Session V: Communication II

Chair: Domenico Ferrari, University of California at Berkeley and ICSI

This session continued the discussion of network support for continuous media. While the first talk described a scheme for communication with performance guarantees in a packet-switching environment, the other three presented real-time protocol designs or implementations.

The first talk was given by Brian Field of the University of Pittsburgh, co-author with Taieb Znati of the paper "Alpha-Channel, A Network Level Abstraction to Support Real-Time Communication." The scheme proposed in it is based on the concept of "alpha-channel," a connection with one parameter, alpha, which represents the fraction of packets that must be delivered on time, i.e., before a client-specified deadline. Alpha = 1 corresponds to deterministic delay service, alpha = 0 to best-effort service, and alpha between 0 and 1 to statistical delay service. Like other approaches previously described in the literature, this is based on worst-case arguments, per-node rate control, and deadline scheduling with priorities (no best-effort packet is transmitted by a node if there is even a single deterministic packet in it).

The main differences between the scheme and, for example, the one on which Berkeley's Tenet protocols are based are the traffic characterization (which in the alpha-channel method has no burstiness parameter), and the establishment procedure (the source chooses the path, and asks the nodes on it to verify that they have sufficient resources for the new connection), while the rate control mechanism coincides with the one proposed by the Tenet Group at the Berkeley Workshop to control jitter in simple networks.

Comparisons with other schemes, that is, with Virtual Clock and Stop-and-Go, were the subject of some discussion at the end of the presentation. The question was also raised of what applications might want a value of alpha between 0 and 1. Clearly, no client would prefer statistical to deterministic delay service if the two services had identical prices: Statistical bounds can only be tolerated in certain less demanding applications if they cost less than deterministic ones. Another topic that was debated at some length was the usefulness that the application mark packets as droppable or undroppable by the network. Considerable skepticism was expressed by some participants about the assumptions that most applications have packets that are more important than the others, that they know which packets these are, and that the network can really choose which packets to drop.

The second paper in the session, "An Implementation of the Revised Internet Stream Protocol (ST-2)," by Craig Partridge and Stephen Pink, was presented by Stephen Pink of the Swedish Institute of Computer Science. The speaker told the audience about the experiences he and his co-author gathered while implementing ST-2 in the 4.3BSD kernel. The philosophy and the good and bad features of ST-2 were evaluated; among the good features, the speaker listed the information (about alternative requests that could be met) the network gives a client whose request cannot be met,

and the ability a source has to specify the destinations of multicast messages; those listed among the problems were the almost complete separation of data and control paths (which may cause major difficulties when the server is recovering from certain failures), and the very large and extendible number of performance parameters to be specified.

The authors' statement that the need for state information to be kept along the path of a real-time connection, which is a fundamental aspect of ST-2's philosophy, is still "a subject of hot debate in the networking community" was a bit surprising, as this need was not seriously challenged by any of the workshop's attendees, unlike what happened in Berkeley during the previous edition of the workshop.

The discussion provided further details about the implementation, and some of the difficulties that were encountered in doing it within 4.3BSD (in particular, the modifications that had to be made to the "connect" system call). The code consisted of about 6,500 lines of C, in spite of the absence of the resource management functions (which ST-2 does not specify); the implemented version is expected to run on top of a protocol that provides performance guarantees and to capitalize on the resource management mechanisms of that protocol.

The last two talks in the session were concerned with two of the four protocols included in the prototype version of the Tenet real-time protocol suite being developed at the University of California at Berkeley and ICSI. The first of the talks was given by Anindo Banerjea of the University of California at Berkeley, and was based on the paper "The Real-Time Channel Administration Protocol" by him and Bruce Mah. The speaker described the structure and some of the features of the implementation of RCAP, the protocol in the suite to be used to set up and tear down real-time channels. A channel in the prototype Tenet suite is characterized by a minimum and a minimum-of-averages throughput bounds, a deterministic or statistical delay bound, an optional deterministic delay-jitter bound, and a loss probability bound. These bounds are specified by the client, and do not have to obey any non-obvious constraints in order to be acceptable by the service.

