Vol. 37 No. 3 Transactions of Information Processing Society of Japan Mar. 1996

Regular Paper

Networking Architecture for Multimedia Services on a Distributed Processing Environment

CHOONG SEON HONG,† SHINKURO HONDA† and YUTAKA MATSUSHITA†

In this paper, we present and discuss a networking architecture model that is built on a Distributed Processing Environment (DPE) for multimedia services suitable for high speed networks such as B-ISDN, for the flexible and rapid introduction of services. In this model, the applications are deployed as units of software building blocks. Each building block provides a layered view for the effective management and control of the multimedia network resources and services according to the concept of TMN and TINA. For the purpose of flexible service provision to users and effective service introduction by service providers, this architecture proposes the adoption of ad hoc service building blocks such as a video on demand building block and a CSCW building block. This paper also proposes a naming structure for the management of user profiles and session profiles using a directory service system, and an effective control model for multimedia logical device objects using a stream process approach. The proposed model is implemented on a DPE platform that provides various transparencies, ANSAware.

1. Introduction

Users' needs for new multimedia services are rapidly increasing. Many multimedia service systems[1)~4)] have been proposed. But those systems are for current network environments. The concept of current network environments is supposed not to be rapidly copying with users' requirements due to too much dependence on network elements (e.g., ATM switch) to introduce new services. So there is currently considerable interest in developing an architecture to flexibly interconnect multimedia applications. Major research in this area includes Telecommunications Information Networking Architecture (TINA)[5)], Information Networking Architecture (INA)[6)], and so on. These networking architectures provide solutions for critical business problems by enhancing the environment of the existing networks to handle multimedia communication and multimedia information, resulting in the reduction of delays and costs in the introduction of networking services. But, since the contents of these deliverable documents[11),13),19),20)] on the introduction of new services are not complete yet, we present here some extensions. For example, TINA does not deal with the cases in which computational objects (in this paper, we call a computational object a "building block") for special services have some amounts of information related to services and interactively use that information. TINA also does not describe interactions between two session managers (in this paper, it corresponds to group call building block and ad hoc service building block) that have different session characteristics (e.g., one for video conferencing and the other for computer supported cooperative work). Therefore, we especially propose the adoption of special purpose computational objects, ad hoc service building blocks. These building blocks have open interfaces with general purpose building blocks to provide potential connectivity and to enable an environment to be capable of flexible multimedia service provision and the rapid introduction of new multimedia services. This modeling architecture also has the objective of promoting the modularity and portability of applications.

In this architecture, the service, connection and resource management functions specified by TMN[7),18)] and TINA[9),11),13)] are achieved by the cooperation of logically distinct building blocks and servers. If we classify and assign those building blocks according to their roles, the group call building block, the connection control building block and transport networks belong to service/network providers, and terminals including the group call agent building block which plays a managing role of Video On Demand (VOD) server, for instance, are oper-

† Department of Instrumentation Engineering, Faculty of Science and Technology, Keio University

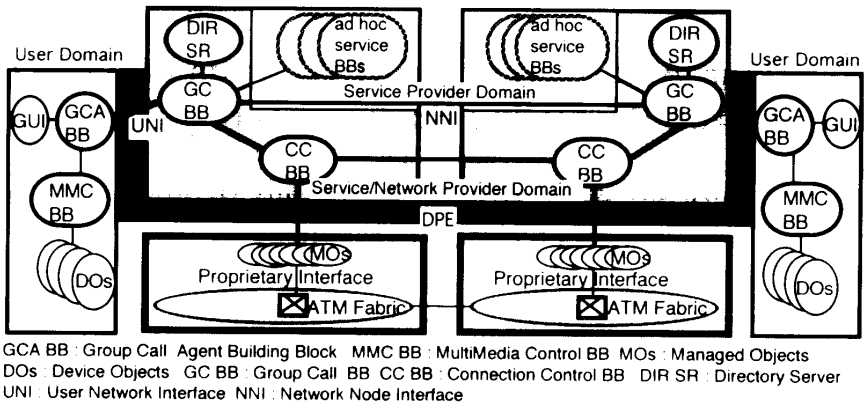

Fig. 1 Configuration of networking architecture.

ated by information providers. Ad hoc service building blocks belong to the service providers' domain.

We implemented this architecture on a Distributed Processing Environment (DPE), which provides a uniform execution environment for distributed applications in order to take advantage of distribution transparencies defined in the Reference Model for Open Distributed Processing (RM-ODP)[12].

In the remainder of this paper, Section 2 addresses the overall networking architecture we propose. Following this, Sections 3 and 4 explain how to manage the user and session profiles and discuss signal interaction between building blocks, respectively. A *QoS* negotiation mechanism is discussed in Section 5. Section 6 describes a scenario for introducing a new service. A multimedia terminal architecture is discussed in Section 7, and realization and evaluation are contained in Section 8.

2. Networking Architecture Overview on DPE

The main purpose of the design of the networking architecture on DPE is to allow the support of applications on a heterogeneous collections of systems, permitting these systems to be arbitrarily distributed. The networking architecture for multimedia services we propose is composed of several functional building blocks on a DPE in order to support the flexible service provision based on a high speed transport network. Our architecture as a TINA-like extension model can more flexibly accommodate new services by adopting ad hoc service building blocks which service providers prepare according to particular service requirements. The simple view of our system is shown in **Fig. 1**, describing the interaction between network components (building blocks, servers, network elements, etc.).

A group call building block plays an activation role for a service. It links the users of the service together so that they can interact with each other and share service entities. This provides operations that allow users to join and leave a service session. For certain services it may also offer operations to suspend and resume involvement in a service. Connection management, which is performed by a connection control building block, is a service-oriented abstraction of connections in the transport network. A connection control building block maintains the state of the connections of a particular service session, such as communication paths, end-points and quality of service (*QoS*) characteristics. The purpose of this concept is to separate out different concerns and to promote distribution of functionality. The separation of call and connection sessions supports the distinguishing of activities of the call session from the set of connections that exist. There are two essential reasons for this split. Firstly, not all services supported by a call session will require the use of the transport network. In these cases, services will be provided by the group call building block. Secondly, even if a service establishes connections on the transport network, other activities may be taking place, and the users may be involved; there is not necessarily a one-to-one correspondence between those that are involved in a service and those that have transport connections as part of the service.

