To appear in ACM Sigmetrics '98

Is Service Priority Useful in Networks?

Sandeep Bajaj Lee Breslau Scott Shenker*
{bajaj,breslau,shenker}@parc.xerox.com
Xerox Palo Alto Research Center
3333 Coyote Hill Road
Palo Alto, CA 94304

Abstract

A key question in the definition of new services for the Internet is whether to provide a single class of relaxed real-time service or multiple levels differentiated by their delay characteristics. In that context we pose the question: is service priority useful in networks? We argue that, contrary to some of our earlier work, to properly address this question one cannot just consider raw network-centric performance numbers, such as the delay distribution. Rather, one must incorporate two new elements into the analysis: the utility functions of the applications (how application performance depends on network service), and the adaptive nature of applications (how applications react to changing network service). This last point is especially crucial; modern Internet applications are designed to tolerate a wide range of network service quality, and they do so by adapting to the current network conditions. Most previous investigations of network performance have neglected to include this adaptive behavior.

In this paper we present an analysis of service priority in the context of audio applications embodying these two elements: utility functions and adaptation. Our investigation is far from conclusive. The definitive answer to the question depends on many factors that are outside the scope of this paper and are, at present, unknowable, such as the burstiness of future Internet traffic and the relative offered loads of best-effort and real-time applications. Despite these shortcomings, our analysis illustrates this new approach to evaluating network design decisions, and sheds some light on the properties of adaptive applications.

*This research was supported in part by the Advanced Research Projects Agency, monitored by Fort Huachuca under contracts DABT63-94-C-0073 and DABT63-96-C-0105. The views expressed here do not reflect the position or policy of the U.S. government.

1 Introduction

The Internet has traditionally provided applications with a single class of best-effort service. The performance requirements of elastic applications, such as file transfer and electronic mail, allow them to adapt to the changing delays and bandwidth provided by this service. More recently, the increasing bandwidth of Internet links as well as the increasing processing power of end hosts has focused widespread attention on the desire to use the Internet for the transport of real-time multimedia content, such as audio and video. The architecture and protocols needed to support the more stringent requirements of these applications have been the subject of considerable research and discussion in recent years. New resource reservation protocols, scheduling algorithms and admission control algorithms have been proposed in the academic literature.[1] Recently, two components of a new Internet architecture have been moved to Proposed Standard in the Internet Engineering Task Force (IETF): the reservation protocol RSVP [3, 17] and new network element services Guaranteed [12] and Controlled-Load [16] service (see also [2, 13] for overviews of this architecture). The key break from best-effort service, where sources need not notify the network before transmitting packets, is that for these real-time services flows must request service from the network – specifying their desired quality of service and their proposed traffic characterization – and the network can accept or reject their requests.

One issue that arose in the discussion of the Controlled-Load[2] service (and in the earlier Controlled-Delay service, which was supplanted by the Controlled-Load service) is whether or not to offer more than one priority level of service; that is, whether to have multiple levels of scheduling priority within the Controlled-Load service.[3] The current service definition offers only a

[1]The literature is far too vast to review here, but see [4, 5, 6, 7, 9, 15, 18] and references therein for a few representative examples.

[2]Controlled-Load service is a relaxed real-time service that provides low delay and low loss, but does not provide delay bounds.

[3]The analogous question remains even if real-time applications are supported by a best-effort network service, a solution advocated by some in the community. That is, would these applications be better suited by a single class of best-effort service (as exists today) or by multiple levels of service differentiated by delay? Our treatment remains valid, although we

single level of service, but the question remains as to what benefits offering additional levels of service would provide.[4] Offering multiple levels of service carries with it the cost of additional complexity to deal with signaling the priority level, merging reservations with different priority levels, and scheduling overhead. We do not discuss those costs here, but instead ask only how large a benefit multiple priority levels might offer. Clearly if such benefits are minimal then there is no need to incur the additional complexity; if the benefits are significant then one needs to more carefully assess the complexity costs.

At first glance, the benefit of multiple priority levels seems obvious. After all, applications have a wide spectrum of delay constraints, from interactive conferencing with its need for small network delays to playback of stored video which can easily tolerate large network delays. Given this wide disparity in delay requirements, it seems only natural that one can increase the overall welfare by offering different levels of service. One can model this analytically (following a very similar model in [14]).

In the following simple example with two kinds of applications we compare a network with two priority classes to one with a single class of FIFO service. Let U_i denote the performance level, or utility, of application i as a function of average network delay. Let V denote the total utility, or total value, of the applications: $V = U_1 + U_2$. Consider a network with a single link modeled by an exponential server (of rate $\mu = 1$) and flows modeled by Poisson arrival processes. Consider two types of network clients, with Poisson arrival rates $r = 0.25$ and with $U_1 = 21-10d_1$ and $U_2 = 2-d_2$ where d_i represents the average queueing delay delivered to client i.[5] Thus, we have two clients with different sensitivities to delay. If we use FIFO service in the network, then $d_1 = d_2 = \frac{1}{(1-0.5)} = 2$ and so $V^{FIFO} = 1$. If we use strict priority service, with preemption, and give client 1 priority, then $d_1 = \frac{1}{(1-0.25)} = 4/3$ and $d_2 = \frac{1}{(1-0.25)(1-0.5)} = 8/3$ and $V^{priority} = 7$. Thus, the strict priority scheduling algorithm is more efficient – delivers a higher value of V at the same bandwidth – than FIFO. In fact, when compared to all possible scheduling algorithms, the strict priority scheduling algorithm gives the most efficient feasible allocation of delay for this simple example.

