Infocom 1994: Toronto, Canada; June 12-16, 1994

An Evaluation of Scheduling Mechanisms for Providing Best-Effort, Real-Time Communication in Wide-Area Networks*

Henning Schulzrinne
GMD-Fokus
Berlin, Germany
hgs@fokus.gmd.de

Jim Kurose and Don Towsley
Department of Computer Science
University of Massachusetts at Amherst
{kurose,towsley}@cs.umass.edu

Abstract

We distinguish between four types of service that may be provided to real-time traffic by packet-switched networks, ranging from "need-blind" and "need-based best-effort" to "guaranteed throughput" and "bounded delay jitter" services. We evaluate a number of scheduling policies that offer need-based, best-effort service. We introduce hop-laxity (HL) scheduling which is based on the time remaining until the packet must reach its destination as well as the number of hops separating it from the destination. HL scheduling is evaluated through simulation and has been implemented within a BSD-based kernel and tested on the DARTnet network. The results indicate that HL scheduling tends to equalize delays between calls with large and small number of hops as compared to a FIFO discipline, reducing the 99.9% percentile of delay and the fraction of late packets. We compare HL scheduling to the FIFO+ discipline suggested by Clark et al.., and find that their delay properties are similar. Other disciplines, such as minimum laxity or transit priority, may actually do more harm than good.

1 Introduction

The increasing bandwidths available in wide-area networks have made applications with real-time constraints, such as interactive voice and video communications, feasible. For example, there has been a significant increase in interest in the use of packetized audio and video over wide-area, packet-switched networks, as exemplified by the live audio- and (slow scan) videocast of the Internet Engineering Task Force (IETF) meetings since early 1992 over the Internet to participants at several hundred sites in a dozen countries [1].

Traditional communication applications such as file transfers and electronic mail are examples of "non-real-time" traffic. The performance metric for such applications is typically average delay and throughput. The characteristics of *real-time* communication applications differ significantly. Most importantly, each real-time packet is characterized by a *deadline* by which it must be received at its intended destination; a packet not received within this deadline is typically considered to be lost. Unlike traditional communication applications, many real-time applications can tolerate such loss, provided that the amount of loss is kept within certain limits. In such cases, the primary performance metric for this class of traffic is the fraction of packets received within their deadline.

These different performance metrics suggest that protocols and architectures previously developed for traditional communication applications may not be well-suited for real-time communication applications. Therefore, we examine the use of different *link-level scheduling policies* for supporting real-time traffic in wide-area packet-switched networks. The role of a scheduling policy is to choose a packet to send from among those queued for transmission on an outgoing link at a network routing/multiplexing point. We evaluate several scheduling algorithms through implementation and experimentation in a cross-country T1-rate network (known as DARTnet [2]), and through simulation.

Our results show that in some circumstances, policies that schedule packets on the basis of deadline information can provide better performance than a simple FIFO discipline. However, in other situations, the performance improvement under deadline-based scheduling is found to be relatively small, particularly when the additional overhead required by deadline-based scheduling is accounted for. Our implementation studies also allow us to assess the various costs associated with different approaches towards implementing deadline-based scheduling.

The remainder of this paper is structured as follows. In Section 2 we discuss the problem of scheduling real-time traffic, discuss past research in the area, and describe the scheduling algorithms to be considered in this paper. The main results of this paper are presented in Sections 3, 4 and 5. In Section 3 we discuss the experimental evaluation of the so-called HL policy (described in section 2) in the DARTnet network and compare its performance with simple FIFO scheduling. In Section 4 we use simulation to compare the performance of HL and several other policies, including the recently-proposed FIFO+ algorithm [3], and in Section 5 we examine the effect of an increasing number of hops on the performance of the real-time scheduling algorithms, in a symmetrically loaded network. Section 6 concludes this paper.

*Supported in part by the Defense Advanced Research Projects Agency under contract NAG2-578. This work was performed while the first author was at the University of Massachusetts.

2 Service Classes and Scheduling Policies for Real-Time Traffic

2.1 Service Classes

Before describing the details of the scheduling policies to be considered in this paper, it may be helpful to provide a framework in which to place these policies. We can distinguish four classes of service that can be provided by a network. (We note that these are not all possible service classes, but merely those which can be implemented by known policies.)

1. **Bounded delay jitter:** The network admits only a certain number of flows and can guarantee that all packets of a flow experience queueing delays between given upper and lower bounds. Barring the degenerate case of all lower bounds being zero, the node has to implement a non-work-conserving queueing discipline, where packets are delayed even if the channel is idle. Hierarchical round-robin [4], jitter-earliest-due-date [5] and Stop-and-go [6] are examples of jitter-bounding policies.

2. **Guaranteed throughput:** In this case, the network assures that each flow can obtain its guaranteed throughput regardless of the behavior of other flows. In addition, such policies have to limit the time period over which a flow can claim accumulated bandwidth credit, so as to avoid unduly delaying other streams. Weighted fair queueing (WFQ) [7], Virtual Clock [8] and similar algorithms fall into this category. These scheduling policies are generally work-conserving. For traffic that is shaped by a leaky bucket at the edge of the network, the per-flow delay can be upper-bounded. The bound is tight in the sense that some packets may actually experience the delay bound for certain traffic patterns [9].

3. **Best-effort, "need-based":** While making no guarantees about its performance, the network tries to provide the performance best suited for the particular type of carried traffic. No upper bounds have been put forward for policies implementing this class. *Subsequently, we will only consider this class.*

4. **Best-effort, "need-blind":** The network makes no commitments about its service characteristics, although some networks, by their physical properties, can guarantee that packets will not be reordered, for example. Also, the queue scheduler does not take the type of service or service requirements of the packet into account. The current Internet clearly falls into this category, as IP datagrams can take an arbitrary amount of time to reach the destination, if they reach it at all. Under certain restrictions on the source traffic and its peak rates, even for FIFO scheduling, deterministic and probabilistic delay bounds can be computed [10, 11].

