ABSTRACT

Three challenges for deploying broadband services are the time-consuming process of service creation, the interoperability over heterogeneous platforms, and the gap between the needs of service management and the functionalities of network management. In this article we describe a methodology to attack the first two challenges, namely, to simplify the process of service creation and provide a platform-independent framework for 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. Web, Java, and CORBA are the technological elements of this methodology. Two examples, video conferencing and video on demand, are given to demonstrate the feasibility of this approach.

Broadband Service Creation and Operations

Ying-Dar Lin, Yuh-Tay Lin, and Po-Ning Chen

National Chiao Tung University

Michael M. Choy, Industrial Technology Research Institute

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 local area networks (LANs) to wide-area Internet, ISDN (integrated services digital network), or broadband ISDN (B-ISDN). One problem encountered in such an environment is that these applications are big bandwidth consumers and, for some real-time applications, they need quality of service (QoS) guarantees. With the introduction of broadband networks such as asynchronous transfer mode (ATM)-based B-ISDN and two-way interactive cable TV (CATV) networks, this problem is expected to be settled down.

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

- To speed up the process of service creation and customization
- To interoperate services over heterogenous platforms
- 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 telephone service as an example of 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 the major task. However, broadband services demand much more than this. A VoD (video on demand) service needs to handle not only connection establishment between a client and its server in order to carry audio, video and possibly text streams, but also the various control messages such as 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/resume" control function, or with a full set of presentation control functions. Thus, services must be customized according to the user's need.

A restriction imposed on a service might be that the service must be deployed on the platform for which it is designed. Running a VoD client, which is designed for personal computers, on a workstation is often impossible. Software has to be rewritten for each platform. This raises deployment cost and impedes service deployment. The third challenge is the gap between service management and network management. Being able to

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.*

0163-6804/97/$10.00 © 1997 IEEE

deploy a service is one thing; being able to manage it is another. Service management, including surveillance and tune-up, requires not only application-level measurements but also network-level management information such as cell loss and delay. Suppose an application establishes multiple network connections. We need a mapping between these two levels so that service-level problems can be mapped to the appropriate network-level problems and effective management procedures can be applied.

We address the first two challenges while leaving the third open. Our approach is:

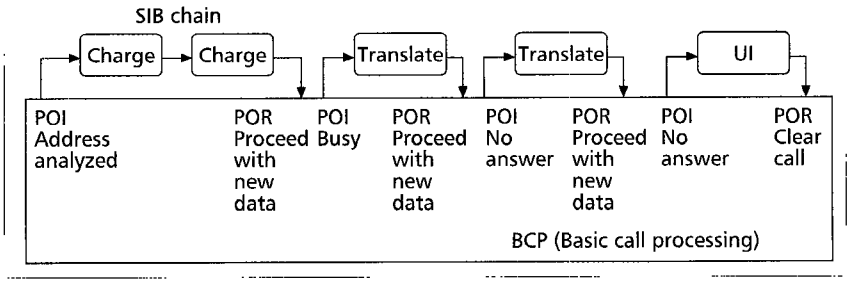
- To create and customize a service by combining service-independent building blocks (SIBs), a concept adopted from intelligent network (IN), into a SIB graph
- To have the service agent interpret the SIB graph

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 graph which represents the service logic is 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 article is organized as follows. The second section provides background on the service creation concept. The third section discusses design alternatives of SIBs. In the fourth section, we focus on the overall architecture and design issues of the broadband service operations environment. The fifth section gives two examples of broadband service creation. System implementation status is reported in the sixth section. In the last section, we describe our future work and conclude the article.

Service Creation

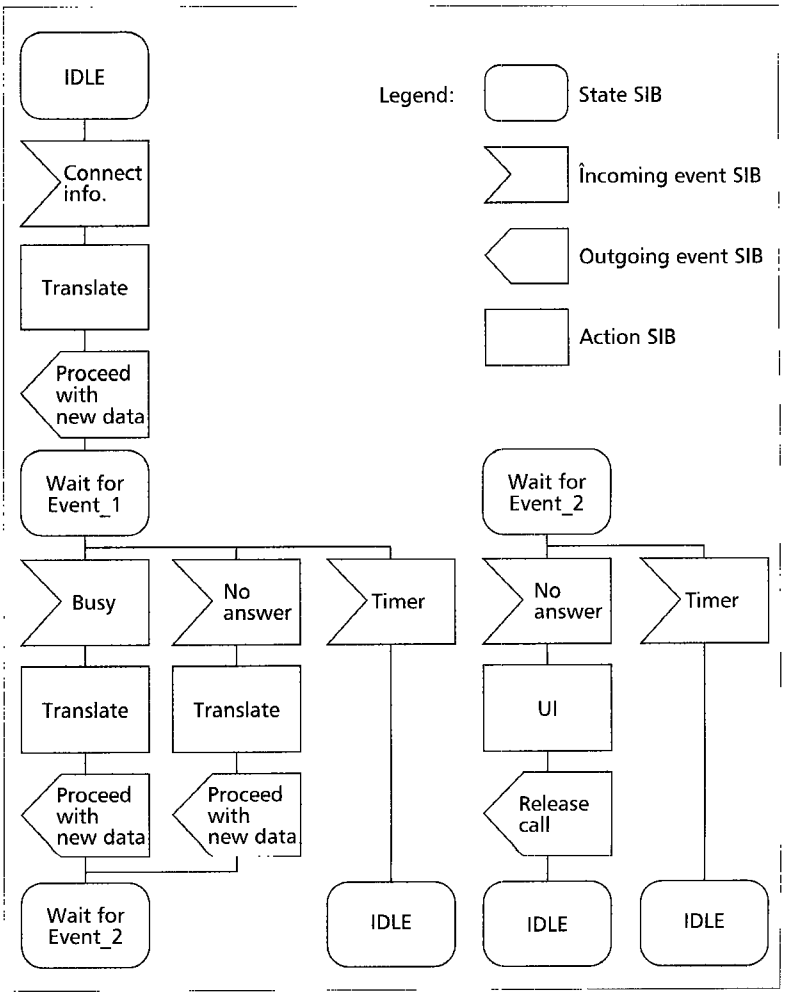
In an extremely competitive marketplace, telephone companies recognize that the speed at which new services are deployed is critical in meeting the needs of customers. Traditionally, the speed of deployment is constrained by the schedules of equipment vendors. Although the telephone companies can discover their customers' needs early, they need to report them to the equipment vendors and wait for their switch software development schedule, which may take years. This process cannot meet today's requirements of fast variations in customers' needs. It also costs 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 IN domain [1]. The environment is a powerful tool that allows users to quick-



