Mobile Collaboration System

For An Electronic Notebook

Shaoqing Shi
(*shi@cs.brown.edu*)

Computer Science Department
Brown University

March 2001

Advisor:

Professor Thomas W. Doeppner

1. Introduction

Collaboration is a very important application of information technology, especially for an educational environment. With fast development of mobile computing, wireless mobile networks and mobile devices are becoming widely used in both commercial and academic organizations. Collaboration turns out to be much more useful in a mobile network. These scenarios that crave for mobile collaboration include conferences, seminars, and group-based course collaboration. There are also a lot of applications that can integrate collaboration into groupware applications.

In this paper, we discuss a Mobile Collaboration System developed in Department of Computer Science at Brown University. This system is designed as an enhancement to electronic notebook, which is utilized in classroom to foster better interaction in lectures, promote more informed discussion, and ease collaborative work. Using the electronic notebook with mobile collaboration system, students can discuss a topic on a shared white board, share information of common interest between each other, and collaborate based on study groups.

In this system, I designed and implemented two sub-systems – helper system and client system. The helper system simulates multicast and peer-to-peer communication by processing, forwarding, or relaying requests from mobile hosts. The client system, through the helper, provides interfaces for multicast and peer-to-peer applications, and implements several collaboration services over them as well. These services are:

- *Membership Management* – Users can manage their membership information through this service. It relies on multicast protocol.
- *Group Discussion* – Members of a group can discuss a topic of common interest on a shared white board. This service runs over multicast protocol.
- *Instant Notification* – A member can notify its group of interesting information in order for others to retrieve it. This service is also based on multicast.
- *File Searching* – A member can search a piece of information throughout its group by sending “file search” multicast request.
- *File Multicast* – A member can sends information to other group members through reliable multicast.
- *Peer-to-peer File Sharing* – Any host can set up a P2P connection to another host and then retrieve files over the reliable P2P connection.

By integrating two or more basic services above, we can achieve some more advanced services:

- *Group Collaboration* – A group of members can collaborate on a subject by sending File Multicast, one by another.
- *Push-based File Sharing* – A member can multicast a file name by sending an Instant Notification to other members. Those who are interested can get the content through P2P File Sharing at any time.
- *Pull-based File Sharing* – A member can find another member who hosts a specific piece of data by File Searching. With the reply, it can get the file via P2P File Sharing.

The rest of this report is organized as follows. In Section 2 I give an overview of the system design and architecture. Then I discuss the implementation of the helper in Section 3. Section 4 is multicast and its applications. Section 5 is peer-to-peer service and its applications. Section 6 is about programming and testing. Section 7 discusses some open issues. Finally I concluded in Section 8.

1.1 Terminology

Here are some frequently used terms in this report:

- *Mobile Host* – Also called *mobile device* or *mobile computer*, which is a HPC, PPC, or PDA. An example of mobile host is Handheld PC such as NEC Mobile Pro 800.
- *Base Station* – Also called *Base Unit* or *Access Point*, a piece of networking equipment with antennas that has simple routing mechanism. All base stations are connected to form an infrastructure of wireless LAN. Mobile hosts communicate with each other through base station(s).
- *Helper (System)* – A networked desktop computer through which mobile hosts can run simulated peer-2-peer and multicast services.
- *Client System* – The programs running on each mobile host. Those programs are peer-2-peer, multicast, and their applications.

1.2 Guidelines

Considering the current mobile computing environment, I followed some guidelines when designing the mobile collaboration system:

- Overcome computing limitation – One part of the system is to simulate those functionalities that are not supported by the present hardware and software.

- Apply in current environment – Although some functionality is achieved by simulation, this system is supposed to be usable in current wireless LAN environment.

- Consider both online and offline – Since using mobile hosts offline is a very important topic, the collaboration is designed so that it can run both online and offline.

- Migrate to ad-hoc wireless network – The system is to simulate the ad-hoc collaboration on a wireless LAN. This can be easily converted to a real ad-hoc collaboration system once the hardware and software can support an ad-hoc environment.

- Integrate with the whole project – As a sub-project of "Electronic Student Notebook", the design of mobile collaboration system should make it easy to integrate with the whole system of the project.

2. System Overview

Due to limited computing and communication power of mobile devices and wireless network, there are many new issues with mobile collaboration. They are characterized by four constraints[1]:

- *Mobile hosts (devices) are resource-poor compared to static ones*
- *Mobility is inherently hazardous (physical security loss or damage)*
- *Mobile connectivity is highly variable in performance*
- *Mobile hosts rely on a limited battery power*

Those constraints cause problems to the design of mobile information systems and require new approach to some traditional computing problems. Currently, mobile computing proposes two prototypes of mobile network. Infrastructure-based mobile network has two distinct sets of hosts – a large number of mobile hosts, and a few powerful fixed base stations. All base stations and the links between them constitute the fixed network (infrastructure). Each base station covers a geographical area called cell. A base station communicates with the mobile hosts in its cell via a wireless medium (radio channels). Here host mobility means moving between cells.

