A Reusable Peer-to-Peer Conversation Tool for Online Second Language Learning

A thesis presented in partial fulfilment of the requirements

for the degree of

Master of Information Science in Computer Science

at Massey University, Palmerston North, New Zealand.

Jun Ye

2008

Abstract

To support extramural learning, Johnson (2005) has proposed the Learning Computer concept, which aims to provide a learning appliance that can be used for studying university courses at any time, from anywhere, and by anybody who might have only basic software and hardware, dial-up Internet connection, and little computer literacy. Lonely extramural students need extra support for interactions and collaboration in learning, especially in second language learning that requires intensive oral language practice between the students and the tutor.

This research project was a trial to extend IMMEDIATE (the prototype of the Learning Computer) to a second language extramural course. To meet the requirements of long distance conversation in such a course, a synchronous/asynchronous bimodal approach was conceptualised based on a review of e-learning, communication, and VoIP technologies. It was proposed that the prototype should automatically adapt to either synchronous mode or asynchronous mode according to different levels of Internet connection speed. An asynchronous conversation mode similar to Push-to-Talk (PTT) was also proposed.

A VoIP SDK was investigated and used in the prototype for fast development. IMMEDIATE messaging protocols have been extended in the prototype to control call procedures and the asynchronous conversation mode. An evaluation of the prototype which was conducted to assess its usability, functionality and integrity of the prototype demonstrated that users can conduct telephone-like synchronous conversation efficiently at high connection speed. Although the PTT-like asynchronous mode has a time lag problem, especially when two users are both at low connection speed, it is a still a good way for novices to practise second language oral skills. The evaluation has given strongly support to the feasibility and effectiveness of the bimodal approach for applying IMMEDIATE in second language extramural learning.

Acknowledgements

This project has been carried out with support and help from many people.

Firstly, I would like to thank Dr Russell Johnson, my supervisor, for the time and patience he has put in to helping me through this master's project. Under his consistent supervision and guidance, I carried out this research project from conceptualisation, prototyping, and evaluation to thesis writing, during which I became familiar with the research field, became confident to do research, and learnt how to write a thesis.

I would like to acknowledge the support from the staff of Computer Science, IIST, Massey University.

I also appreciate my friends for their help in the evaluation and encouragement in the whole study procedure.

Publication

The publication associated with this research is:

Ye, J & Johnson, R. 2008: An embedded bimodal tool to enable second language learners to practise conversation online over unreliable Internet connections. Paper presented at *the 5th International Workshop and Conference on Technology for Innovation and Education in Developing Countries (TEDC 2008)*, Kampala, Uganda.

Contents

Figures and Tables

Figures

Tables

Chapter 1

Introduction

1.1 Background

The Internet has brought people closer together and made it possible for them to share information and knowledge from all over the world. It not only greatly helps people to gather information and learn new knowledge, but also enhances people's social life via innovative forms of communication. However, some realities restrict the application of Internet technologies. The Internet needs investment from governments, organisations and users as its availability of access varies widely due to either poverty or geographical isolation that cause a wide gap between haves and have-nots, not only in developing countries, but also in developed countries.

Education is the main way for people to acquire knowledge, integrate into the workplace and society, and reduce the difference between the rich and the poor. Like other human rights, everyone in the world should have the right to share in educational resources. With the development of the information society, learning has become a life-long task. However, education resources, including teachers and premises, may not be enough to meet the growing demand. Distance learning can meet such requirements by letting students outside the campus learn from their own homes by themselves. The Internet and information technologies allow learners and teachers to interact and collaborate with each other in distance learning.

One contradiction in distance education is between the uniform learning materials and learners' different backgrounds, knowledge and preferences. A major effort in e-learning research is to apply some artificial intelligence strategies to learning systems to meet learners' personal requirements. An intelligent tutoring system (ITS) provides a highly interactive learning environment and individualised learning support to deliver one-to-one instruction automatically and cost effectively (Ong & Ramachandran 2000). For example, LISP Tutor (Anderson & Reiser 1985) plays the role of a human tutor to teach programming using LISP. With the popularity of the Internet and its applications,

ITS has been further evolved to the web-based adaptive learning system or the adaptive hypermedia learning system, which supports learning by suggesting most relevant links in hyperspace according to the learner's goals, knowledge and preference (Brusilovsky 1996). Nevertheless, both stand-alone ITSs and web-based adaptive learning systems are developed or used mainly for research or training purposes in some limited fields or subjects.

The delivery of universities' online courses has been steadily increasing during the past few years (Garrett et al. 2005). LMSs (Learning Management Systems) have been widely used in universities for delivering extramural courses, or for blended learning (Clarey 2007). Web technologies make it possible to integrate course management, teaching and learning in an LMS. From the web pages of an LMS, students can view and download learning materials, submit assignments, and collaborate with other course participants using communication tools (e.g. forums or emails). Multimedia learning supports (e.g. audio or video lectures) are also often provided in an LMS to support the learning. Nevertheless, LMSs are designed mainly for teachers and the university to deliver online courses. They are learning-provider-centred rather than learner-centred. In addition, LMS products - such as the most popular products Blackboard (n.d.[1]) and WebCT (has been merged into Blackboard) - can be very expensive (Olsen 2001).

With the development and popularity of information technologies, there is a trend for designing user-centred information appliances (Norman 1998) for general users who should be able to use an information appliance without being either aware of the underlying technology or concerned about the software, hardware or Internet connection speed. They should also be affordable and be used in an "any time, anywhere, anybody" manner. Information appliances are not generally stand-alone systems. They may communicate with each other or with other systems via the Internet. In many developing countries, or even in some rural areas in developed countries, low speed dial-up connection is still the main way to access the Internet. In such areas, Internet connection can be a major obstacle to the application of information appliances in distance learning. For example, a survey by Statistics New Zealand (2006) shows that dial-up connection is still used to access the Internet by many rural users and some urban users in New Zealand.

[1] In this thesis, the Harvard Referencing System is followed where 'n.d.' is used in a reference with no known publication date, as is frequently the case with web –published documents.

Since e-learning is a mass market, students have different backgrounds, knowledge and conditions. Tools for e-learning should be user-friendly – they must be able to be accessed and used easily by students without much knowledge, effort, or hardware/software investment. Current web-based learning systems are not suitable for learners who are using unstable and slow dial-up connection. Slow Internet connection not only makes learning materials difficult to access but also means that "the level of interactivity, individualisation and collaboration falls well below" what is needed for effective study (Johnson et al. 2002, p. 634).

A learning appliance – Learning Computer (LC) – has been proposed and prototyped (Johnson 2005). The LC is conceptualised as an information appliance, which means that it is user-centred, with the aim of providing a learning appliance that enables universities' extramural courses to be studied from anywhere by any learner who might have only basic software and hardware, dial-up Internet connection, and little computer literacy. To achieve "any time, anywhere and anybody" learning, the LC targets rural area learners and enables them to study universities' extramural courses at both high and low connection speeds. A prototype of the LC, IMMEDIATE, has also been developed and evaluated. In Chapter 2 the LC and IMMEDIATE will be discussed further.

1.2 Requirements for Extramural Second Language Learning

In traditional on-campus learning, the face-to-face tutorial allows direct interactions between students and the tutor to enable to former to clarify ideas, improve understanding and grasp knowledge. In distance learning, interactions and collaboration are extremely important to support learning for isolated learners. Multimedia course materials - such as voice or video - can also enrich online learning experience.

Teachers and students have been broadly using online communication tools - such as Windows Live Messenger (n.d.) and Skype (n.d.) for text chatting, voice chatting and video chatting - in both on-campus learning and distance learning. These communication tools have greatly influenced learning by providing effective means of interaction and collaboration. However, for communication (such as voice chatting), if one user is on low Internet connection speed, these tools may not always be workable.

In second language learning, conversations may be the most difficult task for students because it demands listening and speaking skills. In order to be able to use the language in the real world, students have to spend much time repeatedly practising conversations. Internal students have opportunities to converse face-to-face with the tutor or others. For extramural second language learners, practising oral language with the tutor or others is vital. Telephone or online communication tools (such as Skype) can enable them to conduct oral communication. However, telephone calls are expensive for long distance communication. For instance, in an English as second language extramural course that is delivered by a university in New Zealand, students in Thailand need to practise oral skills with a tutor in New Zealand. Conversation using international telephone calls is financially infeasible. Computer-mediated online conversation tools (such as Skype) are cheap for long distance communication. However, they are usually based on broadband connection, which might not be available to many learners, especially to those who live in some rural areas.

IMMEDIATE is designed for both broadband and narrowband users who are taking universities' extramural courses. In order to be used under both broadband and narrowband connections, IMMEDIATE supports asynchronous communication between course participants by exchanging text messages. To meet the requirement of the English as a Second Language extramural course delivered by Massey University in New Zealand, IMMEDIATE needs a new function to support conversation practice between students and the tutor under either broadband or narrowband connection. The main challenge for this is the bandwidth limit for voice communication because some students may live in areas where only low dial-up Internet connection speed is available. Another challenge is how such functionality can be implemented within the IMMEDIATE architecture.

1.3 Research Objective and Methodology

The goal in this project is to find a way to design a conversation component for IMMEDIATE Learning Computer to apply in an extramural second language course.

To provide some understanding of the background of the Learning Computer, e-learning systems and information appliances will be reviewed first. To form an initial conceptual model for the conversation component, people's communication methods, especially communication in e-learning, will be reviewed. After a description of

IMMEDIATE, technologies of voice communication via the Internet will be investigated. Then a specification will be proposed to outline functions and the user interface of the conversation prototype to meet the requirements for second language learning in IMMEDIATE. A prototype will then be implemented according to the specification. Finally, the prototype will be evaluated. These steps can be listed as follows:

- To give an understanding of the motivation for the Learning Computer, in Chapter 2 some e-learning systems and information appliances will be reviewed. The Learning Computer will also be introduced, followed by a discussion of people's communication methods, especially in the e-learning sector. Finally, an initial conceptual model for the prototype will be proposed.
- To provide some understanding of the possible challenges and problems of voice communication in this project, current technologies for Voice over Internet will be reviewed in Chapter 3.
- For fast implementation of the prototype, a voice communication tool or product that can be used in the prototype will be investigated in Chapter 4. Then a specification for the prototype will be presented.
- In Chapter 5 the implementation mechanisms of the prototype according to the specification will be described.
- An evaluation of the prototype will be conducted and analysed in Chapter 6.
- Finally, in Chapter 7, conclusions will be drawn and areas of further work will be identified.

1.4 Summary

In this chapter the Learning Computer and its background has briefly introduced. Intelligent tutoring systems, adaptive learning systems and learning management systems cannot meet the requirements of distance learning - learning at any time from anywhere by anybody. In the light of the development of information appliances, the Learning Computer – a learner-centred learning appliance - has been proposed and prototyped.

The aim in this project was apply IMMEDIATE to a second language extramural course for improving speaking skills. A conversation component is needed for learners to be able to interact with the tutor or others in IMMEDIATE. Possible challenges have been raised and the research methods and steps have been presented.

Chapter 2

An Initial Conceptual Model

Conceptualisation is a necessary step before detailed prototyping to transform requirements into a conceptual model (Sharp et al. 2007, p. 540). In this chapter, to facilitate some understanding of the background of the Learning Computer concept, some e-learning systems will be reviewed, followed by a description of the information appliance and invisible computer concepts. The Learning Computer will then be introduced with its prototype IMMEDIATE. Next, the communication aspect in e-learning will be discussed. Lastly, an initial conceptual model for the prototype will be presented.

2.1 Immediate Learning

To enable the reader to understand past efforts on e-learning and the current status of universities' online learning systems, I begin this section with a review of intelligent tutoring systems and modern online learning systems. I also explain why the information appliance is needed. Finally, the immediate learning concept is discussed.

2.1.1 E-learning Systems

With the demand for the knowledge society, learning is becoming a lifelong task. Not only school and tertiary students need to study, but also people outside the school or university need to improve their skills to meet new requirements for their careers. Moreover, many people study a special subject just for interest. Internal tertiary education cannot meet the demand for higher education or professional development due to their limited teaching resources and learning premises. There is a large demand for learning university courses by extramural learners. In the following sections I will present two main kinds of e-learning systems that have different focuses – *adaptive learning systems are learner-centred learning systems, and learning management systems are learning-provider-centred learning systems*.

2.1.1.1 Adaptive Learning Systems

A person uses a piece of software because it is smart and can help in his/her job. Cooper (2004, pp. 149-159) stresses the value of defining the goals of a system. Goals are stable over quite a long time while tasks that achieve the goals can vary with different implementation methods or technologies. During interaction with an application or software, besides the practical goal of getting things done, the user needs to satisfy his/her personal goal such as having fun and not feeling stupid or frustrated.

In a traditional face-to-face classroom, students listen and watch the tutor when the tutor gives instructions, and the tutor answers questions and gives advice to each student according to his/her feedback, test results and behaviour. Student-tutor interactions can improve the understanding between the student and the tutor. Through the collaboration of group work, students are motivated to participate in learning activities and achieve their learning goals.

The main obstacle to extramural learning is the isolation of the learner from the teacher and other learners. This isolation makes interactions and collaborations difficult for the learner, which may affect the learning results and lower learners' motivation to learn. Therefore, the practical goal of extramural learning is to learn efficiently and effectively, while the personal goal is to learn easily and enjoyably. E-learning uses information technologies in distance learning to make the learning more effective and enjoyable. There are many research efforts on e-learning to support personal learning. Such efforts include Intelligent Tutoring Systems (ITSs) (Urban-Lurain n.d; Lane 2006; Corbett et al. 1997) and adaptive web-based learning systems (Brusilovsky 1996).

ITSs and adaptive web-based learning systems are adaptive systems whose designers try to meet the different requirements of individual learners (such as different knowledge, backgrounds and preferences). Adaptation and adaptivity are two dimensions of human-computer interactions. A system is called *adaptable* if it provides the user with tools that make it possible to change the system characteristics. A system is called *adaptive* if it is able to change its own characteristics automatically according to the user's needs (Oppermann 1994).

An ITS tries to do the job of a tutor in a traditional face-to-face classroom by delivering flexible individualised tutorial and learning support to the learner according to his/her

knowledge, skill and expertise. There are three key ITS technologies or strategies: curriculum sequencing, intelligent analysis of student's solutions, and interactive problem solving support (Brusilovsky 1999). Different ITSs can apply different levels of intelligence by using four fundamental ITS components - a domain model, a student model, a teaching model and a user interface (Freedman 2000; Wisher et al. 2001). Many ITS prototypes have been developed and evaluated such as LISP Tutor (Anderson & Reiser 1985) for teaching LISP programming skills, and Sherlock (Wisher et al. 2001) for teaching aircraft troubleshooting skills. ITSs have been proved to be especially efficient for professional training (Ong & Ramachandran 2000).

Adaptive web-based learning systems bring the benefit of Internet technologies to stand-alone ITSs. Such systems can include dimensions such as authoring tools (Murray et al. 2003), modelling techniques, navigating supports, multimedia, online forums, and collaborative learning supports. Therefore, they are often very complex. ELM-ART (Brusilovsky et al. 1996a) is the first web-based adaptive system that provides an online adaptive interactive textbook to teach programming. It supports learning by means of adaptive navigation, course sequencing, individualising, and communication and collaboration. The Web technology makes it possible to integrate “the features of courseware management systems, electronic textbooks, learning environments, and intelligent tutoring systems” in ELM-ART (Weber & Brusilovsky 2001, p. 379).

InterBook (Brusilovsky et al. 1996b) is a web-based adaptive hypermedia learning system derived from ELM-ART. InterBook indexes the learning content (textbooks) according to a domain model, which has been designed by domain experts. The student’s navigation in the knowledge space is based on this domain model and tailored by the student model. However, an evaluation (Brusilovsky & Eklund 1998) has shown that students who are not familiar with the tool are easily confused by its conceptual model of the tool (Norman 1998, pp. 154-155). This is because the conceptual model of InterBook is not intuitive and it is a kind of exploratory learning - different from traditional sequential learning. In order to become used to the system, a student has to learn and adapt to the system. One problem of such learning systems is that the learner has to spend much time understanding how the system behaves rather than focusing solely on learning content (Smulders 2003). To let the learner concentrate on learning tasks, the learning system should be simple and easy for the learner to use. He/she should not be distracted by the complex interface of today’s systems and “nothing

should be on the screen that is not directly relevant for learning the material at hand" (Bork 2001).

Murray et al. (2000) evaluated a hypermedia system for an introductory geology course. Compared with the complexity of InterBook, which is based on a networked knowledge space, it has a simplicity that is based upon the hierarchical structure of an authored textbook. The system has a simple user interface and some exploratory tools. It shows that without intelligent technology (such as modelling), a learning system can also let the learner achieve a satisfactory learning experience by means of a good user interface and user-adaptable features. It also demonstrates that a learning system can be more simple and effective if it is designed for a specific kind of course rather than for general courses. This does not mean that intelligent features are of no use, but emphasize the importance of a good user interface and user features in an information system.

2.1.1.2 Learning Management Systems

Adaptive web-based learning systems are usually prototypes for research purposes or special training usage such as for military training (Wisher et al. 2001). For delivering online distance courses, universities need some learning systems that fit all subjects and courses, as well as meeting teachers' general administration requirements.

LMSs (Learning Management Systems), which can also be called CMSs (Course Management Systems) or VLEs (Virtual Learning Environments) (Wolfram 2005), establish distance learning environments for universities. An LMS is "software that manages learning events and learners and serves as a platform to deliver e-learning" (Clarey 2007, p. 23). The primary objective of an LMS is for teachers to manage learners, keeping track of their progress and performance (Brandon Hall 2008). From the students' point of view, some advantages of using LMS courses over on-campus courses include the ability to learn at any time and from anywhere (availability), self-paced learning, and learning by collaboration (communication). Learning materials can be viewed and downloaded from a central server. LMSs also emphasise the facilitation of the interaction and collaboration between the teacher and students by either synchronous or asynchronous communication methods. For example, a major e-learning solution company BlackBoard (n.d.) supports a synchronous virtual classroom, asynchronous discussion board (discussion forum), and email in its LMS. There are also

some open source LMS products such as Moodle (n.d.) for free availability and flexibility, as well as in-house LMSs that provide flexible solutions such as meeting local bandwidth requirements (Garrett et al. 2005).

An LMS focuses on course delivering and management, while an LCMS (Learning Content Management System) focuses on the creation, delivery and reuse of learning objects, which form a learning repository for storage and retrieving (ASTD 2001). Both systems are designed for teachers delivering general instructional strategies, such as learning material availability, feedback, and collaboration. In a review, Bradford et al. (2007) summarised several drawbacks of BlackBoard LMS such as: users found that it was harder to learn than expected, it was restricted to particular operating systems such as Windows NT, and expensive.

The core function of an LMS is information transfer between the teacher and students via the central server. Bork (2001) states the need for the move from information transfer learning (such as the LMS learning) to tutorial learning – the computer cannot be a tool for information transfer learning only, but can be a also tutor for tutorial learning. He urged the conversion from teacher-centred systems to learner-centred systems and the application of the information appliance and ubiquitous computing to support learning.

2.1.2 Information Appliance

The development of information technologies and the computer industry have forced computers to become more affordable and necessary in people's lives. As a result, computers have become all-in-one tools that can include all possible software to fulfil tasks both at work and in life outside work. In the office, people can run special software to fulfil particular tasks for their work. For example, in a building designing company, people use AutoCAD to draw architectural plans. In the office or at home, Microsoft Office Suite can help people prepare different kinds of documentation. When a computer is connected to the Internet, it can be used as a communication tool for tasks such as sending or receiving email, chatting with friends and browsing websites.

Each piece of software on computers has its own menus, functions and interface. The complexity and diversity of software means that, for a novice, a computer is not easy to

use. To use a computer for a special purpose, a novice has to prepare him/herself by spending much time learning how to use both the computer and software.

Don Norman points out that the computer industry has come to a crossroad – traditional technology-driven or feature-driven tools versus consumer-driven or task-driven tools (Norman 1998, pp. 31-39). He raises the notion of the information appliance in which the tool "fits to a task so well that it becomes a part of the task, feeling like a natural extension of the work, a natural extension of the person". In contrast to the all-in-one personal computer, an information appliance is designed particularly for a specific purpose. The user does not need to learn much to use it and is not easily distracted from the task by the tool. It has a clear conceptual model that shows interactions between the device and the user so that the user is able to control the tool easily.

An information appliance has a simple and intuitive user interface. Generally, there are two kinds of user interface: one directly reflects the underlying implementation mechanisms of the system, while the other is task-oriented without much choice or control left for the user (Gentner & Grudin 1990). The former contains technical details on its interface and is suitable for professionals or specialists who know the mechanisms of the system so that they can get direct control over it and get their work done more efficiently. The latter kind of interface is simple and intuitive without the use of detailed technical words. It can be easily learnt and used by novices or general users who are not interested in the underlying mechanisms. In such a system, the underlying engineering model is hidden from the user.

Information technologies should be embedded into an information appliance so well that the user will not be aware of the technologies – they should be invisible to the user. The information appliance in this sense is similar to calm computing and ubiquitous computing (Weiser 1991). Information appliances should not only be easy to use, but should also be able to be used at any location and any time. That is to say, people should be able to use information appliances anywhere and at any time without being aware of their underlying technologies. An information appliance must also be cheap so that it can be affordable for most people. The high cost of many information technologies is a barrier to their use for a large number of people using .

To support ubiquitous features of information appliances, there should be some universal communication standards for exchanging information between different

information appliances (Norman 1998, pp. 62-63). Standardisation is the way to achieve interoperability among different hardware and software provided by multiple vendors. For example, standards such as MPEG-1, MPEG-2, MPEG-4 and MPEG-7 in the MPEG family (ISO standards) cover some special multimedia application areas. MPEG-7, for instance, defines a multimedia (including audio, video, image and animation) description standard for retrieval purpose (Rao et al. 2002, p. 282). There are standardisation organisations for different fields of information technologies.

Before the design of iPOD in 2001, digital music players were "either big and clunky or small and useless" (Kahney 2006). An iPOD is simple to operate, and small and light so it can be easily carried and used anywhere. The interface of the device is very simple with only several buttons on it, so that users can easily perform tasks such as play, pause and select songs. The interface design is focused on ease of use rather than the technological capability. An iPOD can play songs in most audio formats. Apple provides the iTune software, which can quickly download songs from an online music store or transfer songs from a CD. From the author's point of view, iPOD is a successful information appliance for these reasons: simplicity, ease of use, portability, and information exchange ability from outside song resources to the iPOD.

2.1.3 Requirements of Immediate Learning

The starting point for this research is that learning should be available and affordable for everybody, any time and anywhere. In the light of information appliances, the focus of a distance learning system should be on learning rather than on teaching. The system must have an intuitive conceptual model and a simple interface so that the learner does not need to spend much time exploring how to use it. Bork (2001) suggests an invisible interface by simplifying the interface with only directly relevant learning material on the screen. In contrast to an all-in-one system, all irrelevant tasks should be excluded from it.

Learners usually need up-to-date hardware, software and broadband to use modern Web-based learning systems. However, many learners may have only an old computer and dial-up Internet access. This meets only the minimum requirement for some basic tasks such as email and word processing. Johnson et al. (2002) point out that an LMS falls short of the accessibility requirements in slow and unstable Internet accessing

situations, and there is insufficient support for interaction and collaboration between students and the teacher.

Learning is a complex activity. It can include reading, thinking and understanding, which are static personal learning procedures conducted by the student himself/herself. Listening to the teacher, and discussing the course with him/her or other learners can enhance learning. The teacher's role is to help students to understand and grasp the knowledge of the course via instructions, assignments, and interactions. Different subjects have different requirements for learning activities. For example, in second language learning, conversation exercises are necessary for the student to practise with a tutor or other students to improve his/her speaking skill. Extramural study lacks the face-to-face communication and interaction environment which enhance learning. The Internet and multimedia technologies show the potential to provide support for distance learning. By using communication software and broadband Internet connection, the learner can communicate with the teacher and other learners for interaction and collaboration.

The Internet is especially important for rural households - especially farmers - as a main way to link with the outside world by using online banking, online business, e-mail and other services. They really need fast Internet access to cope with their daily requirements of work and life. However, broadband services such as Cable, DSL (Digital Subscriber Line) and wireless connections are not always available in many rural areas. Many people in rural areas are either not aware of the availability of satellite Internet connection or cannot afford it. Additionally, although the trend is to use the broadband instead of the narrowband, the availability and low cost of dial-up make it still a feasible choice for many urban and rural users.

A survey undertaken at the end of 2006 shows that 33.2% of New Zealand households have broadband access to the Internet, while 30.9% only have only dial-up connection (Statistics New Zealand 2006). In addition to the unavailability of broadband access in some rural areas, the survey also shows that another main reason for users to choose dial-up access is that it is cheaper than broadband access. Because of the unstable and slow dial-up connection in some rural areas, distance learning students living in such areas cannot always rely on the dial-up connection to access universities' online learning systems or repositories for their extramural study.

Learning can happen immediately when there is no cost, computer knowledge, or location barriers for a student to access learning resources. By removing requirements of financial investment, knowledge and location, a learning appliance can make learning happen immediately. It provides the feasibility, availability and affordability for anybody and anywhere learning. A learning appliance - the Learning Computer - has been proposed and prototyped by Johnson (2005) as a means of supporting immediate learning.

2.2 The Learning Computer

The Learning Computer (LC) aims to help learners without modern hardware and software, computer literacy or fast Internet connection to study universities' extramural courses. To achieve this, one design guideline of the LC is to maximise the learning functionality for the distance student while minimising the usability and accessibility problems of web-browser-based systems (Johnson et al. 2006).

Currently, Microsoft Windows is the most popular operating system installed on computers to run tools or applications. It has a graphical interface with various features and functions on it. Novices must learn about it before they can use any tool on it. Moreover, people often find themselves spending unnecessary time browsing various websites or playing games. These irrelevant resources can distract students from their real learning tasks.

A prototype IMMEDIATE has been developed to demonstrate the Learning Computer concept. The client application of IMMEDIATE – Learning Shell, which is built on the Microsoft Windows operating system, takes the place of the desktop of the Microsoft Windows so that it hides the irrelevant information on Windows desktop and presents learners with a simple interface.

Unlike web-based online courseware that is thin client and fat server with most functionality on the server side, the Learning Shell is heavyweight with learning materials and most functionalities. The Repository Manager is the server application that deals only with communication and data transfer aspects for IMMEDIATE. The Learning Shell connects only periodically to the server. When the connection is possible, it updates users' messages and learning materials from the repository. When there is no available connection to the server, the study material can be either delayed for download

later or delivered to the learner via CDs to ensure that minimum learning criteria are met. This tactic makes it possible to let learners living in rural areas study a university course extramurally at low Internet connection speed.

IMMEDIATE is a task-oriented learning system. Learning tasks such as taking notes, viewing lecture slides and guidelines, and interacting with tutors can be organised into at least one of several learning modes – learning by textbook, learning by lecture, learning by exploration, learning by collaboration, learning by doing and learning by tutorial. These learning elements or tasks are designed as reusable Delphi[2] components bound with appropriate learning resources. The Learning Shell assembles all learning components on the student's desktop over the Windows operating system. Figure 2.1 (Johnson 2005, p. 101) depicts learning components for two learning modes – learning by lecture and learning by tutorial. The shadowed components in this figure are selected learning components that form a learning mode. The figure shows that learning components can be reused in different learning modes.

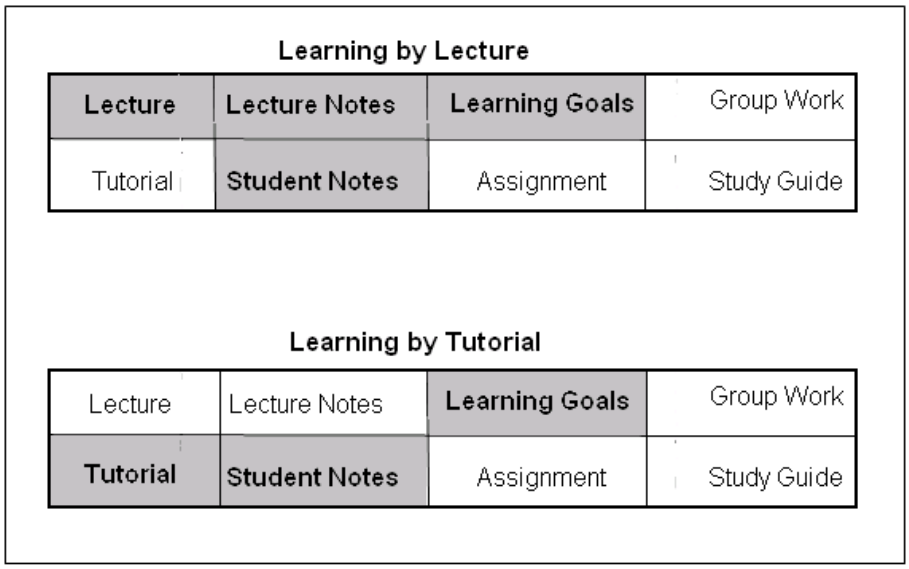


Figure 2.1: Learning Components in Two Learning Modes

(Johnson 2005, p. 101)

[2] Delphi is an object-oriented programming language originated from Object Pascal. Borland Delphi provides an IDE (Integrated Development Environment) for developing Delphi applications. Borland Delphi 7 Enterprise (Enterprise version 7) is used to build prototypes for both IMMEDIATE and this project.

The learner can interchange between these learning modes at any time when s/he logs in. Each course in the database is represented as a hierarchical structure (a tree) - sections, topics and concepts. There are several states for each node (i.e. a section or a topic). The state of each node is defaulted as "Not Attempted" and can be changed to "Attempted", "Understood", "Complete", and other states with the learner's progress through the course. A student model maintains the learner's profile including the current position in the course (section and topic) and the current study mode. This lets the system have memory about each individual learner including his/her current learning status. Next time the learner logs in, the system can show him/her the place and all other relevant information s/he last left.

As a convenient communication method between the learner and the teacher, email is often used. However, sending and receiving emails needs the Internet to be accessible by the user, which might not be always the case for remote learners. IMMEDIATE stores the new message created by the learner in the learner's local machine when the learner sends the message and the Internet cannot be connected. Once the connection is available, the message will be sent to the server and then to the recipient (the teacher or another student). A learner can also ask the system questions in some predefined format. A "Learning Support" function lets the learner ask a question about a particular subject. The system will return a specific response if there is already an answer that has been stored in the database. Otherwise, the system will pass the question to the tutor for an answer.

IMMEDIATE is composed of many reusable components that can be used in different learning modes (see Figure 2.1) to provide some common functions. For a new language-learning version of the Learning Computer, it is desirable to go beyond text-based communications to audio-based communications to provide a conversation practice mode. The conversation function will be designed as a reusable component that can be embedded in learning modes in IMMEDIATE to facilitate communication between the teacher and learners. For example, it can be embedded into the "learning by collaboration" mode or into the "learning by tutorial" mode to help students to interact with other students or the tutor.

