SimPhony: voice group communication

by

Vidya Lakshmipathy

B.S. Symbolic Systems
Stanford University, 2001

Submitted to the Program in Media Arts and Sciences,
Department of Architecture and Planning
in partial fulfillment of the requirements for the degree of

Master of Science in Media Arts and Sciences

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

February 2004

Author .... [illegible]
Program in Media Arts and Sciences,
Department of Architecture and Planning
February 1, 2004

Certified by .... [illegible]
Christopher Schmandt
Principal Research Scientist, MIT Media Laboratory
Thesis Supervisor

Accepted by .... [illegible]
Andrew Lippman
Chairman, Department Committee on Graduate Students

SimPhony: voice group communication

by

Vidya Lakshmipathy

Submitted to the Program in Media Arts and Sciences,
Department of Architecture and Planning
on August 15, 2003, in partial fulfillment of the
requirements for the degree of
Master of Science in Media Arts and Sciences

Abstract

Communication is vital in any workplace. However, as workers become less tied to their desktops and computers, the need to provide them with a flexible, easy to use, mobile method of communication becomes more necessary. This is particularly true in "non-traditional" workplaces like factories or hospitals. Cell phones, PDA's, and walkie-talkies provide the mobility and most are easy to use, however, they are not designed specifically with the workplace in mind and as a result, they do not adapt to a worker's changing environment.

The Simphony communication system is a mobile, voice-controlled, voice communication system built on the iPaq (or any similar PDA) designed specifically for distributed workgroups. It uses the 802.11b network to transmit either synchronous or asynchronous voice data depending on the worker's environment or preference. The system allows for one-to-one or one-to-many communication with voice instant messages or synchronous audio. Simphony transitions between different communication styles as the communication becomes more frequent. When at least 3 voice instant messages are exchanged between two individuals or between an individual and a group, the system automatically transitions them into a synchronous audio chat.

The Simphony interface looks much like an instant messaging client but is accessible by voice commands or by button presses on the PDA screen. Simphony allows users to define groups of individuals with whom they can communicate with simultaneously. When a group that a user is a member of becomes active, the user receives notification of the activity by hearing approximately 10 seconds of the audio from that group. If the user is currently in another conversation, she can decide to remain in her present conversation or switch to the newly active group.

This thesis describes the design and implementation of the Simphony system and its various applications in different areas.

Thesis Supervisor: Christopher Schmandt
Title: Principal Research Scientist, MIT Media Laboratory

Thesis Committee

Thesis Supervisor: [illegible]

Christopher M. Schmandt
Principal Research Scientist
MIT Media Laboratory

Thesis Reader: [illegible]

Thomas R. Gardos
Senior Research Scientist
Intel Corporation

Thesis Reader: [illegible]

Judith Donath
Assistant Professor of Media Arts & Sciences
MIT Media Laboratory

Acknowledgments

I chose to give my thanks today to all of the people who have made me who I am because I just happened to be wearing an MIT shirt bought nearly 10 years before I was born in 1970 by my dad. Between the two of us we've kept this shirt for 33 years and since I was a child, it has reminded me of a funny story.

When my father applied to graduate school in the United States, coming from one of the most prestigious engineering schools in India, the one thing he just could not get right was his name. As a South Indian, he was confused by the idea of a family name and a given name because as a South Indian male, he received two unique names of his own. When taking his GRE, he put down one name at random as his family name and when filling out his application, the other. The reason, he always told me, that he never got accepted into MIT was because they just could not find his GRE scores and he did not find out until too late. Dad, soon I will have a degree from MIT and if it had not been for your mistake, we might not have kept this tattered shirt for 33 years and I might never have wanted to right those old wrongs. This example is just one in a million of how much you and mom have sacrificed to get Naveen and I as far as we've come. I hope this degree does not signify a completion of an era but rather a beginning of a lifetime of success and achievements that you both can always be proud of.

Who knew that 33 years after the purchase of this one shirt that it would lead me to a place where I would find the greatest group of friends and colleagues to support me and entertain me through my education and my life. Without the members of the Speech Interface Group, Natalia, Stefan, Sean, Gerardo, Jang, and Matt Hofmann life at the MIT Media Lab would have been ... well ... unbearable. My officemates, Stefan Marti and everyone's favorite Cotton de Tulear, Nena, kept everyday fun (and as Stefan says "That's the reason we're all here at the lab.") and kept all my friends jealous of what great officemates I had! Natalia Marmasse, who will never let anyone believe that we saw a lion in the streets of Ft. Lauderdale, will forever be inspirational to me as both an excellent student and amazing human being (and friend). Sean, thanks for sharing the drama of my life. Who else will I procrastinate with so well? Matt Hofmann was a UROP in our group who worked with me for part of my first year and throughout my second year at the lab. Although he was only a freshman when he started, Matt is a truly self-motivated, intelligent young man without whom the SimPhony project, particularly the telephone interface, would have never come to be. Thank you, Matt, for all of your hard work and dedication. I have a feeling you will do well in this world and you most certainly deserve it.

For two years in Cambridge, I've kept the company of "the usual suspects." Thanks to Nina, Rick, Gaia, Puneet, and Nigel without whom life would not have come before work. Nina and Gaia, both my roommates in Ashdown house, have always been the type of driven, intelligent, but extremely caring women I've wanted to surround myself with. Nigel, I've always said you should be a co-author on my thesis and perhaps receive 1.5 degrees from MIT. Without you, I would have never survived the long nights, the endless debugging, the cryptic error messages, and the plentitude of life challenges you've helped me withstand. The pleasure of your company has always given the strength to be the person I've always imagined I could be - I hope to never lose it. You are truly the inspiration behind my recent courage and accomplishments.

Nav, if i'd waited till you finished proof reading my thesis, I may never have finished. Your writing has forever been something I have looked up to (with some annoyance since you are younger than me). I hope you consider this thesis worthy of the Lakshmipathy name and you outdo me as you usually do by writing something ten times better when it's your turn. Then i'll finally give you a jumping hug!

The SimPhony project would have never come about if it had not been for a few people at the Media Lab. Chris Schmandt, my advisor, being one of them. I read an excerpt from Chris' book as an undergraduate at Stanford and knew from then on that I would apply to his group at the lab. Thank you, Chris, for accepting me and giving me the chance to work on TalkBack and SimPhony. Tom Gardos, the Intel affiliate to the Media Lab, brought to our group the scenario of fab communication and has since encouraged me and funded my work (and inspired my vacations!). Judith Donath, head of the Lab's Sociable Media group also gave me important feedback and advice. I owe a lot to all of the readers of my thesis for being so flexible about my completion schedule. As an Intel Fellow, the SimPhony was funded by Intel Research. Thank you to all of you.

Chapter 1

Introduction

Whether it takes the form of formal meetings or informal hallway conversations, communication and coordination among members of a team is what allows work to get done and an end product to result. Studies show that informal group communication in the form of synchronous, face-to-face contact serves a variety of roles including coordination of tasks, collaboration, group building and social communication and depending on the job-type, this type of communication could take between 25% and 70% of the time spent in the workplace [1].

Work groups are now distributed over various locations and time zones. Some or all members might work remotely from home. Systems that try to support remote communication and opportunistic interactions have tried to address the need of workers to communicate with each other despite separations in space and time. Certain work situations require an unusually large degree of data transfer and coordination *between* co-workers in addition to the main set of tasks for each worker. These situations might occur in hospitals between doctors and nurses, in buildings between support staff or in factories on production lines. Clean rooms in semiconductor fabrication facilities (fabs) also fall in this category. In these, like many other situations, the task of each individual worker is a critical part of the workflow of the entire group. However, what distinguishes these situations is that the need for cooperation is also imperative. In these scenarios, collaborating with other individuals who are engaging in their own primary task is critical to one's individual progress. Collaboration becomes a task in itself, one that must be performed along with several others. These environments are also divided into time shifts in which a group of individuals work for up to 12 hours at a time and are then replaced by another group.

This thesis prototypes a communication tool which facilitates opportunistic interaction and communication among members of a distributed workgroup. Although the scope of the communication tool proposed in this thesis is actually much greater, and several user scenarios are given to highlight different features of the design, we use the small user group of fab manufacturing technicians (techs) to make specific assumptions about the design and get feedback about the system. The restrictions of their environment make them a particularly focused and interesting group on which to base specific design assumptions.

1.1 The Problem: Group Communication

As workers become more separated in space and time, the opportunity to engage in spontaneous discussion and collaboration greatly decreases. In some cases, even if workers are relatively close in proximity, the environment actually prevents or makes difficult this type of communication. As communication technologies have developed, there has been a lack of focus on technologies that help us remain aware of our friends', family members', and co-workers' activities and whereabouts because of the privacy problem that these technologies pose. Instant messaging clients, like ICQ [2], changed that by creating a notion of awareness which is closely tied to the frequency with which a person uses a computer. This awareness, presented through audio and visual alerts, allows a person to communicate with a "buddy" via the computer only when they are aware that the buddy is available on their own computer. As our work becomes more separated from a desktop computer, however, the same issue once again becomes a problem. How do you remain aware of someone's activities and availability without encroaching on their privacy and personal space?

Particularly in the work setting, workers that we spoke with were apprehensive of managers monitoring their activities and whereabouts, particularly their communication, for fear that what they might say to others in confidence might jeopardize their jobs. Privacy concerns are less of an issue when the users themselves specify when they are available for communication and actively dictate how and when they can communicate with others. The problem then becomes how to share that information with others. In environments like the fab, the notion of availability is a difficult one to convey because techs work in highly regulated "clean rooms," while managers and other personnel work in offices. Similarly, factory workers might be tied to their tools while their colleagues might be tied to desks. Doctors might have to do rounds in the hospital or a clinic, or be in surgery, while nurses and hospital staff tend to other patients or paperwork.

Communication in a fab is notoriously difficult because of additional restrictions posed by the clean room and by the environment and clothing. Clean rooms are outfitted with wired telephones near each tool set or group of similar tools located in proximity. When a tech wants to communicate with someone in another area or outside of the fab, he or she must page that individual on an alphanumeric pager, give the person their contact phone number and wait for the person to call them back at the phone near their tool. In the meantime, they have no indication of the whereabouts of that person or how long it will be before that person can call them back. If they have a question about their tool, the downtime might be valuable time wasted. This might delay the process at other tools and become very costly. Furthermore, while they are communicating, they are most probably a few meters from their station, preventing any further interaction with their tool. This hassle not only prevents spontaneous interaction but inhibits even necessary contact. The full body *"bunny suits"* that the techs are forced to wear also limit their movement, their ability to perform precision input, and their ability to clearly see everything around them. In addition, the yellow lighting and noise added by the tools might play a factor in any form of communication. Many materials, like paper, are restricted from the environment unless they undergo strict cleaning procedures. As a result, materials and data transfer also become exceedingly difficult in this already restrictive environment.

We've all been to hospitals where we hear doctors and nurses being paged over the intercom system or on personal pagers to call the nurses station or to report to a certain area of the hospital. When spread through a building or interacting with patients, hospital staff can be distributed and will never be able to be in all places they are needed. An intercom system allows for immediacy in communication but minimal privacy because not only does the doctor or nurse hear that they are being paged, so does everyone else in the building. While at times this might be helpful, at others it is annoying and inconsiderate. A system which allows members of a hospital team to contact one another directly, independent of location, might allow greater convenience, flexibility and privacy.