■ **Figure 1.** *A SIB graph of call forwarding on busy.*

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/ no answer	

■ **Table 2.** *A set of benchmark CS1 services.*



■ **Figure 2.** *A SIB graph of call forwarding on busy with extended SIBs.*

Services	Service features
Video on demand (VoD)	Select/cancel Start Stop Pause Fast forward/reverse Scan forward/reverse Jump to different scenes Show contours
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 in video scenes Record Voice replay
Distance learning (DL)	Control camera Display documents Control what is displayed on monitor interactivity Schedule classes Browse Join/leave classes
TV listing (TL)	Select Display of distribution programs/schedules Progressive queries
Video conferencing (VC)	A real-time bidirectional exchange of audio/video/data information Announce/establish/control conference
Near VoD	Broadcast of video No real interaction/select channel

■ **Table 3.** *Sample broadband services.*

ly create and customize services without having to know all the details of the physical equipment.

IN Capability Set 1

In the IN conceptual model, 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.

To speed up the service creation process, the services are modularized into some basic building blocks, SIBs. As explained by the name "basic building block," these SIBs are service-independent and common to most services. They can be used to build different kinds 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 chain them together according to the service requirement and logic, which is usually called a *SIB chain*. The SIB chain will be attached to the basic call processing (BCP) SIB through the point of initialization (POI) and point of return (POR). The BCP SIB as defined in IN Capability Set 1 (CS1) [2] is a specialized SIB that provides the basic call capabilities. It enables the proper SIB chain based on the POI received, and results in the appropriate POR. A complete service logic, namely a *SIB graph*, is a combination of a BCP SIB and one or more SIB chains. For International Telecommunication Union–Telecommunication Standardization Sector (ITU–T) CS1 standardization, 13 SIBs are specified (excluding the BCP SIB). Nine POIs and six PORs are also defined for BCP SIB in the same document (Table 1). A SIB graph example of *call forwarding on busy* service is shown in Fig. 1. 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.

Problems with CS1 SIBs

Problems have been found in the SIB graph structure. In Fig. 1, 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 [3]. Such 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 networkwide 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 proposed in [3] is to decompose the BCP into a new set of SIBs, which includes:

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

With this new set of SIBs, we can clearly describe the interactions between different SIB chains. Thus, a complete working service can be described. The *call forwarding on busy* service described in this way is shown in Fig. 2. 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, eliminating the confusion in Fig. 1.

Broadband versus Baseband Service Creation

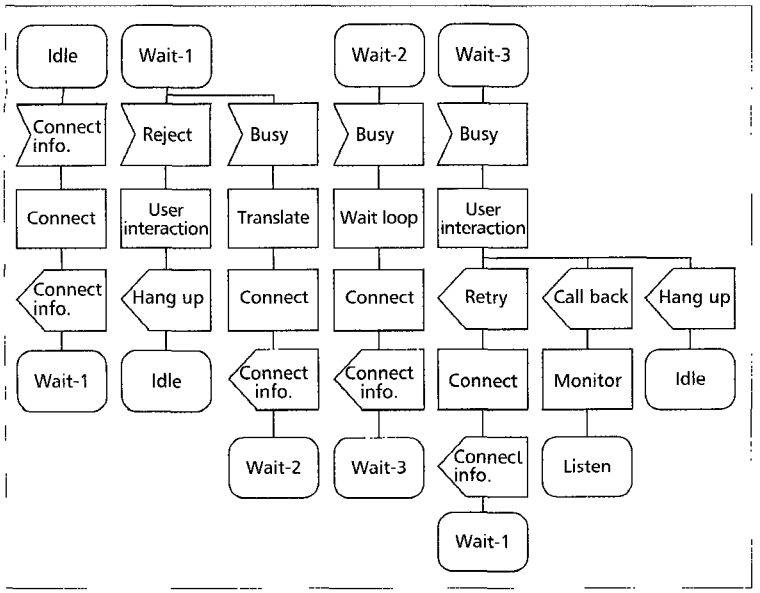
The IN services are baseband. No matter what services (Table 2) are provided, baseband services always establish the same type of connections. The connections may have the same bandwidth allocation and performance requirements. The differences lie in the added service features which change the basic service logic of a call.