An evaluation has been conducted to demonstrate the usability and feasibility of IMMEDIATE. Many efforts are still underway to improve the prototype to become a ubiquitous learning tool. Such efforts include applying IMMEDIATE to a second

language learning course, building the Learning Computer on old or recycled computers, and installing it on a self-booting portable device such as a memory stick to provide a mobile learning appliance (Johnson et al. 2006).

An information appliance or ubiquitous computing is an ideal situation to embed technologies in a seamless and invisible way. Any efforts towards this are worthwhile in the sense of achieving better usability goals for general and majority users even if the system or tool does not result in a pure information appliance.

2.3 Communication Dimension

In this section I will present how people's communication ways have evolved with the emerging of new technologies during recent decades. Then several communication types will be discussed, followed by a discussion of communication in e-learning.

2.3.1 Communication Ways

Communication means "*the passing of ideas, information, and attitudes from person to person*" (Williams 1976). In this sense, media methods such as books, films, television, newspapers, the radio, telephone, email and website all enable communication to happen. Some of these - such as newspapers and websites - are for public communication, while others - such as telephone and email - are communication ways involving private interactions. The following discussion will be focused on the latter form of communication, where the communication content is known by the participants only, and interactions are needed throughout the process of communication.

New technologies can change and improve people's communication ways, which include two basic forms – text and voice. The handwritten letter is a traditional communication way that needs transportation. In the early 19th century, the telegraph achieved popularity for its speed for long-distance communication by carrying information over wire (Meadow 2002). Then the telephone was invented in 1876, which made it possible for two people to talk directly and simultaneously without the limitation of geographic distance.

In the simplest case, a telephone line carries analog voice from the user's premises to a local exchange, which will connect another local telephone line (Meadow 2002). The PSTN (Public Switching Telephone Network) which was originally called POTS (Plain

Old Telephone System), connects exchanges worldwide. Thus, two telephones from anywhere in the world can be connected with each other via the PSTN.

Internet technologies have further reformed and enriched people's communication methods. The Electronic mail (email) uses the Internet technology for transmitting a letter between computers. Because of email, many people now rarely use traditional letters, although they are still used on some occasions. Because of the flexibility and speed of email, it is widely used in communities and business for people's communication.

The other form of communication brought by Internet technology is the online forum, which is very popular for group discussions. Each person in a group can easily review what others have expressed and publish his/her own idea.

The Internet also brings a new way for voice communication by using VoIP (voice over Internet Protocol) technology. Users on computers or VoIP-enabled telephones can talk with each other by using some VoIP software such as Skype, Windows Live Messenger and Yahoo Messenger (n.d.). The benefit of VoIP is that it is cheaper for long distance communication than using landline telephones. The voice quality of VoIP can be very high. The main problem is that it generally requires broadband Internet connections.

VoIP software - such as Skype - can include functions such as calling a telephone or mobile phone, text messaging, voice mail and call forwarding with quite low prices. The voice mail is a voice message sent by somebody to another person or a group of people without holding a direct conversation with them. It is also used to leave a voice message when the other party is not available. Call forwarding can forward the call to the user's telephone or mobile phone when s/he is offline.

A buddy list can be displayed on the interface to show the present status (such as busy, available and away) of each user, so that the user can choose the communication partner. Text or audio chat lets people communicate with each other by plain text messages or real voice, with the choice of video function. Such a combination gives great flexibility for users to choose different communication methods under different circumstances.

The mobile phone has become enormously popular over the last 10 years for its compactness and portability. It uses wireless technology, which has developed from the first generation (1G) transmitting analog voice in the 1980s to the current third generation (3G) transmitting digital voice and data (Talukder & Yavagal 2007, p. 5).

High-quality voice, messaging, multimedia and website browsing services are now available on mobile phones. In addition to the "any time, anywhere" voice communication, the short message service (SMS) makes text communication possible between two mobile phones flexibly due to its omnibus, stateless, asynchronous, self-configurable, and other features (Talukder & Yavagal 2007, pp. 168-169).

Telephones and mobile phones enable the conducting of synchronous communications between people. Here, "synchronous" means that the listener can understand the talker almost simultaneously and they both have the right to talk at any time. On the other hand, there are asynchronous or half-duplex communication methods such as PTT (Push-To-Talk). PTT is a traditional half-duplex conversation method using radio channels. In PTT communication, two parties will take turns to speak and listen.

A walkie-talkie is a portable wireless handset of two-way radio PTT. Only one user is allowed to talk at a time by pressing and holding a dedicated button on a walkie-talkie handset. The user releases the button when s/he has finished talking. The voice is immediately transferred to other call participants in real time (Hotcoding 2005). Walkie-talkies are very useful tools to let a small group of people working in different places communicate with each other. PoC (PTT over Cellular) adds the PTT feature on the cellular (mobile phone). Similarly to the walkie-talkie, the sender needs only to press a key on the device and it facilitates the communication and collaboration among a small working group. It uses VoIP technology, is full-duplex and has potential for long-distance communication. Nextel initiated the PoC service in North America and achieved success there (Lawson 2004). It also provides other services such as user presence, sending an email with a voice attachment, and a Multimedia Messaging Service for sending pictures, video and voice messages (Siddall 2005). Other innovative communication ways - such as video conferencing - are emerging by the application of advanced information technologies.

Some communication methods from the user's perspective have been discussed. From the system view, communication happening between two nodes can be categorized as the following three types (Gallo & Hancock 2002, pp. 45- 49).

- *Serial and parallel communication*. This category of communication means that signals or data are sent bit by bit or all bits at a time. Serial is slower than parallel.

- *Synchronous and asynchronous communication*. Synchronous communication means that the communication between two modes can be monitored by each node, while in asynchronous communication one node does not know when the other node begins and stops sending information. Synchronous communication is more efficient than asynchronous. When sending information in asynchronous communication, the message needs a start indication and a finish indication to notify the receiver of the start and stop points.
- *Simplex, half-duplex and full-duplex communication.* In telecommunication, there are three different communication modes - simplex, half-duplex and full-duplex. The simplex mode allows only one direction communication (from a sender to a receiver). The half-duplex mode enables communication in both directions but only one direction at a time. The full-duplex mode refers to communication that can happen in both directions between two end-points simultaneously. All modern modems allow traffic in both directions at the same time by using different frequencies for different directions (Tanenbaum 2003, p. 129).

Conversations must be serial or one by one, otherwise two parties cannot understand each other. A real-time face-to-face or telephone conversation is synchronous and full-duplex. Since it is synchronous, the listener can know what the speaker says simultaneously; and it is full-duplex so that anyone can speak at any time to interrupt the speaker. The PTT communication is serial, asynchronous and half-duplex. In people's oral communication, synchronous is considered equal to full-duplex and asynchronous is considered equal to half-duplex.

2.3.2 Communication in E-learning

"Educators need to be able to respond flexibly, skilfully and professionally to the idiosyncratic needs of particular learners in particular classrooms" as education is not only about teaching theory or knowledge, but also about practising skills (Marland 1997). In an e-learning environment, communication between learners and the tutor becomes especially important as the learner is isolated from the tutor and other learners. The tutor "may intervene to correct misconceptions, answer questions, challenge the

learners' understanding or promote dialogue", while learners may share their understanding and collaborating with each other (Thomas 2001).

Synchronous and asynchronous communication methods are often adopted between learners and the teacher in e-learning. One example of synchronous learning is the virtual classroom, where the teacher and learners "share documents, web sites and ideas in text/on the whiteboard" (Anderson 2005) for interaction and collaboration. Windows Live Messenger (or MSN) and Skype, for instance, are stand-alone synchronous communication tools for text or voice chat. Audio and video synchronous communication is efficient but needs broadband for all participants. Participants need to join the synchronous communication at the same time from different places. Synchronous communication "seems particularly useful for supporting task and social support relations, and to exchange information with a lower degree of complexity" (Hrastinski 2007).

Asynchronous communication - such as the discussion forum, email, and one-way audio or video - is very flexible and useful in e-learning. Unlike synchronous communication, participants do not need to join the communication at the same time. There is also no high bandwidth requirement, which is the case in synchronous communication. A most common example of asynchronous communication is the discussion forum, in which learners have more time to think over and reflect before giving answers to others. It is therefore considered more suitable for discussing complex ideas (Hrastinski 2007).

Computer-mediated communication in e-learning has been discussed above. Mobile learning is a new area of e-learning due to the popularity, mobility and multimedia ability of mobile devices. A study (Gui 2003) in mobile learning for second language learning reveals some limitations of it - such as the small sized screen and the distraction caused by unpredictable location.

In second language learning, practising speaking skills is an important - but challenging - task. In the next section I will describe how voice communication could possibly be prototyped to meet the requirements of extramural second language learning.

2.4 Toward an Initial Conceptual Model

As mentioned in Section 2.2, the purpose of the prototype is to design a reusable conversation component for IMMEDIATE to use in the extramural second language

course. The conversation should happen between either the learner and the teacher, or two learners - it is a peer-to-peer conversation. The peer-to-peer conversation component should be computer-mediated because IMMEDIATE is implemented on a computer. Since IMMEDIATE deals with both broadband and narrowband connections for learners, both synchronous and asynchronous voice communications can be included in the component. A bimodal approach can be adopted – using synchronous conversation when both users are on fast Internet connections and asynchronous conversation when at least one user is on a low Internet connection.

From the discussion in Section 2.3.1, we understand that VoIP is inexpensive for long distance conversation. It is suitable to use VoIP technology in e-learning systems for voice communication between extramural learners and tutors who might live anywhere in the world. So the VoIP technology will be used in the conversation component. To be embedded into IMMEDIATE in the future, the component should be designed to be compatible with the architecture of IMMEDIATE.

Because IMMEDIATE is designed in the light of invisible computing, the user is not supposed to be bothered with system configurations or choices. Rather than letting the user select the synchronous or asynchronous mode, the system should automatically adapt to either the synchronous mode or the asynchronous mode according the user's connection speeds. As discussed in Section 2.1.2, the component should have a simple and intuitive user interface.

Based on the foregoing discussions, an initial conceptual model for the prototype has been proposed:

- The goal of the prototyping is to design a reusable peer-to-peer computer-mediated conversation component for IMMEDIATE to be used in a second language extramural study course. The component is designed as a tool so that it can be tested and evaluated easily in this project.
- VoIP technologies will be used in the conversation component.
- Implementation of the component will comply with the IMMEDIATE architecture so that it can be embedded and used in IMMEDIATE in the future.

- A bimodal approach - synchronous and asynchronous conversation modes - will be used. The component will automatically adapt to one of the two modes according to each user's Internet connection speeds - using synchronous conversation when both users are on fast Internet connections, and using asynchronous conversation when at least one party is on a low Internet connection. Thus, users will not be aware of different bandwidths and will focus on the learning task itself.
- The user interface should be simple and intuitive. The user's presence (online status) should be displayed on the screen and dynamically updated. The prototype must have basic functions for the conversation.

This initial conceptual model is the guideline for the design of the prototype. Its specification will be further refined (Chapter 4) after some investigation into Internet and voice communication technologies (Chapter 3).

2.5 Summary

The development of information technologies has reached the stage where the information appliance needs to take the role of the all-in-one computer. General users do not care about technologies but are interested only in their own tasks. They need to focus on their activities by using the information tool more easily and naturally without having to adapt to the tool and the technology. This can be achieved by embedding the computing technologies in invisible and ubiquitous ways. Software design can hide all underlying computing technologies to present people with a simple interface.

Current distance learning systems in universities are mainly learning-provider-centred rather than learner-centred. In addition, they are web-based educational systems which require broadband connections. Intelligent tutoring systems, on the other hand, are learner-centred, but they are for research purposes. However, some learners - especially those living in some rural areas - have only unstable, low speed dial-up connections to access the Internet. In addition, some learners have little computer literacy.

To enable immediate learning – "any time, anywhere and anybody" learning, the Learning Computer has been proposed and prototyped. It is a learner-centred learning system dealing with accessibility and usability problems for learners with limited

software and hardware knowledge and available finance. A prototype IMMEDIATE has been evaluated to demonstrate the feasibility of the Learning Computer concept.

An initial conceptual model has been proposed for an application of IMMEDIATE in an extramural second language learning scenario. The prototype will implement a reusable peer-to-peer computer-mediated conversation tool that will adopt VoIP technologies. When users use the tool for conversation, the system will automatically adapt itself to either the synchronous or asynchronous mode according to different Internet connection speeds. The user interface should be simple and easy to use.

Chapter 3

Voice over IP

VoIP means Voice over Internet protocol - a technology that transmits the voice through the Internet. It has been widely adopted in computer systems for cheap and flexible communication. To provide a second language e-learning solution, VoIP technology will inevitably be considered. However, VoIP is a major topic – it includes many networking and telecommunication issues. On this chapter I will try to clarify basic VoIP mechanisms and concepts such as the Internet, audio signal processing, speech coding, QoS (Quality of Service), and VoIP protocols.

3.1 Internet

There are two kinds of communication networks - the *circuit switching* network and the *packet switching* network. The *circuit switching* network lets two parties use the communication channel exclusively - others are not allowed to use it before it is released. It is reliable but expensive. PSTN is a worldwide public circuit switching network, which is designed for telephone calls.

In *packet switching*, the information is divided into small pieces and is then packed together in the form of packet for transmission across shared communication channels. In other words, packets from different sources can share the packet switching network. Each packet has source and destination addresses; and can be routed through the network independently. The Internet is a worldwide public packet switching network.

There is a trend to convert the PSTN infrastructure from circuit switching to packet switching, which will depend on whether or not VoIP can succeed at improving its quality (LINFO 2005). The development of wireless technology gives a good example of the shift from circuit switching to packet switching. The 1G (first generation) and 2G (second generation) technologies use circuit switched voice and data, while 3G (third generation) technology uses packets for both voice and data.

The Internet is the largest worldwide communication network that links governments, universities, companies, communities and other organisations. Any personal computer can become a part of the network via dial-up, ADSL (Asymmetric Digital Subscriber Line), cable modem, and wireless connection. Two computers in two different local area networks (LANs) send information to each other via routers. Routers connect LANs in a particular geographic area into a regional network, which is connected to other regional networks via an Internet backbone (Gralla 2004, pp. 12-13). The Internet backbone, which is built by governments and large companies, carries data over long distances (e.g. between continents) at extremely high speed.

3.1.1 Internet Protocols

The TCP/IP Protocol Suite (Internet Protocol Suite) is a four-layer network protocol model (or five-layer model if the physical layer is separated from the data link layer). The model defines a set of protocols for communication over the Internet and other networks. Internet standard RF1122 (Braden 1989) describes the four-layer model. Figure 3.1 (ePipe, n.d.) gives some example protocols for each layer of the model.

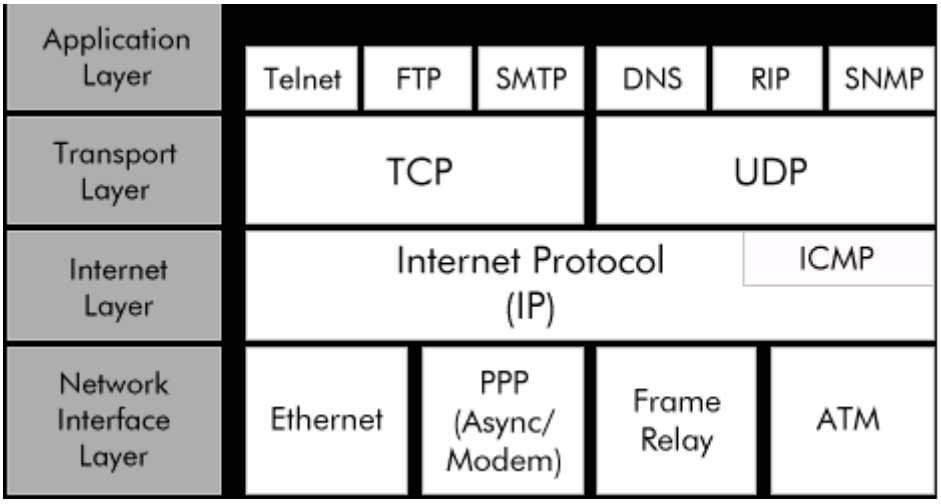


Figure 3.1: The Four-layer TCP/IP Model (ePipe, n.d.)

There are two kinds of protocols in the application layer - *user protocols* such as FTP (File Transfer Protocol) that provide services directly to users, and *support protocols* such as DNS (Domain Name System) that provide common system functions (Braden 1989).

TCP (Transmission Control Protocol) is the main transport layer protocol for reliability and accuracy. TCP is used to break down a block of data into small pieces or packets. It reassembles the arrived packets in correct order and ensures that there is no loss of data and that all data are in correct order. The UDP (User Datagram Protocol) omits some services and does not guarantee the intactness of the received data. It is unreliable, stateless, not accurate and without order, but is faster than TCP. Therefore, UDP is often used in time-sensitive applications with a huge amount of data such as the streaming of multimedia - for example audio or video.

A packet includes a *head*, a *body* and sometimes a *trailer*. The *head* contains the control information such as the destination address and port number. The *body* is the user data. Figure 3.2 (Wang 1997) shows the data encapsulation in the four-layer TCP/IP model. A lower layer protocol adds a header before the upper layer data. That means that each lower layer provides services for the upper layer.

IP (Internet Protocol) is the key protocol in the Internet or network layer. IP is responsible for finding the most suitable route for each packet from a source to a destination, to balance the Internet load and avoid congestion. This means that different packets from the same source might follow different routes to reach the same destination. An IP network layer packet, as shown in Figure 3.2, can be called as a *datagram*.

In the four-layer TCP/IP model, the data link layer includes the physical network layer. The data link protocol - such as PPP (Point-to-Point Protocol) - sets up a direct connection between two nodes including configuration, authentication and teardown (termination) of the connection. Then it chooses one or more network layer protocols for the datagram to be sent from each of two end-points (Simpson 1994). The data link layer provides transfer of frames across the physical link (Leon-Garcia & Widjaja 2004, p. 45).

The physical layer in the 5-layer TCP/IP model is to transport a raw bit stream from one machine to another (Tanenbaum 2003, p. 38) via the physical media such as the copper

wire, fibre optics and radio. It defines characteristics of transmission bit data, for example, digital/analog, asynchronous/synchronous, half-duplex/full-duplex and data rate (bandwidth) based on different physical media (Miller 2006, p. 26).

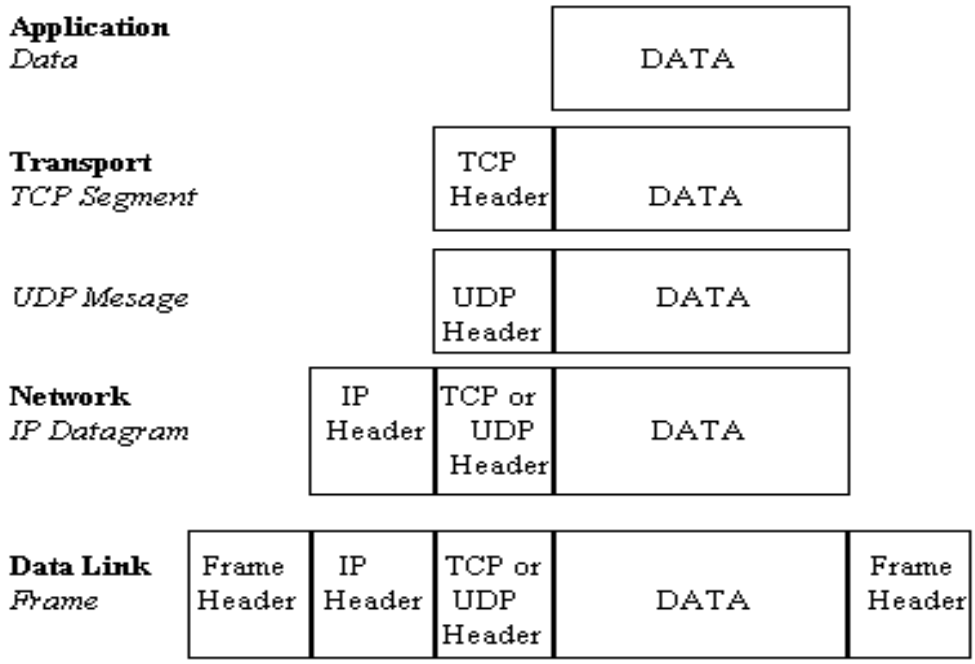


Figure 3.2: Data Packaging in Different Layers of the TCP/IP Model (Wang 1997)

3.1.2 IP Address and Routing

The IPv4 (Internet Protocol version 4) address is 32 bits, often written in four parts such as 130.123.179.59 where each part is represented by 8 bits or 1 byte. Each address maps to a node (such as a computer, printer or other device) in the Internet. Because there are too many nodes connecting to the Internet, this 32-bits (4 bytes) IP address is not sufficient for identifying all devices on the Internet. Although a new IPv6 (Internet Protocol version 6) standard has been designed to solve this problem, at present, IPv4 is still the de facto standard in the network layer of the TCP/IP protocol suite.

Computers and devices inside a private network use private IP addresses to alleviate the IP address shortage and improve security. NAT (Network Address Translation) deals with how computers in a private network access the Internet by translating between private addresses and public addresses (Carr & Snyder 2006, p. 209). When a computer

user in a private network wants to access the Internet, the private IP address will be mapped into a public IP address. The mapping is stored so that return packets can be forwarded to this computer. When an internal computer provides a public service to the outside world, it needs a public IP address for external users to access.

A user's IP address that is gained from an ISP (Internet Service Provider) is usually different each time, although some ISPs can also provide static IP addresses. The dynamic IP address saves the IP address space, because the ISP needs to hold only a limited number of IP addresses.

When one computer is connected to the network, other computers may access a special service or application on it by an opened port. The head of an IP packet such as the TCP or UDP packet includes a port number. Port numbers (16 bits) are used to distinguish different applications or services on a computer. An application on the computer will run if the packet belongs to the corresponding port. Common services or well-known applications have their port number (0 - 1023) pre-defined for running as servers, for example, HTTP using port 80, FTP using port 21. Ports 1024 - 49151 are registered ports and used by end user applications (IANA, 2008).

3.1.3 Internet for Residential Users

Data transmitting speed divides the Internet connection into two categories – *narrowband* and *broadband*. *Narrowband* usually refers to dial-up connection, while *broadband* refers to Cable, DSL (Digital Subscriber Line), satellite or wireless connection.

Home users usually use dial-up or ADSL (Asymmetric Digital Subscriber Line) via telephone lines to access the Internet. The dial-up connection uses a modem to connect to an ISP to build a modem-to-modem connection (Figure 3.3). When a user dials up to connect to the Internet, a modem connected to the user's computer transforms the digital signal to an analog signal for travelling through the telephone line. On the ISP server side, another modem receives the analog signal and changes it into a digital signal. Once the dial-up connection has been set up, no telephone call come in as the line is occupied by the Internet connection, and vice versa.

Dial-up connection speed is determined by the modem type, as well as by other factors such as line noise and congestion. Although the theoretical maximum speed of Internet

dial-up access is 56 Kbps, the actual dial-up speed is normally no more than 53 Kbps due to the modem type and line noise (Gralla 2004, p. 43). Figure 3.3 illustrates computers accessing the Internet via dial-up connections. It depicts the signal changes (digital-analog-digital) among computers, ISPs and the Internet.

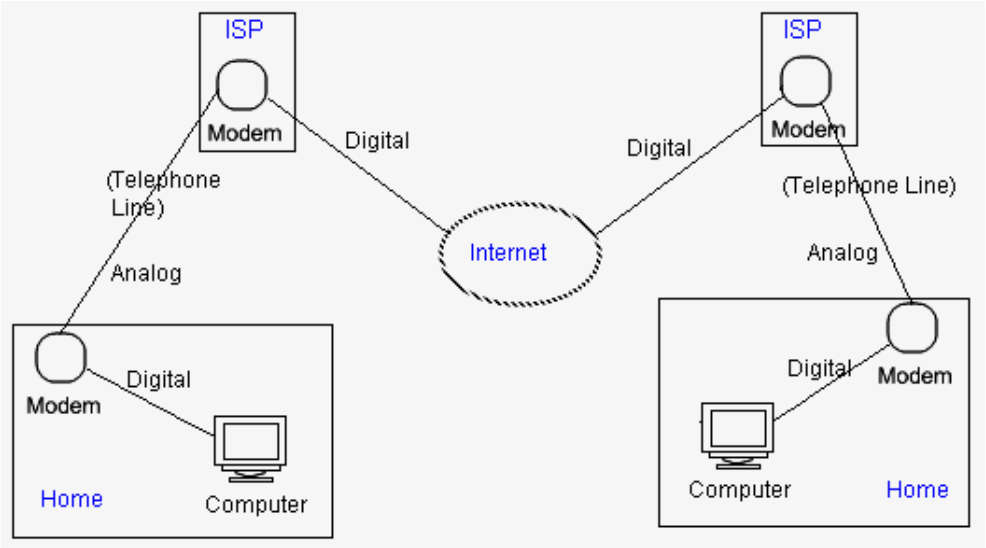


Figure 3.3: Computers Accessing the Internet via Dial-up Connections

DSL is an "always-on" technology. It differs from dial-up, in that it carries both data and voice through the same telephone line at the same time by using high frequency for data and low frequency for voice. A splitter is installed in the user's premises to separate the data from the voice and send them to different devices. Therefore, DSL has much higher speed (128 kbps - 52 Mbps) than dialup. One form of DSL is ADSL, the download speed of which is much higher than its upload speed. Because of this, ADSL is very popular for home and business usages that involve more download tasks (such as browsing web sites) than upload tasks. However, ADSL is not available in some rural areas because it has a distance limitation (Mitchell, n.d.). For users living in such areas, dial-up connection is still the only possible way for cheap Internet access.

3.2 VoIP Concepts

In the PSTN, the voice is transmitted as an analog signal over the telephone line. During the travel, the analog signal needs to be strengthened by amplifiers, because the signal

will become weak and attenuate (Carr & Snyder 2006, p. 17). However, the amplifier also boosts the noise. The benefit of transporting the voice as a digital signal is that the digital signal can be regenerated and amplified without the noise being boosted. Hence, received voice will be much clearer when it is transmitted as a digital signal. The drawback of voice communication a digital signal is that it demands broadband Internet connection; otherwise there might be voice delay, which is not acceptable in real-time communication. The main reason of the popularity of VoIP is its cheap price for long distance communication, which is usually expensive when a landline telephone is used.

In this section some audio signal processing and speech compression issues will be presented to enable the reader to understand how voice is transmitted through the Internet, followed by a description of the QoS of VoIP to assist understanding of the factors that affect the quality of the VoIP service.

3.2.1 Audio Signal Processing

Sample rate or *frequency* (Hertz) refers to the number of samples of an analog signal taken per second to produce a digital signal (Leon-Garcia & Widjaja 2004, p. 118-119). In general, the higher the sample rate, the less the loss of information. A sampling rate of 8 kHz can achieve telephone quality voice. *Sample size* or *resolution* determines how many bits of information describe sound per sample. Sound samples usually contain 8, 16, or 24 bits per sample. The more bits, the more accurate the sound will be (Dove 2003). A *channel* is a stream of audio. A sound file is *mono* when generated by one channel and *stereo* when generated by two channels (left and right). The more channels, the higher the sample rate and sample size, the better the quality of sound and larger the size of the audio file.

The *bit rate* is the number of bits that are conveyed or processed per unit of time (usually per second). The sample rate, sample size and the number of channels determine the bit rate. Usually the lower the bit rate, the worse the sound quality because of some loss of original information, although such differences may be very small.

When a user speaks into a microphone connected to a computer, the sound card in the computer will receive a continuous analog signal from the microphone and convert it into discrete digital signals, which might be then transmitted over the Internet. The raw

digital signals (audio streaming) need to be compressed (encoded) before transmission and be decompressed (decoded) on receipt by *codec*s (compression/decompression algorithms). Figure 3.4 (Hellwig 1989) shows the procedure of the voice signal transmission from a sender to a receiver via a network. The A/D or D/A converter changes the voice between analog and digital signals. The encoder compresses samples in a selected codec and puts them into a frame every certain period of time (New Networks, 2005). The decoder decodes the compressed data using the same codec.

Figure 3.4: Audio Signal Transmission (Hellwig 1989)

Wave and MP3 are two popular audio formats. *Wave* is the default format for digital audio files on Microsoft Windows and other platforms. By default, it uses PCM (Pulse Code Modulation) coding to generate uncompressed digital audio data. Therefore, wave files usually are of large size but have good sound quality. Codecs (such as GSM6.10) can also be used to generate compressed wave files.

MP3 (MPEG-1 Layer 3) is a standard that compresses a sound stream into a very small file without losing the sound quality (Bellis n.d.). Because of its high efficiency and loose licensing terms, it has become popular for an Internet music format (BBC h2g2 2000). MP3 encodes sound at 128 Kbps, which is suitable for broadband Internet applications.

3.2.2 Speech Coding

The purpose of digital signal (such as speech, audio or image) coding (compression) is to decrease the bit rate while maintaining a specified level of signal quality, delay and implementation complexity (Jayant 1997, p. 2). Because the size of a voice, audio or video stream is often huge, low bit rate coding is critical to minimise the bandwidth requirement for multimedia stream transmission.

The voice usually can be compressed more than music, because people can understand with some loss in voice signal. The speech codec bit rate varies from 4 to 64 Kbps. PCM is a standard 64 kbps coding method used for digital telephony (ITU 1988). It provides clear voice with 8-bit sample size at 8 KHz sample rate without compression. With the development of speech coding technology, now the voice bit rate can be around 4kbps via narrowband while keeping reasonable voice quality and performance. CELP (Code Excited Linear Prediction) is the most efficient coding method for high quality speech coding at bit rates around 4.8 kbps (Salami 1989). Many international standards in the 4.8 – 16 kbps range (such as G.723.1 and Speex) are based on CELP or its derivatives such as ACELP (Hersent et al. 2005b, pp. 61-67). Three narrowband speech codecs are listed in the following:

- G.723.1 - is an ITU-T (Telecommunication Standard Sector) standard. It is the most efficient low bit rate codec (5.3 or 6.3 kbps) with very good voice quality. However, it is a multi-party owned patent and its licence seems expensive (Hersent et al. 2005a, p. 24). ITU-T defines application layer protocols to provide audio/visual communications on any packet network (ITU, n.d.).
- GSM 6.10 – is a standard designed for low- to mid-bit rate (13 kbps full rate) voice in cellular digital mobile systems in Europe. It has an open implementation source (The GSM 06.10 Lossy Speech Compression 2006). It is not only used in mobile phone systems, but is also widely used in desktop voice applications.
- Speex (Xiph 2006*)* – is a new, free and open source codec which provides a free alternative to other CELP based commercial speech codecs. It has variant bit rates ranging from 2 to 40 Kbps. Because of its very low bit rate, good voice quality and free availability, it is well suited to VoIP applications.

