Group Communication in Distributed Multimedia Systems

Robert Simon[1*] Taieb Znati[1] Robert Sclabassi[2]

Departments of Computer Science[1], Neurological Surgery[2], Electrical Engineering[2] and Psychiatry[2]
University of Pittsburgh, Pittsburgh, PA 15213

Abstract

This paper presents AlphaDeltaPhi (ADP) Group *communication as a method of defining and conducting multi-process end-to-end connection level management in distributed multimedia systems. α represents the reliability, Δ the end-to-end delay and Φ is the synchronization interval for a group connection. In contrast to other group communication paradigms, an ADP-group is a related set of cooperating processes whose communication is supported by allowing a spectrum of quality-of-service and message delivery options to co-exist within the same group. Developed as part of our work with Multimedia Mednet, ADP-group communication is designed to provide appropriate connection management support and network control within distributed multimedia environments characterized by a heterogenous mixture of equipment types, network performance and user interaction.*

1 Introduction

The use of process groups and group-level communication offers a powerful operating-system level abstraction for developing and controlling cooperating processes in a distributed system. This is achieved by enabling multi-process communication to be done with single operations based upon single group identifiers, rather than by a series of operations to a (potentially unknown or partially known) set of individual processes.

This group abstraction is useful in multimedia communication which often involves sets of cooperating processes, acting as senders, receivers, or both. These sets of communicating processes may have to coordinate the delivery and presentation of their information. Examples of such coordination include audio and video synchronization for teleconferencing, or imaging and signal analysis synchronization from a medical database. Given this description a process group abstraction is a natural way to model communicating multimedia processes. An appropriate and efficient set of group primitives could reduce the complexity of call management[1], application development, and enable better control of network resources. The development of such primitives must be based upon a precise understanding of the semantics involved in multimedia group communication.

This paper presents a type of multimedia group management called ADP-group communication. Developed as part of the DIPCS (Distributed InterProcess Communication System) communication management system in Multimedia MedNet [8], ADP-group communication offers the advantages of group communication and connection management with semantics appropriate for a collaborative distributed multimedia system. These semantics reflect both the nature of collaborative work in a distributed medical environment and the system requirements for multimedia communication. Collaborative work in a medical environment (and many others) often requires that connections and communication be maintained over lengthy periods of time, with significant degrees of user and equipment mobility. There are also various constraints on call privacy and security. These semantics are reflected in group process management policies of DIPCS.

The objectives of conventional process group communication protocols are to provide support for properties such as atomicity, delivery ordering and causality in a reliable fashion. The semantics of these protocols are usually not required in distributed multimedia systems. The endpoints of multimedia communication

*This research is supported by grants from Hewlett-Packard, Inc., Computational Diagnostics, Inc, and the Ben Franklin program of the Commonwealth of Pennsylvania

[1] A multimedia *call* is a single session of logically related data streams, potentially between multiple senders and multiple receivers.

are often a heterogeneous mixture of sources and sinks, with differing performance requirements and communication needs. The communication streams associated with these sources and sinks have Quality-of-Service (QOS) requirements for message delivery, such as a maximum permissible end-to-end delay. Furthermore, many multimedia communication streams can suffer a certain amount of message loss and still perform correctly. Consequently, the "strong" semantics of conventional process group communication may be relaxed and still provide correct support for communication multimedia process groups. A form of these relaxed semantics are embedded in the DIPCS communication protocol. DIPCS is based on a network service model which can offer communication performance guarantees, and ADP-groups permit applications to specify a range of QOS requirements to coexist within a single process group.

The paper next discusses the Multimedia MedNet system. Part 3 describes the DIPCS multimedia device model. Part 4 defines the ADP-group structure, and explains the differences with other process group models in distributed systems. Part 5 shows how the QOS communication requirements for an ADP-group can be effectively supported at the network level, and part 6 offers some conclusions.

