Multimedia Tools and Applications 2, 85–109 (1996)

Design and Implementation of a Multimedia CSCW Platform

ING-CHAU CHANG d1506010@csie.ntu.edu.tw
JAU-HSIUNG HUANG jau@csie.ntu.edu.tw
WEI-HSIN TSENG d0506008@csie.ntu.edu.tw
Communications and Multimedia Lab., Department of Computer Science and Information Engineering, National Taiwan University, Taipei, Taiwan

Abstract. In this paper, a generic four-layer framework for computer supported cooperative work (CSCW) is proposed. With clear separation of functionalities in each layer of the framework, application developers could implement their specific CSCW applications easily. Based on the four-layer framework, the architecture of a general purpose CSCW platform is designed and implemented with the necessary functionalities to help the collaborations among group members. To provide the system with more features and user friendliness, multimedia processing and transmission capabilities for audio and video are incorporated into the platform. Features like application-sharing, group decision support, multimedia mail transmission, etc., are well implemented in the platform. Moreover, an efficient network transport protocol, called VXTP, is designed and implemented to avoid the transmission overhead of conventional TCP. The performance measurements using both VXTP and TCP are given in the paper for comparison. Lastly, compared with other CSCW systems, our design achieves greater flexibility and portability by adopting the generality philosophy in the framework.

Keywords: computer supported cooperative work (CSCW), multimedia, virtual-X transport protocol (VXTP)

1. Introduction

As the interaction among human societies gets more and more closely related, a new type of collaboration, using computers and networks, among geographically dispersed groups of experts is changing the traditional working behavior. Examples of these collaborations are large-scale projects such as the construction of a bridge over a strait, the construction of a subway system in a metropolitan area, or a large computer project developed jointly by several companies. For projects of this nature, it is common that many experts of different fields are required to participate and cooperate in the same project. Usually these experts may live and work in different locations; hence, in order to make such a cooperative work efficient, a computer supported system will be very helpful. This gives birth to the CSCW system, also referred as the groupware [9] system.

The CSCW system refers to groups of people who work for a common goal and seeks to discover how technologies could help them. In [9], a definition for the CSCW system is given as:

> *computer-based systems that support groups of people engaged in a common task (or goal) and that provide an interface to a shared environment.*

Table 1. Taxonomy of the CSCW system.

	Same time	Different times
Same place	Synchronous class e.g., face-to-face meeting	Asynchronous class e.g., shared files
Different places	Distributed synchronous class e.g., teleconferencing, screen and activity sharing, group decision support	Distributed asynchronous class e.g., email, joint design & authoring, newsgroup

While designing a CSCW system which realizes the above definition, two critical obstacles should be overcome. One is time and the other is distance. Groups of people may work at the same time (synchronously) or at different time (asynchronously), and they may be at the same place or at different places (distributedly). With the combination of these two factors, CSCW systems could be divided into four classes, as shown in Table 1 [9].

Our system design is focused on the distributed classes, i.e., the distributed synchronous and the distributed asynchronous classes, in which the synchronous and asynchronous communication supports have to be provided. Moreover, since the nature of works may be very different, it will be impractical to build a general purpose CSCW *system* to efficiently handle all works. Therefore, different CSCW systems may be required for different application domains. Nevertheless, if we examine carefully the features of most CSCW systems, we can observe some common underlying features. Hence, it will be useful if we can provide a general *platform*, which supports all these common features, such that all specific applications can be built upon. With such a platform available, application developers can save a lot of overhead. This is the goal of the platform proposed in this paper. In the following, we name some important features which should be included in the platform.

As mentioned earlier, people involved in the same project may be located in different locations; hence it would be very helpful if the system can provide video conferencing capability for synchronous communication such that the cost of time and money in traveling for meetings can be saved. Other than video conferencing, it would also be helpful to provide these people with multimedia electronic mail systems for asynchronous communication among them. This mail system should support audio, video, image, graphics and text. For these two features, the capability of processing and transmission of multimedia data is required.

For people to jointly design an object, the system should allow these people to share the same view of the designed object in real-time. Hence, a what-you-see-is-what-I-see system is desired in real-time. To avoid conflicts among user demands, capabilities of floor control and concurrency control should also be incorporated for data consistency. Further, in a cooperative work, some decisions would be made under different opinions. In this situation, a distributed voting capability should be provided. For such a distributed voting capability, justice, security and protection mechanisms should be included to avoid malicious intruders.

Besides the above common features, some other features will be addressed in Section 3. Before designing the platform, we first propose a *framework* for CSCW systems. To

achieve greater flexibility and modularity, a layered framework is a good candidate. Hence, a four-layer framework is proposed in the paper. They are Collaboration Support Layer, System Management Layer, Group Communication Layer and Multi-Protocol Network Layer. Detailed description of these layers will be given in section two.

Previous work focused on the design of user interfaces [6, 15], collaborative environment using shared files, designed objects or scientific visualization [2–4, 7, 25, 30, 32], and coordination capability for distributed tasks [8, 18]. Communication issues, such as group communication, multicast transmission, network protocols and architectures, have been considered in [10, 14, 19, 20, 24]. Furthermore, research results have proposed efficient schemes to integrate multimedia data into a CSCW system [21, 22].

The rest of the paper is organized as following. Section 2 presents the four-layer CSCW framework. Based on this framework, the architecture and the common features provided in the proposed multimedia CSCW platform are described in Section 3. The implementation of the platform is illustrated in Section 4. A new transport protocol, namely Virtual-X Transport Protocol (VXTP), which is developed to provide the error-free multicast transmission with much less overhead as compared with Transmission Control Protocol (TCP), is also described in Section 4. Discussions about usability and comparisons with other systems are addressed in Section 5. Lastly, Section 6 concludes the paper.

2. A four-layer framework

We will take the top-down view to examine the four-layer framework. For CSCW application developers, the top layer should support these developers with most required libraries (or, functions) to build their specific applications. This layer is hence named as Collaboration Support Layer (CSL). The layer below CSL should provide the necessary system directory and management function to keep track of all on-going activities. Therefore, this layer is called System Management Layer (SML). Below SML, the system should provide a capability for communication within a group of people or between groups. This layer is called Group Communication Layer (GCL). Finally, the bottom layer will provide various network protocols for connections with different quality requirements. This layer is called Multi-Protocol Network Layer (MPNL). This layering framework is shown in figure 1.

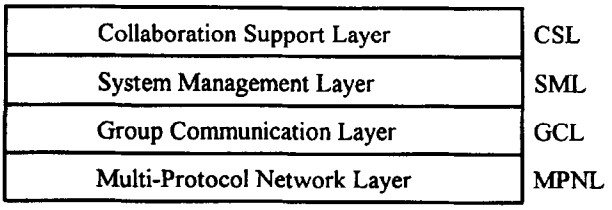


Figure 1. CSCW layering framework.

2.1. *Collaboration Support Layer (CSL)*

The major function of CSL is to provide the required services to the application developers. These services will be implemented as modules and will be called as *agents* in the paper. Each of these agents will provide a specific function to application developers. For instance, these functions may provide multimedia handling capability, application-sharing capability, data security protection, real-time transmission capability, database access capability and load balancing capability.

An application can connect to appropriate agents to acquire the needed services. An agent can also access the services provided by other agents to fulfill its function. For example, to send a mail through MMail Agent, the member list in the system may be needed. In this case, MMail Agent can get the list from Database Agent to access the underlying database system.

With the help of agents in CSL and the functionalities provided by underlying layers, application developers are totally isolated from the details of the platform and only need to focus on the layout design of the user interfaces and the realization of the application semantics. Thus, by calling the primitives in the service access points of the agents shown in Section 4, user customized or domain-specific applications could be developed with less time consumption and efforts.

