IEEE Transactions on Communications, **Vol. 44, No. 2, Feb. 1996, pp. 192-202.**

Cost-Based Scheduling and Dropping Algorithms To Support Integrated Services[1]

Jon M. Peha and Fouad A. Tobagi

Abstract

Applications with diverse performance objectives must be supported on a single packet-switched network. The efficiency of such networks can be greatly improved through use of sophisticated scheduling and dropping algorithms within the queues that form at the network access points and in switches throughout the network. In our approach, arbitrary performance objectives are expressed in the form of *cost functions*, which map the queueing delay experienced by each packet to a cost incurred. Our heuristic algorithms, *cost-based scheduling* (CBS) and *cost-based dropping* (CBD), then attempt to optimize network performance as perceived by the applications by minimizing the total cost incurred by all packets. Appropriate cost functions are presented for common applications. Scheduling and dropping algorithms are defined from these cost functions. It is demonstrated that network performance is better when these algorithms are used as opposed to the common alternatives. Also, contrary to conventional wisdom, some evidence is presented that sophisticated scheduling may be preferable to sophisticated dropping as a means of adjusting loss rates.

1 Introduction

There is interest in supporting diverse applications with a single integrated packet-switched network using asynchronous transfer mode (ATM). The challenge to designers of network algorithms and protocols comes from the fact that performance objectives vary tremendously from one type of service to another. In this paper, we show how diverse application performance objectives can be met efficiently with an appropriate packet transmission *scheduling algorithm* and packet *dropping algorithm*. We also examine the relative importance of scheduling versus dropping to meeting diverse objectives.

The most appropriate measure of performance depends on the application. The traditional measure in data networks is *mean delay* of all packets. Mean delay is meaningful for an application like interprocess communication in a distributed system, but this measure conveys little about quality of service in a packet voice or video application. Such packets are buffered at the destination and played out some fixed delay after they are generated at the source. A more meaningful performance measure for such applications is *loss rate*, the fraction of packets received after their deadlines or not at all. Another possible measure is *tardiness*, which is $max(0,$ actual delay $-$ delay allowance$)$. Consider a central controller that periodically multicasts a poll to all replicas of a database and waits for responses [1]. On a path with small propagation delay, large queueing delay is tolerable; only queueing delay beyond some fixed delay allowance, i.e. tardiness, degrades performance. Finally, consider applications such as resource monitoring in which status messages are transmitted periodically. The utility of this information decays with time, until it becomes useless. An appropriate performance measure for such packets is *earliness*, which is $max(0,$ delay limit - actual delay$)$. Even when the same measure is appropriate for two applications, requirements may differ tremendously quantitatively. For example, interprocess communication generally requires a much smaller mean delay than file transfer, and tolerable loss rates for voice and video streams can easily range from O(.1%) to O(10%).

With such diverse objectives, a network designed so that every packet stream would experience performance that meets all of the most stringent objectives would be grossly inefficient [2, 3]. Thus, algorithms should differentiate packets based on performance objectives and adjust each packet's delay distribution accordingly. (In the special case of public networks, the price should also be adjusted

[1]The work was supported, in part, by the National Science Foundation under grant NCR-9210626.

accordingly [4, 5].) Once a packet's route is selected, the only delay that can be influenced is queueing delay, and this is best accomplished by the *scheduling algorithm* and the *dropping algorithm*. The scheduling algorithm, or service discipline, orders the transmissions of queued packets. The dropping algorithm selects a packet to be dropped when the buffer overflows.

Scheduling occurs at every queue. In a wide-area network (WAN), queues form in two places: at buffers in packet switches inside the WAN, and at network access points. Once an admission control algorithm has admitted a given packet stream, whenever the source decides to transmit a message, that message is in effect queued at the network access point awaiting entry into the network. (It does not matter whether this queue is considered part of the source or part of the network, as long as propagation delay from data to network is insignificant.) For example, in an enterprise network, this queue may form at the interface between the public common carrier and the private local network. The arrival rate of packets into the queue at the network access point at any given time can exceed the rate at which packets can enter the network. Buffer sizes and therefore worst-case mean delays are also less limited at access points. Consequently, queueing delay at the access points is likely to be more significant, making scheduling especially important there. In a local or metropolitan-area network (LAN/MAN), the entire network can be considered a server with a queue that is distributed among all stations. Therefore, in this paper, we address scheduling in a single queue, as is appropriate for scheduling at a WAN access point, or a LAN. Although beyond the scope of this paper, the approach can be extended to a network of queues. To demonstrate the value of sophisticated scheduling, implementation considerations are ignored in this paper. (Such issues are addressed in [6, 3].)

Section 2 reviews scheduling and dropping algorithms. Section 3 describes our approach: *cost-based scheduling* and *cost-based dropping*, and related approaches that use the *cost* abstraction. In Sections 4 and 5, the performance of the scheduling and dropping algorithms, respectively, are compared. Also, in Section 5, some evidence is presented that sophisticated scheduling is more effective in meeting diverse objectives than sophisticated dropping. Section 6 concludes the paper.

2 Current Scheduling and Dropping Algorithms

The common scheduling algorithms are *first-come first-served* (FCFS), *static priority* (SP), and *earliest deadline first* (EDF). Unlike FCFS, SP allows simple differentiation; packets are given a priority from a finite range before entering the network, and the queued packet with the highest priority is selected for transmission at every queue. Among those of equal priority, FCFS is generally used. An advantage of SP is that it is optimal when the measure of performance is weighted mean delay, where weighted mean delay is a linear combination of the delays experienced by all packets. (Priority equals the ratio of a packet's weight to its expected transmission duration [7].) However, SP does not let the urgency of delivering a packet vary with time. For example, a voice packet's priority should increase as its deadline approaches. To handle packets with deadlines, EDF has been proposed. EDF is optimal with respect to mean tardiness in discrete-time G/D/1 (i.e., slotted) queues [8]. The variant of EDF in which packets that have missed their deadlines are dropped is optimal with respect to loss rate in discrete-time G/D/1 queues and continuous-time M/D/1 queues [9]. The problem with EDF is that real traffic is generally not homogeneous. First, not all packets have deadlines (although in an EDF algorithm called *HOL-PJ* [10], packets with mean-delay objectives are assigned an arbitrary deadline, and no packets are dropped after missing their deadlines.). Even if all performance objectives are in terms of loss rate, the loss of some packets is generally more significant than the loss of others, since loss probability objectives can vary quantitatively from one packet to another. This is not considered with EDF. Disadvantages of SP and EDF are further demonstrated in [2, 3, 11].