In an office or factory-type environment, support staff often wear walkie-talkies or other devices which allow them to connect directly to members of their group. Although these devices allow a group to communicate, independent of their location, in a flexible manner, these devices have no knowledge of the environment in which they are used and as a result must be used in the same way, independent of situation. If one person is trying to speak with another, he or she must ask on that channel whether the person is available or wait until he or she is. At this point, any number of people on the same frequency can listen to the conversation. If the speaker encounters a distraction, there is no way of notifying the other members or changing the way in which they interact with the other people.

None of these situations has a communication solution which takes advantage of the technology that is available today. All of these communication solutions can be improved with a tool which had the flexibility to allow them to multitask - that is, engage in their primary task and a communication task simultaneously. This thesis describes a tool which addresses a variety of these problems.

1.2 Related Work

Approaches to designing communication tools for work groups vary but there are some common design principles that might emerge to inform a group voice messaging and communication system. Ethnographic studies of workplaces often look for communication behavior and patterns of specific types of workgroups and design systems to address each group's needs. Systems like *Media spaces* [3] and *VideoWindow* [4] provide open synchronous audio and video links between two remote locations. Studies found that although both systems prompted brief social interactions, neither prompted the type of interaction that would have occurred with similar, face-to-face sightings. A comparison of video and audio conferencing systems also showed that visual cues provided by video conferencing systems improved the structure of conversation [5]. Although participants perceived the video as beneficial, as it created a seemingly richer interaction, this study showed that phenomenon like turn-taking can be maintained in an audio space without the assistance of video.

Other systems look to connect desktops to one or many other desktops [6] [7]. These systems improve the audio and video links by giving some awareness of other users by showing low resolution videos of each user and allowing the user to determine whether or not to communicate with an individual based on a "glance" into their environment [8]. Systems like *Cruiser*, which allow users to look for available colleagues, are inspired by social techniques like "cruising" an office to see who is available to motivate the design of a system that looks for available colleagues [9]. Evaluations of such systems play up the usefulness of rich media like audio and video and the awareness information these systems convey to others [10], because this type of information allows the user to make a more informed decision about how and when to communicate; however, they were still less likely to result in opportunistic interactions than hallway meetings.

Existing communication systems not specifically designed for work environments might be given to distributed workgroups to determine what types of behaviors the technology might support. Although text messaging and group chat applications in the workplace have yet to gain critical mass, they are a mechanism for opportunistic interactions which work well if users are able to overcome the assumption that those communication modalities are used specifically to "goof off" [11]. Systems, like *Hubbub*, which support awareness, opportunistic interactions and mobility show that the right type of notification of availability can have a lot of influence on encouraging opportunistic interactions. *Hubbub* supports sound icons that are triggered when *bubs* (people on the buddy list) become available or come "online". These individual icons not only give users some idea of who is available, they also allow the user to contact that person using a simple, lightweight interaction style. These findings support the idea that the type and amount of awareness should match the style of interaction needed for the specific workgroup. These sound icons and the lightweight communication mechanism work well for mobile users [12] because instead of passively indicating the state of a coworker, they push this information on the user reminding them of others' presence.

Cellular radios or mobile push-to-talk communication services are also changing the metaphors for mobile voice communication. The *Direct Connect* service by Nextel offers such a service nationwide to over 11 million customers and allows groups of up to 25 people communicate using a push-to-talk metaphor on their mobile phones [13]. This feature is becoming so popular that companies like Sprint are also offering similar services. Because of the many affordances provided by this medium, there are a wide variety of interaction styles and behaviors exhibited by users [14]. Unlike other messaging mediums (both text and audio), push-to-talk services not only provide a rich, cotemporaneous medium in which turn-taking is enforced by the technology (i.e. not a full-duplex channel), they also have low production, start-up, delay and reception costs experienced with other mediums. Evaluated against Clark and Brennan's framework [15], this medium provides the most affordances for communication which then result in more frequent copresent activity.

Communication systems that use voice only in environments where users are usually behind a desk or carry a mobile device are known to be useful because they allow for a large amount of multi-tasking and can be used in an eyes or hands free mode. Systems like *Thunderwire* and *Somewire* look at audio-only spaces, the types of interactions supported in these environments, and the types of interfaces needed to manage these spaces. Results of evaluating this space showed that a high quality audio space was a natural mode of interaction between participants. However, the system could have included some indication of who was present in the audio space and a simple way to move in and out of the audio space as the environment demanded it [16]. Users of this system felt that they would be more comfortable with conducting interactions over the audio space if they knew who was present.

Several audio systems designed at MIT show unique ways of grounding an interaction or allowing for a variety of styles of interaction using one system. The *Talking in Circles* system allows for a clever mix between a visual and audio interface. Participants in an audio chat are represented by a circle that they draw on the screen interface and move between conversations represented by groups of circles. The volume of participants that are farther away in the virtual space is reduced to provide audio parallelism of the visual environment. This unique mix of a visual and audio environment allows for peripheral spatial awareness of others in the space [17]. This type of feedback allows users to see activity in a space without having to listen to it. This makes data transferred using the system accessible through the visual or audio interfaces or both together. The *TattleTrail* audio chat system also developed at the MIT Media Lab uses the IP based network to store the audio chat and allows users to browse and catch up to audio conversations, much like they can with instant messaging histories [18]. Interfaces like this have shown that a GUI is not necessary to manage communication in an audio-only interface [19].

One of the first systems that tries to dynamically change an audio space in response to participant behavior, in order to more accurately model real group conversation is the *Mad Hatter* system [20]. This system tries to take into account the fact that the conversational floor changes frequently amongst members of a "gelled social group" (e.g. two participants split off to form a new conversation). By detecting different conversations and dynamically modifying the audio presented to the participants, making their current conversation more salient than the conversations of others, this system hopes to reinforce the dynamic configuration of conversations in audio spaces.

Communication interfaces already in use by particular workgroups, like the *Voice Loops* system, are interesting because they are developed internally by users a need driven basis. Not only does this make the interface specific to their work environment, but in many cases, users have developed cognitive models that allow them to interact on a very expert level with these systems. They become highly integral to the process of workflow and closely tied to the event-driven, hierarchical structure of the environment. NASA uses the *Voice Loops* system in air traffic and space mission control. Voice Loops are essentially open audio spaces in which different people interact and exchange information. The various loops are organized in a hierarchical structure, much like the teams at the control center. Engineers and technicians usually monitor more than one channel simultaneously and are able to extract the information relevant to them. This shows how communication systems that are so closely tied to work processes can become a part of the users' cognitive model [21].

Communication systems on the market today like the *Vocera Communication Badge* allow for distributed, wireless communication amongst two or more co-workers [22]. However, this system provides only a synchronous voice communication (along with some limited text messaging) and an interface to a PBX. Above this communication infrastructure, there exist few features which actually allow communication to adapt to workers needs in a busy environment. The system allows for location or role-based messaging, however this is the extent to which it supports the specific communication style of work groups. The system described in this thesis uses a platform similar to the *Vocera Communication Badge.* It integrates computer supported cooperative work (CSCW) approaches to distributed group communication, along with field research from the fab, to create a communication system that specifically allows small groups to have ongoing interactions in ways that support each individual's situational and informational needs.

Chapter 2

User Scenarios

The following chapter begins with an overview of SimPhony's features and goes on to discuss several user scenarios for the SimPhony system. Each highlights a different part of the system and hopes to show the various flexible ways one can use this group communication tool.

2.1 SimPhony Overview

The SimPhony client runs on a pocket PC or a telephone and connects to the SimPhony server which runs on a desktop or laptop computer. On a basic level, the system functions much like commercial instant messaging (IM) clients. It allows users to define a list of "buddies" or contacts with whom they would like to communicate. These buddies can be listed by nicknames or real names as defined by the user. Users receive audio notification when their buddies sign into the system and can also query whether their buddy is already communicating with someone else. Users can contact these buddies by sending a voice instant message, a short recorded voice message sent asynchronously to the other party, or by directly connecting in a synchronous voice over IP (VOIP) session.

What distinguishes the system from a commercial IM client, however, is that users can also define groups of buddies and have them listed on their buddy list under one name. When a user wants to communicate with this group, he only has to message this one name and all the members are included in the recipient list for voice messages or the contact list of the synchronous conversations. Each user in the group receives audio notification when there is activity in the group and can join in on any of the "conference calls". If a user is in a conversation with a buddy or a group and another group becomes active, the user receives an interrupt alert and hears 10 seconds of audio from the interrupting group. At this time, the user has the option switch to the interrupting group or stay with the current group. In addition, the system transitions users into different styles of communication based on their frequency of communication. If two users are sending voice messages back and forth frequently, the system will automatically transition them into a full-duplex synchronous conversation hoping to expedite their communication by using a more demanding but informative channel.

2.2 The "Fab"

Microchip fabrication plants (fabs) are characterized not by their factory setting but by the additional restrictions placed on the technicians because of the strict cleanroom protocols that they must follow. Each manufacturing technician (tech) wears a full body "bunny suit" with gloves and a head mask and must be extremely careful about the materials that he brings in and out of the fab. For the most part, techs attend to large tools which perform different processes on a wafer lot or set of chips. The tech's primary job is making sure the tools function properly and advancing the lots between tools. Much of the process is automated, however, when something goes wrong, that's when the tech steps in to fix the problem. The following is a hypothetical scenario based on observations made in a semiconductor fabrication facilityh.

Lane is a tech who has worked at the Intel fab in New Mexico for over 10 years. Over his career, he has seen his job become more focused around pinpointing and fixing tools in the cleanroom. During one of Lane's 24 hours shifts, a mission-critical application fails, and subsequently, this results in the failure of the automation system. Lane's first job is to find out whether this is an automation problem, a problem with the process that automates the movement of lots from one tool to the next, or a problem with the individual tool. To simplify this process, the SimPhony system allows him to conference with other techs in the fab at that time, or with the other techs at that tool set. He uses the command *"connect to ToolGroup"* and asks everyone at his tool set whether they are experiencing the same problem. Several people respond saying that they too are having similar problems. He discusses with them exactly what they are experiencing so he can later relay this to others. In a short time, they come to the conclusion that this is in fact an automation problem because nobody seems to be having specific problems with their tools. Without the SimPhony system, this might have taken them all day to figure out and communicate to one another by paging each other back and forth.

Lane then uses the command *"connect to ASC"* to report their problem to a member of the automation support center (ASC). Often the ASC personnel take reports from several techs and try to gather as much information as possible. While talking with Lane, Joan, a member of the ASC decides to conference with other techs as well. She uses the command *"connect to LithoToolSet"* to speak with all of the techs at Lane's tool set. With a more unified communication system, she can speak with several techs at once and get a more detailed description of the problem. She is unable to troubleshoot the problem on her own so she declares a code yellow (an alert to all of the fab to let them know that part of the manufacturing process has gone down). Normally, at this stage, Joan would need to send a text message to all automation personnel informing them that they need to dial into a phone bridge where they can discuss the problem and devise a solution. With the SimPhony system, she is able to *"connect to main bridge"* and speak with everyone at once. All personnel in that group will receive an audio alert indicating the group has become active and if they are in another conversation, they will be interrupted with 10 seconds of audio from the *main bridge* group and can switch to the group since it is important. Group members might be in the fab using a pocket PC or at a desk and receive a phone call connecting them to the active group. Joan can see which members are online and which are offline and know how many people should be present in the group.