In the following, we list some of the required features to be supported by CSL. The functionalities of this layer should include:

1. those which support input multiplexing/demultiplexing [25] for single-user applications.
2. those which provide transmission and synchronization services for real-time traffic, such as audio and video.
3. the functions to handle multimedia data, such as media compression and decompression, coding format conversion and other processing functions.
4. the functions to help access to the database system, e.g., the query operations and data format conversion.
5. the functions to support data security, such as data encryption and decryption.
6. the functions to achieve load balancing and sharing in computation.
7. the functions to support the management of the CSCW desktop working environment.
8. the functions to provide group decision support and to achieve group consensus.
9. the functions to administrate the user schedules.
10. the functions to maintain communication channels to System Management Layer to make information access easier.

2.2. *System Management Layer (SML)*

In a CSCW environment, many sessions may be active at the same time where a session may be a video conference. Hence, in order to keep track of the status of the entire system, system management functionalities should be provided which maintain the global information about what sessions are currently active, who are the attendants in each session, what is the topic of the session and other information about the session, etc. Through the

primitives provided by SML, this information can be accessed by agents in CSL to help the collaboration of application users and to coordinate the operations in each session.

Another major objective of SML is for security and protection of the system. The system has to maintain user membership and their respective access rights to system resources, such as hardwares, applications, databases, etc., in a session. Meanwhile, user authentication processes, such as password checking, are necessary to prevent intruders from illegal access to the system. Whenever a user logs into the system and joins a session, the user's identification is added into the session database and the user can access system resources based on the his/her access privilege.

2.3. Group Communication Layer (GCL)

Communication among group of people or among multiple groups is common in CSCW systems, e.g., *multiparty communication* or *group communication*. Normally, a unique group identifier (GID), which is stored in SML, is assigned to a group. Sending messages to a group using GID isolates the applications from worrying about the group membership and the implementation details for group communication, such as maintaining several low-level one-to-one connections.

Clearly, one of the primary functionalities of GCL is to support group communication in an application transparent way. For example, GCL will map GID into each user's ID, which is sometimes referred as *name resolution operation* [19]. The operation of name resolution results in three types of communication: one-to-one, one-to-many and many-to-many. Another function of GCL is to choose an appropriate transport protocol from the Multi-Protocol Network Layer to satisfy the quality of services demanded by its application. For example, GCL may choose TCP for applications requiring error-free transmission and choose User Datagram Protocol (UDP) for real-time applications with multicasting demands.

In sending a message to a group of users, the message should arrive correctly at all members of the group, or at none of them. This property of all-or-nothing delivery is called *atomic broadcast* [31]. Moreover, in sending a sequence of messages to a group, the arriving sequence of the messages at all group members should be maintained the same to achieve a deterministic behavior. This property is important in applications such as database updates. These two features are also the functions to be supported in GCL.

2.4. Multi-Protocol Network Layer (MPNL)

Different applications may require different kinds of network services. For example, the transmission of a connection may or may not have real-time constraints, may require connection-oriented or connectionless services, may demand constant bit rate or variable bit rate services, may require error-free or can tolerate some packet loss. For instance, the transmission of audio and video has real-time constraints but can tolerate some packets to be lost while a file transfer requires error-free transmission without real-time constraints. The requirements of a connection are often referred as the quality of services (QoS) of the connection.

In a network environment, different protocols are proposed for different QoS required by connections. For example, TCP and Transport Protocol 4 (TP4) provide error-free transmissions while UDP provides faster processing time without guaranteeing error-free transmission. There are also other protocols for high-speed networks, such as XTP [28], and protocols for multimedia transmissions, such as MHTP [11].

In CSCW, the system should be equipped with multiple protocols to support various types of traffic since various kinds of applications can be provided in the system. Hence, the function of MPNL is to provide all these protocols to serve the requests generated from upper layers. In this way, the system can provide users with the most flexible and efficient communication environment to execute their work cooperatively.

3. Platform architecture

Based on the four-layer framework outlined in the previous section, we will present the design and implementation of the proposed multimedia CSCW platform. The design of the platform architecture is presented here and the implementation issues will be given in the following section. Basically an application will make use of the capabilities provided by underlying agents in CSL to help the cooperative work. Session Manager (SM) in SML is responsible for maintaining the session information and supporting security and protection mechanisms in the CSCW platform. The layering relationship of the applications, agents and session manager is shown in figure 2. As mentioned earlier, the advantage of separating applications from agents in CSL and other underlying layers of the framework is to obtain the reusability of agents and to achieve high flexibility and portability of the platform.

Our aim in implementing the CSCW platform is to realize the four-layer framework. The functionalities provided by each layer are implemented as agents in CSL, Session Manager in SML, the transport protocols, for example, VXTP, in MPNL, etc. These modules in the

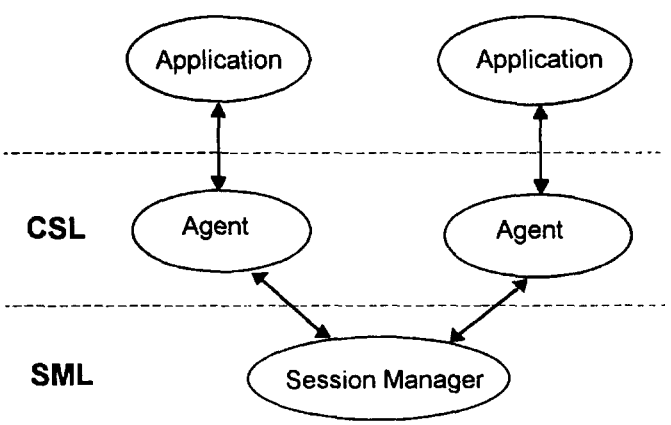


Figure 2. Layering relationship of the applications, agents and SM.

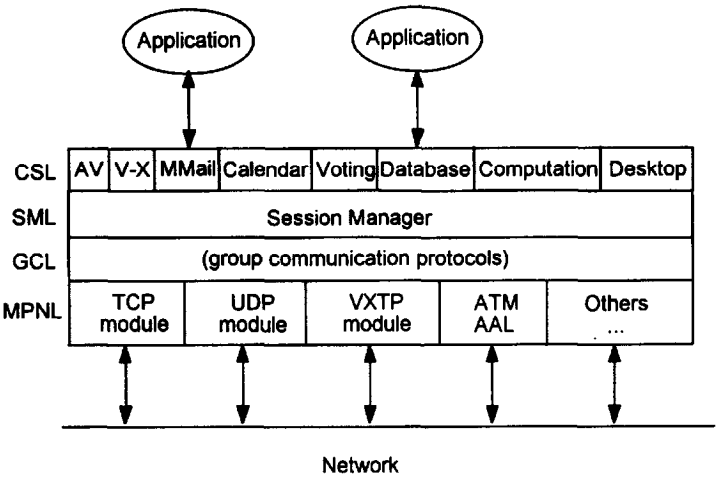


Figure 3. The function modules in the four-layer framework.

four-layer framework are shown in figure 3. System architecture of the platform is shown in figure 4. The relationship among agents, Session Manager, I/O devices and the existing programs such as the database server, X server or a text editor is clearly depicted. In this figure, an ellipse represents an agent and a rectangle represents an existing program. The managers, such as X Window manager or Session Manager, are illustrated as rectangles with rounded corners. The information flows of these processes are indicated by the bi-directional links between any two of them. Supported by different transport modules in MPNL and group communication mechanism in GCL, applications connected to agents such as Virtual-X Agent could achieve real-time (synchronous) transmission, and on the other hand, to MMail Agent the non-real time (asynchronous) transmission.

For CSL, the provided agents and their functionalities are described as following:

(1) *AV Agent*: This agent is very important for processing and transmission of audio and video data, two major data types of multimedia applications. Normally, this agent will require real-time network support from MPNL such that the audio and video quality can be maintained. With this agent, users do not have to know the details of audio and video devices and the underlying network protocols.