Unlike infrastructure-based wireless LAN with base stations providing coverage for mobile hosts, ad-hoc mobile networks do not have any underlying infrastructure. Each mobile host itself acts as a router. Due to the migration of mobile hosts, the topology of ad-hoc mobile network is highly dynamic. To facilitate the

communication between mobile hosts, every mobile host needs to run a specific routing protocol. Royer and Toh discuss those routing protocols in a paper[2].

When we began to design Mobile Collaboration System, the network environment was the first prototype – wireless LAN. Mobile hosts communicate with each other via base stations and the static network. Wireless network card does not support peer-to-peer communication. To support collaboration, we need to provide low-level peer-to-peer and multicast protocols.

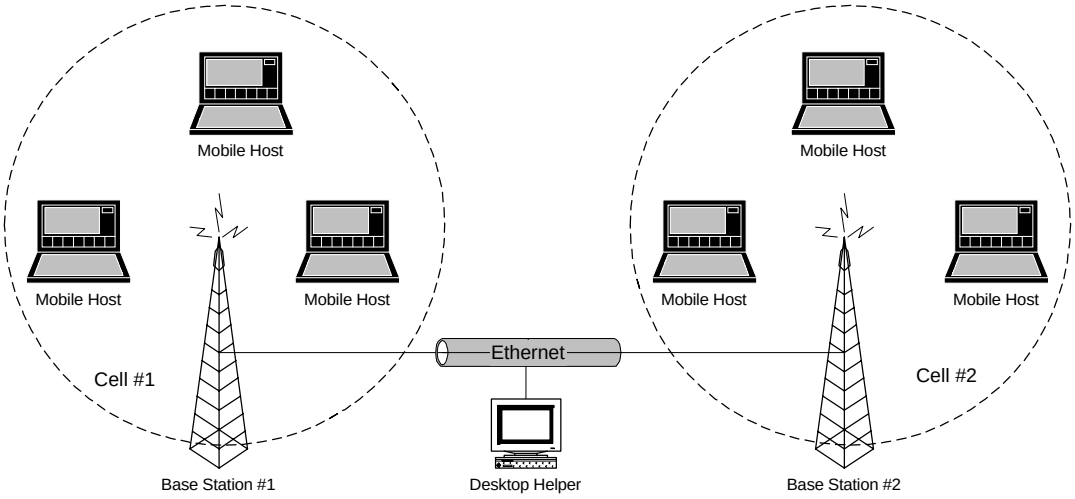

Figure 2.1 System Architecture

The Mobile Collaboration System overcomes those problems by introducing a wired helper computer. Figure 2.1 presents the architecture of the system. Since mobile hosts can always communicate with a wired desktop computer, peer-to-peer can be easily simulated through helper. For multicast simulation, we need a place to store information of multicast groups. This issue exists in both wireless LAN and ad-hoc network. The helper can act as both a group management center and a multicast relay. Therefore, the whole system can be divided into two parts – helper system running on a desktop computer and client system running on mobile hosts. By the hierarchical view, the system can also be divided into two levels – low-level communication system including multicast and peer-to-peer, and multiple services at high application level.

A helper is a wired desktop computer that is much more powerful than a mobile computer. Compared to a mobile computer, a helper is considered to have unlimited resources of computing, communication, memory, and storage. A helper consists of several components. They are request process engine, group management engine, multicast relay, and peer-to-peer relay. The helper is supposed to be running all the time. While it is up, it accepts all kinds of requests from mobile hosts and dispatches

threads of different engines to serve them. The structure of a helper system is shown in Figure 2.2.