Joan, the crisis manager in this situation, describes the problem and its impact to the members of the group. At this point, Matt, the manager on call (MOC) connects to Stefan, the fab sweep coordinator (FSC) and asks him to prepare a team to do a sweep of the fab. Since both Matt and Stefan heard the complete description of the problem from Joan, they are able to find the most skilled people in that area to make up the team. Stefan creates a sweep team of 4-5 members and forms a group using the SimPhony's screen interface. He then connects to the *sweep team* that he has just created to inform them of their tasks. Stefan assigns each member of the team to an area of the fab. The team gathers information at each of their areas. The members are constantly able to discuss issues amongst themselves by messaging the *sweep team* group or remaining connected in their ongoing conversation. Stefan is able to learn from these discussions and ask specific questions throughout the sweep without having to wait for the team to dial back onto the phone bridge that would normally be created for this communication task. Audio interruptions inform members who are communicating in another group about activity on that channel.

Periodically, Stefan connects to the *main bridge* group to update the participants of this group about the sweep. He receives interruptions from the *sweep team* group if he is needed in that conference. Simultaneously, Matt joins a group of *production managers* who also want to be updated about the sweep. He too is still monitoring the *main bridge* group and receiving interruptions when activity occurs in the group. Finally, the *sweep team* is able to solve the problem and Stefan asks them to process the wafers on the tools again. If anyone on the *main bridge* has any further questions or requests, they can easily ask Matt or Stefan directly to relay this to the *sweep team*. This pass down of instructions can occur almost instantaneously using the SimPhony communication system and does not require a complicated series of paging and waiting to receive phone calls.

At this point, the *sweep team* reports a few remaining problems to Stefan. Stefan is not be able to solve the problem himself so he uses the command *"connect to lithoexpert"*, a person he has nicknamed because of his expertise in troubleshooting automation at the lithography machines, to help solve the problem. Stefan reports these remaining issues back to the *main bridge.* Finally, the issues are resolved, and the *sweep team* makes a final sweep of the fab. The sweep is successful so the groups go idle once more until the next problem arises.

The advantage of using the SimPhony system is that communication that normally takes several minutes to perform can occur quickly, with no waiting time. The process of paging someone and waiting for them to call you back has suddenly been reduced to a one step process of connecting to that individual or leaving them a detailed voice message which they can respond to at their convenience. In addition, groups of people can communicate either constantly or intermittently over long periods of time without having to play "phone tag" to get in touch. Liaisons between groups can monitor the activity of several groups and have high-level awareness of when the groups are active and how much communication is occurring. Individuals can join a group and reach several people at once instead of having to disseminate information one at a time or through phone bridges where it is never completely clear who is present.

What distinguishes this system from others coming on the market is the simultaneous management of several channels at once. Most systems try to emulate phone conversations and might allow users to make three-way conference calls or switch between two calls. This system tries to give users a better way to manage a conference of several people by giving more awareness of who is involved in a group, whether there is activity in a channel, and whether or not group members are actively participating. The value of SimPhony is this: while technology might allow people to communicate over remote distances and time, it cannot teach people to communicate in different ways. This system gives users more awareness tools to manage group communication while allowing that communication to occur in ways that they are used to.

2.3 The Hospital

Hospitals are another place where teams are distributed in location but often need to coordinate and communicate with one another to get the job done. Nurses and doctors have very well-planned rounds and shifts of tasks, particularly in more critical patient care areas. Sharing information between shifts and between nurses and doctors working in the same area and with the same patients is critical. Communicating information in a timely manner between caretakers can be the difference between mediocre and great care, and in some patients, this makes a lot of difference.

Nurse Jones is one of the nurses who works the night shift in the pediatric ward four days a week. Her duties over the course of the night are diverse and sometimes difficult but often require alerting a doctor of a patients needs and coordinating with other nurses so that the entire ward gets the best care. Children as young as one year to 15 or 16 years in varying levels of recovery require a great deal of attention. Rarely is she in front of a computer or near an accessible phone and yet her promptness is often critical. Being there when a patient needs her, particularly those who are too young to page her when they need her attention, is what makes her a good nurse.

On one particular Wednesday night, Kyle, one of the babies on her rounds, was running a high fever and having trouble sleeping. Nurse Jones spent as much time as she could trying to calm the child down but she had several other children to look after and give their final dose of medicine. Finally, the child fell into a fitful sleep. At this point, using the SimPhony system, she sent a message to the group *WedNtStaff* asking them too keep an eye on this child as they were doing their rounds. Several nurses in this group who were discussing another patient at the time responded, agreeing to keep an eye on Kyle.

An hour later, Nurse Jones was still helping some of her other patients when one of the other nurses, Nurse Linst, established a connection to her using the SimPhony system and alerted her that Kyle was crying again. Nurse Jones had her hands full at the moment, so she used a voice command to add the *WedNtStaff* to her current conversation with Nurse Linst. She asked whether anyone could look in on Kyle until she could get there. Nurse Johnson was currently in the room next door to Kyle. Although she was in a conversation with a doctor when Nurse Jones send her message, she briefly heard Nurse Jones asking for help as part of the short alert she received as part of the *WedNtStaff*. The SimPhony system alerts all members of a group when there is activity in a group to which they belong, even if they are in another conversation. The notification consists of a brief snippet of audio from the conversation itself. Often, this is enough for the person to decide whether they want to switch to this other group or not. As Nurse Johnson was only consulting about medication with the doctor, she was able to walk next door and look after Kyle until Nurse Jones could make it back. As soon as she had ended her conversation with the doctor, Nurse Johnson switched to the *WedNtStaff* group and let them know that she was looking after Kyle so that the rest of them could continue with their own patients. Because there is no convention for acknowledging that you have received a message in the SimPhony system, nurses make sure they are explicit about all of their actions so that others know that there message was received.

SimPhony's ability to allow group conferences or conversations between individuals, and to transition easily between the two, allows somewhat dynamic configuration of the nurses' duties and practices. This enables them to help as many patients as possible. It gives them eyes and ears in all parts of the hospital ward, which makes their job easier and allows them to be more successful at it.

2.4 The Office

Most modern office buildings have maintenance, security teams, IT personnel or systems administrators in charge of building facilities, safety, or troubleshooting computer and network problems. Many of these teams are structured in much the same way as hospital or fab workers. These teams of three or more people are scattered throughout a building - often away from their desks - helping others. The nature of their work, means that these personnel must often consult with their team for advice. Because of this need, it is helpful for them to carry communication devices that allow them to discuss issues or find one another when necessary. The facility team at our own building in the Media Lab carries walkie-talkies to help them do just this.

The beginning of the academic year is never an easy time for the building's system administrators. This year, there are three of them in charge of setting up new computers and registering accounts for a batch of 35 new students. Although there is an order to the system, there are always exceptions to the rule. Some students bring their own laptops to be registered and have questions about installing software and connecting from home. Some need fixed IP addresses and specific configurations on their machines. Some do not know the first thing about their computers and want someone to help them understand it. The list goes on.

As the staff answers requests and moves from office to office, they can use the SimPhony system to ask questions of one another and, if they want, hear questions and answers being exchanged by other members of the group. They have the flexibility to have private conversations, send personal or group messages or have open group discussions. As a problem arises in office 357, Marcy, one of the facilities staff, calls her colleague Hillary (wherever she might be) to figure out how to configure a new student's machine so that she can access her home directory from abroad. She was unfamiliar with the configuration for the new modem drivers for Windows XP and she knows that Hillary was recently upgrading several machines to XP.

Marcy sends Hillary a message using the SimPhony system. Since Hillary is carrying her PDA, she can access her messages whenever she is free and not just whenever she is at her desk. Hillary is with a student when she receives Marcy's message, so the request has to wait a few minutes. When she has a chance, she responds to Marcy's message with another message. She thinks she knows the answer but is not sure. She explains briefly in the message. Marcy receives that message right away and responds quickly to get clarification. At this point, since both of them have exchanged rich voice instant messages quickly, the system assumes that they are both available for communication and automatically upgrades them to a full-duplex audio session. They are chatting about this matter when Hillary spots Ed walking by. She remembers that he was struggling with a similar problem last year. She calls out for him and he stops to see how he can help. As she is talking to him, she presses the down arrow on her PDA and downgrades her conversation with Marcy back to a voice instant message.

Marcy hears another voice in her conversation with Hillary and is not surprised that she then hears the beep signifying that she is once again recording a message for Hillary as opposed to talking directly to her. She assumes that Hillary must have received another request and picks up some work hoping that Hillary can later help her find a solution to her current configuration problem. Several minutes later, she receives a request for connection from Hillary. Hillary informs her of Ed's suggestion. Marcy tries it and it seems to work. She emails the student telling her that her machine is ready and moves on to her next challenge.

With the SimPhony system, she was able to fix this problem within the hour without having to wait for Hillary to check her mail and perhaps missing Ed's input all together. Because of the convenience and flexibility of the system, she was able to communicate with others without inconveniencing them.

Chapter 3

User Interface

The primary motivation in designing the user interface of the SimPhony system was to create a flexible, simple to use, yet powerful communication system for distributed workgroups, one that would not lack any of the conveniences and transparency of the systems currently in use. Furthermore, by providing location independence, multiple modalities, multiple metaphors for communication and the ability to manage several audio channels almost simultaneously, SimPhony would ultimately provide a more useful system.

The system allows for distributed one-to-one and one-to-many communication in a variety of styles from voice instant messaging to full duplex audio. It allows users to create lists of "buddies" to whom they have quick communication access and a higher level of communication awareness through the system. The system can be accessed by three interfaces, two on the pocket PC and one on a regular telephone or cellular phone. The primary motivation for such a design is because some members of a group might actually be behind a desk or on the road and find it more convenient to access the system on a periodic basis over the phone. A group can actually consist of several telephone users (as a call center might be), and connecting to that group might simply alert the first available member of the group instead of having to access one at a time. The flexibility of providing one relatively new modality for access, the pocket PC, and one older modality for access, the phone, allows for varying construction of the group hierarchy and poses several different design challenges, some of which are addressed