2.1 Service Session Management

A group call agent building block (GCA BB) which is responsible for user's terminal service

session provides a customized view to a user. A GCA BB coordinates the activities of a graphical user interface (GUI), a multimedia control building block (MMC BB) and a group call building block (GC BB). This building block receives abstract requests from a GUI or a GC BB. Then, to realize the requests, the GCA BB issues control messages in a format that a GC BB or a GUI can understand, and sends those messages to the GC BB or the GUI. For the purpose of reserving the local terminal resources, the GCA BB issues a request to an MMC BB before it sends a message to the GC BB. The GCA BB manages the terminal resources and performs *QoS* negotiations for the connection of each medium with the network. The GCA BB can notify to a user through the GUI if other processes want to use terminal resources that the user is using. Some message types for multimedia conferencing are: Call_request, Add_member, Join_member, Add_service, Delete_service, Delete_member, Close_call and Drop_member.

The GC BB which manages a service session for the network side receives and analyses user requirements (adding or deleting an endpoint, and requesting retrieval, etc.) received through the GCA BB, manages the user profiles, and represents states of the present call (or session profile). User and call states include the participants of the current session, the session name, call ownership, call initiating time, *QoS* level for each medium and the bandwidth of the user's contract. The results of analysis based on the messages invoked by the originating endpoint are passed to the Connection Control Building Block for establishing a communication connection. If the terminating endpoints are not in the network managed by the local GC BB, the originating GC BB sends the user's requirements to the terminating or intermediate GC BBs corresponding to other endpoints. If a GC BB receives a user's requirement for a special purpose service, it passes the user's message to an ad hoc service building block. A GC BB corresponds to a user agent, subscription manager and service session manager for normal calls (e.g., phone call, video conferencing call) in the TINA context[11].

2.2 Connection Control and Management

The connection control building block (CC BB), which manages connections establishes, modifies and releases the connections requested through a GC BB. A CC BB enables the transfer of the logical address to the physical address. Also, this manager manages the information about the network's topology. Using the topology information, a CC BB provides the optimal routing as well. For the purpose of effective resource control, this manager adopts an object oriented approach. In the TINA context, this manager plays the role of communication session manager, connection coordinator and connection performer.

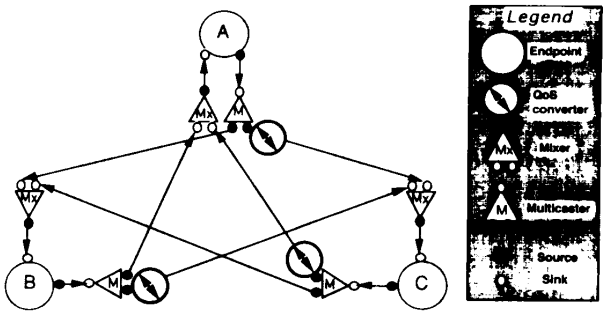

Fig. 2 Logical connection graph for multimedia resource control.

Multimedia resource control model: In order to help flexibly generate new objects and effectively manage the connection, and effectively meet users' requirements, we present a multimedia resource control model. This scheme prepares several controlling objects so that the managed objects (MOs) can handle real resources in the ATM fabric. The managed objects that control network elements are indirectly controlled through management proxy for these objects. Those objects are described in **Fig. 2** by using a logical connection graph. The objects we defined for multimedia connection are multicaster, mixer, link, *QoS* converter, endpoint, and so forth. As an example, if a CC BB is requested to establish a three-party audio connection from a GC BB, the CC BB first creates the objects, then connects endpoints and the mixers/multicasters. Finally, the mixers and multicasters are connected to each other. Accordingly, using this scheme, the request of multipoint-to-multipoint connections for each service type (e.g., audio, video, message, delivery service) from a GC BB can be flexibly constructed by controlling the logical objects. Especially, if it is necessary to supply the different *QoS*s copying with the request of a user and the terminal capability, this scheme can use the *QoS* converter.

2.3 Ad Hoc Service Management

In this paper, we propose a type of interactions between two different session managers. Ad hoc service BB is one of session managers. It has standard interfaces to a group call building block as a general purpose session manager. An ad hoc service BB is used to provide special purpose services such as VOD, home shopping, computer supported cooperative work (CSCW) to users. This building block should be provided by a service provider. Therefore, the service/network provider that manages general purpose building blocks has to prepare a standard interface between the GC BB and ad hoc service BBs to promote the development of special purpose services. Ad hoc service BBs can be categorized into two classes: the dedicated class which is maintained during the entire session life time (e.g., for CSCW) and the semi-dedicated class which is just maintained while selecting a service (e.g., for VOD, home shopping). Each ad hoc service BB manages the special purpose service subscriber's information, the contents of service and its domain to support the user's selection service, the cooperative work domain, and so forth. Additionally, the signal flows for explaining interactions between ad hoc service BB and GC BB is described and depicted in Section 4 and Fig. 5, respectively.

3. The Management of Information for User and Session Profiles

In our architecture, we use a Distributed Directory System (DDS)[14], ITU-T Rec. X.500, for managing user and session profiles. This system provides an open environment to access the DIB because it is standardized by ITU-T. Therefore, users' information can be shared between different carriers if they have contracts with each other. Below, we briefly discuss the directory system and apply it to our architecture, then propose a naming structure for managing multimedia sessions.