An EDF extension has therefore been proposed [12] that differs from EDF in two ways. First, let packets be divided into classes such that class i packets are more important than class $i+1$ for

all i. A class $i+1$ packet has priority over a class i packet if the former's deadline is earlier than the latter's by at least 1 class $i+1$ packet transmission time (where a *deadline* is the time by which a packet must begin transmission). Second, the deadlines of class 1 packets are reduced where necessary such that all currently queued class 1 packets can begin transmission at their deadline without any of these transmissions overlapping. Although the broader framework proposed in [12] may have some value, there is no obvious advantage to this algorithm. If the *laxity*, i.e. time until a packet must begin transmission, of arriving packets is large compared to packet transmission time, then the first modification to EDF has little effect. As for the second modification, the justification in [12] is that their scheduling and admission control algorithms will prevent class 1 packets from ever missing deadlines. However, this claim is incorrect[2].

Some have also advocated polling-based approaches [13]-[19]. With such an approach, for example, voice packets may be given higher priority than data packets if and only if the number of voice packets transmitted in the current cycle has not yet exceeded a fixed threshold. This approach has advantages, such as insuring fairness without need for an external policing function to prevent a source from flooding the network with packets marked as voice. However, the approach does little to optimize performance. In particular, there is no way to set parameters such that a voice packet has priority over data packets only when the former is in danger of missing its deadline. Indeed, if voice arrival rate does not fluctuate, this polling-based approach is roughly equivalent to giving voice a greater static priority, with all of the corresponding disadvantages.

A more sophisticated scheduling algorithm called MARS [20] was proposed to run on top of a polling-based algorithm. A schedule is maintained that contains the fraction of cycle dedicated to each traffic class in future cycles, and that schedule is modified continually based on the arrival stream. MARS assumes all traffic can be divided into just three classes. For class 1 packets, the sole objective is that queueing delay cannot exceed some maximum which is identical for all class 1 packets. There is also a maximum delay which is identical for all class 2 packets, but some class 2 packets can miss their deadlines. Class 3 packets should simply be transmitted as early as possible, but should not degrade class 1 or class 2 performance. MARS is effective for traffic meeting its assumptions about performance objectives, but relaxing these assumptions is difficult. We hope to support more heterogeneity.

Another approach is occupancy-based scheduling. There are multiple classes of traffic, and scheduling decisions depend on how many packets of each class are queued. An example in the networking context [21] defines voice and data classes. Voice has priority over data unless the number of queued data packets exceeds some threshold. Unfortunately, data bursts cause prolonged periods when no voice is transmitted. It would be better if data had priority unless the number of voice packets exceeded a threshold, but this works poorly if voice arrival rate can fluctuate significantly [3].

The most obvious dropping algorithms are *last-come first-dropped* (LCFD), *static priority dropping* (SPD), and *earliest deadline first dropped* (EDFD), each of which is similar to one of the common scheduling algorithms described above, although the choice of dropping algorithms is orthogonal to the choice of scheduling algorithms. With LCFD, when the buffer is full, any arriving packet is dropped, regardless of performance objectives. SPD allows simple differentiation of packets; the queued packet with the smallest static priority is dropped. Like static priority scheduling (SP), SPD does not allow priority to reflect deadlines, but EDFD can. The packet with the earliest deadline is the one most likely to miss its deadline, so EDFD is the best dropping algorithm for minimizing the loss rate of equal-length packets. Like earliest deadline first scheduling (EDF), EDFD does not consider the fact that the loss of some packets is more significant than the loss of others. *Random dropping* (RND) is also advocated sometimes for fairness, but it obviously allows no discrimination.

[2]Let all packet transmission times be 1 unit. Two class 1 packets arrive at time 0 with deadlines (when transmission must *begin*) 10 and 11. Another class 1 packet arrives at time 10 with deadline 11. The algorithm in [12] may spend the time from 0 to 10 transmitting packets from other classes, so one class 1 packet would unnecessarily miss its deadline.

3 Cost-Based Scheduling and Dropping

Several scheduling algorithms capable of handling arbitrary performance objectives have been presented [22, 23, 24]. Section 3.1 discusses how arbitrary performance objectives have been expressed with *cost functions*. In Section 3.2, Clare and Sastry's algorithm [24] is presented. In Section 3.3, *cost-based scheduling* (CBS) [25, 3] is presented of which Nassehi's [23] and Fisher's [22] algorithms are special cases. A novel extension of CBS, *cost-based dropping* (CBD), is presented in Section 3.4. Then, in Section 3.5, we apply this approach to some common performance objectives.

3.1 Cost Functions

To optimize network performance, we must define some quantitative measure that reflects network performance as observed by all users, but as described in Section 1, different applications perceive and therefore measure performance differently. In scheduling theory, such diverse objectives are typically described with *cost functions*. The cost function $c_i(\tau)$ defines the cost incurred if packet i experiences a queueing delay of τ. Cost functions are chosen to best represent the application's performance objectives, so total cost incurred reflects the extent to which all performance objectives are met. For example, for a loss-rate objective, if a packet's queueing delay exceeds its allowance A, a cost C is incurred. Otherwise, no cost is incurred. This cost function is a step function. The less loss an application can tolerate, the more C should be. Cost functions are expressed in cost per unit of packet length. For example, if the value of a digitized voice packet is proportional to its length, all such packets have identical cost functions. Average cost M can be expressed as follows. For $i : 1 \leq i \leq N$, let q_i be the queueing delay experienced by packet i, and L_i be its length. $M = \sum_{i=1}^{N} L_i \cdot c_i(q_i)/N$.

