Communication Systems Supporting Multimedia Multi-user Applications

The development of a service description ensures that protocol designs actually produce the required functional behavior.

Geert J. Heijenk, Xinli Hou, and Ignas G. Niemegeers

Multimedia multi-user applications are becoming more and more important. Computer-based conferencing systems allow people to interact using several types of information. Computer supported cooperative work (CSCW) systems allow geographically distributed groups of people to work on the same project. Communication systems must evolve to support these applications.

Intensive research is underway on the design of protocols and protocol entities for future communication systems supporting multimedia multi-user applications. Assuming that B-ISDN will be the standard public network for providing multimedia multi-user services in the coming decade, a lot of research is based on B-ISDN. Examples of research are as follows: requirements for the user-network signaling and a conceptual call model [1]; a more sophisticated call model and a signaling protocol at the user network interface (UNI) [2, 3]; and service requirements for multimedia applications as well as connection management facilities for multiservice networks [4]. Also, project RACE MAGIC is studying the signaling protocol to support multimedia multi-user services [5]. Apart from B-ISDN, protocols are being designed to support multimedia multi-user applications, e.g., on Internet [6,7].

Work on multimedia multi-user communication systems is based more or less on intuitive feelings about the required capabilities. Because requirements are mostly specified in an informal and imprecise way, they are open to different interpretations, which can cause disagreement about the design choices. Ambiguity results in an intermingling of discussions concerning the required behavior and discussions concerning the way to achieve this behavior, which wastes R & D resources, i.e., manpower and money. Often, disagreements are detected only after progress has been made on the design of protocols or protocol entities. In this article, we develop a more explicit, precise description of the required functional behavior of communication systems supporting multimedia multi-user applications. This description, called a service description, can serve as a common reference for research and development.

The communication services supporting multimedia multi-user applications have been described [8-10]. The present work differs in its scope and its approach to the service description as discussed in the next section.

This article is organized as follows. First, we explain the approach to the description of a multimedia multi-user service. An example illustrates the use of the service description in the design of communication systems. Next, we present the basic requirements of multimedia and multi-user communications. A call model underlies and structures the service description. Finally, we describe the service in terms of service elements.

Approach

Our work on the design of protocols for multimedia multi-user communication systems is based on an abstract and unambiguous service description that structures and leads the design process. Rather than providing a comprehensive description of all aspects of the required behavior of the system, the intention is to provide a first step towards a complete service specification. Furthermore, we focus on those aspects of the service related to the establishment, modification, and termination of calls.

In the definition of the service description, the communication system, i.e., service provider, is regarded as a whole; no component parts and their mutual interactions are visible. The communications system provides services supporting multimedia multi-user applications. Users interact with the service provider at service access points (SAPs). We describe the service by specifying the service primitives, which are the interactions that can take place at the spatially distinct SAPs. Apart from the service primitives and their parameters, the possible sequences for invoking primitives at different SAPs must be described. The service description is at the highest level of abstraction. It does not

GEERT J. HEIJENK is a research staff member of the Center for Telematics and Information Technology at the University of Twente.

XINLI HOU is a research staff member of the Center for Telematics and Information Technology at the University of Twente.

IGNAS G. NIEMEGEERS is head of the Center for Telematics and Information Technology at the University of Twente.

prescribe how the interactions should take place, e.g., by means of function calls or message exchange across the UNI; this decision should be made during the further design of the system. Thus the required behavior of the system can be described without reference to possible implementations.

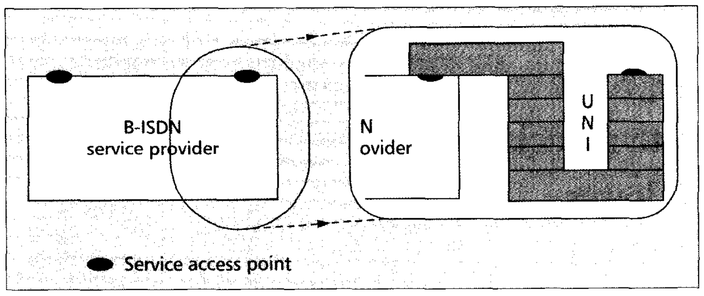
The present approach is in line with that employed in the development of the OSI reference model (OSI/RM) [11]. The OSI/RM was developed with only point-to-point data communications in mind, however, and the service defined here should also support multimedia and multi-user communications. In the OSI/RM, due to the simplicity of point-to-point data communications, there is no call model (or the equivalent). The complexity of multimedia and multi-user applications asks for a sophisticated call model. A confirmed service element in the OSI/RM is at most a three way negotiation among the calling user, the called user, and the service provider, hence, the one-phase confirmation composed of four types of primitives (request, indication, response, and confirm) is enough for such a simple service element. In case of multi-user communications, negotiation may have to be done with multiple called users. A service request on a call may have impact on the users who are not directly involved in the request, and these users should be informed of any change of the state of the call. To meet these requirements, the service elements of the OSI/RM must be extended as discussed in detail in the service description.

Our approach is similar to that of the Touring Machine [8]. Both provide an open platform supporting multimedia multi-user communications. But the points of view are different. The applications programming interface (API) of Touring Machine defines a set of messages passed between applications, and the infrastructure for requesting services from the system. It makes various network-provided capabilities and core services available to a large class of multimedia applications. Our service description merely defines the needed capabilities of communication systems to support multimedia multi-user applications.

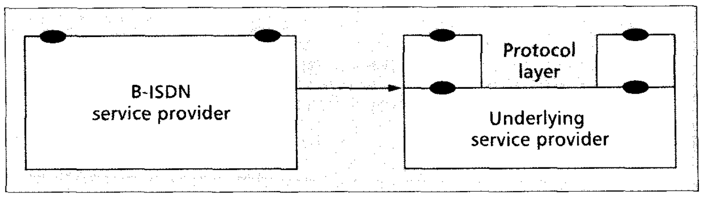
Another work follows an approach similar to ours but has a different scope [9]. In our terminology, it is a multimedia multi-user application that includes not only the capabilities of the communication system but also the functions of application processes. We restrict our service description to the capability of the communication systems to support multimedia multi-user applications.

Another approach describes services at the UNI of the B-ISDN, i.e., the access point to the public network [10]. We do not follow this approach because an abstract service description must be independent of the various technical solutions used in various places, interfaces, or reference points.