3.1 Directory Service System

Primarily, the X.500 directory provides a mapping facility from a user-oriented name to a machine-oriented name which preserves the information of the objects in the real world. The user-oriented name can be mapped to several characteristic objects which have meaning in control and management in the virtual and real worlds. Namely, given the name for an object, we can easily retrieve and modify the attributes of the object, and create or delete subordinate objects and their attributes.

In this paper, we present an example that elaborates our proposed networking architecture. The detailed description for naming objects will be the next sub-section. The DDS is composed of Directory User Agents (DUAs) and Directory System Agents (DSAs), and includes the Directory Information Base (DIB).

- DUA: Each user is represented in accessing the DUA, which is considered to be an application process. DUAs may also provide a range of local facilities to assist users' queries and interpret the responses.
- DSA: This is an OSI application process belonging to the directory. It provides access to the DIB (described below) for DUAs and/or other DSAs. The directory can be distributed over an arbitrary number of DSAs. Each DSA is responsible for a subset of entries in the DIT (Directory Information Tree). This fragment of the DIT is called the naming context of that DSA. A DSA may use information stored in its local database or interact with other DSAs to carry out requests.
- DIB: The contents of a DIB describe both static and dynamic aspects from the perspective of network management and application work.

In Fig. 1, the directory servers (DIR SRs) are interconnected in the intracarrier and intercarrier domain to allow access by a DUA to information that is being managed by remote DSAs.

3.2 Naming Structure of Entries

In our proposal, a DDS is used for constructing the user and session profiles used in a phone call, a conferencing call and the other services. As a client for a DSA, the DUA requests necessary information for a DSA. The DSA holds static and dynamic information. The static information involves those objects whose contents are rarely changed during an operation. Examples of such information are objects related to some addresses and objects' attributes (e.g., user name). The information that is related to an address is called the "user profile". This profile is created when a user subscribes to use a network. On the other hand, objects that change very frequently are dynamic information. Examples are call identification and state in use of a network resource. This information is called a "session profile". This profile is created during a session and deleted with the close of the session. We constructed the DIT in or-

der to make the two profiles that are described above. The DIT structure is shown in **Fig. 3**. ADD_CL, ADD_SUB_CL and User_address entries are static information not to be changed during a session operation. Those entries are components that construct a user profile and are composed according to the ITU-T numbering plan (E.164). The Call_id, a subordinate entry of User_id and the Medium_id entry compose a session profile. In the figure, a user has multiple call identities which have only meaning in one user address.

Each call identity is composed of several media (audio, video, data, etc.) and a medium has multiple users' information. The definition of information to be managed by media identities (Medium_id) is summarized as follows:

$$\begin{aligned}
\text{Medium_type}1 &= \text{Member}1^{\text{in_QoS}}_{\text{out_QoS}} \\
&+\text{Member}2^{\text{in_QoS}}_{\text{out_QoS}} + \cdots + \text{Member}n^{\text{in_QoS}}_{\text{out_QoS}} \\
\text{Medium_type}2 &= \text{Member}1^{\text{in_QoS}}_{\text{out_QoS}} \\
&+\text{Member}2^{\text{in_QoS}}_{\text{out_QoS}} + \cdots + \text{Member}n^{\text{in_QoS}}_{\text{out_QoS}} \\
&\vdots \\
\text{Medium_type}n &= \text{Member}1^{\text{in_QoS}}_{\text{out_QoS}} \\
&+\text{Member}2^{\text{in_QoS}}_{\text{out_QoS}} + \cdots + \text{Member}n^{\text{in_QoS}}_{\text{out_QoS}}
\end{aligned} \tag{1}$$

An in_QoS and an out_QoS mean a user's incoming and outgoing *QoS* levels, respectively. In case of delivery on demand service, the delivery server's in_QoS value would be 'none' because a video server has outgoing channels to provide delivery service. The n in Medium_type$n$ and Member$n$ are determined by a user's terminal capability and a network provider's policy, respectively. So, the above definition can deal with both symmetric and asymmetric characteristics. The CC BB uses this information for establishing the connections.

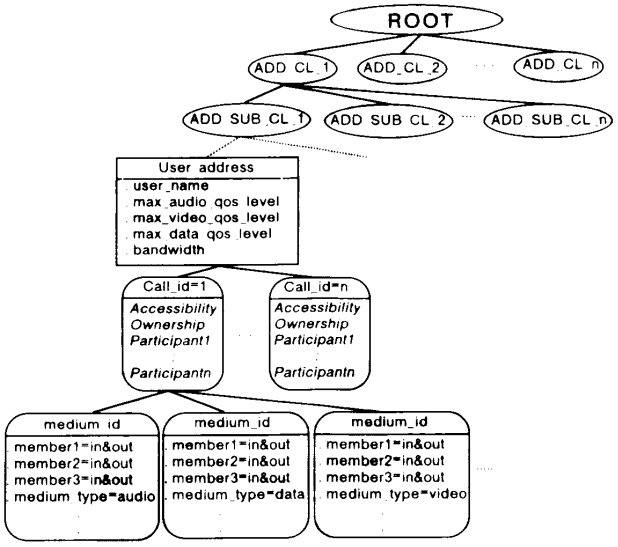

Fig. 3 Structure of DIT for directory system.

4. Signal Interaction between Building Blocks

In this section, we discuss the interaction of signal messages. The signaling protocol which is adopted in our architecture is shown in **Fig. 4**. This signaling procedure does not include that for using ad hoc service BBs. The signaling flow related to ad hoc service BBs is described in **Fig. 5**. In this figure, the signal messages designated by bold italic type are added for ad hoc services. The GC BB differentiates whether a required call is a phone call, normal conferencing call or a call for ad hoc services. In those two figures, o and t stand for the originating and the terminating building block, respectively.