2 Multimedia MedNet

The Multimedia MedNet, currently being developed at the University of Pittsburgh, is a fully integrated and fully distributed CSCW system, designed to facilitate real-time decision making. MedNet extends the NeuroNet system, which is used to do intra-operative real-time monitoring. The NeuroNet system currently includes over 100 computer nodes installed in seven hospitals and multiple research laboratories at the University of Pittsburgh. The framework of the on-going MedNet project is based on the concept that all the various types of media, such as data, voice, video, and imaging, represent the types of data which must be integrated or synthesized into a coherent whole in order for appropriate decisions to be made. MedNet currently incorporates real-time digitized audio and extensive amounts of computer data generated by the output and analysis of neurophysiological monitoring processes during surgery, along with video streams transported by a parallel broadband analog network, into NeuroNet workstations. We are in the process of switching over to a purely digital system. DIPCS is being used as the connection management mechanism for the MedNet project.

One of the major activities within MedNet is distributed collaboration and monitoring between neurophysiologists, neurotechnicans and neurosurgeons during neurosurgery. These activities are of several types [6]. One type of communication can be described as multi-party interactive, where several participants discuss a particular situation or problem. This requires that the communication latency be short enough to not adversely affect the collaboration. Another type of activity is the long term monitoring of a situation, with only an occasional need for multi-way interactive communication. This type of communication can tolerate longer communication latencies and still perform adequately. We have observed a similar range of requirements for the *synchronization* of media types. Here we define synchronization to be simultaneous presentation of media types which are logically related. Some applications require a tight amount of synchronization, such as teleconferencing, while others, such as the coordination of video data with the digital display of neurophysiological data can tolerate much looser synchronization. Finally, it is important in this environment that a certain amount of privacy and communication security be maintained. Participants in a collaborative setting, and those whose actions are being monitored, should all be aware of each other, both to avoid needless effort in communicating and to maintain assumed standards of privacy.

These application requirements suggest that call management in this type of distributed multimedia system should allow the characteristics of the individual call, in terms of media type support and the use of the call (collaborative, monitoring, etc.), to determine the level of system support. This will provide a level of performance appropriate to the call itself, and make the most efficient use of system resources. The rest of this paper describes the DIPCS system and how it achieves those goals.

3 Multimedia Device Model

The DIPCS programming abstraction model for the underlying system is similar to some of the other programming abstractions which have been developed for distributed multimedia systems, such as [1] and [3]. The term "multimedia device" refers to any one of a heterogeneous collection of computing equipment and multi-sensory recording and display devices. Multimedia processing devices are often highly specialized (digitizing video cameras, speakers, etc.). As a result, multimedia devices tend to be uni-directional – they can be senders or receivers, but not both. This applies

as well to multimedia capture and display equipment which may be embedded in a conventional workstation. We will refer to multimedia devices either as *media sources* or *media sinks*. Media sources and sinks are distinguishable by the data stream that they operate on, such as JPEG compressed video, and the speed at which they operate, for example a monitor capable of displaying uncompressed HDTV or one only capable of displaying regular NTSC. Despite this heterogeneity, it has been recognized that it is desirable to treat, in a uniform fashion, all real-time data streams produced either from hardware or software [7]. This simplifies the task of connection management, since translating specific control sequences are now hidden from the application. DIPCS provides basic primitives for the declaration and control of multimedia devices which allow controlling process access and control based upon a simple device handle. A device is registered by invoking dev_cntl = **reg_device** (dev_name,dev_type,*buf,*ops), where *buf is the access point to the buffer space and *ops defines the types of controlling functions allowed on the device. A process accesses a device by invoking dev_id = **device** (dev_cntl). The returned handle allows sharing of devices, when physically feasible.

As the endpoints of multimedia communication, there is no requirement that a connected media source and media sink be able to process data at the same rate or even run at the same speed. This could be due to different device operating characteristics or application needs, or network conditions. To avoid overflow or starvation at the sink, appropriate actions must be taken by modifying either the device's performance or the network connection. Since device input or output performance may not be easily modifiable and since a single data stream from a single media source can be shared by multiple sinks, each of which may have different performance capabilities, it is desirable to be able to use the network to support different device communication performance requirements.