However, this model ignores an important aspect of the problem. Specifically, network utility does not depend only on the characteristics of the packet delivery services provided, but also on how applications deal with different levels of network service. Modern network applications, in contrast to the *rigid* audio and video applications designed for more predictable data delivery services such as the telephone network or cable-TV transmission infrastructures, are *adaptive*; that is, they adapt to the current network conditions. This adaptation can take on several forms; in this paper we consider a class of applications known as *delay adaptive*. We describe these applications in more detail in Section 2. Such adaptivity is now a central piece of the accepted design philosophy in the Internet. The ability of application adaptivity to cope with changing network conditions has strong bearing on the question we ask here. After all, if adaptive applications can adjust essentially without degradation under any reasonable network conditions, there would never be any need for multiple levels of service. In fact, some have made precisely this claim when arguing for a single level of service.

Whatever the extent of adaptivity's ability to mask network delay and jitter (i.e., changes in delay), it is certainly clear that because of this active adjustment, the dependence of an adaptive application's utility on the network service is quite complicated. Simply put, such adaptivity renders simplistic analyses such as the one above invalid. How an application reacts to the network service determines how its performance depends on the network service, and the application's performance sensitivities (to delay, loss of fidelity, or both) determines what adaptation algorithm is most appropriate. As we shall see, for a given packet delivery service the delay experienced can be less in a very delay-sensitive application than in a delay-insensitive one, because the former will use an adaptation algorithm that aggressively attempts to reduce delays. Yet, despite adaptivity's centrality as an Internet application design paradigm, most performance analyses of network designs are performed without careful attention to the adaptive nature of applications.[6] The central purpose of this work is to illustrate how one can incorporate the behavior of adaptive applications into the performance analysis of a network design decision. It turns out that differences in delay (and jitter) in network service can largely be masked by this adaptive behavior for applications that are sensitive to only one of delay or fidelity. Thus, the simplistic analyses based on rigid applications are misleading. However, applications that are sensitive to both delay and fidelity achieve significant performance benefits from additional priority levels under some traffic loads. Therefore, while adaptation is very effective, it is not a universal panacea.

These results do not translate into a facile answer to the question of whether or not to offer multiple levels of Controlled-Load service. Instead, they serve as a cautionary note against simplistic conclusions based on the analysis of more static applications, and also against the ability of adaptivity to remove all sensitivity to performance variations. Our study also highlights our current state of ignorance about how the perceived performance of audio and video applications depend on the underlying network dynamics. We hope that by clarifying the gaps in the current understanding future work can begin closing them.

would need to consider different mixtures of traffic including best-effort as well as real-time applications.

[4] Offering service priority is one form of *service discrimination* within the Internet, where different packets receive different service. Service discrimination can take several other forms; a network may provide unreserved or reserved service, service may be differentiated by dropping priority, or a network may provide pre-emptable and non-pre-emptable reserved services.

[5] Recall that the average delay in the M/M/1 queueing network considered here is just $d = \frac{1}{(\mu - r)}$. If we have two priority levels, with arrival rates r_1 and r_2 respectively, then the delays are given by $d_1 = \frac{1}{(\mu - r_1)}$ and $d_2 = \frac{\mu}{(\mu - r_1)(\mu - r_1 - r_2)}$.

[6] Some work, such as in [8, 10], do analyze different adaptation algorithms, but their purpose was to refine the adaptation algorithm, not ask what implication adaptation had for network design.

The remainder of this paper is organized as follows. Section 2 describes the class of applications we consider in this paper, and then discusses several forms of adaptive behavior. Section 3 presents the results of simulation experiments that study the impact of different classes of delay on the performance of applications. We conclude in Section 4 with a discussion of the implications of our findings.

2 Adaptive Applications

In this section, we describe the class of applications that motivates our work. We begin by describing what we refer to as *adaptive* applications.[7] Then we describe two adaptation algorithms, appropriate for use by audio applications, that we use in our later simulations. We focus on these audio applications because it is in this domain that adaptive algorithms have been most widely utilized. The extent to which delay adaptation is applied to video remains to be seen, but we expect the methodology we use here could be applied to video algorithms as well. Finally, we present our model of utility functions and describe the four classes of applications we consider.

