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 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

---

[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.

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

---

[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.

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]

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;

---

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

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]

---

[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.

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:

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

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.

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

---

[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.

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 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