REAL-TIME COMMUNICATION IN AN INTERNETWORK

Domenico Ferrari
The Tenet Group
Computer Science Division
Department of Electrical Engineering and Computer Sciences
University of California
and International Computer Science Institute
Berkeley, California, U.S.A.

Abstract

Can end-to-end communication performance be guaranteed by a packet-switching internetwork? This paper addresses the question by examining the feasibility of extending to an internetwork the Tenet approach to real-time communication service design. The conditions to be satisfied by an internetwork so that the approach can be extended to it are investigated. These include conditions for the scheduling discipline to be used in the nodes of the internetwork.

The original Tenet approach to real-time communication applies to a network consisting of hosts, homogeneous nodes (or switches), and physical links connecting nodes and hosts in an arbitrary topology. The nodes are store-and-forward, and are scheduled by a multi-class version of the Earliest Due Date deadline-based policy.

The discussion presented in this paper results in extendibility conditions that are quite broad; hence, the Tenet approach may be used to establish and run real-time channels in a vast class of internetworks. A case study is also discussed, involving a simple network, whose nodes are scheduled by FCFS-based disciplines, and the connection of such a network to an internetwork with deadline-based and hierarchical round robin scheduling.

This research was supported by the National Science Foundation and the Defense Advanced Research Projects Agency (DARPA) under Cooperative Agreement NCR-8919038 with the Corporation for National Research Initiatives, by AT&T Bell Laboratories, Hitachi, Ltd., Hitachi America, Ltd., Pacific Bell, the University of California under a MICRO grant, and the International Computer Science Institute. The views and conclusions contained in this document are those of the author, and should not be interpreted as representing official policies, either expressed or implied, of the U.S. Government or any of the sponsoring organizations.

1. Introduction

Among the services to be offered by a broadband ISDN network, those entailing some sort of performance guarantee are particularly interesting, especially for networks based on the packet-switching principle. We call them *real-time communication services*. Since packet switching (or cell switching) seems to be gaining the upper hand in the argument about how B-ISDNs should operate, the question of whether and how performance can be guaranteed in such a congestion-prone environment as that created by packet switching is much more than an intriguing academic problem.

The Tenet Group at the University of California at Berkeley and the International Computer Science Institute gave this question an affirmative answer in the case of a network with arbitrary topology, consisting of hosts and homogeneous store-and-forward switches, in which packet transmissions are scheduled by a multi-class version [FeVe90a] of the Earliest Due Date (EDD) deadline-based discipline [LiLa73]. In the sequel, whenever we use the term *simple network*, we shall refer to such a network. The details of the scheme that applies to simple networks have been published in [FeVe90a], where a general method and the specific algorithms for establishing real-time connections are given, in [FeVe90b], where the allocation of buffer space to such connections is discussed, and in [VeZF91], where a scheme for distributed jitter control is presented. The Tenet approach allows a client of the real-time communication service to choose without restrictions the desired bounds of the important performance indices (delay, throughput, reliability) and to specify, also without restrictions, the values of the parameters of a simple traffic characterization. This information is used to determine whether the client's request can be accepted by the network in its current load conditions.

The Tenet approach differs in many respects from other solutions that provide performance guarantees in a packet-switching network. Unlike several other approaches, it is not based on the use of a particular type of switch or gateway, nor is it confined to only one kind of network topology. Unlike the Flow Protocol [Zhan90], which offers average throughput bounds and, as a secondary and constrained objective, delay bounds, it allows clients to specify and obtain throughput and delay bounds totally independently of each other, as well as jitter and loss bounds. The Session Reservation Protocol [AnHS90] is based on a philosophy similar to ours, but with several important differences; these include a different admission control policy, a different traffic model, the bundling of control and delivery functions within the same protocol, and the compatibility of the SRP header with the IP header, which raises some obstacles to internetworking. The Stream Protocol, Version II (ST-II) [Topo90] is an experimental internetwork-layer protocol developed for the DARPA Internet. Unlike our current scheme, it offers multicast connections; its traffic and performance parameters, and its channel establishment procedure, differ appreciably from those of our approach. No resource reservation policies are detailed in the specification of ST-II, though the usable policies are obviously constrained by the messages exchanged during channel establishment and by the parameters contained in those messages. Another protocol at the same level, the Multipoint Congram-oriented High-performance Internet Protocol (MCHIP) [PaTu90], provides multicast capabilities, and may make use of resource servers, which keep track of the channels established and of the available resources. The establishment procedure and the resource reservation policies adopted for MCHIP have not, to our knowledge, been published yet. The Asynchronous Time Sharing (ATS) approach differs from ours primarily because of its being based on a fixed menu of quality-of-service classes [LaPa91]: each real-time connection obtains from the network performance guarantees corresponding to those of its class. Other differences include a cooperative distributed scheduling discipline based on traffic prediction, and the introduction of the notion of system utility function, whose expectation is to be maximized by the admission control policy.

2. Statement of the Problem

How general is the Tenet approach? Can it be applied to networks that do not obey all of the conditions assumed to hold in [FeVe90a]? What are the limits of validity of that method for the design of a real-time communication service? In particular, can the approach be extended to an internetwork? What conditions does the internetwork have to satisfy? This paper discusses these and other related questions.

To focus our discussion of these questions, we attack the problem as a design problem, and begin by specifying the objectives of our design.

Our goal is to devise a set of algorithms on which one or more protocols implementing a real-time communication service can be based. The service we would like the internetwork to offer is essentially identical to that offered by a simple network implementing the algorithms proposed in [FeVe90a], [FeVe90b], and [VeZF91]. That is, the internetwork would allow a client, to be called a *real-time client*, to request simplex end-to-end connections, called *real-time channels*, that offer the following end-to-end performance guarantees:

(1) *throughput*: both a client-specified lower bound for instantaneous values and a client-specified lower bound for average values, averaged over a client-specified interval;

(2) *delay*:

(2.1) a client-specified absolute upper bound $D_{\max}$, or

(2.2) a client-specified probabilistic bound D_{max}, $Z_{\min}$, where $Z_{\min}$ is the lower bound of the probability that the delay is not larger than $D_{\max}$, or

(2.3) a client-specified delay jitter upper bound $J_{\max}$ which, when accompanied by an absolute delay upper bound $D_{\max}$, is equivalent to specifying an absolute lower bound $D_{\min}$;

(3) *reliability*: a client-specified probabilistic upper bound $1 - W_{\min}$ on packet losses due to buffer overflows; when the client assigns the value 1 to the probability of no losses $W_{\min}$, the reliability bound becomes absolute.

The three types of delay bound listed above are mutually exclusive, even though (2.1) and (2.2) coincide when $Z_{\min} = 1$, and though (2.3) includes an absolute bound of type (2.1). A channel with a delay bound of type (2.1) is said to be a *deterministic channel*. One with a bound of type (2.2) is a *statistical channel*, and one with a bound of type (2.3) is a *bounded-jitter channel*.

Besides selecting the type of channel and assigning values to the relevant performance bounds, a client must declare the bounds that must be obeyed by the traffic on the channel. In our approach, these are the same values of the throughput bounds referred to in (1) above; that is, the minimum[1] interpacket time, $x_{\min}$, and the minimum of average interpacket times, x_{ave}, averaged over an interval I. The specification

[1] The reciprocal of this minimum is the maximum rate at which the client inputs packets into the channel. The network must offer an instantaneous throughput not smaller than this rate. A similar argument applies to averages.

of throughput bounds therefore represents both the client's minimum bandwidth requirements and the client's pledge to control the traffic rate so that it will always satisfy those upper bounds. Since clients cannot be trusted to keep their promises (for instance, a failure or an error in a host might cause the flooding of one or more channels), the network, to protect the guarantees given to its well-behaved clients, must control the packet rate on each channel; in our scheme, a distributed rate control mechanism is used for this purpose [FeVe90a] [VeZF91].