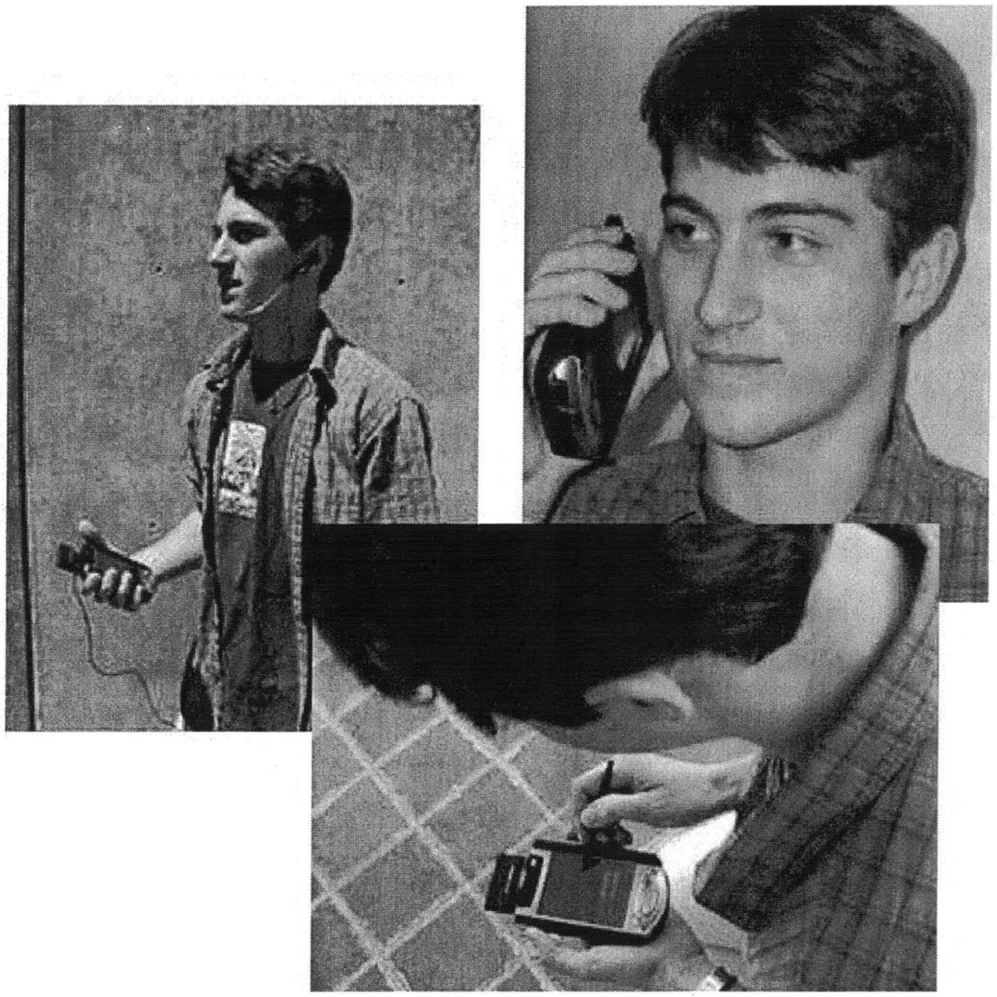


Figure 3-1: User using the SimPhony system in three modes: voice commands only, screen interface and telephone interface.

in this system and some of which are discussed in Chapter 6.

3.1 Audio

Many of the workgroups described in the previous chapter operate in an environment of constant challenges and interruptions. Members are often in transition between one location and the next, or engaged in a task which requires a high amount of attention or, at the very least, both hands. A voice-controlled interface works well for these types of environments because it allows the user to multitask by using their voice to control the system while keeping their hands and eyes on their primary task. There are many considerations that need to be taken into account when designing a speech interface for such a communication system. I will discuss several challenges that we encountered during the design of the SimPhony system and justify and motivate our solutions.

Because we use speech to communicate with others, it is important when designing a speech interface to make sure the interface is aware of the difference between commands issued to the interface and those not. A common solution to such a problem is to have a push-to-talk button much like a walkie-talkie. Pressing this button signifies to the interface that it is either being issued a command or to transmit the audio while the button is pressed. The trouble with a voice-controlled, voice communication system, however, is in differentiating speech used for system commands and those intended as communication to others. Visual feedback was not an option because it would require that the user look at the screen to understand what state the system was in. As a result, the solution had to be either audio or tactile feedback. Since all of the commands were short, we chose to make the command mode push (and hold) to talk. To issue commands to the system, the user must push the button on the side of the iPAQ and hold it as she speaks the command. Releasing the button signals the system to process the command and results in either audio feedback that the appropriate action has been completed or in an error audio icon indicating a false recognition by the speech recognizer or a command that was unable to be processed in the user's current state (i.e. logging in when the user is already logged in). When sending voice to communicate or record a message for a buddy, the user first receives audio feedback that the system is ready to record and instead of pushing and holding the button as she talks, the user simply pushes the button once and the system begins recording and/or transmitting. This metaphor resembles the push-and-lock feature of some buttons. The recording state is then toggled. One additional press ends the recording. In the case of the voice message, this completion of recording also sends the message to its recipient. In the case of a full-duplex connection, it simply ends the transmission and the user must press again to being transmitting once more.

The functionality which is accessible via the speech interface is somewhat restricted to improve the accuracy of the speech recognizer. The most important functionality, that of the messaging, is what is best done using the voice commands. There are six main commands that control the system functionality. They are *login, logout, voice message <name>, connect to <name>, listen to messages, add <name>*. Names of all the users were also added to the speech recognition vocabulary. Nicknames for any of the users can also easily be added to the speech recognition vocabulary. At this stage of the prototype, this requires manually adding the nickname to the vocabulary, however, this can also be added as functionality to the SimPhony interface. A user can enter his or her own nicknames for another user and and the recognizer might either load this user's individual vocabulary or simply have redundant names for several of the users.

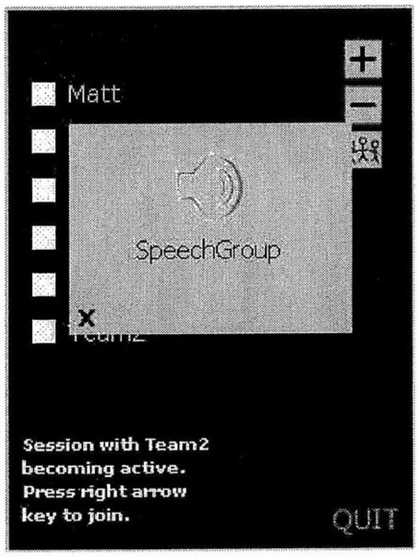
Finally, voice, the most important data type, is used as the primary communication mechanism in the SimPhony system. In an asynchronous, voice instant messaging mode, a user records a message for another user and upon completion of that recording sends it to the other user. At its most extreme case, this type of message might be similar to voicemail where the delay could be up to a few days or more. If a user logs out before listening to a message or is logged out when the message is sent, he or she will have to access the messages through the message archive. The message will most likely be "semi-synchronous" where the delay is small and the recipient of the message listens to it immediately upon arrival (making the delay equal the time of composition plus the time before the recipient listens to the message). The synchronous mode, however allows users to talk in a full-duplex mode (much like a telephone conversation). Voice instant messaging is convenient, particularly for people who may be busy, because it gives the recipient notification that a message has been received but allows them to listen to it when they choose.

3.1.1 Auditory Feedback

Because the majority of SimPhony's features are usable in a voice-only mode, it is important to provide the user with auditory feedback to reinforce and aid their actions.

Many of the audio cues given to the user represent the state of the system. When the user logs in and out, there are complimentary audio cues which tell them their action was successful. When another user logs in, the buddies on their list receive an alert indicating that a user has logged in. Individual alerts can also be substituted so that the user knows, without looking at the screen, who the new user is. When a group session becomes active or a voice message is received, different activity indicating tones alert the user of the new message. Although the current version of the SimPhony system uses standardized audio cues for group activity and new messages, many of these tones can be individualized by the user so that the user has a better idea of who is trying to communicate with him.

If a user is currently in a chat and another group becomes active or another buddy tries to chat with him, he will be interrupted with 10 seconds of audio from the interrupting session, during which time he can switch over to the new session or stay in his current session. This "preview" of the newly active session is meant to serve as a topic indicator. If the user is more interested in the interrupting session after hearing briefly what is being discussed, she can choose to tune in to the new session. This preview is much like hearing a conversation as its members pass by. When the speakers are in the range of the listener, she can interject or tune into the conversation. Once the speakers have passed by, they are no longer in range and neither the listener nor the speakers can communicate. If 10 seconds is too much of



Session with Team2 becoming active. Press right arrow key to join.

Figure 3-2: Currently in a group session when someone else interrupts.

an interrupt for the user, he can configure the system to have a shorter preview (or no preview at all). If the user does not want to hear the preview, he can cancel the preview and go back to his current conversation. Alternatives to playing actual audio from a new conversation might be playing a small alert, perhaps hearing muffled voices in the background to indicate that a group is becoming active. This way the user can excuse himself and switch to the other conversation if necessary. When a session interrupts, the user receives an audio tone and a message on the bottom of the screen shown in figure 3-2 indicating who has interrupted and that she can press the right arrow key to switch to the interrupting conversation. After 10 seconds, if the user does nothing, she will come back to her original conversation. If she decides to switch, she can press the right arrow key and join the interrupting conversation and her screen reflects that change.

3.2 Visual

Although the visual interface for SimPhony might be used less frequently, it is equally important because it allows users to perform some of the high level organizational

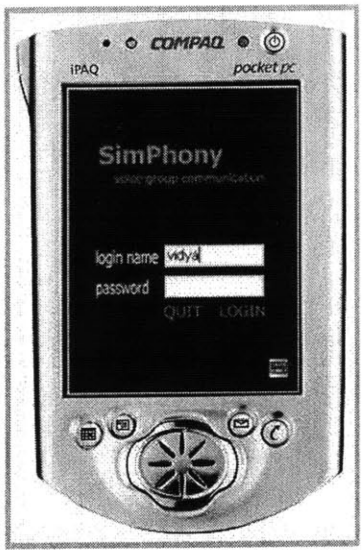


Figure 3-3: Login screen.

and navigational tasks inherent in a complex system. The visual interface looks much like those of today's commercially available instant messaging clients. Users log in on the login screen as shown in figure 3-3 and when the system registers them as logged on, they see their "buddy lists". A buddy list is a list of people also using the system to whom you have a "short cut". Other users who are online or logged on to the system are shown in white; users who are offline are shown in grey. Groups in addition to individuals, can also be listed on the buddy list. Although the current interface does not display the names of the individuals who make up the group, future versions may easily have this accessible to users. Like individuals, groups which are currently active, meaning that one or more individual is sending a message to or chatting in that group, are also shown in white. When a user logs in, not only does their name change color, but an audio icon appears, indicating their presence. This fairly standard practice allows users to distinguish between online and offline users and alert others to new users. This practice promotes spontaneous collaboration (much like meeting in the hallway). More sophisticated versions of the same client might have separate, personalized audio icons to indicate the presence of each user or might have a time stamp showing the amount of time a user has been logged

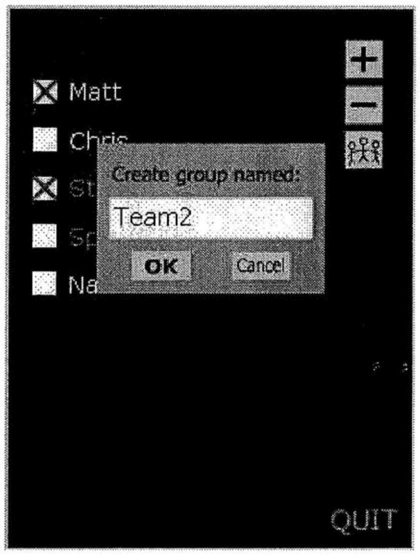


Figure 3-4: Creating a new group.

in or the time since their most recent conversation to give others some idea of how busy their buddy is. These subtle indications of presence hopefully promote smarter communication and collaboration.

From the main screen, users can perform several actions or configurations. To create a group, a user can select the check box next to the names of as many buddies as they would like and then press the group icon (the third button down on the right side). A box shown in figure 3-4 comes up asking them to name the group and that group not only gets added to the database of users but also the current users buddy list. At this stage, the name needs to be manually added to the grammar for the speech recognizer but this integration can be done automatically as well. A further step in this direction might allow users to define groups using voice commands, however, the complexity with a feature like this would be in correctly recognizing the group name said by the user, entering it into the grammar, and having it subsequently correctly recognized. This might require an additional confirmation step between the user and the recognizer, perhaps having the recognizer repeat or synthesize what it heard and repeat it back to the user and allow the user to confirm or change what was recognized.

