Broadband Service Creation and Operations

Ying-Dar Lin, Yuh-Tay Lin

Dept of Computer and Information Science
National Chiao Tung Universit
Hsinchu, Taiwan
Tel: +886-3-5731899
FAX: +886-3-5721490
ydlin@cis.nctu.edu.tw

Po-Ning Chen, Michael M. Choy

Advanced Technology Center
Industrial Technology Research Institute
Hsinchu, Taiwan

Abstract

Two challenges in deploying broadband services are the time-consuming process of service creation and the gap between the needs of service management and the functionalities of network management. We describe a methodology to simplify the process of service creation and service operations. A set of broadband service independent building blocks (SIBs) are designed and used to create and customize SIB graphs for broadband services. During service running time, the service agent interprets a SIB graph and executes SIB procedures which are all downloaded from the service provider. Two examples, video conferencing and video on demand, are given to demonstrate the feasibility of this approach.

1 Introduction

Multimedia applications with various forms of data contents such as text, still image, audio and video are extending their territory from standalone machines to the distributed network environment. The environment can range from locally interconnected LANs to wide-area Internet or ISDN (Integrated Service Digital Network). One problem encountered in such an environment is that these applications are big bandwidth consumers and, for some real-time applications, they need QoS (Quality of Service) guarantee. With the introduction of broadband networks such as ATM-based B(broadband)-ISDN and two-way interactive CATV networks, the problem is expected to be settled down.

Even with broadband networks in operations, there are still two challenges in deploying the so-called broadband services in the broadband networks, namely:

- to speed up the process of service creation and customization,
- to integrate upper-layer service management with lower-layer network management facilities.

Broadband services are far more complicated than traditional baseband services which involve only one data type. Taking the telephone service as an example for baseband services, the service has a fixed bandwidth requirement for its connection. Although there may be many service features, such as call forwarding and call waiting, to be added to the basic service mechanism, establishing a call remains to be the major task. However, broadband services demand much more than this. A VoD (Video on Demand) service needs to handle not only the connections establishment between a client and its server, in order to carry audio, video and possibly text streams, but also the various control messages such forward, pause and rewind. The service itself is beyond establishing a call. Creating a service is thus far more complicated and requires complex engineering efforts. There may be many variations of a service type. For example, we may have VoD services without any presentation control function, with only the "pause" control function, or with a full set of presentation control functions. Thus, services must be customerized according to the user's need.

Our approach to conquer the above two challenges is (1) to create and customize a service by combining SIBs (Service Independent building Blocks), a concept adopted from IN (Intelligent Network), into a SIB graph, and (2) to have the service agent, which interprets the SIB graph. The IN SIBs, for baseband telephone call with various service features, are not applicable and a new set of SIBs has to be re-designed. We examine two design alternatives for SIBs, namely, fine-grain and coarse-grain. We adopt the coarse-grain approach, while retaining the fine-grain mechanism, to design its associated SIBs which can be used to create and customize broadband services. The created SIB graphs which represent service logic are stored at the service provider. The SIB graph can be interpreted by the service agent at the user's site. The service agent, while executing the SIB graph of a service, can process the SNMP queries from the service manager. Two complete examples, video conferencing and VoD, demonstrate the feasibility of this approach.

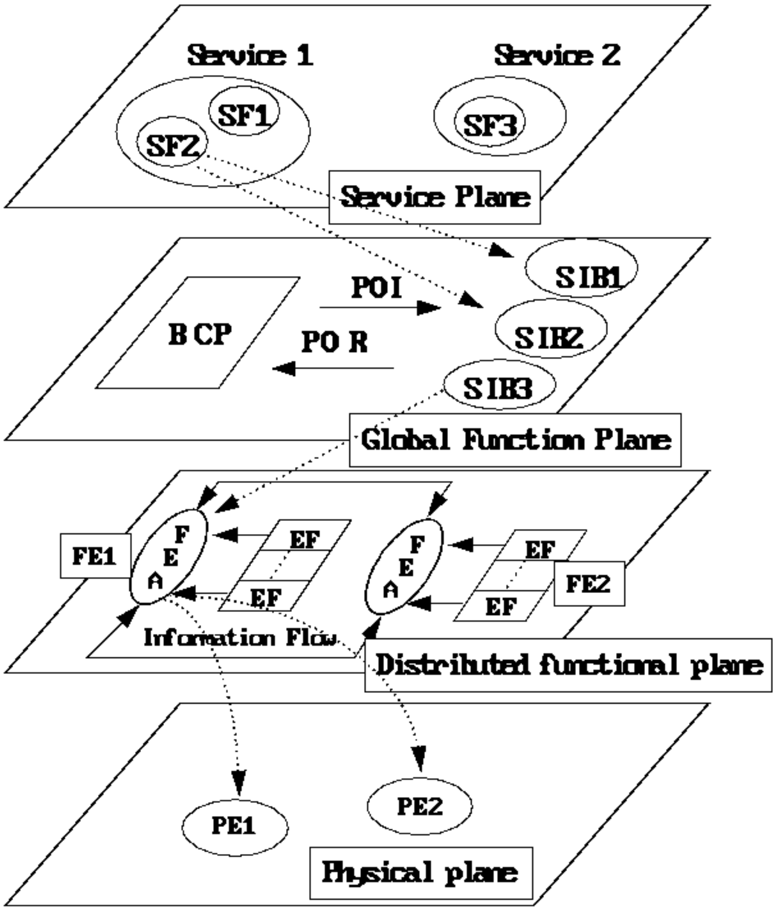
The rest of this paper is organized as follows. Section 2 provides a background of service creation concept. Section 3 discusses the design alternatives of SIBs. In Section 4, we focus on the overall architecture and design issues of broadband service creation and management environment. Section 5 gives two examples of broadband service creation. Section 7 describs our implementation plan. Conclusion and future work follow finally.

2 Service Creation

In an extremely competitive market place, telephone companies recognize that the speed with which new services are deployed is critical to meeting the needs of customers. Traditionally, the speed of deployment is constrained by the schedules of the equipment vendors. Although the telephone companies can find their customers' needs early, they need to report it to the equipment vendors and wait for their switch software development schedule, which may take years. This process can not meet today's requirements of fast variations of customers' needs. It also cost too much to provide a specific service to a specific customer. To solve this problem, a service creation environment(SCE) concept is introduced in the Intelligent Network(IN)[1] domain. The environment is a powerful tool that allows users to quickly create and customize services without having to know all the details of the physical equipments.

2.1 IN Capability Set 1(CS1)

In the IN Conceptual Model(INCM) (see Figure-1), the service in fact contains no information whatsoever regarding the implementation of the service in the network. In other words, the "IN-type" implementation is not visible from the viewpoint of services.