Our service description is at a high level of abstraction without reference to resources in any part of the network, e.g., the UNI, and without reference to protocols to be adopted at any interface. It can be used as a starting point for the design of various (tele-)communication systems that support multimedia multi-user applications, e.g., the design of a broadband customer premises network (CPN), in which no distinction is made between protocol stacks for signalling and for user data transfer. There is no need to define a UNI protocol in a CPN.



■ **Figure 1.** *Design of a distributed UNI.*



■ **Figure 2.** *Decomposition of the B-ISDN service provider.*

The service description can be used in the design of public telecommunication networks. Let us illustrate its applicability to the design of the B-ISDN, e.g., as a starting point for the design and implementation of the UNI signaling protocol. This UNI is in fact a distributed implementation of the SAP (Fig. 1). In present public telecommunication systems, this distributed implementation is usually done for security, network integrity, and network protection. The interactions at the SAP, i.e., the service primitives defined in the service description, should be implemented through exchange of packets across the UNI. These interactions form the basic requirements for communication in the distributed UNI. Additional requirements are imposed by the way the service user and the service provider are implemented.

The service description also serves as the starting point for the design of protocols internal to the B-ISDN. Ideally the B-ISDN service provider should be repeatedly decomposed into a protocol layer and an underlying service provider (Fig. 2). This decomposition entails requirements in addition to the functional requirements of the service description. Those requirements, e.g., performance requirements, are not be provided in this initial service description.

A layered system is obtained by repeatedly decomposing the B-ISDN service provider into a protocol layer and an underlying service provider. The functionality for the bottom three layers can be covered by the B-ISDN Physical Layer protocol, the ATM protocol, and the ATM adaptation layer (AAL) protocols being specified by the CCITT [12]. Our service extends the AAL service with multimedia and multi-user communication facilities. The service includes the functionality needed for the establishment, modification, and termination of calls.

Decomposition can be done vertically as well. One can design a multimedia multi-user communication system according to the B-ISDN protocol reference model. Service primitives should be grouped into categories for this purpose. One category of service primitives relates to the estab-

Together with models of lower levels of abstraction, the call model can structure the protocols to be designed and the state information to be maintained by these protocols.

lishment and control of calls and connections; these service primitives describe interactions with the control plane. The other category contains the service primitives related to the exchange of user data; these describe interactions with the user plane. Our service description deals with the first category of service primitives, so it only gives the requirements on the protocols in the control plane.

Horizontal and vertical decomposition introduces new requirements such as coordination between layers and between user and control planes. These requirements will be satisfied by functions that reside in the management plane of the B-ISDN protocol reference model.

Requirements of Multimedia and Multi-user Applications

The objective is to define a communication service usable as a basic and general facility for a wide variety of applications, including multimedia and multi-user applications. The service should support existing and future applications and contain the functionality common to a large number of possible applications. This service should be easy to extend and modify without affecting the general concepts, making it future proof to some extent.

The different requirements and treatments of information make it advantageous to treat the various information types as separate components, e.g., different resources may be used by the communication network to support different information streams. Another advantage is the avoidance of a large number of different standards for composite signals [1]. Further, the use of functions specific for each component allows the application to be tailored as needed and allows the user to manipulate and use each component independently. Last but not least, the use of separate components provides open-endedness. One cannot precisely predict what applications will be developed and what functionality they will require. By introducing new components the service can be extended to support the new demands. In this article, a component is called medium.

Although the different media of a multimedia application can be treated differently, there is some relation between the various media, the time-relation between video and voice media in a video telephone application. This time-relation creates a problem for the communication network when the various media are treated independently. The communication network must provide some means to synchronize the various information streams.

The requirements to support multimedia multi-user applications are summarized as follows:

- A call can involve multiple media and multiple users.
- Relationships can exist between users and media, expressing the transmission and/or reception capabilities of a certain user for a certain medium. A capability need not imply that the user is actually sending or receiving on that medium.
- The presentation of the information of a medium can be individually controlled for each user, e.g., each user in a video conference can have his own view of the other users. Also, users can access the medium using different coding standards, which could require conversion functions in the network.
- Relationships can exist between media in a multimedia multi-user application, e.g., an audio and a video channel may have to be synchronized.
- Modification of all the above should be allowed, e.g., users may be added and removed dynamically, subject to certain constraints.
- Individual users can receive different permissions to modify the call, and these permissions can be dynamically changed during the call. Only users with the appropriate permissions are allowed to modify the call.

In the following section, a call model which underlies the service description will be presented. This call model describes the elementary constituents of a call, what a call can look like at a certain moment, and how it may change.

Call Model

The complexity of multimedia multi-user communication systems necessitates the use of models to create an abstract view of the aspects that are relevant in the design of such systems. In this section, a call model structures the services provided by the communication system. It can also be used in the design of the system. Together with models of lower levels of abstraction, the call model can structure the protocols to be designed and the state information to be maintained by these protocols. The call model is developed following an object-oriented approach in which objects are defined to describe different aspects of a call.

A call can be defined as a dynamic association between service users and the service provider, whereby the service provider offers the functions of one or more (possibly related) media to the service users. From this definition, the medium is the basic constituent of a call. A medium can be considered as a part of a service, providing a communication capability for a single information type. A medium is characterized by the type of its information and the functions that operate on that information. Examples of information types are voice, audio, text, data, video, and graphics. The main function (operating on the information) is the transport of information. Other functions are access control, representation of the information, mixing of information from different users, and distribution of information to multiple users.

All interactions between users and the service provider concerned with one sort of information are covered by the medium. Besides the exchange of data, there are some related interactions that organize and manage the data exchange.

Examples of media are a voice medium which mixes the voice of all users, a voice medium where only one user may speak at any moment, and a data medium for data transfer.

The type of information is determined for a certain medium. However, this does not imply that all users have to use the same representation for the information of this medium. Examples of representations are ADPCM or PCM for voice information, and Postscript or JPEG for graphics information. In the call model, each user has his own representation of the information of a medium. Mapping between information and the user's representation can be done by the communication system or the user. Negotiation of such a mapping is a local matter at the concerned user's SAP, and hence it is not reflected in the call model or the service description.

More than one medium can be offered in one call, and there can be a correlation between media. When a user transmits the voice and the video image of a person by two different media, the receiving users want to receive the information lip-synchronized, so synchronization between information streams has to be provided by the service provider. The time relation has to be preserved although the information may be merged with other information. To describe such a relationship, an object called "relation" is defined in the call model. This object specifies which media are mutually related by what relationship.