Unlike baseband services, broadband services (Table 3) require high and variable bandwidth, and low latency. They generate multiple types of network traffic load. In addition, broadband services usually contain complex client-server interactions during the service time. Mapping SIBs from baseband to broadband, we need a new set of SIBs for 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, resources and performance should be carefully managed for effective utilization of the system.

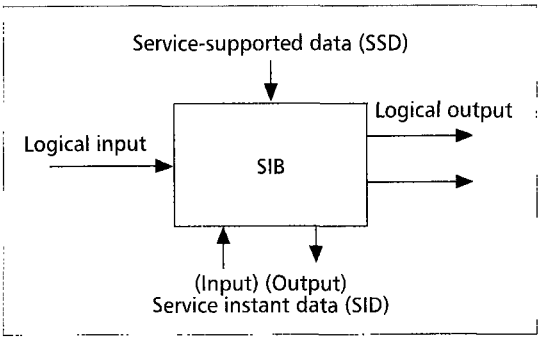
Service-Independent Building Blocks

Design Alternatives for Building Blocks

One of our objectives is to define the SIBs and implement an SCE for broadband services. There are two alternatives to define the SIBs, fine- and coarse-grained. In the fine-grained



■ **Figure 3.** *Another SIB graph of call forwarding on busy with extended*



■ **Figure 4.** *A graphic representation of a SIB.*

approach, the SIBs are more primitive and the service provider can have a detailed description of desired service logic such as handshaking in connection establishment. In the coarse-grained approach, SIBs are more powerful. Each SIB has one or a few predefined procedures; the service provider need not be concerned with detailed activities. These two approaches represent a trade-off between flexibility and friendliness.

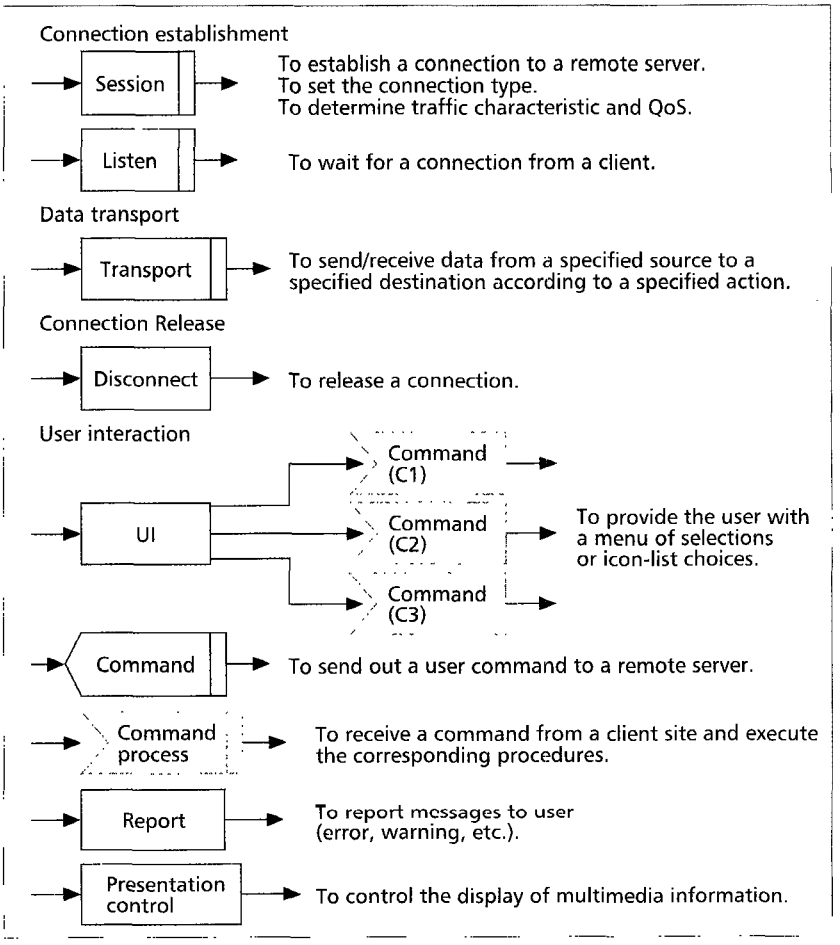
Figure 3 is an example of fine-grained connection establishment. The state SIBs *Wait-1*, *Wait-2*, and *Wait-3* are used to record the status during the process of call establishment. Hence, the proceeding action can be determined upon receiving a specific incoming event. Initially, an originator sends 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 retries.

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 need to pay more attention on transforming the service logic into a SIB graph during the process of service creation.