3.2 Maximum Utility Scheduling for Transmission (MUST)

A heuristic called Maximum Utility Scheduling for Transmission (MUST) [24] was proposed to minimize cost M, and it generalizes previous similar algorithms developed for real-time systems [26, 27]. MUST orders all currently queued packets such that performance would be optimized by transmitting packets in this order, assuming no additional packets will arrive. Packets are transmitted in this order until the next new packet does arrive, at which time all queued packets must be ordered again. (Infinite buffer size is assumed, so no corresponding dropping algorithm has been suggested.)

The biggest problem with MUST is its complexity. Each time a packet arrives and $n-1$ packets are already queued, $n!$ different packet orders must be evaluated, where evaluating a packet order means determining the cost each of the n packets would incur. This is an $O(n \cdot n!)$ operation. For example, when n is geometrically distributed, the expected length of time required to execute the scheduling algorithm is proportional to $\sum_{n=1}^{\infty} a^n \cdot n \cdot n! = \infty$ for any $a > 0$. Thus, this algorithm's extraordinary complexity makes it impractical. The authors' solution [24] is to consider only a finite time horizon, e.g. to order the packets at time t such that cost incurred would be minimized if there were no future arrivals and if any packet still queued at time $t+h$ is dropped. This relieves complexity only if h is small, while thoroughly neutralizing MUST's effectiveness. For example, consider a packet with a step cost function, where the magnitude of the step is great. Unless $h >$ laxity, MUST would select this packet for transmission immediately, even if it has ample time before its deadline.

3.3 Cost-Based Scheduling

We now present *cost-based scheduling* [25, 3], which is designed to avoid these limitations. Selecting a queued packet i for transmission reduces the cost that will be incurred due to the queueing delay of packet i, but packet i then consumes a scarce resource: transmission time. Thus, a packet's priority reflects the estimated cost that will be saved by transmitting the packet rather than keeping it in the queue, divided by the duration of its transmission. Whenever a packet transmission completes or a packet arrives to an empty queue, the scheduling algorithm is invoked, and the packet with the

greatest priority is selected for transmission. As long as the calculation of a single packet's priority is independent of the characteristics of other $(n-1)$ queued packets, the complexity is only $O(n)$.

Since the duration of a packet's transmission is proportional to packet length, priority is the ratio of estimated cost saved to packet length. Cost per unit of packet length incurred by transmitting packet i with a queueing delay of τ is simply $c_i(\tau)$. The difficulty is in estimating cost incurred by not transmitting packet i immediately. To understand the problem, first consider a simpler special case where cost increases linearly with queueing delay. $c_i(\tau) = m_i\tau : \forall i$. Both packets 1 and 2 arrive at time t and $m_1 > m_2$. One will be transmitted immediately, and the other at time t_x. If packet 2 is selected for transmission, then a greater cost of $m_1 \cdot (t_x - t) + m_2 \cdot 0$ is incurred rather than $m_1 \cdot 0 + m_2 \cdot (t_x - t)$. Optimal scheduling can be achieved by assigning each packet i a priority equal to its cost at time t_x minus its cost at time t, i.e. $m_i \cdot (t_x - t)$ for any $t_x : t_x > t$. (Note that t_x must be the same for all packets, even though the lower-priority packet may be delayed again at time t_x by newly arrived high-priority packets.) Returning to the case of arbitrary cost functions, a similar approach is possible, but the selection of t_x becomes more important; a packet's priority is affected by a jump in cost that occurs before time t_x. Our approach is to estimate costs as if $a(\tau)$ were the distribution of anticipated delay, i.e. the time until transmission t_x, and calculate priority accordingly.

$$p_i(t) = \int_0^\infty a(\tau) \cdot c_i(t+\tau) d\tau \; - \; c_i(t)$$

The extent to which priority is affected by the cost of a queueing delay of τ is proportional to $a(\tau)$. Our scheduler should perform better if priority is more strongly influenced by imminent events, so $a(\tau)$ is always-decreasing. There is no way to determine what the optimal function is, but we have found that an exponential distribution $a(\tau) = 1/\alpha \cdot e^{-\tau/\alpha}$ outperforms other functions we have tried in a single queue. This distribution also has intuitive appeal, since it is roughly what would occur if the probability of selecting a packet for transmission, given that it is still queued, did not change over time. More importantly, as will be seen in Section 4, CBS with an exponential distribution for $a(\tau)$ leads to effective scheduling when compared to rival algorithms.

α is the mean of $a(\tau)$, so cost is estimated as if a packet will be delayed on the order of α if not transmitted immediately. For step cost functions, for example, CBS becomes SP as α approaches ∞, and EDF as α approaches 0. For any value in between, CBS considers both the packet's importance and its current laxity, so CBS generally outperforms SP and EDF. For a given traffic scenario, some value of α is optimal. Luckily, it has been shown though simulation [3] that, for a wide range of traffic types and loads varying from .5 to 1.5, performance is insensitive to α near the optimum.

The priority function for CBS with an arbitrary distribution $a(\tau)$ for anticipated delay is a generalization of the algorithms of Nassehi [23] and Fisher [22], although these two algorithms were originally expressed in very different forms. Fisher's algorithm, which only schedules packets with tardiness objectives, is equivalent to CBS when anticipated delay is uniformly distributed from 0 to the time it would take to transmit all currently queued packets.[3] Nassehi's algorithm [23] (which was designed for implementation in attempt-and-defer networks such as Expressnet [28]) has been shown [25, 3] to be roughly equivalent to CBS with an exponential $a(\tau)$ (and identical to CBS for equal-length packets). The generalization of these algorithms facilitates development of our comparable dropping algorithm. Furthermore, although it is beyond the scope of this paper, scenarios involving networks of queues can also be accommodated by choosing alternate functions for $a(\tau)$.

Assuming $c_i(\tau)$ is non-decreasing, $p_i(t)$ can only equal 0 if $c(\tau) = c(t)$ for all $\tau \geq t$. In this case, there is no penalty for giving packet i an infinite delay, so a packet can be dropped once $p_i = 0$.

[3] In Fisher's algorithm, packets can have arbitrary precedence constraints. A packet i is considered queued only if all packets whose transmission must precede packet i have already been transmitted.