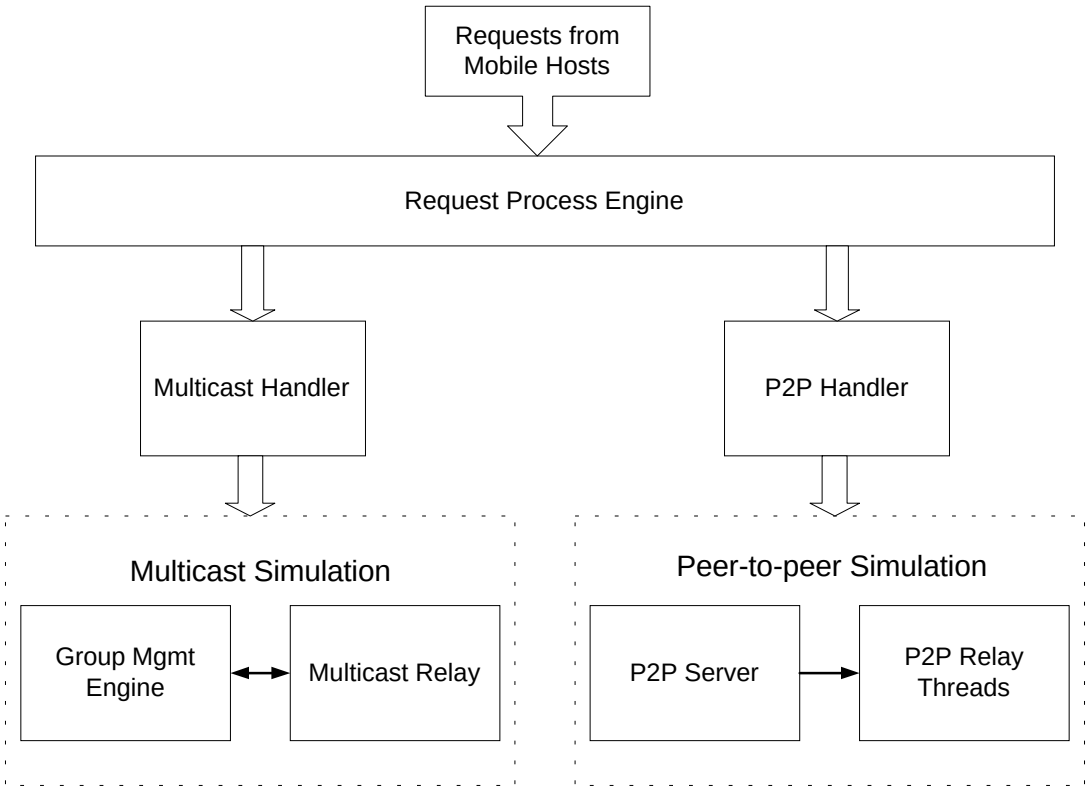

Figure 2.2 Helper System Architecture

A mobile host is typically a handheld PC, although it can be a notebook or a desktop computer as well. The mobile host can connect to the Internet through a wireless LAN. The client system has two categories of functions – multicast services and peer-to-peer services. It also has some mechanism to store shared files. Since multicast and peer-to-peer are low-level communication protocols, it is very convenient to implement multiple services over them. Currently, we have two services upon multicast – group discussion and group collaboration, and one service upon peer-to-peer – group-based file sharing. Some service may make use of both protocols. The client system provides a GUI/API as the interface to users or other applications. The structure of the client system is given in Figure 2.3. The thickness of lines between protocol level and service level stands for QoS. The thicker line means the more reliable service.

For both the client system and the helper system, there is a configuration file through which communication settings can be easily fine-tuned. Membership information is stored in another file at both sides.