In cases where most of the services have the same connection establishment procedure, it is inefficient and unnecessary for each service provider to design a detailed SIB graph as in Fig. 3 in every service. For simplicity, some common service logic can be predefined 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 only a limited degree of flexibility can be provided. SIB parameters are used to support service logic description of the SIB. Those parameters determine the degree of flexibility. To retain full flexibility, a fine-grained approach should be kept such that a service creator may create their own macro SIBs if the available macro SIB cannot match their needs.

SIB Design

We have mentioned that parameters of a SIB are supplied for the description of broadband service characteristics. Service-supported data (SSD), provided at service creation time, and service instant data (SID), generated or received at runtime, are the main parameters. Each SIB is implemented as an independent (executable) object with logical inputs, logical outputs,



■ **Figure 5.** *A functional description of SIBs.*

SSD	1. Uni-/bidirection, symmetric/asymmetric 2. Traffic descriptor (two sets for bidirection/asymmetric) mean bandwidth, peak rate, burst length, direction 3. QoS (two sets for bidirection/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.

■ **Table 4.** *Session.*

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.

■ **Table 5.** *Listen.*

SSD	None
SID	1. Source address (IP address + port#) 2. Destination address (IP address + port#) 3. Data
Procedure	1. Send/receive: Data transfer

■ **Table 6.** *Transport.*

SSD	None
SID	1. Source address (IP address + port#) 2. Destination address (IP address + port#)
Procedure	1. Close: Release the connections

■ **Table 7.** *Disconnect.*

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

■ **Table 8.** *UI.*

SSD	None
SID	1. Command 2. Parameters
Procedure	1. Send

■ **Table 9.** *Command.*

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

■ **Table 10.** *Command process.*

SSD	None
SID	1. Messages
Procedure	1. User notification

■ **Table 11.** *Report.*

SSD	1. List of (data formats, data players)
SID	1. Data format 2. Data
Procedure	1. Call

■ **Table 12.** *Presentation control.*

SSD, and SID (Fig. 4). The SSD includes some fixed data associated with this SIB, some field pointers for dynamic links and information, and some related QoS parameters.

Considering a broadband service, the functionalities can be divided into four parts: connection establishment, data transport, connection release, and user interaction. In the coarse-grained approach, the corresponding SIBs, their symbols, and their functions are illustrated in Fig. 5.

SIB Parameters

According to the SIBs in Fig. 5, their SSD, SID, and procedures are listed in Tables 4–12. Various SSD entries can be filled at service creation time according to the requirements of various broadband services.

When a service such as videophone or video conferencing is invoked, its corresponding SIB graph will be transferred to the client/host who requests such a service. After that, it can create the connections directly. It is worth mentioning 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.

The SIB in Table 4 sets the connection type, traffic characteristics, and QoS. These parameters are can be predefined according to the service type. The service creator may define a set of SSDs such that users may choose their requirements. Multiple SSD-SID sets are allowed for multiple connections. (Note that a broadband service usually requires multiple connections for different types of data content.)

In Table 5 the connection type, traffic characteristics, and QoS are received from the network. An SSD is not necessary.

In Table 6, after connection establishment, the source and destination addresses are received from the session/listen SIB as input SID.

In Table 8 a command may be associated with permitted parameters to increase its flexibility.

In Table 9 it sends out the command accepted by the UI to the remote server.

In Table 10, the SIB contains a list of commands and corresponding procedures. Upon receiving a command from a client, it forks a process to handle the corresponding procedure.

In Table 12 it takes the data as parameters to call the corresponding data player according to the data format.