This approach can also be used to determine when a multipacket message should be transmitted, or preempted. A message is preempted if all of its packets are not transmitted consecutively. The only assumption is that all packets in a message enter the queue as fast or faster than they depart. A priority is assigned to each queued message or portion thereof. Since a message fragment is useless until the entire message has been received, the cost of delaying a fragment equals the cost of delaying a message. However, transmission time of a fragment is less, so cost per unit of message length must be scaled accordingly. Let $p_{i,r}(t)$ be the priority of message fragment i, and $r : 0 \leq r < 1$ is the fraction of the message that has already been transmitted. $p_{i,r}(t) = p_i(t)/(1-r)$.

3.4 Dropping Algorithm

Cost-based dropping (CBD) is analogous to cost-based scheduling (CBS). CBD also attempts to minimize cost by allocating a scarce resource, although that resource is buffer space rather than transmission time. Thus, we assign a *dropping priority* to packet i reflecting the estimated cost that would be saved by storing it in the queue as opposed to dropping it, divided by the buffer space it will require. When the buffer overflows causing the dropping algorithm to be invoked, as many of the packets with high dropping priorities as possible are held in the queue; the low-priority packets are dropped.

The expected additional cost per unit length (of buffer space) that would be incurred if packet i is dropped, or equivalently if it has an infinite queueing delay, is $c_i(\tau)$ as τ goes to ∞. The expected cost incurred if packet i is held in the queue was estimated in the previous section using $a(\tau)$. Thus, dropping priority of packet i with queueing delay t is $d_i(t)$.

$$d_i(t) = \lim_{\tau \to \infty} c_i(\tau) - \int_0^\infty a(\tau) \cdot c_i(t+\tau) d\tau$$

Again, when queueing delay comes from a single queue such as at a network access point, an exponential distribution is used for $a(\tau)$. In the case of a message fragment, dropping priority must be scaled just like scheduling priority is. Dropping priority of message i, $d_{i,r}(t)$, where r is the fraction of the message that has been transmitted already is $d_{i,r}(t) = d_i(t)/(1-r)$.

3.5 CBS and CBD for Common Performance Objectives

We first qualitatively describe two cost functions that we expect to be common. (See [3] for a description of other likely cost functions.) We then address the selection of quantitative parameters.

As discussed in Section 3.1, for performance objectives measured in loss rate, the cost function is a step function. $c(t) = C \cdot U_A(t)$, where $U_A(t)$ is a unit step function that changes value at time $t = A$. The resulting scheduling and dropping priority functions are $p(t) = Ce^{-(A-t)/\alpha} \cdot U_A(t)$ and $d(t) = C(1 - e^{-(A-t)/\alpha}) \cdot U_A(t)$, respectively. Since $p(t) = 0$ for $t > A$, late packets are dropped. If traffic is homogeneous with step cost functions, CBS becomes earliest deadline first (EDF) and CBD becomes earliest deadline first dropped (EDFD), which is optimal. If cost functions are step functions but C varies from packet to packet, CBS generalizes an algorithm in [29]

Temporal characteristics of packet loss may also matter. For example, the loss of two voice packets is generally more noticeable if they are consecutive. One solution (without use of *dependent cost functions*, in which cost incurred by packet i depends on the queueing delay of packet j, $i \neq j$ [25, 3]) is to assign different values to C for different packets in a stream, even if they are equally valuable. For example, let loss penalty be C_1 for odd-numbered packets and C_2 for even-numbered packets. If an odd-numbered packet has a deadline of d_1 and an even numbered packet has a deadline of d_2, the odd-numbered packet has greater priority at time t ($t < d_1$, $t < d_2$) if and only if $d_1 < d_2 + \alpha \ln(C_1/C_2)$. If $C_1 > C_2$, fewer odd-numbered packets are lost. As shown in [25, 3], for a slight increase in overall

loss rate, the frequency of consecutive loss can be reduced significantly.[4] For all periodic streams with identical periods, a random element to C can also improve fairness.

Another common measure of performance is mean delay. The corresponding cost function increases linearly with delay, so priority is constant. If all traffic is of this type, the algorithm becomes static priority (SP) using optimal priorities [7]. However, even data packets become useless eventually, so a deadline is appropriate. Thus, performance is measured in mean delay for those that arrive before some deadline D, while packets that are queued past their deadlines incur an additional cost C and then stop incurring cost. Thus, $c(t) = mt$ for $t \leq D$, but $c(t) = mD + C$ for $t > D$. This objective is a combination of earliness and loss rate. Scheduling priority $p(t) = [m\alpha + (C - m\alpha) \cdot e^{-(D-t)/\alpha}] \cdot U_D(t)$, and dropping priority $d(t) = [m \cdot (D - t) + (C - m\alpha) \cdot (1 - e^{-(D-t)/\alpha})] \cdot U_D(t)$.

Quantitative parameters in these cost functions must be selected next. Consider the step function as an example. The timing of the step is just the amount of queueing delay that can be tolerated, but the step magnitude is unknown. More precisely, relative magnitudes for all packets must be found, since it is relative priorities that matter. Relative magnitudes for packets from the same application can often be derived based on application objectives. For example, one might determine the cost of losing a voice packet based on the extent to which the packet's loss would reduce signal to noise ratio, or if all packet streams advance the same "mission" as in many military or enterprise networks [24], it may be possible to quantify relative importance. However, this is not always possible. For example, consider two packet streams from unrelated applications. Stream 1 consists of voice packets for which loss rate should not exceed 5%. A loss rate of only 1% is desired for stream 2, but a stream 2 packet can tolerate more queueing delay before it is lost. Clearly the cost of losing a stream 2 packet should be greater, but to determine the ratio of costs, one must first declare the range of anticipated network conditions (load and traffic mix). Through experimentation, analysis, or simulation, one must find a ratio that meets the requirements even in the most congested anticipated state. This approach may sound inelegant, but it is required for any scheduling algorithm. Indeed, with static priority, it is not even possible to say which stream should get the greater priority without going through the same process. Luckily, with CBS, values need not be close to optimal to be effective.

