[RaSt93] *Kaliappa Ravindran, Ralf Steinmetz*; **Transport-level Abstractions for Multimedia Systems;** Workshop "ICCC Multimedia Communications '93", Banff, Kanada, April 1993 und als Tagungsband des GI/ITG Workshop "Verteilte Multimedia Systeme", S.101-125, 18.-19. Februar 1993, K.G. Sauer Verlag, 1993.

Transport-level Abstractions for Multimedia Communications

K. Ravindran

Department of Computing & Information Sciences
Kansas State University
Manhattan, KS 66506 (USA)

R. Steinmetz

IBM European Networking Center
Vangerowstr. 18
D-6900 Heidelberg (Germany)

Abstract

With multimedia transport, user entities reside in subscriber terminals and communicate with one another over a network by sending and receiving multiple data streams as application level information (e.g., graphics, audio and text in a catalogue browsing service). The evolving multimedia applications generate requirements for complex *transport capabilities*, i.e., functional features, in the end-to-end communication system such as handling of heterogeneity among communicating terminals, supporting finer levels of user-specifiable quality of data transport service and synchronization of various data streams for delivery at users in real-time. Accordingly, the communication system may be viewed as extending the basic capabilities provided by the backbone network (e.g., bandwidth allocation) into a set of transport capabilities suitable for complex applications. This paper presents an object-oriented view of user interface to the communication system and an approach to the design of underlying transport protocols. The object orientation decomposes an application level data transport into a set of network channel objects, with each channel object handling a seperate data stream. The various objects interact using a 'dataflow programming' style, which allows a high degree of communication level parallelism and an elegant separation of data transport functionalities. The object-oriented view allows easier interworking of a multimedia communication system with existing networks and easier integration multimedia transport into programming environments.

Key words: Transport connection, quality of service (QOS), object granularity, attribute-based object specification, object interconnection, data & control transport architecture.

1 Introduction

The delivery of digital subscriber services of diverse characteristics and features (e.g., digital TV, video telephony, remote visualization, computer information transfers) to subscriber terminals through high speed wide area networks such as fiber optic networks and interconnected LANs is a technological mission of this decade in computing and communications. One of the functionalities required of such a multi-service network is the support for *multimedia data transport*, whereby user entities residing in terminals can communicate with one another over a network by sending and receiving of multiple data streams as application level information (e.g., graphics, audio and text in a catalogue browsing service). The function of the end-to-end communication system is to collect the multimedia data generated by source entities, move the data through the network and deliver the data at destination entities for consumption.

The multimedia data transport in the evolving applications imposes requirements for complex *transport capabilities*, i.e., functional features in the end-to-end systems, such as handling of heterogeneity among communicating terminals, providing mechanisms to exercise finer levels of user control on network resource allocations and synchronization of various data streams for delivery at users in real-time [1, 2]. Accordingly, the communication system may be viewed as extending the basic capabilities provided by the backbone network into a set of canonical transport capabilities suitable for multimedia applications. As an example, 'bandwidth reservation' for a data path is a basic capability provided by the network [3] that can be extended by the communication system to a 'on-demand bandwidth allocation' capability for variable rate data sources by providing appropriate transport level buffering of data and exercising the 'bandwidth reservation' feature of the network as and when data rates change significantly. As another example, 'delay jitter control' for a data path is a basic capability provided by the network, which can be extended to a 'media skew control' capability for multimedia data transport by specifying appropriate bounds on the delay jitter of various media data paths based on the real-time media synchronization needs [4, 5]. In general, the transport requirements of a multimedia application may be mapped into an instance of transport capabilities provided by the communication system which may in turn be mapped (by a transport protocol in the

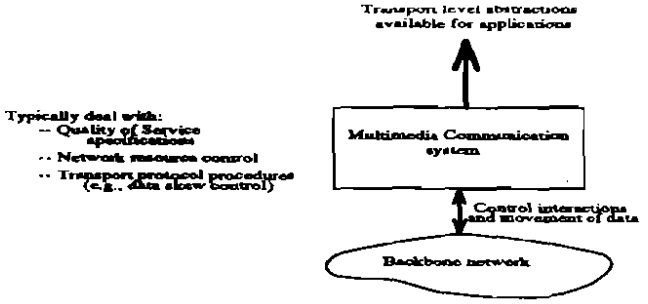

Figure 1: Layers of functions in multimedia communication systems

communication system) to an instance of basic capabilities provided by the network.

Architecturally, the communication system resides on top of the backbone network and exports a higher level transport service to the user entities implementing an application. The service interface, referred to as *user-network interface* (UNI), embeds a set of abstractions characterizing the various transport capabilities. The user entities invoke primitives (or constructs) provided by the UNI to exercise the transport capabilities to the extent required. See Figure 1. In Q.931 ISDN signaling interface for instance [6], a 'call' is a transport level abstraction that binds to a network data path with certain bandwidth and data loss characteristics. An ISDN voice terminal may create a 'call' by specifying a bandwidth of 64 *kb/sec* and a maximum bit loss rate of 10^{-5}. Such a high level 'programming' of the backbone network allows flexibility and extensibility in structuring the complex communications required by multimedia applications.

Viewing each media as being handled by an orthogonal functional component in the communication system for data coding and network resource allocation during data transport, the control relationships between various media manifest as flow of control information between the functional components that handle these media to synchronize their data delivery. In video telephony for example, the individual functional components dealing with the audio and video manage appropriate data paths in the network and exchange timing information to enable simultaneous data delivery at users. Such control interactions between transport components can best be specified and represented using an *object-oriented* model of the communication system that defines a set of object types to handle the transport of media data and a set of primitives (or operations) that allow user entities to invoke object instances for a given application. Our paper focusses on providing suitable abstractions in the UNI towards this end.

In our paper, the UNI decomposes a user level communication into a set of *channel* objects, with each channel handling the transport of data stream belonging to a media through the network and control information exchanged across various objects to exercise inter-media synchronization. The inter-media relationships depict a pattern of flow of control information across various objects by operation invocations. Since the information flow pattern is basically a syntactic aspect, complex inter-media relationships can be easily and flexibly incorporated in the object structure with a uniform mechanism[1]. From the user perspective, this object-oriented view of communication allows 'micro-management' of network resources, enrichening the semantics of data transport and better adaptation to data synchronization procedures. The transport primitives are designed to accrue these benefits. The multimedia signaling protocol developed in the EXPANSE project at BellCore [2] embodies a similar object-oriented approach. In our paper though, we focus more on transport abstractions rather than on protocols in the communication system.

To illustrate the benefits of using an object-oriented approach, the paper discusses salient features of the communication system models designed at the Kansas State University [8] and that of the Heidelberg Multimedia Communication System developed at the IBM European Networking Center [9]. The object-oriented approach also allows easier interworking of the communication system with today's networks such as ISDNs and data networks. It also allows a seamless integration of media data transport into multimedia programming environments.