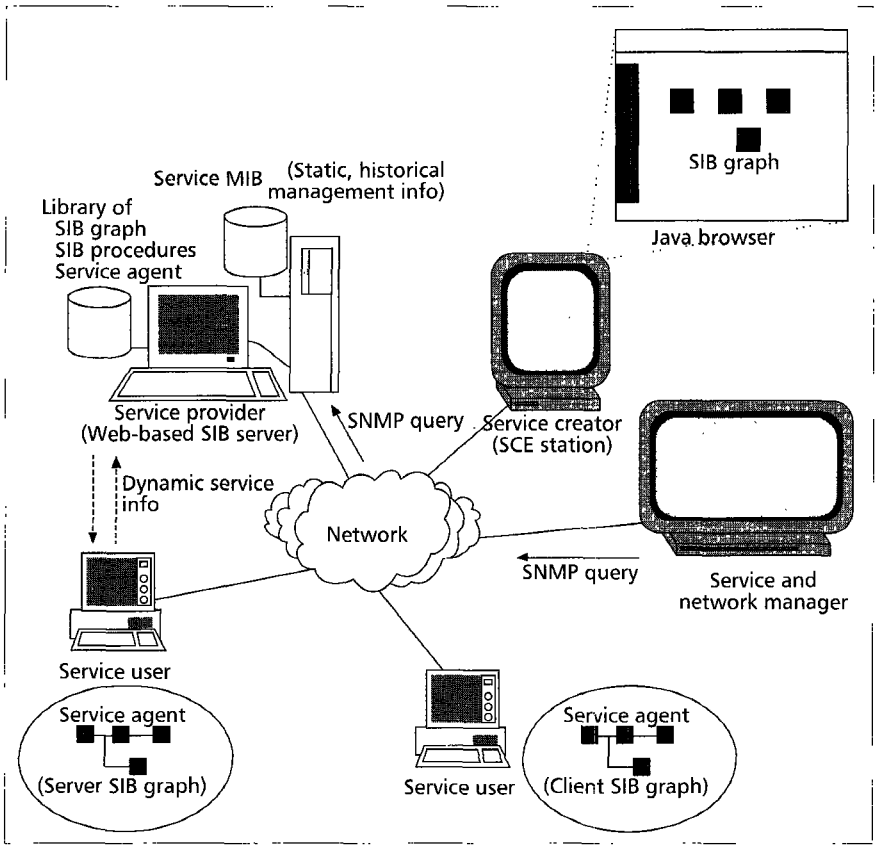
SERVICE OPERATIONS

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 Fig. 6. Different from the IN service creation concept, each service is now made up of two SIB graphs: one for client, one for server. Therefore, with the predefined 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:

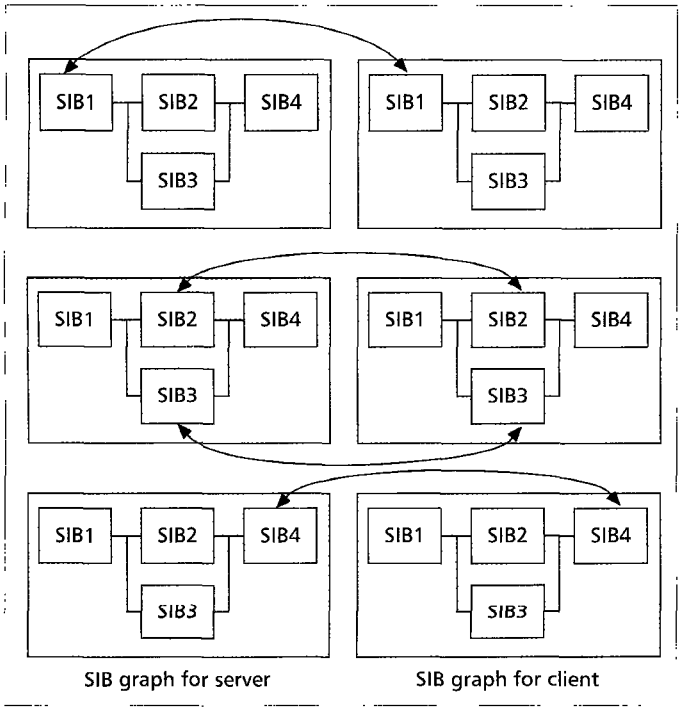
- SIB graphs for services provided
- SIB procedures associated with each SIB
- Service agent which interprets the SIB graph at a service user's site

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.



■ **Figure 6.** *Overall architecture.*

Each time a client requests a service, it invokes its Web browser to enter the Web home page of the service provider. On the home page, the access rights of the clients will be checked, and some readily created services will be displayed. After the client selects a specific service (e.g., VoD), the corresponding VoD SIB graph for the client will be transmitted to the client machine; also, the other half of the VoD SIB graph pair (the SIB graph for server) will be transmitted to one of the available video servers. Since now the SIB graphs are transmitted to the server and client, respectively, they can communicate with each other directly by making the corresponding SIB module active (Fig. 7). Hence, the overhead of services can be reduced to a minimum if the SIB modules can handle the physical components directly.



■ **Figure 7.** *The active SIBs in the SIB graphs directly communicate with the corresponding ones at each time instant.*

In order to interpret the SIB graph and run the service routines, the service user, either client 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 preinstalled before the service invocation time. This service agent and the code of SIBs can be implemented in two ways. One is to implement in Java code and transmit them to the service client or server site together with the SIB graph. This is similar to the diskless client where no installation is needed before the execution of SIB graphs in a workstation. The other way is to preinstall the codes in the workstations which may need to execute the SIB graphs. Performance may be improved by omitting the transmission of the SIB codes 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 the SIB graph, and reports the information about the service to the service provider. A service management information base (MIB) is required to keep track of and log the service provided by the service provider. The service manager may pass a Simple Network Management Protocol (SNMP) query to the service provider; if it queries static or historical management infor-