DIPCS represents each instance of a device's communication requirements to the network by the use of a *stream*. A new stream is created by invoking stream_id = **new_stream**(dev_id, direction, lockable, λ, α), where dev_id is the device handle, direction is source or sink, lockable is whether the device is sharable, λ is the message rate and α represents the percentage of a stream's messages that must be reliably delivered on-time. A logically related set of streams may be clustered together into a coordinated *stream set*. All initiation and modification of connections is done at the group level. The scheme to support a stream's performance is discussed in section 5.

A Multimedia *site* is defined as an instance of a coordinated stream set along with its associated devices, and may be written as the tuple <stream_set,device_set>. Each device which is part of a site can only be associated with a single stream. A physical device which belongs to more than one site can be associated with multiple streams. This allows devices to be shared and different QOS parameters to be specified, thus permitting flexible control of devices. The association of a site to a user process defines a member of an ADP-group.

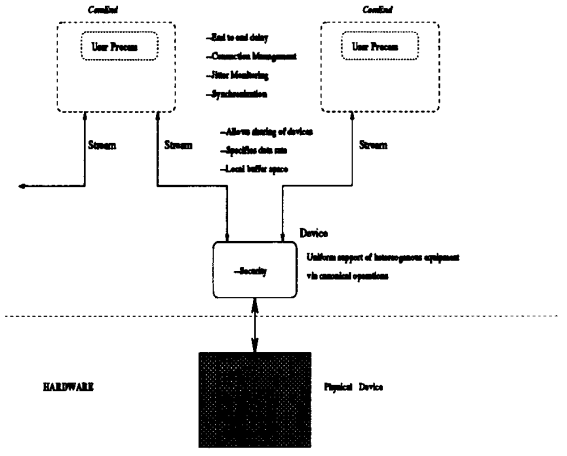

Figure 1: COMEND Strutcture

4 ADP-group Communication

The group level is the level at which connection management is done in DIPCS. Each group corresponds to a specific multi-party communication session or call. The group's performance characteristics represent the specific needs of a particular session, and not of a particular media type.

4.1 DMS Communication Requirements

Multimedia communication patterns are spread along an axis from highly interactive and symmetric peer-to-peer type communication, as in a collaborative systems, to client-server type communication, as in accessing a multimedia database. However, all distributed multimedia connections have timing and synchronization requirements. Some data must be sent from a source to a sink within a certain time, or else it

is of no value and should be considered lost. This time interval is called end-to-end delay. The end-to-end delay depends upon the application and usage. Multiway collaborative calls require a short delay, measured in hundreds of milliseconds, while the end-to-end delay for one-way viewing of a database can be on the order of several seconds. This implies that end-to-end communication delay values must be set on a per call basis, and each stream in a call have the same value.

The presentation of multimedia information also requires synchronization between various streams, to insure that logically related data are presented concurrently at the appropriate times. The synchronization problem can be mapped into a *physical* component and a *logical* component [7]. The physical component, supported at the network layer and below, is concerned with the timely and reliable delivery of packets. The logical components are application-defined synchronization points used to signal logical boundaries in the presentation of information. When such a point is reached all the data from each stream must be available for presentation. Connection management should provide the necessary tools for supporting the physical component of synchronization. This means that the network must be able to deliver all of the data required by a stream within each synchronization interval. This must also be done in the context of maintaining end-to-end delay guarantees.