Thus, from the viewpoint of the client, there must be no difference of any kind between the real-time communication service for a simple network presented in the papers referenced above and the one to be designed for an internetwork. This *internetwork transparency* should, however, not be obtained by the introduction of global, centralized control mechanisms that would reduce the autonomy of each of the networks in the internetwork. Also, the amount and complexity of the real-time communication software that should be installed in the gateways and inside each of the networks should be kept to a minimum.

3. The Principles of a Solution

Perhaps the easiest and most natural way of approaching the problem stated in the previous section is to investigate the feasibility of extending our simple-network solution to the case of an internetwork. In doing this, we can try to exploit the observation that an internetwork may be regarded as a hierarchical structure, where the *level-1 network* consists of nodes that coincide with the gateways interconnecting the various networks, and of links representing the networks between gateways. Since each of these links is an abstraction of an underlying reality that may be very complicated, we call them *logical links*, to distinguish them from the *physical links* connecting the nodes of each component network.

The component networks are *level-2 networks*. Indeed, some of them might in turn be internetworks with their own gateways, in which case they consist of *level-3 networks*, and so on. If a level-2 network is an internetwork, the part of it that is really at level 2 is the network whose nodes are the gateways (*level-2 gateways*); these are connected by *level-2 logical links*, which are really abstractions of level-3 networks.

Consider, for instance, Xunet 2, a wide-area internetworking testbed connecting AT&T Bell Laboratories in Murray Hill, New Jersey, and the Universities of Illinois at Urbana-Champaign, Wisconsin at Madison, and California at Berkeley [FKKM91]. A simple end-to-end path through Xunet 2 involves three networks, as shown in Figure 1: two FDDI rings, which connect the source and destination hosts to their respective local routers, and an ATM backbone, which connects the two routers through a series of ATM switches. Besides level 0, where we have single logical links connecting sources to destinations, our hierarchical model of Xunet 2 includes two levels: level 1 is the level of the routers (see Figure 1(b)), which operate in a store-and-forward fashion, and schedule the real-time component of the traffic according to the Multi-Class EDD discipline; level 2 consists of the three networks interconnected by the routers, which appear as router-router or router-host logical links at level 1. The nodes in Figure 1(b) are level-1 nodes, which run an internetwork protocol. Node 1 represents the network layer and lower layers on the source computer, and shares that computer with source P, which represents the application and the protocols at and above the transport layer.

The delay on each link at level 2 is boundable if the real-time traffic is treated as synchronous by the two FDDI rings. Since the ATM switches are scheduled by the Hierarchical Round-Robin discipline [KaKK90], the delay on the ATM backbone is bounded, and such a bound can be computed [BaKe92]. Hence, the level-1 logical links have known delays bounds.[2]

Our simple-network solution assigns a fixed route to each real-time channel at the time of its creation; this creation precedes the actual use of the channel, since any request for a new channel must be scrutinized, and can only be accepted if there are enough resources (link bandwidth, node processing power, buffer space) to guarantee the performance bounds desired by the client. The route of a channel is fixed because it would be extremely hard, perhaps impossible, to control delays otherwise. We can model also the route of a real-time channel hierarchically as described above: the gateways of the "global" internetwork are the nodes of the level-1 route, whose links are level-1 logical links, representing abstractions of level-2 inter-gateway routes, and so on.

The level-1 network is a simple network, and the level-1 route is a *simple route*. Thus, we can apply to this route our approach, which is summarized in the rest of this section. In doing so, however, we cannot ignore the differences between logical and physical links, nor the assumptions about nodes on which our simple-network approach is based. These matters are discussed in the next section.

During the channel establishment phase, the channel's route is constructed through the simple network with a mechanism similar to that for building virtual circuits. In each node visited by the channel establishment message, however, several tests are performed to determine whether the new channel can be set up without jeopardizing the guarantees given to the existing channels. The mathematical forms of these tests depend on the node's architecture and scheduling policy, as well as on the type of characterization chosen for the traffic by the real-time service designer. Simple tests for the traffic characterization described in Section 2 and for a node obeying either of the models shown in Figure 2, and scheduled by Multi-Class EDD, can be found in [FeVe90a], [FeVe90b], and [VeZF91].

The tests, if successful, are accompanied by computations, some of whose results are transmitted to the destination of the channel. These results include the minimum delay bound the node can offer to the new channel without endangering any of the existing guarantees, the maximum probability of delay bound overflow the node can commit to, and the amount of buffer space still available to real-time traffic in the node.

If one of the tests at a node is unsuccessful, the message is stopped, and a negative reply is sent back to the source, i.e., to the requesting client. The reply message follows the partial route created so far, and release the resources that had been tentatively assigned to the new channel in each node. Note that such resources as link bandwidth and node processing power need not be exclusively and continuously allocated to real-time channels, as long as the priority scheduling policy used in the nodes can effectively preempt their non-real-time users whenever a real-time packet arrives; buffer space, however, is statically allocated to each channel, to prevent malicious or faulty sources from invading space without which the guarantees given to other channels might be violated.[3]

[2] At level 2, packets are replaced by FDDI frames or ATM cells, and packet delays are to be computed from frame or cell delays, respectively. However, since the latter are boundable, and their bounds are known, the former are too.

[3] A more dynamic yet safe allocation of space could be adopted at the cost of additional complexity.

The final tests are run by the destination, using, with other information, the data transmitted to it by the nodes along the route via the establishment message. If these tests are successful too, the destination has to determine the actual (as opposed to the tentatively assigned) amounts of resources the channel are given in each node. To this end, as well as to allow establishment tests for future channels (whose routes in most cases only intersect the one of the channel being considered here), the destination subdivides the end-to-end bounds into per-node components. Then, the channel is characterized by its local bounds in each node.

The value of $D_{\max}$, for all types of channels defined in Section 2, is broken into a sum of per-node delay bounds $d_{max,n}$ ($n=1,...N$, where N is the number of nodes on the channel's route); each $d_{max,n}$ must be greater than or equal to the minimum delay bound the node can offer. In our simple-network approach, where we assume that real-time packets are scheduled in each node by a multi-class version of EDD, the value of $d_{max,n}$ is used to compute the deadline of each packet on the channel for transmission from node n.

Jitter bounds would be additive like delay bounds if jitter were not controlled. However, for a bounded-jitter channel, jitter must obviously be controlled; hence, the end-to-end bound $J_{\max}$ is not subdivided, but, in a simple network, is only compared to the jitter introduced by the last node (node N) [VeZF91] [Ferr91].

The two probability bounds $Z_{\min}$ and $W_{\min}$ cannot be decomposed into sums of per-node terms. Let us assume that, if a packet is delayed beyond its local deadline or lost due to buffer overflow in any of the nodes along the route, it is delayed beyond its end-to-end deadline or lost. This assumption, which is true for losses but very conservative in the case of delay bounds, yields a product-form decomposition:

$$Z_{\min} = \prod_{n=1}^{N} z_{min,n}\,, \tag{1}$$

$$W_{\min} = \prod_{n=1}^{N} w_{min,n}\,, \tag{2}$$

where $z_{min,n}$ and $w_{min,n}$ are the probability bounds in node n for the channel in question.