Why would a real-time application request anything but the first class of service, i.e., guaranteed jitter? The first reason is that a network guaranteeing jitter bounds has to set aside bandwidth based on the application's peak needs. Since many real-time applications have periods with low bandwidth needs, for example, silence periods for audio or no-motion periods for conference video, only a small fraction of the allocated bandwidth will actually be used. Thus, the number of supportable applications in a jitter-controlled network is smaller than in networks with less stringent requirements, raising transmission costs. (It should be noted, however, that the network may recover some of the the unused bandwidth reserved for first class service by "filling the gaps" with low-priority fourth-class service.)

Another important advantage of need-based queueing policies is the ability to shift delay from one flow to another. First, flows traversing short paths with mostly uncongested nodes can assume some of the delay of long-haul flows without noticeably degrading their own perceived quality of service. Thus, need-based scheduling disciplines can help ensure uniform and distance-independent quality of service[1]. Second, since adaptive applications cannot make use of low delays available only for an occasional packet, low-delay packets with time to spare might as well be delayed to give better service to other flows more in need. For services with rigid deadlines, the latter rationale becomes even stronger, as packets arriving before the deadline do not improve the quality of service seen by the application.

Perhaps surprisingly, there are also "greedy" reasons why a real-time application may opt for second or third class service. First, the application may be able to obtain service when first class bandwidth is not available due to network congestion or network provider policies. Also, the guaranteed QOS provided by the first class service may actually be *worse* than the QOS provided by second or third class service, since for first class service, the channel may be idle even though a packet is ready to be transmitted. For example, in [12] all non-zero values of the delay distribution under Stop-and-go queueing were found to lie far beyond the end of the delay distribution under FIFO scheduling.

Clearly, all four classes of service will have their place in an integrated-services network, with decreasing cost from jitter-controlled to need-blind, best-effort service. However, it should be emphasized that better service, higher network utilization, and higher call acceptance rates may result by not automatically assigning real-time services to the jitter-controlled class. (See also [13] for a more elaborate argument why isochronous services do not require an isochronous, that is, jitter-free network.)

2.2 Objectives for Scheduling Policies

As indicated above, the primary performance metric for evaluating packet scheduling disciplines for real-time traffic is the fraction of packets arriving at their destination after their deadline has expired, or

[1] We have come to expect this, for example, in the telephone network.

equivalently, the *tail* of the end-to-end delay distribution. Beyond this basic performance measure, however, other considerations should enter into evaluating the merits of the candidate policies:

Local information only: For general applicability, a scheduling policy should depend only on local queue information, the local node state, and information carried by data packets. Obtaining data from downstream queues will usually not be feasible for multicast.

Bandwidth overhead: Almost all need-based policies have to carry additional information needed by the service policy, increasing network utilization. This may be of particular concern for the short packets that are often found with real-time applications (e.g., packet voice).

Computational cost: A scheduling policy is invoked either at the packet enqueueing time or the packet dequeueing time. All policies we examine, except FIFO, have to inspect every packet in the queue when either inserting the new arrival or when picking the next packet to be transmitted. With appropriate double buffering, however, this is not of concern (from a link utilization standpoint) if the computation can be overlapped with the transmission time in either case.

Robustness: "First, do no harm". The performance of the scheduling policy should not degrade below FIFO even if the parameters (where applicable) are not set exactly right. If the policy estimates local queue behavior, the effect of noisy estimates needs to be determined, as changing loads and bursty traffic will likely make high-precision estimation difficult.

2.3 Performance Evaluation

For the purposes of evaluating real-time scheduling policies, we distinguish two different types of applications. Fixed-deadline applications have a known, constant deadline; here, the appropriate metric is the fraction of packets exceeding that deadline. Adaptive-deadline applications can adapt their deadline, within bounds, over longer periods of times, say seconds or minutes. Adaptive applications will set their short-term deadline such that the fraction of packets exceeding that deadline is small. Thus, an appropriate performance measure is the delay percentile, as it gives an indication how tight the application can set the adaptive deadline.

2.4 Scheduling Policies

We now describe the link-level scheduling policies which will be considered in the remainder of this paper.

First-in, first-out (FIFO): Packets are served in order of arrival.

Transit priority (TP): Packets that enter the network at a given multiplexing point defer to those passing through that multiplexing point (so-called transit traffic). Implemented as two queues, the complexity for enqueueing and dequeuing is independent of the queue length [14].

FIFO+: This policy is proposed by Clark, Shenker, and Zhang [3]. Each node estimates the average delay for all packets of a given class passing through it, for example, by using a first-order estimator. Each packet carries a priority field that is initialized to zero as the packet enters the network. As a packet departs from a node, the difference between the queueing delay actually experienced at the node and the average delay for all packets of the class is added to the priority field. Thus, a positive value in the priority field indicates that the packet has spent more than the class average in the queues traversed so far. Conversely, a negative value indicates that the packet has received better-than-average service. At any queue, packets which have exceeded their class average delay by the largest amount, are served first.