(2) *Virtual-X Agent* (*V-X Agent*): The function of this agent is similar to those provided by the shared-X agent as summarized in [5]. That is, it allows all users in the same session to be able to see the same view of the shared applications and to modify and operate on the applications in real-time. The shared object can be an image, graphic, or text. Hence, with this agent and AV Agent, all multimedia data can be supported.

(3) *Calendar Agent*: This agent maintains the daily schedules for users in the system. With this agent, the system can automatically remind its users about what should be done today, or in the following days. Moreover, meeting arrangement of a group of users can be negotiated among Calendar Agents of these users automatically. These

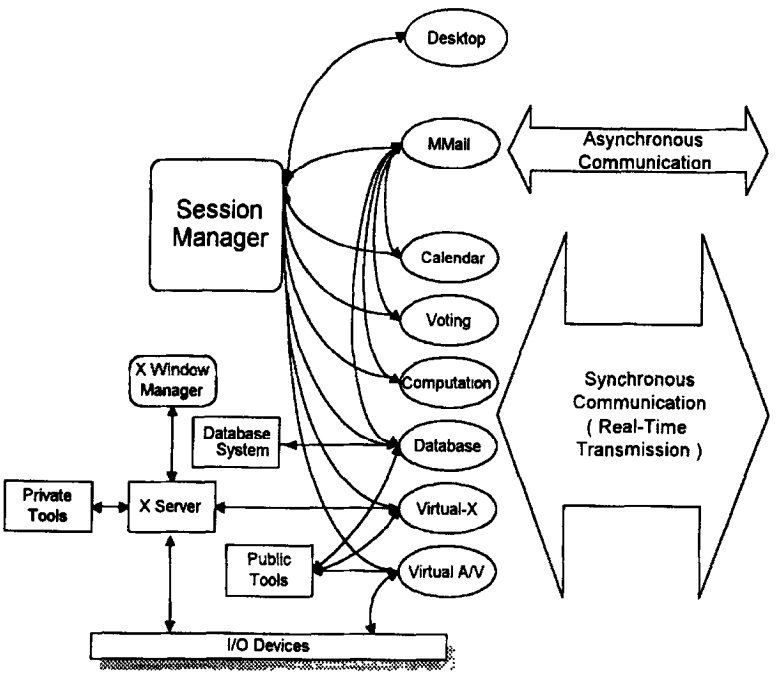


Figure 4. CSCW platform architecture.

arrangements are added into all user schedules after being confirmed. Other than reminding the user at a predefined time for an event, the agent can also activate a scheduled event (for example, start running a program) at a predetermined time.

(4) *Voting Agent*: In a cooperative work, there are times where a decision making should be made based on a majority vote; hence a Voting Agent supporting this function is important for a CSCW system. Such a Voting Agent should be implemented with justice and fairness without human intervention such that the voting result could be trusted by voters at different locations.

(5) *Database Agent*: This agent is responsible for handling query operations of the database system. These queries may come from users or other agents, such as Multimedia Mail Agent as explained later. This agent should also support distributed database system such that all data seems to be local for users.

(6) *Multimedia Mail Agent (MMail Agent)*: This agent supports all functions in traditional electronic mail systems as well as multimedia data types. Our research team has presented this multimedia mail system in [23]. The data structure of the mail format is semi-structured [16] as shown in figure 5 where each mail has two fields of information. One of the fields contains a structured information in which the message type (e.g., for Database, Calendar or Voting Agent) and other necessary information are included. The other field contains the unstructured part of the mail. The video, audio, image and text information of the multimedia e-mail are put in the unstructured field.

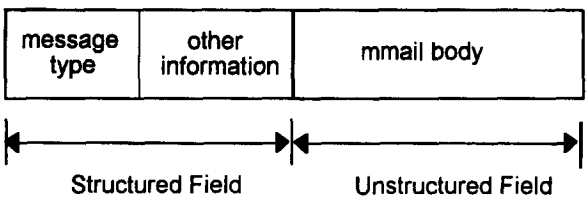


Figure 5. Semi-structured mail format.

Note that this MMail Agent does not only support traditional e-mail functions but also serve as an intelligent server. That is, when MMail Agent receives a semi-structured mail, it first parses the information in the structured field and then decides to which agent it will forward this mail. For example, a mail containing a meeting notice will be forwarded to Calendar Agent to be added into the user schedule.

(7) *Computational Agent*: This agent helps the system to achieve load balancing in the distributed environment for computation-intensive applications, such as graphics or visualization projects. Distributing the computations among processors in the system allows users to largely shorten the processing time if the load balancing function is properly executed.

(8) *Desktop Management Agent*: This agent manages the desktop working environment of the system. It also provides an interface for users to login and this information is checked by Session Manager for user authentication. Whenever the user leaves the system, the agent has to inform Session Manager to correct the session information.

Other than the above agents designed for CSL, a Session Manager should be implemented in SML. Session information such as the membership, login ID, access privilege and location of each user is maintained by Session Manager. Other than the above function, Session Manager also maintains the information and status of all on-going activities.

4. Platform implementation

The prototype system is implemented on workstations connected through an FDDI network. The operating system used is UNIX and both X Windows and Openlook are chosen to build the user interface. Each workstation is equipped with a camcorder, a microphone, a speaker and a video compression/decompression board, which provides motion JPEG compression and decompression function for both image and video data. In this section, we will describe the modules implemented in each layer.

Currently, all CSL Agents outlined in Section 3, except Database and Computation Agents, have already been integrated into the system. The functions of SML is well performed by Session Manager which maintains the session information and performs the security operations. These Agents and Session Manager are run as daemon processes. In MPNL, the VXTP protocol is intergrated into the layer and supports the communication requirements, particularly suitable to the shared applications. Besides these, a user-friendly

desktop working environment is elaborately designed and will be presented in Section 5. However, because the functions in GCL is not the aim of our current implementation, we will skip the discussion of this layer. Related researches about group communication could be found in the ADP-group system [29]. In the following, the implementation of each module in the system will be described.

4.1. Agents in Collaboration Service Layer

4.1.1. AV Agent. Previously, we have implemented a video conferencing system and the design considerations can be found in [12, 13]. In current platform implementation, the audio and video processing capabilities are provided in AV Agent. With the communication support from lower layers, the CSCW system can afford real-time communication services for audio and video applications. The relationship among applications, AV Agent, Session Manager and the underlying system support is shown in figure 6.

Three AV applications, built on top of AV Agent, are currently available as audio phone, video phone and video conferencing. The video processing capability is supported by the primitives of the X video library which is based on X Windows and provided by the video board vendor. For network transmissions, video and audio packets are sent via UDP with multicast capability while control packets are sent through TCP to guarantee error-free services.

Necessary primitives provided by AV Agent is summarized in Table 2. Applications can be easily developed and connected to the platform through these primitives. In Section 5, we will describe a video conferencing application developed upon the platform.

4.1.2. Multimedia Mail Agent. Following the experience from a previously developed multimedia mail system [23], multimedia mails transferred in the system is fully compatible

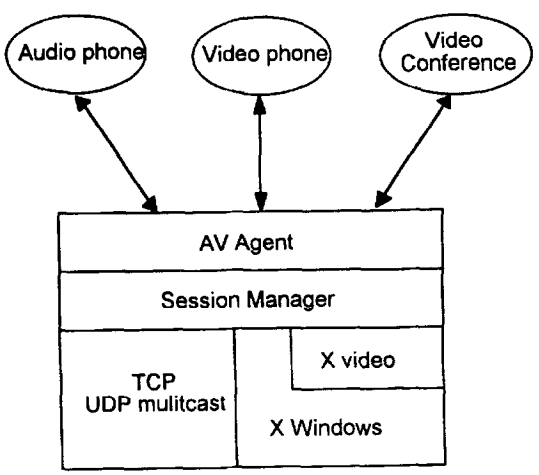


Figure 6. Relationship of AV applications, AV Agent, Session Manager and underlying system support.