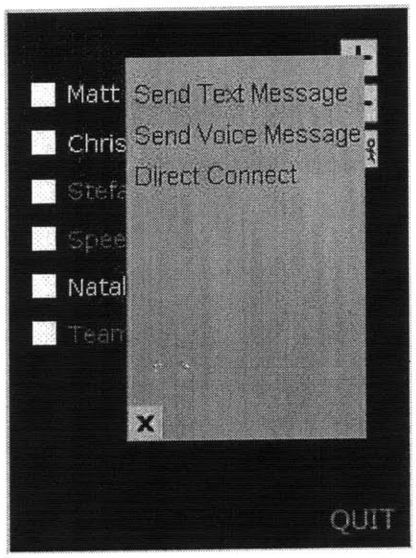


Figure 3-5: Using screen menus to send a message to a buddy.

Users can also use the screen menus to perform all of the same actions they can using voice commands. Clicking on a buddy's name pops up a menu shown in figure 3-5 which allows the user to text, voice message or directly connect to another buddy or a group. When recording a message or synchronously chatting with a buddy/group, the screen reflects the active state by showing a box with the name of the person or group. See figure 3-6. Beginning and ending recording is indicated with a tone in the case of a voice message and with a flashing LED on the iPAQ. In the case of a synchronous connection, a tone is only used when the session begins and ends but transmission is indicated by the same flashing LED.

When a user receives a voice message, a message box pops up with the name of the sender. Clicking the audio icon on the box shown in figure 3-7 plays the voice message. Alternatively, using the voice command *"listen to messages"* will accomplish the same task.

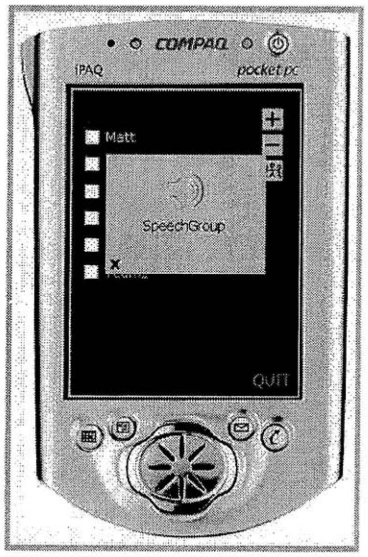


Figure 3-6: Currently in a group chat.

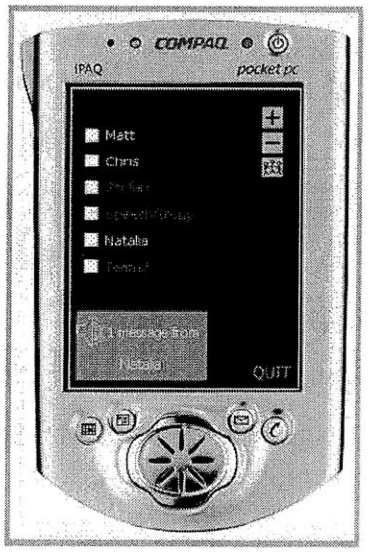


Figure 3-7: Receiving a voice message.

3.3 Telephone

The telephone interface to the SimPhony system is a slightly simpler interface which allows only the most basic but vital functionality. The telephone interface was designed to allow people who are in front of a desk and phone or on the road and only accessible by mobile phone to have access to those using an iPAQ and the SimPhony system. From any telephone, users can dial into the SimPhony system and use voice commands to navigate through the menus and prompts. Unlike many current telephony applications, SimPhony allows users to use the same voice commands on their pocket PC and over the phone. This allows expert users to navigate quickly through the system without waiting to hear the menus. Redundancy is created by allowing touch tones in addition to voice commands to navigate through the system - this prevents any frustration with voice recognition that might occur in noisy environments.

They specify their name to login, and use the same commands used with the pocket PCs to navigate the systems functionalities. From the main menu, they can *voice message <name>, connect to <name>, or listen to messages.* As with the pocket PC client, they can record a message for any other user using the voice message command and they can synchronously connect to any other online user or group with the connect to command. Listen to messages allows them to listen to any messages that have been recorded for them while they were online or offline. Further development will allow groups and other users to place calls to clients normally connected over the phone if a message is urgent. The client will receive a phone call which automatically connects them to the group which tried to reach them. If the user is unavailable, one of the members of the conversation can leave a voicemail message for her.

Chapter 4

Architecture

Because of the distributed nature of the SimPhony project, the various aspects are all separate but interdependent entities. SimPhony primarily acts as a connector between individuals who connect either using a pocket PC or a phone. The intelligent part of the system allows for the connection to take many forms based on the restrictions of the client, the needs of the client, and the communication behavior of the client. Most of the behavior of the system is controlled and maintained by the server while the clients deal with the audio recording and playback and maintain the interface between the system and its users. A communication system like SimPhony might someday interact with other applications on the device and allow users to send and receive or share documents, spreadsheets, or other data from other users. Until then, the pocket PC is used for its processing power, the high power microphone and speaker, and its ability to work on the wireless network and interact with other machines on this network.

4.1 Hardware and Software Requirements

The current implementation of the SimPhony system uses one Compaq iPAQ pocket PC h3950 and one Compaq iPAQ pocket PC h3970. The difference between these two is only in the amount of ROM available is not pertinent to this project. The system also ran on a Toshiba e740 pocket PC. All three of these devices were running

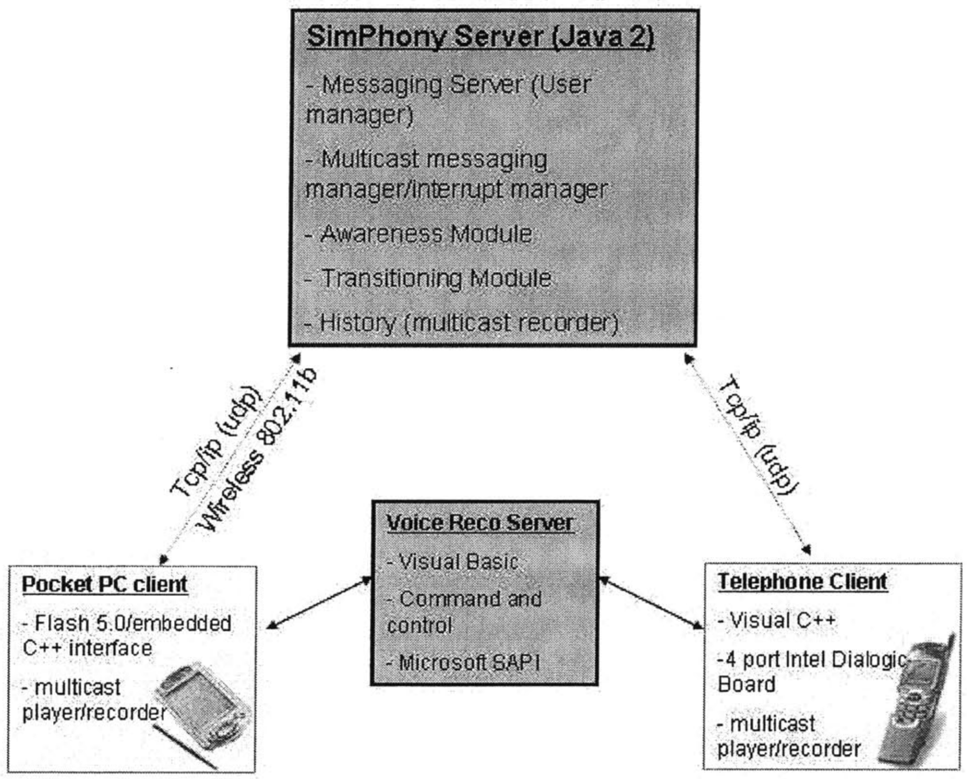

SimPhony Server (Java 2)
- Messaging Server (User manager)
- Multicast messaging manager/interrupt manager
- Awareness Module
- Transitioning Module
- History (multicast recorder)
Tcp/ip (udp)
Wireless 802.11b
Tcp/ip (udp)
Voice Reco Server
- Visual Basic
- Command and control
- Microsoft SAPI
Pocket PC client
- Flash 5.0/embedded C++ interface
- multicast player/recorder
Telephone Client
- Visual C++
-4 port Intel Dialogic Board
- multicast player/recorder

Figure 4-1: System diagram of the SimPhony system.

Windows CE 3.0 and pocket PC 2002 along with Flash Player 5.0 for pocket PCs. The interface was written using Flash 5 and is part of a larger embedded C++ application running on the client machines. Software created by Ant Mobile Software (Flash Assist Pro v.1.25) made it possible to compile a Flash swf file into a embedded Visual C++ project and create an installable pocket PC application.

The server runs on a Windows 2000 desktop (or laptop) machine running Java 2 (j2sdk1.4.0) in the C:/simphony/server directory where all of the recorded files and messages are stored for later access. The clients and the server use TCP sockets to communicate information between one another.

The voice recognition server is a Visual Basic application which uses Microsoft Speech (SAPI 3.0) for recognition. The recognition server sets up a UDP socket connection to the client on which it receives the voice commands.

The telephone client uses the same server and voice recognition server as the pocket PC clients, however, it uses an Intel Dialogic D/41JCT-LS 4-port board to create a gateway between the phone and a telephone client application, written in Visual C++, running on the machine containing the board.

4.2 Modules

The client and server software have all been decomposed into modules for easy development and testing. Each software module depends on a main messaging platform created for the project but could easily be based on a more robust open source messaging platform, like Jabber.

Each of the modules approximately corresponds to a class on the client side or a Java object on the server side. The main classes and objects will be discussed in turn.

4.2.1 Multicast Messaging

IP multicast is a bandwidth efficient way to stream data to a large group at the same time. This fire and forget protocol does not allow for a back-channel or any performance feedback, however it is scalable and works well for streaming media and delivering time sensitive material to large groups simultaneously. Unfortunately, there are many different protocols currently implemented by different routers and as a result, performance varies by router.

The decision to use IP multicast was primarily driven by the motivation to allow one-to-many voice communications in a way that phones cannot provide. Streaming IP multicast requires that UDP protocol is used to exchange packets. Although this is an inherently lossy protocol, we chose to explore the different ways in which we could improve the quality of the audio received and test whether this project proposed a feasible architecture. Previous projects in the Speech Group at the MIT Media Lab have successfully transmitted voice over IP (VOIP) using UDP multicast protocol, however, as traffic on the wireless network increases, the chance that the quality of VOIP is still acceptable decreases. Multicast may not be the best protocol to use for one-to-one voice messaging, particularly because delay is not an issue. For more robust models, we might try replacing multicast in voice instant messaging scenarios with RTP (real-time transport protocol), which allows for services such as payload type identification, sequence numbering, time-stamping, and delivery monitoring to real-time applications [23].

The Server

Several classes make up the multicast messaging aspect of the SimPhony system. The most important is the SimPhony Messaging Server (SimPhonyMessagingServer.java). This object creates and maintains a hashtable of all of the active users. When a user logs in to the system, this class retrieves her information from a user log file (or perhaps someday a more sophisticated database) and stores this information in a user profile object (user.java). The user profile maintains several fields like the users' buddies and current IP address or phone number and currently unused fields such as the users location and nicknames. The user log file is currently the only way to add or remove buddies from one's buddy list or in any way change the members of a group. Most of this functionality is standard in commercial instant messaging clients and did not need to be recreated for a research prototype.