BCP	Basic Call Process	PE	Physical Entity
EF	Elementary Function	POI	Point Of Initiation
FE	Functional Entity	POR	Point Of Return
FEA	Functional Entity Action	SF	Service Feature
IF	Information Flow	SIB	Service Independent Building block

Figure 1: The IN-Conceptual Model

To speed up the service creation process, the services are modularized into some basic building blocks, which are named SIBs (service-independent building block). As self-explained by the name of basic building block, these SIBs are service-independent and common for most services. They can be used to build different kind of services according to the needs of subscribers.

The procedure to build up a service in IN is to first select some proper SIBs and then to chain them together according to the service requirement and logic, which is usually called ***SIB chain***. The SIB chain will be attached to the BCP (basic call processing) SIB through POI (point of initialization) and POR (point of return). The BCP SIB as defined in CS1[2] is a specialized SIB that provides the basic call capabilities. It enables the proper SIB chain based on the POI received, and result in the appropriate POR based on which POR is used to complete the SIB chain. A complete service logic, namely ***SIB graph***, is a combination of a BCP SIB and one or more SIB chains. For ITU-I CS1 standardization, 13 SIBs are specified (excluding the BCP SIB). Nine POIs and 6 PORs are also defined for BCP SIB in the same document(see Table 1). A SIB graph example of *Call Forwarding on Busy* service is shown in Figure-2. In this example, if the response of a call is "Busy" or "No answer", the address will be translated to a new address. The call is then forwarded to the new address. If the response from the new address is still "No answer", the call is cleared.

SIB	POI	POR
User Interaction	Call Originated	Continue with Existing Data
Translate	Address Collected	Proceed with new Data
Log Call Information	Address Analyzed	Handle as Transit
Limit	Prepare to Complete Call	Clear Call
Charge	Busy	Enable Call Party Handling
Queue	No Answer	Initiate Call
Status Notification	Call Acceptance	
Compare	Active State	
Screen	End of Call	
Service Data Management		
Verify		
Algorithm		
Distribution		

Table 1: IN CS-1 SIBs, POIs, and PORs

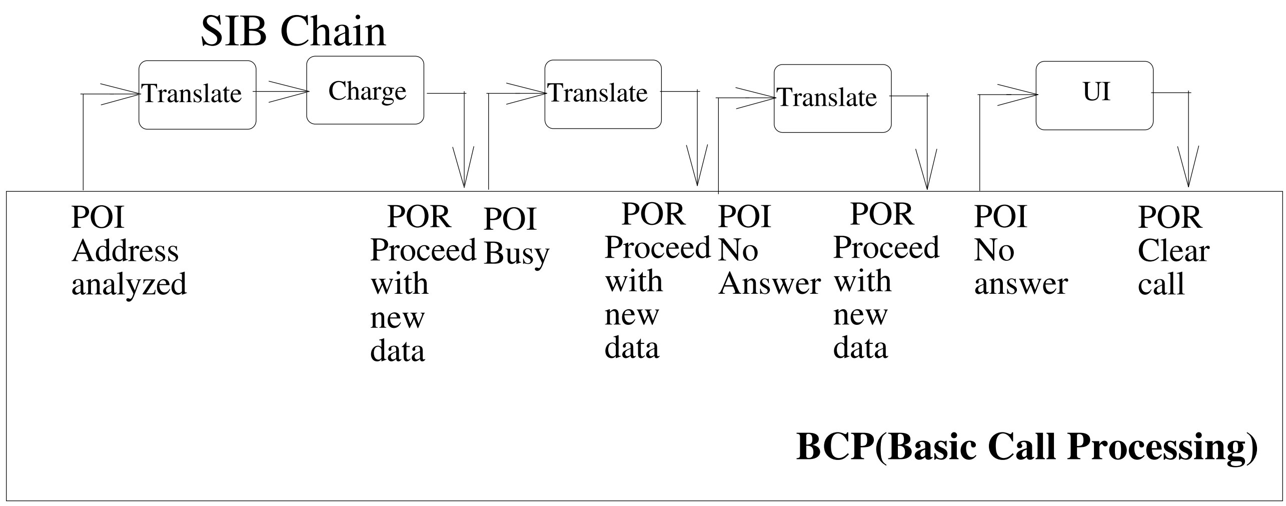


Figure 2: SIB graph of *Call Forwarding on Busy*

2.2 Problems with CS1 SIBs

Problems have been found in the SIB graph structure[3]. In Figure-2, there are two POIs of the type "No Answer." The BCP must maintain a call context and the rules to derive decisions based on the existing call context and the actual call events. Such a functionality is not support by the current BCP definition and it is questionable whether it will be possible to enhance the BCP functionality accordingly. To implement a network-wide BCP SIB, which is responsible for input call/request processing including call connection, call disconnection, and retaining call context for further processing of that call instance, becomes somewhat unmanageable.

A solution also proposed in the same document to solve the problem mentioned above is to decompose the BCP into a new set of SIBs, which includes:

(1) State SIB -- Service states can be directly related to states of the call processing.
(2) Incoming event SIB -- State transitions of the services are caused by incoming events.
(3) Outgoing event SIB -- Outgoing events initiate the sending of the control messages.
(4) Action SIB -- Each implements a service functionality.

With this new set of SIBs, we can describe clearly the interactions between different SIB chains. Thus a complete and working service can be described. The *Call Forwarding on Busy* service described in this way is shown in Figure-3. Upon receiving the same incoming event "No Answer", two state SIBs of "Wait for Event_1" and "Wait for Event_2" are used to distinguish the corresponding actions. It eliminates the confusion in Figure-2.