Table 2. Primitives provided by AV Agent.

Primitive name	Parameters	Description
AV_Reset	None	Close AV devices and reset system data structure
AV_Quit	None	Terminate AV Agent
AV_Init	AV devices (Audio/Video)	Initialize AV devices and system data structure
AV_Set_Status	Status values	Set system data structure

with Internet e-mail system and follows the MIME (Multiple Internet Mail Extension) standard as its message format. Each medium or text encoded is encapsulated with a necessary header and embedded into the mailing message. The system also supports external-body content type, that is, the actual bodies are not included, but merely referred by remote site referencing, either by anonymous FTP or FTP. This implies that a user can reduce the mail size by obtaining some data from other sites.

The main purpose of Multimedia Mail (MMail) Agent is to pre-process the incoming mails. It intercepts the mail received from the network and parses the contents before forwarding it to the mail spool. If different types of medium data are contained in a mail, MMail Agent will decode and separate them from the mail body, then store them in different files. This mail is then modified to refer to these external files. For example:

Content-Type: message/external-body
name = /cmlab4/project/cscw/MMMAIL/doc.av

The medium is stored as *doc.av* in the file system. While the mail receiver reads the content of the multimedia mail, the mail application has to read this file when needed. After the preprocessing, the mail is put into the mail spool waiting to be read. This operation is shown in figure 7. The preprocessing reduces the size of the mail such that the time spent on parsing a mail can be greatly reduced.

4.1.3. Virtual-X Agent. Virtual-X Agent (V-X Agent) performs the functionalities of the *X Protocol Multiplexor* [5]. It allows a single-user application to be displayed on the screens of multiple users at the same time. Furthermore, it regulates the user inputs to the application by transferring the application floor to users one at a time, such that the user who owns the floor can modify and operate on the application in real-time. Supported by this function, the single-user application can be used by multiple users for their cooperative work. Details of V-X Agent can be found in [17]. The subsystem we developed is based on the XTV system in [1]. However, the network transmission of XTV is inefficient since TCP is used. The problem of TCP is that TCP does not provide the service of multicasting, which is important in X protocol multiplexors.

Based on this reason, a new transport protocol, namely Virtual-X Transport Protocol (VXTP), was designed to support network transmission of V-X Agent. The main feature

Table 3. Primitives provided by Virtual-X Agent.

Primitive name	Parameters	Description
Open_AP	Application name	Add a new application to the shared window system
Close_AP	Application name	Terminate an application in the shared window system
Get_Floor	Application name	Request the floor of an application
Drop_Floor	Application name	Drop the floor by the application owner
Snatch_Floor	Application name	Snatch the floor by the application initiator

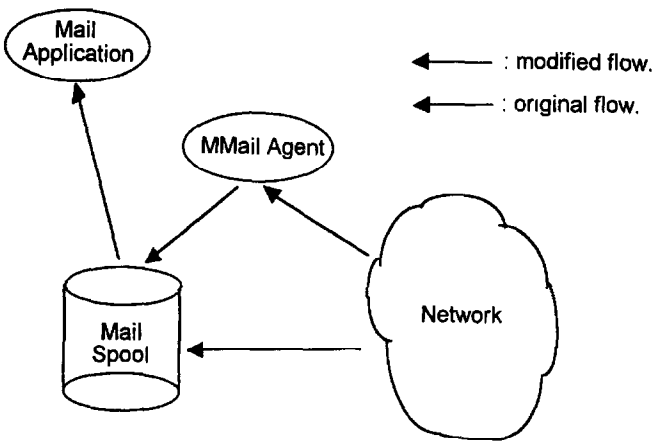


Figure 7. Mail preprocessing flow executed by Multimedia Mail Agent.

of VXTP is to support multicasting functions such that the transmission efficiency can be significantly improved. Details of VXTP protocol are described later in this section.

Important primitives provided by V-X Agent are summarized in Table 3. The last three primitives are related to the floor control operation. The floor is created on a per-application basis. Whenever a shared application user intends to control the application, he/she first sends the request to his/her V-X Agent via the *Get_Floor* primitive, then the agent forwards the request to the V-X Agent of the floor owner. The requester gets the floor only when the floor owner *explicitly* releases it by calling the *Drop_Floor* primitive.

4.1.4. Calendar Agent. Calendar Agent maintains the user schedule, negotiates the time for group activities and automatically activates the events pre-set in the user's schedule. Events in the schedule can be deleted, modified and new events can be added. However, the negotiated events of a group can not be changed unless they are confirmed by all group members. Calendar Agent gets group member information from Session Manager and executes the negotiation process among them using TCP. All events marked as auto-executed will be added to the system CronTable by UNIX *at* command. When the

Table 4. Primitives provided by Calendar Agent.

Primitive name	Parameters	Description
Open_Calendar	UID	Open a user's calendar
Insert_Appointment	Appointment event	Insert an appointment event into the user's calendar
Delete_Appointment	Appointment event	Delete an appointment event from the user's calendar
Edit_Appointment	Appointment event	Edit an appointment event from the user's calendar
Query_Appointment	GID, Time interval	Reserve an empty time slot for an appointment
Close_Calendar	UID	Close the user's calendar

predefined time of an event is met, the system will activate it automatically. Clock synchronization problem is considered in Calendar Agent to adjust the different system time among workstations.

The primitives provided by Calendar Agent are summarized in Table 4. The user can insert, delete and edit appointment events by calling the *Insert_Appointment*, *Delete_Appointment* and *Edit_Appointment* primitives respectively. Furthermore, the user can negotiate the time for group activities by calling the *Query_Appointment* primitive with two parameters, the group ID (GID) and the time.

4.1.5. Voting Agent. The architecture of voting subsystem is shown in figure 8. RSA algorithm [1] for a public key cipher and the Pretty Good Privacy (PGP) [27, 34] algorithm as the encryption/decryption kernel are used in the system. For system and data security, voting application must register itself, via Desktop Management Agent, to Session Manager (SM) with the user password. SM uses PGP to check the password. If the password is legal, SM gives a pass ID to the voting application and both the pass ID and the user password to the Agent. These operations are indicated by marks a, b and c in figure 8. Primitives provided by Voting Agent are shown in Table 5.

After the voting application receives the pass ID, it uses this pass ID to register to the Agent by calling the *Register_To_Agent* primitive. The agent will check the validity of the pass ID and fork a child agent to handle the requests from the voting application. The Agent destroys the pass ID and user password when these works are finished. With these registration operations, the system can avoid unauthorized users and applications to intrude and access the data in voting databases.

After the above procedure, each participant owns a default ballot box to hold all unvoted ballots. As soon as the voting process starts, the Voting Agent of the voting process initiator will create a ballot box to store the ballots sent back by the voting participants by invoking the *Add_Box* primitive. These two boxes are maintained in the voting database to store the voting related data. When the ballot is made, this Voting Agent sends the ballot to all participants by calling the *Send_Vote_To_Group* primitive. The participant can open the default box, read the ballot and make the decision by the *Open_Box* and *Get_Vote* primitives respectively. Then, the ballot will be sent back to the Agent to compute the result by the *Send_Vote_Back* primitive. This voting result will then be sent back to all participants

Table 5. Primitives provided by Voting Agent.