Hop laxity (HL): Packets are served smallest-per-hop-laxity-first. Here, laxity is a measure of the time remaining until the packet's deadline expires, i.e., until its time in the network exceeds the deadline. More precisely, the laxity measure determining the order of service at a packet's ith hop is given by the ratio of laxity to hops remaining,

$$d_i = \frac{d + M - a}{M - i + 1},$$

where a is the age of the packet, that is, the time between its creation and the current time slot at the node, d is the deadline for packet reception at the destination host, and M is the total number of hops between the source and destination. While waiting, the d_i value of a packet decreases with a rate that depends on the number of nodes left to travel. At an output link, the packet with the lowest value of d_i is always transmitted next. Ties between packets with the same laxity are broken in favor of the earliest arrival. This local time-based priority captures the system time remaining until extinction, scaled by the number of nodes that the packet still has to traverse. Thus, packets tend to be "compensated" by a higher priority for above-average delays suffered at earlier nodes. This policy is identical to the "Budgeted Residual-Life Dependent Discipline" (BURD) in [15], developed independently of our HL policy [16].

Minimum laxity (ML):
The earliest-extinction-first "seniority" discipline is similar to HL, but based solely on laxity, without regard to the number of hops left to travel, i.e., $d_i = d + M - a$. Again, the packet with the lowest value of d_i is transmitted. If all deadlines are the same, the policy of selecting the packet

with the largest accumulated queueing delay is equivalent to ML.

Variations on local earliest-deadline-first or minimum-laxity (ML) scheduling are quite common [8, 17–20], as ML is known to be an optimal policy in many single-queue circumstances. Note, however, that the ML policy we consider here is meant to be end-to-end, rather than hop-by-hop.

HL requires $O(N)$ operations, with N as queue length, for dequeueing since the relative ordering of packets may change at every dequeue instant. With appropriate data structures, the sorted lists for ML, WFQ and FIFO+ can be maintained with $O(\log N)$ enqueue and $O(1)$ dequeue complexity. Only FIFO+ requires estimation of the delay average, which needs to be maintained as flow state by the router. Note that all proposed policies are local in the sense that they depend only on information contained within the packets themselves plus the local queue state. FIFO+, HL and ML require additional header fields. For fixed-length packets, none of the policies reorders packets within a flow, while for flows with variable-length packets, FIFO+ and HL may both reorder packets since the priority of a later packet may exceed that of a predecessor.

3 An Experimental Implementation and Evaluation of HL Scheduling

This section describes an implementation and evaluation of HL scheduling within an operational router. Such measurements performed on an actual kernel implementation, as opposed to simulation, allow us to gauge the effect of protocol overhead and queue processing costs on the overall algorithm performance, providing a more realistic picture of the achievable performance gains. It also establishes how readily non-FIFO disciplines can be integrated into a "standard" Unix-based operating system and existing network protocols. Issues of set-up, largely unaddressed by simulation experiments, are also discussed in some detail, as they drastically affect the usability of the proposed scheduling mechanism.

In Section 3.1, we describe the network used for the experiments. The HL algorithm requires some additions to existing network protocols, IP in this case, as described in Section 3.2. We also describe some design trade-offs, the initial distribution of algorithm parameters and the changes to the kernel network modules that implement the protocol. Section 3.3 then describes the audio traffic sources used in the experiments. The experimental results are summarized and discussed in Section 3.4.

3.1 The Experimental Network

All network experiments were performed within DARTnet, the ARPA-sponsored research IP network [2] spanning the continental United States. This network does not carry regular Internet traffic and can be reserved for the exclusive use of individual research groups. Sun SPARCstation 1+, currently running a modified SunOS BSD 4.3-based kernel, serve as routers, allowing modification of packet scheduling, instrumentation, implementation of new routing algorithms, etc., while providing a high-level operating environment. Nodes within DARTnet are connected by T1-lines running at 1.344 Mb/s.

3.2 Protocol and Operating System Support for Laxity-Based Scheduling

3.2.1 Extensions to the Internet Protocol

The HL scheduling mechanism was implemented under the Internet Protocol (IP) [21]. The standard IP protocol does not support time stamp or hop count information, but is extensible through options. Since none of the standard options convey the needed information, we define a new experimental IP option. The format of the option is shown in Fig. 1; details not discussed here are described in the standard. The copy bit is one since fragments should be treated as the original packet. The newly defined option encompasses a total of eight bytes. If the *drop expired* bit is set, packets that have exceeded their laxity are dropped. If that bit is zero, scheduling priority increases with negative laxity. The *min. laxity* bit switches from HL to a simple minimum laxity algorithm by not decrementing the remaining hop count from its initial value of one.

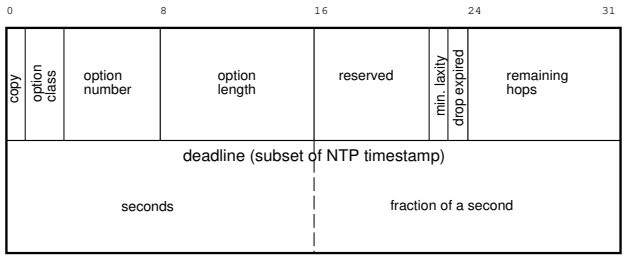

Figure 1: The HL IP option

The *remaining hops* octet, a signed integer, contains the number of hops that the datagram has to travel until reaching its destination. It is decremented by one at each router. In a connectionless protocol such as IP, the path length (hop count) can only be approximately determined, since there is no guarantee that any two packets travel the same route through the network. However, for most existing networks, the hop count and even the route change infrequently. Thus, in our experiments we initially established the path length when setting up a transport association and assumed it remained constant during the call duration.

Our HL algorithm also requires information about the time remaining to expiration. This information can be encoded in the packet, for example, as the time remaining or an absolute deadline, i.e., the latest possible arrival time at the destination. We chose the latter, mainly because we wanted to eliminate state information in the routers and avoid the measurement of packet residence times.

Note that carrying an absolute deadline value requires clocks to be synchronized to within a few milliseconds. This is well within the current state-of-the

art, as evidenced by the sub-millisecond synchronization found within DARTnet [22]. The deadline value uses the middle 32 bits of the 64-bit timestamps of the network time protocol (NTP) [22], with a resolution of about 15 μs.