3.2.3 Quality of Service

The quality of real-time stream (such as voice or video) transmission via networks is determined by many factors. The QoS (Quality of Service) quantitatively describes attributes of the Internet service. It can be measured by parameters such as delay, jitter, loss ratio and error ratio (Kasera et al. 2007, pp. 262-263).

Delay (or *latency*) is the amount of time it takes for a packet to travel from source to destination (VoIP Foro, 2006). Delay can occur for many reasons. One reason is the encoding and decoding procedure and transmission latency through the network (Jayant 1997, p. 17-18). Transmission delay is usually caused by insufficient bandwidth. Delay is important in real time conversation, because when the delay exceeds 400ms, synchronous conversation becomes impossible and the conversation has to be half-duplex communication (Hersent et al. 2005b, pp. 111-112). Delay will cause echo and jitter.

Jitter is the variation of delay over time (Kasera et al. 2007, p. 262). As different packets may take different times to reach the destination, the order of arrival of packets may be different from the original order. This causes breaking and disordering of the voice. The solution is to use a jitter buffer - however, the jitter buffer is not acceptable in real-time applications involving interactions such as telephony, as it will cause further delay when buffering (Tanenbaum 2003, p. 396).

Another parameter for VoIP QoS is *echo*. Echo occurs because of voice delay and the full-duplex communication via which the voice will bounce back from the listener's microphone when it arrives at his/her speaker. It is measured by the delay and the intensity of the voice. When it is a half-duplex communication such as PTT, there is no echo.

Packet loss occurs when there is congestion on the packet's path. In voice applications such as IP telephony, a small degree of packet loss is allowed as people can tolerate it without misunderstanding the meaning of speech. As the transmission channel has some noise that will cause error in packets, the error ratio parameter is introduced in QoS. Because the noise or error rate is very low in VoIP, usually it can be tolerated (Leon-Garcia & Widjaja 2004).

RTP (Real-time Transport Protocol) and RTCP (Real-time Transport Control Protocol) are application layer protocols in the TCP/IP model which are often used for real time audio or video stream. They are typically used on top of UDP, which is an unreliable protocol for real time stream transmission. RTP is designed for solving the jitter problem introduced by IP networks by adding the timestamps and sequence numbers in the header (Hersent et al. 2005a, p. 5). However, RTP does nothing to the quality of the transmission (such as delay), for which RTCP is responsible.

The bandwidth of the network might be the most important factor that affects the VoIP application, since low bandwidth can cause severe delay and jitter. The following may be used to calculate the bandwidth requirement for voice transmission.

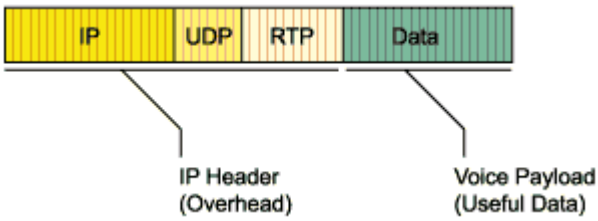


Figure 3.5: IP Header for VoIP packets (Newport-networks, n.d)

Figure 3.5 (Newport-networks, n.d) depicts an IP header including IP, UDP and RTP headers put on the actual voice data. The total size of the IP header is 40 bytes including a 20 bytes header for IP, 8 bytes for the UDP header and 12 bytes for the RTP header (New Networks 2005). Samples are accumulated during a sample period (e.g. 20 ms) and then encoded to form a packet. A frame usually includes one packet, but sometimes it can have two packets.

The minimum bandwidth requirement for using a specific codec can be estimated. In the case of GSM 6.10 (where the bit rate is 13 kbps), each frame contains one packet with a sample period of 20 ms. The size of the actual voice data in a GSM 6.10 packet is 32.5 byte (13000 * 0.02 / 8), while the IP header is 40 bytes. Therefore, the total size of one packet is 72.5 byte. This means that the minimum bandwidth needed for a GSM 6.10 packet is 29 Kbps (72.5 * 8 / 0.02 = 29,000 bit/s). In calculation of this value the extra network headers that might be added on the packet have not been taken into consideration.

As each packet contains at least 40 bytes IP header, selecting longer voice data (increasing the sample period) for each packet will save bandwidth but cause more delay. Conversely, shortening the length of voice data in each packet will reduce delay but may introduce bandwidth shortage. As both delay and bandwidth are problems for narrowband voice applications, selecting the sample period is not very important in such applications (Hersent et al. 2005a, pp. 103-104).

3.2.4 VoIP Protocols

Some mechanisms of VoIP have been discussed in the previous section. Internet telephony uses VoIP technology to carry out telephone functions using protocols such as H.323 and SIP (Khasnabish 2003). By using VoIP, cheap telephone calls can be made among computers and special VoIP telephones connected to the Internet (FCC n.d.).

H.323 (ITU 2006), a recommendation of ITU-T, is the first VoIP protocol, which has been widely used in the telecommunication sector. Because it was designed for video conferencing, it has become a complicated multimedia communication system - it is large and robust but expensive, complex and inflexible. It is capable of dealing with different types of LANs and avoiding congestion. To implement telephone and multimedia functions via the Internet and PSTN, many devices and protocols are needed. Instead of defining its own protocols, H.323 architecture references many existing protocols (Tanenbaum 2003, pp. 686-688).

The Session Initiation Protocol (SIP) is an application-layer control protocol for multimedia conferencing over the Internet. It is designed by IETF and described in RFC3261 (Rosenberg et al. 2002). It uses mainly UDP, which is more efficient than TCP (used by H.323). Because of the Internet boom, SIP has become a very simple, flexible, scalable and expansible protocol by not defining or dealing with all aspects of VoIP application. SIP is suitable for computer-mediated multimedia applications due to its simplicity and flexibility, while H.323 is deployed mainly in PSTN-Internet telephony.

Method	**Description**
INVITE	Request initiation of a session
ACK	Confirm that a session has been initiated
BYE	Request termination of a session
OPTIONS	Query a host about its capabilities
CANCEL	Cancel a pending request
REGISTER	Inform a redirection server about the user's current location

Table 3.1: SIP Methods (Tanenbaum 2003, p. 690)

Since this project is to implement the conversation function between computers, it is of interest to discuss SIP further. SIP is a text-based protocol. It defines five basic dimensions for communications - user location, user availability, user capability, session setup, and session management. A SIP message transfers a request or a response between a server and a client. Table 3.1 (Tanenbaum 2003, p. 690) shows a set of SIP methods that are used to set up a call.

SIP also defines a set of conceptual (or functional) servers such as the proxy server, the registrar server, the redirect server, the location server and the user agent (Rosenberg et al. 2002), which play different roles in enabling communication to occur.

Each SIP user needs to have a SIP address, which is a URI (Uniform Resource Identifier) or URL (Uniform Resource Locator), to locate the user who might use different machines from different locations. SIP also has a presence service to let the user know who is available, who is busy and who is off-line.

Except for H.323 and SIP, other free or proprietary VoIP protocols are used in many applications. For example, Skype uses a proprietary VoIP protocol. The most predominant advantage of Skype is its scalability – it can link computers anywhere in the world. Skype performs especially well (clearer voice than telephone) when the user has broadband Internet access. However, such computers seem more likely to be used by Skype as super nodes, which route calls of other Skype users (JANET 2006). When a user tries to use a computer with dial-up connection to communicate with another user via Skype, the voice might experience severe delay and jitter.

3.3 Summary

In this chapter I have provided an overview of VoIP technologies from Internet, audio signal processing, speech coding, QoS and VoIP protocols aspects. VoIP can be seen as a movement of the traditional telecommunication framework to the Internet architecture. It uses the Internet to carry the voice stream between computers and/or telephones, its main benefit being its low cost for long distance voice communication, which is expensive if traditional PSTN is used. VoIP function can be embedded into software to provide multimedia solutions such as VoIP conferencing. The use of low bit rate speech codecs is critical in VoIP applications, especially when using narrowband Internet connection that might cause delay and jitter that affect VoIP QoS.

Based on the foregoing review, in the next chapter, I will discuss how VoIP technology can be used in the project.

Chapter 4

Towards a Specification

From the previous chapter, it is understood that the key factor to implement VoIP applications under narrowband connection is the use of low bit rate speech codecs. Generally, the lower the bit rate of the codec, the faster the voice transmission and therefore the better the quality of the voice.

Time and cost limits have to be considered when deciding the prototyping methods for a master's project. There are two possible approaches to implementing the prototype. One is to directly adopt a low bit rate speech codec (such as Speex) to implement the conversation function for the prototype. This would make the implementation more flexible and possibly result in a more efficient prototype. However, this way would involve a lot of low-level programming so it was decided this would be too time consuming. Therefore, this approach will not be considered further. Another method is to utilise an existing affordable VoIP SDK (Software Development Kit) that has the ability to transmit voice via narrowband connections by using a low bit rate speech codec. This will simplify implementation by being focused on meeting the particular requirements of the project.

Because IMMEDIATE is a learning appliance that should be very easy to use, any component must be embeddable into IMMEDIATE without any other installation, configuration or start up by the user. Another requirement of IMMEDIATE is that all learners, no matter where they are and how fast they access the Internet, should be able to use it. These requirements determine what kind of VoIP SDK should be selected and used in the prototype.

In this chapter I will flesh out the conceptual model towards a specification for the prototype. The requirements of a VoIP SDK will be analysed and identified first, followed by an investigation and testing for a suitable VoIP SDK. These will result in a possible solution for integrating the synchronous and asynchronous conversation modes

into the prototype by using the SDK. Finally, a specification for the prototype will be proposed.

4.1 A VoIP SDK

The purpose in this section is to find a proper VoIP SDK to use in the prototype.

4.1.1 Requirements of a VoIP SDK

There are two kinds of VoIP SDKs for peer-to-peer conversation. One is a VoIP protocol-based client SDK (such as SIP-based SDK) and the other is a client/server VoIP SDK. The former is used usually to build client applications that can be use worldwide VoIP services. For example, Skype4Com (n.d.) is an API (Application Programming Interface) that can be used to implement VoIP clients. From Chapter 3, it is known that current VoIP services that use SIP, H.323 or other VoIP protocols cannot perform well under narrowband Internet connections, because conversation will be frequently broken and the proper communication sequence between two people will be disordered by jitter. Since this project is to deal with low dial-up connection conversation tasks, VoIP protocol-based SDKs are not suitable for the prototype.

Because IMMEDIATE has a client/server structure, the VoIP SDK should provide both client and server components, by which call procedures such as user registration, call invitation and termination can be implemented for conversation between users. The SDK client must be able to be embedded into the IMMEDIATE client; and participation in peer-to-peer communications must be controllable via the IMMEDIATE server. Similarly to H.323 or SIP, the basic function for the server should be user registration. The server, which will be located in the university network, should also let two users, no matter where they are, connect with each other and conduct a conversation.

Because IMMEDIATE focuses on providing narrowband solutions, the SDK should use a low bit rate speech codec for voice transmission. The SDK should also ensure reasonable voice quality. Delay, jitter and echo should be controlled to be minimal. Since IMMEDIATE has been developed using Delphi 7, the SDK should be used in Delphi 7 IDE (Integrated Development Environment). The SDK should also be affordable.

In summary, the VoIP SDK should have narrowband computer-mediated peer-to-peer conversation ability. The server should have registration, call setup and termination functions based on a client/server structure. It is ideal if the SDK also has other functions such as echo cancelling, jitter buffering, NAT, and UDP.

4.1.2 A Peer-to-Peer VoIP SDK

Several commercial or free communication tools and VoIP SDKs were investigated including Conaito VoIP Enterprise SDK (2005), VaxVoice Proxy SDK (n.d.), Skype4COM (n.d.), Acrobat Connect Pro (n.d.) and GIPS (2007) VoiceEngine. Only Conaito VoIP Enterprise SDK and VaxVoice Proxy SDK meet the requirements that have been discussed in Section 4.1.1.

Acrobat Connect Pro (formerly known as “Breeze”) provides web conferencing and e-learning solutions. It is based on a proprietary speech codec of the Flash Player. Although it provides a VoIP SDK, it has software and bandwidth requirements and needs a web browser and Adobe Flash. The minimum bandwidth requirement for participants is 56 Kbps, but for the presenter it is broadband. Skype4COM is an API based on a Skype VoIP protocol, which is not suitable for narrowband applications (see Section 3.2.4). GIPS VoiceEngine’s voice processing technology is used in many major VoIP providers such as Google Talk, Yahoo and Skype. It also provides VoIP SDKs but is expensive.

The Conaito VoIP Enterprise SDK is a client/server based peer-to-peer VoIP SDK. It is claimed that it can be used in both broadband and narrowband connection by using the GSM 6.10 codec. Because Conaito SDK and VaxVoice SDK are very similar, only the former was fully tested in this project. The following will introduce more features of Conaito VoIP Enterprise SDK.

The SDK has encapsulated in its methods many network and VoIP functions such as voice compression/decompression, UDP full-duplex transmission, NAT, echo cancellation and jitter buffering. It includes additional functions, such as adjusting the volume of the microphone and the speaker, and recording. The SDK also provides functions such as registration and call setup. The user needs to register with the server first so that the server can obtain the user’s information (such as user’s name, IP address and port number). The call invitation function lets one user invite another available user.

If the invitation is accepted, it can build UDP-based direct communication between two users.

Figure 4.1 (Conaito 2005) depicts the procedure of call setup between two clients after they have registered with the server. The SDK is NAT or firewall friendly, which means that two users are able to communicate with each other even when they are not in the same network. To achieve this, the server must have a static public IP address to let users in different networks access the server. A special port number will also be opened for outside users to access the server.

Another important feature of the VoIP SDK is that it enables voice conferencing by directing voice stream from the sender to the listeners via the server. This saves the user's bandwidth by sending the voice to the server only without repeatedly sending it to every listener. Furthermore, this method can be combined with P2P direct voice transmission to meet some special requirements. Although this feature will not be used in this project, when designing the tool, the possibility of expanding the tool for use in group discussions in the future was kept in mind.

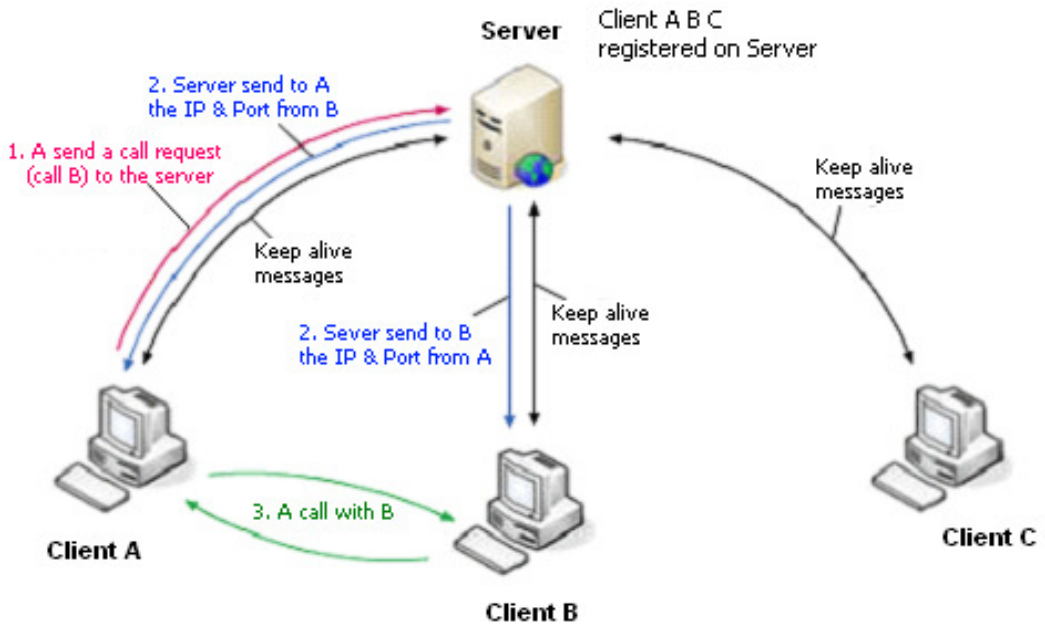


Figure 4.1: Procedure of Call Setup in Conaito (Conaito 2005)

4.1.3 The Threshold of Low Speed and High Speed

The SDK was tested using a sample application provided by Conaito. The sample application can conduct direct peer-to-peer conversation. It functioned well when both parties were on broadband connections, as well as on dial-up connections when the speed was over 50 kbps.

The Conaito SDK provides a method for encryption that is turned on in the sample application. However, when encryption is selected to enhance the security of the voice content, the size of the voice stream will be increased causing greater delay on low bandwidth connections. Voice security is not necessary in this project because learners are just practising a second language. The SDK was tested without encryption with one party on ADSL and another on 37.2 kbps and 40 kbps dial-up connections. The conversation between two users was conducted well with good voice quality (without severe delay, noise or echo). However, when it was tested at the dial-up speed 33.6 kbps, there was severe voice delay with broken words and sentences.

The connection speed mentioned in this thesis is displayed by the Windows desktop when a connection is built. The access speed might shift down or up during the call period according to actual line conditions (Compaq 1998). In 3.2.3, it is calculated that at least 29 Kbps bandwidth is needed to use GSM 6.10 to transmit voice, without calculating other possible information in it.

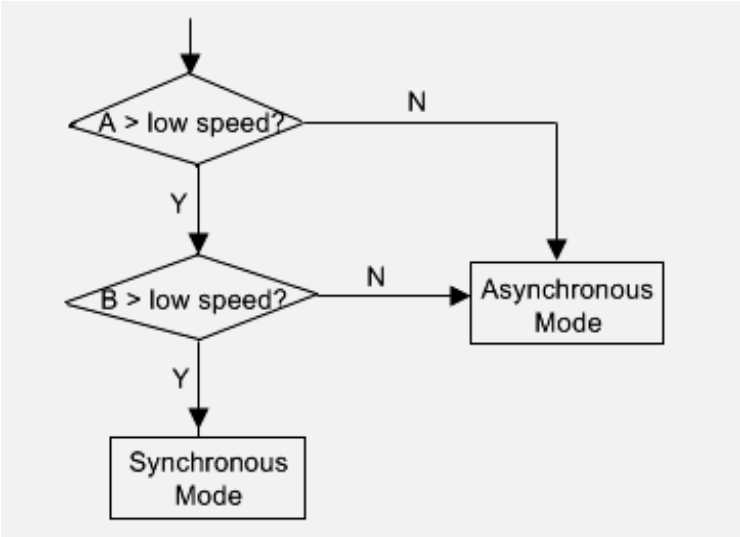


Figure 4.2: Two Peer-to-Peer Conversation Modes

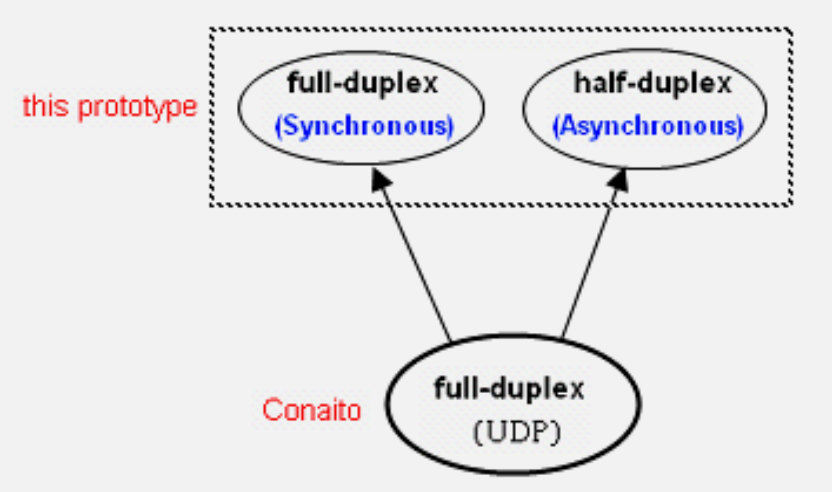


Figure 4.3: Relationship between the Prototype and Conaito SDK

In Chapter 2, a bimodal approach for the prototype of the project was decided. The prototype needs to adapt automatically to synchronous/asynchronous conversation modes according to different levels of connection speeds. Based on the above test and considerations, 40 kbps was selected as a "safe" threshold of low and high speed. In this thesis, "low speed" is used to indicate that a user's connection speed is equal to or below 40 kbps. Any speed higher than that is called "high speed". This threshold could be adjusted based on further tests and trials.

The recording method can record wave files on either the sender (talker) side or the receiver (listener) side. It was found that recorded wave files had very good voice quality even when the user was at low speed. This feature can be utilised in the prototype in the asynchronous mode.

Based on the foregoing considerations, it is proposed to use the synchronous conversation mode when both users are at high speed, otherwise to use the asynchronous conversation mode (Figure 4.2). The synchronous mode is full-duplex, which is similar to telephone conversation. The asynchronous mode is half-duplex in the sense that it allows only one person to talk at a time and another user has to wait, although Conaito actually uses full-duplex communication between two parties. From

Figure 4.3, we can see that full-duplex UDP communication function is provided by the Conaito SDK, while full-duplex (synchronous mode) and half-duplex (asynchronous mode) communication of this prototype will be implemented on the top of the SDK. Full-duplex/half-duplex communication of the prototype means the method of users' interactions, while full-duplex communication of the SDK refers to the physical signal transmission.

4.2 Synchronous / Asynchronous Conversation

In the initial conceptual model (Chapter 2), synchronous and asynchronous conversation modes were proposed for the peer-to-peer conversation. Implementing peer-to-peer synchronous communication will be very straightforward by directly using the VoIP SDK. In the case of asynchronous communication, the situation becomes complex because mechanisms need to be designed for controlling or synchronising the two communication parties. A server is most often used as a controller by passing messages between the two parties. An asynchronous conversation mode similar to PTT is proposed for this prototype.

In PTT communication (see Section 2.3.1), a button must be held down to transmit the voice. The user who holds the button cannot hear other people. If the button is not pressed, then the user can only receive voice. In the prototype, similarly to this PTT function, when one user is at low speed, a button should be used on the user interface to let two parties toggle between talking (sending) and listening (receiving) states. This button can display either "Start" or "Stop" according the user's conversation state to indicate to the user that s/he can click it to start or stop talking.

When user A wants to talk, s/he clicks the button to start talking. Another user, B's, application would automatically record user A's voice. If user B is at high speed, s/he can instantly hear user A's voice when A is talking. When user A wants to stop talking, s/he clicks the button again to stop talking. User B would also stop recording (the "stop talking" message should be sent by A to notify B via the server). If user B is at low speed, user B's application would automatically play back user A's voice. Figure 4.4 depicts the procedure in a PTT-like asynchronous conversation when user A talks to user B. After user A stops talking and user B finishes listening, user B will take his/her turn to click the button to start speaking.

When a learner is practising a second language, s/he may want to listen repeatedly to what the peer has said so that s/he can understand better. This could be implemented in the asynchronous mode. After the user has received the voice from the peer, s/he would play back the voice repeatedly before clicking the button to speak. Therefore, a playback button is needed on the interface. Delphi provides a special component to access the operating system for recording and playing back of audio files.

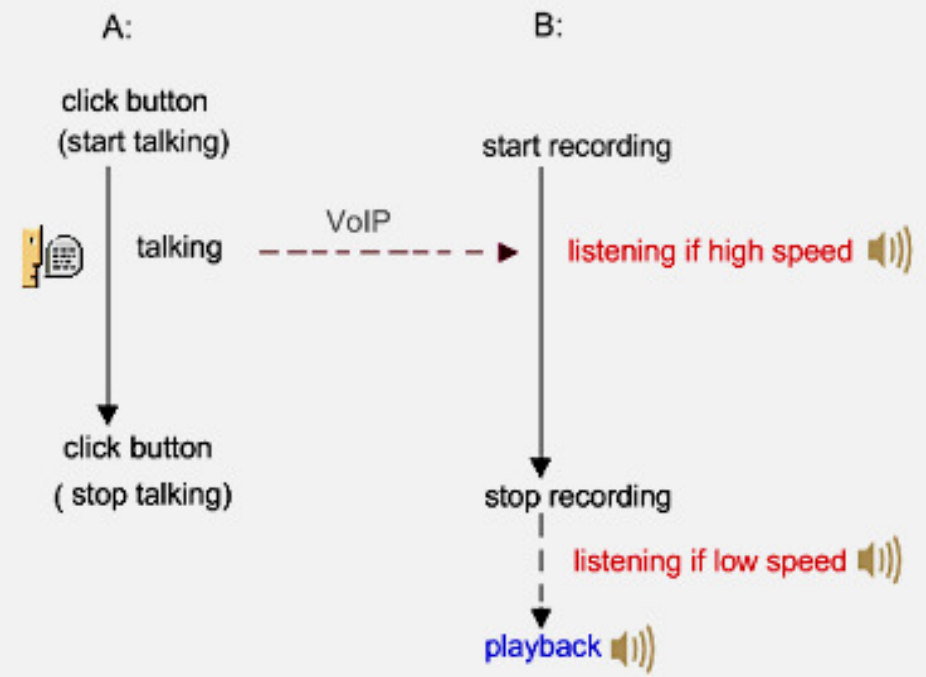


Figure 4.4: PTT-like Asynchronous Conversation

Under both synchronous and asynchronous conversation modes, the prototype should provide the user presence service for all online users. Each student should see all of his/her study group members. The tutor should see all students in the class. After starting the tool, each user should see the status (available/busy/offline) of other users with whom they have conversation rights, so that s/he knows who is available or unavailable. If someone is available, it means that another user can invite him/her. In order to vividly display the actual presence, in addition to showing every user's name and status, a picture will be shown for each user currently online. The user's Internet

connection speed level (low speed or high speed) will also be displayed to indicate his/her bandwidth capability. The user's presence should be maintained dynamically according to the user's status change. The user presence function aims to increase the usability of the prototype because each user is informed of all other users' status.

Under both conversation modes, two track bars are needed on the user interface – one is for adjusting the microphone volume and another for adjusting the speaker volume. Conaito SDK supports these functions. A button is also needed for the user to terminate the conversation.

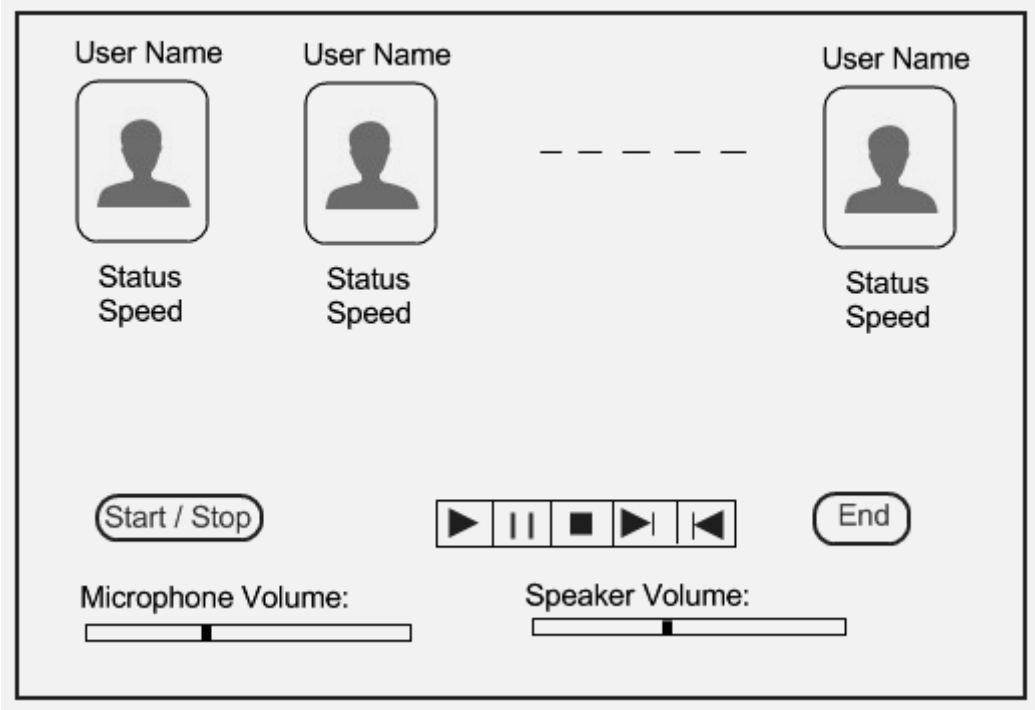


Figure 4.5: Initial User Interface for Asynchronous Conversation

Figure 4.5 is the initial user interface design for asynchronous conversation. It includes a list of users' information (including user name, picture, status and speed level), one start/stop button, one button to end conversation, one playback button set, one microphone volume bar, and one speaker volume bar. The interface for synchronous

conversation is similar but without the start/stop talking button and the playback button set.

4.3 The Specification for the Prototype

An initial conceptual model has been presented in Chapter 2. It is further refined here as a specification for the prototype following on from the foregoing investigations and discussions.

- The goal in the prototype is to design a peer-to-peer conversation component for IMMEDIATE in a second language learning scenario.
- The prototype should be invisibly embedded into the IMMEDIATE architecture without requiring any special installation, configuration or start up by the user.
- The prototype will automatically adapt to synchronous or asynchronous conversation modes according to different users' Internet connection speeds. When both users are at high connection speeds, the synchronous mode will be used; otherwise, the asynchronous mode will be used for the conversation. The asynchronous mode will use PTT-like communication, while the synchronous mode will be telephone-like.
- The prototype will be built on top of the application layer with the help of Conaito VoIP Enterprise SDK, which is based on GSM6.10. The SDK uses a central server to manage UDP-based peer-to-peer communication between clients using call functions such as user registration and call invitation. The threshold of the high and low connection speed for the synchronous and asynchronous modes has been determined by testing the SDK. It will be treated as high speed when the connection speed is above 40 kbps; otherwise, it will be low speed.
- The user interface should be simple and intuitive. The users' presence (online status) should be displayed on the screen and dynamically updated. The interface will also have a button for inviting a peer and another button for terminating a conversation. Other subsidiary functions needed are adjusting the volume of the speaker and the microphone. These functions form the simplest user interface for the conversation component in both modes.

- The asynchronous mode interface will include a dedicated button to switch between the talking and waiting states, and a playback button to allow the user to hear the recorded voice file repeatedly.
- Delphi 7.0 will be used for developing both the client and server applications of the prototype.

4.4 Summary

In this chapter I have analysed the technical requirements of a VoIP SDK for the prototype. Conaito VoIP Enterprise SDK, which is based on GSM 6.10, has been discussed and evaluated. I have shown that the SDK meets most requirement of this project. A threshold of low/high speeds has been decided on by testing the SDK, by which the prototype can automatically fit into either the synchronous or the asynchronous mode. A PTT-like asynchronous conversation mode has also been determined with detailed description of possible interaction and user interface. Lastly, a specification for the prototype has been drawn based on the initial conceptual model.

In the next chapter I will describe how this specification has been implemented in the IMMEDIATE framework.

Chapter 5

Prototyping

The prototyping began from describing a case study and analysing an existing LC (Learning Computer) prototype – IMMEDIATE. The case study (Appendix C7) details the procedure of two learners and one tutor using the tool for practising conversation for second language learning. It includes both synchronous and asynchronous scenarios.