After subdividing the performance bounds among the nodes on the new channel's route, and computing the amount of buffer space to be assigned to the channel in each node, the destination packs all of this information into an channel-accept message and ships it back to the source along the route in the reverse direction. Each node obtains from the message the values of its local bounds and of other parameters, modifies, if necessary, the amounts of resources it had tentatively assigned to the channel, and commits the revised amounts to it. Thus, each traversed node writes values into the forward message and reads values from the reverse message; if these messages are thought of as trains, each node can be seen as attaching a new wagon to the end of a forward-moving train and detaching its wagon from the end of a returning train; such trains are shortest at the source and longest at the destination; the destination uses data from all wagons for its computations and tests, while each node only needs the information in its wagon during the reverse trip, and only contributes its wagon to the forward train. We use this train metaphor again in Section 4 to explain our proposed channel establishment mechanism in an internetwork environment.

4. Extending the Approach to an Internetwork

A simple network is characterized by a number of properties, which must be made explicit now that we want to investigate the possibility of extending the scheme to an internetwork. The fundamental characteristics of the network (besides those we have already discussed in the previous sections, such as the connection-orientedness, the respective roles of hosts and switches, and so on) are:

(1) all nodes operate in the store-and-forward mode;

(2) all nodes use the Multi-Class EDD discipline to schedule packet transmissions;

(3) all links are physical (as opposed to logical).

Property 1 is, we believe, non-negotiable: any scheme involving the scheduling rather than the immediate passing along of packets requires that they be stored and then forwarded, though it may be unnecessary to wait for the whole packet before scheduling its transmission. Thus, level-1 nodes in the internetwork are assumed to be store-and-forward gateways.

The problem of extending Property 2 is discussed in the next section. For the rest of this section, which is devoted to a discussion of Property 3, we shall assume that level-1 nodes in the internetwork schedule packets according to the Multi-Class EDD policy.

A physical link is characterized by the following properties, which are taken implicitly into account by the destination in its decomposition of the client-specified bounds:

(a) constant delay, equal to the speed-of-light propagation delay (note that we consider the transmission delay as part of the node delay);

(b) zero jitter;

(c) zero probability of delay beyond the delay bound, which is the constant value referred to in (a) above ($z_{\min}$=1);

(d) zero probability of packet loss due to buffer overflow, as there is indeed no buffering in a physical link ($w_{\min}$=1).

A logical link such as one that connects two level-1 nodes in an internetwork generally have different properties. However, our scheme is still applicable at level 1 if the delay, the probability of late delivery, and the packet loss probability of each logical link on the channel's level-1 route can be bounded.[4]

Indeed, if the maximum delay on logical link k is $dl_{max,k}$, the destination can use the following equation to set the per-node delay bounds along the route of a channel being established:

$$D_{\max} = \sum_{k=1}^{N+1} dl_{max,k} + \sum_{n=1}^{N} d_{max,n}\,, \qquad (3)$$

where the sum over the links has $N+1$ terms because both the connection between the source and node 1 and that between node N and the destination are considered as links.

[4] All of the quantities we deal with have finite bounds. However, "boundability" here means the ability to change these natural bounds to suitably small (for delays) or large (for probabilities) values by the use of appropriate techniques, or the knowledge of natural bounds when those bounds are already sufficiently small or large, respectively, for the applications being considered.

Similarly, the per-node probability bounds obey the following equations:

$$Z_{\min} = \prod_{k=1}^{N+1} zl_{min,k} \prod_{n=1}^{N} z_{min,n} , \tag{4}$$

and

$$W_{\min} = \prod_{k=1}^{N+1} wl_{min,k} \prod_{n=1}^{N} w_{min,n} , \tag{5}$$

respectively, where $zl_{min,k}$ and $wl_{min,k}$ are the late delivery and packet loss probability bounds for logical link k. If the delay bounds are deterministic, and clocks can be kept in synchrony, that is, if any differences between simultaneous readings of the clocks are known and do not vary appreciably during the lifetime of a packet in the network, jitters can also be bounded deterministically [Ferr91] [Part91].

There are two ways in which the *boundability* condition may be satisfied for delay or probability indices:

(i) the bound on that performance index for the given logical link is known *a priori*, and is satisfactory[5] for our purposes, or it cannot be modified by our procedure; in this case, the real-time service does not have to know how the bound is obtained, and can take it as given; for this reason we say that the logical link is a *black-box link* with respect to that index;

(ii) a satisfactory bound on that performance index for the given logical link can be enforced by the real-time service; in this case, our extended scheme must be applicable to the logical link in question for that index, and the logical link must therefore satisfy the conditions we are investigating in this paper; we shall call that link a *white-box link* with respect to that index.

An example of a black-box link with respect to the delay is a circuit-switched backbone: in this case, the real-time service can even ignore the non-packet-switching nature of it! An example of a white-box link is one that represents what we have called a "simple route", i.e., a route through a simple network.

In Xunet 2, for instance, a path through the ATM backbone can be represented by a white-box logical link with respect to delays, late delivery probabilities, and loss probabilities, since its nodes are scheduled by the HRR discipline, which can guarantee deterministic delay bounds (see Section 5 below), and the amount of buffer space needed to keep losses below a given bound can be calculated and used for admission control.[6] The real-time service could also choose to consider the logical link representing a path through the ATM backbone as a black-box link with respect to loss probabilities, if a minimum loss probability can be computed on the basis of the maximum rate on a channel being established and the amounts of buffer space available to the channel in the nodes.

In the same internetworking testbed, the FDDI rings may be conveniently considered as black-box links with respect to delays, with a bound equal to the sum of twice the negotiated value of the target token rotation time [JoSe87] and the maximum packet processing times in the FDDI interfaces and drivers. Since the amount of buffer space allocated to a channel can be calculated and used in the admission control

[5] Sufficiently small for delay bounds, sufficiently large for probability bounds.

[6] Note that the ATM backbone is not a simple network, as defined in Section 1, since its nodes are not scheduled by the Multi-Class EDD discipline.

procedure, these links may be treated as black boxes even with respect to the probability indices.

The two types of links are dealt with very differently during the channel establishment procedure. The bounds on a black-box link are to be taken into account in the decomposition of end-to-end bounds: delay bounds are subtracted, and end-to-end probability bounds are divided by link probability bounds. Note that a logical link may be a black box for one index and a white box for another.

Let k be a white-box level-1 logical link representing a level-2 path containing N' nodes. Then, the parameters of k to be used by the destination to compute the delay bounds of the level-1 nodes can be obtained from the following equations:

$$dl_{max,k} = \sum_{m=1}^{N'+1} dl'_{max,m} + \sum_{n=1}^{N'} d'_{max,n}\,, \tag{6}$$

$$zl_{min,k} = \prod_{m=1}^{N'+1} zl'_{min,m} \prod_{n=1}^{N'} z'_{min,n}\,, \tag{7}$$

$$wl_{min,k} = \prod_{m=1}^{N'+1} wl'_{min,m} \prod_{n=1}^{N'} w'_{min,n}\,, \tag{8}$$

where $dl'_{max,m}$, $zl'_{min,m}$, and $wl'_{min,m}$ are the bounds of the m-th level-2 logical link of the level-2 path. Similar equations can be used to compute these bounds, and the procedure can be repeated until all white-box logical links have been decomposed into paths that contain only physical links (for which dl is constant, and zl, wl are zero) or black-box logical links. Clearly, since a logical link may be black-box for some indices and white-box for others, there may be different decompositions for different indices.