4.2 ADP-Groups

Management at the call level is done in the context of an ADP-group. All of the entities participating in a single call constitute an ADP-group. ADP-groups contain a collection of entities, called *comend*'s (*com*munication *end*points) which are the endpoints of a communication tree. Each *comend* is a tuple <user, site>. A user may correspond to an actual human user of the system, or to some process which controls some other system activity, such as a database server. The user part of the tuple has two functions. First, it has the required associated system privilege to allow it to be able to use the computer and communication resources of the site. Second, the user is the entity at which all group membership actions are resolved, such as **join**, **quit**, **forward**, etc. The embedded characteristics of the site determine whether multiple users may be bound simultaneously to a site. This corresponds to whether the devices are physically shareable. Figure [1] shows the relationship of devices and streams to two *comend*'s.

An ADP-group G with k members is defined as a set $G =< \Delta, \Phi, comend_1, comend_2, ..., comend_k >$. Δ is the group's maximum end-to-end delay. The value of Φ is the time interval between two logical synchronization points, and so is dependent on the application's synchronization method. Segmenting the temporal axis into a sequence of synchronization intervals, information generated within Φ constitute a data frame presentable at the sink, as illustrated in Figure [2]. The duration of the interval is determined by the persistence duration of various data [4]. By imposing the end-to-end delay on the coordinated streams and limiting jitter, synchronization can be guaranteed. This method is adopted in the ADP-group structure and is described in section 5.

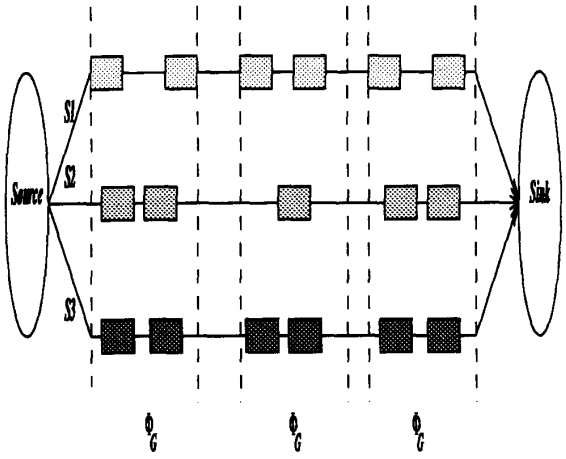

Figure 2: Synchronization Interval for Coordinated Streams

The initiator of the group sets the end-to-end delay requirements for the group. All members of the group must be able to meet this requirement, which also implies that all group members must have the appropriate level of network support. A group is formed by calling **new_group**(group_id) followed by **group_streams**(group_id, *site*, Δ, Φ).

In order to send or receive data messages from an ADP-group, a *comend* must be part of that group. ADP-group membership is only granted if all current members of the group explicitly approve the new member, even if a *comend* consists of only media sinks. An ADP-group is therefore a *closed group*. The closed group requirement stems from the nature of collaborative work and common assumptions about working environments and privacy, as well as our experiences with Multimedia MedNet. The principal is "If I say something, I should know who I am saying it to," and "If I hear something, I should know who else is hear-

ing it." Unlike some other process group abstractions, ADP-groups do not allow inter-group communication.

After a group has been formed, an application sends a request to join a group by calling **join (group_id)**. The group id may either be "well known" to a *comend*, or obtained by sending a character string to a Group Process Manager (discussed below), such as "Operating Room 8." Based upon the encapsulated stream types the appropriate source/sink connections within the network are made. These connections are between streams of opposite direction, i.e. sink to source, but of the same type, i.e. video. Multiple source streams can be connected to a single sink stream, as in multiple audio source streams into a single audio sink stream. This mixing mode is device specific. The group connection also causes the allocation of the appropriate network resources. Other group membership primitives are provided by DIPCS, such as **quit** and **forward**, as well as primitives to temporarily suspend the connection while maintaining group membership.

Each time a *comend* is instantiated it has an associated CM (Connection Manager) and GPM (Group Process Manager). The connection manager handles all group activities which require some form of *User Intervention* (UI), such as group membership actions, as well as the mapping and control of streams and devices. UI's may be handled entirely programmtically, or can be handled by sending a UI-REQUEST for a user response, i.e. by popping up a dialogue window on a workstation. This is what is done in MedNet.