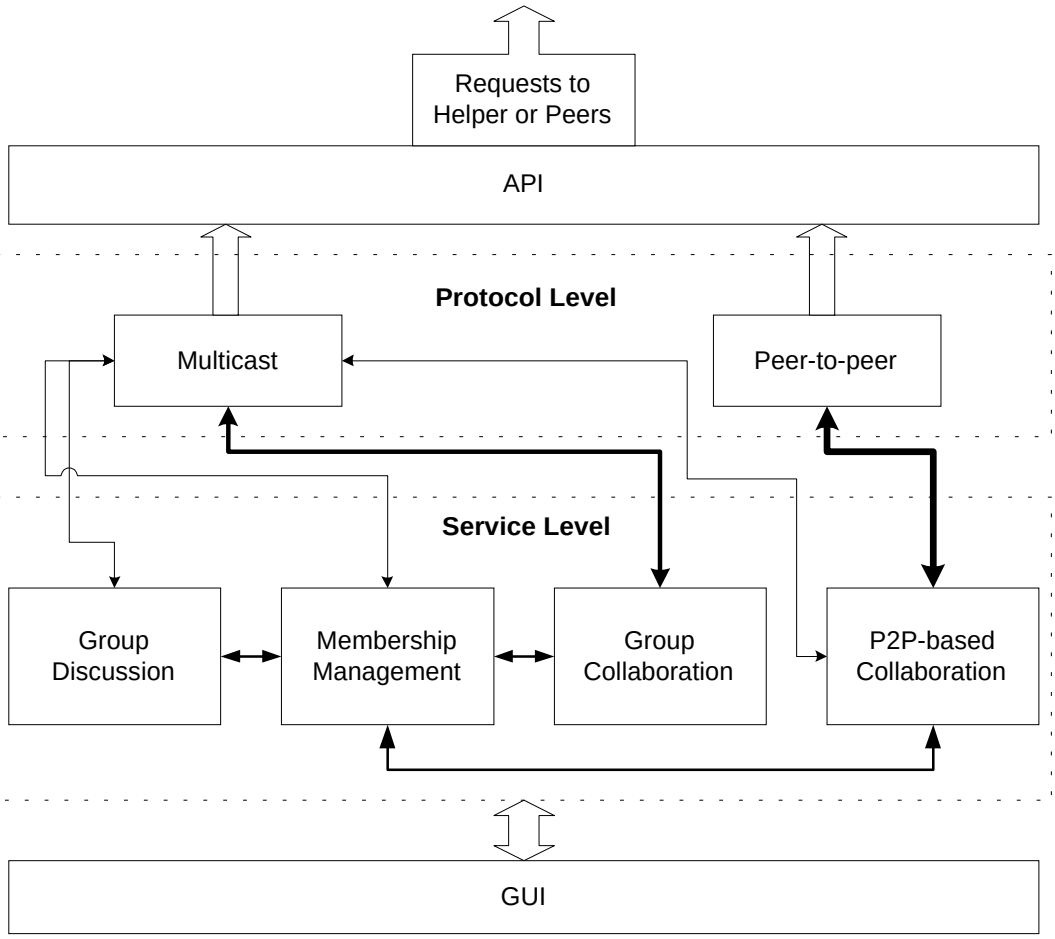

Figure 2.3 Client System Architecture

3. Simulation Via Helper

The helper runs on a wired desktop computer to facilitate multicast and peer-to-peer communications of mobile hosts. In the helper, I implemented two major functionalities – multicast simulation and peer-to-peer relay. Multicast simulation is based on a data-gram socket bound to a specific port. The helper keeps listening to this port for multicast requests from mobile hosts. Each request is served by a separate thread. All threads are synchronized on group information. Peer-to-peer relay is based on a server socket bound to another port. The socket accepts P2P requests from the requesting mobile host, sets up another P2P connection to the serving mobile host, and then builds a pipe from the P2P server to the P2P client by serializing and forwarding information from the server to the client.

As we discussed in Section 2, the helper is necessary because of two reasons. First, the current hardware and software can't fully support peer-to-peer and multicast communications. Second, in spite of the limitation of current platform and network, it is very important and useful to develop a collaboration system as both a test bed and a real application. This helper is designed with the future extension in mind. With new hardware and software supporting more and more features, Helper will be less and less critical in the whole collaboration system. Finally our collaboration system will run without the helper. Refer to Figure 2.2 for the structure of the helper. It functions as a multicast router with add-on peer-to-peer relay agent.

3.1 Multicast Simulation

Multicast simulation deals with two problems. One is centralized group management. The other is serving multicast requests from all mobile hosts. The first role frees mobile hosts from storing and maintaining group information locally. Thus it saves both computation and storage on resource-poor mobile hosts. The second role is a multicast relay. It dispatches a thread for each multicast request from mobile hosts.

Group management is a necessary part of multicast communication. In RFC1112[3] (IGMP), multicast group is identified by a class-D IP address. The system follows this specification and makes some extension for multicast group definition. Here multicast group is identified by either a class-D IP address like 224.1.2.3 or a meaningful name like "CS196-5". Each group has a list of members that are host names. This does not comply with IGMP membership policy that recipients are anonymous. There are a few reasons to do so. First of all, this approach can reduce communication cost of network and computing cost of mobile hosts, which is the first priority for mobile computing. IGMP uses periodic membership query to track groups and sends multicast by putting the multicast packets on each enrolled subnet. Those transmissions are basically local broadcast that incurs high bandwidth consumption. "Anonymous" also means each mobile host itself needs to check each multicast packet, which causes a lot of local computing. Keeping member list minimizes the communication and computing cost above. Secondly, anonymous group scheme of IGMP is based on reliable LAN while wireless network is not as reliable and mobile hosts often get offline. Therefore, it is very likely that no one responds to a membership query although there are still many offline members. Finally, in a wireless LAN, mobile hosts cannot directly hear each other, so all hosts must respond to membership query. The helper gets addresses of all members anyway. As IGMP, the simulated multicast supports both transient and permanent groups. To support transient groups, there are methods to search, add, or delete a group. There is also an initialization file "groups.ini" to store pre-defined permanent groups. Adding, deleting, and querying group members are synchronized among multiple threads serving client requests.

Multicast is simulated using UDP, a best-effort transport protocol. Multicast message has a 24-byte header. The header format is described in Figure 3.1. It is easy to notice that this header is very compact. Based on group management engine, multicast relay serves three categories of requests. The first category is membership request that includes join request and leave request; the second is general multicast request; the third is query-based multicast which means some group member sends a question to the group and wants to hear answers from one or multiple group members. The service type field in the header identifies different multicast request.