3.2.2 Kernel Modifications

Before describing the actual kernel implementation, we briefly outline the procedures a data packet traverses before leaving the network interface. A general description of the BSD networking code can be found in [23]. The data flow is also depicted in Fig. 2.

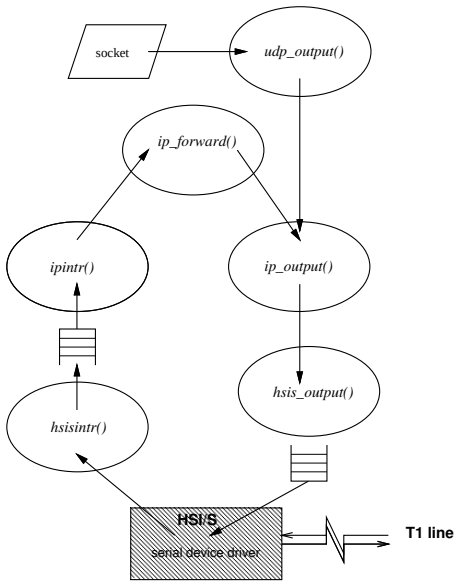

Figure 2: Data flow in BSD kernel for IP

For our purposes, we are only concerned with transmitting an IP packet. The actual implementation of the scheduling algorithm consists of two functions. The first function, `ip_hl_opt`, called early in `ip_output`, places a translated version of the IP option described above into a data structure located at the beginning of the first mbuf of the packet. The first 16-bit word of the mbuf data area designates the resource control family, so that multiple scheduling policies can be supported. The mbuf is then re-typed as `MT_DATASORTED`, a newly defined mbuf type. This translation requires a linear search through all IP options within the packet to determine if the scheduling option is present. The module also decrements the hops-to-go field.

Instead of transcribing the IP option in `ip_hl_opt`, we could have the dequeueing routine parse the IP options directly. This, however, has three severe disadvantages. First, not every packet is "seen" by the output scheduler. If the device is idle when the device driver is invoked, the packet is directly copied to the interface buffer, without going through the output scheduler. Thus, any hops-to-go adjustments need to take place outside the dequeueing routine. Second, it is undesirable to embed knowledge about the network layer header format into a device driver. By separating the IP layer representation and the data structure seen by the driver dequeueing routine, several network layer protocols could transparently use the same queueing mechanism. Third, since each packet may have to be inspected several times while residing in the queue, access to the scheduling information has to be as fast as possible.

Within Berkeley Unix, the actual packet scheduling takes place when the device driver completes an output operation and pulls the next packet off the output queue.

3.2.3 Scheduling Overhead Considerations

The HL algorithm imposes a significant processing burden at packet dequeueing time. However, for the serial interface used in the experiments, channel utilization should not have been affected, for the following reason. The HSI/S device driver manages two 1500-byte output buffers. While one output buffer is being output to the serial line by the on-chip DMA controller, the other buffer can be filled with the next packet to be transmitted. Thus, line output overlaps with dequeueing and copying from system memory to board buffer. For long queues of very short packets, however, it is possible that selecting the next packet to be output may take longer than the packet transmission time, thus idling the channel.

Measurements were made to estimate the cost of HL on the DARTnet routers. The cost consists of the delay incurred each time the output dequeue routine is invoked plus the sum of the processing times for each packet in the queue. The first component consists of obtaining the current system time, converting it into the shortened NTP timestamp, which takes a total of about 10 μs, and the linked list manipulation to extract the selected packet from the queue.

The dominant cost for HL, however, is caused by having to compute the scheduling priority of each packet in the queue, entailing a division operation, and then determining the minimum priority value. With some arithmetic optimizations, each packet within the queue takes about 8μs to process during each call to the dequeueing routine. With long queues, this can clearly be a significant burden for the processor, particularly if it has to attend to a number of busy interfaces. It should be noted, however, that typical implementations of bandwidth management mechanisms such as fair queueing or virtual clock also require searching on average half the queue for each output packet to determine the proper insertion point. However, typically, no arithmetic beyond a simple comparison is required.

3.3 Traffic Sources

For the DARTnet experiments, the traffic was generated from traces based on actual network transmissions rather than from statistical models.

The audio traffic was derived from a packet trace recorded during a 52 minute packet voice conversation across the Internet. The conversation was recorded

using the NeVoT network voice terminal [24]. Each packet of length 180 bytes (plus headers) represented 22.5 ms of 8-bit μ-law audio sampled at 8,000 Hz. For experimental reasons, the packet length was increased to 300 and 600 bytes (to increase link utilization), including headers. A total of 1643 talkspurts and silence periods consisting of a total of 50468 audio packets were transmitted, for an average rate of 16.1 packets per second or an audio rate of 23.1 kb/s at the original packet size.

From the traces, network traffic was generated by a special-purpose program running on each network source node. These programs, rather than actual audio or video sources, were used since the remote site offered no easy way to remotely execute codecs. Also, the overhead of processing actual audio and video data would have made it impossible to run the number of sources at each node that would have been required in order to appropriately load the network. To ensure approximate statistical independence of flows, the traces were read with different initial offsets, i.e., one flow would read the trace starting at the first packet, the next at packet 2000, and so on.

3.4 Experiments

Our first experiment comparing the measured network performance of the HL and FIFO policies had the set of network flows depicted in Fig. 3. The thick arrows originating at node `parc` and terminating at node `mit` indicate the flow of interest, while the thinner arrows represent cross traffic. Each flow consists of seven multiplexed conversational audio traces, as discussed earlier. The assignment of flows to links assures equal loading of each link with the equivalent of 21 voice sources.