The task of determining the values of the parameters that are transmitted to the destination (if a logical link is white-box), or of remembering the values of these bounds and of forwarding them (if a logical link is black-box), can be best performed by the level-1 node where the logical link being considered ends. When a white-box link is encountered, the *hierarchical procedure* described above for the establishment of a real-time channel is used. For example, a level-1 logical link is an abstraction of a level-2 path through a level-2 network. If our scheme applies to the level-2 network, the destination (i.e., the level-1 node at the end of the level-2 route) receives information from every level-2 node, and extracts from that information the data concerning the logical link to be forwarded to the level-1 destination. Instead of decomposing the bounds for the logical link, which are not yet known during the forward trip of the establishment message, into per-node components, the level-1 node forwards the logical link information to the final destination of the channel; this destination subdivides the end-to-end bounds into per-segment components. A *segment* is the union of a node and the logical link incident into that node.

The channel-accept message then informs each level-1 node of the values assigned by the destination to its segment's local bounds, and each level-1 node allocates appropriate fractions of the bounds to itself and to the level-2 segments that constitute its logical link. This procedure can obviously be extended to any number of levels. A destination at level i only knows the level-i route, and decomposes the bounds assigned to that route among level-i segments.

For example, when a real-time channel with a delay bound $D_{\max}$ is to be established in Xunet 2 along the route shown in Figure 1(a), the destination DH decomposes $D_{\max}$ into four local delay bounds, one for each of the level-1 segments depicted in Figure 1(b). This decomposition is based on information concerning propagation and switching delays as well as the minimum delay bound that each node along the path can offer the new channel. The segment delay bounds (whose sum equals $D_{\max}$) are sent to the level-1 nodes in the channel-accept message. When the destination router DR receives the value of delay bound assigned to its segment, it subtracts from it the propagation delay on logical link 3 (the link connecting node 2 to node 3 in Figure 1(b)), and subdivides the result into four terms: its own local delay bound $d_{max,3}$, and three delay bounds in the three ATM switches at level 2. These three values are sent to each switch controller via the channel-accept message, which may have to select a smaller local delay bound; if this happens, the value of the local delay bound for the switch immediately upstream, or for the source router SR, is increased by the same amount.

With black-box logical links, the forwarded logical link information corresponds to the actual bounds (e.g., $d_{max,n}$). This is, for instance the case if a logical link consists of a single physical link; indeed, in a simple network, propagation delays are subtracted from the end-to-end delay bound before decomposing what remains among the nodes. More complex logical links may have variable delays: if we consider the upper bounds of the delays in the decomposition of $D_{\max}$, we introduce additional jitter in the corresponding segments. This can be compensated for by a suitable jitter control mechanism (see for example [Ferr91]). If jitter control is not implemented, then the client-specified jitter bound must be decomposed in the same way as the end-to-end delay bound.

The hierarchical procedure we have described for establishing a real-time channel in an internetwork can be visualized using an extension of the train metaphor introduced in the previous section. The establishment request message at any given point along the route contains a sequence of wagons belonging to various levels. Each level-i node adds a wagon containing the information it needs to forward. Each destination of a level-i route (which is a level-(i-1) node) detaches all the level-i wagons at the end of the train and replaces them with a level-(i-1) wagon that contains the abstracted information describing the level-(i-1) segment. The level-1 destination receives only level-1 wagons, and "uses" them to send the values of the local bounds to the individual level-1 segments. Each end node of a level-i segment computes the local bounds of each level-(i+1) segment belonging to the level-i segment; it reattaches the level-(i+1) wagons it had detached before, and distributes the bound assignments to its level-(i+1) segments. The first node of a level-(i+1) route detaches the last wagon of the level-(i+1) sequence, and the source of the channel receives a train as short as the one that left from it, i.e., with no wagons.

5. Extending the Scheduling Policy Requirements

The discussion in the previous section can be summarized by stating that, in order for the Tenet approach to be extendible to an internetwork, it is sufficient that the internetwork have level-1 logical links with boundable delay, boundable late delivery probability, and boundable packet loss probability, as well as level-1 nodes that schedule packet transmissions using the Multi-Class EDD policy. This conclusion of our discussion in Section 4 applies, of course, also to each component network and internetwork. But should we always require that the scheduling policy implemented by all nodes at all levels on white-box logical links be Multi-Class EDD? Is Multi-Class EDD the only policy that allows us to bound delays, jitters, and late delivery and loss probabilities?

First of all, we observe that loss probabilities can be removed from further consideration. The scheduling policy certainly influences the amount of buffer space a channel needs in a node for guaranteeing a given loss probability bound, but the necessary amount of space can be computed from the values of the local delay bound and of the maximum jitter at the input to the node [FeVe90b] [Ferr91]. Thus, as long as we can compute these values for a given scheduling policy, we can ignore loss probabilities in our discussion.

Furthermore, we can also avoid discussing delay jitter bounds, which, as mentioned in Section 4, can be provided if it is possible to guarantee deterministic delay bounds and if the source and destination clocks are synchronized with each other. Thus, our discussion may be confined to the question of whether delay bounds can be guaranteed without Multi-Class EDD. Initially, we are concerned only with deterministic bounds. Statistical bounds are discussed at the end of this section.

What properties of a packet scheduling discipline make it suitable for real-time services? To be more specific, what characteristics must a discipline have to be able to guarantee a finite delay bound for each real-time packet in a node? The answer is simple: given a finite number of packets present in the node at any time, the discipline must not starve (i.e., ignore forever) any of them. However, in order to obtain finite delays for all packets, even a starvation-free discipline requires that another, *external* condition be satisfied: that the real-time packet population in the node be finite at all times.[7] If the traffic includes non-real-time packets, as is to be expected in a practical internetwork, and we do not want to impose the same finite-population restriction on the non-real-time components, the scheduling discipline must satisfy another condition: that the resource requirements of real-time packets be protected against the demands of non-real-time ones.

These may be regarded as conditions to be met by any scheduling policy to support a real-time service in the environment defined by our assumptions. A set of properties that are sufficient to satisfy these conditions is the following:

(1) the network controls the admission of real-time channels;

(2) either

  a. each source keeps its packet generation rate below a known maximum value,

  or

  b. the scheduling policy treats the packets on each channel independently of those on any other channel, so that a source generating packets at a rate higher than its maximum can only hurt the performance and/or the reliability of its channel but not those of the others;

(3) the discipline that schedules real-time packets guarantees that starvation of any member of a time-variant but finite population of those packets is impossible;

(4) either

[7] Obviously, this must be true even if there were buffer space for an infinite number of packets in the node. We believe that the distinction between policies that may cause starvation and policies that may not is meaningful only if the number of "customers" in the queueing server being considered is bounded by a finite value at all times.

a. real-time packets have higher (not necessarily preemptive) priority than non-real-time ones,

or

b. the bandwidth allocated by the scheduling policy to a real-time channel cannot be appropriated by non-real-time packets except when it is not used.

It is easy to see that Conditions (1) and (2) together satisfy the finite-population condition above, and that Condition (4) is sufficient to protect real-time packets from non-real-time ones. Conditions (1) and (2)a must be satisfied by the network, since sources cannot be trusted to keep their rates below the bounds they have promised not to exceed, and the network has therefore to protect each channel from the others. Condition (2)b, which is for the scheduling policy to satisfy, can achieve the same result as (2)a since an independent treatment protects every channel from the possible misbehaviors of other channels; however, the two alternatives are not mutually exclusive; in fact, using a scheduling policy that satisfies (2)b even when (2)a is satisfied may appreciably improve the real-time communication service. Conditions (3) and (4) are to be satisfied by the scheduling policy. Note that Condition (4)b prevents non-real-time packets from stealing the attention of the scheduler away from real-time ones, which might endanger the guarantees promised to the latter. In this sense, Condition (4)b yields results similar to those of (4)a, but may coexist with it; in fact, even when (4)a is met, a policy that satisfies (4)b generally increases the real-time capacity of a network (i.e., the maximum number of real-time channels that can be supported by that network).