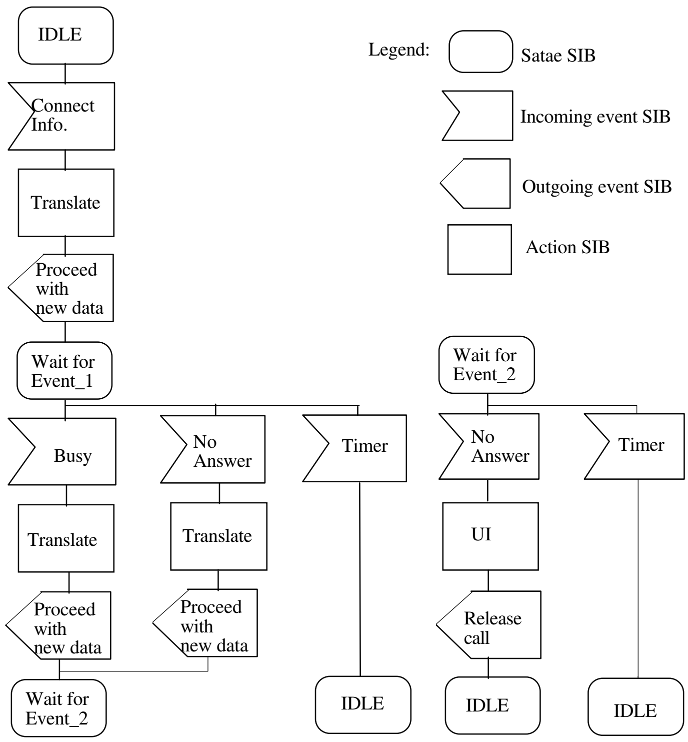


Figure 3: SIB graph of Call Forwarding on Busy with extensions of SIB concepts

2.3 Baseband vs. Broadband Service Creation

The IN services are baseband. No matter what services(see Table 2) is provided, baseband services always establish the same type of connections. The connections may have the same bandwidth allocation and performance requirements. The differences lie in the added service features which change the basic service logic of a call. It is not necessary to consider resource and performance management issues in the baseband environment.

Unlike baseband services, broadband services(see Table 3) require high and variable bandwidth, low latency. They generate multiple types of network traffic load. In addition, broadband services usually contain user interactions during the service. Mapping SIBs from baseband to broadband, we need a new set of SIBs for the functions other than connection establishment. Parameters are also needed to describe various characteristics of different broadband services. With the characteristics of multimedia transmission and representation, resource and performance should be carefully managed for the effective utilization of the system.

Freephone	Universal Personal Telecommunications
User-Defined Routing	Virtual Private Network
Originating Call Screening	Abbreviated Dialing
Call Forwarding	Terminating Call Screening
Destination Call Routing	Call Distribution
Security Screening	Televoting
Split Charging	Premium Rate
Credit Card Calling	Account Card Calling
Mass Calling	Automatic Alternative Billing
Malicious Call Identification	Follow-Me-Diversion
Call Rerouting Distribution	Universal Access Number
Selective Call Forward on Busy/ Don‘t Answer	

Table 2: Set of Benchmark CS1 Services

Services	Service Features
Video on Demand (VoD)	select/cancel; start; stop; pause; fast forward/reverse; scan forward/reverse; jump to different scenes; show counters
Teleshopping (TS)	select; video; text; motion video with audio; audio or graphics
Karaoke on Demand (KoD)	select; instrumental music; vocal music; highlighted display of lyrics video scenes; record; voice replay
Distance Learning (DL)	control camera; display documents; control what is displayed on monitor interactively; schedule classes; browse; join/leave classes
TV Listing (TL)	select; display of distribution programs/schedules; progressive queries
Video Conferencing (VC)	a real-time bi-directional exchange of audio/video/ data information; announce/establish/control conference
Near Video on Demand (NVoD)	broadcast of video; no real interaction/select channel; pause

Table 3: The sample broadband services.

3 Service Independent Building Blocks

3.1 Design Alternatives for Building Blocks

Our goal is to define the SIBs and implement an SCE(Service Creation Environment) for broadband services. There are two alternatives to define the SIBs, namely fine-grain and coarse-grain. In fine-grain approach, the SIBs are more primitive and the service provider can have a detailed description of desired service logic in the service creation process such as handshaking in connection establishment. In coarse-grain approach, SIBs are more powerful. Each SIB has one or a few pre-defined procedures, service provider need not concern with the detailed activities. They represent a trade-off between flexibility and friendliness.

Figure-4 is an example of connection establishment in fine-grain approach. The state SIBs of *wait-1*, *wait-2*, and *wait-3* are used to record the different status during the process of call establishment. Hence, the proceeding action can be determined upon receiving a specific incoming event. Initially, an originator sends out a request of *connect* and enters *Wait-1* state. If the response from the network is *reject*, it notifies the user and returns to *Idle* state; otherwise, the response says that the other end is *busy*, the service logic may try to translate the destination address to a new address or wait for a while, then try to *connect* again. If the response is still *busy*, it may ask the user to make a decision: either retry, give up(hang up), or ask a service program to monitor the status and call back later. The retry count can be increased by inserting different *Wait* state SIBs to represent the number of times that have been retried

.

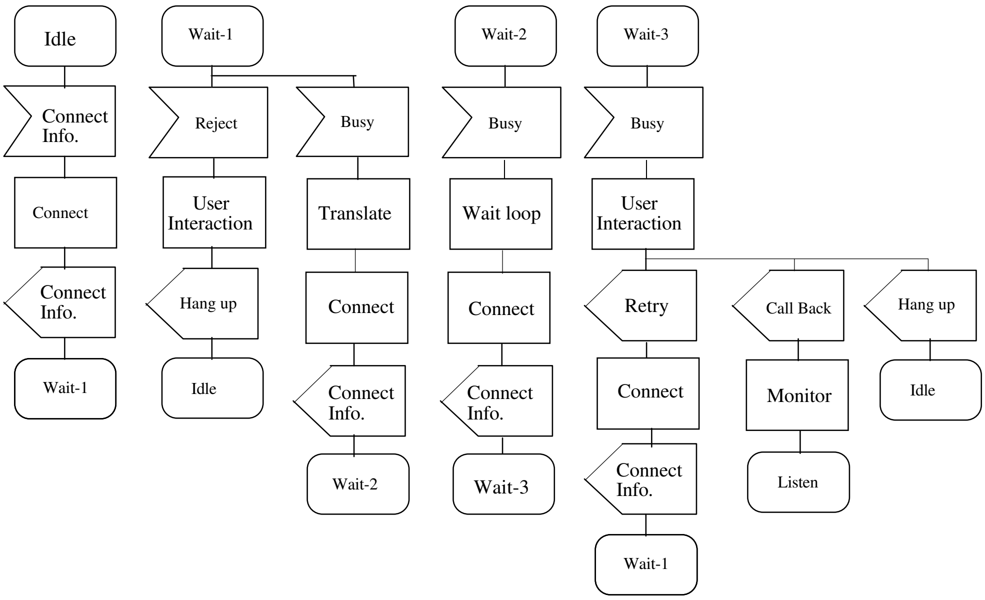


Figure 4: SIB graph for Connection Establishment

In this approach, a service provider can design a specific service logic(e.g., retry count, waiting period, and call back) through the proper combination of SIBs. However, they should pay more attention on transforming the service logic into a SIB graph in the process of service creation.