In this chapter, a prototyping approach based on the IMMEDIATE messaging mechanism is presented and discussed, followed by an explanation of the implementation mechanism of call procedures and asynchronous conversation.

5.1 An IMMEDIATE-based Approach

In the previous chapter, a VoIP SDK was investigated and evaluated. Based on the testing of the SDK, the threshold of low and high connection speeds was determined, at which point the system should automatically adopt either the synchronous mode or the asynchronous mode. The SDK includes functionalities such as registration, invitation, termination, deregistration, recording, and adjusting the microphone/speaker. Methods and events of the SDK for these functionalities will be discussed when the implementation mechanism of call procedures is discussed in 5.2.

However, the SDK does not provide all necessary functions in the prototype, for example, displaying the user's presence. The system should maintain each user's presence including the user's online status, which can be changed at any time when the user registers, deregisters, invites another user and disconnects from another user. In order to give information about these status changes, messages need to be passed between members of the same group. Figure 5.1 shows such an example – when user A changes status, a status change message will be passed to the server and the server will

send a message to all other members in the same group (provided that user A, B, …, N are in the same group) to inform them of this change.

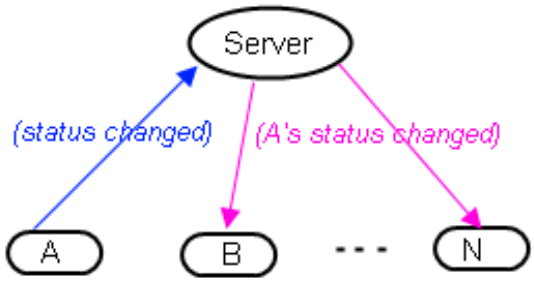


Figure 5.1: Status Change Message among Group Members

On the other hand, to control the asynchronous conversation procedure and to keep a proper talking (sending) and waiting (waiting) order between two parties, messages need to be passed via the server. Figure 5.2 depicts the message transfer when user A finishes talking.

Figure 5.2: Stop Talking Message in Asynchronous Conversation

In this section I will explain how methods and strategies are used to integrate the prototype into the IMMEDIATE architecture. The messaging mechanism will be discussed firstly, followed by the database issues.

5.1.1 Messaging in IMMEDIATE

To cope with asynchronous communication between the client's machine and the server, IMMEDIATE has developed its own messaging methods and protocols. In order to implement all the call procedures and support asynchronous conversation, we also need a method for passing control messages between users' computers. It was logical, therefore, to investigate the messaging mechanism in the existing IMMEDIATE system, to see whether or not we could utilise some of its mechanisms to fulfil our purpose. This will be discussed in the following section.

IMMEDIATE has three separate applications - the Learning Shell, the Repository Manager, and the course management and authoring tool. Each client has a subset of the server's database to store pertinent course information, class information and messages. The content of the client database will be synchronised with the server database when connection is possible. This synchronisation is managed by a message-passing protocol between the server and the learner's machine.

All uploaded messages from users are stored in a Message Queue folder on the central server and processed by the Repository Manager. The Repository Manager places its response to these messages in the user's Message Box on the server. When the server is accessible by the client, the Learning Shell of each user periodically checks its Message Box. If it finds a message in the Message Box, the message will be downloaded and processed by the client's Learning Shell. By this means, users with either broadband or unstable dial-up connections can dynamically update learning resources and communicate with tutors and other students.

Figure 5.3 depicts the procedure of messaging from user A to user B in IMMEDIATE. When user A can connect to the server, A receives a message that has been stored in the database. The message is then uploaded to the Message Queue. The Repository Manager stores the message in the central database. These procedures are shown as (1) to (3) in Figure 5.3. When user B can access the server, it will request any new messages from the repository by sending a request message to the Message Queue. The Repository Manager will query the central database. If a message for B is found, it will be sent to B's Message Box. The message is then downloaded and stored in B's database so that it can be viewed by the user offline. These procedures are shown as (4) to (9) in Figure 5.3.

IMMEDIATE defines several messaging protocols for synchronising messages between the clients and the server. A typical message is used for communication between student and tutor, or between students. A FTP server controls access to the Repository Manager including the Message Queue and Message Boxes. WAR FTP Daemon 1.80 (Aase n.d.) is used as the FTP Server. The details of the IMMEDIATE messaging mechanism and protocols, database structure and server folders can be found in Johnson (2005).

An extension of this approach that is used in the peer-to-peer conversation prototype will be presented in the next section.

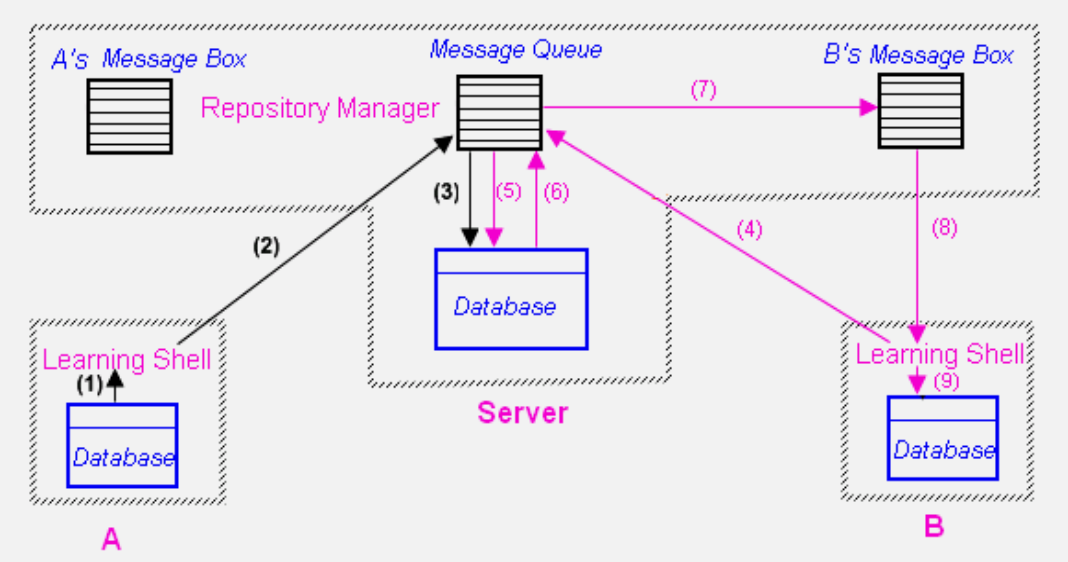


Figure 5.3: Messaging in IMMEDIATE

5.1.2 Messaging in the Prototype

Before the messaging mechanism and protocols used in the prototype are explained, the server folder structure and the database structure used in the prototype will be outlined.

5.1.2.1 Folders on the Server

In the prototype, the server application CallManager controls access permissions, call procedures and asynchronous conversation. It implements the VoIP server from the VoIP SDK and develops its own messaging protocols based on those in IMMEDIATE.

On the server side, similarly to the MessageQueue in IMMEDIATE, a CallQueue folder is used to store all uploaded call messages from clients. Similarly to the MessageBox, a CallBox folder and a RegisterBox folder are used to store call messages for each user. Explanation will be given later about why the RegisterBox is needed. Similarly to the Repository Mananger, CallManager reads and processes the messages in the CallQueue, and generates new messages to put into users' CallBoxes or RegisterBoxes. Figure 5.4 shows messaging in the prototype. In the figure, the sequence of arrows can be interpreted as the CallManager informing user B of user A's status change in call

procedures, or as user A sending a "Stop Talking" message to its peer B via the CallManager in an asynchronous conversation.

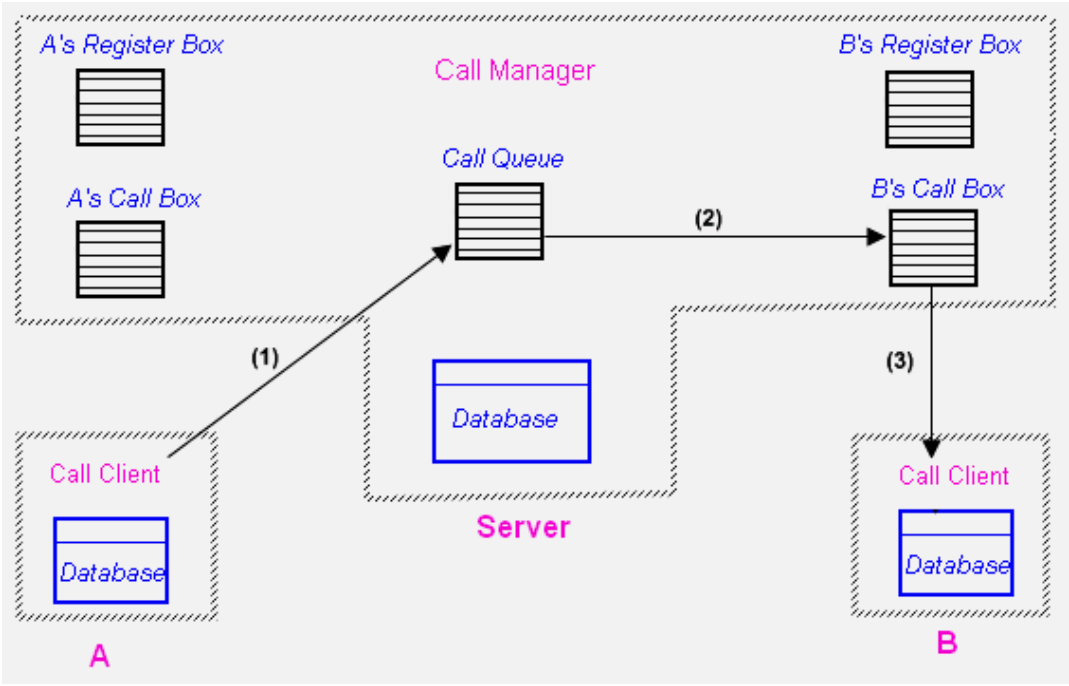


Figure 5.4: Control Message in the Prototype

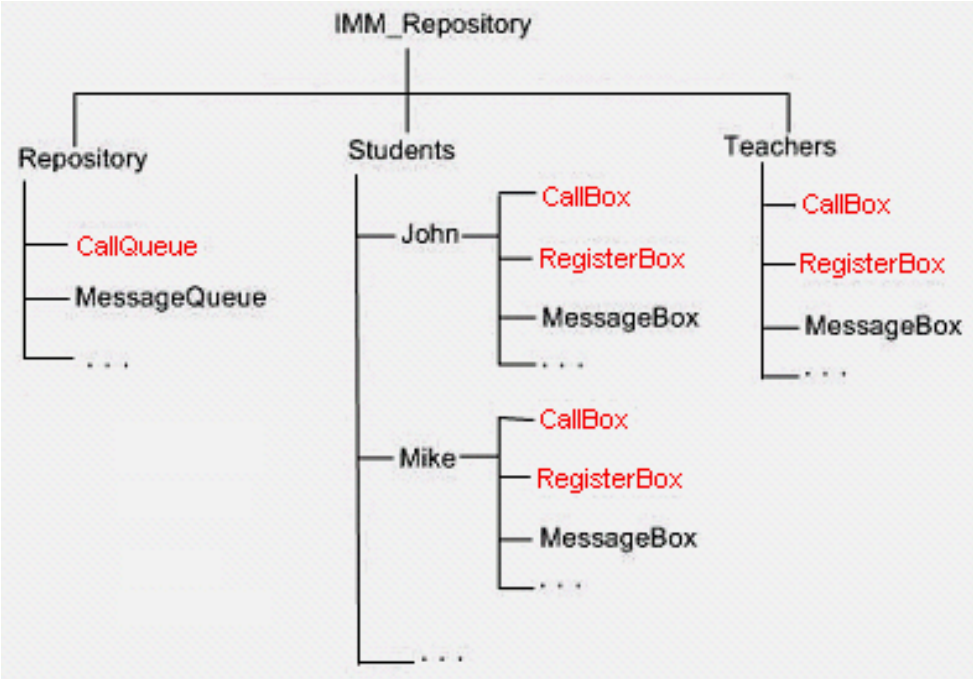


Figure 5.5: Folder Structure on the Server for the Prototype

CallManager uses a similar folder structure to the Repository Manager in IMMEDIATE. Figure 5.5 shows new folders (CallQueue, CallBox, and RegisterBox) added to the existing folder structure on the server. For each student, there is a CallBox and RegisterBox on the server.

5.1.2.2 Database

The client has the same database structure as the server so that learning content and messages can be downloaded and stored on the learner's machine to support offline learning. The prototype has added a table *VoIPUser* (Figure 5.6) to store all online users' information. Paradox[3] is used as the database management tool. Figure 5.6 shows the structure of table *VoIPUser*, which includes the user name, group number, connection speed, IP address, port number, status (Available, Offline, Busy) and peer name. Each student belongs to a study group in each course. Similarly to IMMEDIATE, the student's group number is an integer larger than 0. A tutor does not belong to any study group so his/her group number is defined as -1.

Name	(user's name)
GroupNo	(user's group number)
Speed	(user's Internet connection speed)
IP	(user's IP address)
Port	(user's port number)
Status	(user's current status – available or busy)
Peer	(user's current peer)

Figure 5.6: Table *VoIPUser*

CallManager will query and update this table during call procedures. For example, after a user registers, CallManager will add a new record to this table (status is "available"). After an invited party accepts the inviter's call invitation, the invitee's name will be

[3] Paradox is a relational database, which is integrated in the Borland Delphi IDE Because of the nature of the project and IMMEDIATE (only local database is used), we adopted it for its simplicity and ease of use.

added to the inviter's record and both users' status will be changed to "busy". After the user deregisters, a corresponding record will be deleted from the table.

5.1.2.3 Call Message Types

Several message types are defined (see Figure 5.7) in the system dictionary (immSystemDictionary.pas):

```
vcRegisterResponse = 11;      // server registration response message to the user
vcRegisterInform =12;          // server registration information to other users
vcVoiceInform = 13;            // user status change message
vcsDisconnectInform = 14;     //inform the server that a user has disconnect with its peer
vcsStopTalking = 21;          //inform the server that a user has stop talking
vcStopTalkInform = 22;        //peer stop talking message
```

Figure 5.7: Message Types in the Prototype

These messages can be divided into two categories. One kind of message, the name of which is prefixed with vcs, is generated and sent from the client to the CallManager. The message is uploaded to CallQueue and will be processed by the CallManager. Another kind of message, the name of which begins with vc, is generated by the CallManager, put into the user's CallBox, and downloaded by the user.

Message type vcRegisterResponse, vcRegisterInform, vcVoiceInform and vcsDisconnectInform are used in call procedures. Figure 5.12 shows their roles in call procedures. Message type vcsStopTalking and vcStopTalkInform are used in asynchronous conversation.

The message format in the prototype is similar to an IMMEDIATE message. Each message is a text file, which includes a head and a body. The head (Figure 5.8) includes names of the sender and receiver, sender's group number and the message type. The reserved zero is used for compatibility with the IMMEDIATE message.

Receiver	(message receiver)
Sender	(message sender)
Message type	
Group number	(receiver's group number)
0	(reserve)

Figure 5.8: Message Head Format

For example, for a message generated by the CallManager, the sender is "CallManager" and the message will be stored in the receiver's CallBox. For a message generated by the client application (CallClient), the receiver is "CallManager" and the message will be uploaded to the CallQueue. Their roles of these messages will be described in 5.2 and 5.3.

5.1.3 Inheritance from IMMEDIATE

The client application (CallClient) and the server application (CallManager) of the prototype are supposed to be merged (embedded) into the Learning Shell and the Repository Manager of IMMEDIATE separately in the future (Figure 5.9). Similar implementation mechanisms of the message protocols of both of them are the foundation stone that ensures the embeddability of the former in the latter. The way in which CallClient and CallManager are implemented in the prototype will be shown in this section.

5.1.3.1 CallClient

IMMEDIATE uses a component-based approach for developing new functions for Learning Shell. It uses two templates for generating new forms and components in Delphi 7 - the form template TPLCForm, and the frame template TPLCFrame. A Delphi frame enables an entire form to be encapsulated into a reusable component that can be dropped onto another form in the Delphi IDE. The main form of CallClient, TVoIPClient, was inherited from TPLCForm. TVoIPClient includes TPLCVoiceFrame, which was inherited from TPLCFrame (Figure 5.10). A TPLCVoiceFrame object implements the user interface.

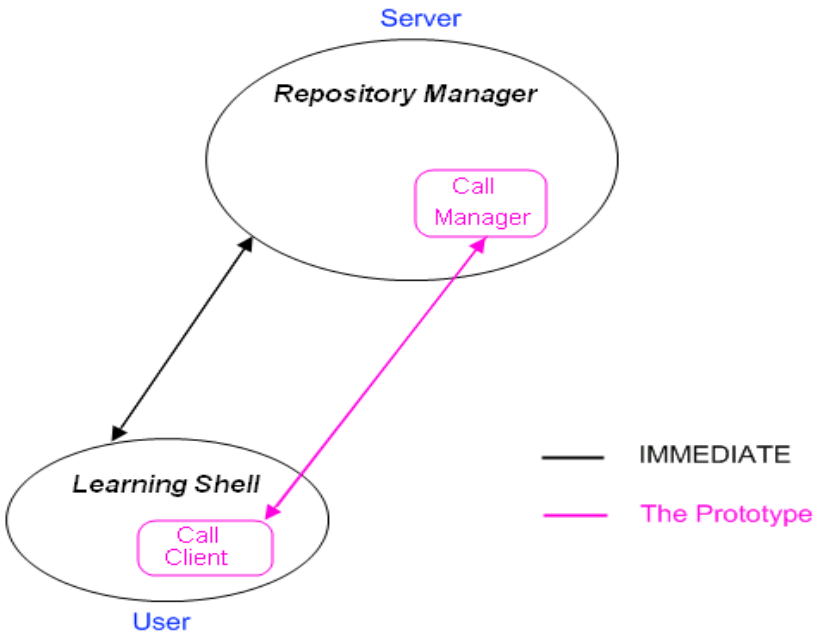


Figure 5.9: Relationship between the Prototype and IMMEDIATE

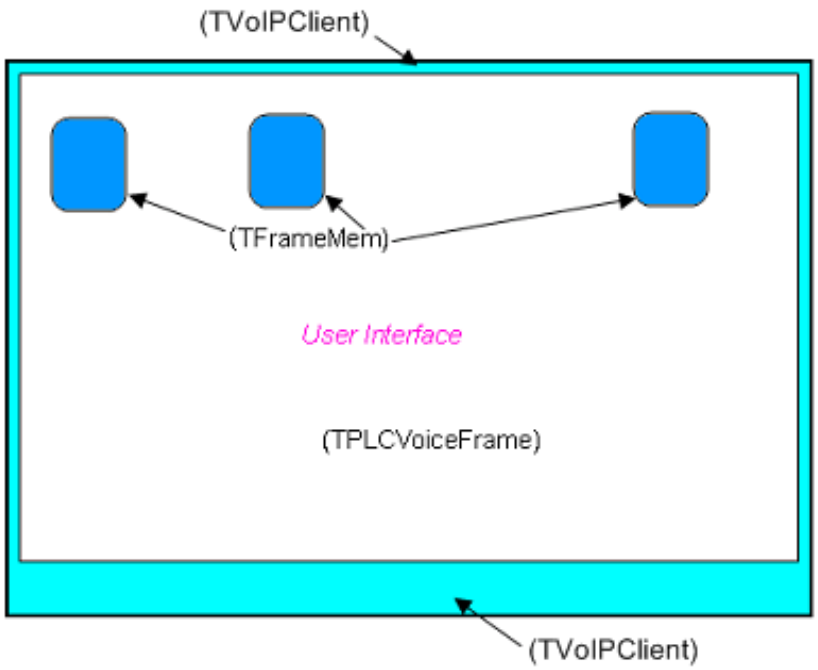


Figure 5.10: CallClient Classes

A list of same group users will be displayed on the user's screen after registration. Each user's information (including user's name, picture, connection speed and status) is maintained by a TFrameMem object, which is dynamically generated from the TPLCVoiceFrame object when the tool runs (see Figure 5.9 and Appendix A). The interface includes a "Start/Stop" button, a playback button group, two volume control bars, a "Disconnect" button, a timer, an FTP client and other components. Appendix A1 shows the detail of the definition of TPLCVoiceFrame. Figures 6.1 to 6.7 show the client application interface.

In order to be embedded into IMMEDIATE in the future, CallCient is designed similarly to the Learning Shell. The VoIP SDK client component TConaitoConfClientVoIP is defined as an object in the TPLCVoiceFrame. Similarly to the Learning Shell, it uses a timer to check the user's CallBox or RegisterBox periodically to see if there are any new messages. Appendix A2 shows the timer event. ImmSystemDictionary.pas is used to store constants defined both in IMMEDIATE and in the prototype. An FTP client component is also defined in CallClient for uploading and downloading messages.

TController class is defined in plcControllers.pas. IMMEDIATE uses a controller object, which is generated by TController, for the Learning Shell interface components to interact with each other and to access system data. User and system information such as user name, password, group number, connection speed, server IP address and server port number can be obtained by using TController methods. However, for rapid implementation and testing, in the prototype, such information is obtained directly from a configuration file – voip.ini. Since there was only a low dial-up connection available when testing, to simulate the low-to-low asynchronous conversation mode, a low speed value was set in this file although the actual connection was high speed.

5.1.3.2 CallManager

The server application, CallManager, manages and controls conversations in two ways - by processing messages from the CallQueue, and by events of the SDK's server component (TConaitoConfServerVoIP). The CallManager's architecture has been designed to support future integrity in the Repository Manager.

The main class of CallManager is TVoIPServer (Appendix B2). It defines new procedures and implements events of TConaitoConfServerVoIP to manage communication between the server and clients. The interface of the CallManager is fairly simple. Figure 5.11 shows the interface when two users are online.

In the following two sections I will show the mechanism of CallManager when explaining call procedures and asynchronous conversation.

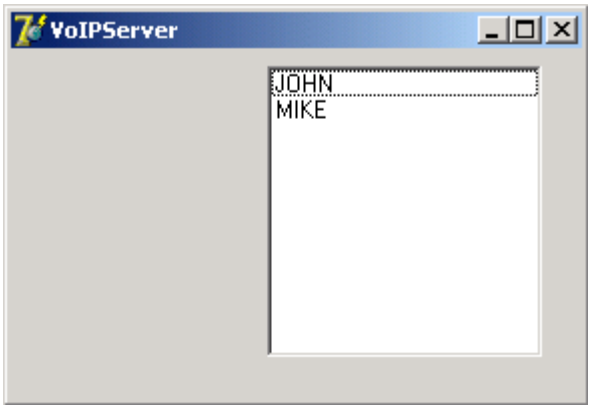


Figure 5.11: Interface of CallManager

5.2 Call Procedures

In this section the implementation methods of call procedures - including user registration, call setup, and call teardown - will be presented. During the explanation of these call procedures, relevant events of the SDK and message protocols that are used will be explained.

An important issue that will be discussed here is the mechanism of implementing the user presence. Although the prototype is for peer-to-peer conversation between two users, all online group users should be informed if any user's status has been changed during these call procedures. If a tutor's status has been changed, all students in his/her class should be informed; if a student's status has been changed, all other group members and the tutor should be informed.

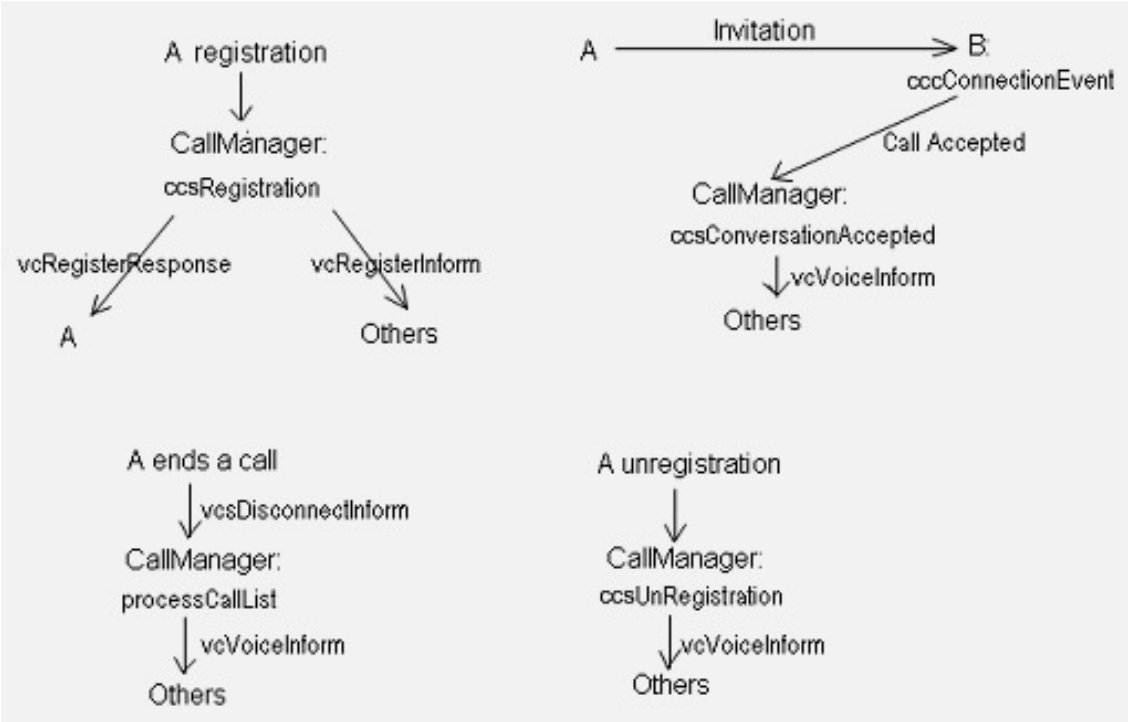


Figure 5.12: Main Events & Message Types in Call Procedures

There are three statuses for each user - Available, Busy and Offline. A user is available when s/he has started the tool successfully and is not in conversation with another user. A user is busy when s/he has connected to another user (peer) in a conversation. A user is offline when s/he has not started the tool successfully. The status of each user will dynamically change when the user switches between registration, deregistration, call setup and tear down procedures.

CallClient displays and maintains all online users' presence changes. When a user successfully registers, his/her screen will show a list of online members with each member's name, the level of connection speed (low or high), online status (available, busy, offline), and a picture. When there is a user whose status has changed, the CallClient will inform the CallManager. The CallManager will query the database to find who should be informed of this change. Then CallManager sends messages to those users.

Figure 5.12 depicts the main events and message types in four call procedures for a user. These procedures will be discussed in the following sections.

5.2.1 User Registration

When a user starts the tool, s/he will be registered on the server automatically - the ccsRegisteration (*sic*) event (Appendix B3) of the server SDK will be triggered on the CallManager. In this event, a new record of the user's information will be added into the *VoIPUser* table. Then two kinds of messages will be generated - the response message (vcRegisterResponse) to the newly registered user; and the user's status change message (vcRegisterInform) to other online users. Figure 5.12 depicts the user registration procedure.

The ccsRegisteration event queries the database and obtains the speed and status of each group member from the database. Then a vcRegisterResponse message will be generated and be put into the user's RegisterBox. The head of the vcRegisterResponse message is as follows:

Username (message receiver – registered user)
CALLMANAGER (message sender)
vcRegisterResponse (message type)
groupno (message receiver's group number)
0

The body of the message includes the following information about all group members. Each member's information ends with an '<END>' mark.

Member name
Member group number
Member Speed
Member Status
<END>

After receiving this message, the user can note on the interface the presence of all group members. Figure 6.1 shows the user interface after two users have logged into the tool.

For all other online users (both available and busy users), a vcRegisterInform message will be sent to their CallBoxes to inform them of the newly registered user. The head of this message is similar to the head of vcRegisterResponse message, but the body of the message includes information about the newly registered user only:

User name (the newly registered user's name)
User group number
User speed
1 (the user is available)
<END>

The reason for using the RegisterBox instead of the CallBox to store the vcRegisterResponse message will be explained in the following section. If the CallBox is used to store this message, when users A and B start the tool at almost the same time, before A's vcRegisterResponse message arrives at A's CallBox, B might have already sent a vcRegisterInform message into A's CallBox via the CallManager to inform A of B's registration. Because A will read the CallBox in the first-in-first-out order, the vcRegisterInform message might be processed before the vcRegisterResponse message. This is a conflict because A cannot accept information from any other user before it finishes its own registration procedure. Using a RegisterBox can avoid this conflict. A timer (Appendix A.2) of the CallClient will read the RegisterBox first if the registration has not finished. If the registration has finished, it can read the Call Box and process any other messages.

5.2.2 Call Invitation

The call invitation is the most complex call procedure. User A can invite user B to join in a conversation by clicking on the user B's picture (Figure 6.2) if user B is available. Then user A's ActConnectExecute procedure (Appendix A5) will be called. In this procedure, the Connect method will be called to connect (or invite) user B.

On user B's application, the cccConnectionEvent event (Appendix A6) will be triggered. In this event, user B will be asked if s/he wants to accept the invitation (Figure 6.3). If user B accepts the invitation, user A's speed will be obtained for user B. If the speed of either user B or user A is low, an asynchronous conversation will be set by user B's application. Otherwise, a synchronous conversation will be set. Then, the AcceptIncomingCall method will be called. This will trigger the ccsConversationAccepted event on the server (Appendix B5), where the SetVoiceStreamIPPort method will be called to build a UDP-based direct peer-to-peer communication between users A and B. In the documentation (Conaito 2005) of the SDK, it is mentioned that when a direct UDP connection cannot be built between two users, SetVoiceStreamThroughProxy will be called to connect two users via the server.

However, this situation did not arise during our implementation and testing. Then vcVoiceInform messages will be sent to other online users to inform them that users A and B are busy (i.e. the status of user A and B has been changed from "Available" to "Busy"). This means that two users have successfully connected.

After user B accepts A's invitation, user A's ActConnectExecute procedure will check to determine whether or not user A or B's speed is low. If it is, then the asynchronous mode will be set for user A. Otherwise the synchronous mode will be set. The procedure of asynchronous conversation will be discussed in 5.4. Figure 5.13 shows the system's procedures when user A invites user B and user B accepts user A's invitation.