The Multi-Class EDD discipline satisfies Conditions (3) and (4)a, as well as (2)b; when coupled with an admission control scheme (Condition (1)) like the one described in [FeVe90a], it can be used to build a real-time service. There are, however, many other policies that satisfy the above conditions. Even the simplest discipline, FCFS, qualifies, as long as it is used in a network where admission and source input rates are controlled (rates must be controlled since FCFS does not satisfy Condition (2)b), and as long as real-time packets have higher priority than non-real-time ones (this is necessary since FCFS does not satisfy Condition (4)b either). If these conditions are met, FCFS qualifies, since it does not starve any packets (Condition (3)).

Unfortunately, a real-time service based on FCFS scheduling in the nodes is not very efficient. Assume for simplicity that a node n is traversed by C_n deterministic real-time channels, all of which have the same maximum rate $1/x_{\min}$ and the same packet processing time dp_n in the node. The maximum number of real-time channels node n can support under these conditions, $C_{max,n}$, is called the *real-time capacity* of n. Assume also that the node can be modeled as in Figure 2(a).

To compute $C_{max,n}$ and the delay bound $d_{max,n}$ (which is the same for all channels) of node n, we must determine how many packets from each channel may be in the node at any given time in the worst case; this number equals the number of packet-sized buffers we need to allocate to each channel in the node if losses due to buffer overflow are to be avoided. In a hierarchical internetwork, this number is given by

$$b_m = \left\lceil \frac{d_{max,m} + j_{in,m}}{x_{\min}} \right\rceil , \tag{9}$$

where the node being considered is the m-th on the channel's route, and $j_{in,m}$ is the maximum jitter for the channel's packets at the input to that node [Ferr91]. If we assume that the network is a simple one, as defined in Section 1, but with a FCFS scheduling policy in the nodes instead of Multi-Class EDD, we have

$$j_{in,m} = \sum_{k=1}^{m-1} (d_{max,k} - dp_k). \tag{10}$$

where dp_k is the processing time in node k. Note that $j_{in,1} = 0$. Thus, the maximum number of packets from another channel that a given packet arriving at node n can find ahead in the FCFS queue is a function of the position of that node in that channel's route. Since $d_{max,k}$ and dp_k are known and constant for all k, this maximum can be computed, and whether the new channel is acceptable by node n is determined when the request for its establishment reaches n. The real-time capacity of a node depends on the values of b_m for all channels traversing the node; it may be useful to denote by $C^*{}_{max,n}$ the maximum number of channels with $b_n = 1$ that can be created through it, and to refer to this number as the *maximum real-time capacity* of a network satisfying all the above assumptions. Note that $C^*{}_{max,n}$ is also the number of packet-sized buffers to be provided in node n if packet losses are to be avoided. The admission criterion for node n in such a network can therefore be written as

$$\sum_{i=1}^{c+1} b_{n,i} \le C^*{}_{max,n}, \tag{11}$$

where c is the number of channels traversing node n at the time the request for the establishment of the new channel (numbered $c+1$ in the node) is received, and $b_{n,i}$ is the number of buffers needed for the i-th channel in node n. The delay bound in the node for channel i is

$$dp_{max,nrt} + dp_n \sum_{k \ne i} b_{n,k} + dp_n, \tag{12}$$

where $dp_{max,nrt}$ is the maximum time spent processing a non-real-time packet if there is no preemption.[8]

Since we must choose for the node the maximum value of this bound, and the maximum value corresponds to the case in which $\sum_{i=1}^{c_{\max}} b_{n,i} = C^*{}_{max,n}$ and $b_{n,k} = 1$ for some channel k, we have

$$d_{max,n} = dp_{max,nrt} + dp_n\ (C^*{}_{max,n} - 1) + dp_n = dp_{max,nrt} + dp_n\ C^*{}_{max,n}. \tag{13}$$

This approach does not make efficient use of the network's resources: the larger the jitter, the smaller the number of channels that can be supported; one could try to increase this number by increasing $d_{max,n}$ (see Equation (13)), but this also increases the maximum jitter (see Equation (10)), and the resulting growth of b_m (see Equation (9)) may partially or totally offset the effects of the increase in $d_{max,n}$. An improvement can be obtained by using a distributed jitter control scheme such as the one described in [Ferr91], which, under the same assumptions on which (10) is based, reduces the maximum jitter at the input to node m to

$$j'_{in,m} = d_{max,m-1} - dp_{m-1}. \tag{14}$$

[8] If there is preemption, $dp_{max,nrt}$ must be replaced in (12) by the time it takes to preempt a non-real-time packet.

Expression (9) is still valid, as long as the value of $j_{in,m}$ in it is replaced by that of $j'_{in,m}$ given by (14). The validity of (11), (12), and (13) is also unaffected. If all nodes are assigned the same delay bound $d_{max,n}$ and have the same value of dp_n, then $C^*_{max,n}$ is the same for all nodes (see (13)), but the $b_{n,i}$'s are identical, and do not depend on the node's position along the route of a channel (except for the first node, whose input jitter is zero). The number $C_{max,n}$ of real-time channels node n can support in this case, when the scheduling policy is J-FCFS (i.e., FCFS preceded by a jitter-control *regulator* [Verm91]), satifies the inequalities

$$C^*_{max,n} \left\lceil \frac{2d_{max,n} - dp_n}{x_{\min}} \right\rceil^{-1} \leq C_{max,n} \leq C^*_{max,n} . \qquad (15)$$

If we are not allowed to make all of the above assumptions, the admission criteria and the expression of $d_{max,n}$ are more complicated, but it is still true that the delay bound in a node is decided once the admission control policy is chosen, and does not depend on the requirements of the client. An FCFS (or J-FCFS) node therefore is, in some sense, a "black-box node": if a channel request can be accepted, the local delay bound for the channel is fixed, and cannot be tailored to the client-specified end-to-end delay bound. A channel traversing only such nodes, if and when it is created, is assigned an end-to-end delay bound smaller (in some cases, much smaller) than the one desired by the client. For instance, if all N nodes are identical and satisfy the above assumptions, the end-to-end delay bound is equal to $Nd_{max,n}$. This is clearly a disadvantage of FCFS and J-FCFS with respect to Multi-Class EDD: as a result of this lack of flexibility, the network's resources are likely to be less utilized, delay and jitter bounds are much less tight, and larger amounts of buffer space are needed.

The problem from which FCFS and J-FCFS suffer can be mitigated by introducing several queues for real-time channels, with different priorities and hence different delay bounds. For example, if there are two kinds of channels in a node, with maximum capacities $C^*_{max,n,h}$ for higher-priority ones and $C^*_{max,n,l}$ for lower-priority ones, and the higher-priority queue is always non-preemptively served before the lower-priority queue, we have

$$d_{max,n,h} = C^*_{max,n,h}\ dp_n + \max\,(dp_n, dp_{max,nrt}), \qquad (16)$$

$$d_{max,n,l} = C^*_{max,n}\ dp_n + dp_{max,nrt}, \qquad (17)$$

where $C^*_{max,n}$ is the same as in single-level FCFS or J-FCFS. We can use Equation (16) to determine the value of $C^*_{max,n,h}$ that produces the local delay bound $d_{max,n,h}$ we want, and compute $C^*_{max,n,l} = C^*_{max,n} - C^*_{max,n,h}$. The admission criterion at either level is obtained from (11) by replacing $C^*_{max,n}$ with either $C^*_{max,n,l}$ or $C^*_{max,n,h}$, and c with the number of channels existing in the node at that level. Extending this approach to multiple queues and multiple levels of priority, it is possible to offer several discrete values for the local delay bound in each node, and to match more closely the requests of the clients. The number of queues, the maximum number of channels that can be accommodated at each level, and hence the corresponding values of $d_{max,n}$ have, however, to be chosen *a priori* and remain fixed for relatively long periods of time, thereby creating a potential for inefficient allocations, mismatches, and unnecessary rejections of channel requests.