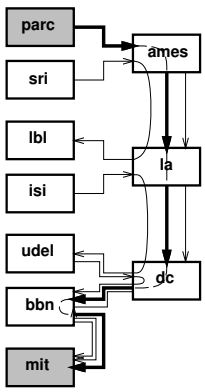

Figure 3: Traffic flows for DARTnet experiment

The flows were assigned an end-to-end laxity of 500 ms. Table 1 shows the average and various percentiles of delay under both FIFO and HL scheduling for the five-hop `parc` to `mit` traffic flow. HL shows a noticeable improvement over FIFO, even taking into account the additional overhead requirements of HL scheduling. The performance improvement given a fixed application deadline can also be estimated from the table. For example, if the application imposes a deadline of 163 ms, losses are reduced from 1% for FIFO to below 0.1% for HL. Of course, the shorter delays received by the five-hop traffic is "balanced" by an increase in delay in the few-hop traffic flows, as illustrated in the second experiment.

	mean	delay percentiles		
	delay	0.90	0.99	0.999
policy	(ms)	(ms)	(ms)	(ms)
FIFO	60.5	80.1	162.5	243.0
HL	55.5	64.8	99.9	133.5

Table 1: End-to-end delay performance (in ms) measured in DARTnet (topology of Fig. 3)

In a second experiment, DARTnet was used to provide the topology described in [3]. The network configuration is shown in Fig. 4. Twenty-two statistically identical flows (the same flows described above) traverse the network, all flowing in the same direction. Twelve of these streams are of length one hop, four each of length two and three, and the remaining two streams reach all nodes for a length of four hops. Each link is shared by ten of the streams for an overall link utilization of slightly over 80%. All flows belong to the same class. In our experimental studies, nodes 1 through 5 were represented by DARTnet nodes `parc`, `ames`, `la`, `dc` and `bbn`, respectively. Here, all packets had a length of 600 bytes, including headers.

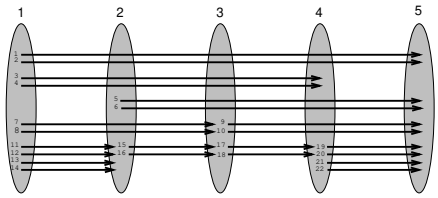

Figure 4: The traffic streams for tandem network

The end-to-end delays (including transmission and propagation delays) of the streams reaching the node `bbn`, are summarized in Table 2. 'HL x' refers to HL with a laxity value of x ms. From the table, we conclude that the performance between FIFO and HL does not differ significantly, indicating that in the case of this topology and set of traffic flows, the performance gains attributable to delay-sensitive scheduling are outweighed by the additional processing costs and packet header overhead. It should be noted that the goal of delay equalization has indeed been reached, by reducing the delay spread between long-haul and short-haul flows from 238 for FIFO to 99 ms for HL.

A third set of experiments (not reported here) used several lecture voice and video streams multiplexed at each source, confirming the basic results shown here.

4 A Comparison of HL, FIFO+, FIFO, ML and TP Scheduling

In the previous section we focussed on the HL policy, its implementation, and a comparison of its per-

	path length							
	1		2		3		4	
traffic	mean	99.9%	mean	99.9%	mean	99.9%	mean	99.9%
FIFO	18.6	370	57.6	529	73.5	590	87.1	608
HL 200	26.3	515	59.9	632	59.9	632	81.4	661
HL 400	26.4	500	56.9	584	65.9	553	74.7	599

Table 2: End-to-end delay performance measured in DARTnet (topology of Fig. 4)

formance with that of simple FIFO scheduling. In this section we compare the performance of HL, FIFO+, FIFO, ML and TP scheduling using simulation.

We again consider the networks and associated flows shown in Figure 4 (see also Figure 1 in [3]). The two-state Markov source, described in [3], is used to generate the audio traffic for each connection. We use this model order to directly compare our results, with the FIFO+ results in [3][2]. In this source model, a source alternates between idle and active periods. The idle period is exponentially distributed with a mean of 29.4 ms, while packets arrive deterministically at the end of 5.88 ms time slots. The number of packets during an active period is geometrically distributed with a mean of 5 packets, for an average active period of 29.4 ms and an average rate of 85 packets/second per source. Note that an active period may contain zero packets and that the minimum packet separation is 5.88 ms. Each packet has a fixed length of 1000 bits. The source is filtered with a (r, b) token bucket regulator. Here, the average rate r is set to 85 packets/second, the token bucket depth to 50 packets. The regulator discarded roughly 2% of the generated packets, so that the original per-node load of 0.85 was reduced to about 0.83. The source model does not represent directly either a voice or video stream but rather a generic bursty arrival process, with a peak rate twice the average rate.

For this set of simulation experiments, the network link speed was assumed to be 1 Mb/s. With packet sizes of 125 bytes, the per-packet transmission time is then 1 ms.

Discrete-event simulations ran for 62 (simulated) minutes. The first two minutes were discarded to reduce the effect of the initially empty system on the results. For the FIFO+ policy, a first-order recursive filter with a weighting constant of 0.001 estimated the mean queueing delay. (The value was chosen to coincide with that used by Clark *et al.*.) We could only achieve results close to those published in [3] by combining FIFO+ with transit priority.

The queueing delays, i.e., *not* including transmission times or propagation delays, determined by our simulations are displayed in Table 4. The number after HL indicates the laxity parameter chosen, in milliseconds.

For easier comparison, the 99.9 percentiles for the delay from Table 4 are also shown in Fig. 5. The performance of ML is not plotted since the 99.9 percentile behavior of ML is uniformly and significantly worse than FIFO.