The messaging server also controls the rest of the commands available to the user. When sending a voice command to the server, the messaging server creates a multicast server (multicastServer.java) object which randomly generates a multicast ip address and port number which it then connects to and sends back to the client to connect to. At this point, any data sent to that address and port number by the client is recorded by the server. Although multicast is not needed when sending messages only to the server, it might server to increase robustness of the system if there was always a back up server, listening to commands and making recordings to assist the main server in case of failure. Using a more reliable mechanism of transport, like RTP, might bypass the need for redundancy. When making a synchronous connection to another user, again a multicast server object is created but this time the address and port generated are sent not only back to the client but also the the person or group to whom she is connecting. Each individual client then connects to this address and receives the data being transmitted. The clients play the audio data and the server, which also remains part of the session records all the data for any members who are unable to listen during the broadcast.

When a client finishes recording a voice message, the messaging server then notifies the recipient of the message that he has received a message and gives him the ID number of the message. Each recording made by the server is labelled with the sender and recipients (or participants) names and the port number of the multicast session. The port number is used to retrieve the voice message when the recipient is online and receives notification of the message soon after it's been recorded. When a recipient is offline when the message is recorded and wants to check messages at a later time or listen to the history of a session, he can search the directory of recordings for all recordings with his user name in the filename. Listening to a message creates a multicast player (multicastPlayer.java) object which opens the specified file, informs the client of the multicast address and port at which that message will be broadcast, and then plays the message.

The Client

On the client side, there are multicast and multicast manager classes (multicast.cpp, multicastManager.cpp) which manage the connections with the multicast address coming from the server. The client is constantly in a message loop in which it is sending data to the server and receiving a response from the server. When the client wants to send a voice message or synchronous connection with another buddy or a group, the server sends back a multicast address and port and the client then connects to that address and informs the client that it is ready to send data. When the user is finished recording or finished with the session, the client terminates the connection to the multicast address. If the user is receiving a voice message, the server again tells the client which address to connect to, the client connects and when the messages are finished playing, the client disconnects from the address. The multicast manager keeps track of the data about the current session, what type of a session it is (voice message or synchronous audio), and if there is an interrupting session manages the transition between the two.

4.2.2 Awareness

The concept of awareness with regards to the SimPhony system is one that is limited to the user's communication behaviors using the system. Although many messaging clients try to include some high level information about whether a user is at their desk or away from the keyboard, the SimPhony system is presently only looking at communication activity to discern availability. This is relatively small scale in comparison with systems which focus on awareness and discerning availability as key components, however, within the context of this project and the user group involved, it seemed sufficient when looking at distributed work groups in a work setting.

The Server

Because the server keeps track of who is online and who is in a conversation and how often they transmit data, the server notifies other clients when one of their buddies becomes more active. When a user logs in, the server broadcasts a message to all other clients notifying them that the current user has gone online. If that user is part of their buddy list, the interface then updates to show the change of status of that user. If a user sends a message or is transmitting to a group, that group appears active on the other client's screens. The current SimPhony interface only distinguishes between online and offline users; however, one might imagine that if a user is currently sending a message or talking to another user, that user's name could also look slightly different, indicating that that user is currently in a conversation.

The Client

The client communicates user status in two ways, first with an audio icon and secondly with a change of the color of that users name on the screen. When a user logs in, an audio icon is played indicating a login and that user's name changed color on the screen. A separate audio icon can also be used for each user creating personalized alerts for each user.

4.2.3 Transitioning

One of the key behaviors of the SimPhony system which sets it apart from other voice messaging systems is the concept of transitioning between styles of communication.

The Server

The SimPhony system tries to find the most efficient method for communication between two individuals by monitoring the frequency of their communication and changing they style accordingly. This behavior is governed by the transition manager (transitionManager.java). This object is instantiated when one user sends a voice message to another. The transition manager records the time that message was sent and notes the sender and receiver of the message. It notes when the recipient listens to the message and when the recipient replies to the sender. Although the system does not notify the user that her message has been heard by the message recipient, it could easily send a notification back to the message sender indicating such. Because it notes every message sent back and forth, it is also able to compare the time between exchanges. If a user sends a message to a buddy and that buddy responds quickly, the transition manager assumes that they are both available for a more synchronous style of communication. When the original sender (the user) requests to send a message the second time, if this request is less than 2 minutes later than the time the original message was sent, the transition manager automatically makes the request a request for a synchronous conversation as opposed to another voice message.

Obviously, this assumption is not always correct so there are two methods in place to prevent this behavior from becoming too annoying. The first is the ability to configure this behavior for different individuals. For some, you might want to increase the number of messages exchanged or the amount of time elapsed before the system automatically transitions to synchronous voice. It is simple to add these rules to the transitionManager class. By checking for a specific user name and associating that user with either the time delay or the number of messages exchanged before the transition, a user can configure to transitions to her preference. Secondly, if the transition takes place and the user does not want to have a full-duplex conversation, she can press the back arrow on the iPAQ and the session is ended. She can easily send a voice message at this stage, without fear that the system will transition her again into a synchronous voice session.

4.2.4 History

This aspect of the SimPhony system is one that might raise concerns of privacy and monitoring particularly among small work groups in a work environment. Recording conversations between members of a team can be a benefit or a way to monitor workers and comment on their productivity. Fab workers have mixed ideas when it comes to having their conversations recorded. When used as a tool by others to find out about something work related or understand a process or troubleshooting procedure then having those conversations recorded is a boon. However, if used by the company to monitor their work styles and productivity, the practice of recording conversations is discouraged.

The Server

In the current version of the SimPhony system, the server records all voice messages and conversations within groups and between individuals. Unless the voice message is delivered while the recipient is online, there is currently no way for the user to query his or her messages from the server using the pocket PC client. However, the messages on the server are stored by type (voice message or conversation), recipient and sender's name, and port number of the multicast session. To query messages, one would simply need to do a search of the server directory and find any messages with his or her buddy name in the filename and play that file. The telephone client allows users to listen to all of the messages recorded for them since they were last online. This feature could easily be integrated into the pocket PC client. Future versions of the SimPhony system will have more control over the recordings, allowing participants in a conversation to decide beforehand whether a conversation will be logged or not.

4.2.5 Voice Control

The Voice Recognition Server

The voice recognition server is written in visual basic and uses the Microsoft Speech API (SAPI 3.0). The server was adapted from a voice recognition server example that came with SAPI 3.0 to include a socket server which waits for an incoming connection from a client and then streams the data to the recognizer and stores the data in a memory buffer until the client signals the end of the stream. The recognizer then forms a hypothesis for each word and when it completes the recognition, it sends back a string of what it recognized to the client, at which point the connection between the client and server is closed. At each new voice command, a new connection is made and the process occurs again.

The voice recognition server is command and control for a small vocabulary of six commands and six names at present. The vocabulary is a simple grammar which looks for a command and if the command takes an argument, usually the name of a buddy, the recognizer looks at the word following the command to find the buddy or user name. The grammar has several entries similar to:

```
<RULE NAME="sendvim" TOPLEVEL="ACTIVE">
    <P PROPNAME="request" VALSTR="sendvim">voice message</P>
    <L PROPNAME="participants">
        <p VALSTR="vidya">vidya</P>
        <P VALSTR="nigel">nigel</P>
        <p VALSTR="stefanm">stefan</P>
        <P VALSTR="geek">chris</P>
        <P VALSTR="matt">matt</P>
        <P VALSTR="gvallejo">gerardo</P>
    </L>
</RULE>
```

See Appendix A for the full vocabulary.

If the client says the command *"voice message vidya"* the recognizer first recognizes the command *voice message* and then assumes that the following word will be one of the "participants". When the recognizer has completed the recognition, it sends back to the client the value string (VALSTR) of the words it has recognized. In other words, instead of sending back "voice message vidya" as a string, it will send back *sendvim vidya* to the client. *Sendvim* is a command that exists on the server which sends back to the client a multicast address to connect to in order to send a voice message to the specified buddy. In the case of a user like *chris*, the actual login handle might be different from the spoken name, as a result, sending the value string as opposed to the string that is recognized allows more flexibility in creating nicknames and shortcuts for users but still maintaining the same handle within the SimPhony program.

The Client

The client side of the voice recognition works in two stages. First, the client pushes and holds the push-to-talk button and issues a command. The client establishes a UDP socket connection to the voice recognition server and sends packets to the server until the client releases the push-to-talk button. At this point, the client sends a 1 byte packet signifying that the command is finished and that the recognition can begin. The client then waits for a string from the recognition server and when it receives the string, closes the socket connection. If the string received is null, this signifies a false recognition by the speech recognition server and the client plays a tone signifying that it was a false recognition and that the user should try again. If the string is a valid command, the client sends the recognized string to the server and waits for a response from the server. If, as in the example above, the client sends *sendvim vidya* to the server, the server will instantiate the multicastServer class, create a multicast address, connect to that address and then send it back to the client so that the client can connect to that address and the user can begin recording her message.

4.2.6 Telephone Module

The telephone module of the SimPhony system is a Visual C++ layer that sits between the SimPhony messaging server, the voice recognition server and the telephone. It runs on a Windows NT desktop which contains the Intel Dialogic D/41JCT-LS 4-port board. This card has 4 input jacks for phone lines and thus is able to create a gateway between a traditional telephone or mobile phone and the wired and wireless network. This connection allows the incoming phone call to send and receive data from the voice recognition server and the SimPhony messaging server. The card acts similarly to a sound card, allowing the phone to be used as a microphone and speaker. The dialogic API gives the client the necessary procedures and functions to access the data coming in from the phone and write outgoing data to the phone.

This module detects when there is an incoming call to the SimPhony system, answers the call and providing some direction to the user on the other end and sends the users commands to the voice recognition server. The voice recognition server sends back the recognized command and the telephone client then sends it to the SimPhony server which sends back the same information to the telephone client that it does to the pocket PC client. At this point, the telephone client behaves similarly to the pocket PC client. It primarily acts as a multicast connector, allowing input to come in over the phone and get multicast to the address sent to it by the server. When checking for messages, the telephone client checks a text file with the same name as the current user which specifies how many messages that user received. The client then plays those files sequentially to the user.

Chapter 5

Feedback

The feedback gathered about the SimPhony system concentrates primarily on the features implemented and the technical feasibility of the design. The primary user groups giving feedback were techs and office workers working with the designers. The reason for this focus is primarily because the original design for the SimPhony system comes from the difficulty in using normal communication tools in the fab hence, the design is most closely tied to that scenario. The strict restrictions about materials brought into the clean rooms made it difficult to test the prototype on short notice. As a result, the most effective first round of feedback came primarily from techs role-playing the user scenario described in chapter 2 and the designers and colleagues refining the design through simple quality assurance methods. A more thorough review will be proposed from the lessons learned from the simple user interface and technical feedback gathered for the first prototype.

The remainder of this chapter will be broken up into two sections. First, we discuss user feedback received from members of the manufacturing team at Intel in Leixlip, Ireland. Secondly, we discuss challenges encountered in the technical design and talk about ways in which those challenges were addressed. Throughout the two sections, we will include comments and suggestions for how to improve or design a more thorough evaluation for the SimPhony system.