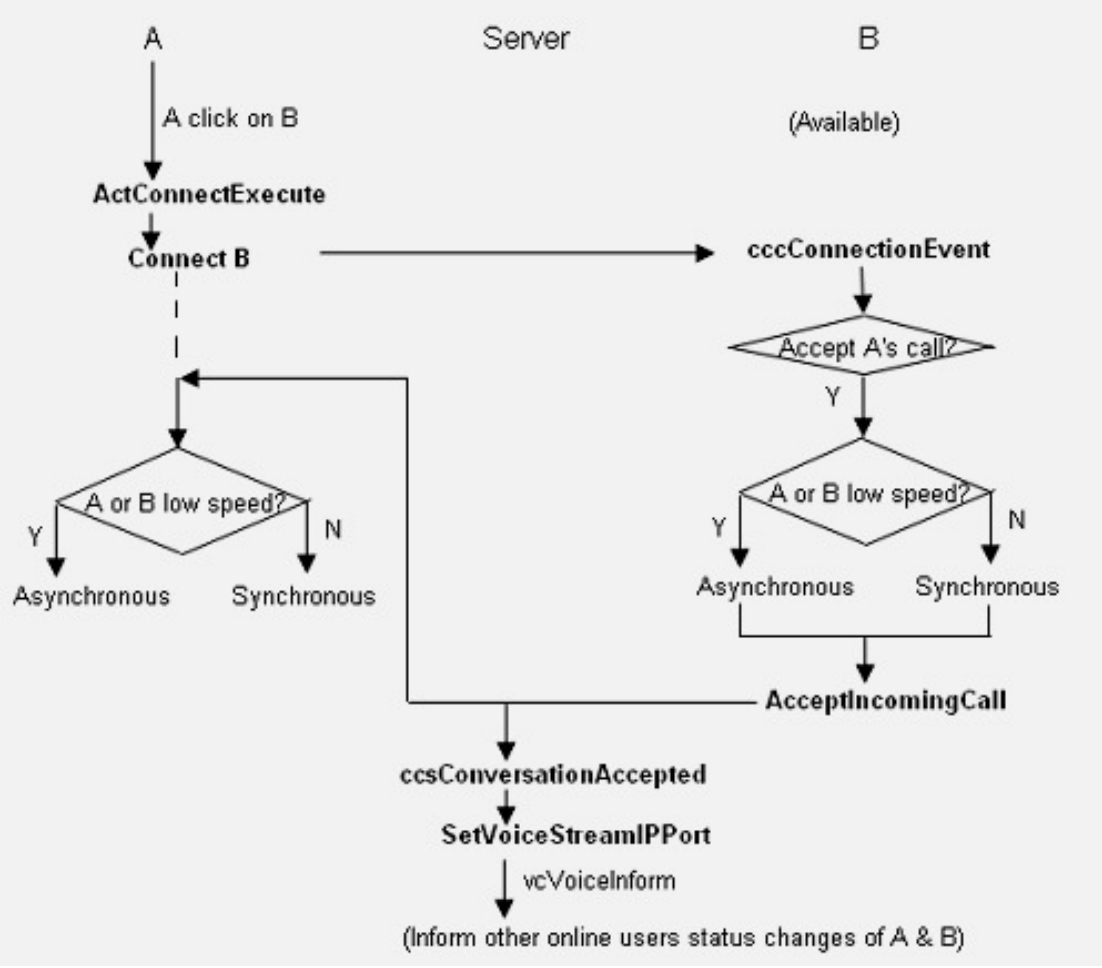


Figure 5.13: System Procedure for Call Invitation

5.2.3 Call Teardown

After a call invitation has been successfully accepted, both users can talk directly via their headsets in either synchronous or asynchronous mode. At any time during the conversation, one user can click the "Disconnect" button to end the conversation. The "Disconnect" method of the SDK will be called in the CallClient. No event of the SDK will be triggered on the CallManager when a user disconnects. A message called vcsDisconnectInform will be generated to inform the CallManager of the disconnection. When the CallManager reads this message, it will change the status of both users from "Busy" to "Available" in the database, and inform all other online users by sending vcVoiceInform messages.

Appendix B4 is the processCallList Procedure of the TVoIPServer class. It shows how CallManager processes the vcsDisconnectInform message, as well as the vcsStopTalking message (used in the asynchronous conversation mode) that will be discussed later.

5.2.4 User De-registration

A user can exit the tool either by clicking the "Exit" button or by closing the window. The UnRegisterToProxy method will be called. This will trigger the OnUnRegisteration event in the CallManager. In this event, the user's record is deleted from the *VoIPUser* table in the database. A vcVoiceInform message will be generated to inform all other online users about this user's status change (from "Available" to "Offline").

If a user is still connected with another in a conversation, then s/he cannot quit the tool unless s/he ends the conversation by clicking the Disconnect button. Sometimes when a user is still in a conversation, the connection between the user and the server will be lost for some reason (see Section 6.4.1).

The ccsConnectionLost event will be triggered in the CallManager after either manual or automatic disconnection. In this event, firstly, the status of the user's peer will be changed from "Busy" to "Available". This status change information will be sent to all other online users. Then the record of the user who lost the connection will be deleted from the database. All other online users will also be informed of this user's status change (from "Busy" to "Offline").

In this section, several message types have been presented during the description of call procedures. Two message types, vcRegisterResponse and vcRegisterInform, are used to inform other users about the registration of a user. The vcVoiceInform message is used to inform status changes including “Available” to “Busy” (invitation), “Busy” to “Available” (disconnection), “Available” to “Offline” (exit) and “Busy” to “Offline” (a exceptional case).

5.3 Asynchronous Conversation

In this section the way in which asynchronous conversation is implemented in the prototype will be explained. Firstly, the procedure of asynchronous conversation will be described from the system perspective. Then, the focus will be put on control messages for asynchronous conversation to give some understanding of how the talking-waiting order is maintained. Lastly, a different implementation approach for asynchronous conversation will be discussed.

5.3.1 An Overview of Asynchronous Conversation

Figure 5.14 shows an example flow chart when John invites Mike to enter into a conversation. It depicts both synchronous and asynchronous conversation modes. In an asynchronous conversation, the inviter always has the right to speak at first (John is the inviter in this example, so he will speak first). Two users will then take turns to speak and wait until one of them terminates the conversation.

In Section 5.2.2 (Call Invitation), it has been explained how the system works when a user invites another one to join in a conversation. After the ccsConversationAccepted event (see Figure 5.13) in the CallManager, two users are connected with each other and are ready to start a conversation. A client event, cccStatusEvent (see Appendix A3), is then triggered with a status code 53 on both the inviter and the invitee. In the corresponding code in this event, the conversation will be initialised for both synchronous and asynchronous modes.

In Chapter 4, it is mentioned that when one user is at low speed, by using the SDK, the user cannot hear a clear voice from another because the arriving voice is constantly broken and distorted. On the other hand, very clear voice files can be recorded on either

the sender or the receiver side. In this prototype, the voice is recorded on the receiver because it is more efficient than recording on the sender (see Section 5.3.3).

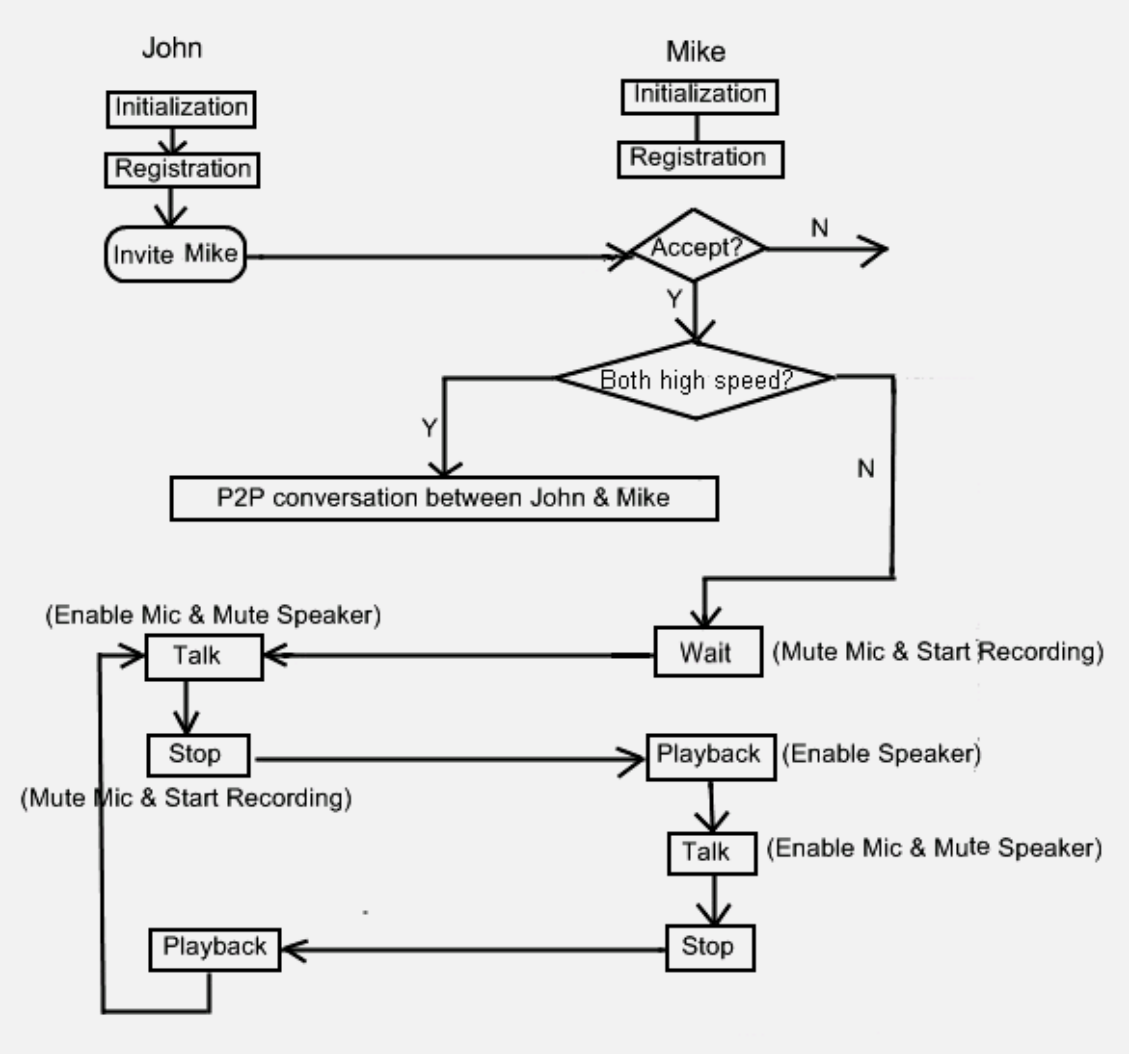


Figure 5.14: User Interaction Flow Chart for Bimodal Converstion

In an asynchronous conversation, if the receiver is at high speed, s/he can hear the peer's voice instantly when the peer is talking. However, if the receiver is at low speed, s/he cannot hear anything until the peer clicks the "Stop (F5)" button to finish talking. When the peer finishes talking, a control message will be sent to the receiver via the server. As soon as the control message reaches the receiver side, it will stop recording

and automatically play the recorded file so that the receiver can hear what the peer said. The receiver can also play the file back any number of times until s/he decides to speak and so clicks the "Start (F5)" button.

5.3.2 Control Messages in Asynchronous Conversation

In the PTT-like asynchronous conversation, at any time only one user can talk and another user has to wait. The tool needs to keep a proper conversation order for communication between the two parties.

There is a button that shows "Start" or "Stop" (Figure 6.5 or Figure 6.7) depending on the current status of the user (whether the user can speak or not). When the button displays "Start (F5)", the user can click it (or press F5) to start talking; when it is "Stop (F5)", the user can click it (or press F5) to stop talking. For both cases, a procedure btTalkClick (Appendix A4) will be called from the CallClient. When the user clicks the "Stop" button, a vcsStopTalking message will be generated and sent to the CallQueue in this procedure.

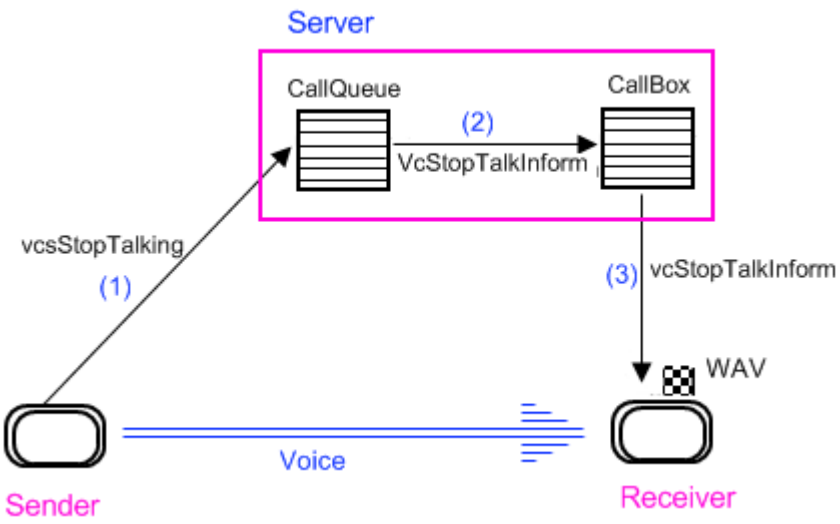


Figure 5.15: Control Messages for Asynchronous Mode

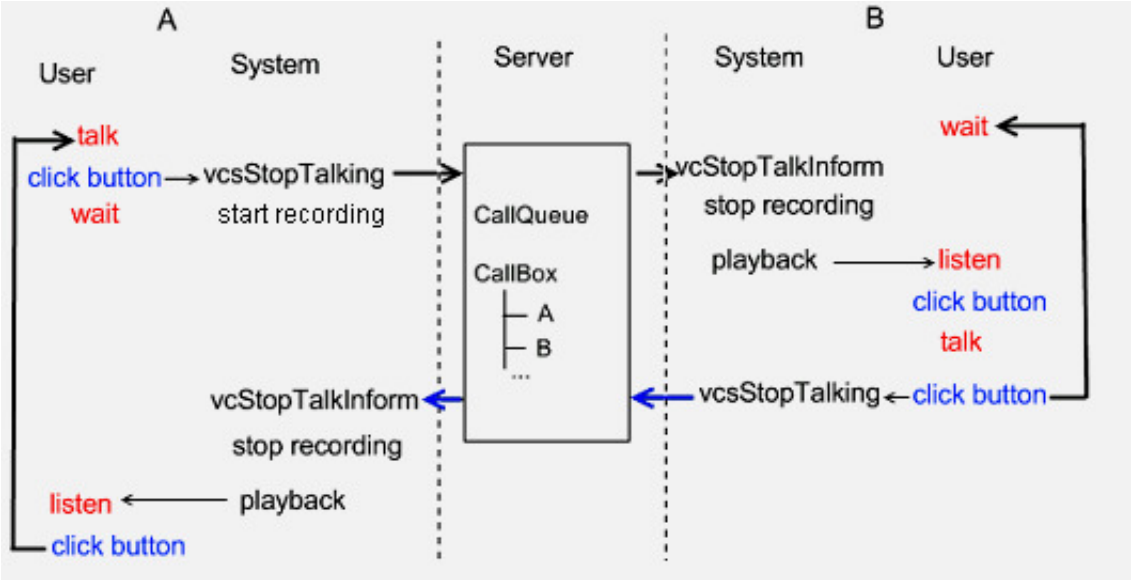


Figure 5.16: System Messaging when Both Users at Low Speeds

The CallManager will process the vcsStopTalking message and generate a vcStopTalkInform message to the receiver's CallBox. On the receiver side, after the message is read from the CallBox, if the receiver is at low connection speed, the recorded wave file will be automatically played back, and the button will be changed to "Start" so that the user can click to speak. Figure 5.15 shows the procedure of message transference when the wave files are recorded on the receiver side. Figure 5.16 shows the relationship of interactions between the users and the system when both users are at low speeds.

5.3.3 Another Approach

The asynchronous mode can be implemented in two ways. The way described above is recording on the receiver side. Another way is to record the voice file on the talker's side and send it to the receiver via the server. Both ways are possible; the difference is in their relative efficiency. The following discussion concerns the latter way.

The TMediaPlayer component of Delphi 7 has been trialled to record wave files using any codec installed in Microsoft Windows such as PCM or GSM 6.10. Because low bit rate codecs such as G.723.1 and Speex are not available in Microsoft Windows by default, GSM 6.10 seems to be the best choice. The TMediaPlayer component needs an

initial wave file before recording. Microsoft's Sound Recorder can produce an empty wave file that uses a codec such as GSM6.10. TMediaPlayer also can play back wave files when corresponding codecs are pre-installed on the computer.

To record wave files using GSM6.10 on the sender and send it to the receiver via the server, a WaveFileQueue should be added in CallManager for storing wave files from the sender. Two message types, vcsWavFileArrived and vcWavFileInform, should be used for sending the wave files.

Figure 5.17 shows the system procedure for the transfer of a wave file from the sender to the receiver. When the sender stops talking, it will upload the recorded wave file to WaveFileQueue and a WAVE_FILE_ARRIVED message (message type vcsWavFileArrived) will be sent to CallQueue on the server. The body of the message includes only the name of the wave file.

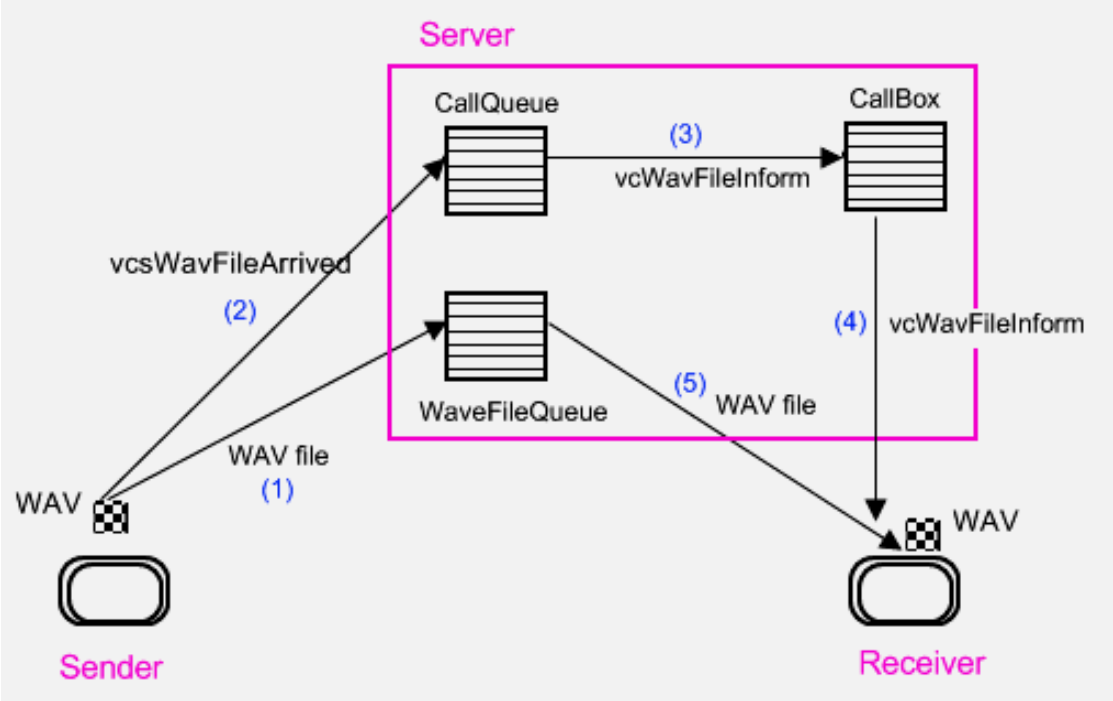


Figure 5.17: Control Messages when Recording on the Sender

When CallManager checks the WAVE_FILE_ARRIVED message from the CallQueue, it will generate a WAVE_FILE_INFORM message (message type vcWavFileInform) to the receiver's CallBox. The body of the message includes the name of the sender and the name of the wave file. The receiver will download this message from the CallBox, read the name of the wave file from the message, and then download the wave file from the WaveFileQueue.

Nevertheless, under testing, this way proved to be much slower than the way of directly recording on the receiving side, which has been described in Section 5.3.2. The procedure in Figure 5.15 is obviously simpler than the procedure in Figure 5.17.

5.4 Implementation

The prototype was implemented using the methods described in previous sections in this chapter. Three PCs were used for the implementation – two client PCs (for two conversation parties) and one server PC. Three computers all had ADSL broadband connections to the university's network. One 33.6 kbps modem was used on one client's computer so that it had low speed dial-up connection to the university's network. The prototype was tested repeatedly with several possible solutions on the three computers during the implementation, especially for the asynchronous conversation mode (see 6.4.1 for further description regarding the prototyping environment). One trial for asynchronous conversation was discussed in Section 5.3.3. During the implementation, problems were identified and proper solutions were found.

In implementation methods and events of the VoIP SDK were used in the prototype. Because the SDK does not provide all the necessary functions for call procedures, the prototype defines its own messaging protocol to implement some functions. For example, no event of the SDK will be triggered on the server when a client calls the Disconnect method to end a conversation. Therefore, a vcsDisconnectInform message is defined to inform the server of the disconnection.

CallClient uses a useful event cccStatusEvent in the client SDK, through which the CallClient can capture some important system incidents with corresponding status codes. For example, when a user loses the connection with the server, the event will be triggered with a status code 58. The event with code 53 is triggered when an invitation has been accepted and a connection between two users has been built. Appendix A3 shows how system situations are dealt with in the cccStatusEvent event.

When implementing the tool, one old sound card in a PC was found to generate voice with a sharp noise (probably because of accumulated dust). However, when it was replaced with a new sound card, the receiver could hear a very clear voice from the sender.

5.5 Summary

In this chapter I have described the implementation methods of the peer-to-peer conversation prototype including both synchronous and asynchronous conversation modes. A VoIP SDK and the messaging mechanism of IMMEDIATE were suitably and successfully utilised and integrated into the prototype for implementing the user presence, call procedures and asynchronous conversation. Message protocols were presented in the prototype. Although the CallClient and the CallManager are developed as stand-alone applications for testing, in order to be integrated into IMMEDIATE in the future, they use a similar structure to the Learning Shell and Repository Manager of IMMEDIATE. The prototyping demonstrates that the IMMEDIATE component-based approach makes it easy to develop a new component. It also proves that the messaging mechanism of IMMEDIATE can be easily extended to new protocols to provide new functionalities.

Chapter 6

Evaluation

In this chapter I will discuss the following aspects of the evaluation of the tool:

- Goals of the Evaluation
- Evaluation Method
- Laboratory Evaluation
- Evaluation Conclusion

6.1 Goals of the Evaluation

The aim in this project was to develop a reusable online conversation tool that can be used at any_time from anywhere by anybody as part of a Learning Computer to apply in a second language extramural course. A bimodal approach using both synchronous and asynchronous conversation modes was proposed in Chapter 2 and refined in Chapter 4, and a prototype was developed in Chapter 5. The prototype needed to be evaluated to demonstrate the feasibility of the bimodal approach.

Sharp et al. (2007, p. 586) point out that the purpose of evaluation is "to check that users can use the product and that they like it, particularly if the design concept is new". In this master's project, the tool was designed as a prototype instead of a commercial product. The design emphasises on achievement of the goal of peer-to-peer conversations under different Internet connection speeds and simplifying the basic interaction rather than refining the interface or integrating other communication functionalities, such as online chat, into the tool. However, it is still important to discover what users who are learning a second language think about the tool. Overall, the evaluation was conducted to prove that the two integrated communication modes can facilitate conversation practice with any connection speeds at any locations.

The goal of the evaluation can be refined further with the following issues:

Usability

- Whether or not the quality of the voice is good enough
- Whether or not the conversation time delay is acceptable in narrowband connections
- Whether or not the user interface is simple enough for use

Functionality

- Whether or not the peer-to-peer conversation including synchronous and asynchronous conversation can be smoothly conducted
- Whether or not the user can control the volume of voice easily

Integrity

- How does the user find experiencing the two conversation modes in the tool?
- How does the user find using the PTT-like asynchronous way for conversation?
- What are the users' preferences between the two conversation modes?

Accessibility

- Whether or not the tool works well for different Internet connections
- Whether or not the tool works well when the user's computers are located outside the university network

Other issues may be included in future evaluations (Chapter 7) when there are more resources available such as time, people, computers, and different connection speeds.

6.2 Evaluation Method

A main part of this project was to design a way that lets conversations become possible for users in rural areas, where usually only very low dial-up connection speed is available. A full evaluation for a prototype can include two stages - laboratory evaluation and field evaluation. In the laboratory evaluation its usability, integrity and functionality were assessed. In the field evaluation its accessibility and usability were planned to be evaluated.

It is very convenient to use the laboratory environment, where the tool was developed, to evaluate its integrity, functionality and usability. However, in the laboratory environment, both the server and client computers are located in the same network, and the dial-up speeds (around 30 kbps) might not be as low as in some rural areas. A field

evaluation can enable an assessment of the accessibility of the tool, and further evaluation of its usability by real learners using the conversation tool from their own homes in rural areas. To enable learners' computers outside the university access the VoIP server in the university, a static public IP address should be assigned to the server. Unfortunately, the university did not permit this at the time of the evaluation. The field evaluation will be discussed as a future work in Chapter 7.

Questionnaires, interviews and log files were be used for obtaining results from testers and data from the system. The results and data were be analysed to enable me to draw conclusions from the evaluation. Questionnaires were designed for collecting users' opinions about the tool. Interview questions were designed just as a guideline. "Interviews typically include many open-ended questions where users are encouraged to explain themselves in depth." (Nielsen 1993, p. 221). Questionnaires were completed before the interview, so that during the interview the experimenter could clarify or verify issues or ideas in the questionnaires.

In the laboratory experiment, the experimenter observed users' behaviour while they were interacting with the tool, and monitored the system's behaviour such as incidents that might occur during the experiment. The experimenter can use observation result to ask questions in the interview. Questionnaires, interviews, observation and log files were analysed to conclude an evaluation result.

6.3 Laboratory Evaluation

In this section I will report on the laboratory evaluation plan including participants, test environment, questionnaire and interview. A pilot evaluation was conducted before the formal evaluation. Both the pilot and the laboratory evaluation were analysed and conclusions concerning their results were reached.

6.3.1 Participants

Two testers are needed in a laboratory experiment. The author played the role of the experimenter in each experiment – supervising the whole experiment including observing, conducting the questionnaire and interview, and solving problems.

There were 3 volunteer participants in the pilot and laboratory experiment. Table 6.1 lists their computer usage from the participant's profile questionnaire (Appendix C3):

Tester	Computer Usage
User 1	Over 6 hrs/day; using Microsoft Word, Excel, PowerPoint; scientific software; Internet; Email
User 2	Around 3 hrs/day; using Microsoft Word; Email; Internet
User 3	Around 1 hr/day; using Microsoft Word; Email; Internet

Table 6.1: Laboratory Experiment Participants' Computer Usage

User 1 works on the computer most of the time. He uses the computer for writing, calculation, experiments and teaching. He also browses the Internet for reviews, email and news; and uses other software. The computer is also a daily tool for User 2 for editing documentation and exchanging emails with others. His daily average working time on the computer is around 3 hours. User 3 does not use the computer very much. He uses it only from time to time for editing documentation, sending or receiving email and browsing websites.

User 1 participated in the pilot experiment with the author (in the pilot evaluation, the author acted as User1's peer, as well as the experimenter); User 2 and User 3 conducted the laboratory experiment. All 3 testers come from foreign countries with English as their second language.

6.3.2 Test Environment

Three PCs were used in the laboratory experiment. On the server PC, Windows XP, WAR FTP Daemon 1.80 (Appendix B1) and the server application of the tool (CallManager) were installed. It used ADSL to connect to the university network. Windows XP and the client application of the tool (CallClient) were installed on two client PCs, which were located separately in two rooms so that two testers could not hear each other. Each client PC could access the university network either via either ADSL or dial-up, in which case two standard 33.6 Kbps modems were used (so the dial-up connection speed was below 33 kbps).

6.3.3 Questionnaires and Interview

Several kinds of questionnaires were used in the laboratory experiment: Participant Profile (Appendix C3), Short Questionnaire (Appendix C4) and Final Questionnaire (Appendix C5).

The participant profile questionnaire was used to obtain testers basic information such as gender, job and their computer usage, since these elements might affect evaluation results. There were four tests in each laboratory experiment. The voice qualities and problems might be different for each test. A short questionnaire was needed for each of four tests. After all the four tests, a tester was asked to answer a final questionnaire. The final questionnaire included integrity, functionality and usability questions.

An interview was conducted for each tester separately after the final questionnaire. Based on the interview question outline (Appendix C6), the experimenter let the tester explain his/her behaviour observed in tests, further clarify some answers in questionnaires, express his/her thoughts about the tool, and offer suggestions for improving the tool.

6.3.4 Tasks in the Laboratory Experiment

In each experiment, there were two testers and one experimenter. Each tester performed the following tasks (Further details are in Appendix C1):

- Read the Information Sheet (Appendix C2)
- Filled in the Participant Profile Form (Appendix C3)
- Conducted four tests with the peer. In each test, the experimenter connected testers' PCs to the Internet via either broadband or dial-up connection according to Table 6.2. The tester clicked the client application (CallClient) and started a conversation with the peer, then answered the short questionnaire.

	Tester A	Tester B	Test No.
Internet Connection Speed	High	High	Test 1
	High	Low	Test 2
	Low	High	Test 3
	Low	Low	Test 4

Table 6.2: Different Connection Speeds of four tests in the Laboratory Experiment

- Completed the final questionnaire (Appendix C5) after finishing the above four tests and four short questionnaires (Appendix C4).
- Interviewed by the experimenter.

During the experiment, the tester could read a "Help and Tips for using the Online Conversation Tool" sheet or press F1 for help. If the tester was still unable to proceed, s/he could ask the experimenter as a last resort.

6.3.5 Test Scenarios

Tasks for each tester in the laboratory experiment are listed in 6.3.4. Appendix C1 shows the detailed steps for each test. Each tester experienced one synchronous conversation and three asynchronous conversations.

When the user clicks the icon of the client application CallClient, all online same group users' photos and other information will be visible on the screen. Figure 6.1 shows a screenshot after two users logged in the CallClient. It shows the status ("Available") and connection speed level (high /low speed) of each online user. Figure 6.2 shows the screenshot when a user invites another available user by clicking the user's photo. The invited user's screen is then shown as Figure 6.3.