2.1 Delay Adaptation

Consider a real-time audio or video application in the Internet. Such an application will typically sample its media source (e.g., an audio input device or video frame grabber) and then send packetized data over the network. Each packet experiences a variable amount of queueing delay in the network, in addition to the fixed propagation and transmission delays (assuming all packets follow the same path). Thus, the packet stream generated by the source arrives at the destination perturbed by the variable network delay. The receiver can remove some or all of the jitter induced by the network by buffering packets for later playback. We refer to the time for which a packet is buffered at the receiver as its *playback delay*. We refer to the *playback point* as the total delay from when a packet is sent until it is played at the receiver. For real-time data, such as audio or video, if a constant playback point is maintained for all packets then there is no loss of fidelity. Otherwise, the incoming signal is distorted and so there is a loss of fidelity in the application.

Determining the playback point for each packet is a key issue in the design of these applications. Any playback strategy can make use of timestamps in packets, such as those provided by the Real-time Transport Protocol (RTP) [11], to determine the relative send times of successive packets, and thus need not assume synchronized clocks at the sender and receiver. If the receiving application knows *a priori* the maximum possible delay experienced in the network it can buffer the first packet for this maximum before playing it. This will enable the receiver to remove all jitter from the signal, since all subsequent packets (other than those that may be lost in the network) will arrive before their playback points, thereby maintaining the proper offset from the previous packet. However, neither the current Internet best-effort service, nor the proposed Controlled-Load real-time service provides applications with information about maximum network delays. While the proposed Guaranteed Service does provide delay bounds, it is an expensive service to provision (precisely because it provides delay bounds), and therefore is not likely to be widely utilized.

Since these applications must operate in environments where no end-to-end delay bound is known, they must be prepared to adjust the playback point of packets based on changing network conditions. That is, the application determines dynamically (in ways we describe below) how long to buffer each packet before playing it out. Buffering will remove some of the jitter introduced by the network, but periodic adjustments to the playback point will cause some distortion in the received signal. Hence, the application's performance is not merely a function of the service provided by the network. Rather, it is also a function of both the total delay in playing back the data (including network and playback delays) and the distortion incurred by varying the playback point over time.

Different applications will have different levels of sensitivity to these performance measures. Throughout this paper we characterize applications by the degree to which they care or do not care about each of delay and distortion. We simplify our study by considering four prototypical applications: those that care about both delay and distortion, those that care about delay only, those that care about distortion only, and those that care about neither. The notion of "caring" or "not caring" (or "sensitivity" and "insensitivity" which we use equivalently) are relative terms. For instance, even a delay insensitive application, such as the playback of recorded audio, has some delay constraints dictated by the user (e.g., delays of minutes, or several seconds, might not be tolerable). Similarly, a distortion insensitive application, such as an interactive session in which some distortion can be tolerated, also has limits to this tolerance (e.g., the speech need not be faithfully reproduced, but it must at least be intelligible). Our point, when using the terms "not caring" or "insensitive", is that these applications will be able to tolerate larger delays, or larger distortions, than other applications while still achieving acceptable performance.

2.2 Adaptation Algorithms

We expect the particular adaptation algorithms employed by delay adaptive applications to vary. For example, an interactive application may employ an adaptation algorithm that attempts to reduce the playback delay (and hence the total delay). Such a strategy, which we will refer to as *aggressive* adaptation, increases the risk that some packets will arrive after their scheduled playback points, in which case they will have to be dropped or the playback point will have to be adjusted. In either case, the resulting signal is significantly distorted. Alternatively, a non-interactive application, such as playback of recorded content or a one-way broadcast, may employ a more conservative adaptation algorithm, choosing larger playback delays and reducing the probability that packets arrive after

[7] Application adaptivity can actually take several forms. In this paper we consider the specific class of *delay-adaptive* applications. *Rate-adaptive* applications vary their sending rate in response to changing network conditions.

their playback points.

We now describe two adaptation algorithms, which we refer to as *conservative* and *aggressive*, that we use later in our simulations. These algorithms are appropriate for use by audio applications that generate blocks of data interspersed with periods of silence (as would be generated by a silence suppression mechanism). The general strategy they employ is to pick a playback point for the first packet in each talkspurt such that all packets within the talkspurt will (ideally) arrive before their respective playback points. When a packet arrives late (i.e., after it should have been played) the adaptation algorithm has two choices. It can discard the packet, or it can play the packet and adjust the playback points of subsequent packets in the talkspurt. It is unclear in general which strategy is better. For the purposes of this study we adopt the latter strategy based on previous studies (e.g., [1]) that have observed correlations in packets with large delays and on our own simulations that have shown that late packets generally arrive in bursts; given the choice between discarding several packets or introducing some jitter, the latter seems preferable.

The conservative algorithm fixes the playback point of the first packet to a predetermined value. All subsequent packets maintain the same playback point assuming they arrive in time. When a packet arrives late, the playback point is doubled and this new playback point is used for all subsequent packets. Hence, the playback point is adjusted upward but never downward. This algorithm attempts to maintain fidelity at the expense of higher delay. The second algorithm is more aggressive, yielding lower delay at the expense of increased distortion. It is taken from the adaptation algorithm in the *Visual Audio Tool (VAT)* developed at Lawrence Berkeley National Laboratory with minor modifications.[8] This algorithm estimates a measure of variance based on the difference in delay between successive packets. At the start of each talkspurt a new playback delay is computed using the previous offset and the estimate of variance.