5.1 User Comments

Because the design for the SimPhony system is so closely tied to the fab scenario, the design of the system has gone through several revisions and comments by techs and others well versed in the manufacturing environment throughout its conceptual and design stages. The primary need for the techs was to have a lightweight communication system with which they could communicate with others in the fab and outside of the fab without having to go through the two step process of paging that person and waiting for him to call them back. Ideally, if that communication device could reside on a device that they already had and they could use it to send and receive data from one another, this system would be the ideal tool for their environment. Something that gave them the ability to have a high level overview about the progress of lots and what was going on in the fab, as well as a tool which allowed them to streamline the process of transferring information from one shift to another. This system focuses on the need for a lightweight communication device yet it hopes to be a flexible enough architecture so that it may someday be expanded to include data as well.

Despite the feeling that many office workers have about instant messaging being a tool used more for "goofing off" than being productive, the fab environment and other similar mobile working environments might be different. Because workers in these environments are constantly moving and often changing tasks, having a communication system that is able to move from being synchronous to asynchronous quickly, like instant messaging, might seem more useful. Rather than become a tool for distraction, it might become the only tool with the flexibility to handle multi-tasking in an environment where workers are not constantly seated behind a desk. Because of the restrictions of the environment and the bunny suits, however, text instant messaging is not a viable option. As a result, voice instant messaging, being able to send short recorded messages quickly back and forth became the primary candidate for communication in this environment. Users preferred the ability to use a lightweight system to send messages back and forth as opposed to constantly having a walkie-talkie system where everyone on the same band heard each other all the time.

Awareness both of activities going on in the fab and of others whereabouts is also important to the techs. Knowing when a lot (a group of microchips) is delayed before them meant knowing how much downtime they have and how to manage their own time. Having some idea of how long it would take someone to get back to them after having paged them is also important, particularly if they are having trouble with their tool. Again, instant messaging seemed like a viable tool because of the presence information that it provides. When a user is logged into an instant messaging client, he can manually set himself as online or away or any number of states or he can have the client automatically assume those states based on his activity at the computer. If he has typed on the keyboard or moved the mouse in the last 15 minutes, the client assumes that you are available but after a period of inactivity, sets you away and tells your buddies how long you've been in that state. This minimal awareness does not become too intrusive but can give a user a good idea of how long it might take for a buddy to respond to a message.

These two motivating factors made it clear that voice instant messaging presented the most fitting style of interaction for that work environment. The response to having a lightweight messaging system to send short voice messages was also positive. How much it might be used in a work environment remains to be seen, but techs felt that this might be a useful tool. In practice, the number of voice messages exchanged and their length and the number of "conversations" these messages spawned might all be interesting to look at to see if in fact this style of messaging was more convenient and frequently used than the synchronous full-duplex connection.

Secondly, the ability to communicate with more than one person at a time appealed to the techs. Their work rarely leaves them in isolation so being able to consult with other techs in the cleanroom and other people outside of the cleanroom seemed really applicable to their work needs. They requested the further ability to add someone in to an ongoing conversation. This is a feature easily added into a second generation of the SimPhony system. Creating groups of people whom they could message as one and talk to together related closely to their need to keep groups aware of the overall activity and work flow inside the fab.

One of the techs commented that they had tried several tools, like instant messaging, to help improve communication amongst team members. However, they had to adapt their work styles to use the messaging system as opposed to the system adapting to their environment. He felt that without trying the SimPhony system in their natural environment, it would be difficult to tell whether the behaviors of the system did in fact adapt to their work styles, however, he said that the type of system which used communication behavior to adapt to work style would probably be more successful than one which simply required users to adapt to it.

To truly get a sense for how well the SimPhony system works within the fab, techs would have to use the system in a long-term trial in both their day-to-day work and perhaps during a drill or some other activity in which their communication behaviors could be controlled and compared with another tool. Such a test is difficult to do in a lab or another external environment because the communication differs so much from one day to the next or between events that a true trail would have to take place in the fab or another similar environment. Testing and improvements done in the lab are addressed in the next section.

5.2 Design Modifications

The SimPhony system was used and is continually being tested for major usability problems and software bugs by its designers at the Media Lab. The primary problems tended to focus on feedback for certain types of behaviors and audio quality for both the instant messages and the multicast synchronous audio sessions.

5.2.1 User Interface Modifications

A majority of the modifications already made to the SimPhony system deal largely with the user interface and the way in which it gives feedback to the user on both the screen interface and the audio only interface. When using voice commands, the system alerts the user to its state by playing tones to indicate different events. Originally, when the system was ready to begin a synchronous conversation, it would play a begin recording tone and when the session was ended by the user, it would play and end recording tone. When sending a voice message, the begin recording tone would only play when the user actually began recording the message as opposed to when the system was ready to have the user begin the recording (i.e. when all parties had received the appropriate multicast address to join). This subtle difference in feedback caused confusion even for users who knew the system well. As a result, we decided to retain consistency and only give feedback when the client had begun or ended recording rather than when the system was ready to begin recording.

5.2.2 Technical Modifications

The technical feasibility of a system that works entirely over the wireless network was something that actually building and testing a prototype of the SimPhony system would tell us. In order to make voice communication successful, the quality of the audio needs to be clear and the delay needs to be minimal [24]. Because the system proposes the use of the multicast protocol implemented in the router, a protocol not yet used and tested to its full potential, we needed to make sure that it was appropriate to use. What we found however might produce significant difficulties in the architecture proposed.

We began by implementing a recording and playback application that let us test the audio quality of the recording and playback on the pocket PC. This code was then integrated into the multicast messaging architecture explained in the previous chapter. On first pass, the audio which was being recorded and simultaneously multicast on the pocket PC was extremely choppy. Packets were being dropped and only about 30% of the audio was being played on the other end. At this stage, we added buffering to the system on both the recording and playback side in case either the recording or the playback was happening faster than the packets were being sent or received. We also changed the size of the audio buffers being sent over the network hoping to find the perfect size which minimized the number of dropped packets while still allowing the audio to sound continuous if one packet was missed. We found the perfect buffer size to be 80ms. Any smaller seemed to incur a great deal of loss and any larger would make the audio incomprehensible if there was a packet lost. On the recording side, we stored up to 4 audio buffers at any given time. This prevented audio from being lost if the sending was not happening as fast as the recording. On the playback side we stored up to 10 buffers at any given time in case the playback was happening slower than the device was receiving packets over the network. This seemed to improve the quality of the audio from the first implementation but we found that we were still only receiving 50-60% of the audio during our tests.

In trying to uncover whether the loss was occurring on transmission or on receipt, we tried recording what we were sending from the pocket PC and comparing it to a recording we made of what was received on the other end. However, the overhead created by writing this data to a file decreased the quality of the audio even further. Because we were using multicast, we decided to compare what was heard on the iPAQ to the recording made on the wired server. What we found was the the recording on the server was more often than not complete and fully comprehensible. At this stage, we theorized that the packet loss we were hearing on the iPAQ might be due to the decreased bandwidth and high traffic on the wireless network. Our first experiment to test this theory was to implement a simple squelch algorithm which allowed us to only transmit when there was speech being recorded. This simple rule only transmitted audio over a certain threshold which was tested for each iPAQ individually since each device had a different microphone. This meant that we were only sending data over the network when there was speech to be transmitted and not sending ambient noise. This improved the full-duplex communication but it did not improve the quality of each individual transmission. Using tools like Winamp and Real Player as a model, we tried time stamping each packet sent and playing each one back with the same delay with which they were recorded. Although this would not prevent packet loss, it might keep the packets that were there playing with the correct spacing between them and would allow us to insert silence where there was a packet loss instead of subjecting the user to the click or pop that we were hearing. This tactic did not improve the quality of the audio for the listener so we decided to test our configuration even further to find the cause of the problem.

To see that there was actually packet loss corresponding to the pops we were hearing in the audio, we put a packet header on each packet with a packet number (0-xxxx) and the time at which the packet was created (hh:mm:ss). When a packet was received, the header was stripped and written to a file. The packet headers were compared on both the receiving pocket PC and the wired server. The pocket PC continually showed greater losses than the wired server. Furthermore, these losses corresponded to the ones we heard in the playback on the pocket PC. Since the primary difference between the server and the client is the network, we decided to test the client on a wired network as well. Using a PCMCIA card, we plugged the pocket PC into the wired network and tried the same test again. With this configuration, the pocket PC incurred much less loss and the quality of the audio was much improved. It seemed that the wired network performance was much better than the wireless network. We tried all of these tests both during the day, during periods of high network traffic and at night, during periods of little or no traffic. Although the quality of the audio improved slightly at night, the improvement was not significant enough to write off the decreased quality as a function of network traffic.

Because we were unsure whether the problem could be attributed to decreased bandwidth and increased traffic on the wireless network, we decided to see whether the problem could be attributed to the implementation of multicast on our wireless router. We altered a version of the SimPhony messaging client to use point-to-point sending as opposed to multicast. We compared the audio transmitted in this version to the multicast version and found relatively little difference between the point-to-point and multicast. Both versions were still dropping approximately 30-40% of the packets making the audio understandable at times and very difficult to understand at others.

In comparison, the voice messages which are recorded on the server and later streamed to the client are much more comprehensible. Perhaps because the messages are short and asynchronous or because they are streamed from the wired server, the fairly infrequent losses do not seem to deteriorate the quality so much that the message is no longer understandable.

Thus far, the solutions we have tried to improve the quality of the synchronous audio for the SimPhony system have made little improvement. What remains to be seen is whether trying the same system with a different wireless router might prove to be more successful. Until the problem is pinpointed or resolved, however, it is difficult to assess the feasibility of this architecture, using multicast protocol to transmit voice over IP in this setting. However, there are a few commercial voice over wireless LAN (VoWLAN) products that are slowly making their way onto the market and into the workplace. Many of these products implement their own quality of service mechanisms [25] to account for issues like security, reliable network coverage and reliable transmission during periods of high traffic.

Chapter 6

Conclusions

Although this is only the first iteration and initial reaction to the SimPhony system, many things can be taken from this design and the initial reactions of users.

The system that was built consisted of a few core features which make it unique from conventional text instant messaging clients and voice instant messaging clients soon to be on the market. The concept of transitioning from one style of communication, voice instant messaging, to another, synchronous full-duplex audio automatically based on frequency of communication is a concept unheard of in instant messaging. Although the ability exists to have synchronous text chats, the merging of the two does not. A feature like this could make communication in an already busy setting more efficient. Secondly, the idea of creating and maintaining groups and allowing them to communicate with the same interface that individuals might use is another feature unique to this prototype. In a setting where groupwork is mandatory, such a tool is in demand. And finally, allowing the user the ability to monitor more than one conversation at a time by allowing for intelligent and meaningful interruptions. Although these features have yet to be tested in day-to-day use by real users, initial reactions to the design and to the features were positive and encouraging.

Contextual design methodology prescribes that it is impossible to design for a specific user group without deeply involving yourself and educating yourself about that user groups work style and needs. Learning so much about group work and a manufacturing technicians job and the requirements not only of each individual but of the group as a whole served as great inspiration when designing features for the SimPhony system. That focus on the user's scenario seems to have paid off particularly in the style of the system and the various metaphors it uses for communication. The balance the system has between notifying other users of a buddy's presence but not overloading them with information at any given time fits both their needs at work and within the demands of their time.

Work in this area is far from complete. The final section touches on a few ideas that the designers of the SimPhony system see coming in the future.

6.1 Future Work

Over the course of designing the SimPhony system, many related areas and new features have come to light. We will discuss a few of them here but this is by no means a complete list of interesting future work that can be pursued.