4 Scheduling Algorithm Performance

Three approaches are used to evaluate scheduling algorithms. First, to qualitatively understand the behavior of the different algorithms, average cost is determined as a function of load in simple traffic scenarios. Second, we demonstrate CBS's ability to trade off the performance of one type of packet with another, thereby meeting more sets of performance requirements. Finally, we employ realistic traffic scenarios and show the performance of one traffic class as a function of load, under the constraint that performance objectives of the other traffic are met. To eliminate influence from the dropping algorithm, we assume an infinite buffer in Section 4. For all simulations in this paper, the 95% confidence interval is, *at worst*, within 5% of the value shown.

4.1 Minimizing Costs

We now compare CBS's ability to minimize cost with the common algorithms. For the scenario where efficient versions are known, the complex MUST algorithm [24] and an unachievable optimal algorithm [2, 3, 11] are also included. The latter makes optimal decisions based on complete knowledge of future arrivals, so it provides a bound on achievable performance. We also vary CBS's anticipated delay function $a(\tau)$ to show its impact.

[4] A minor modification is required if resequencing at the destination is not supported. Out-of-order transmissions are possible if interarrival time $\tau < \alpha|\ln(C_1/C_2)|$. If the ratio of costs is great, then we effectively increase τ by replacing A with $A - (\alpha|\ln(C_1/C_2)| - \tau)$ when defining the cost function for low-priority packets. When queueing delays are large, low-penalty packets are dropped before their deadlines so that high-penalty packets will not miss their deadlines.

We first consider two simple scenarios in which arrivals are Poisson. The laxity of an arriving packet is distributed as follows; for constant s and exponentially distributed random variable B, the laxity of an arriving packet is $s - B$, conditioned on the fact that $s - B > 0$. Packet lengths, cost functions, and interarrival times are all independent. For traffic set 1, average cost M is weighted loss rate, so cost functions are step functions. Step magnitudes are distributed exponentially with mean 1. Packet lengths are equal and defined to be 1. $s = 11$ and $\overline{B} = 3$. In traffic set 2, M is weighted earliness, so cost functions increase linearly until their deadlines, at which point the slope becomes 0. Both cost function slopes and packet lengths are distributed exponentially with mean 1. $s = 9$ and $\overline{B} = 3$. M is shown as a function of the load ρ for both traffic sets in Figure 1. $\alpha = 1.4$, 1.0, and 4.0 for exponential, Erlang-2, and uniform distributions, respectively. (These were found to be effective values for a wide range of traffic types and loads < 1.) For traffic set 2, we also include *latest deadline first* (LDF), since it is optimal with respect to (unweighted) mean earliness of equal-length packets.

The results for both traffic sets are similar. Performance is worst with FCFS and best (excluding the optimal algorithm) with CBS using an exponential $a(\tau)$. CBS is especially effective when load is high, which is when effective scheduling is most important. In Figure 1-a, we see that the performance difference between CBS and OPT is much smaller than the difference between CBS and the common algorithms. Also, CBS slightly outperforms MUST, despite CBS's relative simplicity. Among the simpler algorithms (SP, EDF with late packets dropped, EDF with late packets queued, LDF), the most effective algorithm varies with traffic type and load. (This is also true for traffic types not shown here and when load exceeds 1 for prolonged periods [25, 3].) CBS is effective for all traffic types and loads, which is especially important since the mix of traffic can change over time.

CBS is even better during periods of congestion. For example, load = 1, half with a loss-rate objective and delay allowance of 1000 (e.g. 3.3 ms, 150 Mb/s, 500-bit packets), and half with a mean-delay objective. SP, HOL-PJ, and POLL would yield a loss rate of 1 for the former, infinite mean delay for the latter, or both. CBS can yield a mean delay close to 1 and a loss rate of .3%.

4.2 Performance Tradeoffs

To further observe how CBS meets heterogeneous performance requirements efficiently, let traffic consist of two classes. It is projected that, during periods of heavy utilization, there will be a sustained Poisson load of .9, half from each class. (We will subsequently consider a fluctuating arrival rate.) A scheduling algorithm must therefore meet specific performance requirements for each class at this load. In Figure 2, we compare the *feasible regions*, which show which sets of performance requirements can be achieved. For MUST and SP, weights are assigned randomly with fixed probabilities that depend only on the packet's class, e.g. 60% of class 1 packets are high priority and 40% are low. This approach will transmit packets out of order, and the performance achieved is dangerously sensitive to the aforementioned probabilities, but it yields the best feasible regions MUST and SP can achieve. (This is not necessary with CBS, where all packets in a class are treated the same.) Also, we assume that no algorithm drops packets that have not yet missed their deadlines.

Figure 2-a shows the loss rates that can be achieved for each class at $\rho = .9$, $s = 9$, and $\overline{B} = 3$. For example, the feasible region for FCFS corresponds to the area on the figure bounded by the curve marked FCFS, the line where class 1 loss rate = 1, and the line where class 2 loss rate = 1. The feasible region for CBS incorporates those of SP, EDF, FCFS, and virtually all that of MUST, as well as many sets of loss requirements that cannot be achieved with any of these algorithms. More striking is the fact that virtually all loss requirements that can be accommodated with the optimal algorithm (OPT) that has complete knowledge of future arrivals can also be achieved with CBS.

Figure 2-b shows the feasible regions of FCFS, SP, EDF, HOL-PJ, the occupancy-based approach in [21] (OCC), and CBS when the performance of class 1 is measured in loss rate with $s = 9$ and $\overline{B} = 3$ as in Figure 2-a, but the performance of class 2 is now measured in mean delay. With HOL-PJ [10],

arriving packets with mean delay objectives are assigned a given laxity. Again, the feasible region for CBS includes the feasible regions of all the other algorithms, and considerably more.

Figure 3 shows the loss rates that can be achieved with the more realistic Markov Modulated Poisson Process (MMPP) traffic model. (This paper was submitted to *Trans. on Communications* long before the ground-breaking research on self-similar traffic models in [30] or related advances [31].) Channel bandwidth is 150 Mb/s. Packet arrivals are Poisson at a rate of 1.5 Mb/s times the number of active calls. New calls begin according to a Poisson process, and their durations are exponentially distributed with a mean of 100 ms. Average load is .95. Packets are 500 bits long, and are considered lost after 100 packet transmission times ($s = 100$, $\overline{B} = 0$). Feasible regions are shown for CBS, OPT, OCC (as proposed in [21]), SP, EDF, and FCFS. The results are the same; CBS outperforms the other algorithms and its performance is virtually indistinguishable from the optimum.