In the cases where most of the services have the same connection establishment procedures, it is inefficient and not necessary for each service provider to design a detailed SIB graph as in Figure-4 in every service. For simplicity, some common service logic can be pre-defined as a set of macros. In this way, the connection establishment procedure in a SIB graph may be replaced with one macro SIB(see SIB design in the next section). The drawback is that the service creator should use the pre-defined macro SIBs, only limited degree of flexibility can be provided if SIB parameters supports service logic description. Those parameters decide the degree of flexibility. To retain full flexibility, fine-grain approach should be kept such that service creator may create their own macro SIBs if the available macro SIB can not match their needs.

3.2 SIB Design

We have mentioned that a set of parameters of a SIB is supplied for the description of broadband service characteristics. Service supported data(SSD), provided at service creation time, and service instant data(SID), generated/received at run time, are the main parameters. Each SIB is implemented as an independent (executable) object with logical inputs, logical outputs, SSD, and SID(see Figure-5). The SSD includes some fixed data associated with this SIB, some field pointers for dynamic links/information, and some QoS parameters related.

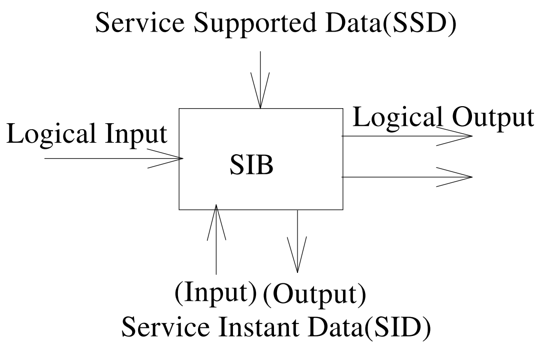


Figure 5: The graphic representation of SIB

Considering a broadband service, the functionalities can be divided into four parts: connection establishment, data transport, connection release, and user interaction. In coarse-grain approach, the corresponding SIBs, their symbols, and functions are illustrated in Figure-6. The parameters and procedures of each SIB will be described in the next section.

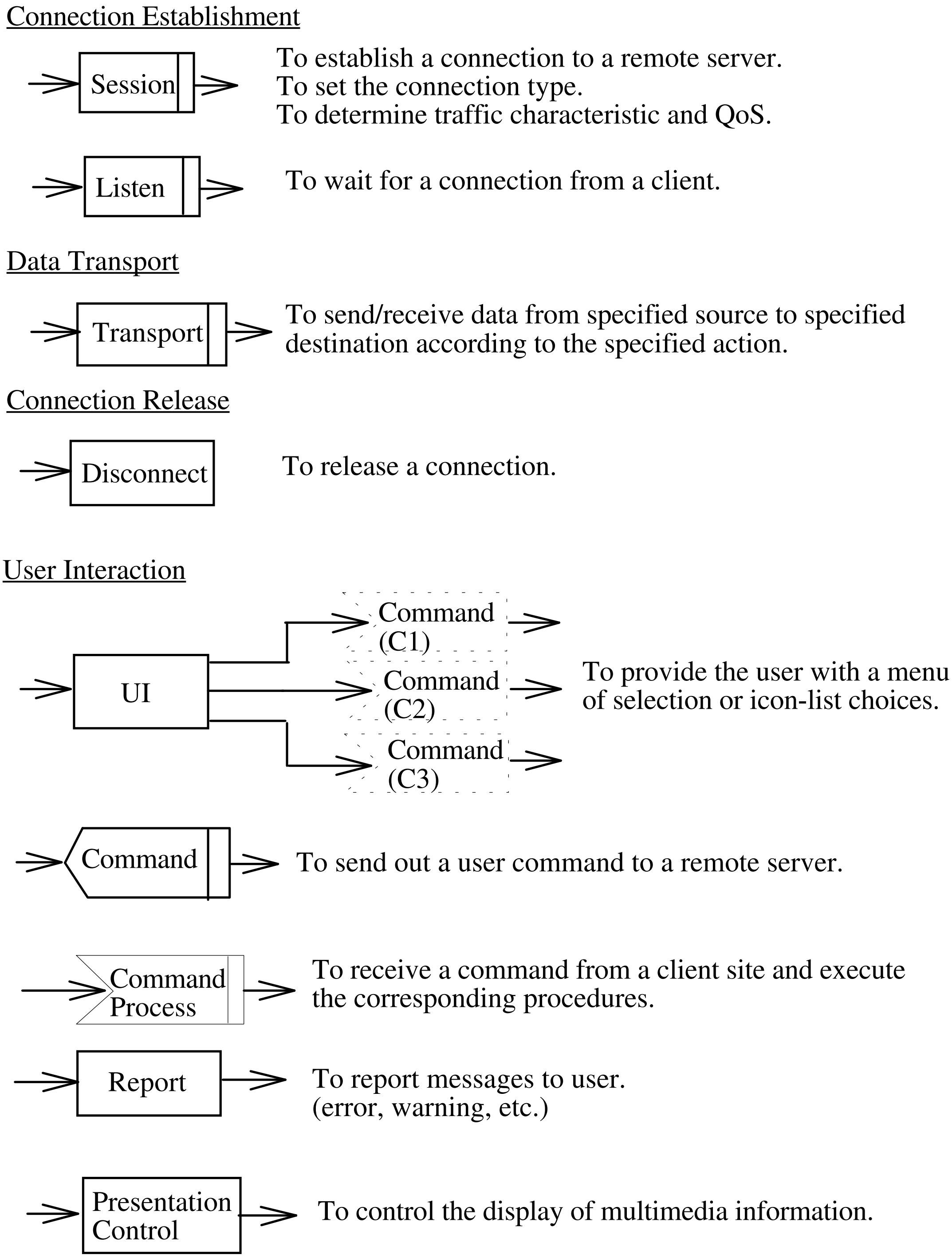


Figure 6: The functional description of the SIBs

4 Architecture and Design Issues

4.1 Overall Architecture

In our model, the broadband network with network-level QoS parameters is assumed. The broadband service provider may have several servers and at least one SCE station attached to the broadband network as depicted in Figure-7. Different from the IN-type service creation concept, each service is now constituted by two SIB graphs: one for client, and one for server. Therefore, with the pre-defined set of SIBs, a diversity of broadband service SIB graph pairs can be created, and are stored in the service provider which is a Web-based SIB server. The library in the service provider contains three categories of data or routines:

- SIB graphs for services provided
- SIB procedures associated with each SIB
- service agent which interprets the SIB graph at a service user's site