The most obvious area to pursue future work would be in refining and perfecting and testing the current features of the SimPhony system. Improving the quality of the audio for the synchronous conversations would be a great boost in the system's ability to truly replace the telephone/pager system working in most cleanroom facilities today. A possible alternative and more scalable version of the current prototype would either develop a different scheme for transmission over UDP multicast or use a different, more robust scheme. Addressing issues like how often another group can interrupt a user when she is already in another conversation, or can there be multiple alerts for a single group when it goes inactive and become active again would also clean up the system's ability to manage multiple ongoing groups. Allowing the user more control in managing groups might also help solidify the notion of grouping within the system. If a user were able to dynamically add or remove people from a group conversation, the seamlessness with which they could move from one conversation to another would allow all of their communication interactions to happen with greater ease. Allowing users to notify others when they add them to groups and allowing them access to the same groups might also liken the idea of groups to a newsgroup or mailing list. Such groups could be published in some address book or other company list. Having a directory of users and allowing different users to configure their buddy lists in different ways might also help individuals adapt themselves and their workstyle to the system. Obviously, including more of the standard features of instant messaging and other similar communication systems would help make SimPhony a product rather than a prototype.

There were several features that we discussed when designing the architecture of the SimPhony system but decided not to include. Many are already implemented in other systems. Integrating text messaging, file transferring, and document sharing might also be useful for the techs and other workgroups. Buddies can be given nicknames by the individual user under the current system but also allowing standard nicknames for people who fulfill a specific role during all of their shifts or are always in a certain location might also be helpful. For example, if there is always a mechanic who works during every shift, messaging *mechanic* will get whomever is the mechanic working during that shift, not one person but the person who fulfills that role. Allowing more dynamic location based and role based messaging would require additional infrastructure but might also be useful. If the pocket PC registered a users location near a tool, calling that tool might get you the tech located closest to it or perhaps the one who's listed on the timetable as being assigned to that tool. In the cleanroom, all techs must use two hands when carrying lots from one location to another. If their gloves were to have sensors at the fingertips, the system might be able to guess when the techs were busy transporting expensive lots and not disturb them. Similarly, if nurses were with patients, the system might sense the ambient noise and not interrupt them with messages.

Classifying and organizing the history of conversations might also open another area of research. Visualizing each conversation by showing a timeline and the activity of each group member over the course of that timeline might allow users to more knowledgably browse the audio of past conversations. Sorting past conversations based on their participants, their length or their subject matter might further promote the audio to be later reused as valuable data for other techs or incoming workers on the next shift. Some groups might be public and allow anyone to join and participate in the discussion while others might be private and be only used to send specific types of information to participants or used only during emergencies.

Enhancing the role of the voice recognition in the user interface would also provide the user with a great deal more functionality without having to look at the screen. The current prototype requires that all users use the same vocabulary for voice recognition but loading a separate vocabulary for each user would allow the user to configure and add short cuts that then would not interfere with other users commands. If the vocabulary were updated dynamically, users could easily set short-cuts and nicknames to groups and users themselves. They could even create groups during a conversation and name the group using voice commands which would be directly input into the vocabulary. Users could also set themselves *away* or *busy* with voice commands and avoid interacting with the interface all together when their environment becomes hectic.

The architecture of the SimPhony system itself might also vary. Some techs were weary of having all of their conversations recorded for fear that the audio might be used to monitor their work or judge their productivity. The organization of the SimPhony system might be changed so that not all conversations are recorded by the server, but only ones that all members feel might be relevant to others. This would prevent techs who are having personal conversations from being penalized or make them afraid to communicate but would still allow consenting techs to record conversations for reference. Having less of the functionality be driven by the server and more driven by the clients themselves might also be a different take to the system. Instead of having the server be the central connection in every interaction between clients, clients might connect on their own when having one-to-one conversations. This difference in architecture might make the system safer and more scalable. Security is also another area in which the system could be greatly improved. Requiring users to login with a username and an encrypted password would be just the first step but would at least help prevent one user from logging in as another.

As is, the SimPhony system just touches the surface of what can and will be exciting in voice messaging. As further interactions of the design progress, the system will hopefully reach its potential in becoming more useful to distributed workgroups than the walkie-talkie or telephone.

Appendix A

Voice Recognition Vocabulary

```
<GRAMMAR LANGID="409">

   <RULE NAME="login" TOPLEVEL="ACTIVE">
        <P PROPNAME="request" VALSTR="login">login</P>
        <L PROPNAME="participants">
            <P VALSTR="vidya">vidya</P>
            <P VALSTR="nigel">nigel</P>
            <P VALSTR="stefanm">stefan</P>
            <P VALSTR="geek">chris</P>
            <P VALSTR="matt">matt</P>
            <P VALSTR="gvallejo">gerardo</P>
        </L>
   </rule>

   <RULE NAME="sendvim" TOPLEVEL="ACTIVE">
        <P PROPNAME="request" VALSTR="sendvim">voice message</P>
        <L PROPNAME="participants">
            <p VALSTR="vidya">vidya</P>
            <P VALSTR="nigel">nigel</P>
        <p VALSTR="stefanm">stefan</P>
        <P VALSTR="geek">chris</P>
        <P VALSTR="matt">matt</P>
        <P VALSTR="gvallejo">gerardo</P>
    </L>
</rule>

<RULE NAME="sendvoice" TOPLEVEL="ACTIVE">
    <P PROPNAME="request" VALSTR="sendvoice">connect to</P>
    <L PROPNAME="participants">
        <p VALSTR="vidya">vidya</P>
        <p VALSTR="nigel">nigel</P>
        <P VALSTR="stefanm">stefan</P>
        <P VALSTR="geek">chris</P>
        <P VALSTR="matt">matt</P>
        <P VALSTR="gvallejo">gerardo</P>
    </L>
</RULE>

<RULE NAME="listen" TOPLEVEL="ACTIVE">
    <P PROPNAME="request" VALSTR="listenvim">listen to messages</P>
</RULE>

<RULE NAME="select" TOPLEVEL="ACTIVE">
    <P PROPNAME="request" VALSTR="select">select</P>
    <L PROPNAME="participants">
        <P VALSTR="vidya">vidya</P>
        <p VALSTR="nigel">nigel</P>
        <P VALSTR="stefanm">stefan</P>
        <P VALSTR="geek">chris</P>
```

```
            <P VALSTR="matt">matt</P>
            <P VALSTR="gvallejo">gerardo</P>
        </L>
    </RULE>

    <RULE NAME="add" TOPLEVEL="ACTIVE">
     <P PROPNAME="request" VALSTR="add">add member</P>
     <L PROPNAME="participants">
            <P VALSTR="vidya">vidya</P>
            <p VALSTR="nigel">nigel</P>
            <P VALSTR="stefanm">stefan</P>
            <P VALSTR="geek">chris</P>
            <P VALSTR="matt">matt</P>
            <P VALSTR="gvallejo">gerardo</P>
        </L>
    </RULE>

</GRAMMAR>
```

Bibliography

[1] S. Whittaker, D. Frohlick, and O. Daly-Jones. Informal workplace communication: What is it like and how might we support it? *Proceedings of Human Factors in Computing Systems*, pages 131–137, 1994.

[2] ICQ Inc., http://www.icq.com/products/whatisicq.html. *What is ICQ?*, 1998-2003.

[3] S. Bly, S. Harrison, and S. Irwin. Media spaces: Bringing people together in a video, audio and computing environment. *Communications of the ACM*, (36):28–45, 1993.

[4] R. Fish, R. Kraut, and B. Chalfonte. The videowindow system in informal communication. *Proceedings of the Conference of Computer Supported Co-operative Work*, pages 1–12, 1990.

[5] A.J. Sellen. Remote conversations: The effect of mediating talk with technology. *Human Computer Interaction*, 10:401–444, 1995.

[6] R. Fish, R. Kraut, R. Root, and R. Rice. Video as a technology for informal communication. *Communications of the ACM*, (36):48–61, 1993.

[7] M. Mantei, R. Baecker, A. Sellen, W. Buxton, T. Milligan, and B. Wellman. Experience in the use of a media space. *Proceedings of the Conference on Computer Human Interaction*, pages 203–209, 1991.

[8] J. Tang, E. Isaacs, and M. Rua. Supporting distributed groups with a montage of lightweight interactions. *Proceedings of the Conference of Computer Supported Co-operative Work*, 23-34 1994.

[9] R.W. Root. Design of a multi-media vehicle for social browsing. *Proceedings of the Conference of Computer Supported Co-operative Work*, pages 25–38, 1988.

[10] E. Pedersen and T. Sokoler. Aroma: abstract representation of presence supporting mutual awareness. *Proceedings of the Conference on Computer Human Interaction*, pages 51–58, 1997.

[11] J. Herbsleb, D. Atkins, D. Boyer, M. Handel, and T. Finholt. Introducing instant messaging and chat in the workplace. *Proceedings of the Conference on Computer Human Interaction*, pages 171–178, 2002.

[12] E. Isaacs, A. Walendowski, and D. Ranganthan. Hubbub: A sound-enhanced mobile instant messenger that supports awareness and opportunistic interactions. *Proceedings of the Conference on Computer Human Interaction*, pages 179–186, 2002.

[13] Nextel, http://www.nextel.com/services/directconnect/ptt_overview.shtml. *Push To Talk Services*, July 2003.

[14] A. Woodruff and P.M. Aoki. Media affordances of a mobile push-to-talk communication service. 2003. submitted for publication.

[15] H.H. Clark and S.A. Brennan. *Perspectives On Socially Shared Cognition*, chapter Grounding in Communication, pages 127–149. APA, 1991.

[16] D. Hindus, M. Ackerman, S. Mainwaring, and B. Starr. Thunderwire: A field study of an audio-only media space. *Proceedings of the Conference of Computer Supported Co-operative Work*, pages 238–247, 1996.

[17] R. Rodenstein and J.S. Donath. Talking in circles: Designing a spatially-grounded audioconferencing environment. *Proceedings of the Conference on Computer Human Interaction*, pages 81–87, 2000.

[18] J. Kim. Tattletrail: An archiving voice chat system for mobile users of internet protocol. M. eng., MIT, June 2002.

[19] A. Singer, D. Hindus, L. Stifelman, and S. White. Tangible progress: Less is more in somewire audio spaces. *Proceedings of the Conference on Computer Human Interaction*, pages 104–111, 1999.

[20] P.M. Aoki, M. Romaine, M.H. Szymanski, J.D. Thornton, D. Wilson, and A. Woodruff. The mad hatter's cocktail party: A social mobile audio space supporting multiple conversations. *Proceedings of the Conference on Computer Human Interaction*, 2003.

[21] J. Watts, D. Woods, J. Corban, E. Patterson, R. Kerr, and L. Hicks. Voice loops as cooperative aids in space shuttle mission control. *Proceedings of the Conference of Computer Supported Co-operative Work*, pages 48–56, 1996.

[22] Vocera Communications. http://www.vocera.com, 2000.

[23] H. Schulzrinne. Some frequently asked questions about rtp. 2003. http://www.cs.columbia.edu/ hgs/rtp/faq.html#reliable.

[24] V. Hardman, A. Sasse, and I. Kouvelas. Successful multi-party audio communication over the internet. *Communications of the Association of Computing Machinery*, 41(5):74–80.

[25] N. Gohring. Voice over wireless lan: Ready or not? *InfoWorld*, September 2003. http://www.infoworld.com/article/03/09/12/36FEvowlan_1.html.