The paper is organized as follows: Section 2 describes an object-oriented view of handling media data transport. Section 3 describes the salient features of a multimedia communication system arising from this view. Section 4 describes our model of object orientation in the communication system. Section 5 discusses the architectural perspective of the communication system that incorporates the object model. Section 6 discusses related works. Section 7 concludes the paper.

2 Transport level media handling

We assume that the temporal axis of an application is segmented into real-time intervals containing data units meaningful to the application. In digital TV for example, the screen image and a set of audio samples corresponding to the image are human perceptible data units occurring over an interval of 30-40 *milliseconds* (*msec*). The communication system activities are specifiable at the granularity of such data units (also referred to as data segments) [8, 9].

2.1 Semantics of media interactions

We view each media as affecting a distinct part of the application level state (e.g., audio and video data gener-

[1] The traditional forms of communications such as virtual circuits and datagrams [7] merely appear as specific instances of binding transport objects to channels with appropriate characteristics.

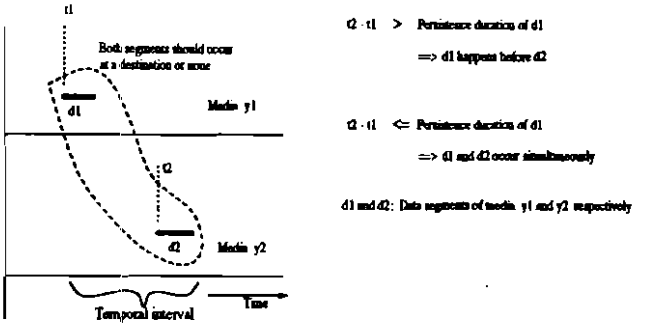

Figure 2: Illustration of inter-media relationships

ate aural and visual cues respectively in a human listener). The control relationships between various media are determined by the various ways in which the state changes can be composed. The occurrence of state changes upon generating and delivering the data units of various media at user entities causes the application to progress along the temporal axis [10]. This view allows us to identify certain control relationships that may exist among various media. See Figure 2.

Atomicity of media data

Atomicity requires that the state changes due to either all or none of the media data take effect in the application during an interval. Consider for instance, a multimedia window based application. Each activity, say on a document displayed on a workstation window, generates multimedia data, possibly consisting of graphic input from a menu using mouse click to highlight a certain portion of the document, text input from a keyboard to update the document and audio input from a voice phone for annotations. A requirement may be that the data of either all or none of these media occur at the window in an update activity. Weaker forms of atomicity are also possible based on application needs (e.g., at least text should be delivered for catalogue browsing).

Temporal ordering of media data

Temporal ordering depicts a sequence in which data units are delivered at destination entities in an interval. This ordering may be specified by the application based on the allowable sequences of state changes caused by the data and real-time persistence effects of the data (i.e., how long the effects of a data unit linger on in the application). The orderings can range between a strong form in which exactly one delivery sequence is possible and weaker forms in which more than one delivery sequence is possible (e.g., text and voice deliverable in any sequence after the graphics image in the example of catalogue browsing). The simultaneity in delivery of data units of different media (e.g., audio and video in digital TV) is also subsumed in the notion of temporal ordering.

The atomicity and temporal ordering of media data can be projected as transport level properties of the application. For instance, the data burstiness and the average data rate may be derived from the temporal characteristics of various media in the application. The transport properties are invariant, i.e., do not change over time or across different instances of the application[2], and often manifest as requirements on the communication system. For instance, the latter may need to buffer a data segment for later delivery after the arrival of a temporally preceding data segment. So the transport properties can be built into the control structure of transport level functional modules implementing the application.

2.2 Media transport and quality of service

The transport requirements of a data stream may be specified in the form of *quality of service* parameters which are indicative of the guarantees required of the network. Typical parameters specifiable at the network level, referred to as QOS_{net}, are the bandwidth to be reserved, tolerable end-to-end data delays and tolerable data loss rate [3, 11, 12, 13, 14]. The various data streams in a multimedia communication may require different QOS_{net} guarantees from the network.

Other higher level service parameters specifiable at the application interface to the communication system (i.e., across the UNI), referred to as QOS_{transp}, include information on:

- Data burstiness, average data rate and sensitivity to real-time skew;
- Intra-media/inter-media synchronization and other control relationships such as atomicity and ordering of data units (e.g., how a data loss in one channel can affect the flow of data on another channel).

See Figure 3. The above information in QOS_{transp} are derived from the application characteristics, and are used in enforcing an appropriate delivery schedule on data units. Enforcement of the high level guarantees consists of mapping the QOS_{transp} parameters into a set of QOS_{net} parameters (e.g., burstiness, average rate and skew sensitivity influence the bandwidth and delay specifications on channels) and executing an appropriate transport level protocol in the end-systems to manage the interactions between various data streams [5, 8, 15].

Such a support for complex transport services can best be realized with an object-oriented model of the communication system.

[2] The invariance of a control relationship between media does not imply that a value instantiating the control relationship cannot change. In the example of catalogue browsing service, the relation 'graphics occurs before text' holds true for every catalogue page data; however, the time delay between the text and graphics can be a variable parameter that may depend on the current execution environment, and may change across different invocations of the service.

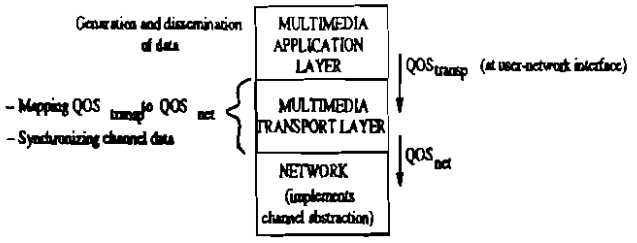

Figure 3: Illustration of transport layer and network layer qualities of service

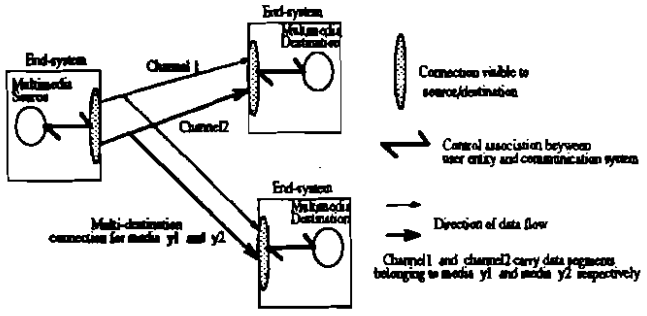

Figure 4: Connection and channel functions in multimedia communication systems

2.3 Functional decomposition of media transport

An object-oriented decomposition of functions in the communication system allows description, representation and enforcement of inter-media control relationships in a uniform manner. Accordingly, a *user level* connection is the setting up of communication between a set of user entities that implement an application. A connection enfolds one or more ***channels***, i.e., network paths, along which various data streams in a multimedia information flow. The user views a connection object as an end-point of communication while the network views the set of channel objects in the connection as providing independent data paths, each possibly with different characteristics. It is the responsibility of the communication system to compose the channels for a unified abstraction of connection visible to the user. See Figure 4. For example, a video telephone connection has two channels, viz., one for video data at 2 mb/sec (for compressed video) and the other for audio data at 64 kb/sec. And these channels may take different routes through the network based on bandwidth availability or may be multiplexed on the same route[3]. The communication system synchronizes the audio and video at the connection level by delivering them simultaneously to the user[4]. The QOS_{transp} is associated with a connection while a QOS_{net} is associated with each channel.