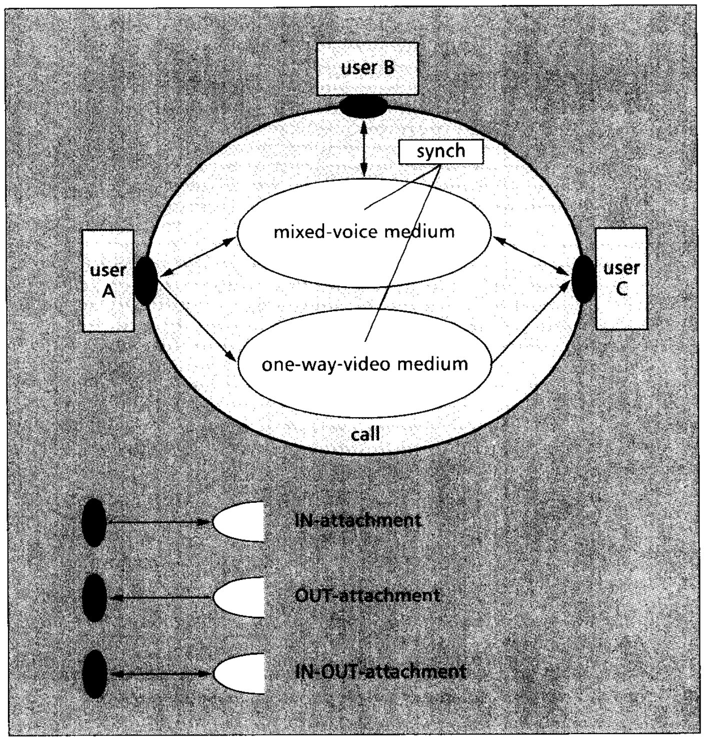
An attachment is an object in our call model that can relate a user to a certain medium. Users can be attached to a medium in three different ways: IN, OUT, and IN-OUT. IN refers to the ability to transmit information to the medium. OUT refers to the ability to receive information from the medium. IN-OUT refers to both abilities. An ability does not imply that information is always exchanged. The decision whether or not to exchange information at a certain moment is up to the user and the service provider. As an example, the service provider may allow only input from one of the IN-attached users at a time, e.g., a voice medium of a conference which selects the loudest voice.

Some objects in the call model are dependent in the sense that the existence of their instantiations depends on the existence of instantiations of other objects. Relation is a dependent object. It exists only when two or more media are related by it. Medium itself is also a dependent object. An instantiation of medium exists only if at least one user is attached to it.

Take a multimedia conference call as an example (Fig. 3). Three users are involved in the call. There is one mixed-voice medium, to which all three users are attached (IN-OUT). There is a one-way-video medium, to which only two of the users are attached (one IN and one OUT). The requirement for synchronizing both media is expressed by a synchronization (synch) relation.

A call is a dynamic association between the service provider and service users. Users may be added to or removed from the call. Media may be added to or removed from the call. The representation of information to users may change. Users may be attached to or detached from a medium. To enable the users to control the dynamic behavior of a call, some modification rights should be given to them. There are several possibilities to do this, varying between two extremes: centralized control and distributed control.

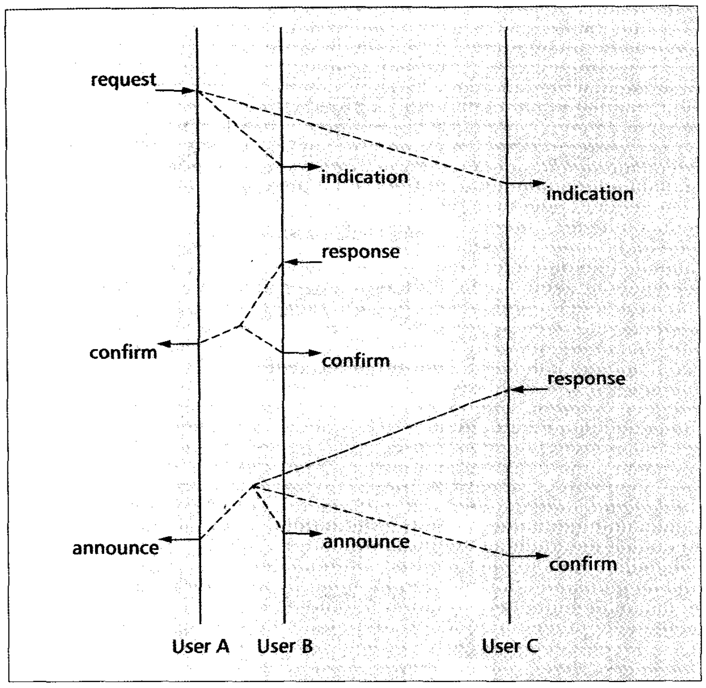
Centralized control refers to a situation where one user has the sole right to add and remove users and media, to specify synchronization between media, and to attach and detach users to media. Distributed control refers to a situation where all users have the above mentioned rights. Some kind of hybrid control is also possible. In the service described in this paper, any user can request for addition of



■ **Figure 3.** *A multimedia conference call.*

new media or new users to the call, or for the attachment of users to media. However, one user, called Owner, is always involved in the negotiation that results from a request. He always has the right to refuse the changes. Furthermore, he is the only user who can explicitly remove other users or media from the call, detach users from media, or terminate the call. Any user can detach himself from a medium or remove himself from the call. Initially, the user who establishes the call will be the Owner of the call. However, the Owner can transfer ownership to another user, provided both that user and the service provider agree. The decision, to what extent the control is distributed, will be subject to debate. The choice made here is based on the assumption that the Owner will pay for a certain call, and he therefore wants to control the facilities.

Although the call model presented here has similarities with those in the literature, there are differences. The EXPANSE call model reflects the local view of each user within the call [2]. Each user may have a different view on the call. There is no explicit global view of the call as presented in our call model. We consider the communication system as an unpartitioned service provider, while functions in EXPANSE are already distributed over local exchanges and terminal equipments. Our call model also differs from that of CMAP [3]. In CMAP, connections instead of media are embedded in the call model. The connection in CMAP is not at the same level of abstraction as the medium in our call model. Connections are needed to support media but they do not cover all aspects of these media. In fact, a connection only specifies those aspects of a medium that are related to the physical resources for the information transport.