RCAP receives the values of these performance parameters and traffic specifications (coinciding with the throughput bounds) from the client, and tries to build a connection from source to destination that will provide the requested guarantees. A mechanism that establishes such a connection, or determines that it cannot be created along that path, in a single round-trip even in a complex internetwork was described in the talk, which covered the other functions of RCAP (channel teardown, status reporting) as well.

The question of call-blocking and of the vastly insufficient information that a yes/no answer by the establishment procedure gives the requesting client were discussed by several participants. An approach similar to the one proposed by ST-2, which provides the client with an alternative set of values for the performance parameters, was felt to be more convenient by some of the attendees. The speaker pointed out that what he had described was the prototype implementation of RCAP, which has been purposely kept simple for ease of experimentation. There is nothing in the design of RCAP that prevents the later introduction of much more sophisticated dialogues and negotiations between client and server.

The second of the Tenet talks in the session, which was also the last, was given by Bernd Wolfinger of the University of Hamburg. He co-designed the CMTP protocol while on sabbatical leave at ICSI, and described it in a paper co-authored with Mark Moran ("A Continuous Media Data Transport Service and Protocol for Real-Time Communication in High Speed Networks"), on which his talk was based. CMTP is one of the two transport protocols in the Tenet real-time suite; it is oriented towards the transmission of continuous-media (voice, high-quality sound, video) streams, while its companion protocol, RMTP, is message-oriented.

The service that CMTP provides, called CMTS, consists of the unreliable, periodic, and sequenced transfer of continuous-stream data units with guaranteed throughput, delay, and loss probability bounds. The client interface is fundamentally different from that of RMTP; the performance parameters and traffic characterization have purposely been designed for continuous-media applications, including descriptions of the burstiness due to compression and the allowed workahead.

CMTP translates the client-specified traffic and performance parameters into those that the underlying real-time network protocol, RTIP, expects. RTIP is relied on to provide the performance guarantees; since the nature of real-time communication precludes the use of the mechanisms for error control and flow control that account for most of the overhead and the size of traditional transport protocols, CMTP is a fairly lightweight protocol.

An aspect of CMTP that attracted quite a bit of attention during the discussion was the mechanism to start, sustain, and stop the transfer of stream data units. Questions were concerned with buffer management, synchronization, the desirability that the receiver communicate with the sender before and/or during transfers, and the handling of unexpected situations, such as a full buffer on the sender's side or an empty buffer on the receiver's. For most of these questions, the answers were easy: CMTP relies on the real-time services offered by RTIP, and can ask for guarantees that most of these situations will not occur, or will occur only with a very small frequency; those situations that depend on the behavior of the application can usually be handled by making allowances in the design for the inevitable uncertainties about that behavior; finally, recovery from failures of the underlying service will heavily rely on the recovery facilities of that service.

Not discussed, because of the shortage of time, were a number of intriguing issues; for example, the best transport protocol arrangement for real-time traffic: should there be a single universal transport protocol (e.g., TP++), a few protocols to handle different types of traffic (e.g., RMTP for message-oriented traffic, and CMTP for continuous-media traffic), or media-specific protocols (e.g., one for video, such as PVP, one for voice, one for high-fidelity sound, and so on)?

Note: Stephen Pink's and Craig Partridge's paper is not contained in these proceedings. For information about their work contact steve@sics.se.

α-Channel, A Network Level Abstraction To Support Real-Time Communication

Brian Field and Taieb Znati† [3]
Computer Science Department
(† Telecommunications Program)
University of Pittsburgh
Pittsburgh, PA 15260

Abstract

In this paper, we propose a network level abstraction called α-channel to support the requirements of real-time applications, based on resource reservation and fixed routing. An α-channel represents a simplex, end-to-end communication channel with guaranteed service characteristics, between two entities. Based on the value of α, we define three classes of α-channels, namely a *deterministic* α-channel (α = 1), a *statistical* (0<α<1) and a *best-effort* (α=0). The semantics of these classes derive from the percentage of on-time reliability that need to be supported by the α-channel.

The primary attribute supported by the α-channel is the *on-time* reliability. At the α-channel establishment phase, the sender specifies the *end-to-end delay* of each packet, the *maximum number* of packets to be generated during this delay, and the on-time reliability required. We described the basic scheme that our model uses to verify the feasibility of accepting a new α-channel with guaranteed performance. A formal description of the proposed model and a sketch of the proof of its correctness is provided. We also present the preliminary results of a simulation experiment implementing the basic functionalities of the proposed scheme.