In general, the above model of transport level handling of media data (including that proposed in the EXPANSE network [2]), viz., moving the data through network channels and enforcing connection level end-to-end synchronization, allows certain desirable features in the communication system, as described in the next section.

3 Desirable features of communication system

As mentioned earlier, the communication system performs two functions that are specific to multimedia transport: i) generating QOS_{net} for various channels from the connection level QOS_{transp}, and ii) invoking control procedures to manage the interactions between channels in the connection. The functions (i) and (ii) are embodied into a transport protocol executed in the communication system. We present the advantages of object-oriented approach from this framework.

3.1 Micro-management of network resources

By partitioning the resource demands of a connection across its channels, the communication system can exercise fine granular control over the allocation of resources for the connection to satisfy QOS_{transp}. This is illustrated below (we assume without loss of generality that the underlying network uses packet switching; so a data segment is realizable by a sequence of packets).

The QOS_{net} specification for a channel has a direct relationship to the demand put on network internal resources in the channel implementation such as allocation of packet buffers and assigning packet scheduling priority during packet transport through the network [16]. For instance, a fixed and deterministic delay, as required for high fidelity audio, represents a strong constraint which puts a heavy demand on resources such as pre-allocation of packet buffers and assignment of higher packet priority. On the other hand, a large tolerance to delay variability, as with text transfer in catalogue browsing, represents a weaker constraint which puts a lighter demand on resources such as buffer allocation upon packet arrival and assigning low packet priority. Similarly, the extent of tolerance to data loss on a channel can determine the extent to which the network needs to exercise, say, 'forward error correction' procedures on packets transported through

[3] It may be noted that a restricted form of this functionality is used in narrow band ISDNs where a Primary Rate ISDN interface may be provided by means of a set of independant Basic Rate ISDN channels [6].

[4] The case of audio and video data independently multiplexed on the same route is different from the case where these data always take a single route and carry timing information alongwith for synchronization. The latter case is due to lack of a proper abstraction that separates these media at the transport level, so the network resource management is coarse and the synchronization procedures are inflexible.

the channel, thereby influencing the amount of packet processing needed in the network. In ATM networks for instance, the cell discard and launch priorities specifiable in cell headers [3] are derivable from the QOS_{net}. In general, the network appears as a 'programmable black-box', with the QOS_{net} specification supplying parameters to instantiate the network activities[5].

One approach to supporting multimedia communication is to extract strong delay guarantees from the network by specifying an appropriate QOS_{net}, whereby all the media data packets experience fixed and non-random delays [4]. Thus many of the transport requirements are implicitly satisfied by the network, thereby putting less burden on the transport protocol. The approach however entails wasteful use of resources since many applications do not need tight guarantees on the packet delay/loss experienced and/or available peak bandwidth.

Instead, the specification of QOS_{net} at a fine granularity, viz., at the channel level, allows resource guarantees to be provided only to the extent required by the application. For instance, a data that is generated in bursts need not be delivered with the same degree of burstiness unless the application requires this. So with traffic smoothing at the transport level, buffers need not be allocated in the network to meet the peak data rate from the application. Consider another case where data segments belonging to various media $y_1, y_2, \ldots, y_N$ are to be delivered simultaneously. The QOS_{net} specified by the transport protocol may indicate $min(\{\delta_i\}_{i=1,2,\ldots,N})$ as the acceptable delay variations on various network channels carrying the segments, where δ_i is the persistence duration of y_i. This can in turn relax the packet scheduling priorities in the channels.

We believe such an application-specific control on the network resource allocation [5, 8] is a desirable feature in future multimedia communication systems. It should be noted that this micro-management of the network does not imply that the application needs to know more about the underlying network than with conventional methods (such as those used with ISO models [7]); it simply means that the UNI allows a richer and canonical specification of the transport requirements of application data which are mappable into a set of network resource control parameters.

3.2 Richer semantics of data transport

The success or otherwise in setting up a connection can be specified at a fine granularity, as illustrated below.

Suppose a channel setup fails, say due to lack of adequate resources to guarantee QOS_{net}. The effects of this failure on the other channels in the connection are application-specific. In one extreme, the application may require that all channels in the connection be successfully set up; if one or more channels fail, the other channels which already have been setup be aborted. This requirement may arise from the media atomicity discussed earlier. Weaker forms of this requirement may allow the connection to exist from the application's point of view, though in a degraded mode. In a video telephone connection for instance, the conversation may continue on the audio channel even if the video channel fails.

A more general relationship between various channels in a connection may be specified using **AND** and **OR** 'syntactic connectives' on the channel setup activities. For instance, a catalogue browising application may allow a degraded form of a connection in which the channel to carry graphics image needs to be successfully set up along with either audio or text channels. Such control relationships are specifiable in QOS_{transp} (a similar solution has also been proposed in [18]).

A variant of the above notion is the level of sustained guarantees from the network on the channels already set up for an on-going connection. Sometimes, a quality degradation on a channel may affect the other channels. In a high fidelity stereo for example, a sustained data loss in one audio channel may cause this channel to be aborted, which in turn may cause the other audio channel also to be aborted. Such control relationships are enforced by the transport protocol, and would influence resource allocations in the network (e.g., bandwidth allocation) by virtue of specifying appropriate QOS_{net} parameters. Typically, an AND relationship on high quality channels (as in the above example) may require resource reservations for the worst case, while with OR relationship, on-demand resource allocations may suffice.

The transport protocol may (indirectly) control the policy

3.3 Better adaptation to data synchronization procedures

The synchronization relationships may be specified in terms of: i) the extent of data skew along the temporal axis tolerable between data streams, and ii) the temporal order in which the data streams may be presented to the user [8, 10, 12]. These relationships may capture the control interactions between the various data streams. In the multimedia window application for example, a data skew (or loss) may be acceptable in voice or text channels but not in both.