■ **Figure 4.** *Temporal ordering of service primitives in a confirmed SE.*

It does not specify the higher layer functions such as intermedia synchronization and presentation.

Service Description

We follow an approach similar to that used in the development of the OSI/RM. The service is fully specified in terms of service elements (SEs), which refer to self-contained parts of the service, and their temporal ordering constraints. An SE is composed of several service primitives (including parameters) with a specific temporal ordering. In general, completion of the execution of an SE causes the call to transit to another well-defined state, i.e., objects are added, modified, or removed by the successful execution of an SE. An SE is executed as an extended transaction, i.e., unlike a standard transaction (which succeeds only if all operations in the transaction succeed), an SE may succeed if a specified part of the operations is successful.

Two types of generic SEs can be identified in our service. First we introduce these generic SEs, which are extensions of the generic SEs used in the OSI/RM. Then, SEs related to the establishment, modification, and release of a call will be specified in detail.

Generic Service Elements

Execution of SEs related to the establishment, modification, and release of a call results in the change of the state of the call. A user who initiates an SE is referred to as the initiating user (of the SE). Changing the state of a call may have to be negotiated among the initiating user, service provider, and other users who are referred to as involved users. Such a negotiation is necessary in some SEs, e.g., if a user is added to a call then negotiation has to happen among the owner, service provider, and the user to be added. Negotiation is not always needed, e.g., if a user wants to quit a call. SEs whose execution needs negotiation are called confirmed SEs. SEs whose execution does not need negotiation are called unconfirmed SEs.

A confirmed SE is initiated upon receiving a service primitive of the type "request" from the initiating user. The request primitive has a parameter called "success condition," which specifies the condition for successfully executing the SE. Other parameters specify, e.g., the requested changes to the state of the call.

Upon receiving a request, the service provider analyzes the requested changes to the state of the call, to see to what extent it can provide them. The result is a new proposal for the change of the state of the call. The new proposal should not contradict the requested changes but it might be different from them. It is forwarded to all involved users by indication primitives. Later on the involved users send a response to the service provider. The response primitive has a success condition parameter, which specifies to what extent the involved user accepts the SE. This parameter could indicate that the involved user is not willing to accept the SE.

As soon as the service provider finds out that the execution of the SE may succeed, the service provider sends a confirmation to the initiating user and all the involved users who have responded. The parameters of the confirmation primitive describe the outcome of the negotiation.

The service provider may still receive responses from some of the involved users after the SE succeeded. In such a case, the service provider sends back a confirmation that describes the outcome of the negotiation and the latest changes to the call. An announce primitive is sent to all users to whom a positive confirmation had already been sent out. The announce primitive has parameters describing the latest change of the call.

If the service provider finds out that the SE fails or if the SE does not succeed within a certain time- out period, then it sends a (negative) confirmation to the initiating user and all users who were sent an indication. The parameters of a negative confirmation may also indicate the reason for failure.

Figure 4 shows a typical sequence of the service primitives that are needed to successfully commit a confirmed SE where the response of one of the involved users is delayed. The success condition in the request is assumed to be such that a positive response from any one of the involved users is sufficient for committing the execution of the SE.

An unconfirmed SE can be initiated by a request primitive from the initiating user, or it can be initiated by the service provider. In the latter case, the SE does not have a request primitive.

Execution of an SE may impact users who are not involved in the SE. These users are referred to as affected users. Affected users must be notified of any change of the state of a call. During the execution of a confirmed or unconfirmed SE, e.g., when the execution of an SE is committed, the service provider may decide to send a notification primitive to all affected users. In an unconfirmed SE, all affected users receive a notification primitive that informs them of the latest change of the call.

CallReq	(Call_Id , Medium_List : { Medium_Id , Medium_Type } , [Media_Relation_List : { Relation_Id , Relation_Type , Medium_List : (Medium_Id , { Medium_Id }) }] , User_List : { User_SAP_Address , [Attachment_List : { Medium_Id , Attachment }] } , Success_Condition)
CallInd	(Call_Id , Owner_SAP_Address , Medium_List : { Medium_Id , Medium_Type } , [Media_Relation_List : { Relation_Id , Relation_Type , Medium_List : (Medium_Id , { Medium_Id }) }] , User_List : { User_SAP_Address , [Attachment_List : { Medium_Id , Attachment }] })
(CallRsp	(Call_Id , [Attachment_List : { Medium_Id , Attachment }] , Success_Condition)
(CallCnf	(Call_Id , [Medium_List : { Medium_Id } , [Media_Relation_List : { Relation_Id , Medium_List : (Medium_Id , { Medium_Id }) }] , User_List : { User_SAP_Address , [Attachment_List : { Medium_Id , Attachment }] }])
(CallAnn	(Call_Id , User_List : { User_SAP_Address , [Attachment_List : { Medium_Id , Attachment }] })
The following notational convention will be used for describing the parameters of the service primitives: ( ) means exactly one; [ ] means zero or one; { } means one or more; a comma (,) means followed by; and a colon (:) means consisting of.	

■ **Table 1.** *Call establishment SE.*

■ ■ ■ ■ ■

An SE is executed as an extended transaction, i.e., unlike a standard transaction, an SE may succeed if a specified part of the operations is successful.

Call Establishment SE

The SE to establish a call is a confirmed one. Five service primitives have been defined for the Call Establishment SE (Table 1). Since this is the first SE invoked during the lifetime of a call, there are no affected users and no notification primitive needs to be defined.

A service user can initiate the creation of a call (with one or more other service users, using one or more media) by issuing a Call Request (CallReq) primitive. The call can be identified by the user by referring to the Call_Id. The Medium_Type is specified for all requested media (Medium_List), which are identified with a Medium_Id. The initiating user can request to relate groups of media by means of a Media_Relation_List, which contains groups of media that have to be related (Media_List), the type of relationship that has to be maintained (Relation_Type), and an identifier (Relation_Id). The User_List indicates who can participate in the call. For all users (including the initiating user) who are identified by a User_SAP_Address, an Attachment_List can be used to propose the attachment of the user to one or more media. For each listed medium, the identifier and the type of Attachment (IN/OUT/IN-OUT) is given. The Success_Condition specifies conditions for successful establishment of the call, e.g., that at least one response is positive, or that all responses should be positive. The Success_Condition can also give some persistent conditions on the existence of the call, i.e., if these persistent conditions are no longer met (e.g., because one of the users leaves the call), the service provider automatically initiates the release of the call. We already assumed some implicit persistent condition by specifying dependencies between objects, e.g., a call should always involve at least one user attached to one or more media in order to have a meaningful call.