1 Introduction

Recent advances in VLSI technology led to the development of innovative packet switching architectures such as Fast Packet Switches and Photonic Switches, and created opportunities for a new class of *real-time* applications [1].

The performance requirements of these applications span a wide range, in terms of throughput, reliability, jitter bounds and delay characteristics. The "best-effort paradigm" offered by the Internet architecture, which proved to be very successful in the realization of a universal network in a heterogeneous environment is not adequate to support real-time traffic. The network protocol offers no guarantees about timely, reliable, or ordered packet delivery [2]. Circuit switching, which is the standard method for providing real-time performance, does not optimize the utilization of the network resources, and may be very inadequate for short transmissions, where the connection setup overhead is prohibitive with respect to the duration of the transaction. Furthermore, the scheme offers no provision for specifying data rates or limits on delays, and supports no guaranteed performance.

More recently, a number of schemes which aimed at providing performance guarantees based on virtual-circuits, resource reservation and fixed routing have been proposed [3,4,5,6]. A rate based network control system which provides guarantees for average throughput was proposed by Zhang [5]. The mechanism used is based on the concept of the "virtual clock", and aims primarily at regularizing the data flows according to the average throughput. The scheme does not provide guarantees to support end-to-end deadlines, and does not support an adequate buffer management scheme to limit packet loss due to buffer overflow. Another project, Multiple Congram Oriented High Performance Internet Protocol (McHIP), aims at providing variable grade services with guarantees for communication across subnets with diverse capabilities [4]. Subnets are characterized by a set of performance parameters, and their interconnection is achieved based on high speed gateways with all data delivery functions implemented in hardware. Furthermore, each subnet may have a designated resource server which keeps record of the established channels and the available resources. The network makes explicit resource allocation decisions at the time of connection establishment based on the requirements specified by the user. The authors, however, do not provide a clear definition of the connection semantics and the scheme used for implementing these connections.

The DASH communication system provides an abstraction called Session Reservation Protocol (SRP) [6]. An SRP is a simplex stream characterized by a set of performance, reliability, and security parameters. The model, designed to be independent of the transport protocol and compatible with the IP protocol, is based upon the "linear bounded arrival process". The DASH resource model describes a set of primitives that allow a user to express workload characteristics and performance requirements. A negotiation algorithm embodied in the SRP allows the user to negotiate reservation of the distributed resources.

Ferrari and Verma have proposed a real-time channel that guarantees end-to-end performance. The real-time channel is viewed as a binding contract between the client and the network. The client describes the characteristics of the traffic and specifies a set of performance requirements to be supported by the network. Based on these specifications, the network attempts to create a connection with fixed routing and performance bounds for each node along the path from source to destination. This is accomplished by computing the best performance guarantees for each node along the path, and making tentative resource reservations. The computation of the offered guarantees is based upon processing power, buffer capacity and the bandwidth of the links, and takes into consideration the requirements of the currently guaranteed real-time channels. Based on the performance guarantees accumulated and the guarantees requested, the destination channel determines whether the new channel is accepted or rejected. In the former case, the destination node adjusts the guarantees of each node along the path to the required guarantees to support the new channel, and sends a confirmation message along the same path to reserve the required resources. By controlling admission of new channels, the network ensures that it can meet its guarantees.

To support the requirements of real-time applications, we propose a network level abstraction called α-channel. An α-channel is a communication channel between a source and a destination, characterized by a set of performance attributes which reflect the requirements of the application. Upper layers, which view the α-channel as an end-to-end abstraction, specify the requested performance characteristics in quantitative terms when requesting the establishment of the connection. Based on the performance specifications requested by the upper layers and the current status of the network, the underlying delivery system either accepts and guarantees the specified performance attributes or denies the request.

Our scheme differs from the scheme described above primarily in terms of the specifications characteristics of the real-time channel, the mechanism used to establish a new channel, and the mechanisms used to maintain the specified rate among the node along the routing path.

In the first section of the paper, we introduce the concept of the α-channel. The proposed resource reservation model used to guarantee the requested performances is described in section 2. Issues related to the the management of the queues of a switching node are discussed in the last part of this section. The last section is dedicated to the description of the preliminary result of a simulation experiment.