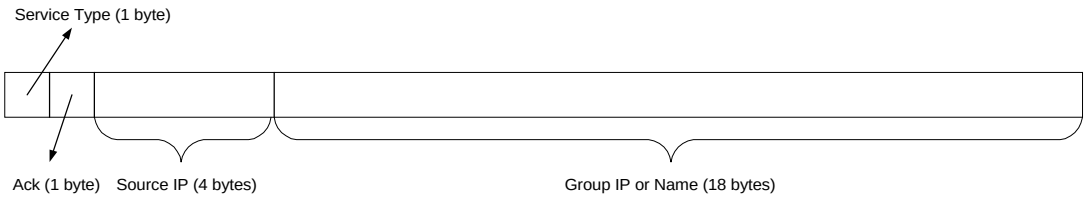

Figure 3.1 Multicast Header Format

The listening thread listens to the simulation port and dispatches a server thread serving each multicast request. For membership request, the server thread talks to group management engine to update valid membership information. For general multicast, the server thread first retrieves addresses of all members except the sender of intended group and then forwards the packet to those addresses by unicast. If the request is a group-based query, after sending the query packet to all other members, the server thread waits for the first reply and then forwards the replier's address back to the requester. When a multicast request is forwarded to members, source IP field in the header is the raw IP address of the requester. If a reply is forwarded back to the requester, the source IP field will be set to the replier's raw IP address. Here raw IP means four-octet representation of an IP address.

3.2 Peer-to-peer Relay

Different than multicast, peer-to-peer simulation is implemented on TCP. This is so because peer-to-peer is mainly used by file sharing that requires reliable transmission. On the other hand, as a computing model opposite to client/server model, many peer-to-peer applications such as shared-space collaboration, instant messaging, and P2P-based search engine can make use of multiple transport protocols.

Peer-to-peer relay actually uses a client/server model. All mobile hosts are clients that request relay service from the helper. The helper runs a relay server that listen to the P2P relay port all the time. Whenever a request comes in, a new TCP connection is established and the relay server starts a relay thread to serve the request. The relay

thread first retrieves the IP address and file name of destination. Then it tries to set up a TCP connection with the destination host. After this, the relay thread can connect two TCP pipes by receiving from the destination and sending to the requester. As a by-product, the peer-to-peer relay can serve request on membership synchronization while client system initializes.

The major problems in the above solution are timeout and closure of TCP connection. Due to limited computing and communication power, mobile hosts have very high packet drop rate and delay latency. Thus a high timeout threshold is necessary. The optimal value is achieved from a lot of experiments and can also be customized by the user. Another issue is to close a TCP connection when the data transfer is finished. One solution is to set a long enough receiving timeout. When there is no more data coming in, the connection finally times out and closes. The disadvantage is, since the timeout value is set very high, it is not efficient for transferring a small amount of data. Another solution is to use close pattern as defined in HTTP upload mechanism. A close pattern is a fixed-length constant byte stream that never appears in a TCP data stream. While receiving data, the receiver tries to find the end of the stream by matching the pattern. The second approach incurs a lot of computing overhead and doesn't fit on mobile hosts. This implementation depends on timeout to maintain and close a TCP connection.

3.3 Customization and Migration

The helper system has two major features – customization and migration. They are close related to each other. Customization means the user can configure all run-time parameters. Those parameters include IP and port addresses, communication timeout settings, packet size, debugging etc. Setting those values is through an initialization file – "helper.ini". This feature lets the helper easily run on any desktop. It also facilitates upgrade of hardware and software in a mobile network. Neither of the changes requires modifying source code.

As mentioned in Introduction, we kept future extension in mind when designing the whole system. Our goal is to finally run the collaboration system without the helper. The multicast simulation is currently required for online collaboration. While in an ad-hoc mobile network where a helper is unavailable, the multicast engine needs to be migrated to the client side. This generally requires some low-level ad-hoc multicast routing protocol running on mobile hosts. Although there are many proposals of mobile multicast protocol[4], a thin and broadcast-based multicast is probably the best choice. A potential scheme is based on flooding with hop-count:

- Mobile host has a way to sense its neighbors.
- Maximum network diameter is assumed known.

- Multicast packet carries hop count in its header.
- Hop-count is decremented each time a packet is re-broadcast (forwarded to neighbors).
- Packets with zero hop count are not forwarded.
- Each host only keeps its own multicast group membership information (i.e. in a local file).
- Every multicast packet is assigned a unique ID as a function of, at least source IP, group ID, and sequence number.
- Each host maintains a cache of IDs for recently processed multicast packets so that it doesn't receive a packet multiple times.

A more complicated approach is proposed to further prune unnecessary re-broadcasts[5]. The paper also discusses some tradeoffs between robustness and efficiency.

