Modeling Service Activation in a Multimedia Environment

The emergence of multimedia applications requires appropriate management and control activities to be provided and organized in an integrated framework. This article presents an architectural approach to the modeling of service activation processing in a multimedia environment.

By A. Lombardo, S. Palazzo, G. Schembra

Introduction

In the last few years the rapid evolution of digital technology and the efficiency it offers in manipulating, storing and retrieving information has led to further considerable change in the field of communications. This time, in the 1990s, the challenge is to make multimedia applications feasible for consumers through networking. Although interest in distributed multimedia applications is quickly growing,[1] up to now some unsolved problems have prevented them from actually being distributed as a multimedia service in a broadband network environment.

Beyond any specific application, in fact, from the user's point of view working with multimedia systems means enabling authorized operators to create, edit, transmit, receive, store, delete, etc. multiple types of information such as data, text, still and moving pictures, audio signals and so on. This alone would require integration in a specific user environment. If, instead of working alone, the user is cooperating with other people, the situation is even more complex. Activation, control, coordination and management of the activities and resources inherent to multimedia services therefore play an important role.[2]

Service activation, in particular, is a key process that has a great impact in making the use of distributed multimedia services appealing to customers. In the past efforts were devoted to designing signalling protocols supporting multiparty multimedia services through proper distributed user/network interfaces.[3–9] However, little attention has been paid to defining a complete architec-

Alfio Lombardo is an Associate Professor of Telematics at the University of Catania, Italy. His current research interests include distributed multimedia applications, network management and broadband communications.

Sergio Palazzo is an Associate Professor of Telecommunication Networks at the University of Catania, Italy. His current research interests include broadband networks, mobile communications and multimedia systems.

Giovanni Schembra is a Ph.D. Student at the University of Catania, Italy. He is presently working in the field of broadband traffic analysis and distributed multimedia applications.

tural model to be used as a general framework in which to locate the functional entities that are the building blocks of a distributed processing environment for service activation. An initial step in this direction was the work by Crutcher and Waters,[10] where a five-layer architecture for the management of connections in ATM networks is presented. The architectural framework of Crutcher and Waters, however, does not provide for customization of the multimedia services to be activated. Nor is any description given of the functional entities in the various levels. A full architectural framework for the control of high-speed packet switching networks providing support of multiparty multimedia applications is described in reference 11.[11] This focuses on a control system performing functions in order to set up, maintain and release calls in a multimedia environment. However, how the user makes use of the multimedia service, after it is established, is beyond the scope of the control system introduced in reference 11.

In this article we present a comprehensive approach to modeling the processes that are needed to allow a user to create, customize and activate a multimedia service in a broadband network environment. The architectural model exhibits the following main features:

- It takes the distinction between requirements at the user level and requirements at the network level into account.
- It uses the object-oriented approach for representing and handling information within the network, as required for the development of advanced distributed processing software, according to the principles of modularity and reusability.
- It complies with the Manager–Agent paradigm, already adopted by the CCITT for the specification of TMN[12] and by the ISO for the management of heterogeneous systems.[13]

The model is clearly an abstract representation of the behaviour of the service activation process and, as such, does not imply any specific implementation. It can, however, be considered as a step toward a standard information model for service activation which may be used across a range of products and vendor platforms. In reference 14[14] it has been shown how the model can be effectively applied as a basis to support the development and customization of prototype multimedia cooperative editing services over a broadband network.

In the following sections we will outline the internal organization, functionalities and scopes of the architectural model.

Basic Principles of the Model

The trend of public carriers and administrators of telecommunication networks has been to provide customers with the capabilities of on-demand access and management of information at any time, in any place, in any volume and in any form. In perspective, the so-called *value-added principle* will be a constraining premise in approaching the design of future intelligent networks.[15,16] In view of the value-added approach, we may consider that in a typical broadband network environment there should exist a kind of *network server* which provides all the network users with a proper interface towards the network services they require. The signalling process through which the user can formulate its service requests and set up the corresponding call is a key element in the design of such an interface.

In perspective, the so-called* value-added principle *will be a constraining premise in approaching the design of future intelligent networks.

This article presents an architectural model of the user/network signalling interface in a broadband scenario in which complex *multimedia services* are provided. Following the BISDN protocol model defined in the CCITT recommendation on broadband aspects of ISDN,[17] we can consider that service activation is conceptually contained in the Control plane. In the ATM environment, the Control plane relates to the information and functions required to control a network connection (e.g. for establishment), or to control the use of an already established connection (e.g. renegotiation of service characteristics). Analogously, in the environment of a distributed system supporting multimedia applications, the Control plane also relates to the information and functions required by the user or the network to select, activate and configure a (set of) multimedia service(s).