In cases where high level recovery from data skew (or loss) is required, recovery actions such as 'restricted blocking' may be specified by the application whereby a different data unit can be 'plugged' in the place of a missing data unit [10]. In a TV application for example, the loss of a picture frame may be handled by replaying the previous frame on the video channel (a similar recovery is possible

[5] In a given implementation of the communication system however, the generation of QOS_{net} parameters will be more network-specific than as discussed above. Usually, the transport level overhead in processing the data units (such as operating system process scheduling delays and CPU cycles required in end-systems) is also taken into account when generating the QOS_{net}. See [17] for a detailed discussion on resource allocation policies in end-systems and networks.

for audio channels also). Such application-specific recovery handlers are easily attachable to the transport protocols providing the channel abstraction in the communication system on a per-channel basis.

Similarly, the temporal ordering can be specified as constraints on the data delivery, ranging between strong to weaker forms, as required by the application. In the above example, the text and voice data may be presented in any sequence. Such an ability to specify weaker ordering constraints may manifest in more asynchronism in data transport in an underlying transport protocol [8]. Typically, these complex synchronization relationships may be specified using **AND**, **OR** and **NOT** 'syntactic connectives' on the channel data. Basically, such a specification indicates the inter-dependencies among various data streams, and may be included in QOS_{transp}.

Note that the object-oriented structure of the communication system leaves the door open for a variety of paradigms and protocols for resource management, data transport and synchronization. The paper describes, in the next section, the elements of UNI and the underlying structural components of communication system to provide such a flexibility and extensibility[6].

4 Object-oriented structure of communication system

Each media is handled by a distinct object that implements transport level functions to deliver the data stream in the media to the user entity (e.g., segmenting the data stream into suitable data units for transport). The objects are basically functional modules that can interact with one another over a pre-defined interconnect structure consisting of directed edges between objects. An object communicates with one or more other objects in the form of propagating *control data items* through appropriate edges and invoking executions in these objects.

The UNI incorporates the above 'dataflow programming' model of multimedia communication. The transport level constructs satisfy two requirements:

- Allow parallelism among object invocations and provide expressibility of the functional relationships between various media represented by objects

  A communication activity in the catalogue browsing service example may manifest in parallel handling of the text, graphics and audio data, and in simultaneous delivery of text and audio after the graphics display at the terminal;

- Be usable at various levels of media granularity for transport processing

  In a digital TV with sterophonic audio, the left and right voice data form a single media to be synchronized with the video; at a lower level, the left and right voice data form separate media for delay compensation purposes.

These constructs are based on the attribute-based naming of transport services, and are discussed below.

4.1 Attribute-based naming of objects

The communication system is characterizable by a set *Att* consisting of: ***attribute names*** which refer to transport features supported and a set of ***attribute values*** which refer to feasible instances of each feature, as specified by

Att = { (attribute name, {attribute value}) }.

A transport level object implementing a media is basically an instantiation of the various attributes. In other words, a user entity specifies a value for each attribute, i.e., the necessary (attribute,value) pairs, to create and manipulate a transport level object for each media.

The transport attributes of a multimedia object may include the average rate, burstiness, loss sensitivity, skew tolerance, and the coding techniqne employed for the various media data, specified as RATE, BURST, L_SENS, SKEW and CODE respectively. These attributes are illustrated below:

$$\begin{aligned} Att = \{ \quad & (RATE, \{150, 75, 25, 1.5, \ldots, 0.064, \ldots\}) \\ & (BURST, \{0.0, 0.1, 0.2, \ldots, 1.0\} \\ & (L\_SENS, \{0.0, 10^{-5}, 10^{-4}, \ldots\} \\ & (SKEW, \{10^{-3}, 2 * 10^{3}, 5 * 10^{-3}, 10^{-2}, \ldots\} \\ & (CODE, \{JPEG, MPEG, \ldots\}) \\ & \vdots \qquad \}. \end{aligned}$$

We illustrate[7] the transport level specification of objects in terms of these attributes nsing some examples:

Example 1: A user may specify an instance of the catalogne object discussed earlier as follows:

$$\begin{aligned} & cat\_obj = \\ & \{(AUDIO, \{(RATE, 0.064), (BURST, 0), (L\_SENS, 10^{-5}), \\ & \qquad (SKEW, 2 * 10^{-2})\}) \\ & (TEXT, \{(RATE, 0.010), (BURST, 0.2), (SKEW, 10^{-1}), \\ & \qquad (L\_SENS, 0)\}) \\ & (GRAPHICS, \{(RATE, 0.1), (BURST, 0.5), (SKEW, 10^{-3}), \\ & \qquad (L\_SENS, 0)\})\}. \end{aligned}$$

Example 2: Sample transport attributes of a video

[6] The study of different paradigms and protocols for resource management, data transport and synchronization is itself outside the scope of this paper.

[7] The data coding functionalities used in the transport of many media data such as JPEG and MPEG for video images are primarily used to reduce the bandwidth and storage requirements of the media data in the communication system. These functionalities have nothing to do with presentation of data to the application entities such as display of video images to human viewers in digital TV. So data coding falls in the multimedia transport layer.

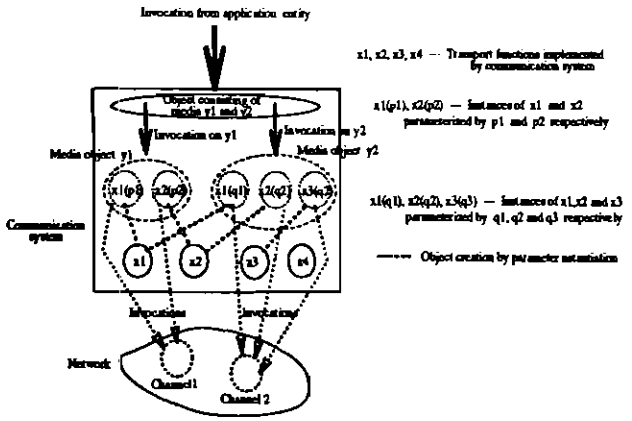

Figure 5: Illustration of object instances in the communication system

camera may be the type of video data encoding, the video data bandwidth and the tolerable skew in real-time presentation of video frames. A user may specify such a video camera object as follows:

$$vid\_obj = \{(CODE, JPEG),\ (RATE, 1.5),\ (SKEW, 0.066)\}.$$

Thus the communication system appears as a collection of object instances dynamically created, manipulated and then destroyed by user entities. See Figure 5 for an illustration.

In general, the use of attribute-based naming allows one to specify decomposition of an object to various levels of granularity of transport features. We expect the number of attributes required for typical objects to be less than 10. It should be noted that in an (attribute,value) pair, the value may itself be another (attribute,value) pair. Such a nesting of attributes allows fine-grained specification of various media. Since the attribute-based characterization is syntactic, i.e., does not require knowledge of the meanings of attribute names or their values, it can be uniformly used for different types of media[8].