Regarding peer-to-peer relay, it is required only when wireless network card doesn't support peer-to-peer communication. Even in this case, peer-to-peer relay is unnecessary when two mobile hosts are on different subnets and thus their communication has to go through base station. When peer-to-peer is fully supported by the network card and the platform, peer-to-peer application can run without relay of the helper. In an ad-hoc wireless environment, besides platform support, there must be some low-level ad-hoc multi-hop routing protocols. Those protocols are divided into two classes – table-driven routing and on-demand routing. The latter is more preferable because the former incurs substantial signaling traffic and power consumption. A practical on-demand mobile routing scheme is AODV[6].

4. Multicast and Its applications

In this part of the client system, I implemented the multicast interfaces and some services over them. The multicast interfaces, according to IGMP, include join, leave, and send multicast. Through the low-level multicast protocol, applications can submit all kinds of multicast requests. The multicast requests always go to the multicast simulation port on the helper. See section 3.1 for details of request processing.

Over the multicast protocol, I generally deployed two applications. One is shared white board or group discussion, where a group of students can discuss their common topic or project. The other is group collaboration that distributes information (slides, annotation, quiz, etc.) to particular groups. Those services require different QoS. Join

and leave request are just UDP messages. Group discussion is built upon UDP while group collaboration needs some level of reliability. This is because the discussion only involves short messages that fit into one packet while the collaboration has to guarantee that all other members receive every packet of a relatively large file.

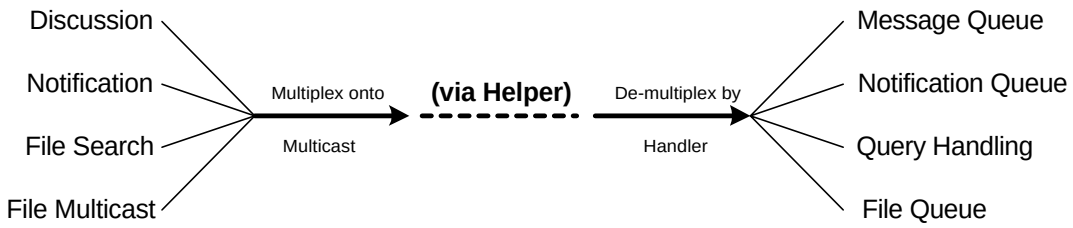

Figure 4.1 Multiple Services over Multicast

Before discussing group management, discussion, and collaboration, let's first look at multiplexing services over multicast. Every client runs a listening thread all the time to receive multicast packets. Like the way SunOS implements "Interrupts as Threads"[7], the listening thread starts a handler thread to process the current packet and continues to receive next packet. This way is to guarantee that the listening thread receives multicast at its full capacity. Many services running upon multicast are multiplexed into the same stream of multicast packets. The handler thread de-multiplexes those multicast packets into different services according to the service type field in the header. Figure 4.1 illustrates this process. All types of service in this figure are all implemented in the system.

4.1 Membership Management

When the client system initializes, it tries to load membership information from "groups.ini" and download group list from the helper. This way, a mobile host can immediately become a member of groups in "groups.ini". Through the UI, a user can join or leave a group and view currently participated groups. Before exit, the changed membership information is written into "groups.ini".

4.2 Group Discussion

Group Discussion is an immediate application of multicast. Members of the same group can exchange information and discuss common-interest topic on a shared white board. This service directly depends on multicast, without any additional layer between them.

The client needs to select a participated group for discussion. It can also disable the discussion so that incoming messages are discarded. As a service on multicast, discussion includes sending and receiving messages. Sending is very straightforward; each message is packed into a single multicast packet and sent out by low-level multicast. Receiving is a little more complicated. It is basically a reader-writer problem with multiple writers. The multicast listener picks up multicast packets from multiple members and the handler puts only those for current discussion into a message queue. Another reading thread retrieves messages from the message queue and displays them in the discussion panel. Synchronization among reader and writers is implemented in the queue. It is a thread-safe FIFO queue.

4.3 Group Collaboration

Multicast-based collaboration is mainly one-to-many file distribution. It is very useful for electronic notebooks. Consider such a scenario: the teacher is presenting a solution for a complicated problem. Every student is viewing it on an e-notebook and thinking of better solutions. Some student may think out one and share it with others; after studying this better solution, another student may give an even better one; and so on. Solution sharing is achieved by the group-based file multicast.

This service is very similar to group discussion but it requires some guarantee of delivery. The file multicast is done in this way: first the file is partitioned into a sequence of multicast packets with the same size (except the last one), then a special packet, which includes meta-data of the file, is sent through multicast for several rounds; after this, all data packets are sent through multicast in round-robin mode for several rounds. At each receiving mobile host, the handler thread tries to assemble the original file by receiving all those packets. Since the receiver may simultaneously receive several files, there is a queue to store incoming files. Each file is represented by a data structure that has a title, an identifier, an IP (from), a status, and content. When the meta-data packet comes in the first time, the handler creates a file object and sends a notification to the UI window. The UI can have two modes of receiving files – quiet mode and interactive mode. In quiet mode, the UI tries to save the file into local disk and then tells the user by a message box; in interactive mode, the UI asks the user whether to save the file or not. Until the file object reaches the status “FINISHED”, the reading thread blocks on it. The file object is removed from the queue afterwards.