Figure 1 shows the architectural model we propose. In the figure, six *layers* are singled out. The precise definition of each layer will be dealt with in the following sections. For the time being, it is worth noting that the three upper layers refer to functionalities which are strictly related to the user's view of the multimedia service, while the two lower layers refer to functionalities which strictly relate to the handling of the network resources that support the multimedia information exchange. The intermediate layer describes the bearer service needed to support the service activation process and the relevant user requirements, regardless of the network resources actually avail-

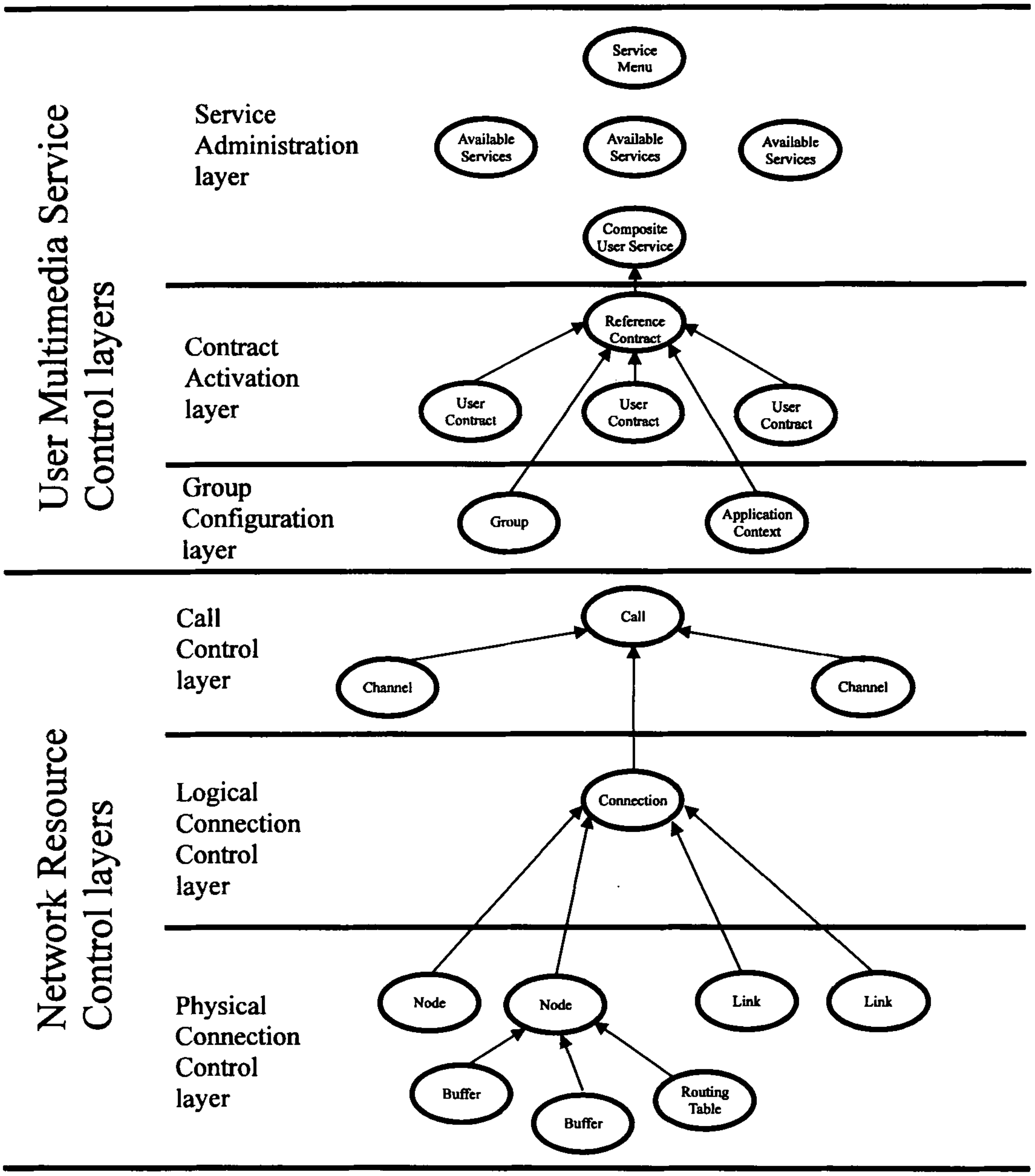

Figure 1. Service Activation architectural model.

able. Even though the above layering is merely conceptual, the main advantage of such an approach in conceiving the model is that the service implementation is decoupled from network dependencies, by allowing, for example, the binding of service components across heterogeneous networks, freedom to allocate components to network nodes, and the independence of services from network scale and topology.

Below we will globally refer to the three upper layers as the *User Multimedia Service Control* layers, and to the three lower ones as the *Network Resource Control* layers. As in the well-known ISO/OSI Reference Model for data networks, the entities which perform the functions provided in one layer must be considered distributed in several systems.

The contents of each layer are specified using the object-oriented approach. The use of Managed Objects to define a layer has implications not only for information structuring but also for modeling of the interaction among entities, which is done through the *Manager/Agent* paradigm, as in reference 13.[13] This paradigm enables a Manager to invoke operations on Managed Objects and an Agent to perform operations on the associated Managed Objects. Interaction between two adjacent layers is specified in terms of relationships between the Managed Objects which reside in the two layers.

—The User Multimedia Service Control Layers—

The main task of the User Multimedia Service Control layers is to allow users and the network server to handle the stages of selection, activation and configuration of a multimedia service.

Hereafter, a *multimedia service* is viewed as a user-selected collection of *elementary services*. In this respect, for instance, a teleconference service may be regarded as a collection of such elementary services as audio-conference, video-conference, intermedia synchronization, etc. Similarly, at a lower level of granularity, it may be viewed as an elementary service by itself and possibly selected, together with a number of telewriting services, to form a multimedia cooperative editing service.