4.3 Typical Traffic Scenarios

To compare CBS with other algorithms in typical scenarios, we employ two traffic classes. The performance requirements of class 1 are held constant, and each scheduling algorithm is used such that class 1 requirements are met. Algorithms are compared by showing the relation between the performance required for class 2 and the load from class 2 that can be carried. Three traffic types are considered for class 1: constant bit-rate (CBR) voice, variable bit-rate high-definition television (HDTV), and image transfer. We compare cost-based scheduling (CBS), static priority (SP), earliest deadline first (EDF), and polling-based algorithms (POLL) such as [13]-[19]. Also, although the specific algorithm in [21] could not meet voice or video requirements as explained in Section 2, we consider a similar occupancy-based algorithm (OCC) in which class 1 packets have priority over class 2 if and only if the number of queued class 1 packets exceeds a threshold.

Voice traffic has a maximum delay of 30 ms, and no more than 5% of voice packets can be lost even during periods of congestion. Video packets are also lost after 30 ms, but we assume each loss is noticeable to viewers. Video uses the model which Maglaris et al. [32] found experimentally to be representative: a video stream is represented by ten CBR on-off streams with exponential on and off periods of roughly 116 and 229 ms, respectively. Results are shown for mean arrivals rate of 22.5 Mb/s and 45 Mb/s. In the case of image transfer, 30 Mb must be transferred within .5 sec. In each case, class 2 traffic consists of exponentially distributed data bursts of mean length 500 kb. These bursts arrive according to a Poisson process. Performance of class 2 traffic is measured in mean queueing delay per burst, and bursts must be transmitted first-come-first-served, not shortest burst first. Transmission is at 150 Mb/s. During periods of duration T where class 1 arrives at a constant rate, the amount of class 1 data arriving is proportional to T. Results can be achieved analytically for the CBR traffic [33], and efficiently via simulation in the other scenarios [34].

Results are shown in Figures 4 through 6. The simple SP performs poorly in each case, since class 2 packets must always be given low priority. EDF is equivalent to SP in these cases, since all class 1 packets arrive with the same laxities, and class 1 requirements cannot be met by assigning arriving class 2 packets some finite laxity [10]. POLL must give high priority to enough class 1 packets per cycle that requirements are met even at maximum arrival rates. This is almost as bad as giving voice and video packets higher static priorities, but it is valuable in the last scenario when a large burst of data arrives at once and can be transmitted gradually over time. In contrast, the threshold in OCC must be set such that requirements are met even when the class 1 arrival rate is at its minimum. This is effective for CBR traffic, but not for VBR video or image transfer, where arrival rate can be negligible for an extended period. CBS outperforms all alternatives in all cases.

5 Dropping Algorithm Performance

The dropping algorithms will now be evaluated by imposing a maximum buffer size. Simulation results are presented for two traffic sets. The cost functions of traffic set 1 are step functions, with loss penalties distributed exponentially with mean 1. In a cost function from traffic set 2, cost increases linearly until the packet's deadline. At that instant, cost jumps by 4, and remains constant thereafter as was suggested for data traffic in Section 3.5. The slopes of the cost functions are distributed exponentially with mean 1. For both traffic sets, packet lengths are equal, and the buffer can hold 10 packets, excluding the packet being transmitted. Since the ratio of the laxity of arriving packets to buffer size greatly affects performance, s is varied with $\overline{B} = s/3$. Arrivals are Poisson with load 1.5 to simulate a period of congestion. Because such extreme loads must be tolerated, $\alpha = 10$.

We first evaluate the relative effectiveness of using sophisticated scheduling versus using sophisticated dropping as a means of improving performance. An astonishing number of researchers have simply assumed that the only way to adjust the fraction of packets lost due to buffer overflow for each traffic type is by assigning heterogeneous dropping priority values. In fact, the scheduling algorithm is also important; a packet that has a high scheduling priority is unlikely to be in the buffer long enough to be dropped. We compare average cost with the most sophisticated scheduling algorithm (cost-based scheduling) in conjunction with the simplest dropping algorithm (last come first dropped) versus average cost with the simplest scheduling algorithm (first come first served) and the most sophisticated dropping algorithm (cost-based dropping). If and only if this ratio exceeds 1, then sophisticated scheduling is more effective. Figure 7 shows the ratio as a function of the maximum tolerable delay, s, for both traffic sets. This ratio is large for small s, but decreases as s grows large. This is because when s is small relative to buffer size, with a scheduling algorithm that drops packets that have missed their deadlines, queue length remains small, and the dropping algorithm is seldom invoked. However, when s is large with respect to buffer size, since the mean delay of transmitted packets is proportional to buffer size, there is less need to control queueing delays, so scheduling is less important. What is important in this case is which packets are dropped and which are transmitted, which is influenced more by the dropping algorithm.

Even with a load as great as 1.5, s must greatly exceed the buffer size (10) before the cost ratio falls below one, where sophisticated dropping is more effective than sophisticated scheduling. With future networks and applications, it is likely that s will be less than the buffer size at the network access point, making scheduling more important than dropping. Furthermore, although we have only considered scheduling in a single queue in this paper, this condition could also hold in the switch buffers that constitute a network of queues. Current memory costs little more than a penny per packet that can be stored, so large buffers make sense. Sophisticated scheduling is probably also simpler to implement than sophisticated dropping, since there is at least one packet transmission time between invocations of the scheduling algorithm. Thus, if complexity is limited by implementation considerations, and the laxities of arriving packets are not much larger than the buffer size, that complexity should probably be invested in scheduling rather than dropping. Further research is in order here.

Figure 8 shows average cost as a function of s for traffic set 1 with first come first served (FCFS) scheduling, and with CBS. With FCFS, the relative effectiveness of static priority dropping (SPD), earliest deadline first dropped (EDFD), and random dropping (RND) depends on s, but CBD performs as well or better than all the other algorithms for all s. (The same is true with traffic set 2 [25, 3].)