In our architecture, *QoS* negotiation is carried out in 2 phases. The first phase is achieved by negotiation between GC BBs based on users'

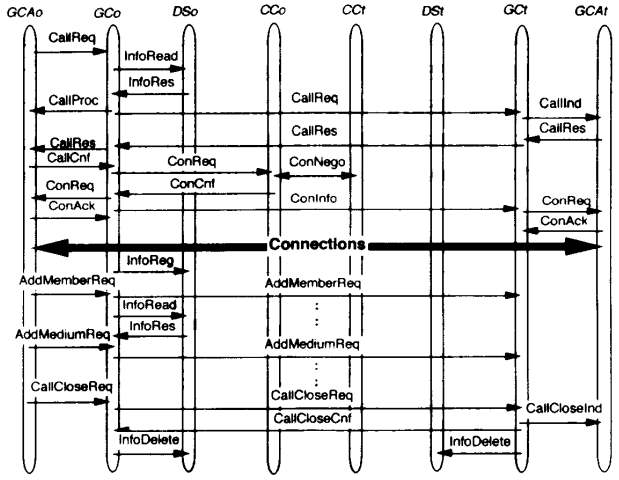

Fig. 4 Signal flow for conferencing call of proposed architecture.

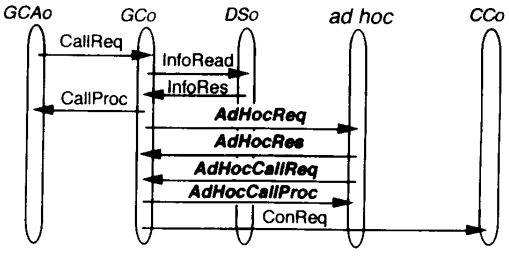

Fig. 5 Signals added for ad hoc service.

contracts and terminal capabilities, the second one is performed between CC BBs for the purpose of checking the availability of network resources. The detailed description of the *QoS* negotiation mechanism is discussed in next section. Additionally, a mapping of our protocol on DPE to B-ISDN signaling of ITU-T requires further study.

5. *QoS* Negotiation Mechanism

There are some recent studies[21),22)] of *QoS* negotiation mechanisms. But those are not considered service session level *QoS* negotiation mechanisms. We realized a novel *QoS* negotiation mechanism that considers service session level based on the networking architecture proposed in this paper, as shown in **Fig. 6**.

The negotiation is accomplished in two domains, user and network domains. In the user domain, a user initiates a call in idle state by setting up users' and members' *QoS* requirements using *QoS* level selection buttons as shown in Fig. 10. Then, the local terminal checks *QoS*s of resources to learn whether the *QoS* for each medium is available or not in its terminal. If those are available, it reserves resources corresponding to required *QoS*s. The network domain has two levels, service session and connection, to perform a *QoS* negotiation. The service session level that is managed by GC BBs checks users' subscriptions for use of the network resources and services. If negotiation between all participants including caller is successful, the negotiation is moved to the connection level. At this level, the mechanism negotiates *QoS*s between global connection managers (i.e., CC BBs), and reserves network resources for connections. Then, it translates the customized *QoS* to the presentation of transport network's *QoS*. One of the most important features of this *QoS* negotiation mechanism is that can save network resources by adopting two levels of *QoS* negotiation. For example, if a session level *QoS* negotiation fails, it does not progress toward the connection level *QoS* negotiation to reserve network resources for establishing connections.

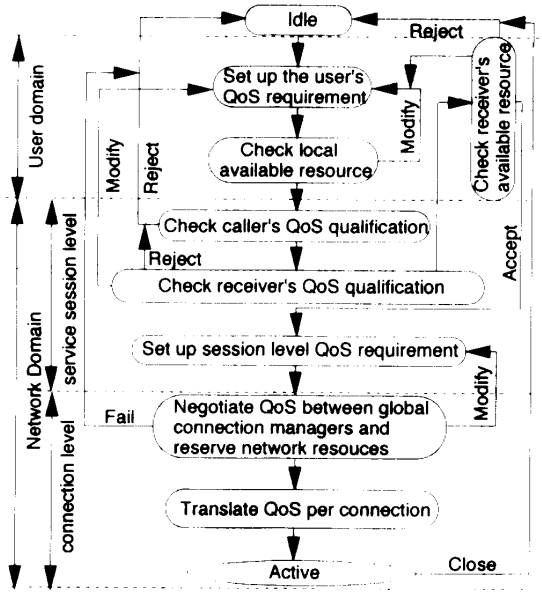

Fig. 6 *QoS* negotiation management mechanism.

6. Scenario for Introduction of New Services

We describe an ad hoc service BB which plays a role as VOD service agent. This building block is called VOD BB in the remainder of this paper. As an agent, a VOD BB has the functionality to provide, to users for whom it has the subscription profile, contents available from the information providers or other VOD BBs, the agents' characteristic information, and so on. The VOD BB is simply added by adopting interfaces defined between a GC BB and an ad hoc service BB as described in Section 2. The control and management of the video delivery server is not discussed in this paper. Using the proposed architecture, a service by VOD BB is introduced as follows: We assume that the local GC BB can directly access the VOD BB without passing to another GC BB and every BB uses a trader for importing the reference interface that is needed to access another BB. The configuration of the scenario is illustrated in **Fig. 7**.

① A GCA BB requests to the network (GC BB) to receive VOD agent services, for example, video selection service.

② The GC BB sends a request message to a VOD BB that provides the decision-making service for which video to choose.

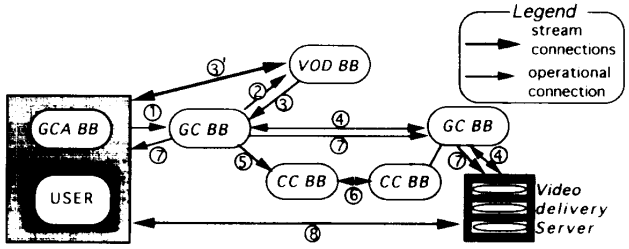

Fig. 7 Scenario for receiving a video delivery service.

