by introducing what can be described as *sound icons* and *Sound Instant Messages (SIM)*. They also provided the client and the service free-of-charge over the internet [3], attracting users and thus being able to collect important usability data through interaction monitoring and interviews.

In Hubbub each user selects a sound icon, a simple melody of a few notes. This melody is then used to identify the user to others. Every time a person becomes active their melody is played for those who have her on their contact list. This way users can be aware of who is around and who has just arrived, without keeping the Hubbub window visible or suffering popups. This was felt as an acceptable level of interruption as it is possible to identify another user without a context switch from whatever the user was doing or reading at the moment.

Sound icons are also used for simple notification-like messages, either stand-alone or attached to normal text messages: *Hi, Bye, Talk?, Busy, Ready, Yes, No, OK, Cool, LOL (Laugh Out Loud), Bummer, Thanks, No Problem, and BRB (Be Right Back)* . A user was notified of an incoming message by playing the melody of the sending person. Thus it was possible to register a simple reply from another user without any visual cues.

The Hubbub project records all communication that takes place in the service for studying IM communication patterns. They studied both the context and goal of recorded dialogues as well as the patterns in discussion lengths, continuity and distribution over time. Their results are a valuable comparison [7] for other usage pattern studies.

C. Fuego Core IMP Service

The Fuego Core (FC) project [4] experimented with novel middleware services for mobile computing. The project also experimented with an instant messaging and a presence services built on top of the project's mobile distributed notification service which provided advanced features like content based filtering and session transfer. Using these features the IMP prototype could achieve virtual peer-to-peer operation: both messaging and presence dissemination were conducted and controlled purely between clients by publishing and subscribing to notifications.

Using a notification service a simple instant messaging service is trivial to design and implement. The novelty in the

[3]http://www.hubbubme.com/

[4]See http://www.hiit.fi /fuego/fc/

messaging design was the support for threading discussion: replies to earlier messages could be identified as replies and be grouped to a single thread. Using the notification service filtering it could also have been possible for a client to filter out one or more such threads on a discussion channel. Another small novelty was that if a user had more than on device to access the services messages read on one device could be hidden on or removed from the others to reduce clutter and redundancy.

The FC presence service consisted of three software components that could be used by a client application in any combination. The early design of these resembled the architecture of the SIP/SIMPLE [8] service but evolved towards the final serverless design. The idea on the presence dissemination was that the application interested in monitoring a user establishes a subscription relationship to the relevant data. A subscription defines the types (location, mood, availability) of data of interest. If data sources for these types are found, a subscription can be established if allowed.

What is notable about the design is that there are very few consistency requirements. While this allows sidestepping many synchronization and selection problems it somewhat complicates application design as it is impossible to reliably know about the lifecycle of other applications and their components.

g) Presence Producer: the simplest of all, providing a way for an application, capable of detecting some kind user presence or context information, to make that information available to other applications and users. The producer should know what kinds of data it may provide and reacts on requests on those types.

If the application using the producer has not created a Resolver component a producer must seek out and consult a remote Resolver before it can either deny or allow a subscription to the data it produces. A producer may also receive notifications of policy or subscriptions changes that may cause termination of existing subscriptions.

h) Presence Consumer: a component that provides an application means for retrieving presence information about users in the system. This component sends out the subscription requests and manages the established subscription states. It also provides aggregation and local storage (including expiration) for received data.

i) Access Resolver: responsible for making access control decisions. A Resolver should ideally be instantiated by an application capable of user intaraction: to alert and query a user when an access control decision needs to be made. Once made the Resolver will store the decision for future reference. A Resolver will also remember decisions made by other Resolver instances. If the user does not make a decision within a pre-determined time-frame a Resolver may issue a provisional access control decision based on policies. Supporting access policy list and privacy profile export and import allows user applications to implement a permanent storage for these.

One major advange of the disjointed operation is that there are no bottleneck components that have to be waiten upon. The prototype design is however very crude and does little to address issues like security and trust.

D. Dawn IM

A major area of current interest are mobile ad-hoc networks (manets) where there is no fixed networking infrastructure, but rather mobile devices communicate with whatever other devices are in range. Research in manets is hoped to take us further toward the goal of ubiquitous computing where the computing infrastructure disappears into the environment yet is always and everywhere available.