Primitive name	Parameters	Description
Register_To_Agent	PassID	Register to the agent with the PassID
Add_Box	Box index	Create a box to store the ballots
Delete_Box	Box index	Delete the box
Open_Box	Box index	Open the box to read
Number_of_Votes	None	Query the number of unvoted ballots in the default box
Get_Vote	None	Get the unvoted ballot from the default box
Get_Result	Box index	Get the voting result in the box
Get_Vote_Name	None	Get the subject name of the ballot
Send_Vote_To_Group	GID, Vote Content	Send the ballot to a group of people
Send_Vote_To_Person	UID, Vote Content	Send the ballot to a person
Send_Vote_Back	Box index, Vote content	Send the voted ballot back to the box of the voting initator

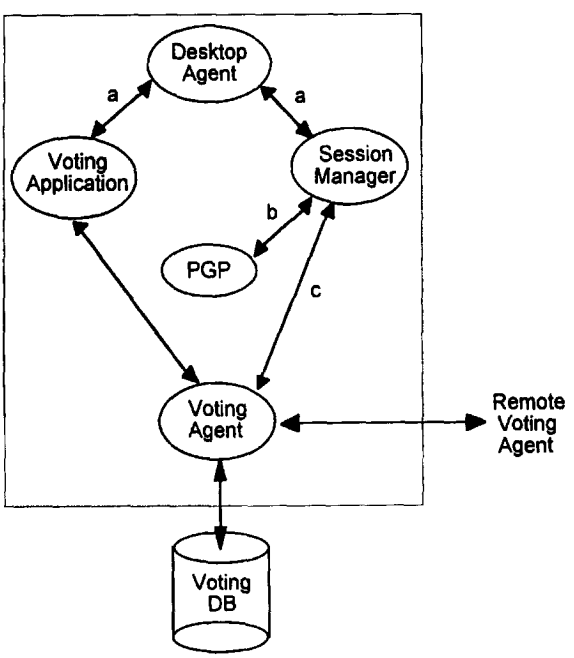


Figure 8. Architecture of voting subsystem.

Table 6. Primitives of Desktop Management Agent.

Primitive name	Parameters	Description
Join_Group	Group ID	Join an existing group after registration
New_Group	New group ID	Join a new group after registration
Register	User ID, password	Register to system when a user logs in
Leave	User ID, Group ID	Notice system when the user leaves

to conclude the voting process. Note that the Voting Agent in the system also supports encryption and decryption of the ballots to further increase the system security.

4.1.6. Desktop Management Agent. The primitives provided by Desktop Management Agent are listed in Table 6.

4.2. Session Manager in System Management Layer

As discussed earlier, SM maintains group lists and the member list of each group. When a user logs into the system, the user ID (UID) and password are checked by SM through Desktop Management Agent. If they are valid, the user could enter the system to join existing groups or to create a new group. Each group has its own group ID (GID) for group activities. When they leave the system, they also have to notify Desktop Management Agent which then informs SM to delete them from these lists. Hence, all agents can query SM to get the most up-to-date lists for use. Current available primitives of SM is shown in Table 7.

4.3. Virtual-X Transport Protocol in multi-protocol network layer

Virtual-X Transport Protocol (VXTP) is designed to support network transmission of Virtual-X Agent. However, because the protocol is not trivial, we will only briefly describe

Table 7. Primitives of Session Manager.

Name	Parameters	Return values	Description
SGL_Query	NONE	Group count,	Query group count and group names
SG_Query	GID	Group size, Group members, Others	Query the size, members and other information of a group
SU_Query	User ID	IP address of the member	Query the IP address of a member
S_Register	User ID, Password, IP address	GID	Register to SM when a user logs in
S_Leave	GID, User ID	NONE	Notice SM when a user leaves

how multicasting is achieved in VXTP here. For other information of this protocol, please refer to [17].

In the X Windows environment, some kinds of X protocol requests could be lost but others must be received without error. To achieve multicasting capability and the error-free transmission for those requests, a connection using connectionless transport protocol is used for multicasting and another connection using connection-oriented transport protocol is used for error recovery. A pre-negotiated port number is used to specify the intended receivers of these X requests in a Virtual-X application. Other than using a connectionless transport protocol for multicasting, the broadcasting function of local area networks, such as FDDI in our environment, is used such that only one copy of each request will be sent. With such a design and other issues proposed in [17], the performance of VXTP is much better than that of TCP as used in XTV. The results will be shown in next section.

5. Usability tests and system comparisons

In this section, we will first present the experiments about the usability of this platform. There are two perspectives to the usability test. One is to present the prototype applications developed on the platform, which shows the correctness and usability of the layering framework design. Another is emphasized on the VXTP protocol, which is focused on the communication issue of the CSCW system to give the geographically dispersed group members a reasonable system response time for their interactive activities. These will lead to the success of the CSCW system in the future. Furthermore, comparisons to some systems are given.

5.1. Usability tests

5.1.1. Prototype applications. From the above discussions for the platform architecture and the implementation details, application development is much easier than that needed in the traditional implementation process. These applications help the users to engage in their common tasks and provide them with a shared environment. By connecting to the agents in CSL through the primitives, applications could easily access the capabilities provided by the platform to fulfill these requirements. In the following, some prototype applications available now are shown in snapshots. The presentation begins with the user interface of this platform, that is, the desktop working environment.

Whenever a user logs into the system, the user will see a user-friendly desktop working environment as shown in figure 9. In this environment, a housekeeper, which is connected to the CSL Desktop Management Agent, managing the working environment is shown as the lady in the figure. Desktop Management Agent then sends a request to Session Manager for user registration. After the registration process, the housekeeper will tell the user how many unread mails are in his/her mailbox and remind the user about the schedule of today.

In this working environment, the user only has to click on an object, which represents an application, to use it. For example, as shown in figure 9, the envelope provides multimedia mail service, the computer monitor provides video conferencing service, the ballot

Figure 9. Desktop working environment of the platform.

provides the voting service, etc. After receiving the click event, the housekeeper forks a child process to execute the application, which then connects to the underlying agent for collaboration.

The video conferencing application allows a user to take part in many conferences simultaneously. Each conference is controlled by its chairman. There are two types of conferences. One is the public conference where the participants do not need any permission from the chairman to join. The other is the private conference where participants are invited by the chairman only. The video window in the conference has a resolution of $360 * 240$ for each participant. Audio data from remote participants are mixed and played back synchronously with the accompanying video data.

As mentioned before, the multimedia mail application can read the Internet e-mails as well as multimedia mails following the MIME standard. The user can see a list of received mails in the mail spool and just need a click to read it. A snapshot of the multimedia mail application is shown in figure 10 in which the mail contains a stream of video and its synchronized audio, an image picture and a text file. The text is shown directly, while the video/audio stream and the image picture are presented as icons. Whenever the user clicks the icon, the associated viewer program for the media would be invoked to display them.

The voting application is important to achieve group consensus for decision making. The voting results can be shown as a bar chart or as a pie chart. The voting application has an

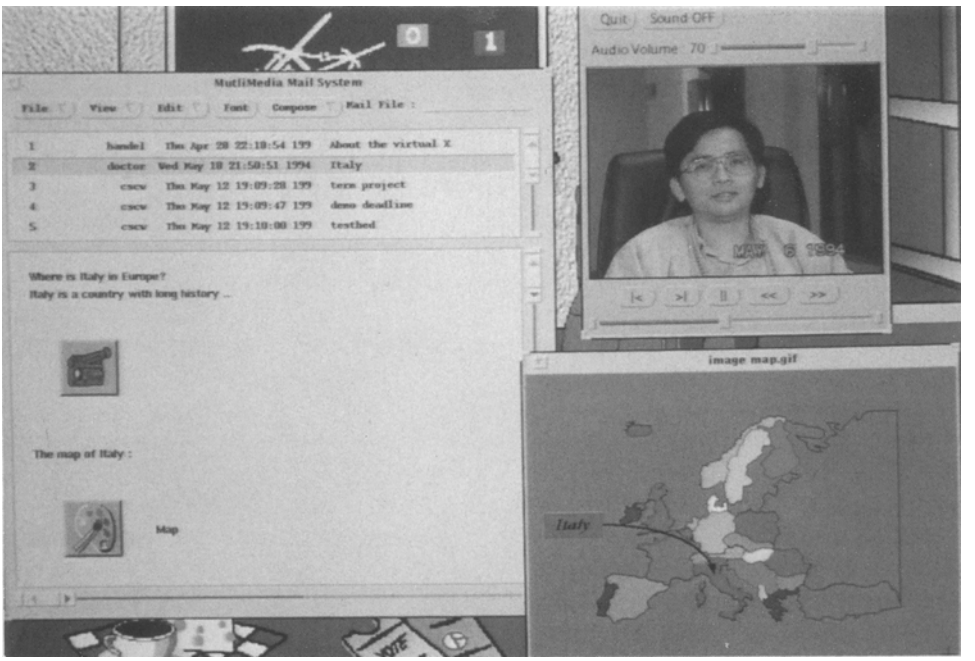


Figure 10. A snapshot of a multimedia mail.

option which allows the ballot maker to choose whether to encrypt the ballot and the result or not. It increases the data security across the network.