—The Service Administration Layer—

The highest layer of the module—the *Service Administration* layer—encompasses the functions needed to offer the user a set of alternative elementary services. It thus allows the user to select the objects corresponding to the individual services and compose them into a multimedia service. This is achieved through suitable operations on some object classes, which are reported in Figure 2, where all the classes mentioned in here are specified according to the notation introduced in reference 8.[8]

Within the *Service Administration* layer, the network operator creates a *serviceMenu* class instance acting as a demon process in the network server system. The user interested in looking through the elementary services offered by the network has to invoke a *displayServices* operation; this gives rise to the creation of as many *availableService* class instances as the elementary services offered. Then, in order to define the multimedia service required, the user can invoke the *specifyService* operation on the *service menu* object to create a *compositeUserService* class instance; hereafter, the user who invokes the *specifyService* operation will be identified by the network server as the *convener* of the multimedia service being activated. In Figure 2, the *availableService* and the *compositeUserService* classes are derived from a single *superclass*, termed *Service*.

It is worth devoting some space to the attributes of the *availableService* class. For each elementary service, the network server specifies its identifier, the required Hw and Sw resources (if any), the service cost, the media by which it is supported, and the Quality of Service offered. At this level of interaction between network server and user, the Quality of Service can only be measured roughly, that is, in terms of operator satisfaction. For this reason, we refer to it as *Perceived Quality of Service (PQoS)*.[19,20] The problem of specifying the PQoS attribute for each service in some way is a critical (and still open) issue. For the sake of illustration, in Figure 3 we report the specification of the *pQoS* attribute concerning a full motion video distribution service. This attribute defines such parameters as *videoServiceType, speedRatioRange, utilizationRange, frameRateRange, pixelsPerFrameRange.* More specifically, *videoServiceType* distinguishes between constant-rate video services (CR), and variable-rate video services (VR). The *speedRatio-*

```
serviceMenu MANAGED OBJECT CLASS
  DERIVED FROM top;
  CHARACTERIZED BY
serviceMenuPackage              PACKAGE
  ACTIONS
  displayServices,
  specifyService,
  notifyService;
  NOTIFICATIONS
  selectableServices;;;
REGISTERED AS ...;
```

```
service MANAGED OBJECT CLASS
  DERIVED FROM top;
  CHARACTERIZED BY
servicePackage                  PACKAGE
  ATTRIBUTES
  serviceId                     GET,
  reqHwSwResources              GET,
  serviceCost                   GET;;;
REGISTERED AS ...;
```

```
availableService MANAGED OBJECT CLASS
  DERIVED FROM service;
  CHARACTERIZED BY
availableServicesPackage        PACKAGE
  ATTRIBUTES
  pQoS                          GET,
  supportingMedia               GET;;;
REGISTERED AS ...;
```

```
compositeUserService MANAGED OBJECT CLASS
  DERIVED FROM service;
  CHARACTERIZED BY
compositeUserServicePackage PACKAGE
  ATTRIBUTES
  services           GET-REPLACE,
  convenerId         GET,
  cooperativeService GET;
  ACTIONS
  activateContract;;;
REGISTERED AS ...;
```

```
referenceContract MANAGED OBJECT CLASS
  DERIVED FROM top;
  CHARACTERIZED BY
referenceContractPackage        PACKAGE
  ATTRIBUTES
  convenerId         GET,
  startTime          GET-REPLACE,
  endTime            GET-REPLACE,
  invited            GET-REPLACE ADD-REMOVE,
  participants       GET-REPLACE ADD-REMOVE,
  servicePassword    GET-REPLACE,
  services           GET,
  hwSwResources      GET ADD;
  ACTIONS
  join,
  abandon;
  NOTIFICATIONS
  referenceContractCreation,
  serviceActivation,
  serviceAbandonedByUser,
  joinReq;;;
REGISTERED AS ...;
```

```
userContract MANAGED OBJECT CLASS
  DERIVED FROM top;
  CHARACTERIZED BY
userContractPackage PACKAGE
  ATTRIBUTES
  services                  GET REMOVE,
  startTime                 GET,
  endTime                   GET,
  userId                    GET,
  servicePassword           GET,
  serviceCost               GET,
  hwSwResources             GET REMOVE,
  convenerId                GET;
  NOTIFICATIONS
  contractSettled;;;
REGISTERED AS ...;
```

```
group MANAGED OBJECT CLASS
  DERIVED FROM top;
  CHARACTERIZED BY
groupPackage        PACKAGE
  ATTRIBUTES
  organization              GET-REPLACE,
  userRole                  GET-REPLACE;
  NOTIFICATIONS
  roleUpdated,
  organizationUpdated;;;
REGISTERED AS ...;
```

```
applicationContext MANAGED OBJECT CLASS
  DERIVED FROM top;
  CHARACTERIZED BY
applicationContextPackage       PACKAGE
  ATTRIBUTES
  applicationContextStructure  GET-REPLACE;;;
REGISTERED AS ...;
```