However, traditional IMP systems are focused on the fixed network case. It is expected that clients are constantly connected to the network, updating their presence information and available to receive messages. IMP servers are relied on to manage and distribute presence information and to route instant messages from one client to another.

In a manet these assumptions break down. There is no network to which clients are constantly connected. It is not possible to rely on a server for anything as any node selected as the server may move away from connectivity range. Furthermore, use of the network needs to be conserved to avoid draining a mobile device's battery too quickly.

Researchers at Dublin's Trinity College have implemented the Dawn IM system [9] for instant messaging and presence in manets. It uses an overlay-based approach for resource discovery, which allows communication between nodes. The main contributions are in the areas of presence information management and distribution as well as in carrying on conversations.

In the Dawn IM system each client, or entity, is responsible for its own presence information. The semantics of the presence information are borrowed from XMPP [10], as well as some considerations on how to distribute the information. In a fixed network with assumed connectivity and a centralized presence server it makes sense to use a subscription-based approach where the server informs interested clients whenever the presence status of an entity changes.

In a manet this subscription-based approach is still possible to use, but here subscriptions are made directly to the entity in which the interest is. However, if an entity is not contactable, the subscriber will need to retry at a later time. Furthermore, with intermittent connectivity being the norm, a subscriber can never be sure whether an entity still remembers previous subscription information (e.g., it may have been switched off), so it will need to re-establish its subscriptions. This essentially reduces the subscription-style presence distribution to periodic polling.

Similarly, in the case of conversations, each entity will need to retain state on all the conversations it is participating in. Each entity is directly responsible for sending its messages to all the other entities in the conversation. This sending is accomplished by using the uni- and multi-casting capabilities of the underlying overlay network.

Dawn IM allows conversation groups to be updated dynamically by entities either joining or leaving a group. Like with messages, these updates need to be sent to all the members of the group to properly synchronize the view of group membership at all entities. Furthermore, presence updates are used to detect when a node has moved outside the communication area to automatically remove them from the group temporarily.

With the current proliferation of capable devices it certainly seems that ad hoc networking needs to be taken into account in the future. Users will want the same services available everywhere they go, as they now have a device capable of providing those. Considering how instant messaging and presence can be integrated into the ad hoc world is one necessary step in this.

The Dawn IM system may be a step in the right direction. However, it is still a research prototype and therefore has not considered issues such as privacy or communication efficiency. Furthermore, there appears to be no intent to be interoperable with existing IMP architectures, making Dawn IM usable only inside an ad hoc network in a limited geographical area.

E. FriendZone

With mobile devices comes the concept of location awareness. As a user is mobile, his location will change, and potentially this location could be used to determine what is appropriate to present for the user. A hope of some is that these location-based services could be the killer application of mobile Internet.

The FriendZone system [11] attempts to provide location-based messaging to its users. The main components of it are an instant messenger and locator, a mobile chat application, and an anonymous instant messenger (which is confusingly shortened to AIM in [11]). FriendZone explicitly considers both the capabilities and the variety of mobile terminal devices, by providing simpler interfaces and a reduced level of services to less capable devices.

The instant messenger and locator component of FriendZone is very similar to most of the existing IMP systems. The only addition is location information to a user's presence. Location in this component is considered to be either absolute based on cell information or relative to the querying user. Using this latter, a user can determine which of his buddies are close by, e.g., in case he wants to meet them.

The mobile chat application is a simple chat room enhanced with location information. In this case absolute location is not available, but relative locations are. By exploiting location information it is possible to create local chat rooms, which are only available at a certain geographical area.

The anonymous instant messaging is based on each user having a personal profile, which includes both demographic and personal information. Users can also select profile data that is of interest to them. The messaging service then matches users' interests and tries to find matching users in the pool of all users. This means that the service can pair users who have never even heard of each other.

The service lists matching profiles and users can then select to talk with these matching users. Relative location information is available as in the mobile chat, and can be used to limit the selected profiles. There is also the possibility to connect the matched users by checking buddy lists and showing to users in how many steps they can be connected through various buddy lists.