4.2 Object definitions

The communication system implements the feature represented by an attribute name as a function. Users supply the attribute names to invoke various functions as objects with the attribute values as parameters.

An invocation on an object S may be abstracted as a function $\Phi(\underline{X_{N_S}})$ whose execution generates a control data item R to be sent to one or more other objects, where $\underline{X_{N_S}}$ is a set of attributes defined in the UNI to describe S ($N_S \geq 0$). The invocation may be represented as $R \equiv \Phi(\underline{X_{N_S}})$.

Let $\underline{X_{N_S}} = \{x_1, x_2, \ldots, x_{N_S}\}$. The attributes $x_1, x_2, \ldots, x_{N_S}$ are *orthogonal* to one another, i.e., the object executions that exercise different attributes do not interfere with one another. The orthogonality allows $\Phi(\underline{X_{N_S}})$ to be decomposed into a set of independent functions $\{\phi(x_i)\}_{i=1,2,\ldots,N_S}$, where $\phi(x_i)$ refers to the execution of Φ that deals with x_i. Consider the video camera example. The video skew handler module, specified as SKEW, and the video rate controller module, specified as RATE, are transport protocol drivers generating QOS_{net} specifications to dynamically control the channel delay behaviour and bandwidth allocation respectively [5, 14], while the video code handler module, specified as CODE, is a data encoder to map between video and JPEG/MPEG formats. An operation to create a transport level instance of camera object C, represented as $Create(\{(\text{SKEW}, 0.066), (\text{RATE}, 1.5), (\text{CODE}, JPEG)\})(C)$, may then be decomposed into three orthogonal invocations $Create(\text{SKEW}(0.066))(C)$, $Create(\text{RATE}(1.5))(C)$ and $Create(\text{CODE}(JPEG))(C)$.

The set of fnnctions $\{\phi(x_i)\}_{\forall\phi}$ constitutes the implementation of the feature represented by x_i. Consider a write operation to modify the parameters of C in the earlier example. It causes orthogonal executions: $Write(\text{SKEW}(\ldots))(C)$, $Write(\text{RATE}(\ldots))(C)$ and $Write(\text{CODE}(\ldots))(C)$. So the implementation of, say, SKEW
consists of $\{Create(\text{SKEW}), Write(\text{SKEW})\}$ and that of RATE consists of $\{Create(\text{RATE}), Write(\text{RATE})\}$.

The result R of the function Φ may be composed from the results returned by the various $\phi(x_i)$'s:

$$R \equiv \phi(x_1) \oplus \phi(x_2) \oplus \ldots \oplus \phi(x_{N_S}),$$

where $\oplus$ is a composition function. See Fignre 6. In the previous example of creating a camera object, R may be a concatenation of the outcomes — success or non-success — of the $Create(\text{SKEW})$ and $Create(\text{CODE})$ functions, with $\oplus$ representing the concatenation. As another example, consider the catalogue browsing application. A user entity specifies $Read(\{\text{AUDIO}, \text{TEXT}, \text{GRAPHICS}\})$ in its invocation to read the audio, text and graphics data from the communication system for updating the local window. The functions $Read(\text{AUDIO})$, $Read(\text{TEXT})$ and $Read(\text{GRAPHICS})$ may activate the audio driver, set up text buffers and map graphics images to window display respectively. The $\oplus$ function in this case is to combine the three results to generate a new update to the window subject to temporal ordering and atomicity requirements.

4.3 Representation of media relationships

The temporal relationships between the data units $d_1, d_2, \ldots, d_N$ belonging to the media $y_1, y_2, \ldots, y_N$ respec-

[8] The 'attribute-based naming' has been used in specifying general distributed services, as described in An Overview of UNP (Universal Naming Protocol) by L. Peterson in *Computer Communication Review (ACM SIGCOMM)*, May 1989. We expect that the user specified QOS_{transp} may be bound to specific object instances using some form of 'attribute-based name service' for transport functions.

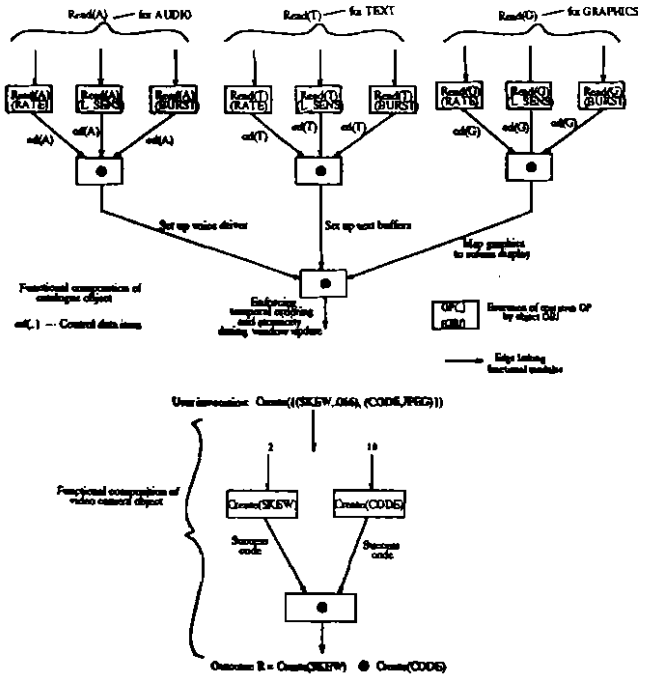

Figure 6: Illustration of object invocation and functional parallelism

tively may be specified as predicates on the delivery of data units in an interval, and may appear as an interconnect structure among objects implementing $\{y_i\}_{i=1,2,\ldots,N}$. Since the temporal relationships are invariant, the interconnect structure depicts a static pattern of flow of control data items. We discuss this below in the case of atomicity and ordering (see Figure 7):

Atomicity: AND and OR connectives on the attributes describing the media $y_1, y_2, \ldots, y_N$ may be used to express atomicity. A strong form of atomicity may be expressed as (y_1 AND y_2 AND ... AND y_N), while weaker forms may include OR connectives. The AND form maps to an interconnect structure in which an edge originating from each object $y_i|_{i=1,2,\ldots,N}$ is incident on the object $\oplus$. So $\oplus$ waits for control data items on all edges before delivering each d_i to the application. For example, a requirement that both video and audio should be delivered in a video distribution may be expressed as $\oplus \equiv (VIDEO$ **AND** $AUDIO)$. In the example of catalogue browsing, a requirement that graphics and text should be delivered without necessarily delivering the audio may be expressed as

$$\oplus \equiv (GRAPHICS \textbf{ AND } TEXT \textbf{ AND} \oplus') \text{ and}$$
$$\oplus' \equiv (\textbf{NOT}(AUDIO) \textbf{ OR } AUDIO)),$$

where NOT(X) generates a control data item if the incident edge on X does not have successful data arrival in an interval, $\oplus'$ generates a control data item when either