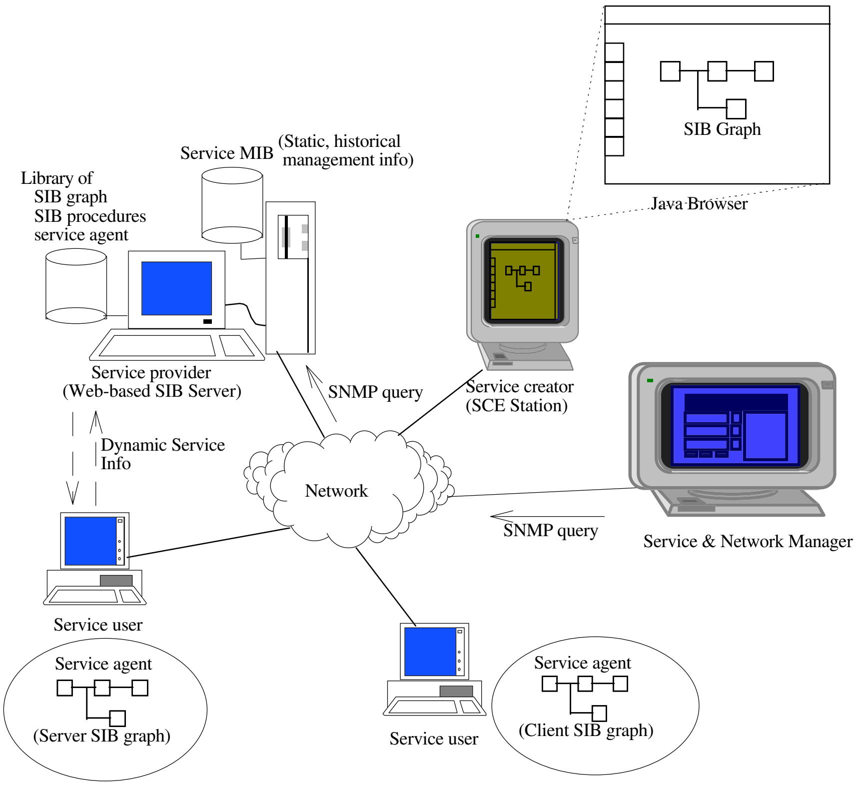


Figure 7: Overall Architecture

The service creation environment is embedded in a JAVA[5] browser. A friendly graphical interface with drag-and-draw and routing capabilities is implemented using JAVA code. Therefore, for the SCE station, we are free to choose among those platforms that support JAVA interpretation capabilities. The created SIB graphs are stored back to the SIB server.

Each time a client requests for a service, it invokes its WWW browser to enter the WWW home page of the service provider. On the home page, the access right of the clients will be checked, and some readily created services will be displayed. After the client selects a specific service, for example a VoD service, the corresponding VoD SIB graph for client will be transmitted to the client machine; and also, the other half of the VoD SIB graph pair (the SIB graph for server) will be transmitted to the available video servers. Since now the SIB graphs are respectively transmitted to the server and client, they can communicate with each other directly by making the corresponding

SIB module active(see Figure-8). Hence, the overhead of services can be reduced to minimum if the SIB modules can handle the physical components directly.

In order to interpret the SIB graph and run the service routines, the service user, either client site or server site, may need to download the service agent and the SIB procedures of invoked SIBs from the service provider. This download operation can be eliminated if the agent and procedures are pre-installed before the service invocation time. This service agent and the code of SIBs can be implemented in two ways. One is to implemented in JAVA code, and transmits to the service user site together with the SIB graph. This is similar to the diskless client, no preparation is needed before the execution of SIB graphs in a workstation. The other way is to pre-install the codes in the workstations which may need to execute the SIB graphs. This restricts the execution SIB graphs in the workstations which have the codes been pre-intstalled. Performance may be gained by the omit of transmission of those codes for the SIBs and the service agent.

In addition to interpreting the service logic, the service agent can also act as a service management agent. It monitors the execution of SIB graph, and reports the information about the service to the service provider. A service MIB is required to keep track of and log the service provided by the service provider. It records two categories of information, namely, the service-level activities and the mappings between service and network MIBs.

The service manager may pass a SNMP query to the service provider, if it queries static or historical management infomation, e.g, service usage, connection type, connection duration, or traffic load, the service provider will extract those information from its service MIB. If the query asks for information about the SIB graph which is being run by some service agent, it will be passed to the service agent. Dynamic information will be reported by the service agent according to the query.

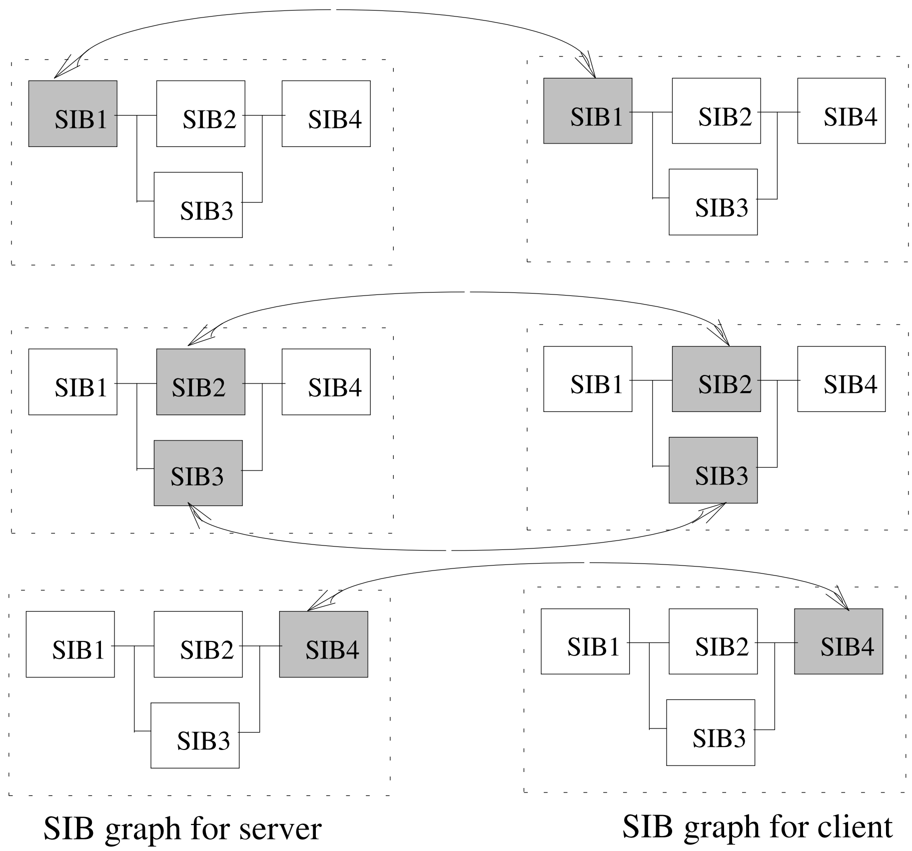


Figure 8: The active SIBs in the SIB graphs directly communicates with the corresponding ones at each time instant