2 α-Channel, A Network Level Abstraction

Without proper characterization of the network components and without any resource allocation, it is extremely difficult, if not impossible, to provide predictable performance [11]. Two approaches are used to provide network performance guarantees. The first approach is to over-engineer the network to the extent that an application is certain to get

the resources it needs. Thus, the applications can be unconstrained in its resource usage and still receive the guaranteed level of performance. In a high speed network, this approach would require prohibitively large amount of resources.

The second approach involves monitoring and controlling resource allocation and usage for each application. This approach requires that an application specifies its resources needs a priori, and unless its needs can be met, it is blocked or rejected. Once started, mechanisms are provided to ensure that the application does not use more resources than requested. Because every application uses only its share of resources, all performance needs can be met. This approach is adopted in our definition of the α-channel.

2.1 Specification of the Semantics of α-channels

An α-channel represents a simplex, end-to-end communication channel between two entities, a *sender* and a *receiver*. The sender requests an α-channel by specifying the real-time performance characteristics required by the application. The primary performance attribute supported by the α-channel is the *on-time reliability*. This attribute represents the minimum percentage, α, of this channel's packets that must be delivered on time, assuming that no failures occur among the nodes of the fixed routing path. The value $(1-\alpha)$ represents, therefore, the percentage of traffic that need not be delivered on time. In our scheme, this percentage of traffic will be considered best effort and will only be serviced if the current conditions of the node permits.

Based on the value of α, we define three classes of α-channels. The semantics of these classes derive from the percentage of on-time reliability that need to be supported by the α-channel. The first class, *deterministic* α-channel ($\alpha = 1$), requires that all of the generated traffic is delivered on time. The second class, *statistical* ($0<\alpha<1$), requires that only a percentage α of the generated traffic is delivered on time, while the rest of the traffic will only be considered if the current conditions of the node permit. Finally, the third class, *best-effort* ($\alpha=0$), is characterized by the absence of guarantees.

In the following section, we describe the resource reservation model used to guarantee the requested performance.

2.2 Resource Reservation Model Specification

At the α-channel creation time, a sender specifies the *end-to-end delay* of each packet, the *maximum number* of packets to be sent during this delay, and the *on-time reliability* required.

For each request, the network must identify a path from the source to the destination that can support the requested α-channel. This identification process consists of two parts. In the first part, a possible set of paths from the source to the destination is identified. The second part of this process consists of verifying that each node on a possible path can

support the real-time requirements of the new α-channel without violating those already guaranteed. In this paper, only the second issue is considered. The issues related to efficiently identifying possible paths to handle real-time traffic is currently being investigated. We assume that the network can generate a list of possible paths from the source to the destination. Each node along the path is required to support exactly α percent of the source generated traffic. The switching nodes rely on the source node to appropriately tag α percent of the traffic as requiring deterministic service.

Given a possible path, the proposed α-channel verification algorithm is then run on the nodes to determine the feasibility of accepting the new α-channel. This is achieved in our scheme by the *Channel Acceptance* process. All accepted α-channels are then managed by the *Run-Time Support* process.

2.3 Channel Acceptance Process

This process is responsible for determining whether to accept or reject a newly generated α-channel. Based on the specifications of the α-channel, the process determines whether the new α-channel can be serviced such that its real-time requirements are met, and all existing α-channels requirements continue to be guaranteed.

As stated above, the only information provided to the node specifies the maximum number of arrivals per node delay and does not provide any specification of the arrival pattern. The per node delay is the end-to-end delay divided by the length of the routing path. Consequently, the Channel Acceptance process must consider the worst case arrival pattern. This occurs when all packets arrive simultaneously at the node, and all α-channels are operating at their maximum rate. Under these conditions, the Run-Time Support process must be capable of processing the offered traffic without violating the deadline requirements.

2.4 Verification Model

The main idea underlying our 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. This relationship may be stated as follows:

> *The knowledge of both the maximum number of packets received by a given node over a given time interval and the per-packet processing time determines the maximum delay a packet may suffer at that node.*

Given that our scheduling policy is driven by the earliest deadline packet first, the process of determining whether an α-channel is acceptable consists of verifying that the above relationship holds in the case where:

- The given time interval is the per-node delay of the α-channel,
- The maximum number of packets is the total number of packets received from all α-channels over this time interval that require deterministic service, and
- The packet processing time for a given α-channel.