A snapshot of these applications is shown in figure 11, in which a two-user video conferencing application together with the Virtual-X, calendar and multimedia mail applications are executed for group collaboration.

5.1.2. Performance measurement of VXTP. In the following, we will describe the performance of VXTP. Five workstations are connected through an FDDI network. Three applications are used as representatives of the commonly used applications, which are *xterm, textedit* and *xv*. We will measure the time from the command is issued until the display of the application is completed. The number of X-protocol requests generated in the operations for these three applications are 72, 352 and 1608, respectively, and the program sizes of them are 3696, 16841 and 324800 bytes. They represent the applications which belong to three different scales in terms of the number of X-requests generated.

Each request sent by the X client is processed by V-X Agent and then multicasted to the destination X servers. The flow of request handling is shown in figure 12. We will measure the time between b and c, that is, the multicasting time, and the time between a and c, that is, the total processing time of V-X agent. In the following, we will compare its performance to the ones using TCP point-to-point connections, as in XTV.

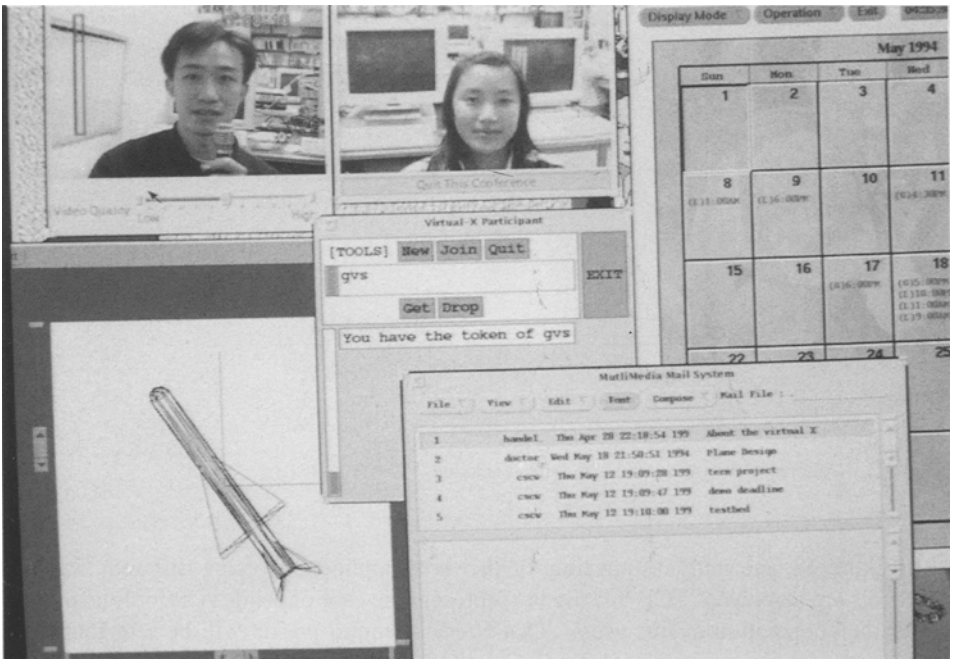


Figure 11. The snapshot of the applications built on the platform.

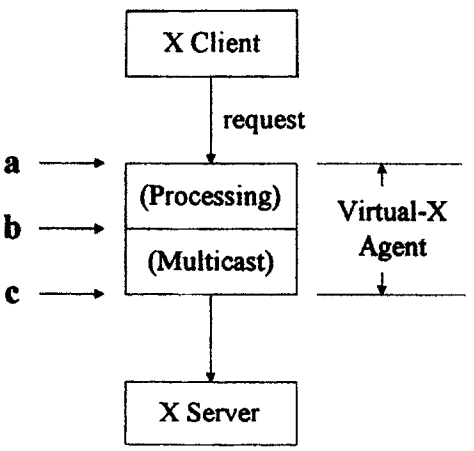


Figure 12. The flow of request handling through V-X Agent.

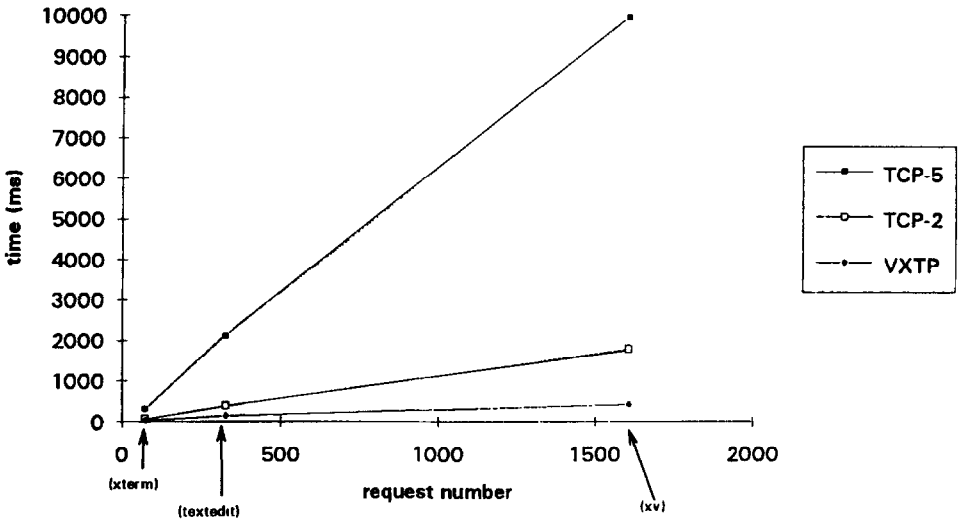


Figure 13. Amount of time spent on multicasting operation.

In figure 13, the multicasting time of the three applications using different transport protocols are displayed. TCP-5 is used to represent the case of using TCP for transmission under the cooperation of five users. That is, each request packet will be sent four times to all participants except the local one. Similarly, TCP-2 uses TCP for the case of two users. That is, each request is sent only once. From this figure, the VXTP performance is much better than other cases because the simplicity and multicasting capability of the protocol.

In figure 14, we compare the total processing time of the system under these three cases. It is also shown that the time spent using VXTP is much smaller than the other two, due to its multicasting capability. From the curves of TCP-2 and TCP-5, we know that TCP is not proper for multicasting transmissions. On the other hand, VXTP achieves excellent performances and helps the group communication in CSCW.

Note that VXTP also outperforms TCP-2 for the following two reasons.

(1) As mentioned in Subsection 4.3, VXTP is a light-weight connectionless transport protocol for multicasting transmissions where TCP is a connection-oriented protocol. Hence, the state monitoring and connection management can be saved in VXTP.
(2) In the X Window environment, some X requests can be lost without hurting the system. VXTP takes advantage of this feature to save many retransmission overheads (compared to TCP) when the network traffic is heavy.