If the service provider is able to support the requested call, a Call Indication (CallInd) primitive will be issued at the SAPs of the invited users listed in the User_List of the Call Request primitive (except for the initiating user, who is also listed). The Call_Id, Medium_List, Media_Relation_List, and User_List are also parameters of this primitive. The Call_Id is local to the SAP, i.e., it is unique for a certain call at a certain SAP; a call may have different Call_Id values at different SAPs. As long as the success_condition in the request primitive is not violated, the service provider may degrade the contents of the Medium_List, Media_Relation_List, and User_List with regard to the request primitive, e.g., if not all media can be supported. The Owner_SAP_Address is also a parameter of the Call Indication primitive, to inform the invited users about the initiator of the call.

The Call Response (CallRsp) primitive indicates to what extent the invited user is willing to participate in the call, and to what extent he wants to attach to the media specified in the Attachment_List parameter. The invited user can put constraints on his participation by means of the Success_Condition parameter. He can also use this parameter to indicate that he is not willing to participate (negative response).

Sooner or later a Call Confirm (CallCnf) primitive is issued at the SAPs of the initiating and invited users, either to confirm the successful establishment of the call or to confirm that the call cannot be established. The contents of the Medium_List, Media_Relation_List, and User_List may again be degraded for several reasons. First, the User_List may not contain all the invited users, if some users have not yet responded or have responded negatively. Second, a participating user may have degraded his attachment to a certain medium in his response primitive. Finally, the success conditions of the request and the responses may have forced the service provider to degrade some of the parameters. After receiving a positive Call Confirm primitive, a user should consider the call established.

Additional media can be added to a call using the Add Media SE. Six service primitives have been defined for this confirmed SE.

AddUsrReq	(Call_Id , User_List : { User_SAP_Address , [Attachment_List : { Medium_Id , Attachment }] } , Success_Condition)
AddUsrInd	(Call_Id , User_List : { User_SAP_Address , [Attachment_List : { Medium_Id , Attachment }] })
AddUsrRsp	(Call_Id , Success_Condition)
AddUsrCnf	(Call_Id , [User_List : { User_SAP_Address , [Attachment_List : { Medium_Id , Attachment }] }])
AddUsrAnn	(Call_Id , User_List : { User_SAP_Address , [Attachment_List : { Medium_Id , Attachment }] })
AddUsrNtf	(Call_Id , User_List : { User_SAP_Address , [Attachment_List : { Medium_Id , Attachment }] })

■ **Table 2.** *Add users SE.*

Responses received after call establishment also result in a Call Confirm primitive to indicate the success or failure of the call to the concerned users. Users already in the call are informed of any additional users by means of a Call Announce (CallAnn) primitive.

Add Users SE

The Add Users SE enables the invitation of additional users to an existing call. It is a confirmed SE and it can be initiated by the owner or a non-owner. Six service primitives have been defined for this SE (Table 2). At the SAPs of the new users, primitives of the Call Establishment SE will be issued, since these users do not have to notice any difference between being invited into a new or an existing call.

An Add Users Request (AddUsrReq) primitive can be used by any participant in a call to ask for the addition of one or more new users to the call. The primitive results in a Call Indication primitive at the SAPs of the users to be added. The User_List parameter of the Call Indication is based on the corresponding parameter in the Add Users Request. Other parameters can be derived from the current state of the call. If the initiator of the SE is a non-owner, the owner of the call will receive an Add Users Indication (Add UsrInd) primitive. By means of the Success_Condition parameter of the Add Users Response (AddUsrRsp) primitive, the owner can express to what extent he agrees with the addition of new users to the call. The service provider evaluates this condition together with the parameters of the Call Response primitives of the invited users, to determine which users are added and how they are attached to the media of the call. The result is expressed to the owner and the initiating user of the Add Users SE by means of an Add Users Confirm (AddUsrCnf) primitive. The other users already in the call will be informed by means of an Add Users Notify (AddUsrNtf) primitive, and the new users in the call receive a Call Confirm primitive. Invited users whose responses are delayed also receive a Call Confirm primitive after they have responded, and they cause the invocation of a Call Announce primitive to the new users who had already been confirmed, an

AddMediaReq	(Call_Id , Medium_List : { Medium_Id , Medium_Type } , [Media_Relation_List : { Relation_Id , [Relation_Type] , Medium_List : { Medium_Id } }] , User_List : { User_SAP_Address , Attachment_List : { Medium_Id , Attachment } } , Success_Condition)
AddMediaInd	(Call_Id , Medium_List : { Medium_Id , Medium_Type } , [Media_Relation_List : { Relation_Id , [Relation_Type] , Medium_List : { Medium_Id } }] , User_List : { User_SAP_Address , Attachment_List : { Medium_Id , Attachment } })
AddMediaRsp	(Call_Id , [Attachment_List : { Medium_Id , Attachment }] , Success_Condition)
AddMediaCnf	(Call_Id , [Medium_List : { Medium_Id } , [Media_Relation_List : { Relation_Id , Medium_List : { Medium_Id } }] , User_List : { User_SAP_Address , Attachment_List : { Medium_Id , Attachment } }])
AddMediaAnn	(Call_Id , User_List : { User_SAP_Address , Attachment_List : { Medium_Id , Attachment } })
AddMediaNtf	(Call_Id , Medium_List : { Medium_Id , Medium_Type } , [Media_Relation_List : { Relation_Id , [Relation_Type] , Medium_List : { Medium_Id } }] , User_List : User_SAP_Address , Attachment_List : { Medium_Id , Attachment } })

■ **Table 3.** *Add media SE.*

AttMediaReq	(Call_Id , User_List : { User_SAP_Address , Attachment_List : { Medium_Id , Attachment } } Success_Condition)
AttMediaInd	(Call_Id , User_List : { User_SAP_Address , Attachment_List : { Medium_Id , Attachment } })
AttMediaRsp	(Call_Id , [Attachment_List : { Medium_Id , Attachment }] , Success_Condition)
AttMediaCnf	(Call_Id , [User_List : { User_SAP_Address , Attachment_List : { Medium_Id , Attachment } }])
AttMediaAnn	(Call_Id , User_List : { User_SAP_Address , Attachment_List : { Medium_Id , Attachment } })
AttMediaNtf	(Call_Id , User_List : { User_SAP_Address , Attachment_List : { Medium_Id , Attachment } })

■ **Table 4.** *Attach media SE.*

DetMediaReq	(Call_Id , User_List : { User_SAP_Address , [Attachment_List : { Medium_Id , [Attachment] }] })
DetMediaNtf	(Call_id , User_List : { User_SAP_Address , [Attachment_List : { Medium_Id , [Attachment] }] })

■ **Table 5.** *Detach media SE.*

Add Users Announce (AddUsrAnn) primitive to the owner and the initiating user, and an Add Users Notification primitive to the users already in the call.