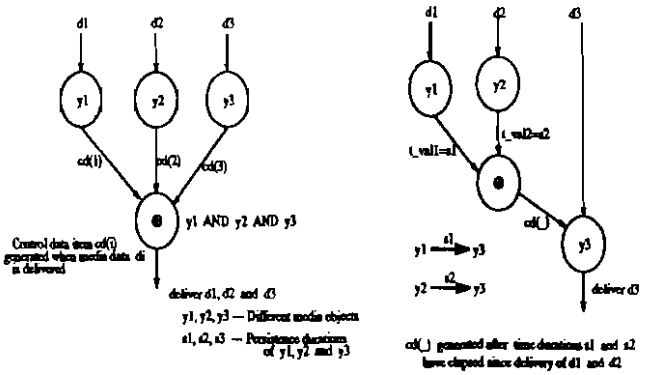

Figure 7: Mapping inter-media temporal relationships to inter-object 'dataflow'

NOT($AUDIO$) or $AUDIO$ generates a control data item, and $\oplus$ delivers the available media data after receiving control data items from $GRAPHICS$, $TEXT$ and $\oplus'$.

Temporal ordering: A 'happens before' relation on media data units can be used to express the temporal ordering requirements. A relation 'd_1 happens before d_2', denoted as $y_1 \xrightarrow{\delta_1} y_2$, means that, in a given interval, a data unit d_2 belonging to media y_2 is delivered after a data unit d_1 belonging to media y_1 is delivered and the persistence duration δ_1 of y_1 has elapsed[9]. Such a relation can be embedded into the object interconnect structure as follows. The object y_2 has a directed edge from the object y_1. The control data item generated by y_1 upon delivering d_1 carries a 'time value' $t\_val = \delta_1$ that causes y_2 to wait until the time duration t_val has elapsed before delivering d_2. For example, the catague browsing application may be expressed as $\{(GRAPHICS \xrightarrow{\delta_g} VOICE), (GRAPHICS \xrightarrow{\delta_g} TEXT)\}$ where δ_g indicates the persistence duration of $GRAPHICS$ media.

The atomicity specification is distinct from the temporal ordering specification, and manifests as an additional constraint on the delivery of media data units. For instance, a superposition of the relations y_1 AND y_2 and $y_1 \xrightarrow{\delta_1} y_2$ causes the communication system to hold the delivery of d_1 until d_2 becomes available, whereupon d_1 is delivered followed by d_2 with a time separation of δ_1.

The attribute specification describing the media and the atomicity/ordering requirements constitute the QOS_{transp}. Such a specification can be embedded into some form of programming tool for applications (such as the Heidelberg Multimedia Application Toolkit [19]).

[9]The 'happens before' relation is a modified form of Lamport's relation on occurrence of 'events' in a distributed system, as discussed by L. Lamport in the paper **Time, Clocks and Ordering of Events in Distributed Systems** in the journal *Communications of the ACM* (1978).

4.4 Exploiting parallelism in object implementation

An (attribute,value) pair is *self-describing*, i.e., contains the necessary information to exercise the corresponding feature. So a function implementing an attribute $\phi(x_i)$ can execute as soon as x_i is available. Since the various $\phi(x_i)$'s are orthogonal, there is potential for parallelism among them. Referring to Figure 6 for the video camera example, the functions *Create*(SKEW) and *Create*(CODE) can execute in parallel. Similarly in the example of catalogue browsing, the functions *Read*(AUDIO), *Read*(TEXT) and *Read*(GRAPHICS) can execute in parallel. So the various transport driver objects handling these media can execute simultaneously. The fine-grained parallelism arises due to functional decomposition of the object and due to the user's ability to invoke the decomposed functions.

Summary of our model

The attribute-based decomposition of multimedia data transport as objects in the communication system requires user entities to *explicitly* identify and represent functional relationships among various media data. This object orientation underscores two distinct aspects:

- A 'dataflow programming' style of structuring object interactions in the commnnication system;
- A declarative programming of the UNI in the form of predicates on the delivery of data.

These aspects allow flexibility in transport level programming of applications and interworking of the communication system with heterogeneous networks, as illustrated in the next section.

5 Architectural perspectives

In this section, we describe how our model of UNI can accomodate various desired transport features and allow seamless integration of existing networks and service environments into the communication system.

5.1 Functional architecture of communication system

Our architecture implements three layers of functions: i) channel abstraction in the network layer, ii) connection abstraction in the transport layer, and iii) user entities in the application layer. The functions (i), (ii) and (iii) may be projected on two different planes: control and data, as follows (see Figure 8).

Multimedia application layer: On the 'data plane', the function consists of generating and dissemintating data streams such as video, audio and text. On the 'control plane', it consists of mapping the data synchronization requirements (such as simultaneity in data delivery) into a specification of QOS_{transp}.

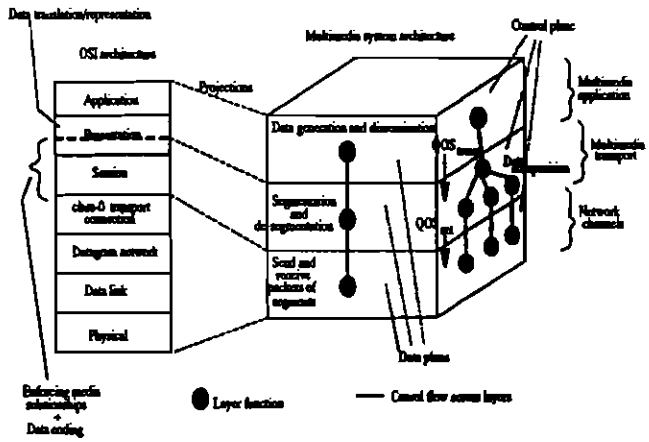

Figure 8: Functional architecture and relationship to OSI

Multimedia transport layer: On the 'data plane', the function consists of segmentation of data streams for transport over the network. On the 'control plane', it consists of decomposing the multimedia data into separate segments for dispatch over network channels and composing the segments received over varions channels for delivery to the application, subject to synchronization constraints specified in QOS_{transp}.

Network layer: On the 'data plane', the function consists of sending and receiving packets over the communication links. On the 'control plane', it consists of link level mechanisms to implement the channel abstraction, with the user specified QOS_{net} enforced in each of the channels.

The functional relationships among media that exist on the 'control plane' are not visible on the 'data plane'. For instance, the control action of a segment wait in a buffer pending synchonization manifests as a real-time pause in the flow of data on the 'data plane'. Similarly, the actions exercised by the transport entity on the 'data plane', say for error recovery and/or window management (e.g., sending acknowledgement to a data), do not appear on the 'control plane'. However, any delay in data arrival caused by such actions appear as a real-time pause in the temporal interval on the 'control plane'. The temporal relationships among a set of segments (e.g., AND) map on the 'data plane' one-to-one to a linearized sequence of various segments. So data segments of a media on the 'control plane' appear on the 'data plane' in the sequence given by this mapping.