③ After a VOD BB checks accessibility to use its service, it sends a message to the GC BB for setting up connections that support the interaction between the user and the VOD BB, and the the GC BB sends a message to the CC BB to establish the connections ③′ shows the connections) for supporting the user's interactive service. If a user selects a service based on the information provided by an agent of the VOD BB, the related parameters for connections (video server address, video name, *QoS* for selected service, etc.) are sent to the GC BB.

④ The local GC BB performs *QoS* negotiations with a remote GC BB that manages a video delivery server and with the video delivery server through the remote GC BB.

⑤ If the response from a remote GC BB is that a video delivery server is available, the local GC BB requests the establishment of connections to the CC BB.

⑥ The CC BB that received the connection request from the GC BB checks whether the resources chosen through VOD BB by the user, and the last result determined through the negotiations in ⑤, are available or not. If the resources are available, the CC BB negotiates with the other CC BB in order to establish connections.

⑦ If network elements' resources as a result of negotiation between CC BBs are available, each GC BB sends a connection request message for creating terminal resources and connecting to networks.

⑧ After the completion of connections, service from the video delivery server is provided to a user.

For the second example of an ad hoc service BB, we describe a CSCW BB that is used for network resource management. The operation scenario is same as the scenario adopted for the VOD BB except the CSCW service does not use the stages after ④ for receiving service streams. We prototyped the CSCW BB using the proposed networking architecture.

7. Multimedia Terminal Architecture

The multimedia terminal mechanism for managing multimedia resources and users' requirements uniformly treats the various types of multimedia resources as logical devices. To realize this mechanism, we adopted GCA BB and MMC BB in the multimedia terminal. We described GCA BB in Section 2. In this section, we discuss MMC BB. MMC BB is a functional manager that manages the resources for a terminal. This manager receives control messages from a GCA BB, prepares necessary device objects, and manipulates those objects. There are several classes of device objects.

Connection model of MMC BB: An MMC BB is composed of 2 levels. The lower level handles data between Device Objects (DOs). The higher level manages all DOs. A DO sends the data to the lower level of the MMC BB as an UpStream, then the DO receives the data from the MMC BB as a DownStream. By this principle the MMC BB directly controls the data flow, and the problem of synchronization between media can be solved using this principle. The management of the stream that is described in the last part of this section is performed in the higher level of the MMC BB.

Device object: An MMC BB creates, connects and controls DOs. DOs are classified according to the kinds of media involved and their functions. Logical devices such as windows and files also can be handled as DOs. Using these principles, the MMC BB can manage the terminal resources uniformly. DO classification is as follows:

- Classification by means of input/output function
  - Source class: DO class generating information
  - Sink class: DO class consuming the information
- Classification by kind of media
  - Audio class, Video class and Data class (not continuous data)

The concept of stream process: An MMC BB manages a set of DOs that are immediately related as a "stream process". The control messages that are sent by a GCA BB mostly are handled as a stream process. There are some reasons why we adopted the idea of stream process:

- Generally, a set of DOs related to the same data stream are operated in the synchronization state. For such kinds of DOs, the operation for controlling DOs normally does not need to be performed on an individual DO. So, after DOs are connected to each other, operations on a stream process can be performed effectively.
- The set of DOs that belong to the same data stream is required to be scheduled by

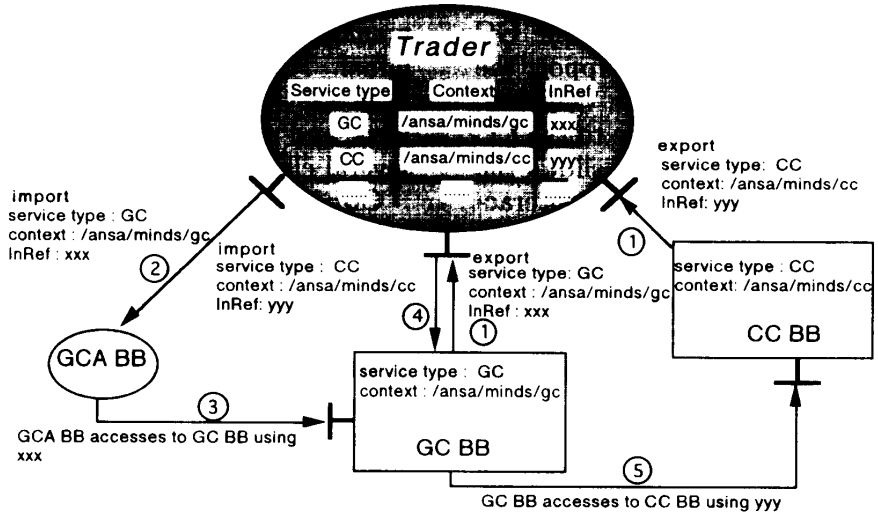

Fig. 8 Procedure using trader to establish connection.

Table 1 Example C++ listing for stream process.

```
// Creation of device object
DefaultMic mic(100);
DefaultSpeaker spk(100);

// Connection of device object
mic.Connect(&spk);

// Creation of stream process
Stream stream(15);

// Registration device objects to stream process
stream.AddDevice(&mic);
stream.AddDevice(&spk);

// Request of resource reservation to all device
      objects
  // Registered to stream process
if ( stream.Open() == mmcFalse){
 cerr << "Open Failed\n";
 exit(1);
}
// Resource reservation and waiting for
stream.BlockTillOpen();
```

a predefined procedure; for example, from source to sink for simplicity of operation. In order to achieve this, we introduced the concept of stream process.

Operation commands on a stream process include add, close, start, stop and resume. The group of DOs that must be synchronized is defined as a stream process. A programming example consisting of a stream process for effective management of DOs is shown in **Table 1**. To enhance execution performance of the MMC BB, mixers and multicasters on the network side, we used a thread[15].