However, if an error happens to other X requests requiring reliability, a retransmission will be triggered over the connection-oriented connection as described in Subsection 4.3. In this case, the performance of VXTP will be similar to that of TCP.

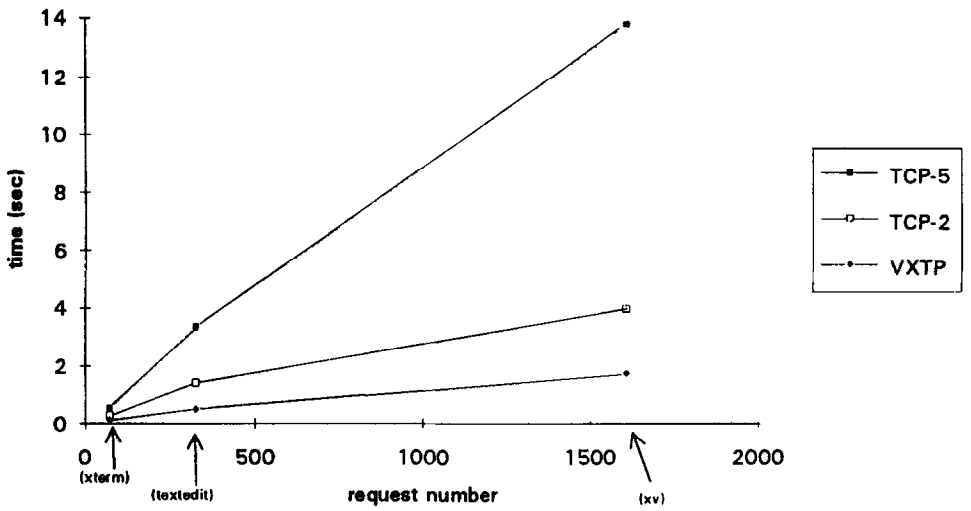


Figure 14. Total processing time spent in V-X Agent.

5.2. *Comparisons with other systems*

In the following, four systems are briefly described and compared to our system.

BERKOM [2]: It is a Broadband ISDN trial project, which concentrates on providing a single uniform communication infrastructure for future broadband multimedia applications. There are three areas of work. (1) BERKOM MultiMedia Transport service (MMT): it provides the communication platform for audiovisual communication. It is based on STII, the Internet Stream Protocol, which supports multi-endpoint connections with guaranteed throughput and delay. (2) BERKOM MultiMedia Mail service (MMM): it facilitates the exchange of multimedia documents. It is conceived as a standard-conformant extension to the X.400 and ODA functionality and provides interworking with MIME. (3) BERKOM MultiMedia Collaboration service (MMC): it supports joint working in a distributed environment. It allows users to share applications and to participate in audiovisual conferences from their workstations.

SHASTRA [4]: SHASTRA strengthens collaboration in scientific and engineering design by providing an infrastructure for user- and application-level cooperation. Tools are provided for the creation, manipulation and visualization of multi-dimensional geometric data. The collaboration is supported by the collaboration substrate that supports synchronous multi-user applications. The connection and communication distribution substrate emphasizes the distributed problem solving. These substrates are function libraries with well-defined abstract programming interfaces that establish a framework for session management, data sharing and multimedia communication. It uses a replicated computation model for multi-user system, that is, a copy of the application runs at each site involved in the collaboration. The session manager

provides the multicast facility for information exchange in synchronous multi-user conferencing.

CECED [7]: It provides a collaboration support environment that connects a designer to appropriate knowledge sources easily. This environment has three key areas: workspace support, conversational support and process history capture. Media are integrated in the same multicast connection, for example, the audio/video communication and content-independent sharing of drawing and viewing surface. It also supports the replication of applications and databases of each site, quick feedback to all conferees, even with low network bandwidth.

Argo [10]: It allows medium-sized groups of users to collaborate from desktops by combining multi-party video compressed with the JPEG standard and full-duplex audio with telepointers, shared applications and whiteboard in a uniform environment. It also provides an object-oriented, client/server conference control system to support teleporting, i.e., moving the desktop environment from one workstation's display to another. Three kinds of sharing are supported, i.e., the window-system-based ShX, the window-system-independent Trestle object-oriented toolkit, and the groupware, for example, the AV applications, approaches. Each approach is supported by a sharing agent. The conference control server also maintains objects representing users, conferences and members.

From the above descriptions, the CECED and SHASTRA systems are designed for special application domains, the Argo system is a general collaboration supporting system. The BERKOM project concentrates on providing a broadband multimedia communication infrastructure which helps the collaboration and conforms to the international standards.

Our work is focused on the design of a layering model for group collaboration, and furthermore, the deployment of a generic CSCW platform in which critical functionalities are supported. The most important features are that with clear separation of functionalities, from network protocols to user interfaces, in the four-layer model and the well-implemented platform, special-purpose applications can be easily developed on the platform, which leaves the application programmers isolated from all implementation details. As a result, the realization of any CSCW system can be accomplished in a cost and time effective way.

Besides the most common collaborative supports in these related systems, such as audio/video conferencing, application-sharing, conference control, and session management, our platform also provides group members with a group decision support system, i.e., the voting facility, to achieve group agreement and the calendar facility to maintain the user's schedule. Furthermore, the multimedia mail facility is included in the platform to enhance the communication service among group members, i.e., not only the synchronous but also the asynchronous communication services.

6. Conclusions

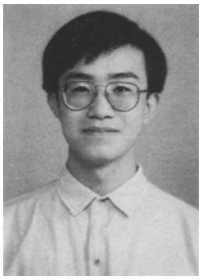
Since group cooperation is becoming more and more common in human society and the complexity of projects is getting higher and higher, the importance of a good CSCW system

is without doubt. In this paper, a layering model for CSCW systems is proposed. Based on this model, a multimedia platform for CSCW environment is presented. The multimedia capability supported by the platform includes audio, video, image and text. Further, on-line interactive multimedia applications such as video conferencing are supported. Other than the multimedia capability, several agents are provided in the platform to support functions of voting, calendar, multimedia mail, and shared X applications. Last but never least, a new protocol called VXTP is supported to provide reliable multicast transmissions to enhance the network efficiency. With such a flexible, efficient and friendly platform, users can develop their CSCW systems with specific requirements easily.