Figure 2. Object classes definition for (a) the User Multimedia Service Control layers (Continued)

```
call MANAGED OBJECT CLASS
 DERIVED FROM top;
 CHARACTERIZED BY
callPackage          PACKAGE
 ATTRIBUTES
 callLog                        GET,
 resourceIntegrationPolicy      GET-REPLACE;
 ACTIONS
 establishChannel;;;
REGISTERED AS ...;
```

```
node MANAGED OBJECT CLASS
 DERIVED FROM top;
 CHARACTERIZED BY
nodePackage  PACKAGE
 ATTRIBUTES
 nodeId                 GET,
 inputPortIds           GET-REPLACE,
 outputPortIds          GET-REPLACE,
 cellLossProbability    GET,
 i/oDelay               GET;;;
REGISTERED AS ...;
```

```
channel MANAGED OBJECT CLASS
 DERIVED FROM top;
 CHARACTERIZED BY
channelPackage          PACKAGE
 ATTRIBUTES
 channelId              GET,
 supportedMedia         GET,
 qoS                    GET,
 endUserIds             GET;;;
REGISTERED AS ...;
```

```
buffer MANAGED OBJECT CLASS
 DERIVED FROM top;
 CHARACTERIZED BY
bufferPackage PACKAGE
 ATTRIBUTES
 bufferId          GET,
 bufferPosition          GET,
 bufferDimension   GET,
 queuePolicy       GET,
 bufferEdges       GET-REPLACE ADD-REMOVE;;;
REGISTERED AS ...;
```

```
routingTable MANAGED OBJECT CLASS
 DERIVED FROM top;
 CHARACTERIZED BY
routingTablePackage PACKAGE
 ATTRIBUTES
 routingTableId    GET,
 i/oTable          GET-REPLACE ADD-REMOVE,
 updatingPeriod    GET-REPLACE;;;
REGISTERED AS ...;
```

```
connection MANAGED OBJECT CLASS
 DERIVED FROM top;
 CHARACTERIZED BY
connectionPackage          PACKAGE
 ATTRIBUTES
 connectionId       GET,
 connectionType     GET-REPLACE,
 endUsers           GET-REPLACE ADD-REMOVE,
 connectionMedia    GET-REPLACE ADD-REMOVE,
 connectionDirectionality     GET-REPLACE,
 requiredQoS        GET-REPLACE ADD-REMOVE,
 providedQoS        GET-REPLACE ADD-REMOVE,
 routingStrategy              GET-REPLACE,
 resourcesIntegrationPolicy   GET-REPLACE,
 multiplexedChannels   GET-REPLACE ADD-REMOVE,
 aALType                      GET-REPLACE;;;
REGISTERED AS ...;
```

```
link MANAGED OBJECT CLASS
 DERIVED FROM top;
 CHARACTERIZED BY
linkPackage   PACKAGE
 ATTRIBUTES
 linkId                 GET,
 maxBitRate             GET,
 minTransferDelay       GET,
 lossProbability        GET;;;
REGISTERED AS ...;
```

Figure 2. Continued. Object classes definition for (b) the Network Resource Control layers.

Range and the *utilizationRange* parameters indicate the maximum/minimum ratio between the actual and the nominal presentation rate and the maximum/minimum ratio between the actual presentation rate and the available delivery rate of an information video sequence, respectively.[19] The *frameRateRange* and the *pixelsPerFrameRange* parameters specify the ranges of permitted values for the number of frames per second and the number of pixels per frame, respectively.

—The Contract Activation Layer—

The *Contract Activation* layer allows a user desiring to participate in a multimedia work session to

```
fullMotionVideoDistributionPQoS  ATTRIBUTE
 WITH ATTRIBUTE SYNTAX
AttributeModule.fullMotionVideoDistributionPQoS;
 MATCHES FOR                EQUALITY;
REGISTERED AS...;
```

ASN.1 module

```
AttributeModule
DEFINITIONS ::= BEGIN
fullMotionVideoDistributionPQoS::= SEQUENCE {
        videoServiceType IA5String   ::= {"CR","VR"}
        speedRatioRange              ::= SEQUENCE {
                                     MinSpeedRatio::= REAL,
                                     MaxSpeedRatio::= REAL},
        utilizationRange             ::= SEQUENCE {
                                     MinUtilization::= REAL,
                                     MaxUtilization::= REAL},
        frameRateRange               ::= SEQUENCE {
                                     MinFrameRate::= REAL,
                                     MaxFrameRate::= REAL},
        pixelsPerFrameRange          ::= SEQUENCE {
                                     MinPixelsPerFrame::= REAL,
                                     MaxPixelsPerFrame::= REAL}}
END
```

Figure 3. The fullMotionVideoDistributionPQos attribute template.

characterize its *user contract* in terms of some of its aspects: here, following reference 21, we will refer to a user contract as a specific multimedia service being supplied to a given user, after its characteristics have been negotiated and agreed upon between the user and the network server. In general, each of the users requesting access to the same multimedia service stipulates its own contract independently of the other users; nevertheless, all the individual contracts must clearly have some common attributes. For this reason, any user must derive some of the attributes characterizing its user contract from a *reference contract*.

In order to define the contract, the convener has to prompt the Manager process that resides in its station to invoke an *ActivateContract* operation on the object modeling the *composite user service*. As a consequence, this object creates an instance of the *referenceContract* class, whose attribute values are to be assigned by the convener. The following is a short list of examples of such attributes. The *convenerId* is a data type which specifies the convener of the work session. The *startTime* and *endTime* attributes allow the convener to set the opening and closure dates of the contract (in some cases their values may be left undetermined). The *invited* attribute is a data type whose value is assigned by the convener in order to define a list of users invited to the work session. The *participants* attribute identifies the users joining the multimedia service, although not invited. The *servicePassword* attribute is a data type which must be specified whenever a security requirement is applied. The *services* attribute, already defined for the *compositeUserService* class, is defined as a list of the elementary services selected, each of which is represented, in turn, by a sequence of the service identifier and the offered PQoS. The *hwSwResources* attribute allows the convener to specify the minimal user requirements in terms of resources.