5.2 Interworking with current networks

The communication system may exploit the network level facilities that are available in current networks by enfolding these facilities into the transport abstractions. We

illustrate this with the cases of audio-video conferencing and heterogeneous networks.

Audio-video conferencing

Our transport model can be used to support applications requiring audio-video conferencing on top of current networks that provide only audio conferencing facilities. We perceive this to be a future trend because audio conferencing support is easily providable by networks. For instance, ISDN network layer protocols have been evolved towards a conference service located at a central site and reached by user stations [6], with various stations interacting with one another over voice channels. However, the diverse data formats and coding potentially required for video transport make it less likely that video conferencing support from current networks will be available in the near fnture (even with H.261 data formats). Given this technological limitation, our model can encapsulate an audio conferencing facility into a channel, create a video conferencing facility with a customized data format and rate on another channel, and compose these channels together using an application-specified control relationship.

Heterogeneous networks

The basic facilities provided by some existing networks may not be exercisable at a fine granularity due to lack of compatibility between our UNI and the network facilities. For instance, a network may not provide bandwidth allocation for a channel as an attribute specifiable in QOS_{net}. In such networks, the transport protocol dealing with QOS_{transp} specifications can create a single channel for all the media data units and provide a degenerate form of QOS_{net} specification conforming to the characteristics of a given network. Since the resonrce management, data transport and data synchronization features are viewed from an end-to-end perspective in our model and the transport level enhancements made feasible with the model are only desirable elements, a functional incompatibility may not as such hamper the multimedia communication though it may not accrue the incremental benefits of using the model. In other words, an end-to-end transport spanning across heterogeneous networks may not be able to fully exploit the desirable elements of the model on these networks.

5.3 Relationship to OSI protocols

We compare the various functions in our architecture with the OSI architecture [7]. Refer to Figure 8. A network channel provides data delivery with datagram reliability (though QOS_{net} can control the extent of packet loss), and roughly corresponds to the ISO 'class 0' transport that provides no error recovery. The dispatching and receiving of data packets over multiple channels of a connection roughly map to the OSI session layer implemented using one or more 'class 0' transport paths.

Given a set of media data units available from the OSI session layer, the lower part of OSI presentation layer provides a mapping on the data before delivering them to the application. Typically, the mapping may involve data encoding (such as JPEG/MPEG for video)[10] and temporal ordering of data units (based on '$\xrightarrow{\delta}$' relations on data units specified in QOS_{transp}). The temporal ordering appears as a linearized sequencing of the data units over real-time at the application entities.

Thus, the OSI session layer implemented on 'class 0' transport paths and the lower part of OSI presentation layer are projected into the multimedia transport layer in our architecture[11]. Generation and dissemination of multimedia data fall in the OSI application layer and the upper part of OSI presentation layer.

We believe such a flexibility to interwork with existing networks and protocols is essential for the evolving multimedia communication systems.

5.4 Integration into object-oriented programming environments

The object-oriented structure of the communication system allows rapid prototyping of UNI software and extensibility of protocols, which is useful in the development and maintenance of multimedia systems. In addition, it enables a seamless integration of transport level components into object-oriented programming environments. In this context, various existing methods of object orientation in multimedia communication such as the use of library interfaces [20], the multimedia virtual circuits [21] and the operating system tool kits [22] are suitable for system level integration of multimedia by providing an 'object-oriented glue' to conventional network models, but these methods are restrictive for program level integration. In our approach, we provide an object-oriented structure of the communication system top-down, taken dowu to network level which, we believe, naturally blends into programming methods.

6 Related works

The structuring of multimedia data transport, as described in our paper, is quite similar to the 'functional signaling' approach proposed in existing works for broadband multimedia services [2, 23, 24] in terms of its heavy

[10] The media data coding functionalities encapsulatable in a multimedia communication system are unlike the upper part of OSI presentation layer functions that often deal with syntactic transformations on data for translation and representation of data at application entities (i.e., deal with data heterogeneity in the communicating end-to-end terminals).

[11] Existing OSI higher layer protocol implementations above the 'class 0' transport (such as 'class 4' transport protocol implementation) may be viewed as special cases of the transport and application layers in our architecture.

reliance on object-oriented modeling of the communication system. However, our work focusses more on the specification of control relationships among media and their implications on end-to-end and network resource management, whereas the existing works focus more on transport level signaling protocols that allow embodying a variety of control relationships. The work on structuring telecommunication switching software using object-oriented approaches [25] deals with some representative transport level intermedia relationships for complex multimedia applications. In comparison, our work deals with general tools and mechanisms for specification and exercising of such control relationships.

Also, our 'dataflow' model of structuring the communication system is quite different from the 'state machine' models suggested in [2] and [25] where functional relationships between various media are implicitly built into the 'state transition' procedures in the object internal structure. Given the need to tap the communication parallelism inherent in an application, our approach does not impose any additional complexity in the transport level view since the functional relationships between various media need to be built into the control flow of object executions in any chosen model.

The multimedia transport model suggested in [26] focusses more on extending OSI protocols to provide media synchronization and real-time delivery guarantees. Though the solutions suggested in this model are in general useful, the lack of object-oriented modeling of data transport in the OSI framework makes a communication system incorporating OSI protocols less flexible and/or extensible to deal with the complexities of multimedia communication.

7 Conclusions

This paper provided an object-oriented model of a multimedia communication system by decomposing the transport functions into orthogonal channels that carry the media data through the network and by providing a methodology for defining a set of control relationships that allow the media data to be composed in an application-specific manner. The object model enables many desirable features in the communication system such as management of network resources at finer granularities, providing richer semantics of data transport and better adaptation to media synchronization.

The paper described a model of UNI using attribute-based decomposition and invocation of transport functions. This results in a 'dataflow programming' view of the communication system. Using this model, the paper illustrated the architectural aspects of the communication system. The object model allows flexible interworking of a multimedia communication system with many existing networks. The model allows seamless integration of transport components into object-oriented programming environments.

The separation of the user level and the network level views of data transport has been advocated also in [2, 25]. In our work however, we encapsulate these views in a uniform object-oriented framework and use it to provide richer functionalities in the communication system[12]. Conceptually, the approach puts the onus of enforcing the control relationships between various data streams explicitly in the end-systems, with only support mechanisms in the network to control the data delay and/or loss in the network. This partitioning of functions however is not obvious in many existing models of communication systems.

The various elements of our transport interface have been incorporated in the prototype communication system currently being built at Kansas State University [8]. These elements are also supported, in one form or the other and in varying degrees, in the Heidelberg Transport System Prototype built at the IBM European Networking Center [9]. We believe that the transport interface may be integrated into an object-oriented programming environment with more ease, flexibility and extensibility than with conventional approaches.