The GPM handles the network control portion of ADP-group support. After a connection is agreed to, the GPM inserts the name of the group in to the network naming system and establishes a communication tree linking *comend*'s together. GPM's know about the other GPM's in the system. Based on the encapsulated characteristics of the media sink streams the GPM calculates the required level of network support.

4.3 Group Message Semantics

Several variants of process group management and group communication protocols have been proposed [5]. There exist a broad range of message ordering options in process group definitions, but despite different descriptions many process group protocols embed in their semantics of intra-group communication the following properties: atomicity, delivery ordering and causality. The atomic property guarantees an "all-or-nothing" delivery. Ordered delivery ensures that all recipients receive the same sequence of messages, while causal delivery guarantees the delivery sequence conforms to a predetermined ordering. These properties can be effectively supported by multicasting at the network level. The ADP-group semantics enforce a looser form of of these properties over a synchronization interval Φ.

ADP-groups have two classes of messages which can be sent — control and data. Control messages involve basic group management functions, such as requests to join or leave the group. Since all *comends* must agree to processes joining, or be notified of processes leaving, control messages are sent atomically in a request/reply format. A reply to a request guarantees that either all *comends* in the group have received it and replied to it, or no members will act on it.

Data messages in ADP-group communication have α-reliable semantics, where α reflects the degree of on-time delivery guarantees required by the underlying application. In order for the network to guarantee an α of greater than 0 the ADP-group communication network model uses resource reservation. α-reliable communication implies that the same quality of service may still be achieved at all participating sites, even though all active members do not receive the same messages. It also implies that multiple sinks of the same source data stream may receive a different QOS level. Since it is not realistic for the source to send multiple streams out to each of these, different QOS levels can be supported at the network level or at the sink level. The advantage of doing it at the network level results in better utilization of resources, since routes to sinks with lower QOS do not require as much of the network resources if the paths are at all disjoint.

5 Network Support for ADP-Groups

Network support for DIPCS is based on a resource reservation model. Based on the traffic specification, the network scheme either accepts or rejects the group connection. The scheme assumes that the network can produce a potential route or multicast tree capable of supporting the performance requirements of the streams. Route selection is outside the scope of this paper and is the subject of current research [9].

The network support policy guarantees that at each synchronization interval Φ enough packets from each stream arrive reliably, without overflow or underflow from any stream. It does this by insuring that each node in a stream's routing path supports over each synchronization interval the number of packets required from that stream for proper synchronization.

Communication management and specification for multimedia connections should allow for control at

both the source and the sink. However, the sink should specify the level of network support for a particular connection. There are several reasons for this. Since a source stream might be connected to multiple sinks, each of which could have a different playback rate, it makes sense to allow the sink to specify the data stream rate, not the sender. The sink itself may be coordinating the activities of several sources, each of which may not be aware of the existence of the other sources. Additionally, each sink is fully aware of its own operating characteristics and requirements. Thus a ADP-group is a spanning tree, or series of trees, with each source connected to multiple sinks. It is up to the GPM to determine the matching of source to sink streams and the network support required. One resource reservation protocol which makes sink-based reservations with differing requirements within a routing tree is described in [10].

5.1 Channel Acceptance Procedure

Consider an ADP-group G composed of k members $g_1, \ldots, g_k$, and assume that there is a set Γ of q source streams originating at g_1, $\{\gamma_1, \gamma_2, \ldots, \gamma_q\}$. Group G has an associated end-to-end communication delay Δ_G and synchronization interval Φ_G. Each of the q streams may take at least a partially different path to a given group member g_j. To setup and verify network support for the Γ streams to g_j, a two step procedure is applied to each $\gamma_i, 1 \leq i \leq q$. The first part verifies a rate match between source g_1 and sink g_j, while the second part is an acceptance procedure which verifies that all the nodes on the routing path from source to sink can support a stream's requirements, given each node's previous commitments.