An idea similar to that of multi-level FCFS is the basis of the Hierarchical Round Robin (HRR) scheduling discipline [KaKK90] [ZhKe91], a multi-level generalization of Round Robin (RR). RR satisfies Conditions (2)b and (3); if the maximum number of slots in an RR cycle is fixed, then also Condition (4)b is satisfied. Otherwise, (4)a must be satisfied, and RR is used to schedule real-time packets, while the policy for scheduling non-real-time packets is immaterial. If Condition (1) is also met, then RR is capable of bounding delays, though, like FCFS, RR suffers from lack of flexibility and inefficient use of resources. The inefficiencies can be partially cured by introducing J-RR, in the same vein and with similar results as for J-FCFS, and the lack of flexibility can be mitigated by HRR.

HRR introduces multiple levels of RR cycles, all of which are served in an RR fashion, though with decreasing rates. Packets are served during fixed time slots, and each slot at different levels corresponds to a different amount of bandwidth and has a different delay bound. As in the multi-level FCFS scheme, bandwidth allocation and local delay bound vary together and cannot be independently chosen.

So far in this section, only deterministic delay bounds have been considered. Statistical delay bounds can be obtained by overbooking (for example, with single-level FCFS scheduling, by admitting more than $C_{max,n}$ real-time channels into node n). However, this approach makes all the channels through the node statistical, and does not allow us to provide any of them with deterministic delay guarantees. To allow for the coexistence of deterministic and statistical channels, at least two queues need to be used: for instance, in the two-level FCFS scheme described above, one could overbook the lower-priority queue and send all packets arriving on statistical channels to that queue; this scheme unfortunately forces the local delay bound of statistical channels to be greater than that of deterministic channels, at least under the assumption that dp_n and $dp_{max,nrt}$ are equal or very close. This restriction (which cannot be mitigated by introducing multiple levels of deterministic and/or statistical queues, unless their scheduling is made substantially more complicated than described above) is an advantage of Multi-Class EDD over simpler scheduling disciplines.

With disciplines based on "slots" that are assigned to the various channels, such as RR and HRR, statistical bounds can be obtained by overbooking some slots, i.e., by multiplexing them among two or more statistical channels. Even though the calculation of late delivery probabilities may be complicated, this method allows deterministic and statistical channels to coexist in the same node, and offers the destination of a new channel several values of $z_{min,n}$ to choose from for that channel in node n.

6. A Case Study

In this section, we apply some of the ideas discussed in the preceding sections to a specific situation. We first design a real-time communication service in a network whose nodes are scheduled by the FCFS, J-FCFS, and two-level J-FCFS disciplines, and then connect that network to the Xunet 2 internetworking testbed.

The network to be studied is called Casenet; it is a local-area network consisting of four hosts A, B, C, D and four nodes a, b, c, and d, interconnected as shown in Figure 3. The real-time component of the traffic on Casenet consists of identical continuous-media streams, each with an interpacket interval $x_{\min} = x_{ave}$ equal to 15 time units, and a packet processing delay dp_n equal to 1 time unit in each node.

The nodes obey the model shown in Figure 2(a), where the queueing server has at least two queues, a higher-priority queue for real-time packets, and a lower-priority queue for non-real-time packets (Condition (4)a in Section 5). The packets and local tasks in the lower-priority queue never occupy the node's CPU for more than 1 time unit ($dp_{max,nrt} = 1$). Each source controls its packet generation rate so that this rate never rises beyond $1/x_{\min}$ packets per unit time (Condition 2(a) in Section 5). The application just requires that the continuous-media stream packets be delivered within a given deterministic delay bound and without losses due to buffer overflow.

The initial version of Casenet, called Casenet 1, is expected to offer a real-time communication service using the FCFS discipline to schedule the packets in the higher-priority queue of each node. We must choose either the value of $d_{max,n}$ or that of $C^*_{max,n}$ for each node. Assuming that the network is uniform even in this respect, we set $d_{\max} = 15$ time units, and derive $C^*_{\max} = 14$ from (13). Equations (10) and (9) yield $b_1 = 1$, $b_2 = 2$, $b_3 = 3$, and $b_4 = 4$. Since propagation delays are negligible, the contribution of the network to an end-to-end delay bound equals the sum of the local delay bounds in the nodes traversed by the channel. Thus, for example, a channel connecting host A to host D has a delay bound of 60 time units (plus the delays in the upper-layer protocols running in the hosts), one from A to C a delay bound of 45 time units, and one from A to B a delay bound of 30 time units. A channel desiring a delay bound of 100 time units between A and D is given a 60 time unit bound anyway, and one between the same two hosts requiring a delay bound of 50 time units is not established, unless the admission criteria to one or more of the nodes are made stiffer.

Table 1

Effects of channel establishment requests in Casenet 1

Channel requested		Number of buffers in				Actual delay
hosts	delay bound	a	b	c	d	bound
AB	35	1	2			30
CA	39	-	-	-		-
AC	50	1	2	3		45
AD	72	1	2	3	4	60
BD	28		-	-	-	-
DA	68	4	3	2	1	60
DB	51		3	2	1	45
AD	56	-	-	-	-	-
DC	33			2	1	30
CB	26		-	-		-
BA	34	2	1			30
CA	52		*			*
BD	48		1	2	3	45
CD	36			*		*
Total buffers in node		9	14	14	10	
Total channels in node		5	7	6	5	

- request denied: delay bound too low

* request denied: insufficient buffers and delay bound in node

These observations are confirmed by Table 1, which shows the results of a specific sequence of channel establishment requests submitted one by one (so that a decision on each has been made and implemented before the next request is issued) to Casenet 1. A total of 8 channels are established, while 6 requests are rejected; 4 of these cannot be accepted due to the too sequenceent delay bounds required by the corresponding clients, and the other 2 because of the insufficient room (in terms of both delay bound and buffer space) available in the node indicated by the asterisk in the table.

In the second version of our network, called Casenet 2, the real-time packets in each node are scheduled by the J-FCFS discipline described in Section 5. Again, we set $d_{\max}$ to 15 time units, and obtain $C^*_{\max} = 14$ time units from Equation (13), as in Casenet 1. However, the maximum jitter at the input to *any* node along a path is now 14 time units (see Equation (14)), and $b_m = 2$ buffers for $m = 2, 3, \ldots$ (see Equation (9)), while $b_1 = 1$ buffer. Inequalities (15) tell us that the maximum number of channels in a node is between 7 and 14.

If Casenet 2 were to receive the same sequence of channel establishment requests as Casenet 1 in our Table 1 example, the results would be those shown in Table 2. Out of 14 requests, 10 would be successful; the 4 channels whose delay bounds are smaller than those the network can offer would be the only ones that would not be established. The nodes would be better utilized (by the real-time components of the traffic), as they would collectively be traversed by 28 node-channels instead of 23.