References

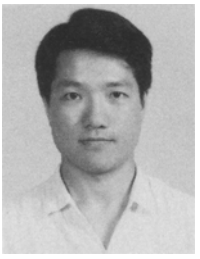
1. H. Abdel-Wahab and M. Feit, "XTV: A Frame-Work for Sharing Xwindow Clients in Remote Synchronous Collaboration," Proc. of IEEE Tricomm: Communications for Distributed Applications & Systems, Chapel Hill, pp. 159–167, Apr. 1991.
2. M. Altenhofer, J. Dittrich, R. Hammerschmidt, T. Kappner, C. Kruschel, A. Kuckes, and T. Steinig, "The BERKOM Multimedia Collaboration Service," Proc. of ACM Multimedia, pp. 457–463, 1993.
3. V. Anupam and C. Bajaj, "Collaborative Multimedia Scientific Design in SHASTRA," Proc. of ACM Multimedia, pp. 447–456, 1993.
4. V. Anupam and C. Bajaj, "SHASTRA: Multimedia Collaborative Design Environment," IEEE Multimedia Mag., pp. 39–49, Summer 1994.
5. J. Baldeschwieler, T. Gute Kunst, and B. Plattner, "A Survey of X Protocol Multiplexors," ACM SIGCOMM Computer Communication Review, Vol. 23, No. 2, pp. 16–24, 1993.
6. U. Borghoff and G. Teege, "Application of Collaborative Editing to Software-Engineering Projects," ACM SIGSOFT, Software Engineering Notes, Vol. 18, No. 3, pp. A-56–A-64, 1993.
7. E. Craighill, R. Lang, M. Fong, and K. Skinner "CECED: A System for Informal Multimedia Collaboration," Proc. of ACM Multimedia, pp. 437–443, 1993.
8. M. Cutkosky, R. Engelmore, R. Fikes, M. Genesereth, T. Fruber, W. Mark, J. Tenenbanm, and J. Weber "PACT: An Experiment in Integrating Concurrent Engineering Systems," IEEE Computer Mag., pp. 28–37, 1993.
9. C.A. Ellis, S.J. Gibbs, and G.L. Rein, "Groupware: Some Issues and Experiences," CACM, Vol. 34, No. 1, pp. 39–58, 1991.
10. H. Gajewska, J. Kistler, M.S. Manasse, and D. Redell, "Argo: A System for Distributed Collaboration," Proc. of ACM Multimedia, pp. 433–440, 1994.
11. J.H. Huang and S.H. Lee, "MHTP—A Multimedia High-speed Transfer Protocol," Proc. IEEE GLOBECOM, pp. 1363–1368, 1992.
12. J. Huang, C. Yang, W. Tseng, C. Lee, B. Tsaur, L. Chuang, and W. Liu, "Design and Implementation of Multimedia Conference System on Broadcast Networks," Proc. of 18th Annual Local Computer Network Conference, Minneapolis, pp. 337–341, 1993.
13. H. Huang, J. Huang, and J. Wu, "Real-Time Software-Based Video Coder for Multimedia Communication Systems," ACM Multimedia Systems, Vol. 1, pp. 110–119, 1993.
14. T. Kirsche, R. Lenz, H. Luhrsen, K. Meyer-Wegener, H. Wedekind, M. Bever, U. Schatter, and C. Schottmuller, "Communication Support for Cooperative Work," Computer Communications, Vol. 16, No. 9, pp. 594–602, 1993.
15. D. Kohlert, K. Rodham, and D. Olsen, "Implementing a Graphical Multi-User Interface Toolkit," Software-Practice and Experience, Vol. 23, pp. 981–999, 1993.
16. K. Lai, T. Malone, and K. Yu, "Object Lens: A "Spreadsheet" for Cooperative Work," ACM Trans. on Office Information Systems, Vol. 6, No. 4, pp. 332–353, 1988.
17. C. Lee, "The Design and Implementation of Virtual-X system in CSCW," Master Thesis, Dept. of Computer Science & Information Engineering, National Taiwan University, Taiwan, June 1994.

18. T. Liang, H. Lai, N. Chen, H. Wei, and M. Chen, "When Client/Server Isn't Enough: Coordinating Multiple Distributed Tasks," IEEE Computer Mag., pp. 73–79, 1994.
19. L. Liang, G. Neuteld, and S. Chanson, "A Name Model for Nested Group Communication," IEEE/ACM Trans. on Networking, Vol. 1, No. 4, 1993.
20. K. Maeno, S. Sakata, and T. Ohmori, "Distributed Desktop Conferencing System (MERMAID) based on Group Communication Architecture," Proc. of ICC, pp. 520–525, 1991.
21. S. Masaki, H. Yamaguchi, Y. Hayshi, T. Nishimura, and K. Shimamura, "Multimedia Handling Scheme in a Grouphware System for B-ISDN," Proc. of IEEE Globecom, pp. 747–751, 1992.
22. S. Minneman and S. Harrison, "Where Were We: Making and Using Near-Synchronous, Pre-Narrative Video," Proc. of ACM Multimedia, pp. 207–214, 1993.
23. M. Ouhyoung, W. Chen, Y. Lei, K. Chang, C. Liang, S. Wang, Y. Yan, J. Wu, H. Chen, N. Liu, Y. Wang, T. Hwu, W. Su, R. Liang, K. Fu, Y. Chen, and T. Yang, "The MOS Multimedia E-Mail System," Proc. of IEEE Multimedia Computing and Systems, pp. 315–324, 1994.
24. M. Pendergast, "Multicast Channels for Collaborative Applications: Design and Performance Evaluation," ACM SIGCOMM Computer Communication Review, Vol. 23, No. 2, pp. 25–37, 1993.
25. P. Pewan and J. Riedl, "Toward Computer-Supported Concurrent Software Engineering," IEEE Computer Mag., pp. 17–26, 1993.
26. J. Plalmer and N. Fields, "Computer-Supported Cooperative Work," Guest Editors' Introduction, IEEE Computer Mag., pp. 15–17, May 1994.
27. R. Rivest, A. Shamir, and L. Adelman, "A Method of Obtaining Digital Signatures and Public Key Cryptosystems," CACM, Vol. 21, No. 2, pp. 120–126, 1978.
28. R.M. Sanders and A.C. Weaver, "The Xpress Transfer Protocol (XTP)—A Tutorial," A Quarterly Publication of ACM SIGCOM, Vol. 20, No. 5, pp. 67–80, 1990.
29. R. Simon, R. Scalabassi, and T. Znati, "Communication Control in Computer Supported Cooperative Work Systems," Proc. of CSCW, pp. 311–321, 1994.
30. M. Stefik, G. Foster, D. Bobrow, K. Kahn, S. Lanning, and L. Suchman, "Beyond the Chalkboard : Computer Support for Collaboration and Problem Solving in Meetings," CACM, Vol. 30, No. 1, pp. 32–47, 1987.
31. A. Tanenbaum, Modern Operating Systems, Prentice-Hall International: New Jersey, pp. 452–459, 1992.
32. I. Tou, S. Berson, G. Estrin, Y. Eterovic, and E. Wu, "Synchronous Group Applications," IEEE Computer Mag., pp. 48–56, 1994.
33. P. Wallich, Electronic Envelopes, Scientific American, pp. 30–32, 1993.
34. P. Zimmermann, "A Proposed Standard Format for RSA Cryptosystems," Advances in Computer Security, Vol. 3, Artech House, 1988.



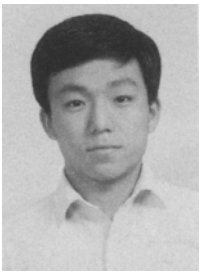
Ing-Chau Chang received his B.S. degree in Computer and Information Science from National Chiao-Tung University, Hsin-Chu, Taiwan, in 1990 and his M.S. degree in Computer Science and Information Engineering from National Taiwan University, Taipei, Taiwan in 1992. He has been a Ph.D. student in Computer Science and Information Engineering at National Taiwan University, Taipei, Taiwan.

His current research topics include computer supported cooperative work, distance learning, multimedia network protocols, and multimedia systems.



Jau-Hsiung Huang received the B.S. degree in Electrical Engineering from National Taiwan University, Taiwan, in 1981 and the M.S. and Ph.D. degrees in Computer Science from the University of California, Los Angeles, CA, U.S.A., in 1985 and 1988, respectively.

Since 1988, he has been a member of the faculty in the Department of Computer Science and Information Engineering, National Taiwan University, where he is currently a professor. He co-founded the Communications and Multimedia Laboratory in the department in 1990 to launch a series of researches in the areas of distributed multimedia systems and virtual reality. He has published over 40 technical papers in the areas of multimedia networking, high-speed networking, parallel and distributed systems and performance evaluation of computing systems.



Wei-Hsin Tseng received his B.S. degree in Computer Science and Information Engineering from Tatung Institute of Technology, Taipei, Taiwan, in 1989 and M.S. degree in Computer Science and Information Engineering from National Taiwan University, Taipei, Taiwan in 1991. He has been a Ph.D. student in the department of Computer Science and Information Engineering at National Taiwan University, Taipei, Taiwan.

His current research topics include high speed networking, video modeling, admission control, and video server architecture.