5. Peer-to-peer Communication

Peer-to-peer computing technologies have actually been around almost as long as computing itself. Up until recently such systems were exceedingly limited; the reach

of peer-to-peer sharing was confined to the circle of computer users an individual knew and agreed to share files with. An example of such primitive system is Microsoft® workgroup network model. Today, thanks to the development of a number of advanced P2P file sharing applications, the reach and scope of peer networks has increased dramatically. The two main models that have evolved are the centralized model, as used by Napster, and the decentralized model, as used by Gnutella. Although file sharing is a major application of peer-to-peer computing, more and more other applications are developed on peer-to-peer communication. They include collaboration, distributed computation, and instant messaging. If we look closely at the proposed simple multicast protocol in Section 3.3, it is really a P2P-based collaboration infrastructure. Peer-to-peer is especially important in an ad-hoc wireless network, where there is no relatively powerful central server to support client/server computing.

Current Proxim network card does not support peer-to-peer connection. All communication is through the base station. In this part, I implemented both P2P server and P2P client on each mobile host. P2P server is based on a server socket and accepts P2P requests from peers or relayed requests from the helper. P2P client is an interface for applications to submit all kinds of P2P requests. When and only when two peers are on the same subnet, their P2P communication goes through the helper. Through P2P interfaces, I implemented an application to share files between peers.

Peer-to-peer based collaboration includes two ways of file sharing. One is pull-based; the other is push-based. Unlike multicast collaboration, P2P-based collaboration is one-to-one file transfer over reliable TCP. A peer is also called a “servent”, which combines “server” and “client”. Both server and client are running on each mobile host. P2P server is running all the time and accepts request from peers. It starts a P2P thread to serve each request. Currently P2P server can provide two services – get a file or list a directory. A file is transferred in binary mode while a directory can be transmitted as text, one entry in one line. Of course, peer-to-peer has the same problem with TCP closure and timeout.

Pull-based File Sharing

Pull-based file sharing is the general way of peer-to-peer file sharing. The requester first tries to find the address of the host where a specific file resides by “file search” multicast service (refer to Figure 4.1). File search is a multicast query sent to a group. When a group member receives the query, it will reply with its IP address if it has the file. Due to its real-time property, the query is not put into a queue but processed immediately. The requester then sets up a peer-to-peer connection with the first replier and retrieves the file into the local disk.

Push-based File Sharing

Push-based file sharing is a more interesting peer-to-peer communication. If someone (called a “publisher”) wants to share a file within a group, it sends a packet containing the file name by “notification” multicast service (refer to Figure 4.1). When a group member receives this notification, it can decide whether or not to retrieve the file. If a member (called a “subscriber”) wants to receive the file, a peer-to-peer connection to the publisher’s address will be set up and the file will be transferred. Push-based P2P file sharing is very similar to push-based file multicast. But they are different. The former only informs others of the name of the file but does not push the file content. People who get the notification are responsible for file transfer. So traffic from the publisher is peer-to-peer transmission and probably sporadic. It might be more useful if it is designed as an instant messaging engine. Everyone has a list of files published by others and can get any file at any time as they like. The latter not only sends the file name, but also sends the content to all members. Traffic goes in a multicast way.

Because multiple notifications may come in at the same time, there is a queue to store notifications. Thus we need another reading thread to process notifications one by one in this queue. Here is another case of reader-writer problem with multiple writers.

6. Implementation and Integration

The system is coded in Java due to its high inter-operability. Both the helper and the client are tested on the following environment: the helper runs on either a Sun workstation (Ultra 10) or Windows 2000 PC; mobile hosts are NEC Mobile Pro 800 running Windows CE version 2.11; wireless network cards are Proxim cards.

Parameters for communication are based on experiments. For networks and mobile hosts from different vendors, those parameters may need modifying. There are setting files for both the helper (helper.ini) and the client (clinet.ini) to facilitate the modification.

Another important feature of the system is that it is easy to reuse and integrate with the running system. I plugged group collaboration into "Electronic Student Notebook" project, which includes proxy cache[8] and slide viewer. It turns out that APIs of both multicast and peer-to-peer follow the principles of object-oriented design and provide easy and powerful interface for other applications. Peer-to-peer communication can also be integrated with proxy cache to enhance the offline functionality of the system.

The current system requires the helper. When the network and the platform fully support peer-to-peer communication, the system will be able to run without helper. Both the design and programming facilitate the migration to an ad-hoc mobile collaboration system.