Add Media SE

Additional media can be added to a call using the Add Media SE. Six service primitives have been defined for this confirmed SE (Table 3).

Any user in the call can initiate the SE by issuing an Add Media Request (AddMediaReq) primitive and specifying the media to be added (Medium_List), possible relations between media (Media_Relation_List), and users to be attached to the new media (User_List). In the Media_Relation_List, the initiating user can ask to relate two or more media to each other (including at most one existing medium); or to relate one or more new media to an existing relation. A user is not allowed to ask the service provider to relate existing media to each other, since it is probably impossible for the service provider to realize a relation between existing media.

An Add Media Indication (AddMediaInd) primitive is issued to all users who are requested to be attached by the initiator, and to the owner if he is not the initiating user of this SE. Apart from the Success_Condition that is specified with the request, the parameters of the Add Media Indication are the same as those of the request primitive, although the service provider may degrade parameters, e.g., if he is not able to provide all media. All users that received an indication, can respond with an Add Media Response (AddMediaRsp) indicating in its parameters to what extent they want to attach to the new media or, in case of the owner, indicating to what extent the addition should succeed. All users that have responded and the initiating user are informed of the success or failure of the media addition by means of an Add Media Confirm (AddMediaCnf) primitive. The affected users (those who were not requested by the initiator to attach to the new media) are informed by means of an Add Media Notify (AddMediaNtf) primitive. Users who respond later are also informed by means of an Add Media Confirm primitive. If these users want to be attached to the new media, the users who had already responded and the initiating user are informed of this by means of an Add Media Announce (AddMediaAnn) primitive, while the affected users get another Add Media Notify primitive.

Attach Media SE

To enable communications, users need to be attached to a medium. With the Attach Media SE, a user can ask to be attached to media, invite other users to attach, or upgrade attachments (i.e., change from either IN or OUT to IN-OUT). We have defined six service primitives for this confirmed SE (Table 4).

An Attach Media Request (AttMediaReq) primitive results in an Attach Media Indication (AttMediaInd) primitive to be issued at the SAPs of all users invited to attach (including the SAP of the owner if he is not the initiating user of the SE). Based on the received Attach Media Response (AttMediaRsp) primitives, Attach Media Confirm (AttMediaCnf) primitives are issued at the SAPs of all responding users and the initiating user. Attach Media Notify (AttMediaNtf) primitives are issued at the SAPs of all affected users, i.e., those who were not invited to attach to the media. Delayed responses are also confirmed and result in Attach Media Announce (AttMediaAnn) primitives at the SAPs of users that had already responded, and in notifications to the affected users.

Detach Media SE

The Detach Media SE can be used by any user to detach himself from media, by the owner of a call to detach other users from media, or by the service provider to indicate that certain attachments can no longer be provided. It can also be used to degrade a user's attachment from IN-OUT to either IN or OUT. It is an unconfirmed SE (Table 5).

■ ■ ■ ■ ■

With the Attach Media SE, a user can ask to be attached to media, invite other users to attach, or upgrade attachments.

■ ■ ■ ■ ■

Removal of users can be initiated by either the owner of the call or the service provider by means of the unconfirmed Remove Users SE.

A user can detach from one or more media by issuing a Detach Media Request (DetMediaReq) primitive. All other users are informed with a Detach Media Notification (DetMediaNtf) primitive. The owner of the call can use the Detach Media Request primitive to detach one or more users from one or more media. This is indicated to the detached users and all other users by means of a Detach Media Notification. If the detachment of one or more users from one or more media is initiated by the provider, all users in the call will receive a Detach Media Notification.

Remove Media SE

The unconfirmed Remove Media SE has been defined for removal of media (Table 6). Only the owner of a call and the service provider can explicitly remove media from a call. Other users can implicitly remove a medium if the medium object was dependent on a removed attachment object, e.g., if the user was the last user attached to the medium. Removal of a medium from a call implicitly causes all attached users to be detached. It may lead to the deletion of a media relation, if only one medium is left in the relation after medium removal.

With a Remove Media Request (RmvMedia Req) primitive the owner can remove one or more media from the call. All other users are informed by means of a Remove Media Notification (Rmv MediaNtf) primitive. This primitive can also be used by the service provider to indicate to all users in a call, including the owner, that it is no longer able to provide certain media.

Remove Users SE

Removal of users can be initiated by either the owner of the call or the service provider by means of the unconfirmed Remove Users SE (Table 7).

To remove one or more users from a call, the owner issues a Remove Users Request (RmvUsr-Req) primitive. This results in Release Call Notification primitives at the SAPs of the users specified in the User_List because the call is terminated from their point of view. All other users in the call receive a Remove Users Notification (RmvUsrNtf) primitive. If the service provider initiates the removal (by issuing a Release Call Notification at some users' SAPs) then all other users in the call, including the owner, are informed by means of a Remove Users Notification primitive.

Call Release SE

All users in a call can initiate the unconfirmed Call Release SE (Table 8). If a non-owner initiates it, the removal of the initiating user results. Only the owner of the call and the service provider can terminate the entire call.

If a non-owner in the call issues a Release Call Request (RelCallReq) primitive, Remove Users Notification primitives are issued at the SAPs of all other users in the call, including the owner, because from their point of view only that particular user has left the call. A Release Call Request primitive issued by the owner results in Release Call Notification (RelCallNtf) primitives at the SAPs of all other users, and the call is terminated. If the service provider wants to terminate the call, it issues a Release Call Notification primitive at the SAPs of all users including the owner. At any moment after the owner has started the call establishment with a Call Request primitive, the call can be terminated.