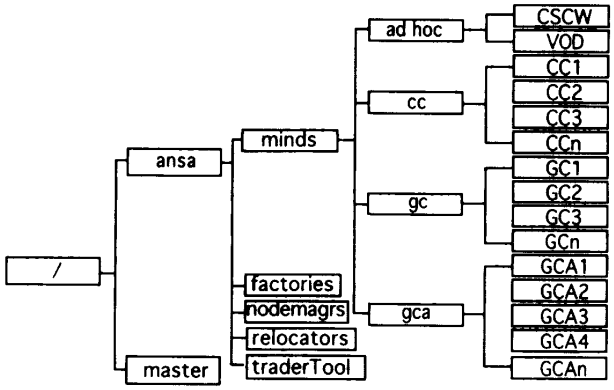

Fig. 9 Trader contexts in proposed architecture.

8. Realisation and Evaluation

The architecture we proposed in this paper is implemented on an ANSAware environment[16]. The service components were registered in a trader with service types and interface references. We now explain how to pass a message between building blocks as shown in **Fig. 8**.

① A GC BB and a CC BB export their own interfaces to the trader. ② A GCA BB imports from the trader an interface reference (InRef) to request service to the GC BB using the InRef. ③ A GCA BB sends a message to a service access point to the GC BB using the InRef that is imported from trader. ④ If a GC BB needs to establish a connection using the operation message of ConReq, it imports an InRef using the service type CC. ⑤ Then, the GC BB accesses the CC BB using the InRef. Trader contexts used in our architecture are shown in **Fig. 9**.

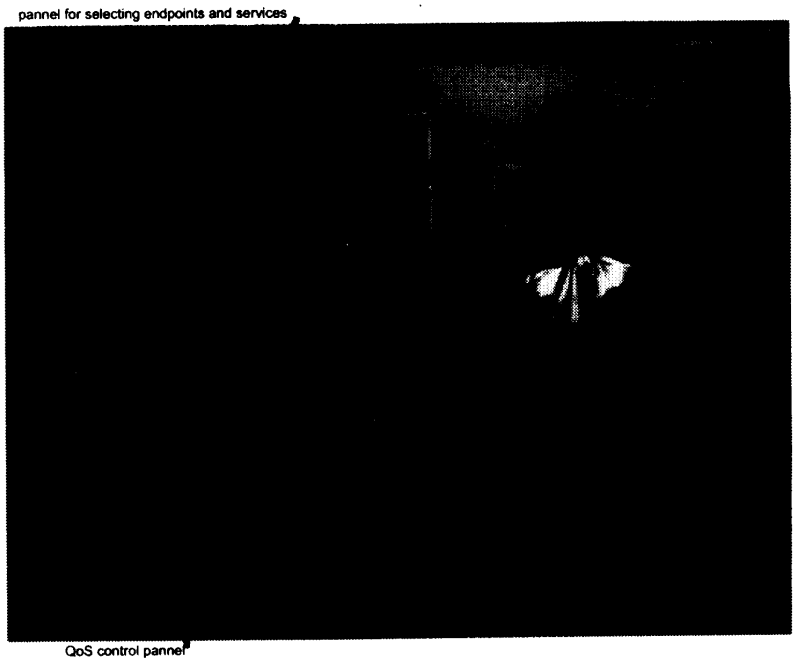

Fig. 10 Demonstration view for video conferencing.

Table 2 Result of measurement operation on entries.

Functionality	Average Access Time	
	DAP	LDAP
read_entry	64.233 ms	N/A
add_entry	414.638 ms	319.271 ms
modify_entry	423.368 ms	325.993 ms
delete_entry	399.121 ms	307.527 ms

Also, if we adopt the process of Fig. 8, the processing time for importing two InRefs (interface references) in video conferencing is about 20.2 ms. Recently, there is considerable interest in developing a real time DPE platform. Therefore, the performance of processing time using trader shall be rapidly improved.

The directory system used for managing user/session profiles by the GC BB made use of ISODE/QUIPU[17]. Especially, in order to incorporate the DUA function into our system, we implemented DUA functionality using a procedure call provided by ISODE/QUIPU. A session profile in the DIT (described in Section 3.2) can be flexibly changed during operation.

In **Table 2**, we show the results of time measurement taken to read and modify entry information and to add or delete an entry. This measurement was performed on a SUN SPARCstation 4. The read_entry function affects a user's service time. On the other hand, the other functions are not related to a user's service time. To enhance these processing times, we could use the LDAP (Lightweight Directory Access Protocol)[18] proposed by the ISODE consortium. By adopting this protocol in our architecture, performance was enhanced about 24% in comparison with DAP.

Since current small commercial ATM fabrics don't provide a media mixing function, we used a TCP/IP transport network for flexible control of the audio mixing function and the media multicasting function that we implemented by software.

To prove the usefulness of the ad hoc service BB, we constructed a CSCW BB. The demonstration views for the video conferencing and the special purpose network management that are realised using a CSCW BB as an ad hoc service BB are shown in **Fig. 10** and **Figs. 11–13**, respectively. These demonstration views (Figs. 11–13) illustrate group work for the effective management of a permanent virtual path (VP) as a leased line. In these figures, an agent manages the network information and deals with users' requirements that modify the bandwidth and path of the VP connection. It also has a negotiating function with carriers as the network provider. The configuration describing the role to be played by the CSCW BB is shown in **Fig. 14**. In this figure, the CSCW BB broadcast to all participants (i.e., user, agent, carriers) the data sent by the user, agent or carriers. So, the CSCW BB pro-