2.3 Performance Measures and Utility Functions

The performance of an adaptive application can be characterized by two measures: delay and distortion. These measures are a function of both the packet delivery service and the adaptation algorithm. Delay includes both delays experienced in the network as well as playback delays. Distortion captures changes in the playback point. For our delay measure, we use the average of the delay experienced by each packet. Thus,

$$Delay = \frac{\sum_i d_i}{n}$$

where d_i is the delay experienced by packet i and n is the total number of packets. For distortion, an individual distortion value is first computed for each packet as follows:

$$dist_i = min(\frac{d_i - d_{i-1}}{t_i - t_{i-1}}, thresh)$$

[8] Source code for the VAT application, including its adaptation algorithm, is available at http://www-nrg.ee.lbl.gov/vat.

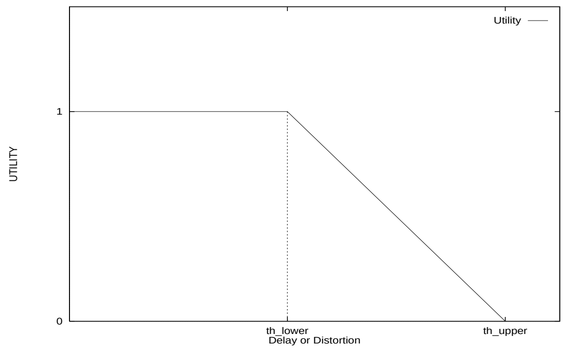

Figure 1: General form of utility functions for delay and distortion.

where t_i is the send time of packet i, and $thresh$ is a constant, set to 2 in our experiments. Including the difference in send times of successive packets in the denominator gives higher weight to intra- rather than inter-talkspurt adjustments in the playback point. $thresh$ bounds the maximum per packet distortion penalty (at 40 ms since inter-packet times, $t_i - t_{i-1}$, are 20 ms within talkspurts in our source model.) The overall measure of distortion is merely the average of the per-packet measures:

$$Distortion = \frac{\sum_i dist_i}{n}$$

We normalize these values so that they are reported in milliseconds of distortion per packet.

From the delay and distortion performance measures, we derive measures of application performance or utility using *utility* functions. The general form of the utility functions we use (for both delay and distortion) is shown in Figure 1. These functions have the following characteristics. First, below some threshold (th_{lower} in the figure), applications do not suffer any perceptible effects from delay or distortion. Second, above another threshold (th_{upper}), applications derive no utility. Finally, between th_{lower} and th_{upper}, utility degrades linearly. Total utility for an application is merely the product of its individual delay and distortion utility values:

$$U_{tot} = U_{del} \times U_{dis}$$

An application must receive good performance on both measures to achieve high overall utility, and poor performance on either leads to overall unhappiness. For each of the utility functions, we vary the values of th_{lower} and th_{upper} to capture the relative sensitivity or insensitivity of applications to each of delay and distortion. Thus, for sensitive applications, th_{lower} and th_{upper} will be set to lower values than for insensitive ones.

The relationship between performance measures and application utility is certainly not a simple as the model we use. For instance, actual functions are likely not linear, may depend on how performance varies in time rather than on static measures, and may involve subtle interactions between delay and distortion. However, we believe that our simple model captures the most important aspects of performance and utility, and at the very least is sufficient for this initial investigation. Subsequent research into the true nature of these utility functions would provide useful guidance for our modeling; at present, the relevant literature is quite sparse.

3 Simulations

We used discrete event simulation to study the effects of service priority on application utility given our model of applications and their utility described above. Our simulation environment built on version 2 of the *ns* simulator developed at the University of California at Berkeley. To the base simulator, which provides event management, measurement functions, packet transmission and traffic generation, we added additional functionality, such as adaptation algorithms, utility functions and priority queueing, needed to carry out our experiments.[9] In this section we first describe our simulation methodology, and then report our results.

3.1 Simulation Model

The purpose of our simulation experiments was to compare the utility of a network providing a single level of service for real-time applications to one providing two levels of service in the simplest possible network context. The simulation topology consisted of a single 2Mbps link connecting two nodes.[10] Each simulation consisted of a set of source/receiver pairs generating background load on the network and test applications whose performance and utility was measured. This study is concerned with real-time applications, so we assume the existence of real-time services in the network. However, since we directly control the level of offered load in our experiments (by adjusting the number of source/receiver pairs in the network), we did not need to model resource reservation or admission control functions explicitly in the simulated network. Instead, we assume that all traffic in the network has passed an admission control test, has an installed reservation, and is receiving real-time service. No best-effort traffic was included in the simulations. We discuss the implications of this later in Section 4. When testing a single level of service, all packets are served in a single FIFO queue. For priority service, we used two FIFO queues served in strict priority order (without pre-emption). When reporting our results, we refer to these as the FIFO and Priority tests, respectively. We will also sometimes refer to the high priority service in the Priority tests as Level 1 service, and the low priority as Level 2. In all experiments, offered load was controlled and enough buffers provisioned so that there were no dropped packets.