In part 1, a stream γ_i from g_1 specifies a rate of packet production λ_{i,g_1}. The corresponding stream from g_j specifies a possibly *different* rate of λ_{i,g_j}, and α_{i,g_j}. If the connection is a valid one we must have $\lambda_{i,g_1} \geq \alpha_{i,g_j} \times \lambda_{i,g_j}$. If this condition is verified then the connection will support the performance specified by g_j.

Part 2 uses a modification of the schemes presented in [2] and [11] to guarantee network support at each node on the routing path for each stream. The modified scheme is presented below; see [2] and [11] for a rigorous proof of deterministic guarantees for channels accepted for service at a node, based on worst-case packet arrival patterns and maximum acceptable per-node per-channel time delays.

Consider the situation at node n on the routing path P_i for a stream γ_i. We assume that each node on the routing path will support the same per node delay. At node n, this delay is $\delta_i^n = \frac{\Delta_G}{|P_i|}$. Over each time interval δ_i^n the node must be able to guarantee a packet delivery rate pdr_i

$$pdr_i = \left\lceil \frac{\Delta_G}{\Phi_G} \right\rceil \times \Phi_G \times \alpha_{i,g_j} \times \lambda_{i,g_j} \qquad (1)$$

where $\Phi_G \times \alpha_{i,g_j} \times \lambda_{i,g_j}$ is the number of packets from stream i which g_j must receive over each synchronization interval. If the node cannot support this rate than synchronization at the sink cannot be maintained.

Node n verifies that enough resources are available to support this channel, given its current commitments to other channels. The information n receives for γ_i is the tuple $[\delta_i^n, pdr_i]$. The verification scheme is based on the relationship among the maximum number of packets received by a given node over a given time interval, the per-packet processing time and the delay a packet may suffer at that node, along with the runtime support mechanism discussed in section 5.2. The maximum delay suffered by a packet at each node cannot exceed its associated δ_i^n. This delay is determined by the maximum number of packets received by a given node over a given time interval, and the per-packet processing time. In formulating the guaranteed acceptance policy we consider the worst case arrival pattern; all packets from all accepted connections arrive simultaneously at node n, and all the connections are operating at their maximum rate.

Assume without loss of generality that the per-node processing time for all packets is ρ. Let 1, 2, ... N be the streams currently supported by node n such that $\delta_i^n \leq \delta_j^n$ if $i \leq j$, for all $k = 1, 2, \ldots N$. Then, if for all $k = 1, 2, \ldots N$,

$$\sum_{i=1}^{k-1} W(i,k) + pdr_k \times \rho + \rho \leq \delta_k^n$$

where δ_k^n is the delay interval for stream k, then $delay_{p^k} \leq \delta_k$, where $delay_{p^k}$ is the maximum delay observed by a packet, p^k, from connection k, and $W(i, k)$ ($i < k$), represents the amount of time to service the maximum number of deadlines from i that occur during any interval of size δ_k^n, in the worst case.

The worst case pattern of arrivals of two stream connections i and k occurs when the packets from connection i arrive in a way such that the deadline of the first packet coincides with the arrival of a packet from k and continue to arrive at their maximum predefined rate, as illustrated by Figure [3]. Based on this observation, the quantity $W(i, k)$ may be computed by counting the maximum number of deadlines

that can occur during an interval of size δ_k^n. The number of deadlines in this interval is distributed among the three regions, R_1, R_2, and R_3. The first region, R_1, contains pdr_i deadlines. The cumulation of servicing these deadlines specifies the size of the region. The second region, R_2, contains the number of intervals of size δ_i^n completely embedded in an interval of size $\delta_k^n - R_1$. The amount of time required to service the deadlines that may occur within R_2 may be computed as $\left\lfloor \frac{\delta_k^n - R_1}{\delta_i^n} \right\rfloor \times pdr_i \times \rho$. Finally, R_3 can be expressed as $R_3 = \delta_k^n - R_1 - R_2$. A deadline from a stream i will only need to be serviced during R_3 if $\delta_i^n - pdr_i \times \rho \leq R_3$, in which case the number of deadlines from i that will require service during R_3 is $\lfloor R_3 - (\delta_i^n - pdr_i) \rfloor + 1$. Therefore, the value of $W(i, k)$, in the worst case, during an interval of size δ_k^n, may be expressed as:

$$W(i,k) = R_1 + \left\lfloor \frac{\delta_k^n - R_1}{\delta_i^n} \right\rfloor \times pdr_i \times \rho + (\lfloor R_3 - (\delta_i^n - pdr_i) \rfloor + 1) \times \rho$$

Based on the above discussion, a node n may use the following procedure to decide whether to accept a new stream connection. Let C_n be the set of ADP-group streams currently supported by n. Assume that $|C_n| = N - 1$. Furthermore, let k be a new stream requesting to be established and assume that $\delta_1^n \leq \delta_2^n, \ldots \delta_{k-1}^n \leq \delta_k^n \leq \delta_{k+1}^n \ldots \leq \delta_N$. The new stream k is accepted if and only if:

$\sum_{i=1}^{k-1} W(i, k) + pdr_k \times \rho + \rho \leq \delta_k$, and

$\sum_{i=1}^{k+j-1} W(i, k+j) + pdr_{k+j} \times \rho + \rho \leq \delta_{k+j}$, for all j = 1, 2, ... (N − K).

5.2 Runtime Support Mechanism

The run-time support strategy provides the mechanism for network support of each ADP-channel. This is achieved by maintaining the traffic rate at the specified levels across the channel's path and guaranteeing the on-time delivery requirements of the supported channel. This goal is achieved by two scheduling policies. The first policy, *inter-node traffic regulation*, preserves the per node traffic rate specification. The second policy, *packet scheduling service*, aims at servicing packets within their per node delay bounds.

Traffic regulation between intermediates nodes is achieved by maintaining a *waiting room* for each channel. A packet entering the network for the first time

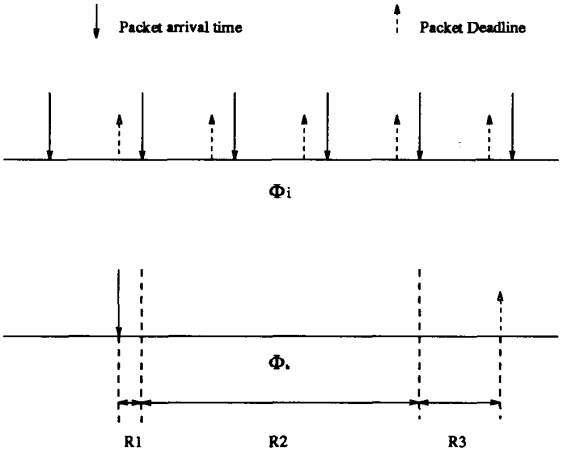

Figure 3: Worst case packet arrival pattern for streams i and k

is immediately eligible for service at the intermediate node, and its eligibility time is set to the current time. As the packet gets forwarded from one intermediate node to the next one along the path, its eligibility time is augmented with the channel's per node delay value, δ. At given node, a packet can only become eligible to be serviced when the current time matches the eligibility time. The eligibility mechanism used by the intermediate nodes scheduling policies prevents clustering of packets at a given intermediate node and guarantees that the node-to-node specified traffic rate is preserved along the path.

To support servicing packets within their per-node delay, which is a function of δ and the processing time at the node, the intermediate nodes maintain two service queues. The first service queue is for *basic traffic*. This traffic is constituted by the α portion of the traffic rate specified by each stream currently accepted at a node. The ADP-group framework uses a moving window based policing mechanism to enforce rate control at the edge of the network. All other traffic is called *enhancement traffic*. This is traffic generated from sources such as the 1-α portion of traffic from supported ADP-group streams, or best effort traffic.