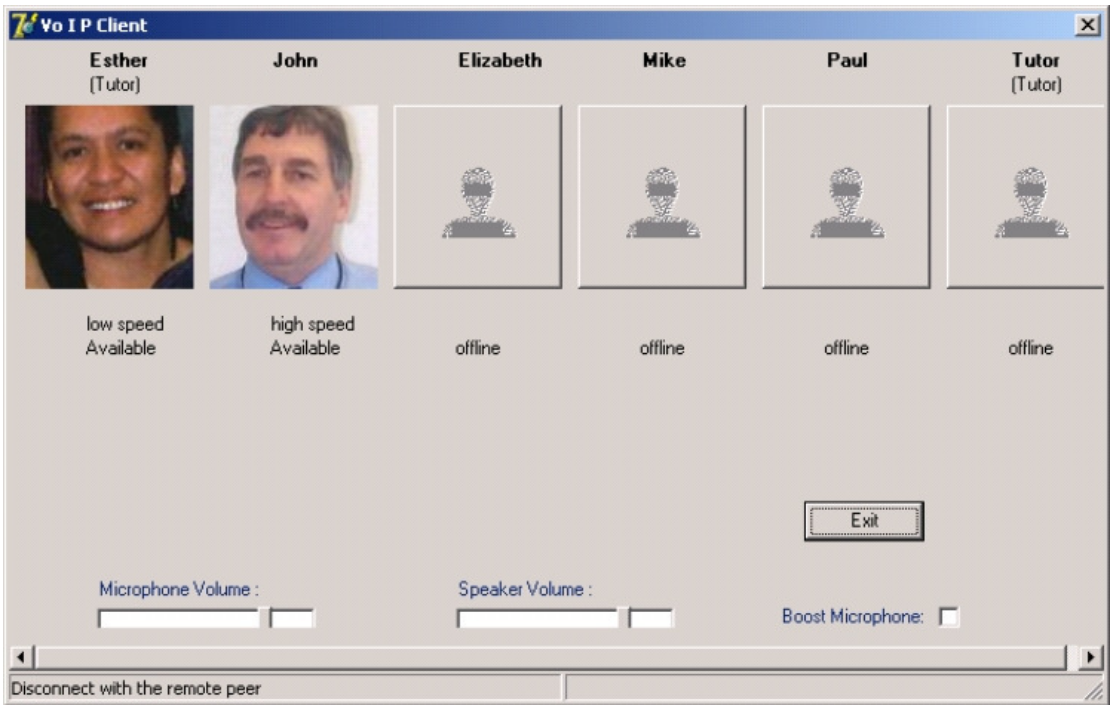


Figure 6.1: Screenshot when Two Users Logs In

Figure 6.2: John's Screenshot when John Invites Esther

Figure 6.3: Esther's Screenshot when Being Invited by John

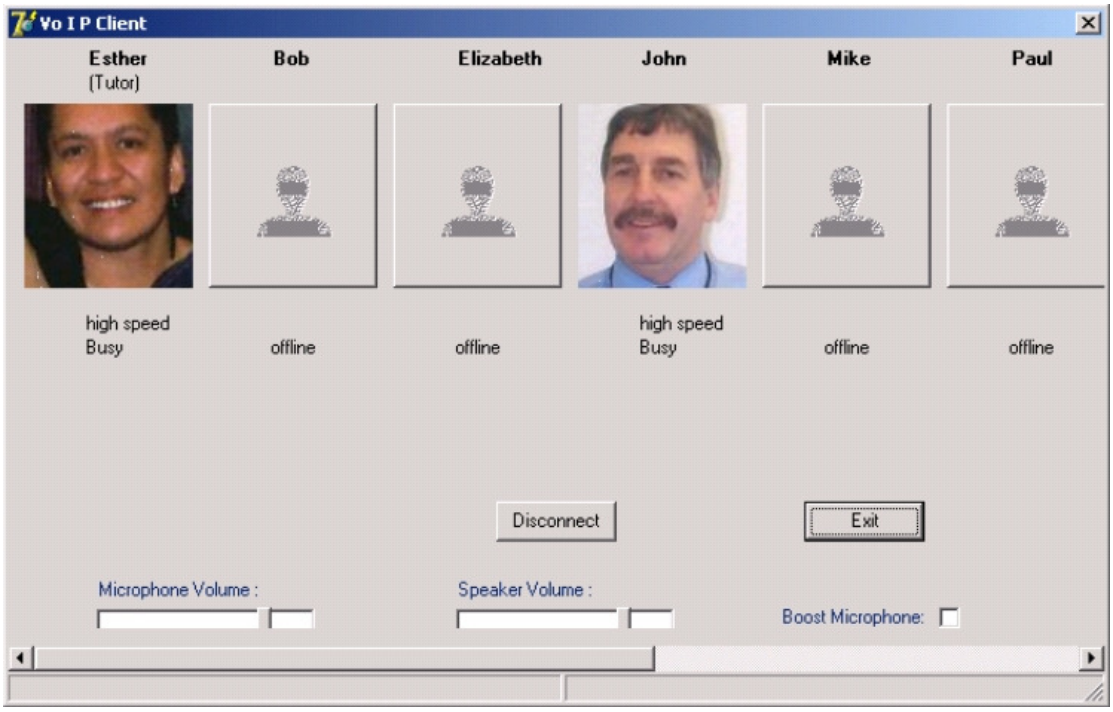


Figure 6.4: Screenshot in Synchronous Conversation

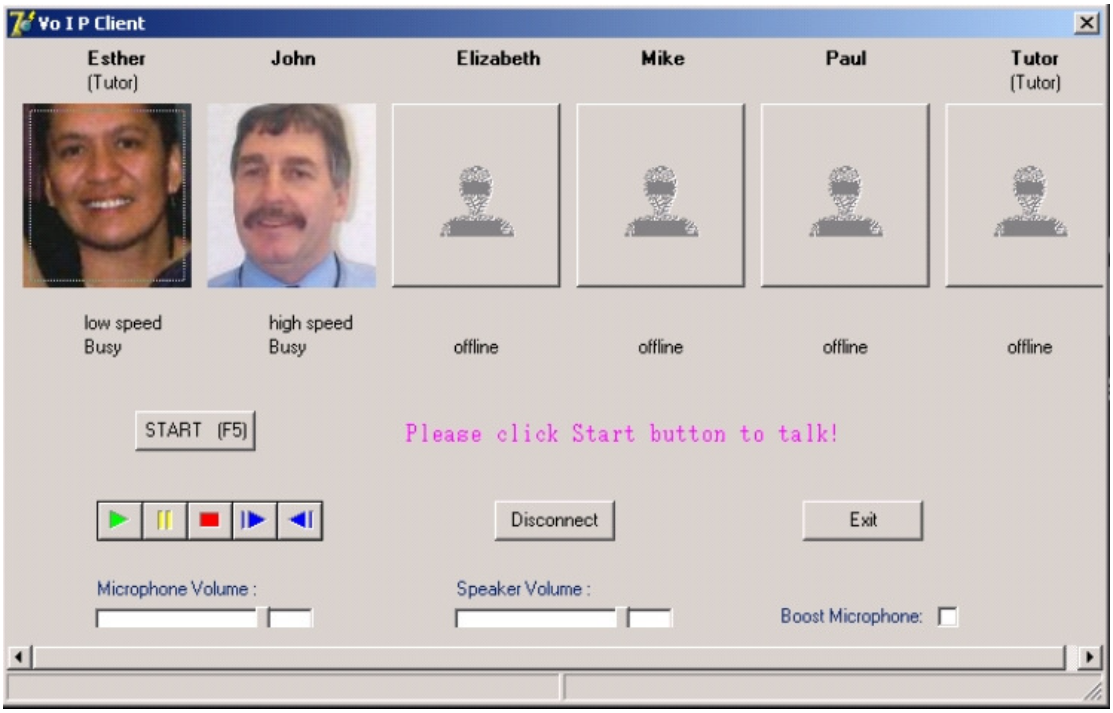


Figure 6.5: Inviter's Screenshot after a Call Invitation (asynchronous)

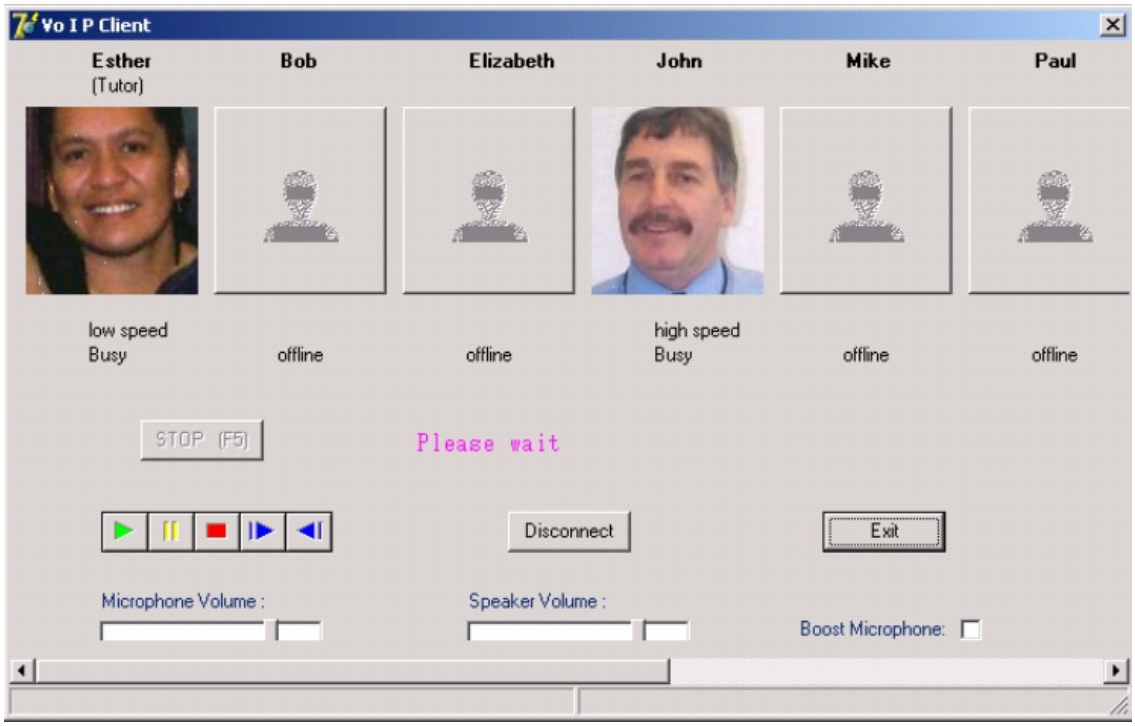


Figure 6.6: Invitee's Screenshot after a Call Invitation (Asynchronous)

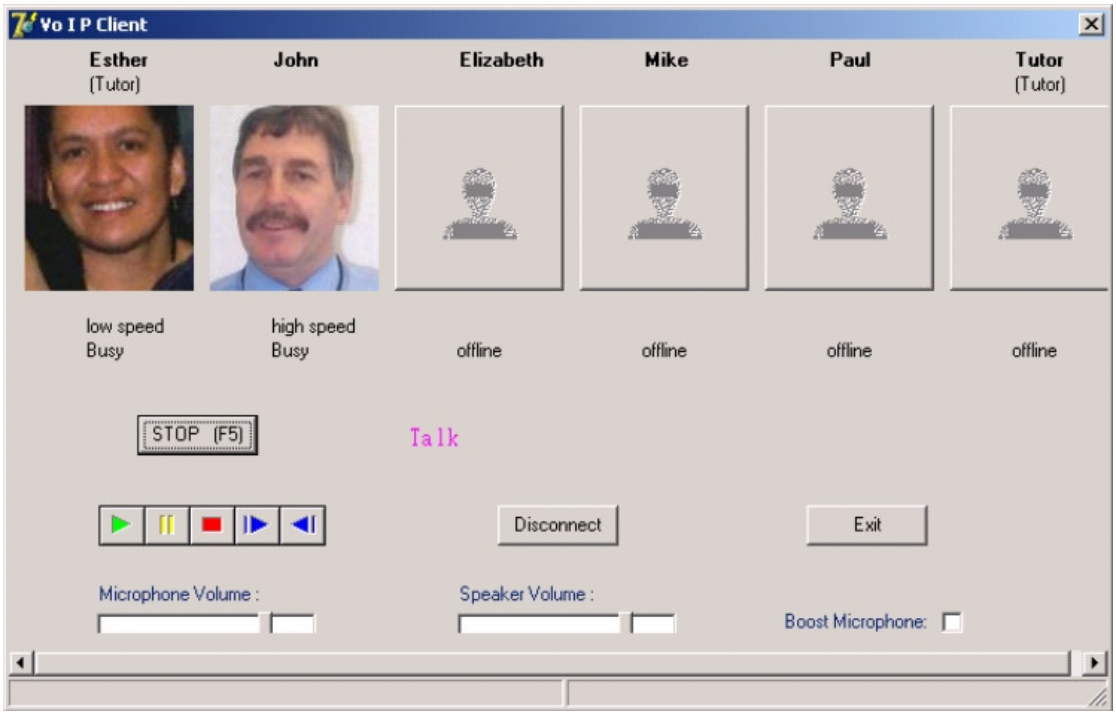


Figure 6.7: User's Screenshot when Talking (Asynchronous)

After the user has accepted a call invitation from another, the status of both parties becomes "Busy". If both users are at high connection speeds, a synchronous conversation interface will be shown on both users' screens (Figure 6.4). If at least one user is at low connection speed, an asynchronous conversation interface (Figure 6.5 and Figure 6.6) will be shown on both sides. The inviter (Figure 6.5) has the right to talk first - by clicking the "Start" button (or F5). The invitee (Figure 6.6) has to wait until the peer finishes talking and clicks the "Stop" button (or F5). Figure 6.7 shows the screenshot when a user is talking in an asynchronous conversation. Then they continue to take turns talking and waiting, until one of them ends the conversation by clicking the "Disconnect" button. When one party terminates the conversation, the status of both parties will then resume from "Busy" to "Available".

It should be noted that a "Boost Microphone" checkbox is displayed on the interface of both the synchronous and asynchronous modes (Figure 6.1 – Figure 6.7). When the speaking person selects it, the voice will be amplified for the listener.

6.3.6 The Pilot Evaluation

The foregoing laboratory evaluation plan had been revised and refined according to a pilot test. In the pilot laboratory experiment, the author acted both as the experimenter and as a tester. Another independent tester (User 1) also participated. Two client PCs were both located in one room. User 1 followed the scheduled plan, held a conversation with the author by using the tool, answered questionnaires, and was interviewed by the author. Results acquired from questionnaires, the interview and recorded log files were analysed.

From the pilot experiment, some adjustments were made in the original evaluation plan. These included:

- Two client PCs should be separated in two different rooms. In the pilot experiment, two testers could hear each other clearly because they were in the same room. The author recalled that in asynchronous conversations, the author sometimes clicked the "start" button to talk soon after hearing the voice of the peer without thinking if the voice was coming from the speaker of the author's PC or from the air.
- The dialogue used is unnecessarily long in asynchronous conversations. Half of the content of the original dialogue was cut out.
- The estimated time was adjusted to 2 hours from the original 2.5 hours
- After analysing the results from questionnaires, the interview and log files of the pilot experiment, the author considered that one or two laboratory experiments would be sufficient to obtain users' views of the tool and system data.
- Log files recorded in the client and server PCs were not enough to analyse the experiment. The FTP server log file was needed for better and more detailed records.
- Before the laboratory experiment, testers should be given a short tutorial about how to operate in the asynchronous mode

After User 1 finished the first test (synchronous conversation) and started the second test (asynchronous conversation), he was confused by the interface of the PTT-like conversation (Figure 6.5 or Figure 6.6). He took it for granted that the other party should hear him instantly after he talked – similarly to the synchronous conversation. When he could not hear from the peer, he did not know either what happened or what to do next. While many people are accustomed to PTT radio communication, this could not be assumed in all cases. Therefore, before the laboratory experiment, testers should be given a short tutorial about how to operate in the asynchronous mode.

6.4 Laboratory Evaluation Results

To conclude the results of the laboratory evaluation, participants' questionnaires, interviews and system's log files in the pilot experiment and the laboratory experiment will be discussed and analysed in this section.

6.4.1 Observations

From the laboratory experiments, it was found that all testers could easily conduct telephone-like synchronous conversations. However, in the first asynchronous conversation, all users (especially User 1 and User 2) became lost until a short instruction was given. After testers had learnt how to operate in the PTT-like way, they said it was easy to handle. Nevertheless, it was observed that after the short instruction, User 1 did not respond quickly when the button changed from "Stop (F5)" to "Start (F5)" to indicate to him to start talking. In the interview, he explained that in PTT-like asynchronous conversation, the party who was waiting needed to be more strongly alerted to talk. In the final questionnaire, he said, "… it lacks communication between the tool and the operator" and suggested "(could be improved) by 'sounding' or 'imaging' other than text".

However, User 2 and User 3 did not give such opinions. This maybe because they were separated in two rooms and they could not hear each other, so they had more focus on the interface and interaction. User 1 did the tests with the author in the same room. This might be the main reason why User 1 was not easily alerted because he might be confused by two kinds of voice he heard – voice from the air and voice from his speaker. However, the effort of highlighting the button text or sounding will still be worthwhile.

User 3 tried to start talking by double-clicking the "Start (F5)" button. Of course, he could not start talking because the button should be clicked only once to start/stop talking. Double-clicking the button means stopping talking instantly after starting talking. The participant profile and interview showed that User 3 used a computer only occasionally. He might have become used to double-clicking on an icon on Windows desktop to start an application.

During the prototyping, there was only one low dial-up speed connection available. The low-to-low connection speed conversation scenario was simulated by one party using low dial-up connection and another party using ADSL connection, and by setting the speed parameter in the initial file with a low speed - instead of a high speed - value. When the prototype was tested under such low-to-low scenario, there was no connection loss between the two parties and the server.

During the laboratory experiments, low-speed dial-up connection was available for each client machine. When testing the low-to-low asynchronous conversation, the connection was sometimes lost between one party and the server. Because running smoothly without breaking down is a key factor for an information appliance, this problem needs to be further investigated.

6.4.2 Questionnaires

In order to capture system performance and other problems, testers were asked to complete a short questionnaire after each of four tests. After all four tests, testers were asked to answer a final questionnaire to gather general users' feedback on the tool.

6.4.2.1 Short Questionnaire

Each tester's four short questionnaires have very similar results (Table 6.3).

For echo and noise questions (Questions 1 & 2), there are three choices – Severe, Acceptable and No noise. In all tests, User 1 chose "Acceptable" for both echo and noise. When being asked in the interview, he said that he thought there was only a small amount of noise and echo. No one complained of any echo or noise problems in either the questionnaires or interviews.

Question & question number	Test No.	User 1	User 2	User 3
1. Echo in the voice	Test 1	Acceptable	Acceptable	No echoes
	Test 2	Acceptable	No echoes	No echoes
	Test 3	Acceptable	No echoes	No echoes
	Test 4	Acceptable	No echoes	No echoes
2. Noise in the voice	Test 1	Acceptable	No Noise	Acceptable
	Test 2	Acceptable	No Noise	Acceptable
	Test 3	Acceptable	Acceptable	Acceptable
	Test 4	Acceptable	Acceptable	Acceptable
3. Overall voice quality	Test 1	Very good	Good	Very Good
	Test 2	Good	Very Good	Very Good
	Test 3	Good	Acceptable	Very Good
	Test 4	Acceptable	Acceptable	Very Good
4. Lost words	All	No	Missing once in Test 2	No
5. Lost words affect understanding	All	/	No	/
6. Problems	All	*	/	*

* Answer not included due to the space limitation

/ Not answered

Table 6.3: Results of Short Questionnaires

User 1 and User 2 both answered “Acceptable” for the “Overall voice quality” in the last test, when both parties used low connection speeds and the time delay was the longest of four tests. When asking User 1 and User 2, they said they chose “Acceptable” just because they were not satisfied with the long time delay, not because of the voice quality itself. All testers said, in interviews, that the voice quality when using the tool was good enough for conversation.

By checking the recorded wave files, it was found that the last word was occasionally lost – it was “trimmed” from a sentence. User 2 reported words lost (Question 4) during the second test. However, User 2 said it did not affect his understanding of the conversation (Question 5). The other users did not detect any word loss, while although there were several lost words. It is concluded that occasional word loss is not a critical factor that affects the understanding or the conversation.

Every tester reported some problems in each test. Most of the "problems" were actually their opinions about the usability and the time delay of the tool, which will be discussed in the following sections.

6.4.2.2 The Final Questionnaire

Table 6.4 lists some results from the final questionnaire. It shows that all testers successfully adjusted the volume of both the speaker and the microphone (Question 1). Two of them also successfully selected or deselected the Boost Microphone (Question 2), although the observation shows that the boosting microphone function might not be needed. All testers thought that the tool was simple enough (Question 3) to carry out conversation tasks without being confused by the two conversation ways (Question 4).

Question & question number	User 1	User 2	User 3
1. Successfully adjust speaker/microphone volume?	Yes	Yes	Yes
2. Successfully (de)select Boost Microphone?	No	Yes	Yes
3. If the tool is simple enough or not?	Yes	Yes	Yes
4. If two conversation ways are confusing or not?	No	No	No
5. Prefer which conversation way? A. Instant (Telephone-like) B. PTT-like	A	B	A
6. Advantages and disadvantages of two conversation ways	*	*	*
7. Prefer which of following methods and why? A. Let the user choose conversation way B. Let the system choose conversation way	B *	A *	A *
8. Suggestions	*	/	*

* Answer or part of answer not included due to the space limitation

/ Not answered

Table 6.4: Results from the Final Questionnaire

User 1 and User 3 preferred the telephone-like conversation way (Question 5) because they thought it was straightforward, efficient and like a real conversation (Question 6). User 1 did not like the PTT-like way because it was "too slow to communicate". Only User 2 preferred the PTT-like way although he hesitated to select this option (Question 5). He thought the advantage of the PTT-like way was "(I) can control the speed easily

myself", while the disadvantage was "not easy to catch up with speaking and listening". User 3 also expressed similar opinions about Question 6. He considered that the disadvantage of the telephone-like way was "you have no time to 'reflect' (on) what you talk or what you hear", while the advantage of the PTT-like way was "you can speak in your own pace" but "It's not real. In the outside world, you cannot speak like that".

From these answers, as well as from User 1's opinion as expressed in the interview, it is suggested that the PTT-like way might be more suitable as the main conversation way for beginning learners of a foreign language, while advanced learners might prefer the telephone-like way – if the choice is available.

The system is designed to choose automatically one of two conversation ways according to both parties' connection speeds. When at least one party is at low speed, the tool has to be used in PTT-like way because the telephone-like conversation way is not possible. When both connection speeds are high, it is technically possible to use either of the two conversation ways. Question 7 asks users' opinion about this. Two testers preferred the user choosing the conversation way rather than the tool automatically choosing, because they thought, "different people have different preferences" (User 2) and "Everybody's learning style is different" (User 3). Only User 1 preferred the system automatically choosing because "Users do not like to do too much except talking. However, users do like to choose when they meet problems". When both parties are at high speeds, if the tool could be designed as the telephone-like synchronous mode while offering the PTT-like mode as an alternative, the tool might be more flexible for some users.

All testers thought that the PTT-like way was too slow when both parties were at low speeds. In the short questionnaires, User 3 expressed his dissatisfaction with asynchronous conversation because "It is very annoying because you have to press the button to start and stop. It makes the conversation very unnatural, very unreal", "It is too slow and too much waiting". After it was explained that if the method was not used then, the conversation could never have happened when one user is at low speed, User 3 said, "OK, it is better than nothing" but (in the final questionnaire) "Try to speed up the low mode, or do some design to occupy the waiting time. It is quite boring to wait. Try to put some 'fun' if waiting is necessary". In the interview, he stated, "The tool should meet users' requirement of happy learning experience".

Although User 2 did not answer the "problems" and "suggestions" question in questionnaires, in the interview, he said that the PTT-like way is too slow. After being asked if he would like to do other tasks when waiting, he said, "That's good. But if the user needs to wait for a long time, he might forget what they were talking about when he returns to the tool". He is a busy person. He said he liked text chat tools because he could leave it and do other things, when he returned he could easily remember what had happened by glimpsing the text chat record on the screen. He also suggested, "It will be very helpful if there is some progress bar showing how long the user will have to wait for the voice".

6.4.3 Log Files

Log files (Appendix C8) were used in both client machines for recording the system's behaviour during asynchronous conversation. After comparing, it was found that the FTP server log file provided more detailed information making it a better resource to analyse the time delay of message exchanging (sending or receiving) with the server during asynchronous conversation.

Time	User	Action	Folder	Speed
10:49:05	John	Uploaded file	CallQueue	29 bytes in 0.2sec. (0.140Kb/s)
10:49:06	Esther	Downloaded file	CallBox	32 bytes in 0.02sec. (1.953Kb/s)
10:49:16	Esther	Uploaded file	CallQueue	32 bytes in 0.02sec. (1.953Kb/s)
10:49:25	John	Downloaded file	John	29 bytes in 0.02sec. (1.770Kb/s)
10:49:48	John	Uploaded file	CallQueue	29 bytes in 0.17sec. (0.165Kb/s)
10:49:49	Esther	Downloaded file	CallBox	32 bytes in 0.02sec. (1.953Kb/s)
10:49:56	Esther	Uploaded file	CallQueue	32 bytes in 0.02sec. (1.953Kb/s)
10:50:03	John	Downloaded file	John	29 bytes in 0sec. (28.320Kb/s)
10:50:23	John	Uploaded file	CallQueue	29 bytes in 0.17sec. (0.165Kb/s)
10:50:23	Esther	Downloaded file	CallBox	32 bytes in 0.02sec. (1.953Kb/s)
10:50:30	Esther	Uploaded file	CallQueue	32 bytes in 0.02sec. (1.953Kb/s)
10:50:36	John	Downloaded file	John	29 bytes in 0.02sec. (1.770Kb/s)

Table 6.5: FTP Server Log File in Asynchronous Conversation

(John is low speed and Esther is high speed)

Table 6.5 is a FTP server log file in an asynchronous conversation when John is at low connection speed and Esther is at high connection speed. The FTP server log file shows that Esther always receives (downloads) the "stop talking" message immediately (0 - 1

seconds) after John sends (uploads) this message, while John takes more time (7 -12 seconds) to wait until his computer receives (downloads) the "stop talking" message from Esther. The speed column in Table 6.5 shows that John's download speed is actually very fast (no more than 0.02 seconds for downloading a file). This indicates that John's 7-12 seconds waiting latency is because his bandwidth is fully occupied by the voice stream, preventing the "stop-talking" message from being downloaded from the FTP server to the client.

That is to say, when user A stops talking (sends a "stop talking" message), if the peer (user B) is at low dial-up speed, the voice stream from user A is still transmitting to user B and occupying B's bandwidth. Therefore, user B cannot download the "stop-talking" message (via TCP) from the server until the voice stream has completely transmitted to user B.

Table 6.6 lists the range of the time delay for sending and receiving messages by analysing log files from both the pilot and laboratory experiments.

	Client A (low)	Client B (high)
Time needed for a client to download the "stop talking" message from the server (after the server received the message)	7 – 12 seconds	0 -1 seconds
Time needed for uploading a Stop message	1 – 2 seconds	0 – 1 seconds

Table 6.6: Time Delay of Control Message in Asynchronous Conversation

The user's waiting time in asynchronous scenarios will be analysed here. In the laboratory experiment, according to the Practice Dialog (Appendix C1), each time a tester needs to speak about 3 - 8 seconds from clicking the "start" button to clicking the "stop" button. After that, the tester has to wait until he receives the "stop talking" message from the peer.

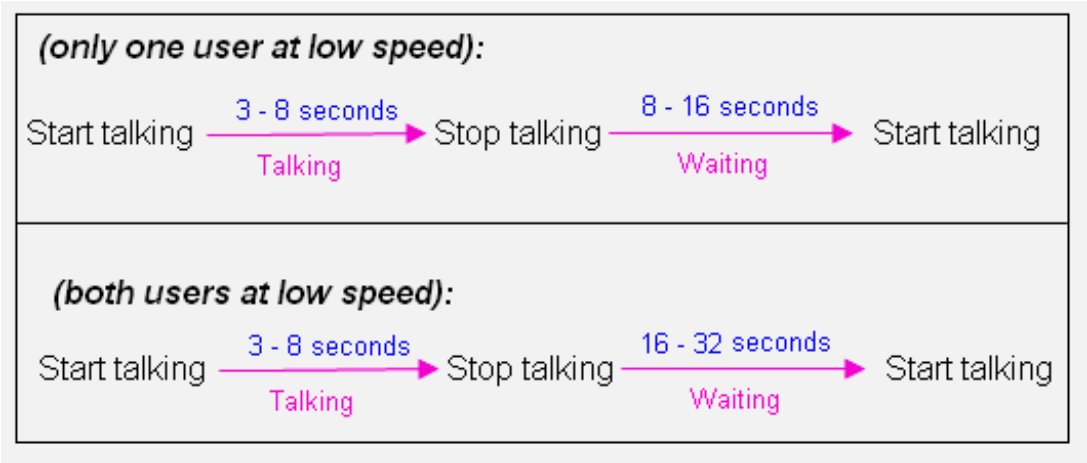


Figure 6.8: Time Delay in Asynchronous Conversation

From the log files, it is estimated that when one user is at low speed and another is at high speed, each user has to wait about 8 – 16 seconds each time after stopping talking until s/he can start talking again. In the experiment, testers complained that it was very annoying that you had to wait every time before you could speak. When both users are at low speeds, each user has to wait around 16 – 32 seconds each time. The waiting time is quite long compared with the 3 - 8 seconds of speaking time. Testers expressed strong objections to such a long waiting time. Figure 6.8 compares the talking and waiting time for each party in an asynchronous conversation. It depicts scenarios of only one user at low speed and both users at low speeds.

In a typical extramural learning environment, the performance of conversation practices between a student and a tutor should be acceptable because at least the tutor should have high speed Internet connection to use IMMEDIATE. In the case where both students are on low speed connections, the performance of conversation practices between them might be very low by using the prototype. Nevertheless, it is still better than nothing - users who have only low dial-up connections have no other cost effective solutions to conduct conversations via PCs.

The foregoing laboratory evaluation results demonstrated the usability, functionality and integrity of the tool. Since the main problem of the prototype becomes obvious – the time lag - the author thought that no more laboratory experiments would be needed before this problem could be solved. To improve the usability of the prototype, the

focus of the future development should be on reducing the time lag in the asynchronous mode.

6.5 Summary

In this chapter I have presented the goal and method of evaluation of the conversation prototype, with details of the laboratory evaluation, and analysed the results. The following conclusion has been drawn from the evaluation.

The overall quality of the voice is very good for conversations with low noise and almost no echo. The user interface is simple to understand and operate. The integrity of synchronous/asynchronous conversation modes works well. The evaluation shows that call functions, user presence, and the PTT-like asynchronous mode have been successfully implemented.

Synchronous conversation is fast, straightforward and telephone-like. Although the asynchronous mode is not very efficient or natural, it might be a very helpful way for beginning learners of a second language. The evaluation reveals that the time lag in asynchronous conversation is acceptable when only one user is at low speed, while it is too long when both users are at low speeds. Because tutors usually have high speed Internet connection, this indicates that student-tutor conversation practice should have acceptable performance. The performance of the student-student conversation can be slow when both students are at low connection speeds. Nevertheless, it is better than nothing – for students having compromised performance is better than having no way to talk at low cost for long distance conversations.

According to the laboratory evaluation, in the next chapter, some suggestions of future work concerning the prototype will be discussed, including a field evaluation, interface refinement, and improving the performance of asynchronous conversation.

Chapter 7

Conclusion

In this chapter I will review the research project, summarise the contribution to knowledge that have been made, and outline further work that could be usefully carried out.