7. Open Issues and Future Work

This collaboration system, initially designed and implemented by simulation, can be extended into a real mobile collaboration system. To achieve this goal, we first need to address a few outstanding issues.

The first issue is the naming scheme for shared files. In this system, we used a dedicated folder to store all shared files. Under this folder, there are sub-folders named by the host name from which the local host gets shared files. Each sub-folder contains general files. One problem is that we cannot have multiple versions of the same file from the same host. Another problem is that we cannot organize information by groups. We may use group name as the first-level sub-folder and host name as second-level sub-folder. The best naming scheme should be able to handle multiple versions, shared writes, and views from different perspective. A promising method is to set up a logical naming layer between the physical file system and the application. A hash table can map physical names onto logical names.

The second issue is the naming standard for multicast groups. The easiest way is to name groups by class-D IP addresses. But within a mobile network, without DNS service, IP is neither friendly nor meaningful. This system uses two names for a multicast group – one is class-D IP address, the other is a meaningful name. In an offline ad-hoc mobile network, name is enough for collaboration; in an online wireless LAN, mobile hosts may be a member of traditional multicast group and can use IP for multicast communication. Therefore, we can consider name as local and IP as global. There are also issues related to dynamic group forming and IP-to-name mapping.

The third issue is mobile multicast transport protocol. Even for static network, no IETF standards have yet been established for reliable IP multicast. It is even more difficult to implement a reliable and efficient transport protocol for mobile multicast. It is due to high packet drop rate and low-bandwidth transmission channel. We need a lot of experiments on different approaches, such as selective acknowledgement, before deploying it in real system.

Ad-hoc collaboration is the future of mobile collaboration. We have discussed above some approaches and major issues of migrating this system into ad-hoc networks. First, there need to be some efficient ad-hoc routing protocols for both unicast and multicast. Then we need some reliability over IP layer unicast and multicast. Once we have reliable unicast and multicast, the service layer and UI can work in ad-hoc network with minimal modification. It will be more interesting to integrate ad-hoc collaboration with proxy cache in offline mobile networks.

8. Conclusion

In this article, we discussed a *Mobile Collaboration System* that utilizes both multicast and peer-to-peer communication. As a test bed for mobile collaboration, it gives us an inside view of important functionalities and difficult problems in a mobile collaborative environment; as a real collaboration system, it can be integrated into the *Electronic Notebook System* and facilitates in-class and after-class collaboration.

We gave a detailed review of its design, implementation, and application. We also looked at many open issues with mobile collaboration. Finally we proposed a few directions for future extension and migration.

Acknowledgements

I thank my advisor, Professor Thomas W. Doeppner for his directing and help throughout my project. I also want to thank my colleagues working on project “Electronic Student Notebook”, including Liye Ma, David A. Grunwald, Ravi A. Jeyaratnam, George Cabrera III, Charles Thompson, Peter A. Griess, and Zhongfa Yang, for their helpful discussions and suggestions.

Reference

[1] "Fundamental Challenges in Mobile Computing", M. Satyanarayanan, Invited talk, *In Proc. of Principles of Distributed Computing PODC* 1996

[2] "A Review of Current Routing Protocols for Ad-Hoc Mobile Wireless Networks", E. Royer and C-K Toh, *IEEE Personal Communications Magazine*, April 1999, pp. 46-55

[3] "Host Extensions for IP Multicasting", Steve Deering, *RFC 1112*, August 1989.

[4] "A Performance Comparison Study of Ad Hoc Wireless Multicast Protocols", Sung-Ju Lee, William Su, Julian Hsu, Mario Gerla, and Rajive Bagrodia, *Proceedings of IEEE INFOCOM 2000*, Tel Aviv, Israel, Mar. 2000.

[5] "Multicast Routing Issues in Ad Hoc Networks", Katia Obraczka, Gene Tsudik, *IEEE International Conference on Universal Personal Communication (ICUPC'98)*, Oct. 1998.

[6] "Ad hoc On-Demand Distance Vector Routing", Charles E. Perkins and Elizabeth M. Royer, *Proceedings of the 2nd IEEE Workshop on Mobile Computing Systems and Applications*, New Orleans, LA, February 1999, pp. 90-100.

[7] "Beyond Multiprocessing: Multithreading the SunOS Kernel", J. Eykholt, S. Kleiman, S. Barton, R. Faulkner, A. Shivalingiah, M. Smith, D. Stein, J. Voll, M. Weeks, and D. Williams, in *Proceedings of the Summer USENIX Conference*, (San Antonio, Texas), June 1992.

[8] "Communication and Document-Management System for an Electronic Notebook", Liye Ma, *Technical Report at Computer Science Department of Brown University*, April 2000.