Each experiment was repeated with the test applications using the conservative and aggressive adaptation algorithms described in Section 2.2. In addition, experiments were run with a non-adaptive, or *rigid*, receiver algorithm, which we describe in Section 3.2.2. At a given level of offered load, measures of delay and distortion were computed for each algorithm (conservative, aggressive, rigid) and for each network service (FIFO, Priority Level 1, Priority Level 2). The performance measures were mapped into application utility as follows. First, we chose an appropriate adaptation algorithm for each of the four types of applications (recall the two by two taxonomy of applications based on their level of sensitivity to each of delay and distortion.) Applications that were sensitive to both delay and distortion and applications that were sensitive to delay only used the aggressive algorithm.[11] Applications that were sensitive to distortion only, and those that were sensitive to neither, used the conservative algorithm.

Given an application's performance sensitivities and adaptation algorithm, utility values for each kind of service were computed. The following values of th_{lower} and th_{upper} were used. For delay sensitive utility, we used values of $th_{lower} = 50$ ms and $th_{upper} = 100$ ms. For delay insensitive utility, we set $th_{lower} = 1000$ ms and $th_{upper} = 2000$ ms. For distortion sensitive applications we used $th_{lower} = .25$ ms/pkt and $th_{upper} = 1.0$ ms/pkt. For distortion insensitive applications, we set $th_{lower} = 2.5$ ms/pkt and $th_{upper} = 10.0$ ms/pkt. The delay sensitive values are set to represent tolerances for interactive applications.[12] The delay insensitive utility is appropriate for non-interactive playback applications, but where response time does matter to the user (i.e., pointing and clicking and receiving stored audio over the network). Deciding on distortion values for utility was difficult without a better sense of the actual effect of playback distortion on users. We chose values such that distortion sensitive and insensitive applications perceived distortion in very different manners.

Two different kinds of source models were used in the simulations. Test sources were represented by an on/off source model that generates "talkspurts" and idle periods like those generated by voice data with silence suppression. Sources transmit 200 byte packets at a rate of 80kbps during "on" periods and are silent during "off" periods. These parameters are consistent with 8 KHz 8-bit mu-law PCM audio sent in 20 ms frames with 40 bytes of overhead per packet. Both the on and off times were taken from exponential distributions with a 500 ms average.

Background traffic was generated by capturing a trace of low frame rate video taken of one of the authors during a network videoconference. The trace,

[9] The *ns* simulator is available at http://www-mash.cs.berkeley.edu/ns. Our extensions to the simulator, and the simulation scripts we ran to generate the results in this paper can be found at ftp://ftp.parc.xerox.com/pub/net-research/sigmetrics98.

[10] If, as many have claimed, there is a single bottleneck link on any network path, then the simple topology is sufficient to understand the behavior of application adaptation algorithms. The verification of this claim, or a better understanding of the effect of queueing delays at multiple hops, is a subject for future study.

[11] While it should come as no surprise that this algorithm is appropriate for delay sensitive distortion insensitive applications, it is not clear a priori which algorithm is better for applications that are sensitive to both delay and distortion. We determined, through experimentation, that the aggressive adaptation algorithm was more effective than the conservative adaptation algorithm for these applications.

[12] Note that in addition to the variable delay captured by these utility functions, there will be other fixed sources of delay.

which lasts for approximately 1,400 seconds and has an average rate of 32kbps, was produced by the *vic* video program.[13] Within a single simulation run, multiple sources sending from this trace started at random points in the trace file to avoid synchronization. This background traffic is more bursty than the traffic generated by the on/off source model. Space prevents us from presenting data using additional kinds of background traffic, such as other source models or video traces produced with different codecs or content. However, as we discuss in Section 4, additional traffic models would not provide us with a more definitive answer to our question.

Each data point in the graphs below is an average of 20 simulation runs each with different seeds to the random number generator. Individual runs lasted for 5,000 simulation seconds.

For each scenario (adaptation algorithm, service discipline) the number of sources generating background traffic was varied to generate different load levels. Our results are generally reported as a function of utilization, with each point on the x-axis representing a fixed number of background sources. These values are reported in terms of percentage of the link bandwidth generated by the background and test sources together. For the Priority experiments, 25% of the background traffic was in Level 1 (high priority) and 75% was in Level 2. All traffic in our experiments represents real-time traffic.

In reality, best-effort traffic will continue to make up an important part of Internet traffic. Our analysis does not suffer by omitting best-effort traffic from the model, since we assume it would receive lower priority than real-time traffic, and therefore would not impact the delays seen by real-time traffic. However, the presence of best-effort traffic would impact the amount of real-time traffic in the network. If one assumes, for instance, that 20% of network traffic will be best-effort, then utilization levels higher than 80% in our experiments fall outside of expected operating conditions. Thus, an important, but unanswerable, question in analyzing our results is how much of the link bandwidth will be taken by best-effort traffic. If it is a large percentage, then one need only consider fairly low levels of real-time utilization, and there the comparison of two levels of priority versus one is quite different than at higher levels of utilization.