This attribute comprises all the Hw and Sw resources required for each elementary service selected. As soon as its attribute values have been assigned, the *reference contract* object sends notification to all the users whose identifiers appear in the *invited* attribute to prompt them to create a proper instance of the *userContract* class. By default, the attributes of the *user contract* objects are set to the values of the corresponding attributes of the *reference contract*, already specified by the convener. However, some of these attributes may be replaced by the user in order to characterize its own contract. For example, a user might join the multimedia service, possibly selecting a reduced number of elementary services or modifying the values of the *pQoS* attribute inside the range of the values permitted for each service. Then, each *user contract* object notifies the convener of the successful contract settlement.

At the date specified by the *startTime* attribute, the *reference contract* object sends notification of *serviceActivation* to every participant. Moreover, if the convener has set the *cooperativeService* attribute in the *composite user service* object, it creates an instance of the *group* class and an instance of the *applicationContext* class in the *Group Configuration* layer.

—The Group Configuration Layer—

The organization of activities inside a group strongly depends on the nature of the service invoked. For example, some multimedia applications like video-conferencing, joint editing, software planning, etc. require the structure of the group to reflect a hierarchical organization of roles (e.g. editor, author, referee, consultant, etc.), which may be modified at run-time by a participant who is appointed as a coordinator. Some other applications, like distance learning and multimedia information retrieval, require the group to be structured again in hierarchical roles (e.g. teacher, student, inquirer, etc.), but, unlike the previous case, this organization is fixed. Lastly, such applications as research consulting, telemedicine or remote engineering require no structure inside the group because all participants must have the same rights. Besides, the hierarchical organization may be asymmetrical (that is, assignment of roles implies a source-sink relation among the role players) or symmetrical (that is, any role player may be source as well as sink). In a distance learning service, for example, the organization may be either symmetrical (that is, students are also allowed to write on the multimedia blackboard) or asymmetrical (that is, only the teacher can do so).

To achieve this, the *Group Configuration* layer allows the user to specify the values of the *organization* and the *userRole* attributes defined in the *group* class (the latter being defined as a sequence of data types specifying the participant identifiers, their roles and the Hw and Sw resources they require to be activated). It also allows the user to specify the structure of the multimedia information to be shared in the work session in terms of the logical, temporal and spatial relationships among its component parts. This is done through the *applicationContextStructure* attribute in the *applicationContext* class. This attribute is defined as a choice of different data type structures, in order to reflect the possible document structures specified by the various relevant standards (e.g., ODA for conventional documents,[22] its extensions for documents with audio information,[23] HyperODA for documents with temporal relationships,[24] MHEG for documents based on Multimedia/Hypermedia information,[25] etc.). In Figure 4, two structures, called *ODA document* and *MHEG document* respectively, are envisaged.

As soon as the attribute values of all the classes in the *Group Configuration* layer have been assigned, the object modeling the group sends notification to prompt all the user systems in which the participants reside to activate the Hw and Sw resources providing the proper support for the multimedia work session.

Note that the objects modeling the reference contract, the user contract, the group and the application context are related to the same object modeling the user composite service by a containment relationship, since they, as a whole, represent all those aspects of the multimedia service that are of interest to the user.

The Network Resource Control Layers

In a distributed multimedia environment, competition for network resources can easily become a tangible problem, due to the large variety of sources generating traffic with very different statistical characteristics. Actually, multimedia sources may generate both Continuous Bit Rate (CBR) and Variable Bit Rate (VBR) signals, possibly with a wide spectrum of burstiness values; also, service categories with different QoS network requirements can easily be encountered. The functions of the Network Resource Control layers are

```
applicationContextStructure   ATTRIBUTE
  WITH ATTRIBUTE SYNTAX
  AttributeModule.ApplicationContextStructure;
  MATCHES FOR                    EQUALITY;
REGISTERED AS...;
```

ASN.1 modules

```
AttributeModule
DEFINITIONS ::= BEGIN
IMPORTS Oda-Document       FROM     Oda-Document-Module
        Mheg-Document      FROM     Mheg-Document-Module;
ApplicationContextStrcture ::= CHOICE {
        oda-document Oda-Document,
        mheg-document    Mheg-Document}
END
```

```
Oda-Document-Module
DEFINITIONS ::= BEGIN
EXPORTS  Oda-Document;
IMPORTS Document-Profile-Descriptor
        FROM CCITT Rec.T.415 Document-Profile-Descriptor {2 8 1 5
6}
        Layout-Class-Descriptor, Layout-Object-Descriptor,
        FROM CCITT Rec.T.415 Layout-Descriptors {2 8 1 5 8}
        Logical-Class-Descriptor, Logical-Object-Descriptor,
        FROM CCITT Rec.T.415 Logical-Descriptors {2 8 1 5 9}
        Presentation-Style-Descriptor, Layout-Style-Descriptor,
        FROM CCITT Rec.T.415 Style-Descriptors {2 8 1 5 10};
Oda-Document              ::= SEQUENCE {
 document-profile      Document-Profile-Descriptor,
 other-data-elements   SEQUENCE OF CHOICE {
        layout-object-class [0]   Layout-Class-Descriptor,
        logical-object-class[1]   Logical-Class-Descriptor,
        presentation-style  [2]   Presentation-Style-Descriptor,
        layout-style    [3] Layout-Style-Descriptor,
        layout-object   [4] Layout-Object-Descriptor,
        logical-object  [5] Logical-Object-Descriptor}}
END
```

```
Mheg-Document-Module
DEFINITIONS ::= BEGIN
EXPORTS  Mheg-Document;
IMPORTS Media-Content-Descriptor             FROM        ISOmheg-MH
{2 1 7 1003}
 Spatial-Projector-Descriptor FROM        ISOmheg-MH {2 1 7 1005}
 Audio-Projector-Descriptor   FROM        ISOmheg-MH {2 1 7 11}
 Numeric-Projector-Descriptor      FROM   ISOmheg-MH {2 1 7 12}
 Projectable-Descriptor       FROM        ISOmheg-MH {2 1 7 13}
 Interaction-Descriptor       FROM        ISOmheg-MH {2 1 7 14}
 Link-Descriptor                   FROM   ISOmheg-MH {2 1 7 15}
 Composite-Descriptor         FROM        ISOmheg-MH {2 1 7 16};
Mheg-Document              ::= SEQUENCE OF CHOICE {
        media-content-class               Media-Content-Descriptor,
        spatial-projector-class   Spatial-Projector-Descriptor,
        audio-projector-class     Audio-Projector-Descriptor,
        numeric-projector-class   Numeric-Projector-Descriptor,
        projectable-class         Projectable-Descriptor,
        interaction-class         Projectable-Descriptor,
        link-class           Link-Descriptor,
        composite-class           Composite-Descriptor}
END
```