4.2 SIB Parameters(????)

<< In implementing independent executable SIBs, the Common Object Request Broker Architecture (CORBA) has proved to be very useful[4]. Its object oriented technology can make distributed system design easier and provide high level abstractions. Roughly speaking, each SIB module is implemented as a CORBA object. >>

According to the SIBs in Figure-6, their SSD, SID, and procedures are listed in the following. Various values of SSD can be filled at service creation time according to different requirements of broadband services

A. Session

SSD	1. Uni/bi-direction, symmetric/asymmetric 2. Traffic Descriptor (two sets for bi-direction/asymmetric) mean bandwidth peak rate burst length direction 3. QoS (two sets for bi-direction/ asymmetric) delay, loss, jitter, direction.
SID	1. Client Address (IP Address + Port#) 2. Server Address (IP Address + Port#)

Procedure	1. Open: Open a connection device. 2. Bind: Get a service access point. 3. Connect: Send out a connection request.

This SIB sets the connection type, traffic characteristics, and QoS. These parameters are can be pre-defined according to the service type at service creation time. Service creator may define a set of SSDs such that users may choose their requirements. Multiple sets of SSD and SID are allowed for multiple connections. (Broadband service usually requires multiple connections for different types of data contents.)

B. Listen

SSD	none
SID	1. Source Address (IP Address + Port#) 2. Destination Address (IP Address + Port#) 3. SSD of Session
Procedure	1. Open 2. Bind 3. Listen: Wait for connection request. 4. Accept: Accept a connection request from server.

Here the connection type, traffic characteristics, and QoS are received from the network. No SSD is necessary.

C. Transport:

SSD	none
SID	1. Source Address (IP Address + Port#) 2. Destination Address (IP Address + Port#) 3. Data
Procedure	1. Send/Receive: Data transfer

After connection establishment, the source and destination addresses are received from Session/Listen SIB as input SID.

D. Disconnect

SSD	none
SID	1. Source Address (IP Address + Port#) 2. Destination Address (IP Address + Port#)
Procedure	1. Close: Release the connections

E. UI

SSD	1. List of (Command)
SID	1. Command 2. Parameters
Procedure	1. Command Selection

A command may be associated with permitted parameters to increase its flexibility.

F. Command

SSD	none

SID	1. Command 2. Parameters
Procedure	1. Send

It sends out the command accepted by UI to the remote server

G. Command Process

SSD	1. List of (Command, Procedure)
SID	1. Command 2. Parameters
Procedure	1. Fork

This SIB contains a list of command and its corresponding procedures. Upon receiving a command from a client, it forks a process to handle the corresponding proedure.

H. Report

SSD	none
SID	1. Messages
Procedure	1. User Notification

I. Presentation Control:

SSD	1. List of (Data Format, Data Player)
SID	1. Data Format 2. Data
Procedure	1. Call

It takes the data as parameters to call the corresponding data player according to the data format.

As for services such as Videophone and Video-conferencing, the corresponding SIB graph will be transmitted to those clients/hosts who request such services. After that, they can create the connections directly. It is worth mention that all the limitations of the service, such as allocated bandwidth, are specified in the SSD of the SIB. Any violation of these limitations will be rejected by the SIB module. If, say, the client requires more bandwidth, it should send its new request to the SCE station (possibly pay more bills), and the SCE station will modify the SSD of the SIB. In short, only the SCE station has the right to modify the SSD of the SIB.

4.3 Service Agent

When a client requests for a service, a pair of SIB graphs for client and server are transmitted to client and server respectively. An internal representation of a SIB graph, instead of the logical views is transmitted. Upon receiving the internal representation of a SIB graph, the service agent takes the responsibility to interpret this internal representation into its corresponding service logic.

The information stored in the internal representation of a SIB graph includes the SSDs of used SIBs and the link status between those SIBs. Let each data type have a unique identifier, the information of a SIB can be divided into fixed length data, encoded in the form of (Code, Value), and variable length data, encoded in the form of (Code, Length, Value). Code means the unique identifier, while Length represents the length of variable length data, and Value contains the real information. The overall internal representation in our model is a hierarchical structure, multiple occurrences are permitted for multiple sets of the same data type.