In the following, a more formal description of the scheme is provided.

2.4.1 Formal Specification of the Model

Let c be an α-channel going through node n. The α-channel c is characterized by :

- The maximum amount of time, Δ_c, any packet from α-channel c may be delayed at node n,
- The maximum number of packets, $N_c(\Delta_c)$, generated by α-channel c during any interval of size Δ_c. Therefore, $\alpha_c \times N_c(\Delta_c)$ represents the number of packets from α-channel, c, that need to be serviced by node n during any interval of size Δ_c.

The deadline of packet p from α-channel c may be define as $\delta_c(p) = a_p + \Delta_c - \rho$, where a_p is the arrival time of packet p at node n, and ρ is the processing time of a packet from α-channel c.

Based on the above characteristics, the model used to verify the feasibility of accepting an α-channel is based on the following theorem.

Theorem:

Let $1, 2, \cdots N$ be the α-channels currently supported by node n such that $\Delta_i \le \Delta_j$ if $i < j$, and $N_k(\Delta_k) \times \rho \le \Delta_k$, for all $k = 1, 2, \ldots N$. Furthermore, assume that the packet scheduling algorithm is based on the earliest deadline. Then, if for all $k = 1, 2, \cdots N$,

$$\sum_{i=1}^{k-1} W(i,k) + \alpha_k \times N_k(\Delta_k) \times \rho + \rho \le \Delta_k,$$

then

$$delay_{p^k} \le \Delta_k, \text{ for all } k = 1, 2, \ldots, N,$$

where $delay_{p^k}$ is the delay observed by a packet, p^k, from α-channel k, and $W(i,k)\ (i < k)$, represents the amount of time to service the maximum number of deadlines from α-channel i that occur during any interval of size Δ_k, in the worst case.

The worst case pattern of arrivals of two α-channels i and k occurs when the packets from α-channel i arrive in a way such that the deadline of the first packet coincides with the arrival of a packet from α-channel k and continue to arrive at their maximum predefined rate, as illustrated by Figure 1. 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. 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 $\alpha_i \times N(\Delta_i)$ 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 completely embedded in an interval of size $\Delta_k - R_1$. The amount of time required to service the deadlines that may occur within R_2 may be computed as $\left\lfloor (\Delta_k - R_1)/\Delta_i \right\rfloor \times N_i(\Delta_i) \times \alpha_i \times \rho$. Finally, R_3 can be expressed as $R_3 = \Delta_k - R_1 - R_2$. Thus a deadline from α-channel i will only need to be serviced during R_3 if $\Delta_i - \alpha_i \times N_i(\Delta_i) \times \rho \le R_3$, in which case the number of deadlines from α-channel i that will require service during R_3 is $\left\lfloor R_3 - (\Delta_i - \alpha_i \times N_i(\Delta_i)) \right\rfloor + 1$. Therefore, the value of $W(i,k)$, in the worst case, during an interval of size Δ_k, may be expressed as:

$$W(i,k) = R_1 + \left\lfloor (\Delta_k - R_1)/\Delta_i \right\rfloor \times N_i(\Delta_i) \times \alpha_i \times \rho + (\left\lfloor R_3 - (\Delta_i - \alpha_i \times N_i(\Delta_i)) \right\rfloor + 1) \times \rho$$

Based on the above theorem, the Acceptance Algorithm may be stated as follows. Let C_n = the set of α-channels currently supported by n. Assume that $|C_n| = N - 1$. Furthermore, let k be a new α-channel requesting to be established and assume that $\Delta_1 \le \Delta_2, \cdots \le \Delta_{k-1} \le \Delta_k < \Delta_{k+1} \cdots \le \Delta_N$. The new α-channel k is accepted if and only if the two following conditions are satisfied.