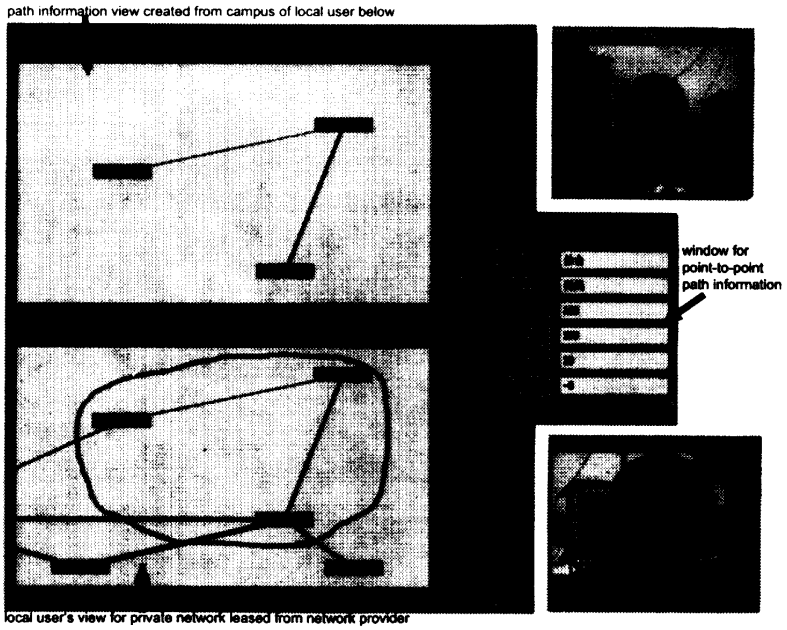

Fig. 11 User's windows in CSCW between user and agent.

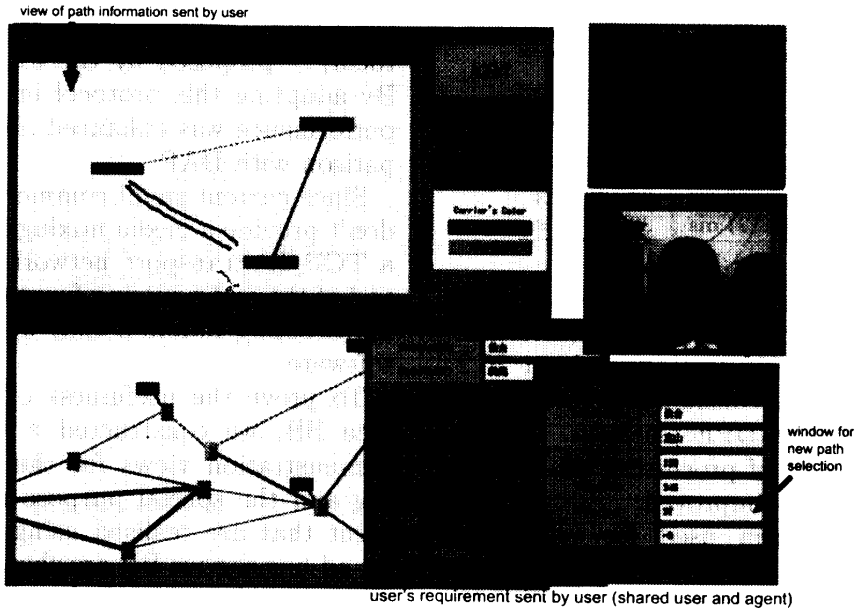

Fig. 12 Agent's windows view for CSCW between agent and carrier.

vides a cooperative work domain for the user, agent and carriers. The details of our CSCW application are beyond the scope of this paper.

9. Conclusions

This paper proposes service architecture and effective management architecture on a DPE that provides flexible multimedia services over a high speed transport network, and describes their implementations. The networking architecture has the underlying features to manage the multimedia network elements for connection/session managements and service provisions, and an effective *QoS* negotiation mechanism. The directory system making use of X.500 was constructed for the easy, more flexible management of service information such as *QoS* level, bandwidth, and the subject and owner of a session, and for network resource management in an open system environment.

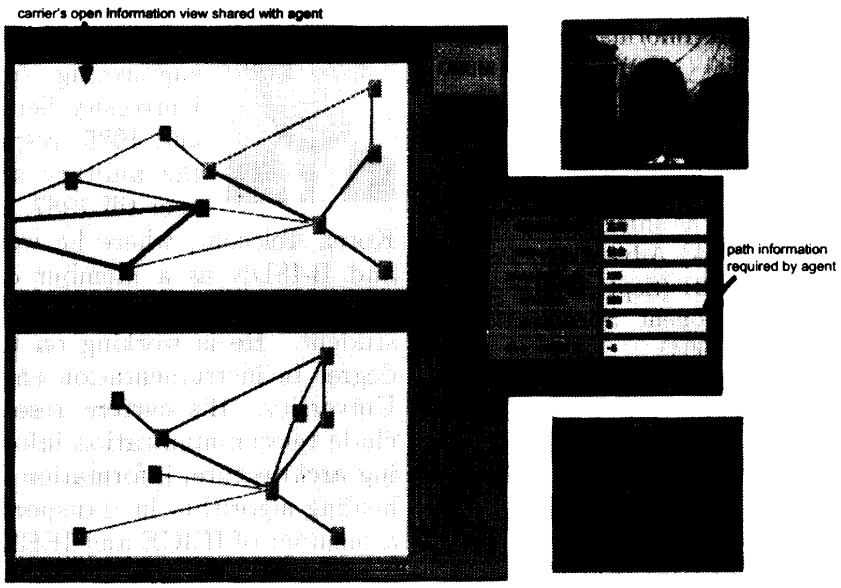

Fig. 13 Carrier's windows view for CSCW between carrier and agent.

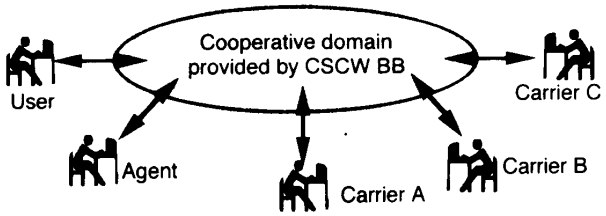

Fig. 14 Concept of domain provided by CSCW BB for cooperative work.

The architecture realised by a multimedia control building block in a multimedia terminal hides the complexity of physical elements and provides flexible connection to each device object. For the purpose of proving the usefulness of proposed architecture, we realized a CSCW building block as one of the ad hoc service building blocks, playing a role of service provider.