7.1 Review of the Project

The purpose in the project was to design a reusable peer-to-peer conversation practice tool for online second language learning. The tool would be integrated into the IMMEDIATE Learning Computer, and had to be in harmony with its information appliance and invisible computing approach. This meant that it:

- should work over any available Internet connection
- should require no special installation, configuration or initialisation by the user
- should have simple intuitive interface
- should be able to be visibly embedded into the IMMEDIATE architecture

The project grew out of the need to apply the Learning Computer to an extramural second language course at Massey University, New Zealand. Past research efforts and existing systems in e-learning and invisible computing were investigated to acquire a background understanding of the concepts behind the Learning Computer. The large number of distance learning students from diverse backgrounds demands a learner-centred and user-friendly e-learning tool. Intelligent tutoring systems and web-based adaptive learning systems are adaptive to users but they are confined mainly to research prototypes. Learning Management Systems, on the other hand, while widely in use, are learning-provider-centred and do not meet the requirements of extramural students. The goal of the Learning Computer was to provide a learning appliance to enable extramural students - who might only have basic software and hardware, dial-up Internet connection, and little computer literacy - to study university courses.

The importance of communication and collaboration in e-learning was also reviewed. Collaborative learning is especially important in language learning where students need to practise conversation with each other and with their tutors. There are a number of online voice chat tools available, but they are not easy for less experienced computer users to install and use, and do not work over low speed dial-up Internet connections upon which many distance students rely.

An initial conceptual model was proposed involving a bimodal approach to the conversation tool. Under fast Internet connection speed, students would talk in synchronous mode, as in a telephone conversation. Under slow dial-up connection, they would talk asynchronously, in a similar way to PTT communication. In order to let learners focus on the learning tasks, the tool would automatically adapt to one of the two conversation modes according to users' connection speeds.

To give the reader some understanding of computer-mediated voice communication mechanisms, Internet and VoIP technologies were reviewed. The Conaito VoIP SDK was analysed and tested, and then selected for use in the component, to avoid low level implementation requirements. Based on the SDK, the threshold of low and high connection speeds was determined. When users are both at high speeds, synchronous conversation is conducted; otherwise, asynchronous conversation mode is adopted. An initial user interface including the user's presence and a PTT-like conversation was designed. The prototype was to be implemented within the IMMEDIATE architecture for convenience and integrity.

Based on the foregoing specification, a prototype was built and tested. The VoIP SDK was used to implement call procedures including user registration, call invitation and call termination. The IMMEDIATE text messaging method was extended to create new message protocols for dynamically maintaining the users' presence and controlling the PTT-like asynchronous conversation. To ensure the integrity of the conversation component in IMMEDIATE, class templates from IMMEDIATE were extended using the Delphi IDE to develop the prototype.

A laboratory evaluation was then carried out to assess the prototype's usability, functionality and integrity using volunteer testers. The simplicity of the user interface and ease of use of the prototype were verified, although it was found that some initial guidance may be necessary for users without PTT radio experience. It showed that the

prototype could switch between synchronous and asynchronous modes automatically according to different connection speeds. It also proved that the voice quality was very good during conversation. The telephone-like synchronous mode was very efficient. The asynchronous mode was less efficient because of the time lag, especially when two parties were both at low dial-up speeds. Typically, in a second language extramural course, a student is expected to undertake conversation practices with a tutor, who should have broadband Internet access. In such a scenario, the PTT-like way of the conversation tool is still effective even when the student is using a dial-up connection.

7.2 Contribution

Through this project the following main contributions to knowledge have been made:

- The conceptualisation of a way of simplifying and integrating real time conversation practice for second language learning into an e-learning environment, for use by any distance student any time and anywhere there is an Internet connection available. The concept centres on an information appliance that switches automatically between synchronous and asynchronous modes depending on the speed of the Internet connection.

  This approach not only makes long distance conversation affordable, but also lets extramural students practise conversations without worrying about their connection speeds or bothering with installing a third party VoIP communication tool. The asynchronous mode may have advantages for beginning learners to practise conversation skills in second language learning, while the synchronous mode may be more challenging for advanced learners.

- The development of a prototype to demonstrate and evaluate the bimodal conversation concept. The prototyping combined two approaches. One was to use IMMEDIATE templates and a messaging mechanism to create new classes and new message protocols for maintaining user presence and controlling asynchronous conversation. Another was to use a VoIP SDK to implement call procedures.

  The evaluation showed that the basic concept was sound and identified where further research was needed to fully meet the requirements. The most challenging task was to develop a PTT-like asynchronous conversation mode for low speed

connections. The evaluation also showed that the time lag in the asynchronous mode was still a problem when both parties were using low speed dial-up connections.

7.3 Future Work

In this section future work that could be usefully done - including further evaluations, interface refinement, improving the performance of asynchronous conversation, and embedding into IMMEDIATE - will be proposed.

Further Evaluation

A field evaluation should be conducted to evaluate the accessibility of the prototype. To do this, the VoIP server needs a static public address and a special port number to let users outside the university network access it. The threshold of low and high speed (assumed as 40 kbps) needs to be re-evaluated in field evaluations where different low dial-up speeds might be available. Field evaluations will be most effective if they are integrated into an actual distance second language learning course.

Tests are necessary to see whether or not the whole performance of the system might drop, when more than one pair of users conduct conversations at the same time. However, to do this, more resources - such as computers, participants and broadband/narrowband Internet connections – will be needed.

Interface Refinement

A heuristic evaluation should be carried out to refine the user interface. The evaluation should be concluded against a checklist of recognised usability principles.

One possible issue is to modify the interface to more clearly differentiate between the two conversation modes. Neilsen (1993) observes that users can become confused and make errors with a system that has two modes. There was some evidence of such confusion in the evaluation.

Improving the Performance of Asynchronous Conversation

The most important future work is to improve the performance of the asynchronous conversation mode. To decrease the time lag for the asynchronous conversation mode, ways need to be investigated for reducing network traffic including voice data and control messages. For instance, a voice codec (e.g. Speex codec) whose bit rate is lower

than GMS 6.10 should be considered either by finding a suitable VoIP SDK or by direct implementation.

Embedding into IMMEDIATE

The prototype needs to be embedded into IMMEDIATE to evaluate how a second language learner can improve oral language skills using the voice conversation practical tool; and the usability of the conversation component in IMMEDIATE also needs to be evaluated. This requires the CallManager to be integrated into the Repository Manager, and the CallClient to be integrated into the Learning Shell.

References

Aase. J. n.d. *War FTP Daemon.* Retrieved 8 July 2007, from http://www.warftp.org/.

Acrobat Connect Pro, n.d. Retrieved 19 May, 2008 from http://www.adobe.com/products/acrobatconnectpro/systemreqs.

ASTD, 2001: *Learning Circuits Glossary*. Alexandria, Virginia. Retrieved 28 April 2008, from http://www.learningcircuits.org/glossary.html.

Anderson, J.R. & Reiser, B.J. 1985. The LISP Tutor: It Approaches the Effectiveness of a Human Tutor. *Lecture notes in computer science*, Vol. 174, pp. 159 – 175.

Anderson, M. 2005. *Teaching & Learning with Technology: Best Practices for Online Course Design*. Blackboard Asia Pacific Users Conference. Retrieved 5 May 2008, from http://www.blackboard.com/docs/uc05/Building_Communities_of_Practice.ppt.

BBC h2g2, 2000. *MPEG Audio Layer 3 (MP3) Technical Guide*. Retrieved 14 March 2008, from http://inventors.about.com/gi/dynamic/offsite.htm?zi=1/XJ&sdn=inventors&cdn=money&tm=795&f=20&su=p554.2.150.ip_&tt=2&bt=0&bts=0&zu=http%3A//www.bbc.co.uk/dna/h2g2/A157178.

Bellis, M. n.d. *History of MP3.* Retrieved 21 January 2008, from http://inventors.about.com/od/mstartinventions/a/MPThree.htm.

Blackboard, n.d. Retrieved 14 March 2008, from http://www.blackboard.com/us/index.bbb.

Bork, A. 2001. Tutorial Learning for the New Century. *Journal of Science Education and Technology*, March, Vol. 10, No. 1, pp. 57-71.

Braden, R. 1989. *Requirements for Internet Hosts - Communication Layers, RFC 1122*. Retrieved 15 February 2008, from http://tools.ietf.org/html/rfc1122.

Bradford, P. et al. 2007: The Blackboard Learning System: The Be All and End All in Educational Instruction? *Educational Technology Systems*. Vol. 35(3) 301-314, 2006-2007.

Brandon Hall, 2008. *LMSs and lcmss demystified. Sunnyvale*, California. Retrieved at 28 April 2008 from http://www.brandon-hall.com/free_resources/lms_and_lcms.shtml.

Brusilovsky, P. 1996. Methods and techniques of adaptive hypermedia. *User Modeling and User Adapted Interaction*, v 6, n 2-3, pp. 87-129. Retrieved at 28 April 2008, from http://www.sis.pitt.edu/~peterb/papers/UMUAI96.pdf.

Brusilovsky, P., Schwarz, E. & Weber, G. 1996a. ELM-ART: An intelligent tutoring system on World Wide Web. In: C. Frasson, G. Gauthier and A. Lesgold (eds.) Intelligent Tutoring Systems. *Lecture Notes in Computer Science*, Vol. 1086, Berlin: Springer Verlag, pp. 261-269.

Brusilovsky, P., Schwarz, E. & Weber G. (1996b). A tool for developing adaptive electronic textbooks on WWW. *Proceedings of WebNet'96, World Conference of the Web Society*, San Francisco, CA, pp. 64-69.

Brusilovsky, P. & Eklund, J. 1998. A Study of User Model Based Link Annotation in Educational Hypermedia. *Journal of Universal Computer Science*, vol. 4, no. 4 (1998), pp. 429-448.

Brusilovsky, P. 1999. *Adaptive and Intelligent Technologies for Web-based Education*. Retrieved 27 April 2008, from http://www2.sis.pitt.edu/~peterb/papers/KI-review.html.

Carr, H.H. & Snyder, A.S. 2006. *Data Communication & Network Security*. McGraw-Hill, Boston.

Clarey, J. 2007. *E-learning 101: An Introduction to E-learning, Learning Tools, and Technologies*. Retrieve 28 April 2008, from http://www.brandon-hall.com/publications/elearning101/elearning101.shtml.

Compaq, 1998: *Modem Communications - An Overview of Analog Dialup Modem Performance, Environments, and Impairments*. Retrieved 1 April 2008, from ftp://ftp.compaq.com/pub/supportinformation/papers/prt005a0798.doc.

Conaito VoIP Enterprise SDK, 2005. Retrieved 10 May 2007, from http://www.conaito.com/.

Cooper, A. 2004. *The Inmates are running the Asylum*. Sams, Indiana, USA.

Corbett, A.T. et al. 1997. Intelligent Tutoring Systems. *Handbook of Human-Computer Interaction*. 2nd, Elsevier Science B. V., 1997, Chapter 37. Retrieved 17 May 2008, from http://128.2.67.57/papers/173/Chapter_37_Intelligent_Tutoring_Systems.pdf.

Dove, J. 2003. *Sound World: Digital Audio for Your Web Site*. Retried 11 February 2008, from http://webjunction.org/do/DisplayContent?id=1183.

ePipe, n.d. *ePipe VPN and Security Family: Key Networking Concepts*. Retrieved 5 May 2008, from http://www.ml-ip.com/html/documentation/vpn-ug-key-concepts-1.html.

FCC, n.d. *Voice over Internet Protocol*. Retrieved 28 March 2008, from http://www.fcc.gov/voip/.

Freedman, R. 2000. What is an Intelligent Tutoring System? *Intelligence*, 11(3): pp. 15–16. Retrieved 7 Feb 2008, from http://www.cs.niu.edu/~freedman/papers/link2000.pdf.

Gallo, M.A. & Hancock, W.M. 2002. *Computer Communications and Networking Technologies*. Brooks/Cole, USA.

Garrett, R. et al. 2005. *E-learning in Tertiary Education: Where Do We Stand?* Retrieve 3 May 2008, from http://miranda.sourceoecd.org/vl=5474124/cl=36/nw=1/rpsv/cgi-bin/fulltextew.pl?prpsv=/ij/oecdthemes/99980029/v2005n4/s1/p1l.idx.

Gentner, DR. & Grudin, J. 1990. *Why Good Engineers (Sometimes) Create Bad Interfaces*. CHI'90 Proceedings, April, ACM.

GIPS, 2007. Retrieved 28 August, 2008 from http://developer.gipscorp.com/.

Gralla, P. 2004. *How the Internet Works.* 7th ed. Que, USA.

Gui Q. 2003. Mlearning: A New Development towards More Flexible and Learner-Centred Learning. *Teaching English with Technology: a Journal for Teachers of English*, Vol. 3. Retrieved 9 May 2008, from http://www.iatefl.org.pl/call/j_nt13.htm.

Hellwig, K. et al. 1989. *Speech Codec for the European Mobile Radio System*. Global Telecommunications Conference. IEEE, 1989.

Hersent, O., Petit, J.P. & Gurle, D. 2005a. *IP Telephony – Deploying Voice-over-IP Protocols*. John Wiley & Sons, England.

Hersent, O., Petit, J.P. & Gurle, D. 2005b. *Beyond VoIP Protocols – Understanding Voice Technology and Networking Techniques for IP Telephony*. John Wiley & Sons, England.

Hotcoding, 2005. *EMCC Software: An Introduction to PTT* (Revised). Retrieved at 22 March, 2008, http://www.hotcoding.com/os/symbian/36209.html.

Hrastinski, S. 2007. *Participating in Synchronous Online Education*. Retrieve 9 May 2008, from http://luur.lub.lu.se/luur?func=downloadFile&fileOId=600490.

IANA, 2008: *Port Numbers*. Retrieved 10 May 2008, from http://www.iana.org/assignments/port-numbers.

ITU, 1988. *G.711 : Pulse code modulation (PCM) of voice frequencies*. Retrieved 11 March 2008, from http://www.itu.int/rec/T-REC-G.711-198811-I/en.

ITU, 2006. *Recommendation H.323*. Retrieced 6 August 2007, from http://www.itu.int/rec/T-REC-H.323-200606-I/en.

ITU, n.d. *Telecommunication Standardization Sector (ITU-T).* Retrieved 15 March 2008, from http://www.itu.int/ITU-T/index.html.

JANET, 2006. *Skype and JANET*. Retrieved 5 May 2008, from http://www.ja.net/documents/development/voip/skype-and-janet.pdf.

Jayant, N. 1997. *Signal Compression – Coding of Speech, Audio, Text, Image and Video*. World Scientific, Singapore.

Johnson, R., Kemp, E., Kemp, R. & Blakey, P. 2002. From Electronic Textbook to Multidimensional Learning Environment: Overcoming the Loneliness of the Distance Learner. *Proceedings of 2002 International Conference on Computers in Education*, IEEE, Vol. 1, pp632-636.

Johnson, R. 2005. *Developing an extramural e-learning environment to bridge the digital divide*. PhD Thesis, Massey University.

Johnson, R., Kemp, R., Kemp, E. & Blakey, P. 2006. *The learning computer: a low bandwidth tool for bridging the digital divide in distance education*. The Fourth IEEE International Workshop on Technology for Education in Developing Countries (TEDC'06).

Kahney, L. 2006. *Straight Dope on the iPod's Birth*. Retrieve 11 March 2008, from http://www.wired.com/gadgets/mac/commentary/cultofmac/2006/10/71956.

Kasera, S., Narang, N. & Narang S. 2007. *Communication Networks – Principles and Practice*. McGraw-Hill, New York.

Khasnabish B. 2003. *Implementing Voice over IP*. John Wiley & Sons, New Jersey.

Lane, H.C. 2006. *Intelligent Tutoring Systems: Prospects for Guided Practice and Efficient Learning*. Retrieved 12 May 2008 from http://people.ict.usc.edu/~lane/papers/ITSProspectsLane-Aug06.pdf.

Lawson, S. 2004. *Nextel keeps moving on push-to-talk*. Retrieved 8 May 2008, from http://www.infoworld.com/article/04/03/22/HNnextelmoving_1.html.

Leon-Garcia, A. & Widjaja, I. 2004. *Communication Networks - Fundamental Concepts and Key Architectures*. 2nd ed. McGraw-Hill, Dubuque.

LINFO, 2005. *PSTN Definition*. Retrieved 12 May 2008, from http://www.linfo.org/pstn.html.

Marland, P. 1997. *Towards More Effective Open and Distance Teaching*. Kogan Page Limited, London.

Meadow, C.T. 2002. *Making Connections: Communication through the Ages*. The Scarecrow, Lanham.

Miller, D. 2006. *Data Communications and Networks*. McGraw-Hill/Irwin, New York.

Mitchell, M. n.d. *DSL Crib Sheet*. Retrieved 12 May 2008, from http://compnetworking.about.com/od/dsldigitalsubscriberline/l/aa063000a.htm.

Moodle, n.d. Retrieved 2 March 2008, from http://moodle.org/.

Murray, T. et al. 2000. *Evaluating the Need for Intelligence in an Adaptive Hypermedia System*. ITS 2000, pp. 373-382.

Murray, T. Blessing, S. & Ainsworth, S. (Eds.) 2003. *Authoring Tools for Advanced Technology Learning Environments: Toward Cost-Effective Adaptive, Interactive and Intelligent Educational Software*. London: Kluwer Academic.

Nielsen, J. 1993. *Usability Engineering*, Academic, Amsterdam.

New Networks, 2005. *White Paper - Voice Bandwidth Calculation*. Retrieved 31 March 2008, from http://www.newport-networks.com/whitepapers/voip-bandwidth1.html.

Norman, A. D. 1998. *The Invisible Computer*. MIT, Cambridge, Massachusetts.

Olsen, F. 2001. Getting Ready for a New Generation of Course-Management Systems. *Chronicle of Higher Education*, 48:17, pp. 25-28.

Ong, J. & Ramachandran, S. 2000. *Intelligent Tutoring Systems: the What and the How*. Retrieved 7 May 2008, from http://www.learningcircuits.org/2000/feb2000/ong.htm.

Oppermann, R. 1994. *Adaptively supported Adaptability*. International Journal of Human-Computer Studies, pp. 544 – 472.

Rao, K.R., Bojkovic, Z.S. & Milovanovic, D.A. 2002. *Multimedia Communication Systems: Techniques, Standards and Networks*. Prentice Hall PTR, New Jersey.

Rosenberg, J. et al. 2002. *SIP: Session Initiation Protocol, RFC 3261*. Retrieved 4 October 2007, from http://tools.ietf.org/html/rfc3261.

Salami, R.A. 1989. Binary Code Excited Linear Prediction (BCELP): New Approach to CELP Coding of Speech without Codebooks. *Electronics Letters*. 16th March 1989 Vol. 25 No. 6.

Sharp, H., Rogers, Y. & Preece, J. 2007. *Interaction Design: beyond human-computer interaction*. 2nd ed. John Wiley and Sons, Spain.

Siddall, D. 2005. *SMS and Consumer Confusion Hinder Growth of Push-to-Talk*. Q&A-0205-0009 GartnerG2.com © 2005 Gartner, Inc.

Simpson, A. (ed.) 1994. *The Point-to-Point Protocol (PPP), RFC1661*. Retrieved 28 Jan 2008, from http://tools.ietf.org/html/rfc1661#page-6.

Skype, n.d. Retrieved 20 June 2007, from http://www.skype.com.

Skype4Com, n.d. Retrieved 25 March 2008, from https://developer.skype.com/Download.

Smulders, D. 2003. *Designing for Learners, Designing for Users*. Retrieved 21 December 2007, from http://www.elearnmag.org/subpage.cfm?section=best_practices&article=11-1.

Statistics New Zealand, 2006. *Household Use of Information and Communication Technology*. Retrieved 10 March 2008, from http://www.stats.govt.nz/NR/rdonlyres/BA872497-4B85-4386-8395-3ACBEBDA7C4A/0/householduseofict2006hotp.pdf.

Talukder, A.K. & Yavagal R.R. 2007. *Mobile Computing: Technology, Applications, and Service Creation*. McGraw-Hill, New York.

Tanenbaum, A.S. 2003. *Computer Networks* (4th ed.). Prentice Hall PTR, New Jersey.

The GSM 06.10 Lossy Speech Compression. 2006. Retrieve 5 June 2007, from http://kbs.cs.tu-berlin.de/~jutta/toast.html.

Thomas, R. 2001. *Interactivity & Simulations in e-Learning*. Retrieved 8 May 2008, from http://elearning.typepad.com/thelearnedman/files/Interactivity_Simulations_in_eLearning.pdf.

Urban-Lurain, M. n.d. Intelligent Tutoring Systems: An Historic Review in the Context of the Development of Artificial Intelligence and Educational Psychology. *Retrieved 12 May 2008, from http://www.cse.msu.edu/rgroups/cse101/ITS/its.htm.*

VaxVoice Proxy SDK, n.d. Retrieved 23 May 2007, from http://www.vaxvoice.com/VaxProxySDK.asp.

VoIP Foro, 2006. *Latency*. Retrieved 9 March 2008, from http://www.en.voipforo.com/QoS/QoS_Latency.php.

Wang, C. 1997. *Network Application Design Using TCP/IP Protocol in Windows*. Retrieved 9 May 2008, from http://www.ksi.edu/thesis/wangc/index.html.

Weber, G. & Brusilovsky, P. 2001. ELM-ART: An adaptive versatile system for Web-based instruction. *International Journal of Artificial Intelligence in Education*. 12 (4): 351-384. Retrieved 7 May 2008, from http://www.sis.pitt.edu/~peterb/papers/JAIEDFinal.pdf.

Weiser, M. 1991. The Computer for the 21st Century. *Scientific American*. 265(3), pp. 94-104.

Williams, R. 1976. *Communications*. Penguin Books, London.

Windows Live Messenger, n.d. Retrieved 22 October 2007, from http://download.live.com/messenger.

Wisher, R.A. et al. 2001. The Virtual Sand Table: Intelligent Tutoring for Field Artillery Training. Retrieved 8 May 2008, from http://www.hqda.army.mil/ari/pdf/rr1768.pdf.

Wolfram, E. 2005. Learning Management System -- an evaluation of LMS solutions, software and services**.** Retrieved 3 May 2008, from http://dir.wolfram.org/learning_management_systems.html.

Xiph, 2006. *Speex*. Retrieved 9 April 2007, from http://www.speex.org/.

Yahoo Messenger, n.d. Retrieved 25 May 2007, from http://fr.messenger.yahoo.com.

Appendix A

Client Application

A1: The main part of the definition of TPLCVoiceFrame class

```
TPLCVoiceFrame = class(TPLCFrame)
   ActionList1: TActionList;
   ActConnect: TAction;
   ccc: TConaitoConfClientVoIP;
   FTP_Client: TIdFTP;
   Timer: TTimer;
   spkVolume: TTrackBar;
   Label1: TLabel;
   btDisconnect: TButton;
   FileListBox1: TFileListBox;
   btTalk: TButton;
   mPlayer: TMediaPlayer;
   lbCallStatus: TLabel;
   Label2: TLabel;
   micVolume: TTrackBar;
   boostMic: TCheckBox;
   procedure ActConnectExecute(Sender: TObject);
   procedure cccConnectionEvent(ASender: TObject; const UniqueId: WideString);
   procedure TimerTimer(Sender: TObject);
   procedure spkVolumeChange(Sender: TObject);
   procedure micVolumeChange(Sender: TObject);
   procedure cccStatusEvent(ASender: TObject; StatusCode: Integer;const UniqueId:
     WideString);
   procedure btDisconnectClick(Sender: TObject);
   procedure btTalkClick(Sender: TObject);
   procedure boostMicClick(Sender: TObject);
 private
   (some variables …)
   register_finished: boolean;
   tmpmember: TFrameMem;
   procedure errmsg();
   function processTransHeader(var tmpfile: textfile;
            const tnsfile: textfile; var transType: integer): boolean;
   procedure processRegisterResponse;
   procedure processInform(const fname: string);
   function addHeader(var tnsfile: textfile; const transType: integer): boolean;
   procedure clearCallBox;
   procedure processDisconnect;
   procedure tellUser(msg: string);
 protected
   memberlist: TObjectList;
    ••••••
```

A2: The TimerTimer event in TPLCVoiceFrame class

```
procedure TPLCVoiceFrame.TimerTimer(Sender: TObject);
var
  dir: TStringList;
  i,j: integer;
  fname: string;
  outfile: textfile;
begin
   try
     with FTP_Client do
       begin
         if not register_finished then ChangeDir('RegisterBox')
         else ChangeDir('CallBox');
         dir := TStringList.Create;
         List(dir, '*.*', false);
         if dir.Count = 0 then begin
           ChangeDirUP;
           exit;
         end;
         for i :=0 to dir.Count-1 do begin
              fname := dir[i];
           if RetrieveCurrentDir = '/' then
                    ChangeDir('CallBox');
             if RightStr(dir[i],27) = VOICE_REGISTER_RESPONSE
             then begin
                      fname := RightStr(fname,27);
               Get(fname,DB_UPDATES + dir[i], true);
                      Delete(dir[i]);  //delete the file on the FTP server
                      ChangeDirUp;    //to( '\');
                      processRegisterResponse;
                      register_finished := true;
             end else begin
                  if (RightStr(dir[i],length(REGISTER_INFORM)) = REGISTER_INFORM) or
                    (RightStr(dir[i],length(VOICE_INFORM)) = VOICE_INFORM) or
                    (RightStr(dir[i],length(STOP_TALK_INFORM)) = STOP_TALK_INFORM)
                    then begin
                   System.Delete(fname,1,2);
                  Get(fname,DB_UPDATES + fname, false);
                  Delete(dir[i]);  //delete the file on the FTP server
                  ChangeDirUp; //to( '\');
                  processInform(fname);
                      end;
                end;
           end;
        end
   except
      timer.Enabled := false;
      tellUser('Connection to the server has been lost. Exit and re-enter the tool.');
   end;
end;
```

A3: The cccStatusEvent event in TPLCVoiceFrame class

```
procedure TPLCVoiceFrame.cccStatusEvent(ASender: TObject;
  StatusCode: Integer; const UniqueId: WideString);
var
  i, j: integer;
begin
  inherited;
  case StatusCode of
    50:
    tellUser('Connection failed to start conversation.');
    52:
    begin
      tellUser('Connection to the remote end is lost.');
      btDisconnectClick(btDisconnect);
    end;
    53:    // Connection Connected and can start conversation
    for i:= 0 to memberlist.count-1 do
      with memberlist[i] As TFrameMem do
        if uppercase(getusername) = uppercase(curuser) then
        begin
          setstatus(stBUSY);
          isConnected := true;
          btDisconnect.Visible := true;
          btDisconnect.Enabled := true;
          btDisconnect.Hint := 'Disconnect with the remote peer';
          btDisconnect.ShowHint := true;
          btDisconnect.ParentShowHint := true;
          mPlayer.Visible := false;
          btTalk.Visible := false;
          ccc.SetSpkVolume(255);
          ccc.SetMicVolume(255);
          peerdisconnect := false;
          if isAsynchronous then begin
            mPlayer.Visible := true;
            mPlayer.Enabled := true;
            btTalk.Visible := true;
            if callRight then begin
              btTalk.Enabled := true;
              btTalk.Hint := 'Start talking';
              btTalk.ShowHint := true;
              btTalk.ParentShowHint := true;
              btTalk.Caption := START_TALK;
              lbCallStatus.Visible := true;
              lbCallStatus.Caption := 'Please click Start button to talk!';
            end else begin
              btTalk.Enabled := false;
              btTalk.ShowHint := false;
              ccc.MuteMic(true);
              if speed < LOW_SPEED then begin
                ccc.MuteSpk(true);
                lbCallStatus.Caption := 'Please wait';
```

```
            end else begin
              ccc.MuteSpk(false);
              lbCallStatus.Caption := '';
            end;
            ccc.StartRecording;
            lbCallStatus.Visible := true;
            addToLog(' start waiting.');
          end;
        end;
        break;
      end;
  54:
  begin
    peerdisconnect := true;
    tellUser('Remote person closed the connection.');
    btDisconnect.Visible := false;
    btTalk.Visible := false;
    mPlayer.Close;
    mPlayer.Visible := false;
    lbCallStatus.Caption := '';
    processDisconnect;
    isConnected := false;
    peerdisconnect := false;
  end;
  55: showmessage('Remote person is busy.');
  56: showmessage('Remote person is unavailable.');
  57: tellUser('Remote person cancelled start conversation request.');
  58:     //Connection to the server lost
  begin    //close the form
    isConnected := false;
    addToLog('Connection to the server lost. Exit and restart the tool.');
    Timer.Enabled := false;
  end;
  59: showmessage('Person to which you want to add into the conversation is already busy in
conversation with other person');
  60: showmessage('Adding a person into the conversation');
  61: showmessage('Person is added successfully into the conversation.');
  62: showmessage('Person to which you sent ADD conversation request, has cancelled your
request.');
  63: showmessage('Failed to add person into the conversation.');
  64: showmessage('Cryptography not enabled.');
 end;
end;
```

A4: The btTalkClick procedure in TPLCVoiceFrame class

```
procedure TPLCVoiceFrame.btTalkClick(Sender: TObject);
var
  toname, fname, st: string;
  outfile: textfile;
  t: TDateTime;
begin
  if callRight and (btTalk.Caption = STOP_TALK) then begin
      btTalk.Enabled := false;
      btTalk.ShowHint := false;
      callRight := false;
      lbCallStatus.Caption := 'Please Wait';
      addToLog(' click the STOP button, start waiting.');
      fname := curuser + '_' + STOP_TALKING;
      assignFile(outfile, DB_UPDATES + fname);
      addHeader(outfile, vcsStopTalking);
      closeFile(outfile);
      with FTP_client do begin
        ChangeDir('CallQueue');
        Put(DB_UPDATES + fname, fname, false);
        ChangeDirUp;
      end;
      ccc.MuteMic(true);
      if speed < LOW_SPEED then begin
        ccc.MuteSpk(true);
      end else begin
        ccc.MuteSpk(false);
      end;
      lbCallStatus.Caption := 'Wait';
      ccc.ResetRecording;
      ccc.StartRecording;
  end else if callRight and (btTalk.Caption = START_TALK) then begin
      ccc.MuteMic(false);
      ccc.MuteSpk(true);
      mPlayer.Close;
      mPlayer.Enabled := false;
      btTalk.Caption := STOP_TALK;
      btTalk.Hint := 'Stop talking';
      btTalk.ShowHint := true;
      btTalk.Enabled := true;
      callRight := true;
      lbCallStatus.Caption := 'Talk';
      addToLog(' click the START button, start talking.');
  end;
end;
```