$$\sum_{i=1}^{k-1} W(i,k) + \alpha_k \times N_k(\Delta_k) \times \rho_k + \rho \le \Delta_k \qquad \text{(i)}$$

and

$$\sum_{i=1}^{k+j-1} W(i,k+j) + N_{k+j}(\Delta_{k+j})\,\rho_{k+j} + \rho \le \Delta_{k+j}, \text{ for } all\ j = 1, 2, \cdots (N-K). \qquad \text{(ii)}$$

Sketch of the proof:

According to the previous theorem, if conditions (i) and (ii) are verified, then any channel $i \in C_n$, $delay_{p^i} \le \Delta_i$ for all $p^i \in i$. Consequently α-channel k must be accepted. Otherwise admission of the new α-channel into the network causes the guarantees of at least one α-channel to be violated. Consequently, the new α-channel must be rejected.

In the next paragraph, we describe the run-time support process to service deterministically packets in a switching node.

2.5 Run-Time Support Process

In order to guarantee real-time requirements, the Run-Time Support process must provide deterministic packet service. This can be accomplished if the process can determine the order in which all packets are to be serviced, given any arrival pattern. In the following sections, a description of the basic scheme used to manage different queues in the switching node and the policy used to allocate the CPU to these packets is provided.

2.5.1 Processing Node Configuration

Each packet switching node contains a *service queue* and a set of *α-channel rate regulator queues*. These queues are managed by two entities, namely a *service queue manager*, and an *α-channel rate regulator*. The configuration of the switching node is illustrated in Figure 2.

The service queue manager waits on the service queue. This queue contains all packets that are currently eligible to be serviced by the CPU. The α-channel rate regulator monitors the rate regulator queues. The switching node provides a rate regulator queue for each α-channel currently supported by that node. The rate regulator is responsible for moving packets from the regulator queues into the service queue. In the next sections, a more detailed description of the basic operations performed by these two entities and the policies used to move packets from the rate regulator queue into the service queue is provided.

2.5.2 Service Queue Manager

As stated previously, the service queue contains all packets that are eligible to be serviced. These packets are ordered based on their deadlines. Packets with shorter deadlines are serviced before those with larger deadlines. When a packet has finished receiving service from the CPU, the service queue manager provides the CPU with the packet currently holding the shortest deadline in the queue. This packet may be labeled as best-effort, in which case the service queue manager must first verify that no packets requiring deterministic service is currently awaiting service. If there are no deterministic packets in the service queue, then the best-effort packet is serviced. Otherwise, the best-effort packet is dropped, and the next packet in the queue is considered. This continues until the next most eligible deterministic packet is located. Consequently, best-effort packets are only serviced if no deterministic packets are awaiting service.

2.5.3 α-channel Rate Regulator

The primary responsibility of the flow regulator is to maintain the flow rate of each channel at each switching node. This is necessary to ensure that the number of packets arrival a node may see does not exceed the prescribed rate of the α-channel. The rate regulation can be done in one of several ways. One way to monitor the rate of the α-channel is to maintain a list of departure times for the last several packets from this channel. The first packet in the regulator queue is moved into the service queue only when servicing it, and therefore sending it to the next node, will not violate the arrival rate at the next node. This can be achieved if the current history is maintained and used to determine when the first packet can be moved into the service queue. Maintaining the current history list, however, may necessitate the recording of a large number of packets an α-channel over its prescribed interval, thereby making the technique very expensive. This problem can be alleviated by

observing that the source must obey its prescribed arrival rate. Consequently, maintaining the original relative gaps of these packets throughout the nodes along the routing path ensures that the prescribed arrival rate is not violated at any intermediate node. This method was adopted in our scheme.

The network entry point of the α-channel keeps track of the elapsed time between each two consecutive packets. Each packet entering the network is then tagged by the corresponding elapsed time. This gapping information is used by the flow regulator to maintain the original time gap between two consecutive packets. This ensures that bursts of packets from the same α-channel can not develop within the network. Consequently, the packet at the head of the the regulator queue is moved to the service queue only after the appropriate amount of gap time has elapsed since the previous packet of the same α-channel was serviced. This mechanism ensures that the observed gaps among adjacent packets never decrease below their original values, thereby guaranteeing that no packet bursts that violates the prescribed arrival rate can develop.