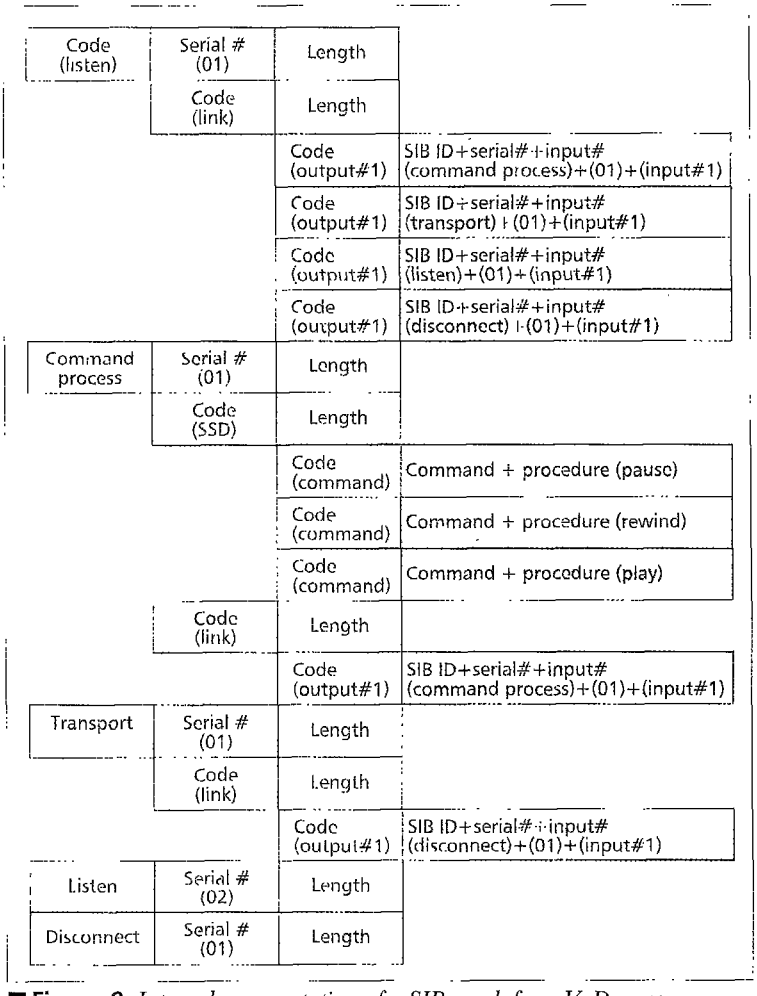


Figure 8. *Internal representation of a SIB graph for a VoD server.*

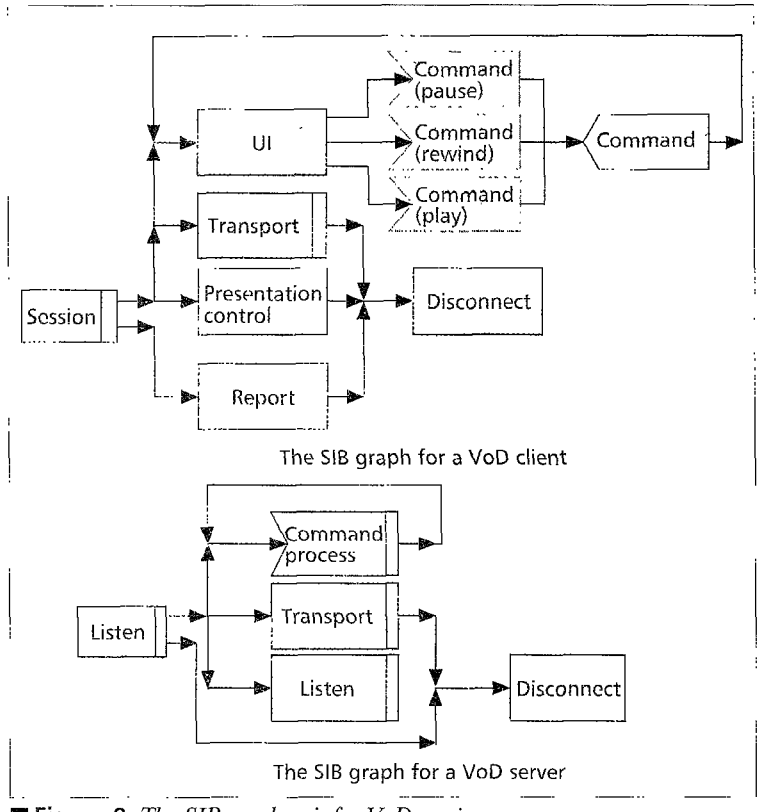


Figure 9. *The SIB graph pair for VoD services.*

mation (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 where dynamic information is reported.

Service Agent

When a client requests a service, a pair of SIB graphs for client and server are transmitted to the client and server, respectively. An internal representation of a SIB graph, instead of the logical view, 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 the involved 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 8 is an example of the internal representation of the SIB graph for the VoD server in Fig. 9. 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 #).