Figure 8 also shows that the situation is different with FCFS. The differences in performance between these dropping algorithms are much smaller. (Performance differs even less with traffic set 2 [25, 3].) When the scheduler considers laxity (or importance), there is less reason for the dropping algorithm to do the same; late (or unimportant) packets are not transmitted any way. Thus, with CBS, SPD performs about as well as CBD, and SPD is simpler to implement. Implementation is even

simpler if low-priority packets are blocked from entering the buffer when it is nearly full, and it has been argued that resulting performance is still comparable to that of SPD [35].

6 Conclusions

This paper has presented a new scheduling algorithm, *cost-based scheduling* (CBS), and a new dropping algorithm, *cost-based dropping* (CBD), to efficiently support traffic with diverse performance objectives in a packet-switched network. CBS was evaluated through comparison with a variety of scheduling algorithms, including static priority, earliest deadline first [8, 9, 10], occupancy-based scheduling [21], polling-based approaches [13]-[19], MUST [24], and an optimal algorithm [2, 3, 11]. The latter makes optimal decisions based on complete knowledge of future arrivals, so it provides a bound on achievable performance. A variety of traffic scenarios were employed. It was shown that at a given load, CBS can achieve better performance, and meet more stringent and more diverse traffic requirements than these alternatives. Furthermore, CBS can meet the same performance requirements at a significantly greater load. Indeed, CBS's ability to meet diverse performance objectives and minimize cost was close to the unachievable optimum in those scenarios where an efficient optimal algorithm is known.

By demonstrating the effectiveness of CBS over the proposed alternatives, this paper has shown that network performance and efficiency can be improved significantly with sophisticated scheduling. In some environments, like real-time systems, CBS may be suitable exactly as described in this paper, since its implementation is already simpler than alternatives such as [26]. In very high-speed networks, however, implementation can be difficult. One solution is to modify the algorithm slightly [6, 3]. As an alternative, we have developed a new set of algorithms called the *Priority Token Bank* [36], based on lessons learned from CBS. This scheduler is almost as effective as CBS, is far simpler to implement, and has the added advantage that it is integrated with an admission control algorithm.

As for dropping, with both voice and data packets, CBD outperformed the common dropping algorithms when FCFS is used with a load of 1.5. However, even with this heavy load, we found that if one must choose between implementing sophisticated dropping such as CBD and sophisticated scheduling such as CBS, sophisticated scheduling is more valuable unless the amount of queueing delay an application can tolerate (as measured in transmission periods) greatly exceeds the buffer size. In addition, when sophisticated scheduling such as CBS is used rather than FCFS, there is little difference in performance between CBD and the simpler static priority dropping (SPD).

References

[1] A. R. Downing, I. B. Greenberg, and J. M. Peha, "OSCAR: An Architecture for Weak-Consistency Replication," Chap. 4 in *Databases: Theory, Design, and Applications*, N. Rishe, S. Navathe, and D. Tal, editors, IEEE Press, 1991, pp. 55-72. Based on earlier version in *Proc. IEEE Parbase,* Mar. 1990, pp. 350-358.

[2] J. M. Peha and F. A. Tobagi, "Evaluating Scheduling Algorithms For Traffic With Heterogeneous Performance Objectives," in *Proc. IEEE Globecom-90,* Dec. 1990, pp. 21-27.

[3] J. M. Peha, "Scheduling and Dropping Algorithms to Support Integrated Services in Packet-Switched Networks," Ph.D. Thesis, Tech. Report CSL-TR-91-489, Computer Systems Laboratory, Stanford University, Stanford, CA, 94305, June 1991.

[4] Qiong Wang, Jon M. Peha, and Marvin Sirbu, "The Design of an Optimal Pricing Scheme for ATM Intergrated-Services Networks," to appear in Journal of Electronic Publishing, Special Issue on Internet Economics, University of Michigan Press.

[5] S. Tewari and J. M. Peha, "Competition Among Telecommunications Carriers That Offer Multiple Services," *Proc. 23th Telecommunications Policy Research Conference*, Oct. 1995.

[6] J. M. Peha and F. A. Tobagi, "Implementation Strategies for Scheduling Algorithms in Integrated Services Packet-Switched Networks," in *Proc. IEEE Globecom*, Dec. 1991, pp. 1733-40.

[7] J. M. Harrison, "Dynamic Scheduling of a Multiclass Queue: Discount Optimality," *Operations Research*, vol. 23, no. 2, Mar. 1975, pp. 370-382.

[8] P. P. Bhattacharya and A. Ephremides, "Optimal Scheduling with Strict Deadlines," *IEEE Trans. Automat. Contr.*, vol. 34, no. 7, pp. 721-728, July 1989.

[9] S. S. Panwar, D. Towsley, and J. K. Wolf, "Optimal Scheduling Policies for a Class of Queues with Customer Deadlines to the Beginning of Service," *J. ACM*, vol. 35, no. 4, pp. 832-44, Oct. 1988.

[10] Y. Lim, J. E. Kobza, "Analysis of a Delay-dependent Priority Discipline in an Integrated Multiclass Traffic Fast Packet Switch," *IEEE Trans. Commun.*, vol. 38, no. 5, pp. 659-65, May 1990.

[11] J. M. Peha, "Heterogeneous-Criteria Scheduling: Minimizing Weighted Number of Tardy Jobs and Weighted Completion Time," *Computers and Operations Research*, vol. 22, no. 10, Dec. 1995, pp. 1089-1100.

[12] D. Ferrari and D. C. Verma, "A Scheme for Real-Time Channel Establishment in Wide-Area Networks," *IEEE J. Sel. Areas Commun.*, vol. 8, no. 3, Apr. 1990, pp. 368-79.

[13] S. J. Golestani, "A Framing Strategy for Congestion Management," *IEEE J. Select. Areas Commun.*, Vol. 9, no. 7, pp. 1064-77, Sept. 1991.

[14] K. Kümmerle, "Multiplexer Performance for Integrated Line and Packet-Switched Traffic," *Proc. ICCC,* 1974, pp. 517-523.

[15] C. R. Kalmanek, H. Kanakia, and S. Keshav, "Rate Controlled Servers for Very High-Speed Networks," *Proc. IEEE Globecom-90,* Dec. 1990, pp. 12-20.