As mentioned, FriendZone supports a variety of end-user terminals. Mostly the presentation of buddy lists differs, with graphics used wherever feasible and simple textual clues replacing graphics where only text is available. There is no information on whether messages can also be enhanced on more capable platforms, but it seems that in the current system only textual messages are available.

FriendZone also includes a host of privacy management tools. These provide similar functionality to existing presence services by making the sharing of presence and location information an explicit choice of the user. It is possible to block other users from seeing the information, and in some cases this blocking is done automatically. However, the user survey reported in [11] notes that the users mostly did not care at all about privacy management, and preferred to show their location to everyone. This is somewhat interesting when considering that the privacy features of FriendZone were implemented because of legal requirements.

F. Context Awareness

The concept of presence of a user can be seen as a subset of the more general field of context awareness. Context is seen as any information on the situation that is relevant to the computing application [12]. In this sense a user's presence is just some context information that is especially relevant to instant messaging applications.

In context-aware systems research the focus is on automatically determining context. This is typically achieved by utilizing a variety of sensors available either as attached on the device or embedded in the environment. Through sensors it is possible to determine location, movement, temperature, etc.

Context-aware system also typically aggregate and fuse context data from sensors into higher-level context [13]. For instance, sensing a user's movements and muscle activity it is possible to determine whether the user is walking, running, etc. Context data is also typically used for inference in an attempt to proactively configure the system so that it anticipates the user's future actions.

For presence systems the most interesting facet is the inference of high-level context from gathered low-level information. For instance, a typical example given is that when a room's light level is low, one person is standing and speaking and others are sitting quietly, the person standing is typically giving a presentation. Such inferences are very useful to provide accurate presence data.

The AWARE architecture [14] is a work to build social awareness, i.e., awareness of the context of other people, into applications. It is mainly concerned with a hospital environment with nurses, doctors, and patients. The research work is in the field of cooperative work where interruptions can be frequent and disruptive. The hope is that computer assistance could be used to minimize the disruptiveness of these interruptions.

A case study for the AWARE architecture is the AwarePhone, which is an application for mobile phones that supports cooperative work. The AWARE architecture is responsible for monitoring the context of each user, which is then available to other users through the service. The AwarePhone user can

hence see the context before trying to contact another person. By being implemented on a mobile phone, the messaging can take place using the normal phone system, i.e., calls or text messages.

The AwarePhone itself is not a proper IMP system, as it only provides the standard contact capabilities of the phone and not any instant messaging functionality. However, the authors point out [14] that the awareness service of the AWARE platform has also been used to implement an actual instant messaging system. In addition, while the AwarePhone uses only context relevant to hospitals, because of its generic handling of context, it is simple to extend to handle and provide a much richer view of a user's context. This would enable rich presence applications without the need for special-purpose coding.

III. Visions and challenges

A. Security

When discussing on securing an IMP system we must first consider on where to draw the line of defense what are the interests to protect. If these are not compared with the intended use cases of the system it is possible to end up with a complex implementation of questionable effectivity [15].

A basic requirement is confidentiality in that no user can intercept messages (personal or system level) that are not meant for her. This can be further reinforced by encrypting the client-server communication. When using persistent identities this also indicates that users should be authenticated to prevent a breech through identity theft.

An extension of protection from malicious users is the protecting the service providers themselves from other, fradulent or compromised providers. This is akin to email providers securing against intrusion and open relays.

A stricter requirement would be that the service provider can not read the messages it passes around, often referred to as end-to-end security. In this case the confidentiality requirements of the communication outweight the trust towards the service provider or the network of service providers that participate. Of course, a (compromised) provider can still track who communicates with whom and how often, which in itself may be valuable information.

To deprive third parties from even the information above is possible, for example by not trying to solve the problem globally. A company can hide it's internal IMP traffic by running a service in-house. Likewise a community could use their own server run by trusted persons. Perhaps implementing a peer-to-peer service utilizing some onion routing technology [5] could be more resilient.

Since security requirements vary greatly between use cases the meta-requirement is that an IMP systems should support varying levels of security levels and procedures. If not for the immediate users of that service then for interoperability with services configured differently (III-C).