Table 2

Effects of channel establishment requests in Casenet 2

Channel requested		Number of buffers in				Actual delay
hosts	delay bound	a	b	c	d	bound
AB	35	1	2			30
CA	39	-	-	-		-
AC	50	1	2	2		45
AD	72	1	2	2	2	60
BD	28		-	-	-	-
DA	68	2	2	2	1	60
DB	51		2	2	1	45
AD	56	-	-	-	-	-
DC	33			2	1	30
CB	26		-	-		-
BA	34	2	1			30
CA	52	2	2	1		45
BD	48		1	2	2	45
CD	36			1	2	30
Total buffers in node		9	14	14	9	
Total channels in node		6	8	8	6	

- request denied: delay bound too low

Further improvement in real-time capacity can be achieved by replacing the single higher-priority queue in each scheduler with two queues. This third version of our network, based on the two-level J-FCFS scheduling policy, is called Casenet 3. We keep the lower-priority value of $d_{\max}$ at 15 time units, but set the higher-priority value to 7 time units. Equation (17) yields $C^*_{\max} = 14$ as for the previous two versions of Casenet; from Equation (16), we derive $C^*_{max,h} = 6$; hence, $C^*_{max,l} = 8$. For the higher-priority channels, Equation (14) yields $j'_{in,m} = 6$, and from Equation (9) we have $b_m = 1$ for all m. Lower-priority channels have the same values of $j'_{in,m}$ and b_m as in Casenet 2.

To compare this case to those of Casenet 1 and Casenet 2 under the same sequence of establishment requests, we choose the following strategy: we try to assign each new channel to the lower-priority class; if the delay bound offered by the lower-priority service is too large, or if the lower-priority queue is full in one of the nodes along the channel's path, then the channel is assigned to the higher-priority class; if this class is also full in at least one of the nodes to be traversed, or the offered delay bound is still too large, then the request is rejected. Note that this priority assignment policy does not exploit the possibility that the same channel be treated as a lower-priority one in some nodes and as a higher-priority one in the others. This more flexible policy would be more complicated to implement and slightly more expensive at runtime (as it would also require indexing into the channels table in each node for each packet), but it would generally reduce the difference between the requested and the offered delay bounds, and might therefore increase the real-time capacity of the network.

Table 3

Effects of channel establishment requests in Casenet 3

Channel requested		Number of buffers in				Actual delay
hosts	delay bound	a	b	c	d	bound
AB	35	1	2			30
CA	39	**1**	**1**	**1**		**21**
AC	50	1	2	2		45
AD	72	1	2	2	2	60
BD	28		**1**	**1**	**1**	**21**
DA	68	2	2	2	1	60
DB	51		**1**	**1**	**1**	**21**
AD	56	**1**	**1**	**1**	**1**	**28**
DC	33			2	1	30
CB	26		**1**	**1**		**14**
BA	34	**1**	**1**			**14**
CA	52	*	*	*		*
BD	48		*	*	*	*
CD	36			**1**	**1**	**14**
Total buffers in node		**3**+5	**6**+8	**6**+8	**4**+4	
Total channels in node		7	10	10	7	

* request denied: insufficient buffers and delay bounds in node

boldface entries correspond to higher-priority channels

Table 3 shows the responses of Casenet 3 to our sample sequence of channel establishment requests. All the channels that could not be created in the two previous versions of the network because of too low delay bound requirements can now be established. Only 2 of the 14 requests cannot be satisfied due to space shortages or queue saturation. That nodes are now collectively traversed by 34 node-channels instead of 28 (Casenet 2) or 23 (Casenet 1) is a confirmation of the greater real-time capacity provided by the two-level scheduling discipline.

If a version of Casenet were to be connected to Xunet 2, what would be the conditions under which the resulting internetwork could offer real-time services (as defined in Section 2)? For simplicity, but without losing generality, we shall focus our discussion on a channel from A to R, and assume that node d of Casenet is to be connected to either node 1 or node 2 of Xunet 2; connection to node 1 requires that SH (see Figure 1(a)) have a Casenet interface as well as an FDDI interface, and that both networks run the same real-time internetwork protocol (e.g., RTIP [VeZh91]); connection to node 2, in the context of a channel from A to R, could result from interfacing node d either to the FDDI ring on the SH side of the route shown in Figure 1(a) or to SR directly. The former solution is shown in Figure 4(a), the latter in Figure 4(b). In both cases, hosts A and R are able to communicate with guaranteed performance only if they run the same real-time transport protocol (e.g., RMTP [VeZh91] or CMTP [WoMo91]).

If, on the other hand, the two networks run two different real-time internetwork protocols, their connection must go through a protocol-converting gateway, as shown in Figure 4(c). This solution is to be adopted also when real-time services are requested from a backbone such as the one connecting SR and DR (i.e., nodes 2 and 3) in Figure 1. Let us consider, for example, the case of an organization that wants to connect two local-area or metropolitan-area networks, geographically far from each other, via an existing backbone (see gateways GW and FW in Figure 4(c)). The two networks belonging to the organization provide performance guarantees; in order to extend such guarantees to connections involving both networks, also the backbone must guarantee performance bounds, but the organization cannot or does not want to convert the networks to the real-time protocols that run on the backbone and on the other networks attached to it. The gateway in this case must convert internetwork protocols into one another on a packet-by-packet basis, and run the Tenet channel establishment and teardown protocol.

Since Figures 4(a) and 4(b) are very similar, we may focus our discussion on the route in Figure 4(a). Let Casenet 3 be the version of Casenet to be connected to Xunet 2. If the Casenet 3 portion of the channel is classified as high-priority, that portion could be regarded as a single black-box logical link with respect to delays, characterized by a deterministic bound of 28 time units. The same conclusion may be reached for loss probabilities (with the allocation of buffers reported in Table 3, the contribution of the black-box link to the end-to-end probability of no losses would be 1) and for delay jitters (the maximum jitter produced by the black-box link would be 6 time units, i.e., the maximum jitter at the output of node d).

It is remarkable that no changes to the parameters or to the procedure for channel establishment in each node of either network have to be made for their connection to become operational. The only modifications required are those to names and addresses, and the necessary extensions to the contents of name servers.

A black-box approach to the treatment of Casenet 3 would be less convenient if one of the two levels of priority could not be effectively chosen during the forward trip of the establishment message; the white-box approach, which would consider all four Casenet nodes as level-1 nodes, would assign the task of selecting the level of scheduling priority to the destination (R). The same process would be followed if the level of priority of a channel could vary from node to node. The gateway GW in Figure 4(c) would be treated by the channel establishment scheme as though it were a source host for Xunet 2 like SH in Figure 1(a), and a destination host for Casenet 3; since Casenet would be governed by a different real-time internetwork protocol, R would not have jurisdiction over Casenet, and the values of the performance requirements for Xunet 2 would be computed by GW on the basis of the results obtained for Casenet. If the channel goes through FW into another network, the destination's functions must be performed by FW itself, which therefore emulates R (or, to be more precise, host DH in Figure 1(a)).

7. Conclusions

In this paper, we have discussed the extendibility of the Tenet approach to internetworks. Having defined such a service in terms of performance guarantees offered to clients (i.e., minimum and minimum-of-averages throughput bounds, deterministic and statistical delay bounds, deterministic jitter bounds, and reliability bounds), we considered a hierarchical model of an internetwork, and derived simple conditions to be satisfied by the level-1 path of a channel if performance guarantees must be provided to that channel.

The conditions for each of the logical links in the path led to similar conditions for level-2 paths, then for level-3 paths, and so on, until this hierarchical decomposition resulted in paths to which the original Tenet approach can be applied. An important observation is that the need for a hierarchical decomposition does not force us to abandon the single round-trip establishment procedure that is a very useful feature of the Tenet approach: indeed, the original procedure can be easily adapted to a hierarchical structure, making it capable to change levels whenever necessary during the establishment process in a general, uniform way, no matter how deep the hierarchy is.