Details of the transport interface such as the transport level primitives, the application structure and the underlying transport protocols are other interesting research issues.

References

[1] D. R. Spears. **Broadband ISDN Capabilities From a Services Perspective.** In *IEEE Journal on Selected Areas in Communications*, 1987.

[2] S. E. Minzer. **A Signaling Protocol for Complex Multimedia Services.** In *IEEE Journal on Selected Areas in Communications*, Vol. SAC-9, No.9, pp.1383-1394, Dec. 1991.

[3] M. E. Anagnostou, M. E. Theologou, K. M. Vlakos, D. Tournis and E. N. Protonotarios. **Quality of Service requirements in ATM-based B-ISDNs.** In *Journal of Computer Communications*, Butterworth-Heinemann Publ. Co., vol. 14, no. 4, pp. 197-204, May 1991.

[4] D. Ferrari. **Design and Applications of a Delay Jitter Control Scheme for Packet Switching Internetworks.** In Proc. 2nd *International Workshop on Network and Operating System Support for Digital Audio and Video*, Heidelberg, Nov. 1991.

[12] A fine granular separation of transport functionalities has also implications on the *Operations, Administration and Maintenance* plane of the broadband ISDN systems architecture such as billing of end-system users for the service actually enjoyed by them (viz., the number of channels active at various points in time and the QOS_{net} on the channels) and other 'admission control' functions on users [2]. In the video telephone example, a failure of the video channel may cause the customer to be billed only for the audio.

[5] K. Ravindran and V. Bansal. **Delay Compensation Protocols for Synchronization of Multimedia Data streams.** Accepted in Nov. 1992 for publication in *IEEE Transactions on Knowledge and Data Engineering*, Special issue on **Multimedia Information Systems** (available as *Technical Report* from Kansas State University, May 1992).

[6] CCITT Specifications. **Q931 ISDN User-Network Interface**, In Fascicle III (Red Book), 1988.

[7] U. Black. **OSI: A Model for Computer Communication Standards.** Prentice-Hall Inc. publ., 1991.

[8] K. Ravindran. **Transport Models for Synchronization of Multimedia Data streams.** *Technical Report 93-8*, Kansas State University, Jan. 1993.

[9] D. Hehmann, R. G. Herrtwich, W. Schulz, T. Schuett and R. Steinmetz. **HeiTS — Architecture and Implementation of the Heidelberg High-Speed Transport System.** In Proc. 2^{nd} *International Workshop on Network and Operating System Support for Digital Audio and Video*, Heidelberg, Nov. 1991.

[10] R. Steinmetz. **Synchronization Properties in Multimedia Systems.** In *IEEE Journal on Selected Areas in Communications*, Vol.SAC-8, No.3, pp.401-412, April 1990.

[11] D. Ferrari and D. Verma. **A Scheme for Real-time Channel Establishment in Wide-Area Networks.** In *IEEE Journal on Selected Areas in Communications*, Vol. SAC-8, No.4, pp.368-379, April 1990.

[12] T. D. C. Little and A. Ghafoor. **Multimedia Synchronization Protocols for Broadband Integrated Services.** In *IEEE Journal on Selected Areas in Communications*, Vol.SAC-9, No.9, Dec. 1991.

[13] D. Jordaan, M. Paterok and C. Vogt. **Layered Quality-of-Service Management in Heterogeneous Networks.** In *Technical Report*, IBM European Networking Center, 1993.

[14] H. Tokuda, Y. Tobe, S.T.C. Chou and J.M.F. Moura. **Continuous Media Communication with Dynamic QOS Control Using ARTS with an FDDI Network.** In Proc. *Symp. on Communication Architectures and Protocols*, ACM SIGCOMM, Baltimore (MD), pp. 88-98, Aug. 1992.

[15] C. Nicalaou. **An Architecture for Real Time Multimedia Communication Systems.** In *IEEE Journal on Selected Areas in Communications*, Vol.SAC-8, No.3, pp.391-400, April 1990.

[16] C. Vogt, R. G. Herrtwich and R. Nagarajan. **HeiRAT: The Heidelberg Resource Administration Technique — Design Philosophy and Goals.** In proc. *Kommunikation in Verteilten Systemen*, Munich (Germany), 1993.

[17] R. G. Herrtwich. **The Role of Performance, Scheduling and Resource Reservation in Multimedia Systems.** In *Operating Systems for the 90's and Beyond*, Lecture Notes in Computer Science, Springer-Verlag, July 1991.

[18] S. Nakano, N. Nagai, P. Sawada and O. Miyagishi. **Network Control Method and Human-Machine Interface Design for ISDN Multimedia Terminal.** In *IEEE Journal on Selected Areas in Communications*, Vol. SAC-8, No.3, pp.360-367, April 1990.

[19] T. Kappner, D. Hehmann and R. Steinmetz. **An Introduction to HeiMAT: The Heidelberg Multimedia Application Toolkit.** In *3rd Intl. Workshop on Network and Operating System Support for Digital Audio and Video*, LaJolla (CA), Nov. 1992.

[20] J. Ruckert, H. Schmutz, B. Schoner and R. Steinmetz. **A Distributed Multimedia Environment for Advanced CSCW Applications.** In Proc. of *Multimedia '90* IEEE Comp. Soc. and IEEE Comm. Soc., Bordeaux (France), Nov. 1990.

[21] W. F. Leung and et al. **A Software Architecture for Workstations Supporting Multimedia Conferencing in Packet Switching Networks.** In *IEEE Journal on Selected Areas in Communications*, Vol.SAC-8, No.3, pp.380-390, April 1990.

[22] D.P. Anderson and P. Chan. **Toolkit Support for Multiuser Audio/Video Applications.** In Proc. 2^{nd} *International Workshop on Network and Operating System Support for Digital Audio and Video*, Heidelberg, Nov. 1991.

[23] S. E. Minzer and D. Spears. **Principles for Defining a Signaling Protocol for Complex Connections in an ISDN.** In Proc. *International Conf. on Communications*, ICC, 1987.

[24] M. Kawarasaki and B. Jabbari. **B-ISDN Architecture and Protocol.** In *IEEE Journal on Selected Areas in Communications*, Vol. SAC-9, No.9, pp.1405-1415, Dec. 1991.

[25] A. S. Gopal and P. E. White. **A Paradigm for Modular Decomposition of Distributed Telecommunications Switching Software.** In Proc. *Intl. Conf. on Computer Communications*, pp.1153-1158, 1987.

[26] D. Shephard and M. Salmony. **Extending OSI to Support Synchronization Required by Multimedia Applications.** In *Computer Communications*, Butterworth-Heinemann Publ. Co., Vol.13, No.7, pp.399-406, Sept. 1990.