3.2 Results

We present our results in three stages. First, we present "raw" data of queueing delays and jitter induced by the network. This shows the service provided by the network, before any processing by the applications. Then we add the application performance and utility to our analysis in the context of non-adaptive applications. Finally, we present results of experiments using the adaptive algorithms described earlier. This incremental approach demonstrates the importance and impact of the specific characteristics of applications (i.e., their utility and adaptation algorithms) we consider.

[13]The *vic* program is available at http://www-nrg.ee.lbl.gov/vic/.

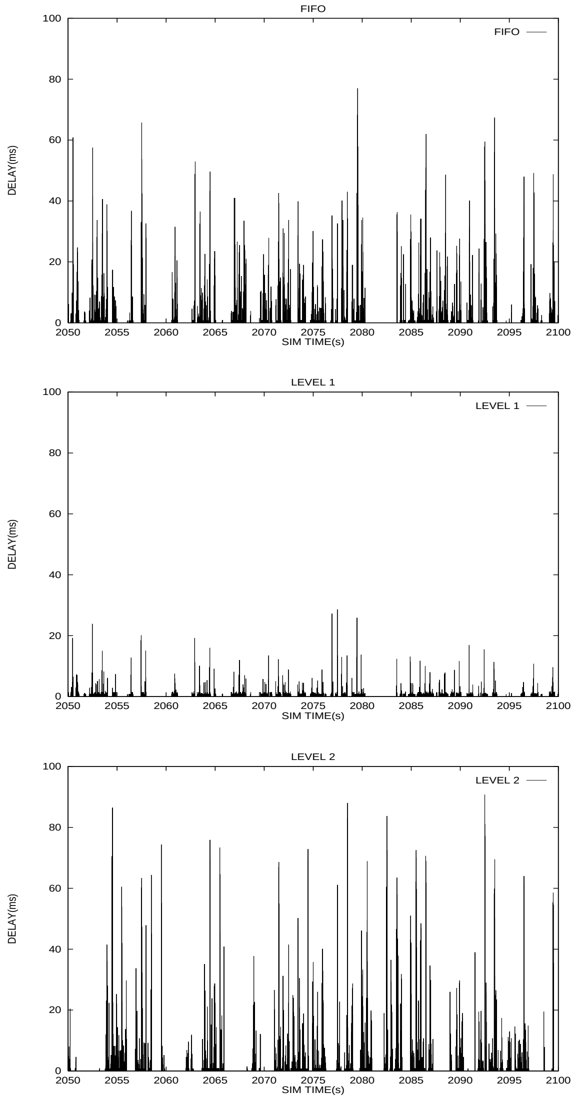

Figure 2: Delay trace at 55% utilization

3.2.1 Raw Network Performance

Figure 2 shows a plot of delay versus time over a 50 second simulation interval for a single test source in the FIFO case and for test sources in each level in the Priority case. Average utilization is 55% in both experiments. Histograms of delay (over a 500 second simulation interval) are shown in Figure 3. These graphs depict, as expected, that the service provided by Level 1 is better than that of Level 2 and of FIFO, and that FIFO was better than level 2 (although we were surprised by how small this latter difference was in the histograms). The unanswered question is whether or not these performance differences matter significantly to applications. Consider first the average delays: 0.71 ms for Level 1 and 6.73 ms for Level 2. While in absolute terms, this difference is significant, it is likely to be dwarfed by other sources of delay in the network, such as propagation time. Hence, if average delay matters, then one may conclude that multiple levels of service