Figure 4. The applicationContextStructure attribute template.

primarily concerned with supporting the information streams to be conveyed with the proper QoS requirements through the utilization of suitable network resources. With respect to this we should also take into account the characteristics of the transport system. In this paper we assume that the multimedia environment lies on the top of an ATM network. In this kind of network, there exists sufficient capability of real-time control to consider that PQoS-to-QoS mapping can be effectively managed in the network. In fact, in this case, one can rely on the presence of the proper types of ATM Adaptation Layer (AAL), like those specified by the CCITT[26] corresponding to the four BISDN Service classes.[27]

In a distributed multimedia environment, competition for network resources can easily become a tangible problem, due to the large variety of sources generating traffic with very different statistical characteristics.

The Network Resource Control functions are thus organized into three layers—the *Call Control* layer, the *Logical Connection Control* layer and the *Physical Connection Control* layer.

—The Call Control Layer—

The *Call Control* layer covers the following functions:

- Providing an interface which decouples user application requirements from network dependencies
- Achieving an appropriate mapping between PQoS and QoS attributes
- Selecting adequate resource management policies according to the QoS requirements.

The entities which perform such functions are typically spread out in the user systems, in the network server, and in the network operation center (in turn, this may be either centralized or distributed and, in the former case, may be located at the server site).

The *Call Control* layer is mainly concerned with two distinct concepts: *call* and *channel*. *Call* is a set of channels that are logically associated for the purpose of supporting the multimedia service; it may be multiparty. *Channel* is a logical means which models the end-to-end transfer of a monomedia stream between two users of the multimedia service.

In this layer the *call* class is defined. The *callLog* attribute of this class summarizes the characteristics of the multimedia work session that is going to be set up. It is defined as a sequence of data types consisting of the identifiers of the elementary services selected, the relative media involved and the PQoS required, for each of the users participating in the work session. The *resourceAssignmentPolicy* attribute may assume the values 'Guaranteed' or 'BestEffort', according to whether the user has requested the network to undertake to provide the required QoS levels.[28]

As soon as the *call* class is created, any instance of this class in turn creates one or more instances of another class residing in the same layer, the *channel* class. More specifically, for each pair of participants, it creates as many objects as the media specified in the *callLog* attribute. These objects are related to the object modeling the call by a containment relationship. Each of the objects modeling a channel is characterized by the values of such attributes as the *channelId*, the *supportedMedia*, the *qoS* to be provided and the *endUserIds* (defined as the sequence of two identifiers).

—The Logical Connection Control Layer—

To achieve point-to-point or multipoint communication relating to a call, the network operation center (possibly through a Manager/Agent remote operation, if it resides in a site other than the network server) invokes an *establishConnection* Action on the object modeling the call. As a consequence, the object creates one or more instances of the *connection* class in the underlying layer—the *Logical Connection Control* layer.

A *connection* is a communication path for a flow of monomedia or multimedia information, connecting the network access points (terminal nodes) for the users taking part in the call by several, one or no nodes.[8] A connection can thus be considered to be the implementation of the call. To open the connection for a call, the network server creates an instance of it, specifying in the *connectionType* attribute whether the connection is point-to-point or multipoint, in the *endUsers* attribute the call participants to be linked, in the *connectionMedia* attribute the media involved in the communication (video, audio and/or data), and in the *connectionDirectionality* attribute whether the connection is unidirectional, symmetric bidirectional, or asymmetric bidirectional. Finally, the two attributes *requiredQoS* and *providedQoS* are used so that the connection provides the PQoS requested by the users in the contract stage and specified in the *call* object. The *requiredQoS* attribute specifies the parameters requested by the user, such as peak bandwidth, average bandwidth, grade of synchronization, delay, and jitter; the *providedQoS* attribute, on the other hand, contains the values they progressively assume in the network. If the performance of the connection does not meet the required QoS parameters, a reconfiguration mechanism may fire, e.g. to recalculate the routing tables for the nodes involved in the communication.