A key reason for the viability and the generality of our scheme is that the end-to-end performance bounds of a channel can be decomposed into per-node terms that are independent of each other, as well as independent of the terms that represent the same node's contributions to other channels' bounds. Among other effects, this independence allows different channels to be treated differently: for example, the same logical link may be white-box for a channel and black-box for another; for a given channel, it may be white-box with respect to some bounds and black-box with respect to other bounds. Similar conclusions can be reached for the scheduling disciplines in the nodes: even though the discipline that schedules real-time packets at each output link of each node must be the same[9] for all packets leaving the node via that link, there is no need for all output links to adopt the same discipline, or for a channel to require the same scheduling discipline in all the nodes it traverses.

A necessary condition for exploiting these results in an internetwork is that scheduling disciplines other than Multi-Class EDD can be used for real-time services in a simple network. Our investigations of the conditions under which a scheduling discipline can be used in the implementation of a real-time service showed that the class of real-time packet scheduling policies that satisfy those conditions is quite large. When a policy is unable to support a sufficiently large number of real-time channels, adding to the scheduler in each node a regulator for distributed jitter control and/or (if possible) multiple queues at different levels of priority is likely to increase its efficiency.

Much work remains to be done to increase the generality and the flexibility of the prototype real-time protocols the Tenet Group has designed and implemented. This effort must be accompanied by investigations of the costs and benefits of simple disciplines, such as FCFS and Round Robin, in their various versions (with distributed jitter control, or multilevel queues, or both).

Acknowledgments

[9] Note that channels without distributed jitter control may coexist in a node (and on an output link) with channels whose packets have their jitter reduced to zero within that node (for example, FCFS may coexist with J-FCFS).

The author is grateful to all members of the Tenet Group, and in particular to Anindo Banerjea, Bruce Mah, Jean Ramaekers, Giorgio Ventre, Dinesh Verma, and Hui Zhang, for the many enlightening discussions on the subjects treated in this paper. Anindo Banerjea, Jorg Liebeherr, and Giorgio Ventre gave detailed comments on a previous version of the manuscript, and contributed substantially to the readability of the current version.

References

[AnHS90] Anderson, D., R. G. Herrtwich, and C. Shaefer, "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.

[BaKe92] Banerjea, A., and S. Keshav, "Queueing Delays in Rate Controlled Networks," Tech. Rept. TR-92-015, International Computer Science Institute, Berkeley, CA, March 1992.

[Ferr91] Ferrari, D., "Design and Applications of a Delay Jitter Control Scheme for Packet-Switching Internetworks," Proc. Second International Workshop on Network and Operating System Support for Digital Audio and Video, Heidelberg, November 1991.

[FeVe90a] Ferrari, D., and D. C. Verma, "A Scheme for Real-Time Channel Establishment in Wide-Area Networks," *IEEE J. Selected Areas in Communications,* SAC-8, April 1990, 368-379.

[FeVe90b] Ferrari, D., and D. C. Verma, "Buffer Allocation for Real-Time Channels in a Packet-Switching Network," Tech. Rept. TR-90-022, International Computer Science Institute, Berkeley, CA, June 1990.

[FKKM91] Fraser, A. G., C. R. Kalmanek, A. E. Kaplan, W. T. Marshall, and R. C. Restrick, "Xunet 2: A Nationwide Testbed in High-Speed Networking", submitted for publication, July 1991.

[JoSe87]

[KaKK90] Kalmanek, C. R., H. Kanakia, and S. Keshav, "Rate Controlled Servers for Very High-Speed Networks," *Proc. GlobeCom '90 Conf.*, San Diego, CA, Dec. 1990, 300.3.1 - 300.3.9.

[LaPa91] Lazar, A. A., and G. Pacifici, "Control of Resources in Broadband Networks with Quality of Service Guarantees," *IEEE Communications Magazine*, 29, 10, October 1991, 66-73.

[LiLa73] Liu, C. L., and J. W. Layland, "Scheduling Algorithms for Multiprogramming in a Hard Real Time Environment," *J. ACM*, 20, January 1973, 46-61.

[Part91] Partridge, C., "Isochronous Applications Do Not Require Jitter-Controlled Networks," Network Working Group, Request for Comments 1257, September 1991.

[PaTu90] Parulkar, G., and J. Turner, "Towards a Framework for High-Speed Communication in a Heterogeneous Networking Environment," *IEEE Network Magazine*, March 1990, 19-27.

[Topo90] Topolcic, C., Editor, "Experimental Internet Stream Protocol, Version 2 (ST-II)," Network Working Group, Request for Comments 1190, October 1990.

[Verm91] Verma, D. C., "Guaranteed Performance Communication in High Speed Networks," Ph.D. Dissertation, University of California at Berkeley, November 1991.

[VeZF91] Verma, D. C., H. Zhang, and D. Ferrari, "Delay Jitter Control for Real-Time Communication in a Packet Switching Network," *Proc. TriComm '91 Conf.*, Chapel Hill, NC, April 1991, 35-43.

[VeZh91] Verma, D. C., and H. Zhang, "Design Document for RTIP/RMTP," unpublished Tenet Group report, May 1991.

[WoMo91] Wolfinger, B., and M. Moran, "A Continuous Media Data Transport Service for Real-Time Communication in High Speed Networks", *Proc. Second International Workshop on Network and Operating System Support for Digital Audio and Video*, Heidelberg, Germany, November 1991.

[Zhan90] Zhang, L., "Virtual Clock: A New Traffic Control Algorithm for Packet Switching Networks," *Proc. ACM SIGCOMM Conf. '90*, Philadelphia, PA, September 1990, 19-29.

[ZhKe91] Zhang, H., and S. Keshav, "Comparison of Rate-Based Service Disciplines," *Proc. ACM SIGCOMM Conf. '91*, Zuerich, Switzerland, September 1991, 113-121.

Fig.1. A simple host-to-host route in the Xunet 2 testbed (a), and its level-1 model (b). Xunet 2 is an internetwork consisting of FDDI local-area networks and a wide-area ATM backbone; its level-1 view contains the hosts, the routers, and logical links representing paths through these three level-2 networks. SH: source host; SR: source router; SW: ATM switch; DR: destination router; DH: destination host; P: source (running the sending application and the higher-layer protocols); R: destination (running the higher-layer protocols and the receiving application).

Fig.2. The two node models to which the original Tenet approach to real-time communication applies. The model in (a) is a classical single-server queue. The rectangular box at the input in (b) represents a non-blocking, self-routing interconnection network; the only queueing in (b) occurs at each output link. Note that each of the queueing servers in the picture may actually consist of several queues (e.g., in the original Tenet scheme, there are three queues, for deterministic, statistical, and non-real-time packets, respectively, feeding the same queueing server).

Fig.3. The topology of Casenet. The graphical notation is the same as in Figure 1(b); thus, node a shares the same computer with host A, b with B, and so on.

Fig.4. Connecting Casenet to Xunet 2. The figure represents the path of a channel that has its source on host A and its destination on host R. In (a), the two networks are connected by virtue of the direct connection of nodes d and 1. In (b), there is a direct connection between nodes d and 2. A gateway GW between node d and 2 connects the networks in (c). Gateway FW illustrates a possible arrangement for allowing Casenet and another network connected to FW to obtain guaranteed-performance services from the ATM backbone between nodes 2 and 3 in Xunet 2.