The basic traffic queue has the highest priority. The scheduler provides a non-preemptive service to all packets in the basic traffic queue. Packets with the smallest per node delay bounds are serviced first. Ties among eligible packets are broken based on the time these packets were inserted into the service queue. When the basic traffic queue becomes empty, the scheduler moves to service any packets in the enhance-

ment traffic queue. Packets in this queue are also serviced based on their delay bounds. Service at the enhancement traffic queue may be interrupted after completing the transmission of a packet, if a packet arrives at the basic traffic queue.

6 Conclusions

This paper has presented ADP-groups and the DIPCS as a way of controlling multimedia communication in a distributed system. Based on our experience in developing Multimedia Mednet we have argued that communication management in a distributed multimedia system should have performance requirements, such as end-to-end delay and the synchronizatin interval, which depend upon the context of a call and not the encapsulated types of media streams. ADP-groups form the basis of communication and connection management by defining a closed group which supports a heterogeneous equipment base and communication performance needs. Basic canonical group communication operations appropriate for a collaborative system are defined. In contrast to some other semantics for message delivery in process groups, message delivery causality and atomicity are only needed for control messages; intra-group data messages have α reliable semantics, where reflects the percentage of packets which must be delivered reliably from each stream. We have also described a network scheme for supporting α reliable semantics. Future work includes expanding Multimedia MedNet to a purely digital system, as well as research into routing issues in distributed multimedia communication.

7 Acknowledgement

The authors wish to thank Brian Field, Andre Srinivasan, and the anonymous reviewers for numerous helpful comments and suggestions.

References

[1] Bellcore Information Networking Research Laboratory, "The Touring Machine System," *CACM*, Vol. 36, No. 1, pp. 68-77.

[2] Field, B., and Znati, T. "α-Channel: A Network Framework to Support Real-Time Performance Guarantees," *IEEE JSAC*, Vol. 11, No. 8, August 1993.

[3] Hill, W., Ishizaki, A., "A Call Model for Distributed Multimedia Communications," Hewlett-Packard Labs Technical Report HPL-93-06.

[4] Herrtwich, R.G., "Timed Data Streams in Continuous Media Systems," in TR90-017, I.C.S.I., Berkeley, CA. May 1990.

[5] Liang, L., Chanson, S.T., and Neufeld, G.W., "Process Groups and Group Communication: Classifications and Requirements," *COMPUTER*, Vol. 23, No. 2, Feb. 1990, pp. 56-67.

[6] Nardi, B., Schwarz, H., Kuchinsky, A., Leichner, R., Whittaker, S., and Sclabassi, R., "Turning Away from Talking Heads: The Use of Video-as-Data in Neurosurgery," *InterCHI '93*.

[7] Nicolaou, C., "An Architecture for Real-Time Multimedia Communication Systems," *IEEE JSAC*, Vol. 8. No. 3, April 1990, pp. 391-400.

[8] Sclabassi, R.J., Leichner, R., Kuchinsky, A., Krieger, D.N., and Prince, F., "The Multimedia Medical Monitoring, Diagnosis, and Consultation Project," in *Proc. of the HICSS*, Vol 24(3), pp. 717-723, 1991.

[9] Simon, R., and Znati, T., "Routing and Path Establishment in Integrated Networks for Distributed Multimedia Systems," in *Proceedings of the 1994 Pacific Distributed Multimedia Systems Workshop*, Taipei, Taiwan, February, 1994, pp. 131-148.

[10] Zhang, Lixa, Deering, S., Estrin, D., Shenker, S, and Zappala, D., "RSVP: A New Resource ReSerVation Protocol," *IEEE Network*, Vol. 7, No. 5, September 1993, pp. 5-18.

[11] Znati, T. and Field, B., "A Network Level Channel Abstraction For Multimedia Communication in Real-Time Networks", *IEEE Trans. on Knowledge and Data Eng.*, Vol. 5, No. 4, August 1993, pp. 590-599.