3 Simulation Implementation

In order to test the validity of the proposed scheme, a simulation model was developed. The simulation was written using CSIM, a simulation package that combines the C programming language and a library of routines to provide basic simulation functionalities [12]. The simulated network testbed consisted of an arbitrarily topology were each node had the same processing capacity. In this study, the links connecting nodes were assumed to have 0 propagation delay.

For each simulated node, two CSIM processes were created: one to handle the service queue maintainer requirements, and one to handle the flow regulator requirements. Additionally, each accepted channel was implemented as a CSIM process. Each channel was configured to simulate packet arrivals in one of the following ways: bursty, were all packets were sent one after the other, periodic, were packets were distributed evenly over the delay interval, and exponentially, where packets arrivals where generated from an exponential distribution.

The simulation experiment was run with several network topologies. Different α-channels with various real-time requirements and various arrival pattern were tested. These requirements as well as the arrival pattern were changed to exercise different load on the network and verify the on-time reliability of the scheme. The results obtained for the network configuration specified by Figure 3 are reported in Table 1. The results show that the required guarantees of the supported α-channels were not violated. Furthermore, the observed guarantees of the statistical α-channels most of the time exceeded their required guarantees. We are currently in the process of extensively investigating the performance of the scheme and the per node utilization. Further work is aimed at modifying the basic

scheme to support packets of variable size. In addition, we are currently investigating models to characterize the excess capacity of the switching nodes and ways of incorporating this information in determining the set of nodes that can support the α-channel.

4 Conclusion

In this paper, we introduced a network level abstraction called α-channel to support the requirements of real-time applications. An α-channel represents a simplex, end-to-end communication channel with guaranteed service characteristics, between two entities. We described the basic scheme that our model uses to verify the feasibility of accepting a new α-channel with guaranteed performance. A formal description of the proposed model and a proof sketch of its correctness was provided. In addition, the preliminary results from a simulation experiment implementing the basic functionalities of the proposed scheme and were presented. The results show that the guarantees of the supported α-channels were preserved.

5 Bibliography

[1] Chen, T.M., and D.G., Messerschmitt, "Integrated Voice/Data Switching", IEEE Commun Mag., vol. 26, pp. 16-26, June 1988.

[2] Postel, J., "Internet Protocol; DARPA Internet Program Protocol Specification", RFC 791, September 1981.

[3] Ferrari, D., and Verma, D., "A Scheme for Real-Time Channel Establishment in Wide-Area Networks", IEEE Journal on Selected Area of Communications, Vol. 8, No. 3, April 1990.

[4] Parulkar, G.M., and Turner, J.S., "Towards a Framework for High Speed Communications in a Heterogeneous Networking Environment", in Proc. INFOCOM, Ottawa Canada, April 1989, pp. 655-688.

[5] Zhang, L., "A New Architecture for Packet Switching Network Protocols", PhD. Thesis, Massachussetts Institute of Technology, July 17, 1989.

[6] Anderson, D.P., Herrwich, R., and Shaefer, C., "SRP: A Resource Reservation Protocol for Guaranteed-Performance Communication in the Internet", Tech. Rept. No. TR-90-006, International Computer Science Institute, Berkeley, February 1990.

[7] Field, B., and Znati, T., "Experimental Evaluation Of Transport Layer Protocols For Real-Time Applications", in Proceedings of the 16th Annual Conference on Local Computer Networks, Minneapolis, Minnesota, 1991.

[8] Schwetman, H.D., "CSIM: A C-Based, Process-Oriented Simulation Language", *Microelectronics and Computer Technology Corporation*, Technical Report, PP-080-85.