Finally, we believe that the architecture proposed in this paper is a useful type for providing various multimedia services on novel networking environments which are being studied in the TINA-Consortium and elsewhere.

Acknowledgments

The authors would like to thank Dr. Kenichi Kitami, Mr. Osamu Miyagishi, Mr. Fumito Sato. Mr. Sakae Chikara and Mr. Wataru Takita at NTT Telecommunication Networks Laboratories, Professor Jonathan Grudin at U. of California, Irvine, for their useful discussions and comments. The authors particularly want to thank our colleagues: Mr. Kiyoto Kawauchi, Mr. Hirofumi Abe, Mr. Dai Kashiwa and Miss Tomoko Yasui.

References

1) Ohmori, T., Maeno, K., Sakata, S., Fukuoka, H. and Watabe, K.: Distributed Cooperative Control for Sharing Applications Based on Multiparty and Multimedia Desktop Conferencing System: MERMAID, *Proc. IEEE ICDCS*, pp.538–546 (1992).
2) Rangan, P.V. and Swinehart, D.C.: Software Architecture for Integration of Video Services in the Etherphone Systems, *IEEE Journal on Selected Areas in Communications*, Vol.9, No.9, pp.1395–1404 (1991).
3) Hong, C.S., Yoneda T., Tanaka, K., Honda, S. and Matsushita, Y.: Multimedia Communication Networking Architecture Model for High Speed Network, *Proc. IEEE LCN '94*, pp.257–265 (1994).
4) Rangan, P.V., Vin, H.M. and Ramanathan, S., Designing an On-Demand Multimedia Service, *IEEE Communications Magazine*, pp.56–644 (1992).
5) Barr, W.J., Boyd, T. and Inoue, Y.: The TINA Initiative, *IEEE Communications Magazine*, pp.70–81 (1993).
6) Natarajan, N. and Ferro, M.: The Distributed Processing Framework in the INA Architecture, *Proc. TINA '93*, Vol.2, pp.255–267 (1993).
7) ITU-T: Principles for a Telecommunications Network Management, ITU-T Draft Rec.

M.3010 (1991).
8) ITU-T: Generic Network Information Model, ITU-T Draft Rec. M.3100 (1992).
9) Appeldorn, M., Kung R. and Saracco, R.: TMN+IN = TINA, *IEEE Communications Magazine*, pp.78–85 (1993).
10) Rubin, H., Caram, B., Flinchbaugh, G., Fratini, S. and Hall, M.: The Information Networking Architecture (INA) Advantage, *Proc. of TINA '93*, Vol.1, pp.3–21 (Sept. 1993).
11) TINA-C: Service Component Specification, TINA Draft Doc. (Nov. 1994).
12) ITU-T: Basic Reference Model of Open Distributed Processing-Part 3: Prescriptive Model, ITU-T Rec. X. 903, pp.63–68 (1994).
13) TINA-C: Connections Management Architecture, TINA Draft Doc. (Dec. 1994).
14) ITU-T: Data Communication Networks Directory, Rec. X.500-X.521 (Nov. 1988).
15) Abe, K.: Portable Thread Library, Ver. beta 1.3, http://sunfish.ics.es.osaka-u.ac.jp/PTL/ (1993).
16) APM Ltd: ANSAware 4.1 Application Programming in ANSAware, Document RM.102. 02 (1993).
17) Robbins, C.J. and Kille, S.E.: The ISO Develope Environment: User's Manual Vol.5, QUIPU, ISO Consortium (July 1991).
18) Yeong, W., Howes, T. and Kille, S.: X.500 Lightweight Directory Access Protocol, RFC1487, ISODE Consortium (July 1993).
19) TINA-C: Overall Concepts and Principles of TINA, TINA Draft Doc. (Dec. 1994).
20) TINA-C: Definition of Service Architecture, TINA Draft Doc. (Nov. 1994).
21) Campbell, A., Coulson, G. and Hutchison, D.: A Quality of Service Architecture, *ACM Computer Communication Review*, pp.6–27 (April 1994).
22) Nahstedt, K. and Smith J., M.: The QOS Broker, *IEEE MultiMedia* pp.53–67 (Spring 1995).

(Received March 16, 1995)
(Accepted December 8, 1995)

Choong Seon Hong received his B.S., M.E. in electronic engineering from Kyung Hee University, Seoul, Korea, in 1983 and 1985, respectively. He had the military service as a engineer till 1987. In 1988 he joined Korea Telecom, where he worked on N-ISDN and B-ISDN as a member of technical staff. From 1994, He joined Keio University as Ph.D. student. He is working on toward the Ph.D. degree in instrumentation engineering at Keio University. His current research interests include telecommunication information networking architecture, information filtering and self-healing algorithm in transport networks. He is a member of IEICE and IEEE.

Shinkuro Honda received the B.S. and M.E. in instrumentation engineering from Keio University, in 1993 and 1995, respectively. He is working on the Ph.D. degree at Keio University. His current research interests are in the design of virtual office and distributed systems.

Yutaka Matsushita received the B.S. in electrical engineering from Keio University, M.S. in computer science from the University of Illinois and a Ph.D. in electrical engineering from Keio University, in 1963, 1969 and 1978, respectively. From 1963–1988, He joined OKI Electric Industry Ltd. where he had served as the director of the integrated systems division. From 1989, He is a professor in department of instrumentation engineering of Keio University. He is author or coauthor of 17 textbooks and published the above of 100 papers. He received best author award from IPSJ in 1993. His more recent works have been in the area of groupware, mobile networks and telecommunication architecture. He served as a program chairman of IEEE ICDCS and general chairman of IEEE ICNP in 1992 and 1995, respectively. He is currently serving as the president of SIG GW of IPSJ and SIG MIS of IEICE. He is a member of IEEE, ACM, IEICE, Japanese Society for Artificial Intelligence and Japanese Society for Fuzzy Theory and Systems.