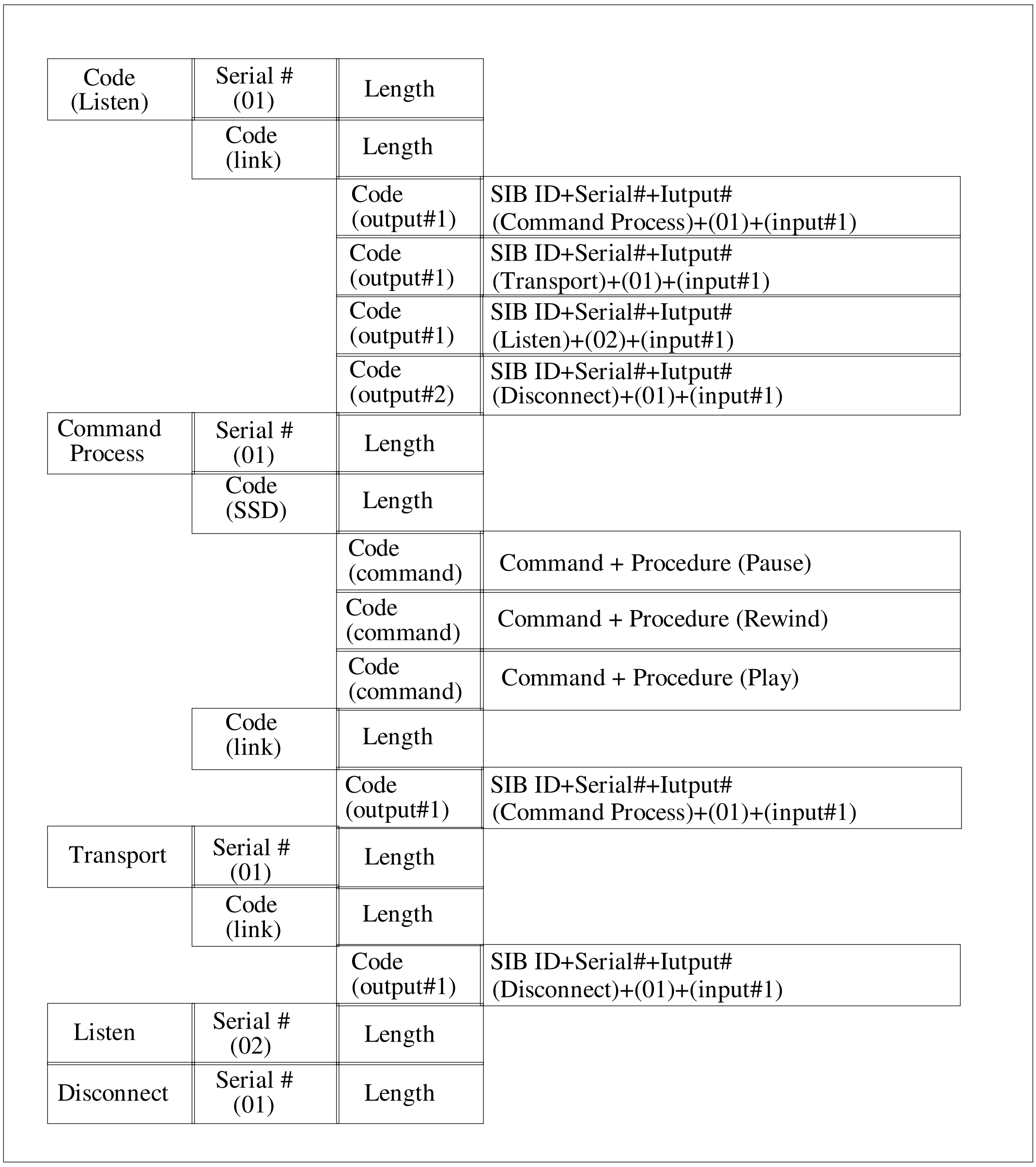


Figure 9: Internal representation of the SIB graph for VoD server

Figure-9 is an example of the internal representation of the SIB graph for VoD server in the next section. Although each SIB has a unique identifier(SIB ID). it may appear twice or more in a SIB graph, a serial number(Serial #) is tagged to form a real unique identifier. A link between SIBs is identified by a logical output identifier (output #) to a SIB identifier(SIB ID+Serial #) plus its logical input identifier (input #).

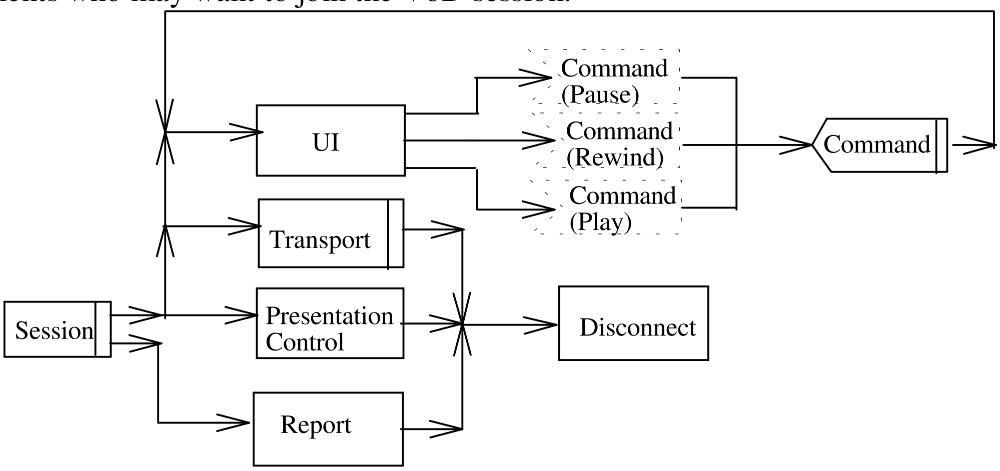
5 Examples

Following the architecture and design issues described above, we use two broadband services, video on demand and video conferencing, as examples of service creation to show its feasibility.

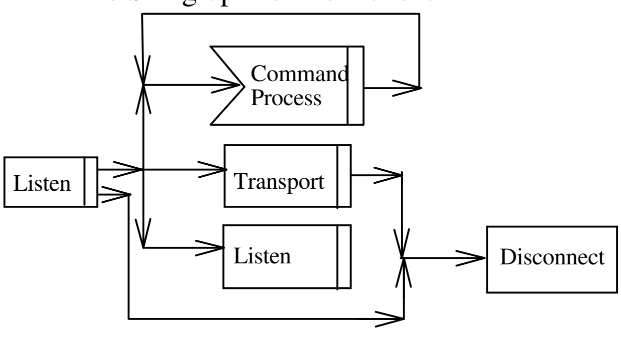
5.1 Video On Demand(VoD)

Figure-10 shows the SIB graph pair of VoD. At the client site, three processes(UI, Transport, and Presentation Control) will be forked if the connections are established successfully; otherwise, an error message is reported to the user and then the connections are disconnected. The UI SIB accepts user's command and then sends it to the server via the Command SIB. The command list in the UI SIB may be variable due to different charge policies. The Transport SIB receives data from the server, and sends the data to the Presentation Control SIB to control the display of multimedia information.

Also, three processes(Command Process, Transport, and Listen) are forked at the server site when the connections are established successfully. The Command Process SIB accepts user's command from the client, and then call the corresponding procedures for the command. The Transport SIB transmits multimedia information to the client, and the Listen SIB keeps waiting for the other clients who may want to join the VoD session.



The SIB graph for VoD client

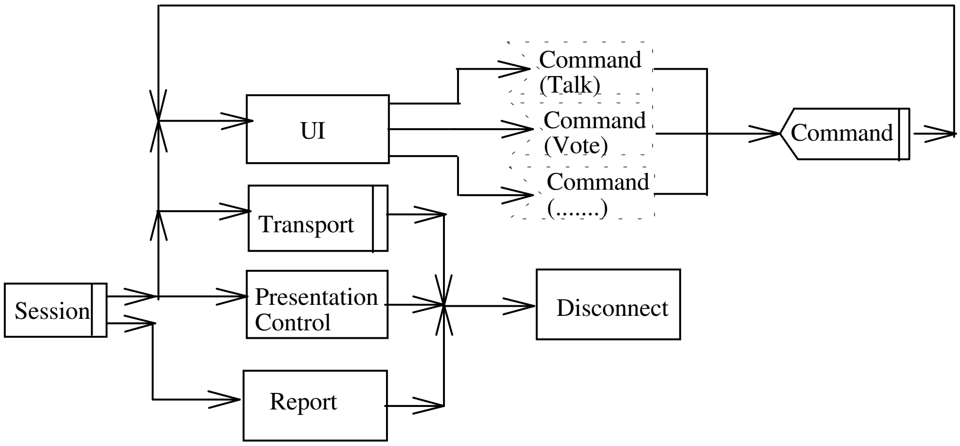


The SIB graph for VoD server

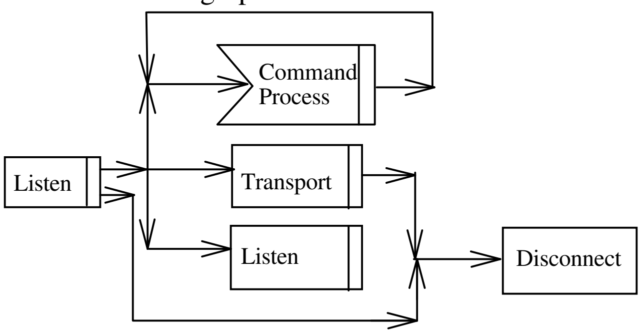
Figure 10: The SIB graph pair for VoD services