[5] http://tor.eff.org/

B. Reversal of Presence Provision

In current IMP systems a user's presence is set by the user and made visible to other users, with access controls to determine who is allowed to view the presence information. While some current systems may allow a user to specify his own information, this is typically only possible as free-form text that does not have any significance to the system.

Some research in context-aware computing attempts to change this, in the larger scale of context information, by specifying ontologies for context information [16]. This would allow the system to understand the presence information, and to make inferences based on this information. Furthermore, context-aware computing promises to provide much more fine-grained information than is possible with current presence applications.

This kind of automated gathering of fine-grained presence information is not without its costs. As information gathering becomes automated, user privacy is in danger of being compromised. The development of a usable interface for specifying what kinds of presence information can be shared does not seem likely. The ContextPhone system [17] has an explicit interface that shows all the information that is currently being shared, but apparently fine-tuning of the shared information is still ongoing work.

The DeDe system [18] for instant messaging takes a different approach. It is built as a presence system on top of existing SMS and MMS infrastructure on mobile phones. In DeDe a user's presence information consists of time, location, and nearby devices, though extensions would naturally be possible, but the presence information is not shared at all.

When a DeDe user sends a message, he may specify that the message be delivered only if the recipient's presence information matches a specific condition. For example, a user can specify that a message be delivered only if the recipient is at a certain location like on the way home from school. The store-and-forward system of mobile phone messaging will then hold onto the sent message, and only deliver it to the recipient when his presence matches the specified one.

The system described in [18] is merely a prototype, but it is simple to see how to extend it. Specification of finer-grained filtering rules for when to deliver messages is an obvious extension. Similarly the gathering of more presence information and allowing the use of this in delivery rule specifications would enhance the system. An issue identified by the authors is delivery notification: a message sender would often want to see whether his message was received, but this would naturally compromise the idea of not sharing presence information.

It could be argued that the DeDe system's concept of not sharing presence information invalidates the whole concept of having presence information. After all, presence of friends and colleagues is also used in various other ways than just for messaging. We can, however, envision that any presence-using system could utilize the same idea of matching against a user's presence at the user's own service. On the other hand, presence information is used by people in various ways, some of which probably cannot be automated.

C. Unification of Services

Currently, there exists a variety of different IMP services, none of which are directly interoperable. Many people therefore have several IM accounts with various service providers, some using a different client for accessing each while others use a single client that supports several different protocols.

The most popular IM services have all been designed independently without thought of interoperability. While theoretically many systems include the possibility of gateways to convert between two different systems, such gateways have not been widely deployed. Furthermore, service providers have typically been hostile towards any reverse engineering of their protocols, wanting everyone to use their own provided client.

With Internet email there is a single standard protocol, SMTP and the associated message syntax, and a single format for addresses. This is admittedly a boon for email as there is no need to try to extract the needed protocol or syntax from a user's address, and communication with anyone is possible easily. A similar solution to IMP could likewise be very useful, but email is a different application, so the same solutions may not be applicable.

Email is a very asynchronous form of communication. A message is sent, but the sender will not know when or even whether the recipient received it. The recipient will read the email at a time convenient for him, not when it was sent. The parties need not even be online simultaneously. In contrast, instant messaging is a very synchronous application, more intended for conversation consisting of short messages than for sending extensive information. Furthermore, presence services typically require a user to be online, though offline messaging is supported in some systems.

Current popular IMP systems are built on the centralized server architecture, again in contrast to email, which is fully distributed. Evidently this is sufficiently scalable for current needs, but building a global unified IMP system would raise the question of who is to administer this service. A distributed architecture, like in XMPP, where users get their IMP account from their own service provider would provide a solution to this administration issue.

The standardization and acceptance of a single IMP architecture would hopefully bring with it a wide variety of different clients, giving users more choice and potentially an application better suited to their communication needs. The business models of established players in the IMP space would possibly not even need to change, as users would still associate their IMP service with their favorite client.

However, there needs to be a consideration of the features of the potential standard. To enable seamless migration, the system would effectively need to support the union of the functionalities of all current systems. This is a typical end result of a standardization process, often leading to a standard that is too much work to implement completely, therefore marginalizing the whole standard.