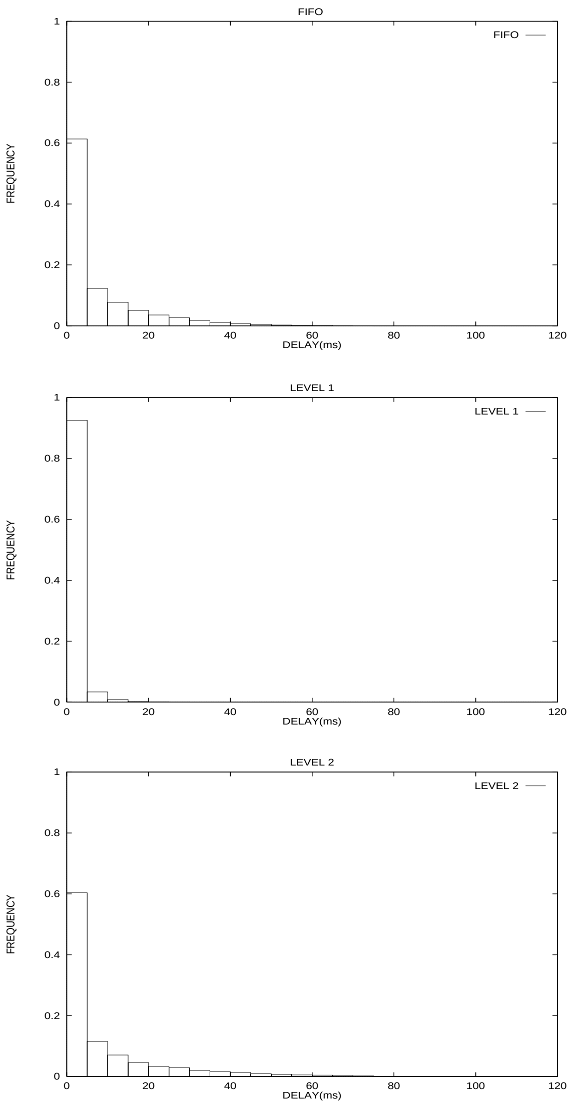

Figure 3: Delay histogram at 55% utilization.

does not provide significant benefits to applications. However, the tails of the delay distributions are dramatically different. For example, the maximum delay experienced for Level 1 and Level 2 are 30 and 100 ms, respectively. In contrast to the averages, the differences between these figures are likely to be significant to some applications (e.g., interactive ones). Hence, it is apparent that one cannot address the design issue we raise here without considering the effect of the network service on the applications that use it. Specifically, how do applications adapt to the service, and how do they ultimately perceive the service?

3.2.2 Rigid Application Performance

We first consider the relationship between network service and application performance in the context of rigid applications that do *not* adapt to current network conditions. Rigid applications remove network jitter by maintaining a constant playback point for all packets.

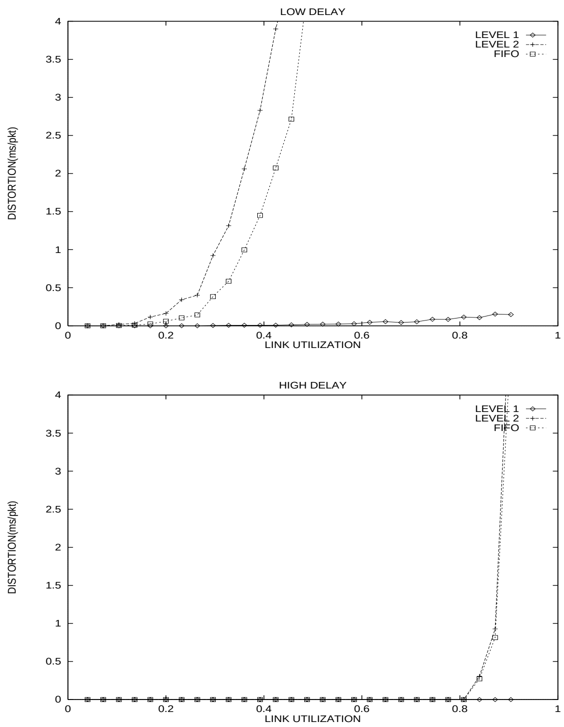

Figure 4: Distortion for rigid applications.

That is, all packets are buffered so that the sum of their network and playback delays are equal. Packets that arrive after their playback points must be discarded. This receiver behavior maintains perfect fidelity as long as packets arrive "in time", but degrades when packets arrive late. While such an application is impractical for the Internet (because applications have no way of knowing where to set the playback point when the first packet arrives in a way that will produce an acceptable level of distortion), we consider its behavior here to motivate the need to include adaptation in our analysis.

For rigid applications, the delay performance measure is merely the fixed delay experienced by all packets. To measure distortion, we assign a penalty of 120 ms (or three times the maximum penalty incurred by the adaptive algorithms) for each packet that arrives late and is dropped by the application.[14] The playback point for a rigid application is determined by the utility functions for delay. A delay sensitive application using the rigid playback algorithm sets its playback delay to 25 ms, half the delay threshold at which utility starts to degrade, while delay insensitive applications set the playback delay of the first packet to 500 ms.[15]

[14] Relating the distortion measure of rigid and adaptive applications is problematic, as it involves comparing the cost of late packets dropped by the application to the cost of adjusting the playback algorithm. Given our performance measures and utility functions, utility starts to degrade at .2% packet loss and utility is zero when packet loss reaches .8% for distortion sensitive rigid applications. For distortion insensitive applications, the corresponding thresholds are 2% and 8%.