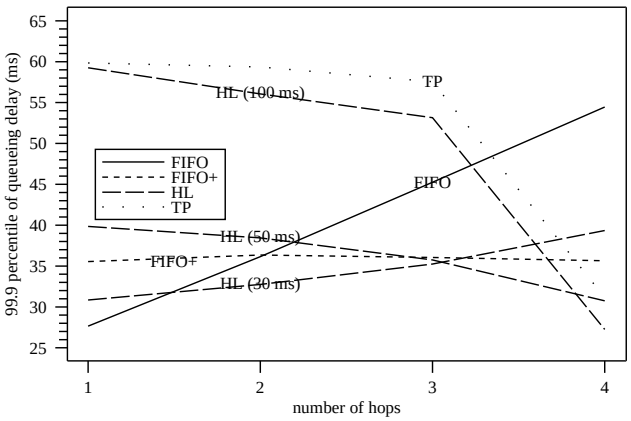

Figure 5: 99.9-percentile values for queueing delays in tandem network

The simple transit-priority (TP) policy shifts too much of the delay from the long-haul paths to the short-haul paths. This effect can be partially explained by the particular topology, as the 4-hop traffic in this example never has to yield to other traffic; when the 4-hop traffic enters node 1, all other traffic is also entering the network. Through the laxity parameter, HL can be tuned to achieve a constant or even decreasing end-to-end delay (as a function of hop count). Queueing delays decreasing with hop count may be desirable to compensate for longer propagation delays. For long paths, HL cuts the queueing delay by half compared to FIFO. The delay performance of HL as a function of path length seems to strongly depend on the value of the laxity parameter.

HL and FIFO+ are clearly the policies with the best performance in the configuration studied here. FIFO+ and HL require roughly the same amount of computation. HL can be used for both FIxed-deadline and adaptive-deadline applications (see Section 2.3). In the adaptive-deadline case, the choice of the HL deadline provides an additional degree of freedom, but may also lead to suboptimal behavior if it is chosen inappropriately. In the fixed-deadline case, the HL laxity value will usually be chosen to be the application deadline. HL offers a straightforward way to pro-

[2]Note, however, that the delays reported in [3] differ somewhat, apparently due to a problem with the random number generator used (L. Zhang, private communication).

		path length							
		1		2		3		4	
source	policy	mean	99.9%	mean	99.9%	mean	99.9%	mean	99.9%
M (CSZ)	FIFO	2.54	30.49	4.73	41.22	7.97	52.36	10.33	58.13
	FIFO+	2.71	33.59	4.69	38.15	7.76	43.30	10.11	45.25
	WFQ	2.65	45.31	4.74	60.31	7.51	65.86	9.65	80.59
M	FIFO	2.60	27.65	4.99	36.15	7.28	45.25	9.84	54.45
	FIFO+	3.23	35.55	4.98	36.35	6.44	36.05	7.76	35.65
	HL 30	4.87	30.85	4.94	32.75	4.99	35.25	4.03	39.35
	HL 50	5.29	39.85	4.93	38.45	4.83	35.75	3.35	30.75
	HL 100	5.50	59.25	4.92	56.06	4.82	53.15	3.10	27.25
	TP	4.47	59.85	4.54	59.35	5.05	57.55	4.26	31.15
	ML	4.14	58.45	4.88	101.8	5.43	119.4	5.21	128.0

Table 3: Delays (in ms) for tandem network of Fig. 4 with two-state Markov source (M); results from [3] marked with "M(CSZ)"

vide different classes of service to different real-time streams according to their urgency; FIFO+ can only achieve this indirectly through a strict scheduling priority mechanism. FIFO+ requires specification of the estimator parameter; the parameter cannot be used to influence the delay performance.

5 The Effects of Symmetric Loading and Path Length

The DARTnet and tandem topologies considered in the previous two sections were quite sparse and consisted of a relatively small number of hops. In this section we consider the performance of FIFO, HL, and FIFO+ in the context of a more densely connected network with a larger network diameter. The overall conclusions drawn from our earlier experimentation and simulation remain the same for these larger, more densely connected networks.

We consider a 21-node *fully connected* network with 1 Mb/s links. Calls arrive to the network and are assigned a path through the network that is uniformly distributed between 1 and 8 hops, with the nodes on the path chosen at random, but without loops. The call parameters bear resemblance to packet voice in a medium-speed network, with each call having an average rate of 39 kb/s. Calls are distributed so that all nodes are roughly equally loaded. Due to the relative duration of the calls and the simulation, the same call pattern persists throughout the simulation.

We consider two different models for the traffic generated by a call. In both models, links are loaded at 80% of capacity and packets have an exponentially distributed length of 220 bytes. In the first (Poisson) model, packet interarrival times are exponentially distributed with a mean of 45 ms. The second model again uses the two-state Markov characterization described in the previous section. However, the parameters differ: the average idle period lasts 603 ms, with each active period consisting of, on average, 24 packets spaced 20 ms apart. The second source is far burstier than the first source.

Following Section 2.3, we measured performance for both fixed-deadline and adaptive-deadline applications. For adaptive-deadline applications, Figures 6 and 7 show the 99.9 percentile of delay for paths of various lengths under FIFO, HL, and FIFO+, with Poisson sources and two-state Markov sources, respectively. (Note that each "curve" in these two figures is actually a set of eight points at integer number of hops; the points have been connected in the graph to indicate the trends in the results). As in our previous results, we see that under both FIFO+ and HL, for larger number of hops, the 99.9 percentile of delay is about 40% less than under simple FIFO queueing. (This number agrees well with the improvement seen for DARTnet and the tandem network, despite their significant differences in topology and traffic sources.) Of course, short-haul traffic has to pay the price for the decrease in delay of long-haul traffic.