The *routingStrategy* attribute specifies whether the connection is a Virtual Path Connection (VPC) or a Virtual Circuit Connection (VCC). In the first case the network only uses the Virtual Path for routing at each intermediate node, and thus does not translate the Virtual Channel Identifier (VCI). This allows the VCI to be used for source discrimination in multipoint connections, provided that each user places a unique VCI in all of its outgoing cells. In the second case, the network uses the VCI to route the cells and may also translate the Virtual Path Identifier (VPI) on internal links (for intertrunk grouping, where multiple VCCs are carried on preconfigured internal trunks). The VCI, therefore, has no end-to-end significance. If VCCs are used for multipoint communication, higher-level information embedded in the ATM cell payload must be used for source discrimination, and for connections that want to take advantage of rapid set-up, thanks to preconfigured trunks.

The *resourcesIntegrationPolicy* attribute may assume one of two values: 'MVC' or 'IVC'. The first value corresponds to adopting the approach introduced in reference 29,[29] where the monomedia channels relative to the same multimedia service are multiplexed onto a single Multimedia Virtual Circuit (MVC). The second value corresponds to adopting the approach proposed in reference 30,[30] where the individual monomedia channels are mapped onto Independent Virtual Circuits (IVC). An IVC connection has the advantage of allowing the QoS parameters to be selected separately for each channel, and thus for each media. On the other hand, it presents the problem of keeping the various channels involved in a communication synchronized. This can be achieved by transmitting synchronization points, which allow the receiver to reconstruct the global multimedia signal correctly. The value of this attribute is decided by the network server according to the values of the attributes of the *channel* object.

Other attributes of the *connection* class are the *multiplexedChannels*, representing the identifier(s) of the channel(s) composing the connection, and the *aALType* to be supported. The actual values of these attributes are assigned according to those assumed by the objects modeling the call and the channels.

As far as the *aALType* attribute is concerned, its actual value is set to the Adaptation Layer type required by the channel which, among all those multiplexed, presents the most constraining QoS requirements. In order to illustrate how the decision-making function encapsulated in the *connection* object infers the *aALType* value from the *qoS* of the *channel* object, let us address the case of a connection consisting of a monomedia channel which supports a video stream transmission. Referring to the PQoS attribute specified in Figure 3 for the full motion video distribution service, two different types of video service have been envisaged, each of which corresponds to a distinct value of the *videoServiceType* parameter. The first and most demanding video service type, which we have called CR (Constant Rate), refers to those situations in which the user does not accept any degradation in the quality of presentation at the receiving user site: in practice, the end-to-end path must behave as an isochronous channel. As a consequence, the QoS of the relevant channel must specify a transfer delay jitter equal to zero and a throughput value equal to the product of the nominal frame rate, the number of pixels per frame and the number of bits per pixel. For this reason, the CR video service type can only be mapped on either the Type 1 AAL (corresponding to the BISDN Service class A for Connection Oriented CBR traffic) or the Type 2 AAL (corresponding to the BISDN Service class B for Connection Oriented VBR traffic), depending on whether the video coding scheme used produces isochronous or asynchronous traffic. The second service type, called VR (Variable Rate), refers to the case in which the user may perceive and tolerate a degradation in the quality of presentation on the occurrence of a sudden increase in the network load. The VR video service type can only be mapped on the Type 2 AAL.

—The Physical Connection Control Layer—

In the *Physical Connection Control* layer there are the objects which represent all the network devices that allows a call to be activated: *nodes* and *links*. Each *node* is in turn made up of a number of *buffers* and a *routing table*.

A *node* is a data processing and switching device whose task is to route calls towards the desired destination. It has *n* input ports and *m* output ports, the identifiers of which are contained in the attributes *inputPortIds* and *outputPortIds;* a node allows one or more input ports to be connected with one or more output ports. Each node has a cell loss probability (*cellLossProbability* attribute) and an input/output delay (*i/oDelay* attribute). Cell loss may be due to contention on an output port or overflow in an internal buffer. Delay, on the other hand, may be due to the internal processing times and the buffer queueing times.

A *buffer* is an object characterized not only by its position in the node (*bufferPosition* attribute), which can be input or output, but also its size (*bufferDimension* attribute), expressed as a number of cells, and the management policy used (*queuePolicy* attribute), which is normally FIFO. The network server can establish thresholds in the *bufferEdges* attribute, so as to measure the amount of traffic transiting through the node and eventually signal possible imminent congestion in the node if one of these thresholds is exceeded. If this happens, the network server will try to modify the network topology by changing the routing table of some nodes and, if this is not enough, will renegotiate the PQoS parameters previously agreed on or, as a last resort, will cause some of the connections to crash.

The *routingTable* is an object containing all the routing information for the node, i.e. which outputs are connected with which input; this information is contained in the *i/oTable* attribute. As the communications are point–multipoint, multipoint–point, and multipoint–multipoint, the inputs may be connected with several outputs. According to the dynamics of the network traffic, the routing tables have to be periodically updated. This operation is centralized if there is only one server calculating the tables for all the nodes, or distributed if the calculation is made locally on the basis of information received from adjacent nodes. The updating period is established in the *updatingPeriod* attribute.

Two adjacent *nodes* are connected by one or more *links* and provide physical support for *channels*. According to the underlying network layers and the transmission medium used, a *link* is characterized by parameters such as maximum admissible bandwidth (*maxBitRate* attribute), minimum transmission delay between the two endpoints of the link (*minTransferDelay* attribute), and link loss probability (*lossProbability* attribute).

Conclusions

In this article we have introduced a model for the activation of multimedia applications in a broadband network environment in which the service provision is achieved through a server. The model introduced has two basic features:

- It reflects the distinction between requirements at the user level and requirements at the network level
- It uses the object-oriented approach and the Manager–Agent paradigm.