Examples

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

Video on Demand

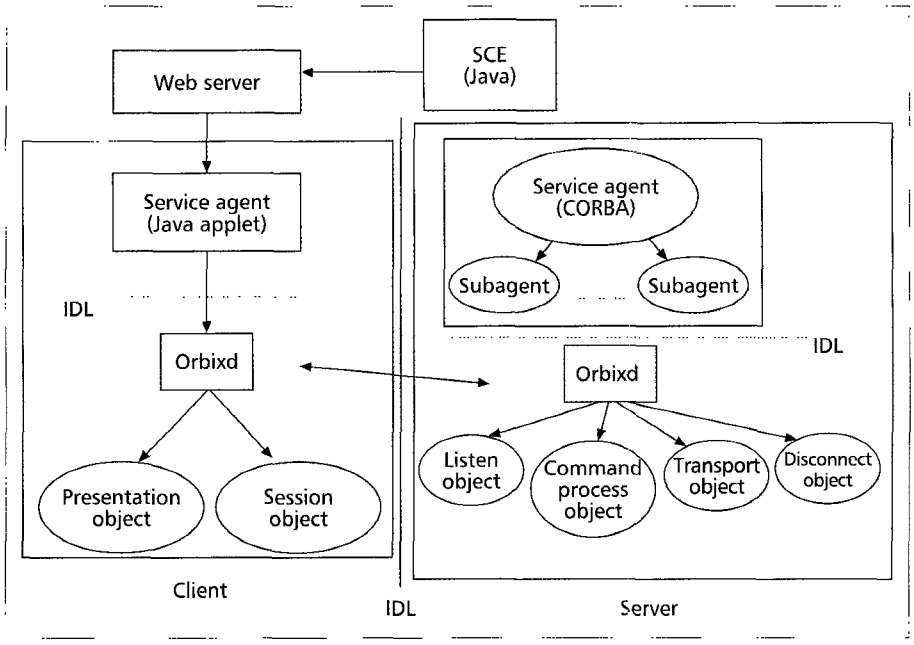
Figure 9 shows the SIB graph pair of VoD. At the client site, three processes (UI, transport, and presentation control) will be forked if the connections are established successfully; otherwise, an error message is reported to the user and the connections are then disconnected. The UI SIB accepts the 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 charging 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, Transport, and Listen) are forked at the server site when the connections are established successfully. The Command Process SIB accepts the user's command from the client, and then calls the corresponding procedures for the command. The Transport SIB transmits

multimedia information to the client, and the Listen SIB waits for other clients who may want to join the VoD session.

Video Conferencing

As in VoD, we assume that there is a video conference server, and that the chair of the conference initially set up the connections between his client site and a video conference server. The Listen SIB at the server site waits for the other members of the meeting to join. The multimedia information is distributed to all members of the video conference through the video conference server. The SIB video conferencing graph pair is almost the same as VoD. The differences are the command list (e.g., Talk, Vote) of the UI SIB and the type of connections (symmetric, bidirectional). These differences are encoded in SSD and SID values.



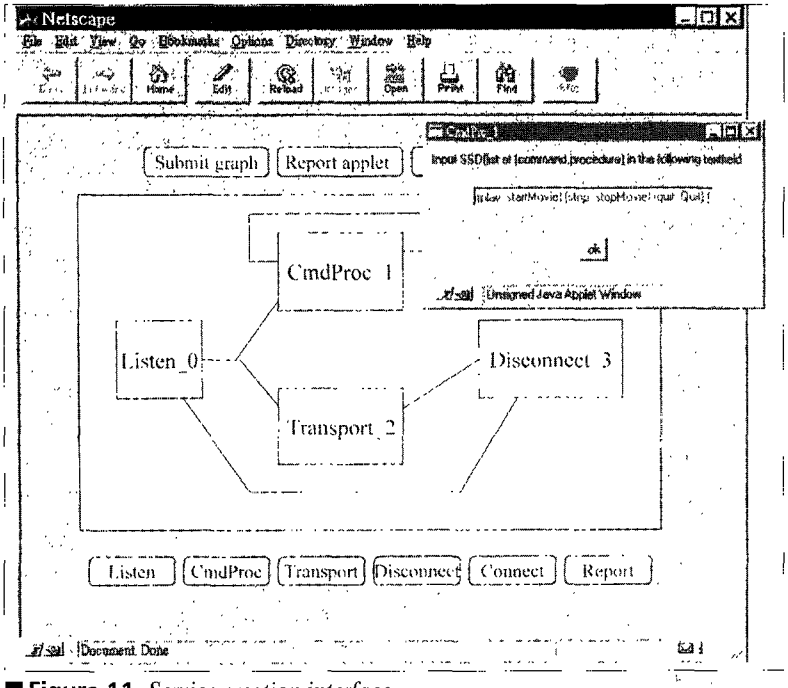
■ **Figure 10.** *Implementation architecture of service creation and operations.*

Implementation Status

Figure 10 illustrates our implementation architecture of the service creation and operations environment. This architecture is based on three technology components: World Wide Web, Java [5], and CORBA(Common Object Request Broker Architecture) [4].

The Web server provides the platform to create and store SIB graphs of various broadband services through a Java-coded SCE running on any Web browser. This Web server is also a service access point and service provider where clients with any Web browser can invoke the service presented in the Web server home pages. Upon service invocation, a Java-coded service agent and the service's corresponding SIB graph are downloaded and run in the client Web browser; at the same time, one of the available servers is selected to serve this client. SIBs, the modules that physically carry out the service tasks, are implemented as CORBA objects. Since the location transparency of CORBA objects is supported by a CORBA daemon, orbixd [6] in our case, it does not matter where these CORBA objects reside; but, for performance reasons, we put the objects dealing with screen presentation and session establishment at the client site while putting most objects at the server sites.

The system is now operational with the ability to create and run VOD services. Figures 11 and 12 are screen dumps of the service creation and operation interfaces. Our implemented SIB modules are now limited in functionality. To be able to create and operate a rich set of services, we need only enrich our SIBs to support more SSD and SID parameters. In summary, the overall architecture has been proven to be feasible and promising.