For fixed-deadline applications, Figures 8 shows the fraction of packets exceeding the 100 ms deadline for exponential traffic, which is also used as the HL deadline parameter. As in the DARTnet example, we see about an order of magnitude improvement in the loss probability for long-haul traffic.

6 Conclusion

In this paper, we have examined the use of different *link-level scheduling policies* for supporting real-time traffic in wide-area packet-switched networks. We examined the performance of the various scheduling disciplines considered both through simulation and via implementation of the policies in an operational, experimental cross-country network. The implementation studies also allowed us to asses the various costs associated with different approaches towards implementing deadline-based scheduling.

Our results showed that in some circumstances, policies (such as HL and FIFO+), which schedule packets on the basis of deadline information, can provide better performance than a simple FIFO discipline. Conversely, we saw that some deadline-based scheduled policies (e.g., ML) can provide *poorer* performance than simple FIFO scheduling.

The two deadline-based policies we examined closely, HL and FIFO+, were shown to have the ad-

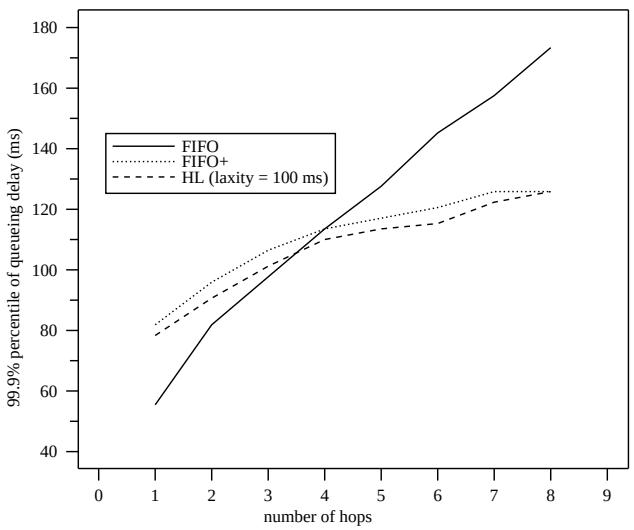

Figure 6: 99.9 percentile versus number of hops: exponential traffic in a densely connected 21 node network

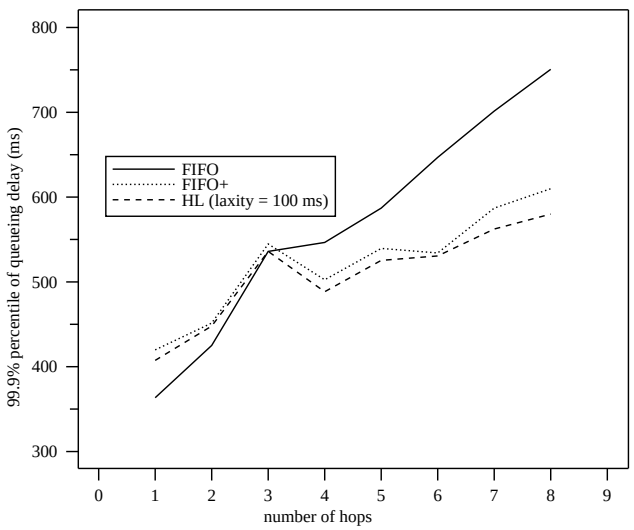

Figure 7: 99.9 percentile versus number of hops: 2-state Markov traffic in a densely connected 21 node network

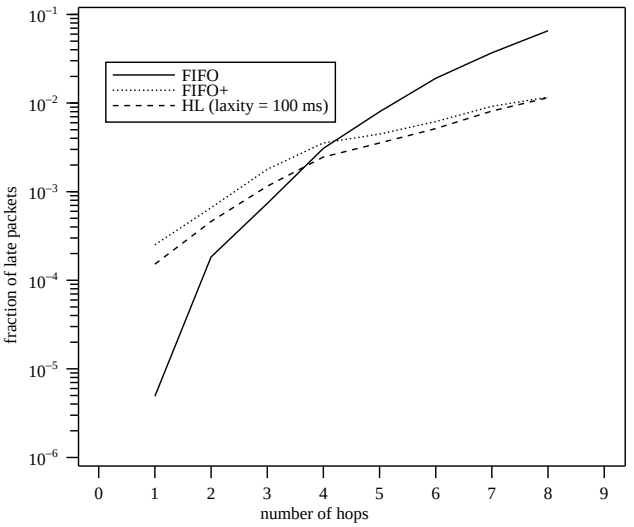

Figure 8: Loss probability versus number of hops: exponential traffic in a densely connected 21 node network

vantageous property of balancing the delays between short- and long-haul paths through the network. This generally allowed decreased delays (over FIFO) on long-haul paths to be obtained at an expense of an increase in the delay in short-haul paths, providing a more uniform and distance-independent quality-of-service.

Some specific issues remain to be addressed, such as the sensitivity of our results to different load and topology conditions. Furthermore, heuristics for choosing the FIFO+ estimation parameter and the HL laxity parameter are desirable.

A more fundamental research issue is the efficacy of the basic approach behind both HL and FIFO+, namely, that a packet be given priority "downstream" after it has suffered delays "upstream." Approaches based on this premise will only improve performance when "downstream" scheduling can have an impact, i.e., when there is more than one bottleneck node along a path. The question of whether or not multiple bottlenecks along a path are commonplace or rare (and as a consequence, whether or not dynamic scheduling disciplines such as FIFO+ or HL will be of value) is a tantalizing question for future research, particularly through the observation of operating networks.

7 Acknowledgements

The DARTnet serial device driver was written by Van Jacobson. The careful comments of one reviewer helped to improve the presentation of the results.