In devising the model we have adopted an approach, based on modularity and re-usability principles, which is in line with the current trend in service deployment engineering for emerging intelligent networks.

As a result, we have proposed a layered architecture, in which each layer is a set of Managed Objects that provides a common perspective of some prominent concepts concerning the activation of multimedia applications and the providing of the bearer services needed. By way of illustration, we have also addressed a multimedia cooperative service scenario and introduced a description of the relevant Managed Object classes and their attributes.

References

1. J. Rosenberg, R. E. Kraut, L. Gomez and C. A. Buzzard, Multimedia communications for users, *IEEE Communications Mag.*, **30**, No. 5, May 1992.
2. S. R. Ahuja and J. R. Ensor, Coordination and control of multimedia conferencing, *IEEE Communications Mag.*, **30**, No. 5, May 1992.
3. S. Minzer, A signalling protocol for complex multimedia services, *IEEE Journal on Select. Areas in Communications*, **9**, No. 9, December 1991.
4. S. Mazumdar and A. A. Lazar, Modeling the environment and the interface for real-time monitoring and control, *Proc. ICC 91*, Denver, June 1991.
5. M. Tsuchida, A. A. Lazar, and N. G. Aneroussis, Structural representation of management and control information in broadband networks, *Proc. ICC 92*, Chicago, June 1992.
6. A. Paglialunga and M. Siviero, ISCP (ISDN Signalling Control Part): The best candidate for the target broadband—ISDN signalling protocol, *European Transactions on Telecommunications and Related Technologies*, **4**, No. 2, March–April 1993.
7. M. E. Gaddis, R. Bubenik and J. D. DeHart, A call model for multipoint communication in switched networks', *Proc. ICC '93*, Chicago, June 1993.
8. T. F. La Porta and M. Veeraghavan, Description of a functional signaling architecture for broadband networks', *Proc. Globecom '93*, Houston, November 1993.
9. G. J. Heijenk, X. Hou and I. G. Niemegeers, Communication systems supporting multimedia multiuser applications, *IEEE Network*, **8**, No. 1, January–February 1994.
10. L. A. Crutcher and A. G. Waters, Connection management for an ATM network, *IEEE Network*, **6**, No. 6, November–December 1992.
11. X. Hou and I. G. Niemeegers, A control system architecture for high speed packet switching networks, Memoranda Informatica TIOS 94-04, University of Twente, Enschede, The Netherlands.
12. CCITT Study Group IV, Draft Recommendation M.3010—Principles for a Telecommunication Management Network, 1992.
13. ISO/IEC 10040, Information technology—Open Systems Interconnection—System management overview, 1991 (also CCITT Recommendation X.701).
14. A. Lombardo, S. Palazzo and G. Schembra, A model for multimedia service creation and activation, *Proc. ICC '94*, New Orleands, 1–5 May 1994.
15. J. J. Garrahan, P. A. Russo, K. Kitami and R. Kung, Intelligent network overview, *IEEE Communications Mag.*, **31**, No. 3, March 1993.
16. W. J. Barr, T. Boyd and Y. Inoue, The TINA initiative, *IEEE Communications Mag.*, **31**, No. 3, March 1993.
17. CCITT, Recommendation I.121—Broadband aspects of ATM, 1988.
18. ISO/IEC 10165-4, Structure of management information—Part 4: Guidelines for the Definition of Managed Objects, 1992 (also CCITT Recommendation X.722).
19. T. D. Little and A. Ghafoor, Network considerations for distributed multimedia object composition and communication, *IEEE Network Mag.*, **4**, No. 6, November 1990.
20. F. Davoli, M. Iudica, A. Lombardo, M. Maresca and S. Palazzo, A network management and control scenario for multimedia applications, *Annales des Telecommunications*, **49**, No. 1, January–February 1994.
21. N. Natarajan and G. M. Slawsky, A framework architecture for multimedia information networks, *IEEE Communications Mag.*, **30**, No. 2, February 1992.
22. ISO/IEC 8613-1, Office Document Architecture (ODA) and Interchange Format—Part 1: Introduction and general principles, 1989 (also CCITT Recommendation T.410 Series).
23. ISO/IEC 8613-9, Office Document Architecture (ODA) and Interchange Format—Part 9: Audio Content Architecture, 1992.
24. ISO/IEC 8613 PDAM 7, Office Document Architecture (ODA) and Interchange Format—Addendum 7: HyperODA—extensions for temporal relationships, 1992.
25. ISO/IEC, JTC1/SC29/WG12 (MHEG), Coded representation of multimedia and hypermedia objects—MHEG Working Document E.5—Part 1: Base Notation, 1992.
26. CCITT, Recommendation I.362—BISDN ATM Adaptation Layer (AAL) Functional Description, 1990.
27. CCITT, Recommendation I.211—BISDN Service Aspects, 1990.
28. A. Campbell, G. Coulson and D. Hutchinson, A quality of service architecture, *Computer Communication Review*, **24**, No. 2, April 1994.
29. W. F. Leung, T. J. Baumgartner, Y. H. Hwang, M. J. Morgan and S. Tu, A software architecture for workstations supporting multimedia conferencing in packet switching networks, *IEEE Journal of Select. Areas in Commun.*, **8**, No. 3, April 1990.
30. C. Nicolau, An architecture for real-time multimedia communication systems, *IEEE Journal of Select. Areas in Commun.*, **8**, No. 3, April 1990. ■

CCC 1055–7148/96/060299–13$17.50