RmvMediaReq	(Call_Id , Medium_List : { Medium_Id })
RmvMediaNtf	(Call_Id , Medium_List : { Medium_Id })

■ **Table 6.** *Remove media SE.*

RmvUsrReq	(Call_Id , User_List : { User_SAP_Address })
RmvUsrNtf	(Call_Id , User_List : { User_SAP_Address })

■ **Table 7.** *Remove users SE.*

RelCallReq	(Call_Id)
RelCallNtf	(Call_Id)

■ **Table 8.** *Call release SE.*

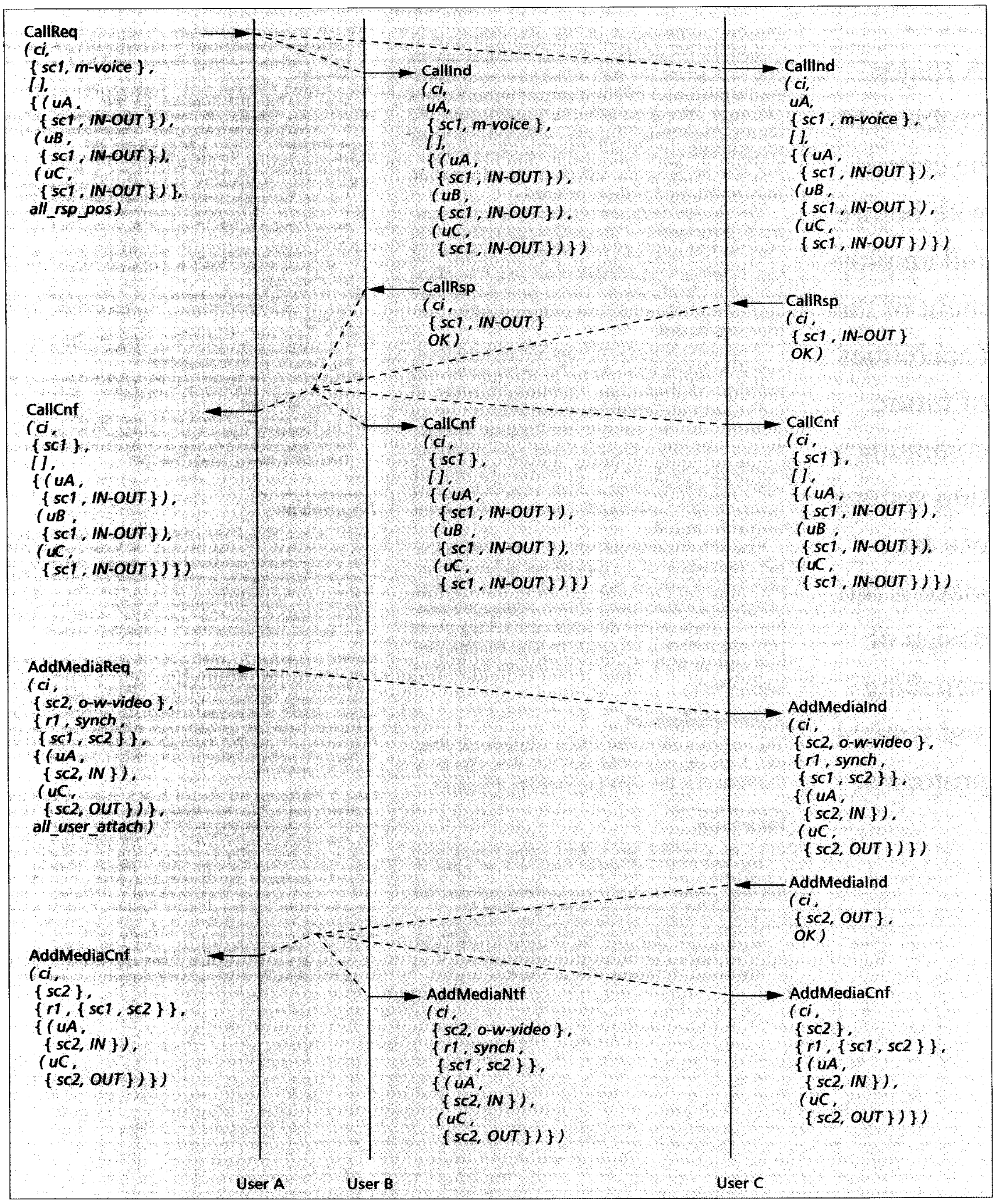
Remarks

In addition to the above listed SEs, a confirmed SE needs to be defined to enable the owner of the call to transfer the ownership to another user in the call.

We have defined SEs for the establishment, modification, and termination of calls. Once a call has been set-up for a user, i.e., as soon as the Call Confirm primitive has been received, the media can be used. Additional SEs need to be defined for this purpose. Most are related to the transport of information, but some SEs control the functions present in the medium, e.g., for a voice medium where only one user is allowed to speak at any moment, SEs are needed to pass the right to speak (floor control). The definition of SEs for the use of media is beyond the scope of this paper, since we focused on the control aspect of a multimedia multi-user service.

We defined a number of SEs to modify a call, each operating on one object type and its dependents. Alternatively, we could have defined a single sophisticated SE that operates on all object types. We feel that such an SE is too complicated and may still not be able to meet all the requirements of users, some of which might not be foreseeable. Our choice implies that it is not always possible to modify different aspects of an ongoing call in a single SE, i.e., a single extended transaction. It is up to the user to combine several SEs into one transaction if needed, e.g., to add a user and a medium to a call.

We have mainly discussed the normal execution of SEs up to now. We have foreseen abnormal execution in the sense that the initiating user or the service provider can abort the execution of confirmed SEs. All confirmed SEs have a complementary counterpart, so the execution of such an SE can be aborted at any moment after it has been invoked, by means of invoking its complementary SE. For example, to abort an Add Media request, a Remove Media request can be issued at the same SAP.



■ **Figure 5.** *Establishment of a multimedia conference call.*

Example

The use of SEs can be illustrated for the case of the successful establishment of a multimedia conference call with three users, a mixed-voice, and a one-way-video medium (Fig. 3). Figure 5 shows the time sequence diagram where the call is established by first executing a Call Establishment SE, and subsequently executing an Add Media SE.