[16] K. Sriram, "Dynamic Bandwidth Allocation and Congestion Control Schemes for Voice and Data Multiplexing in Wideband Packet Technology," *Proc. IEEE ICC-90*, Apr. 1990, pp. 1003-9.

[17] A. Demers, S. Keshav, and S. Shenker, "Analysis and Simulation of a Fair Queueing Algorithm," *Proc. ACM Sigcomm-89*, Sept. 1989, pp. 1-12.

[18] A. K. Parekh and R. G. Gallager, "A Generalized Processor Sharing Approach to Flow Control in Integrated Services Networks: The Single-Node Case," *IEEE/ACM Trans. Networking*, Vol. 1, no. 3, pp. 344-57, June 1993.

[19] D. D. Clark, S. Shenker, L. Zhang, "Supporting Real-Time Applications in an Integrated Services Packet Network: Architecture and Mechanism," *Proc. ACM Sigcomm-92*, Aug. 1992, pp. 17-20.

[20] J. Hyman, A. A. Lazar, G. Pacifici, "MARS: The Magnet II Real-Time Scheduling Algorithm," in *Proc. ACM Sigcomm,* Sept. 1991, pp. 285-93.

[21] R. Chipalkatti, J. F. Kurose, and D. Towsley, "Scheduling Policies for Real-Time and Non-Real-Time Traffic in a Statistical Multiplexer," in *Proc. IEEE Infocom,* Apr. 1989, pp. 774-83.

[22] M. L. Fisher, "A Dual Algorithm for the One-Machine Scheduling Problem," *Mathematical Programming*, vol. 11, 1976, pp. 229-251.

[23] M. M. Nassehi, "Channel Access Schemes and Fiber Optic Configurations For Integrated-Services Local Area Networks," Ph.D. Thesis, Dept. of Electrical Engineering, Stanford University, Stanford, CA, Mar. 1987.

[24] L. P. Clare and A. R. K. Sastry, "Value-Based Multiplexing of Time-Critical Traffic," in *Proc. IEEE Milcom*, Oct. 1989, pp. 395-401.

[25] J. M. Peha and F. A. Tobagi, "A Cost-Based Scheduling Algorithm To Support Integrated Services," in *Proc. IEEE Infocom,* Mar. 1991, pp. 741-753.

[26] E. D. Jensen, C. D. Locke, and H. Tokuda, "A Time-Driven Scheduling Model For Real-Time Operating Systems," in *Proc. IEEE Real-Time Systems Symp.,* Dec. 1985, pp. 112-122.

[27] S. R. Biyabani, J. A. Stankovic, K. Ramamritham, "The Integration of Deadline and Criticalness in Hard Real-Time Scheduling," in *Proc. IEEE Real-Time Systems Symp.,* Dec. 1988, pp. 142-51.

[28] F. A. Tobagi, F. Borgonovo, and L. Fratta, "Express-Net: A High Performance Integrated-Services Local Area Network," *IEEE J. Sel. Areas Commun.*, vol. 1, no. 5, pp. 898-912, Nov. 1983.

[29] S. Pingali and J. F. Kurose, "On Scheduling Two Classes of Real-Time Traffic With Identical Deadlines," *IEEE Globecom*, Dec. 1991, paper 14.4.

[30] W.E. Leland, M.S. Taqqu, W. Willinger and D.V. Wilson, "On the Self-Similar Nature of Ethernet Traffic," in *Proc. ACM Sigcomm*, Sept. 1993, pp. 183-193.

[31] J. M. Peha, "Retransmission Mechanisms and Self-Similar Traffic," submitted for publication.

[32] B. Maglaris, D. Anastassiou, P. Sen, G. Karlsson, and J. D. Robbins, "Performance Models of Statistical Multiplexing in Packet Video Communications," *IEEE Trans. Commun.,* vol. 36, no. 7, pp. 834-44, July 1988.

[33] J. M. Peha, "Analysis of Scheduling Algorithms for Integrated-Services Networks using a Semi-Fluid-Flow Model," in *Proc. IEEE Globecom*, Dec. 1992, pp. 330-334.

[34] J. M. Peha, "Simulating ATM Integrated-Services Networks," to appear in *Proc. 29th Annual IEEE/ACM/SCS Simulation Symposium*, April 1996.

[35] H. Kröner, "Priority Management in ATM Switching Nodes," *IEEE J. Sel. Areas Commun.*, vol. 9, no. 3, pp. 418-27, Apr. 1991.

[36] J. M. Peha, "The Priority Token Bank: Integrated Scheduling and Admission Control for an Integrated-Services Network," in *Proc. IEEE Intl. Conf. on Commun. ICC-93*, Geneva, Switzerland, May 1993, pp. 345-51.

Figure 1: Average cost M, vs. load ρ, (a) traffic set 1 (weighted loss rate), (b) traffic set 2 (weighted earliness).

Figure 2: Feasible regions. Poisson arrivals. Load of .45 for each class. Class 1 performance measured in loss rate, $s = 9$, $\overline{B} = 3$. (a) Class 2 performance measured in loss rate, $s = 9$, $\overline{B} = 3$. (b) Class 2 performance measured in mean delay.

Figure 3: Feasible regions, i.e., the loss rates that can be achieved by two traffic classes, each consisting of MMPP arrivals with load of .475. $s = 100$, $\overline{B} = 0$.

Figure 4: Mean queueing delay of class 2 messages versus load ρ_2 from class 2. Class 1 is voice with load $\rho_1 = .5$, and a maximum tolerable delay of 30 ms

Figure 5: Mean queueing delay of class 2 messages versus load ρ_2 from class 2. Class 1 is VBR HDTV with a maximum tolerable delay of 30 ms.

Figure 6: Mean queueing delay of class 2 messages versus load ρ_2 from class 2. Class 1 is 30 Mb images that must be transmitted within .5 sec.

Figure 7: Average cost with FCFS and CBD divided by average cost with CBS and LCFD versus maximum laxity s. Traffic sets 1 and 2. $\rho = 1.5$.

Figure 8: Average cost for traffic set 1 versus maximum laxity s. $\rho = 1.5$. Various dropping algorithms. (a) FCFS scheduling. (b) CBS scheduling.