5.2 Video Conferencing(VC)

Figure-11 shows the SIB graph pair of VC. It is similar to VoD. As in VoD, we assume that there is a VC server, the chair of the VC initially set up the connections between his client site and a VC server. The Listen SIB at the server site keeps waiting for the other members of the VC to join. The multimedia information is distributed to all the members in the VC through the VC server. The differences from VoD are the command list of the UI SIB and the type of connections which are bi-directional.



The SIB graph for VC client



The SIB graph for VC server

Figure 11: The SIB graph pair for VC services

6 Implementation Status

Since CORBA is the most feasible candidate for iplementating the SIBs, we adopt **orbix[?????],** a full implementation of CORBA, as our implementation tool.

We have developed a VoD system. So far, it only has “PLAY” and “QUIT” functions. VoDclient and demandServer are two CORBA objects. The drawback of this release is that only VoD service can be served by these CORBA objects. The executing process is illustrated in Figure-12.

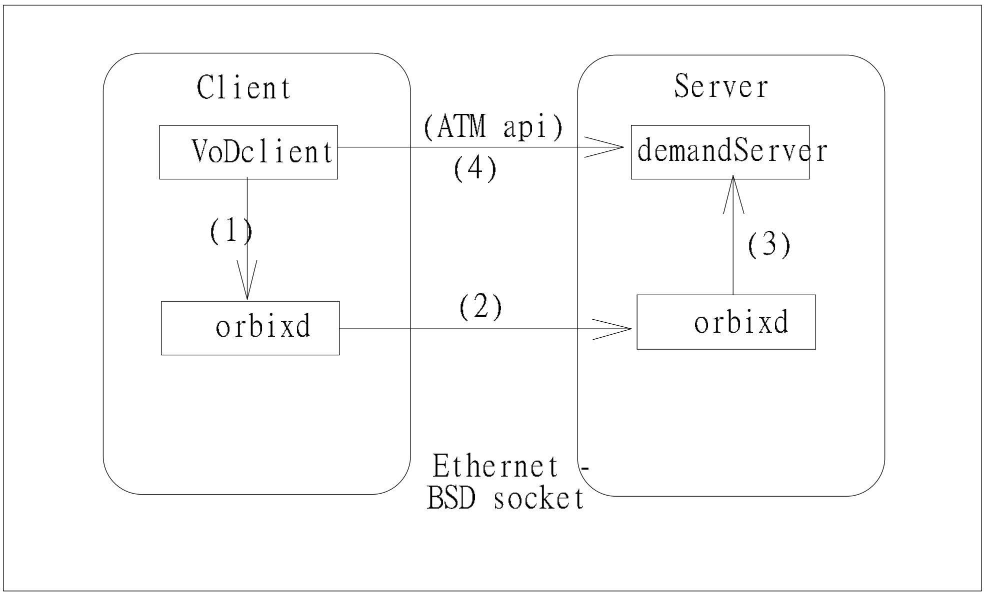


Figure 12: The executing process of VoD system.

In Figure-12, we can find that there are **orbixds** in both client and server sites. **orbixd** is a daemon process. It is used to wait for the request to the server and to connect client to a existing server process. When the connection has been established, orbixd is no longer needed. After the SIB graph pair are transmitted to server and client site respectively, the process can be divided into three steps. In step one, VoD client tells the orbixd his wish, then the orbixd in the client site will send a request to the orbixd in the server site. They communicate with each other by BSD socket over ethernet. When the orbixd in the server site receives the request, it will invoke the demandServer, this is the step 3. Finally VoDclient and demandServer will communicate with each other directly via ATM API over ATM network. This can reduce the service overhead just as we described before.

This verdion of VoD service just demonstrates the feasibility of implementing SIBs in CORBA objects. To increase the flexibility, we must divide the objects into more primitive objects. For example, session, listen, command, command process, etc. Then we can construct various services, e.g., VoD, VC, from these objects.

6.1 Service creation Environment

6.2 Service Agent

7 Conclusion and Future Work

An architecture of service creation has been presented in this paper. What we have discussed include design alternatives of SIBs, SIB design, service creation, and finally the operations of created broadband services. There are also two examples, video conferencing and

VoD, to show the feasibility of this architecture. Some design issues about service management are also covered.

Being able to run the services is one thing; being able to manage them is another. Traditionally, network management concerns the lower-layer activities of the distributed environment. However, service management watches activities beyond the connection level. A service may setup several connections at the same time and involve system resources other than the network resources. Therefore an integration of and mapping between these two are required to ensure a proper service quality. The detailed design about the service MIB and the mappings between service management and network management are parts of our future work.

The client-server service creation architecture has the advantage that the processing load has been distributed over the network, and the network overhead has been reduced due to direct communication among clients and servers. Implementation of such independent SIB modules is, nevertheless, still dependent on the physical platforms, as well as the physical components installed on the client machine. In order to make the service running environment more transparent to platforms and physical components, an interpreter-like environment, such as a JAVA browser, may be a solution. The main problem of using an interpreter-like terminal could be the performance, for which in our opinion can be greatly improved by an interpreter accelerator. The SUN MicroSystems has announced that the picoJAVA processor will be available around mid-1996. A more powerful microJAVA processor with application-specific I/O, memory, communications and control functions will be available in the spring of 1997. Therefore, a possible future work for us is to evaluate the performance of the JAVA-coded SIB modules, and use them as bases for service creation in the broadband network testbed.

Reference:

[1] Q-Series Intelligent Network Recommendations, ITU, Helsinki, March 1993.
[2] Q-1213 Global Functional Plane for Intelligent Network Capability Set 1, ITU, March 1993.
[3] Ralf Rieken, etc., “IN Service Creation -- Problems of the CS1 SIBs and How to Solve Them”, Proceeding of the Intelligent Network Technology Conference (INTC), Taipei, Taiwan, pages C2-1~C2-11, 1995.
[4] Object Management Group Inc. The Common Object Request Broker: Architecture and Specification, December, 1993.
[5] Sun MicroSystems, JAVA, URL: http://java.sun.com, 1996.