Conclusions and Open Issues

An abstract service description of communication systems supporting multimedia multi-user applications has been developed. It describes the required functional behavior of such systems independent from possible implementations. We focused on those aspects of the system related to the establishment, mod-

A major problem to be coped with before full employment of the capabilities of future communication systems can take place is the design of signalling and control protocols.

ification, and release of calls. Analysis of the requirements led to the definition of a call model that enables one to describe the state of an advanced multimedia multi-user call by means of instantiations of objects. The call model has been used to structure the described service. Together with models at lower levels of abstraction, it can also structure the protocols to be designed, and the state information to be maintained by these protocols.

For the specification of our service, the standard confirmed and unconfirmed SEs found in the OSI/RM needed to be extended, to allow for negotiation among multiple users and/or notification to multiple users. These generic SEs are applicable to other multi-user communications environments as well.

A number of issues have been left open in our service description up to now. The exact syntax and semantics of the success condition parameter in the request and response primitives are being studied. Also, we are still investigating whether we can explicitly represent a user's permission to modify the call in the call model (i.e., by defining additional objects). Such a solution may be preferred to consulting the owner each time a non-owner wants to modify the call.

One of the major problems to be coped with before full employment of the capabilities of future communication systems can take place is the design of signalling and control protocols. Although a number of efforts in this direction are taking place [15], much work is necessary to have communication systems truly support multimedia multi-user applications.

Acknowledgment

The authors wish to thank M.A. Jordense, R. Sturrus, J. Daemen, and M. van den Bos who have contributed to the work presented in this paper.

References

[1] S. E. Minzer and D. R. Spears, "New Directions in Signaling for Broadband ISDN," *IEEE Commun. Mag.*, vol. 27, no. 2, pp. 6-14, Feb. 1989.

[2] S. Minzer, "A Signaling Protocol for Complex Multimedia Services," *IEEE J. Select. Areas in Commun.*, vol. 9, no. 9, pp. 1383-94, Dec. 1991.

[3] J. DeHart, M. Gaddis, and R. Bubenik, "Connection Management Access Protocol (CMAP) Specification," Washington University, Department of Computer Science, Technical Report 92-01, Aug. 1992.

[4] L. A. Crutcher, and A.G. Waters, "Connection Management for an ATM Network," *IEEE Network*, vol. 6, no. 6, pp. 42-55, Nov. 1992.

[5] C.M.W. Gabriel, *et al.*, "*Heading Towards an Advanced Signalling System for Multimedia, Multiparty Services in Broadband ISDN," Electron. & Commun. Engng J.*, vol. 5, no. 2, pp. 103-12, April 1993.

[6] D. J. Legare, J. W. Stewart, and L. J. Keiper, eds., *Proc. 27th Internet Engng Task Force*, IETF, Amsterdam, July 1993.

[7] E.M. Schooler, "Case Study: Multimedia Conference Control in a Packet-Switched Teleconferencing System," *Internetworking: Research and Experience*, vol. 4, no. 3, pp. 99-120, June 1993.

[8] Bellcore Information Networking Research Laboratory, "Touring Machine System," *Commun. ACM*, vol. 36, no. 1, pp. 68-77, Jan. 1993.

[9] RACE project 1044 deliverable, Specification of Customer Service Functions, Deliverable No.44/RIC/CSF/DSA/009/b1, Sept. 1992.

[10] R. Carli, *et al.*, "A Conceptual Framework for B-ISDN Multiconnection / Multiparty Calls," *Proc. ICCC 92*, Genova, Italy, pp. 283-90, 1992.

[11] ISO, "*Information technology — Open Systems Interconnection* — Reference Model — Part 1: Basic Reference Model," ISO/IEC DIS 7498-1, 1992.

[12] CCITT SG XVIII, Recommendations of the I.360-Series Submitted for Approval at the Xth CCITT Plenary Assembly, COM XVIII-R116-E, July 1992.

[13] W. H. Leung, *et al.*, "A Set of Operating System Mechanisms to Support Multimedia Applications," *Proc. 1988 Internat'l Zurich Semin. Dig. Commun.*, pp. 71-76, March 1988.

[14] L. Aguilar, *et al.*, "Architecture for a multimedia teleconferencing system," *Proc. ACM SIGCOMM 86*, Symp. Commun. Arch. and Protoc., pp. 126-36, 1986.

[15] I. G. Niemegeers, G. J. Heijenk, and X. Hou, "Signalling and Control in B-ISDN," to appear in the proceedings of the 4th International Conference on Advances in Communication and Control (COMCON 4), Rhodes, Greece, June 1993.

Biographies

GEERT J. HEIJENK [S] received an M.Sc. degree in computer science from the University of Twente, Enschede, the Netherlands, in 1988. He is a research staff member of the Center for Telematics and Information Technology at the same university, working toward his Ph.D. degree. During the summer of 1992, he worked as a visiting researcher at the University of Pennsylvania, Philadelphia. His research interests include broadband networking, ATM networks, and performance analysis. His e-mail address is heijenk@cs.utwente.nl.

XINLI HOU received the B.Sc. and M.Sc. degrees in computer engineering from Xi'an Jiaotong University, Xi'an, China, in 1984 and 1987, respectively. He also received an M.E.E. degree from Philips International Institute, Eindhoven, the Netherlands, in 1990. He is currently a research staff member of the Center for Telematics and Information Technology at the University of Twente, Enschede, the Netherlands, pursuing a Ph.D. degree. His interests are broadband networking, traffic modeling, and performance analysis. His e-mail address is xinli@cs.utwente.nl.

IGNAS G. NIEMEGEERS [M] received the M.S. degree in electrical engineering from the Rijksuniversiteit Gent. He received the M.S. degree in computer engineering and the Ph.D. degree from Purdue University, West Lafayette, IN, in 1972 and 1978, respectively. From 1978 to 1981, he was employed by the research department of Bell Telephone Manufacturing Company, Antwerp, Belgium, where he was involved in the design of packet switching systems. From 1981 to 1986, he was an associate professor with the electrical engineering department of the University of Twente, Enschede, the Netherlands. Since 1986 he has been a professor of computer science at the same university. He is currently head of the Center for Telematics and Information Technology at the University of Twente. His present research interests are high-speed networking, B-ISDN, optical networks and performance analysis. His e-mail address is niemegee@cs.utwente.nl.