A5: The ActConnectExecute procedure in TPLCVoiceFrame class

```
procedure TPLCVoiceFrame.ActConnectExecute(Sender: TObject);
var
  cmp: TComponent;   i, peergrp: integer;
  member: TFrameMem;   ri: boolean;
  peer, peerspeed, fname: string;  outfile: textfile;
begin
  inherited;
  cmp := (sender As TAction).ActionComponent;
  for i:=0 to memberlist.Count-1 do
  begin
      member := memberlist[i] As TFrameMem;
      if member.ismembtn(cmp As TBitBtn) then
           begin
                peer := member.getusername;
                peerspeed := member.getspeed;
                if peer = curuser then
                     showmessage ('You cannot select yourself. Please select another available user
                     to talk.')
                else if member.getstatus = stBUSY then
                     showmessage ('This user is busy now. Please select another available user to
                     talk.')
                else begin
                  if Application.messagebox(PAnsiChar('Would you like to talk to '+
                              peer+' ?'),'',MB_YESNO) = IDYES
                  then
                     begin    //if both is above the low speed, then go to the synchronize mode

                              addToLog(' invited ' + peer);
                              ri := ccc.Connect(peer,20);
                              if not ri then begin
                                errmsg;
                                addToLog(' could not connect to the peer, invitation failed.');
                              end else addToLog(' connected with ' + peer + ', peer is ' + peerspeed );

                              if (peerspeed = 'low speed') or (speed < LOW_SPEED) then begin
                                callRight := true;
                                isAsynchronous := true;
                                ccc.MuteSpk(true);
                                ccc.MuteMic(true);
                              end else begin
                                callRight := false;
                                isAsynchronous := false;
                              end;
                     end;
                end;
             break;
        end;
  end;
end;
```

A6: The cccConnectionEvent event in TPLCVoiceFrame class

```
procedure TPLCVoiceFrame.cccConnectionEvent(ASender: TObject;
  const UniqueId: WideString);
var
  ri: boolean;
  peerspeed: string;
  i: integer;
  member: TFrameMem;
begin
  inherited;
  if MessageDlg('Do you want to accept call from: ' + uniqueID +
              '?',mtInformation,[mbYes, mbNo], 0) = mrYes
  then begin
        addToLog(' accept call invitation from ' + uniqueID);
        for i:=0 to memberlist.Count-1 do
        begin
             member := memberlist[i] As TFrameMem;
                if uppercase(member.getusername) = uppercase(UniqueId) then begin
                  peerspeed := member.getspeed;
                  break;
                end;
           end;
        callRight := false;
        if (peerspeed = 'low speed') or (speed < LOW_SPEED) then
              begin
                isAsynchronous := true;
                ccc.MuteSpk(true);
                ccc.MuteMic(true);
              end else
              begin
                isAsynchronous := false;
              end;
             ri:=ccc.AcceptIncomingCall(uniqueId, 20);
             btDisconnect.Enabled := true;
  end else begin
             ri:=ccc.CancelIncomingCall(uniqueId);
             addToLog(' cancel call invitation from ' + uniqueID);
  end;
  if not ri then errmsg;
end;
```

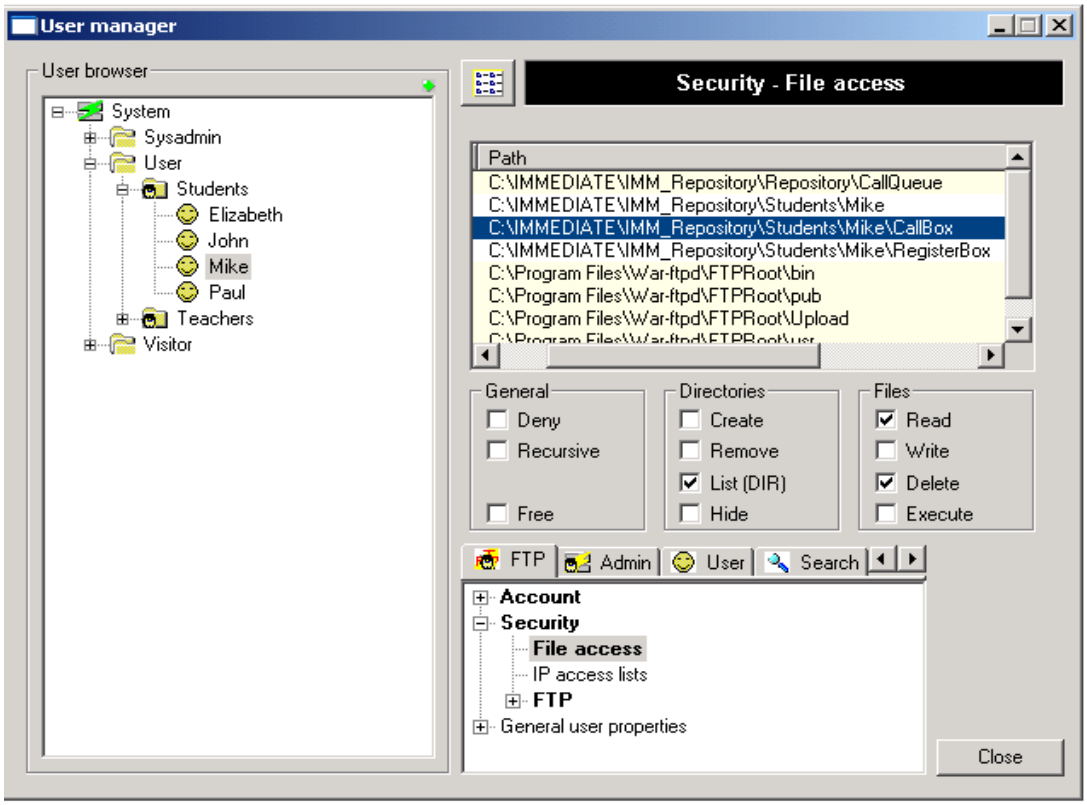
A7: The main part of the definition of plcVoiceMember class

```
TFrameMem = class (TFrame)
private
  memlbname: TLabel;   //group member name
  memlbtutor: TLabel;    //if member is a tutor then display
  memlbspeed: TLabel;  //internet connection speed
  memlbstatus: TLabel;  //online status - available/busy/offline
  membitbtn: TBitbtn;     //display of user name & picture
public
  constructor create (AOwner: TWinControl; ActOwner: TAction; const usernm:string;
groupno:integer; speed:string; status:integer; imgpath:string; var x,y:integer); overload;
  function ismembtn (const selmembtn: TBitbtn): boolean;
  function getusername: String;
  function getspeed: string;
  function getstatus: integer;
  procedure setspeed (const sp: string);
  procedure setstatus (const st: integer);
  procedure setPicture (const imgpath: string);
  procedure setHelp (isavailable: boolean);
    … …
```

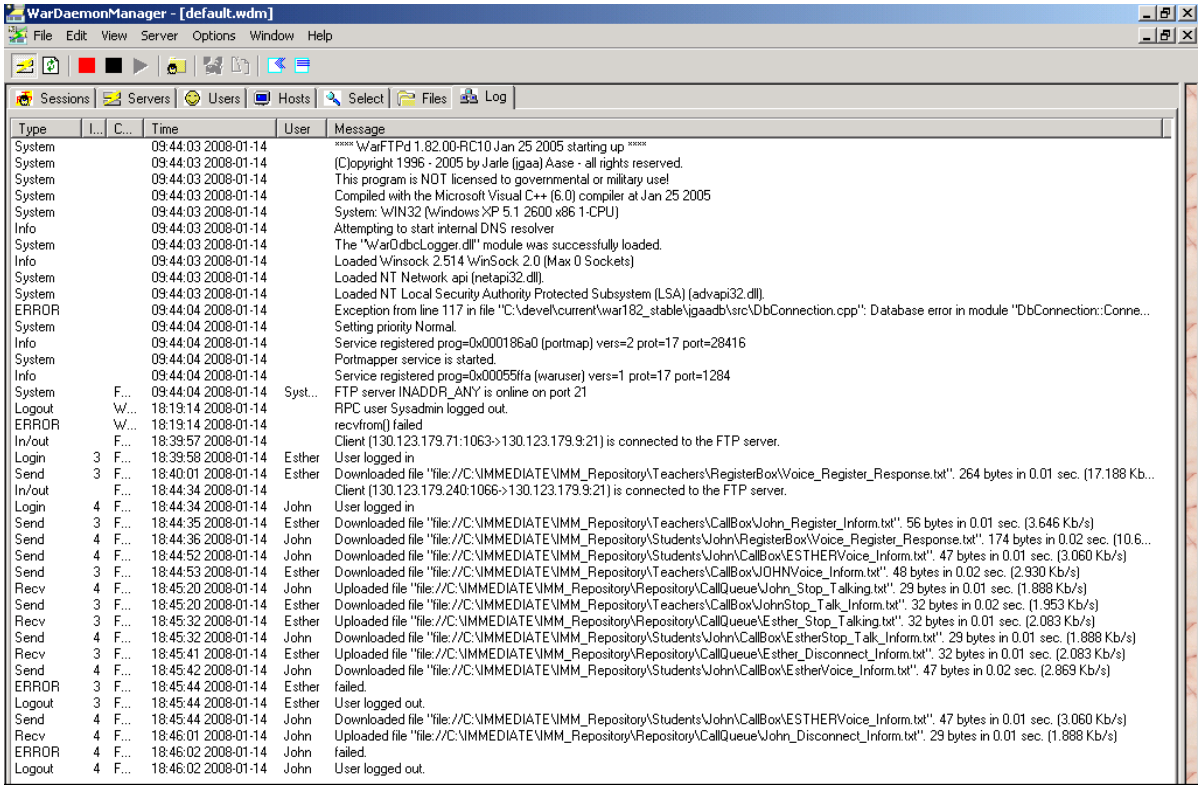
Appendix B

Server Application

B1: WAR FTP Daemon 1.80 – Screens shots



An Example of Mike's Access Right Definition in the WarDaemon FTP Server



An Example of the WarDaemon FTP Server Log in an Asynchronous Conversation

B2: The main part of the definition of TVoIPServer class

```
TVoIPServer = class(TForm)
  ccs: TConaitoConfServerVoIP;
  rc: TListBox;
  dsVoIP: TDataSource;
  QueryVoIPUser: TQuery;
  dsOnlineUser: TDataSource;
  Timer1: TTimer;
  flbCallQueue: TFileListBox;
  procedure FormCreate(Sender: TObject);
  procedure ccsRegisteration(ASender: TObject; const UniqueId, Param1,
    Param2, Param3, Param4, Param5, FromIP: WideString;
    FromPort: Integer; const LocalIP: WideString; LocalPort: Integer);
  procedure ccsConversationAccepted(ASender: TObject;
    EventHandle: Integer; const FromUniqueId, ToUniqueId: WideString);
  procedure ccsUnRegisteration(ASender: TObject; const UniqueId: WideString);
  procedure ccsConnectionLost(ASender: TObject; const UniqueId: WideString);
  procedure ccsConversationConnected(ASender: TObject;
    EventHandle: Integer; const FromUniqueId, ToUniqueId: WideString);
  procedure Timer1Timer(Sender: TObject);
private
  function addHeader(var tnsfile: textfile; touser: string; transType, grpno: integer): boolean;
  function getGroupNo(const userId: WideString): integer;
  function processTransHeader(var tmpfile: textfile; const tnsfile: textfile;
    var transType: integer; var username: string; var groupNo: integer): boolean;
  function findPeer(username: string; var peername: string): boolean;
  procedure informOnlineUsers(const usernm: string; grpno, status: integer);
  procedure processCallList(callFile: string);
  procedure processOffline(const username: string);
  procedure Errmsg;
  procedure getclient;
  procedure addToLog(msg: string);
  ......
```

B3: The ccsRegisteration event of TVoIPServer class

```
procedure TVoIPServer.ccsRegisteration(ASender: TObject; const UniqueId, Param1,
    Param2, Param3, Param4, Param5, FromIP: WideString; FromPort: Integer;
    const LocalIP: WideString; LocalPort: Integer);
  var
    outfile, membfile: textfile;
    queryResult, reguserInfo, speed, username, membname: string;
    groupno, membgrpno: integer;
  begin
    username := UniqueId;
    groupno := strtoint(param4);
    speed := param5;
    ccs.SendParamValueReturned(1001, 'Param Value', FromIP, FromPort);
    ccs.RegistrationDone(username, Param1, Param2, Param3, Param4, Param5, FromIP,
      FromPort, LocalIP, LocalPort);
    getclient;
    dsVoIP.DataSet.Active := true;
    dsVoIP.DataSet.AppendRecord([username, Param1, Param2, param3, groupno,
                    speed, FromIP, inttostr(FromPort), stAVAILABLE,NULL]);
    addToLog(UniqueId +' registered.');

  //Add a message file to the registered user's Register Box
    if groupno = -1 then   //all online users in the same class with the tutor
    begin
      queryResult := AUTHOR_DIR + REGISTER_BOX + VOICE_REGISTER_RESPONSE;
      with DataMod.quOnlineUser do begin
        close;
        open;
      end;
      dsOnlineUser.DataSet := DataMod.quOnlineUser;
    end else   //other online same group member
    begin
        queryResult := STUDENT_DIR + username + REGISTER_BOX +
                              VOICE_REGISTER_RESPONSE;
        with DataMod.quOnlineGroupUser do begin
          close;
          params[0].Value := groupno;
          open;
        end;
        dsOnlineUser.DataSet := DataMod.quOnlineGroupUser;
    end;
    AssignFile(outfile, queryResult);
    addHeader(outfile, username, vcRegisterResponse, groupno);
    with dsOnlineUser.DataSet do
      while not EOF do
      begin
          membname := Fields[0].AsString;
          membgrpno := Fields[1].AsInteger;
          writeln(outfile,membname);
          writeln(outfile,membgrpno);
          writeln(outfile,Fields[2].AsString);    //Connection Speed
          writeln(outfile,Fields[3].AsInteger);   //Online Status
```

```
      writeln(outfile, '[END]');
      if uppercase(membname) <> uppercase(username) then
      begin  //Add a message file to inform all other online users of the registered user
        if membgrpno = -1 then
            reguserInfo := AUTHOR_DIR + CALL_BOX + username+ '_' +
                           REGISTER_INFORM
        else
            reguserInfo := STUDENT_DIR + membname + CALL_BOX + username+ '_' +
                           REGISTER_INFORM;
        AssignFile(membfile, reguserInfo);
        addHeader(membfile, membname, vcRegisterInform, membgrpno);
        writeln(membfile, username);
        writeln(membfile, groupno);
        writeln(membfile, speed);
        writeln(membfile, 1);
        writeln(membfile, '[END]');
        closefile(membfile);
      end;
      Next;
    end;
    if groupno = -1 then
    begin       //all offline users in the same class with the tutor
      with DataMod.quofflineuser do begin
        close;
        open;
      end;
      dsOnlineUser.DataSet := DataMod.quOfflineUser;
    end else
    begin      //other offline same group member
      with DataMod.quOfflineGroupUser do begin
        close;
        params[0].Value := groupno;
        open;
      end;
      dsOnlineUser.DataSet := DataMod.quOfflineGroupUser;
   end;
   with dsOnlineUser.DataSet do begin
    while not EOF do begin
      writeln(outfile,Fields[0].AsString);    //Name
      writeln(outfile,Fields[1].AsInteger);   //Group No
      writeln(outfile,'0');                   //  Connection Speed
      writeln(outfile,0);                     //Online Status: Offline
    writeln(outfile, '[END]');
      Next;
    end;
    closefile(outfile);
  end;
end;
```

B4: The processCallList procedure of TVoIPServer class

```
procedure TVoIPServer.processCallList(callFile: string);
var
        datafile, outfile, infile: textfile;
      tempfile, wavename, filenm, user, peername, speed, st : string;
      tnsType, group, confirm, peergroup: integer;
      ri: boolean;
begin
   tnsType:=0;
   try
        callfile:= RP_CALL_DIR+ callFile;
      assignFile(datafile,callFile);
      tempfile:= REPOSITORY_DIR+ 'tempfile';
      assignFile(infile, tempfile);
      ri := processTransHeader(infile, datafile, tnsType, user, group);
      if not ri then begin
         if tnsType = -1 then
            addToLog('CallManager cannot read from: ' + ExtractFileName(callFile))
         else addToLog('Header Error: Not addressed to CallManager');
         exit;
      end;
      if tnsType = vcsDisconnectInform then begin
         with QueryVoIPUser do begin
                  Close;
                  SQL.Clear;
                  st := 'Update VoIPUser Set status=1, peer=NULL where
                        upper(name)="'+uppercase(user)+'"'+ ' and status=2';
                  SQL.Add(st);
                  ExecSQL;
                  Close;
          end;
          informOnlineUsers(user, group, stAvailable);
          addToLog(user + ' disconnected.');
      end else if tnsType = vcsStopTalking then begin
         if findPeer(user, peername) then begin
            peergroup := getGroupNo(peername);
                  if peergroup = -1 then  filenm := AUTHOR_DIR + CALL_BOX + user +
                                    STOP_TALK_INFORM
               else filenm := STUDENT_DIR + peername + CALL_BOX + user +
                                    STOP_TALK_INFORM;
         end;
         assignFile(outfile, filenm);
         addHeader(outfile, peername, vcStopTalkInform, peergroup);
         closefile(outfile);
         addToLog('CallManager has read a stop talking message from ' + user + '.');
       end;
       deleteFile(tempfile);
       deleteFile(callFile);
   except
      showmessage('Process Call Queue Failed.');
  end;
end;
```

B5: The ccsConversationAccepted event of TVoIPServer class

```
procedure TVoIPServer.ccsConversationAccepted(ASender: TObject;
  EventHandle: Integer; const FromUniqueId, ToUniqueId: WideString);
var
  ri: boolean;
  voiceIP, FromremoteIP, ToRemoteIP: widestring;
  voiceport, grpnofrom, grpnoto: integer;
  st: string;
begin
  FromRemoteIP := ccs.GetRegisteredClientRemoteIP(FromUniqueId);
  ToRemoteIP := ccs.GetRegisteredClientRemoteIP(ToUniqueId);
  If FromRemoteIP = ToRemoteIP Then
    begin           //when the peer on the same machine
      VoiceIP := ccs.GetRegisteredClientLocalIP(FromUniqueId) ;
      VoicePort := ccs.GetRegisteredClientLocalPort(FromUniqueId);
    end else begin   //when the peer on different machines
      VoiceIP := ccs.GetRegisteredClientRemoteIP(FromUniqueId);
      VoicePort := ccs.GetRegisteredClientRemotePort(FromUniqueId);
    end;
  ri := ccs.SetVoiceStreamIPPort(EventHandle, VoiceIP, VoicePort);
  if not ri then begin
    ri := ccs.SetVoiceStreamThroughProxy(EventHandle);
  end;
  addToLog(ToUniqueId + ' accepted an invitation from ' + FromUniqueId);

//Change status of users in the database
  with QueryVoIPUser do begin
    Close;
    SQL.Clear;
    st := 'Update VoIPUser Set status=2, peer="' + ToUniqueId +
       ""+' where upper(name)="'+uppercase(FromUniqueId)+"";
    SQL.Add(st);
    ExecSQL;
    Close;
    SQL.Clear;
    st := 'Update VoIPUser Set status=2, peer="' + FromUniqueId +
       ""+' where upper(name)="'+uppercase(ToUniqueId)+"";
    SQL.Add(st);
    ExecSQL;
  end;

//inform other online users of peers' status change
  grpnofrom := getGroupNo(FromUniqueId);
  grpnoto := getGroupNo(ToUniqueId);
  informOnlineUsers(FromUniqueId,grpnofrom,stBusy);
  informOnlineUsers(ToUniqueId,grpnoto,stBusy);
end;
```

Appendix C

Evaluation

C1: User Testing Steps for the Laboratory Experiment

User Testing Steps for the Laboratory Experiment

- Read the Information Sheet;
- Fill in the Participant Profile Form;
- Read the Help and Tips of using the conversation tool (or press F1 for help) while doing the following test:

1. Double click the "Conversation Tool" icon on the desktop
2. In "VoIP Client" window, invite another available user (double click the user's image) as your peer or accept a call invitation from another user so that the pair can start a conversation.
3. Exchange greetings with the peer to test if you can hear the voice of each other (if necessary, adjust the microphone and speaker's volume bars on the tool, or adjust the volume turner on the headset line, or select or deselect Boost Microphone).
4. Conduct a conversation by practising the following dialogue with your peer.
   - During the conversation, you can make some casual talk when necessary.
   - In the PTT-like way, the inviter must start the conversation at first. There is also a playback button on the screen that you can use to repeat what your have heard.
   - Ensure of both you and your peer to experience the call invitation, disconnection and other functions on the tool.)

<Practice Dialogue: At a Restaurant>

A. Hi. How are you doing this afternoon?
B. Fine, thank you. Can I see a menu, please?

A. Certainly, here you are.
B. Thank you. What's today's special?

A. Grilled tuna and cheese on rye.
B. That sounds good. I'll have that.

A. Would you like something to drink?
B. Yes, I'd like a coke.

A. Thank you. (returning with the food) Here you are. Enjoy your meal!
B. Thank you.

5. One party clicks the "Disconnect" button to end the conversation. The other party responds to the disconnect message. Both click the "Exit" button to quit the tool to the desktop.
6. Both participants do a short questionnaire.

7. There are 4 tests for this experiment to test the tool in 4 different Internet connection speed pairs with your peer (high-high, high-low, low-high, low-low). The tasks and steps are similar for each of tests. Repeat step 1 to 6. The experimenter will reset the speed of your machine every time before a new test.

- Take a break (15 minutes)
- Complete the questionnaire.
- Interviewed by the experimenter.

(Total time estimated: around 2 hours)

C2: Information Sheet

Information Sheet

What is the study about?

The experiment is investigating a possible way for students to conduct conversation practice online for second language learning that will work even with slow dial-up Internet connection speed in rural areas. We use two different conversation methods to meet high and low connection speed situations: telephone-like two-way conversation when both users are at high or medium connection speeds, otherwise walkie-talkie radio-like one-way conversation (where only one user can speak at a time) when at least one user is at low connection speed. The conversation tool integrates these two ways for conversation practice in second language learning, automatically switching between them as necessary. We need to test the usability and functionality of this tool.

Who is conducting the study?

The study is being conducted by Jun Ye as a part of her master's research at Massey University. The research supervisor is Dr. Russell Johnson from the Institute of Information Sciences and Technology, at the Turitea campus of Massey University in Palmerston North.

What will participants do?

The experiment will be held either at a Laboratory in Massey University or at users' own place using their own computers. You will complete a short questionnaire summarising your previous computing experience. You will then be asked to complete some conversation tasks using the tool with another person. You will be interviewed shortly after each test before you finally doing a questionnaire for the whole experiment.

By agreeing to take part in this study, you will be giving permission for me to analyse your data for these tasks as part of this experiment.

How much time is involved?

Each participant will be expected to spend 2 hours on the experiment.

What will happen to the information?

The results will used anonymously without identifying the participants and only for the purpose of this study. The results will be used in the researcher's master's thesis, and might also be published in professional journals or conference proceedings. A brief report of the initial results will be sent to every participant who is interested. When the research is completed, the raw data will be held securely for a suitable time period and then destroyed.

Summary of your rights

While participating in this research project you have the right to:

- Refuse to answer any particular question
- Withdraw from the experiment at any time
- Ask questions about the experiment at any time
- Provide information on the understanding that your name will not be used
- Receive a summary of the findings from the experiment when it is completed

If you have any questions or concerns about the experiment, or would like further information about the experiment, please contact any of the people whose names appear below.

Contact details:

Researcher: Jun Ye

Institute of Information Sciences and Technology

Tel: 356 9099 extn 2461 Email: yecjn@hotmail.com

Supervisor: Dr. Russell Johnson

Institute of Information Sciences and Technology

Tel: 356 9099 extn 4863 Email: R.S.Johnson@massey.ac.nz

C3: Participant Profile

Participant Profile

1. Username:

2. Gender (circle one): Female/Male

3. What is your current job/study field?

4. Computer usage:

   - How many years have you been using Microsoft Windows?

   - In a typical day, how many hour do you spend at a computer?

   - What kinds of purpose do you use a computer? What sorts of software/tools do you use for these purposes?

   - Do you sometimes listen to audio materials (such as music) from a computer? Yes/No

   - Do you understand the difference between a speaker and a microphone? Yes/No

   - Can you control the volume of the speaker and the microphone in Microsoft Windows? Yes/No

   - Have you used an online communication tool such as MSN Messenger or Skype before? Yes/No

5. Is English your first language? (circle one): Yes/No

C4: Short Questionnaire

Short Questionnaire

1. How about the echo in the voice?

   A. Severe B. Acceptable C. No echoes

2. How about the noise in the voice?

   A. Severe B. Acceptable C. No noise

3. How do you find the overall voice quality?

   A. Very Good B. Good C. Acceptable D. Bad

4. Were you aware of some words lost in some sentences? Yes / No
5. If you answered Yes to Question 4, could you still understand the meaning of the sentence in that case? Yes / No
6. Have you met any problem when you using the tool? If yes, can you describe it?

C5: Final Questionnaire

Final Questionnaire

1. Did you successfully adjust the speaker/microphone volume? (Yes/No)
2. Did you successfully select or deselect the Boost Microphone to meet peer's volume expectation? (Yes/No)
3. Do you think the tool is simple enough that you can easily understand and handle? (Yes/No)
4. Did you find confusing between the telephone-like conversation way and the PTT-like conversation way in the tool? (Confused / Not confused)
5. Which way do you prefer for your second language learning?

   A. Telephone-like conversation B. PTT-like conversation

6. What advantages and disadvantages do you think for both conversation ways?

   Telephone-like conversation:

   PTT-like conversation:

7. In the following two methods, which one would you like to suggest to the tool designer?

   A. Let the user choose one of the above two conversation ways in the tool
   B. Let the system choose the conversation way automatically according to the connection speeds of users

   Why?

8. What suggestions do you have to improve the tool?

C6: Interview Question Outline

Interview Question Outline

1. Can you easily understand the use of speaker/microphone volume bars or the Boost Microphone item on the tool?
2. Are you satisfied with the overall quality of the voice in terms of second language learning? If not, what improvement for the tool would you like to suggest?
3. Are you aware of your Internet connection speed (low or high) when you using the tool?
4. Did you meet any problem when using this tool (such as logging in, loggging out, inviting a person for conversation, ending a conversation, or using the speaker/microphone volume bar or the Boost Microphone function)? Where?
5. Do you think the time delay in a PTT-like conversation acceptable or unacceptable? Would you like to be able to do other things while you waiting?
6. What features in the tool do you like?
7. What features in the tool do you not like?
8. What suggestion do you have for improving the tool?

C7: Case Study

Case Study

John and Mike are extramural students of a second language (English) course. John lives in a town and Mike lives in a farm in a rural area. John uses ADSL while Mike uses dial-up to connect to the Internet. They are both in a same group of this course. Esther is the tutor of the course. She connects to the Internet with high speed.

John wants to practise conversations with the tutor or somebody in his group. So he clicks the conversation tool and the screen displays online status of all same group students and tutors. He see only Esther is available now, all others are offline. So he clicks Esther's photo on the screen and the system prompts "Would you like to talk to Esther?" He answers "Yes". Then Esther's screen shows "Do you want to accept call from: John?" Esther answers "Yes". The status of both of them is changed to "Busy" instead of "Available". After they put headphones on head, they can hear and talk with each other just like on telephone. They talks in English. John asks Esther some English language usage questions. He needs frequently to ask Esther to repeat what she just said and thus can clarify words or things that he does not understand. Esther also help John by correcting his pronunciation or expression problems.

During their conversation, Mike also enters the tool. Mike wants to practise some conversation with somebody. However, he cannot talk to either John or Esther as both of their status is busy. When he clicks each of their photos, the tool tells him that he cannot talk to the person because the person is busy. Because nobody else is available so he can only wait and do other things.

After John say good-bye to Esther, John clicks "Disconnect" button to disconnect with Esther. Esther's screen shows "Remote person closed the connection". Both of their screen display "Available" status again. John clicks "Exit" to quit the tool. On Esther's desktop, it shows John is offline now so is not available.

After a while, Mike finds that the tutor Esther is available now so he invites Esther. Esther accepts the invitation. As Mike's dial-up speed only has 31 kbps, so the system displays them with an interface different from the interface that John and Esther have just experienced. Mike clicks a button on the screen to start talking, while Esther cannot

talk but can hear Mike's voice instantly. Mike clicks the button again to indicate that he finishes talking. Then Esther takes turns to talk by clicking a button. Mike cannot hear Esther immediately until Esther stops talking by clicking the button. Then Mike hears what Esther has said. After he finishes listening, he starts to speak by click the button …

C8: Log File Examples

The FTP server log file in an asynchronous conversation

(John is low speed and Esther is high speed)

Time	User	Action	Folder	Speed
10:49:05	John	Uploaded file	CallQueue	29 bytes in 0.2sec. (0.140Kb/s)
10:49:06	Esther	Downloaded file	CallBox	32 bytes in 0.02sec. (1.953Kb/s)
10:49:16	Esther	Uploaded file	CallQueue	32 bytes in 0.02sec. (1.953Kb/s)
10:49:25	John	Downloaded file	John	29 bytes in 0.02sec. (1.770Kb/s)
10:49:48	John	Uploaded file	CallQueue	29 bytes in 0.17sec. (0.165Kb/s)
10:49:49	Esther	Downloaded file	CallBox	32 bytes in 0.02sec. (1.953Kb/s)
10:49:56	Esther	Uploaded file	CallQueue	32 bytes in 0.02sec. (1.953Kb/s)
10:50:03	John	Downloaded file	John	29 bytes in 0sec. (28.320Kb/s)
10:50:23	John	Uploaded file	CallQueue	29 bytes in 0.17sec. (0.165Kb/s)
10:50:23	Esther	Downloaded file	CallBox	32 bytes in 0.02sec. (1.953Kb/s)
10:50:30	Esther	Uploaded file	CallQueue	32 bytes in 0.02sec. (1.953Kb/s)
10:50:36	John	Downloaded file	John	29 bytes in 0.02sec. (1.770Kb/s)

John's log file in the asynchronous conversation

19/12/2007 10:49:00 a.m. click the START button, start talking.

19/12/2007 10:49:04 a.m. click the STOP button, start waiting.

19/12/2007 10:49:26 a.m. receive stop talking message from server and save John_45099.wav

19/12/2007 10:49:37 a.m. click the START button, start talking.

19/12/2007 10:49:47 a.m. click the STOP button, start waiting.

19/12/2007 10:50:04 a.m. receive stop talking message from server and save John_45143.wav

19/12/2007 10:50:11 a.m. click the START button, start talking.

19/12/2007 10:50:22 a.m. click the STOP button, start waiting.

19/12/2007 10:50:37 a.m. receive stop talking message from server and save John_45182.wav

Esther's log file in the asynchronous conversation

19/12/2007 10:48:49 a.m. start waiting.
19/12/2007 10:49:06 a.m. receive stop talking message from server and save Esther_45076.wav
19/12/2007 10:49:08 a.m. click the START button, start talking.
19/12/2007 10:49:16 a.m. click the STOP button, start waiting.
19/12/2007 10:49:49 a.m. receive stop talking message from server and save Esther_45126.wav
19/12/2007 10:49:51 a.m. click the START button, start talking.
19/12/2007 10:49:56 a.m. click the STOP button, start waiting.
19/12/2007 10:50:24 a.m. receive stop talking message from server and save Esther_45166.wav
19/12/2007 10:50:25 a.m. click the START button, start talking.
19/12/2007 10:50:30 a.m. click the STOP button, start waiting.