References

[1] S. Casner and S. Deering, "First IETF Internet audiocast," *ACM Computer Communication Review*, vol. 22, pp. 92–97, July 1992.

[2] DARPA/ISI, *DARTnet planning and review workshop*, (Marina del Ray, California), Dec. 1991.

[3] D. D. Clark, S. Shenker, and L. Zhang, "Supporting real-time applications in an integrated services packet network: architecture and mechanism," in *SIGCOMM Symposium on Communications Architectures and Protocols*, (Baltimore, Maryland), pp. 14–26, ACM, Aug. 1992. *Computer Communication Review*, Volume 22, Number 4.

[4] C. R. Kalmanek, H. Kanakia, and S. Keshav, "Rate controlled servers for very high-speed networks," in *Proceedings of the Conference on Global Communications (GLOBECOM)*, (San Diego, California), pp. 12–20 (paper 300.3), IEEE, Dec. 1990.

[5] D. Ferrari and D. C. Verma, "Scheme for real-time channel establishment in wide-area networks," *IEEE Journal on Selected Areas in Communications*, vol. 8, pp. 368–379, Apr. 1990.

[6] S. J. Golestani, "Congestion-free communication in high-speed packet networks," *IEEE Transactions on Communications*, vol. 39, pp. 1802–1812, Dec. 1991.

[7] A. Demers, S. Keshav, and S. Shenker, "Analysis and simulation of a fair queueing algorithm," in *Proceedings of Sigcomm'89: Communication Architectures and Protocols*, (Austin, Texas), pp. 1–12, ACM, Sept. 1989.

[8] L. Zhang, "VirtualClock: A new traffic control algorithm for packet switching networks," in *SIGCOMM Symposium on Communications Architectures and Protocols*, (Philadelphia, Pennsylvania), pp. 19–29, ACM, Sept. 1990.

[9] A. K. Parekh and R. G. Gallager, "A generalized processor sharing approach to flow control in integrated services networks — the single node case," in *Proceedings of the Conference on Computer Communications (IEEE Infocom)*, vol. 2, (Florence, Italy), pp. 915–924 (7A.3), IEEE, May 1992.

[10] R. L. Cruz, "A calculus for network delay, part I: Network elements in isolation," *IEEE Transactions on Information Theory*, vol. 37, pp. 114–131, Jan. 1991.

[11] J. F. Kurose, "On computing per-session performance bounds in high-speed multi-hop computer networks," in *Sigmetrics 1992*, (New Port, Rhode Island), pp. 128–139, ACM, June 1992.

[12] D. Yates, J. Kurose, D. Towsley, and M. G. Hluchyj, "On per-session end-to-end delay distributions and the call admission problem for real-time applications with QOS requirements," in *SIGCOMM Symposium on Communications Architectures and Protocols* (D. P. Sidhu, ed.), (San Francisco, California), pp. 2–12, ACM, Sept. 1993. also in *Computer Communication Review* 23 (4), Oct. 1992.

[13] C. Partridge, "Isochronous applications do not require jitter-controlled networks," Network Working Group Request for Comment RFC 1257, Swedish Institute of Computer Science (SICS), Kista, Sweden, Sept. 1991.

[14] F. Kamoun, "A drop and throttle control policy for computer networks," *IEEE Transactions on Communications*, vol. COM-29, pp. 444–452, Apr. 1981.

[15] J. Kobza and S. Liu, "A head-of-line approximation to delay-dependent scheduling in integrated packet-switched networks," in *Proceedings of the Conference on Computer Communications (IEEE Infocom)*, (Ottawa, Canada), pp. 1106–1113, IEEE, Apr. 1989.

[16] H. Schulzrinne, J. F. Kurose, and D. Towsley, "Congestion control for real-time traffic in high-speed networks," Technical Report TR 89-92, Department of Computer and Information Science, University of Massachusetts, Amherst, Massachusetts, 1989.

[17] T. M. Chen, J. Walrand, and D. G. Messerschmitt, "Dynamic priority protocols for packet voice," *IEEE Journal on Selected Areas in Communications*, vol. SAC-7, pp. 632–643, June 1989.

[18] Y. Lim and J. Kobza, "Analysis of a delay-dependent priority discipline in a multi-class traffic switching node," in *Proceedings of the Conference on Computer Communications (IEEE Infocom)*, (New Orleans, Louisiana), pp. 889–898, IEEE, Mar. 1988.

[19] J. M. Peha and F. A. Tobagi, "Evaluating scheduling algorithms for traffic with heterogeneous performance objectives," in *Proceedings of the Conference on Global Communications (GLOBECOM)*, (San Diego, California), pp. 21–27, IEEE, Dec. 1990.

[20] H. Zhang and S. Keshav, "Comparison of rate-based service disciplines," in *Sigcomm '91 Symposium – Communications Architectures and Protocols*, (Zürich, Switzerland), pp. 113–121, ACM, Sept. 1991.

[21] J. Postel, "Internet protocol," Network Working Group Request for Comments RFC 791, Information Sciences Institute, Sept. 1981.

[22] D. L. Mills, "Network time protocol (version 3) – specification, implementation and analysis," Network Working Group Request for Comments RFC 1305, University of Delaware, Mar. 1992.

[23] S. J. Leffler, M. K. McKusick, M. J. Karels, and J. S. Quarterman, *The Design and Implementation of the 4.3BSD UNIX Operating System*. Reading, Massachusetts: Addison-Wesley, 1988.

[24] H. Schulzrinne, "Voice communication across the Internet: A network voice terminal," Technical Report TR 92-50, Dept. of Computer Science, University of Massachusetts, Amherst, Massachusetts, July 1992.