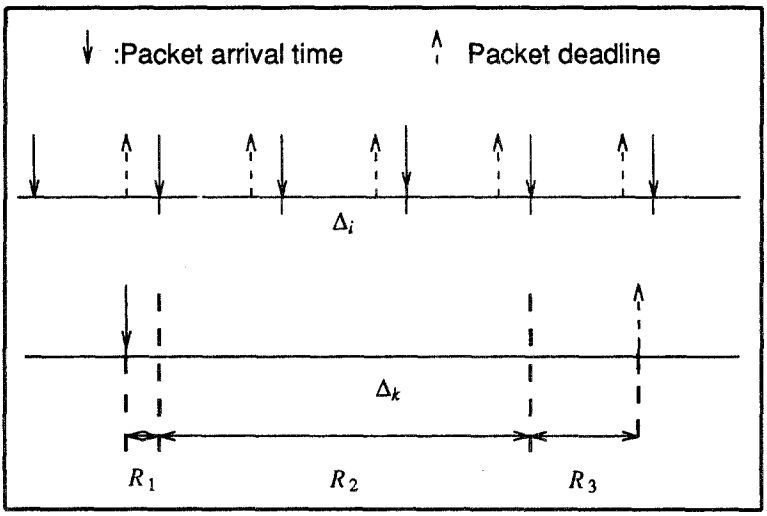

Figure 1- Worst case packet arrival pattern (channel i and k).

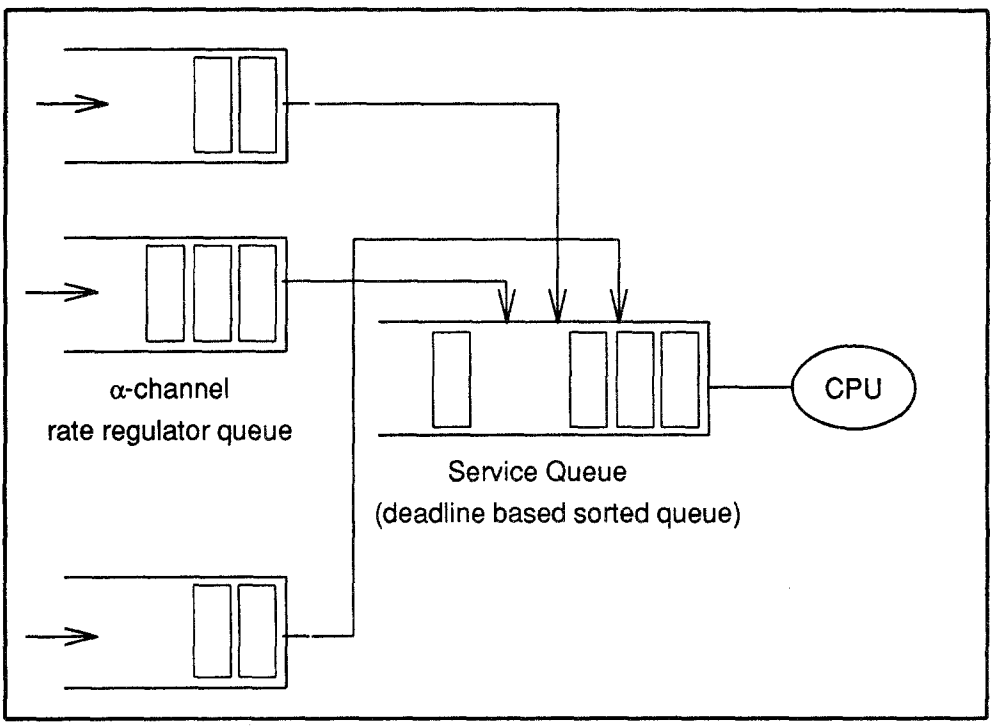

Figure 2- Configuration of a switching node.

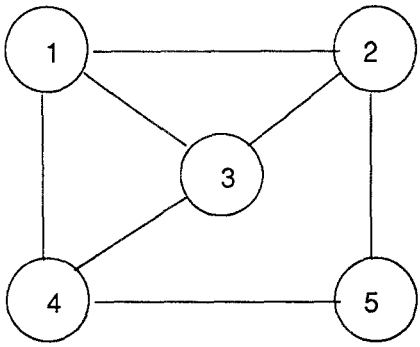

Figure 3- Network Configuration.

α-channel id	$N_{id}(\Delta_{id})$	Per node delay	Path	Required α	Observed α
0	25	65.0	1	0.90	0.94
1	13	54.0	2	0.80	0.86
2	18	47.0	3	1.00	1.00
3	16	99.0	4	1.00	1.00
4	22	47.0	5	0.84	0.96
5	15	51.0	2 -> 5	0.80	0.80
6	16	56.0	3 -> 4	1.00	1.00
7	17	49.0	2	1.00	1.00
8	11	171.0	1 -> 2 -> 5	1.00	1.00

Table 1- Performance Results for Various α-channels