It is not unlikely that a single protocol could satisfy the needs of various IMP requirements. After all, email has been extended with capability for non-ASCII text and attachments in a (mostly) backwards-compatible manner, even though neither was supported in the first systems. However, instant messaging adds audio and video to this mix, which are not well handled by the same protocols as text and file transfers.

A further concern when considering a single standard is the semantics of presence information. In current systems this is not a problem, since the presence information is typically free-form text or a choice from a limited selection. However, when we imagine the rich presence applications of the future, and the requirements to automatically process these, the question of semantics is a crucial one. Considering that there is yet no agreement on even what kinds of information are classified as presence, let alone how to represent them, it seems premature to begin standardization work on a presence service now.

D. Mobile Presence

The future computing environment is expected to be widely different from the current one, with ongoing interest in ubiquitous computing. The vision is that everyone will be constantly carrying a computing device with them, and that the infrastructure provides both networking as well as various sophisticated sensor capabilities to these devices. With a user's preferred computing system being constantly with the user the management of presence information becomes more challenging.

The point of view when considering the existence of modern networked applications in the ubiquitous world is typically that of relevant resource constraints. Mobile devices are much smaller, with weaker processors and less memory, than typical personal computers. This requires applications to be written in a much more constrained manner, taking into account the processing limitations.

A more significant concern is the networking environment available for mobile devices. As the device is mobile, its network connection is by necessity a wireless one. Such connections have lower bandwidth and higher latency than typical fixed connections, and their use consumes a significant amount of battery power, which is seen as the major issue for mobile devices.

However, the resource constraints are not the only new concern in the new environment. As users will be mobile, their context, including their presence information, will be much richer than when they are just sitting at their desktop computers. Utilizing this information fully could launch presence services to a completely new level of applicability.

A simple extension to the presence concept would be location. In current systems a user's location is not often presence information, as users are expected to use only their desktop computer. Most systems even do not support logging in from multiple terminals, effectively assuming that each user only ever logs in to the service from a single location and computer. Therefore location awareness in current IMP offerings is not very sensible.

With a mobile device a user's location can be practically anywhere, and the user's activity can also be anything. No longer is it safe to assume that a logged-in user is sitting at their computer, typing away, available for anything. However, taking advantage of location information can also be a valuable addition to a system, e.g., when asking someone to pick up

something it is useful to know that they are close to the correct place.

IV. Conclusions

The current state of IMP systems is not a very optimistic one. Different service providers intentionally keep their protocols incompatible in an attempt to secure most users to their particular network. Third-party implementers and reverse engineers are not often tolerated, and they exist only because of the lack of energy to fight very aggressively.

The two areas of research, context-aware computing and ad hoc networks, may prove to be crucial to IMP systems of the future. After all, context awareness would allow the automatic determination of rich presence information, coupled with extensive rules that determine how the presence information is shared and how it interacts with the desires of others to send messages. Ad hoc networks will provide the capability to be constantly online, and to communicate directly in cases where feasible. This coming heterogeneity can be both a boon and a curse for IMP (and more generally, distributed) applications, as network connectivity can no longer be relied on, and the environment is constantly changing.

We suspect that the future of IMP continues to be dominated by the current systems, which are in essence designed only for the single desktop with good connectivity and a network infrastructure, and which do not provide very sophisticated presence management capabilities. Current research is still very preliminary, and typically only addresses a single issue. Furthermore, the visions of researchers on the future of computing may not coincide with the actuality of that future.

However, we believe that, as with many other areas, good ideas from research come to be integrated to existing systems at some point, assuming that the ideas are relevant then. The main problem with research is that the ideas are often not allowed to be implemented properly, which creates a gap between research prototypes and usable applications.

References

[1] H. Joe, “Nine im accounts and counting,” *ACM Queue*, vol. 1, no. 8, pp. 44–50, Nov. 2003.

[2] S. Tarkoma, R. Balu, J. Kangasharju, M. Komu, M. Kousa, T. Lindholm, M. Mkel, M. Saaresto, K. Slavov, and K. Raatikainen, “State of the art in enablers for applications in future mobile wireless internet,” Helsinki Institute for Information Technology, Helsinki, Finland, HIIT Publication 2004-2, Sept. 2004.