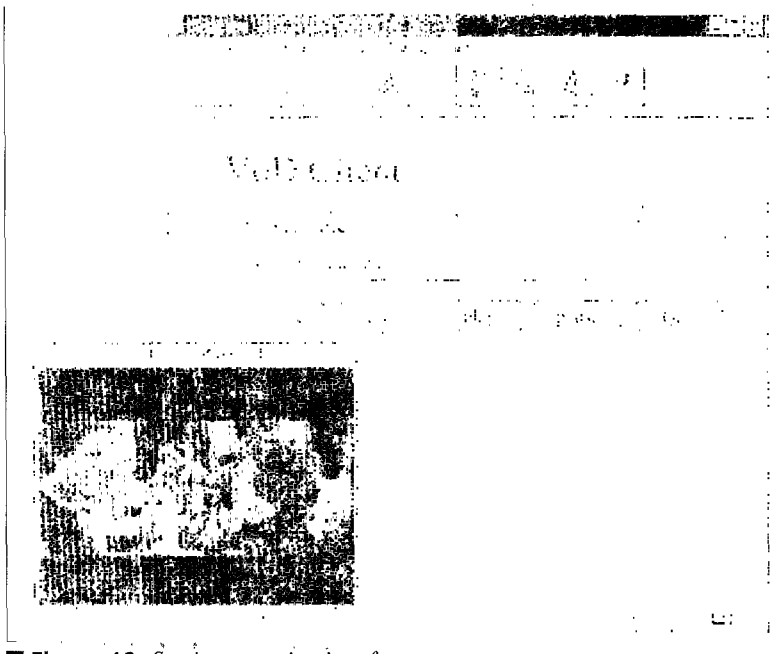


■ **Figure 11.** *Service creation interface.*

Conclusion and Future Work

An architecture of service creation and operation is presented in this article. Included in our discussion are design alternatives of SIBs, SIB design, service creation, and finally the operation of the created broadband services. There are also two examples, video conferencing and VoD, to show the feasibility of this architecture.

The client-server service creation and operation 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 devices installed on the client machine. In order to make the service running environment more transparent to platforms and physical devices, an interpreter-like environment, such as a Java browser, seems to be a solution. The main problem of using an interpreter-like terminal could be performance, for which in



■ **Figure 12.** *Service operation interface.*

our opinion can be greatly improved by an interpreter accelerator. SUN Microsystems has announced a licensable, configurable CPU core, picoJava, which provides direct execution of Java byte code and higher performance for embedded Java applications. microJava and UltraJava are coming processors based on the picoJava design. Therefore, possible future work for us is to evaluate the performance of Java-coded SIB modules, and use them as bases for service creation in the broadband network testbed. Recently, OMG (Object Management Group) has started to discuss the asynchronous message passing [7] issue. The performance problem of transporting multimedia stream-type data over CORBA is expected to be solved.

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 set up several connections at the same time and involve system resources other than the network resources. Therefore, integration of and mapping between these two are required to ensure proper service quality. The detailed design of the service MIB and the mappings between service management and network management are parts of our future work; that is, the third challenge.

References

[1] ITU Q-Series, "Intelligent Network Recommendations," Helsinki, Finland, Mar. 1993.
[2] ITU Q-1213, "Global Functional Plane for Intelligent Network Capability Set 1," Mar. 1993.
[3] R. Rieken *et al.*, "IN Service Creation — Problems of the CS1 SIBs and How to Solve Them," *Proc. Intelligent Network Tech. Conf. (INTC)*, Taipei, Taiwan, pap C2-1–11, 1995.
[4] OMG, "The Common Object Request Broker: Architecture and Specification," Dec. 1993.
[5] Sun Microsystems, Java, http://java.sun.com.
[6] IONA Technologies, http://www.iona.ie.
[7] OMG, "Control and Management of A/V Streams Request For Proposal," doc. telecom/96-08-01).

Biographies

YING-DAR LIN (ydlin@cis.nctu.edu.tw) received a Bachelor's degree in computer science and information engineering from National Taiwan University in 1988, and the M.S. and Ph.D. degrees in computer science from the University of California, Los Angeles in 1990 and 1993, respectively. He joined the faculty of the Department of Computer and Information Science at National Chiao Tung University in Taiwan in August 1993 and is now an associate professor. His research interests include high-speed LANs/MANs/WANs, traffic characterization, service and network management, and network centric computing.

YUH-TAY LIN (yeti@csie.nctu.edu.tw) received B.S. and M.S. degrees in computer science and information engineering from National Taiwan University in 1988 and 1990. He is currently a Ph.D. student in the Department of Computer Science and Information Engineering at National Chiao Tung University. His research interests include high-speed networks and network management.

PO-NING CHEN (poning@cc.nctu.edu.tw) received B.S. and M.S. degrees in electrical engineering from National Tsing-Hua University, Taiwan, in 1985 and 1987, and a Ph.D. degree from the University of Maryland, College Park, in 1994, respectively. Since 1996 he has been an associate professor with the Department of Communication Engineering at National Chiao Tung University, Taiwan. His research interests include information theory, statistical communications, pattern recognition, and network management.

MICHAEL M. CHOY (mmchoy@ccl.itri.org.tw) was with the Industrial Technology Research Institute at the time this article was written.