[3] M. Day, J. Rosenberg, and H. Sugano, *RFC 2778: A Model for Presence and Instant Messaging*, IETF, Feb. 2000, http://www.ietf.org/rfc/rfc2778.txt.

[4] M. Day, S. Aggarwal, G. Mohr, and J. Vincent, *RFC 2779: Instant Messaging / Presence Protocol Requirements*, IETF, Feb. 2000, http://www.ietf.org/rfc/rfc2779.txt.

[5] R. Want, A. Hopper, V. Falco, and J. Gibbons, “The active badge location system,” *ACM Transactions on Information Systems*, vol. 10, no. 1, pp. 91–102, Jan. 1992.

[6] E. Isaacs, A. Walendowski, and D. Ranganathan, “Hubbub: A sound-enhanced mobile instant messenger that supports awareness and opportunistic interactions,” in *Proceedings of the Conference on Computer-Human Interaction*, Apr. 2002, pp. 179–186.

[7] E. Isaacs, A. Walendowski, S. Whittaker, D. J. Schiano, and C. Kamm, “The character, functions, and styles of instant messaging in the workplace,” in *Proceedings of the 2002 ACM conference on Computer supported cooperative work*. ACM Press, Nov. 2002, pp. 11–20.

[8] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, and E. Schooler, *RFC 3261: SIP: Session Initiation Protocol*, Internet Engineering Task Force, June 2002. [Online]. Available: http://www.ietf.org/rfc/rfc3261.txt

[9] D. Greene and D. O’Mahony, “Instant messaging & presence management in mobile ad-hoc networks,” in *IEEE Annual Conference on Pervasive Computing and Communications Workshops*, Mar. 2004, pp. 55–59.

[10] P. Saint-Andre, *RFC 3920: Extensible Messaging and Presence Protocol (XMPP): Core*, Internet Engineering Task Force, Oct. 2004. [Online]. Available: http://www.ietf.org/rfc/rfc3920.txt

[11] A. Burak and T. Sharon, “Usage patterns of FriendZone — mobile location-based community services,” in *Proceedings of the 3rd International Conference on Mobile and Ubiquitous Multimedia*, Oct. 2004, pp. 93–100.

[12] A. K. Dey and G. D. Abowd, “Towards a better understanding of context and context-awareness,” College of Computing, Georgia Institute of Technology, Tech. Rep. GIT-GVU-99-22, 1999. [Online]. Available: ftp://ftp.cc.gatech.edu/pub/gvu/tr/1999/99-22.pdf

[13] A. K. Dey, D. Salber, and G. D. Abowd, “A Conceptual Framework and a Toolkit for Supporting the Rapid Prototyping of Context-Aware Applications,” *Human-Computer Interaction (HCI) Journal*, vol. 16, no. 2–4, pp. 97–166, 2001.

[14] J. E. Bardram and T. R. Hansen, “The AWARE architecture: Supporting context-mediated social awareness in mobile cooperation,” in *ACM Conference on Computer Supported Cooperative Work*, Nov. 2004, pp. 192–201.

[15] P. Riikonen, *Secure Internet Live Conferencing (SILC) Protocol Specification*, July 2003, [Internet Draft]http://www.ietf.org/internet-drafts/draft-riikonen-silc-spec-07.txt.

[16] T. Strang, C. Linnhoff-Popien, and K. Frank, “CoOL: A context ontology language to enable contextual interoperability,” in *Distributed Applications and Interoperable Systems*, ser. Lecture Notes in Computer Science, J.-B. Stefani, I. Demeure, and D. Hagimont, Eds., vol. 2893, Nov. 2003, pp. 236–247.

[17] M. Raento, A. Oulasvirta, R. Petit, and H. Toivonen, “ContextPhone: A prototyping platform for context-aware mobile applications,” *IEEE Pervasive Computing*, vol. 4, no. 2, pp. 51–59, April-June 2005.

[18] Y. Jung, P. Persson, and J. Blom, “DeDe: Design and evaluation of a context-enhanced mobile messaging system,” in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, Apr. 2005, pp. 351–360.