[15] Choosing the playback point for rigid applications is also problematic. If the application has knowledge about the queueing delay of the first packet it receives, it could set the playback

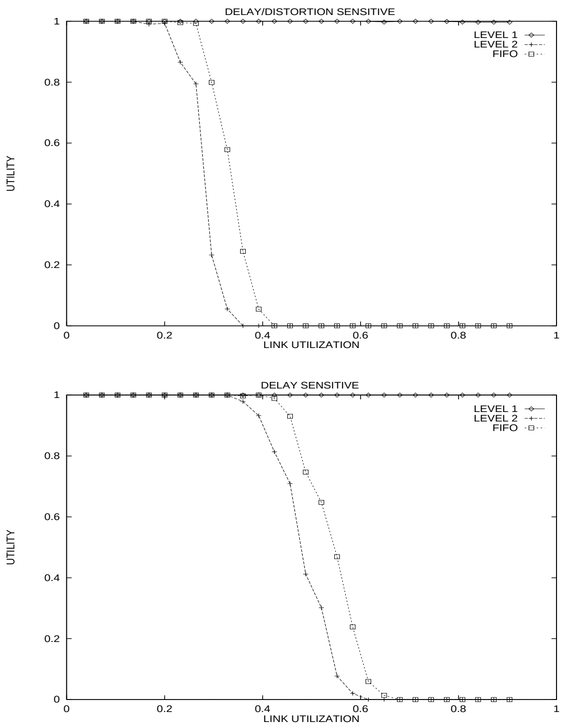

Figure 5: Rigid application utility for delay/distortion sensitive and delay sensitive applications.

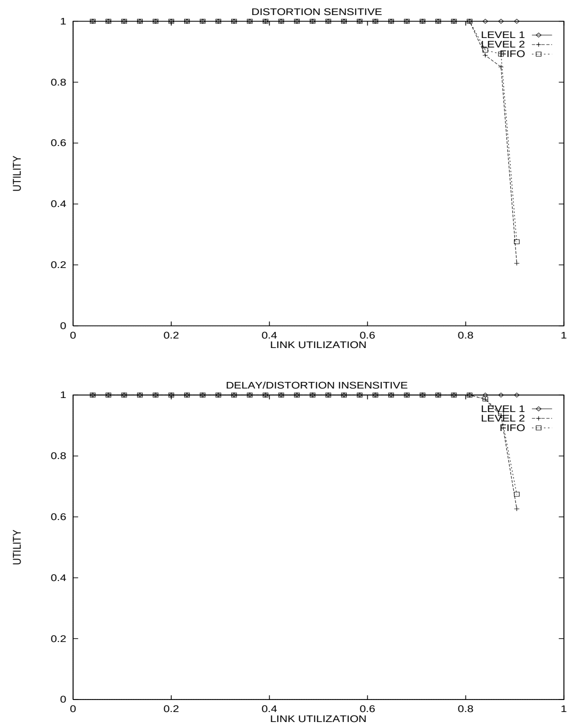

Figure 6: Rigid application utility for distortion sensitive and delay/distortion insensitive applications.

Figure 4 shows distortion as a function of offered load for rigid applications. For both delay sensitive and insensitive applications, data are shown for FIFO service (all traffic in a single service class) and for each of two levels in the Priority service case. With a low delay threshold (set for a delay sensitive application) there is no distortion up to about 20% utilization for the FIFO case. Beyond that, distortion starts to increase, deteriorating rapidly beyond 40% utilization. In the case of priority service, the Level 1 traffic experiences negligible distortion up to levels of utilization exceeding 80%.[16] The distortion of the Level 2 traffic is similar to the distortion of the FIFO service with the increases occurring at slightly lower levels of load. When the playback point of the rigid application is set to satisfy delay insensitive applications, no distortion is experienced (except at very high loads with Level 2 and FIFO service) as the playback point is large enough to enable almost all packets to arrive before their playback times.

These figures indicate how much distortion (resulting from discarded late packets) applications experience. However, they do not provide any indication about the effect that this distortion has on application performance. We employ the utility functions described in Section 2.3 to each of 4 classes of applications (characterized by their relative sensitivity or insensitivity to each of delay and distortion) for the FIFO case and for each level of service in the priority case. The results are shown in Figures 5 and 6. Applications that are sensitive to both delay and distortion achieve high utility up to about 25% load with FIFO service, then performance deteriorates rapidly. Applications that are only sensitive to delay and not to distortion experience this performance degradation at higher levels of load (45%). Applications that are sensitive to distortion only and applications that are not sensitive to either performance measure achieve high utility, except at the very highest levels of load.

In the Priority case, Level 1 service allows all applications to achieve high utility at all levels of load. Above load levels of 45%, Level 2 is only useful for applications that are insensitive to delay.

3.2.3 Adaptive Application Performance

We now consider the impact of the aggressive and conservative playback algorithms described in Section 2.2. To make comparisons between these algorithms as fair as possible, the minimum playback delay at the start of a talkspurt was set to 25 ms in the aggressive algorithm. This is consistent with the playback delay in the rigid algorithm. Within a talkspurt, when a packet arrived late and the playback point was adjusted, the minimum additional playback delay was set to 5 ms. The initial playback delay for the conservative algorithm was also 25 ms. We repeated our simulation

point optimally (i.e., to the delay value at which utility starts to degrade) and minimize packet loss. However, without this information, it can only guess. We used half the optimal value as the playback point, and set the minimal playback point of the adaptive algorithms to the same value to make the performance comparisons fair.

[16] Recall, the X axis is total offered load; hence in the Priority experiments, 80% load consists of 20% high priority traffic and 60% low priority traffic since we hold the ratio of high to low priority fixed at 1:3.

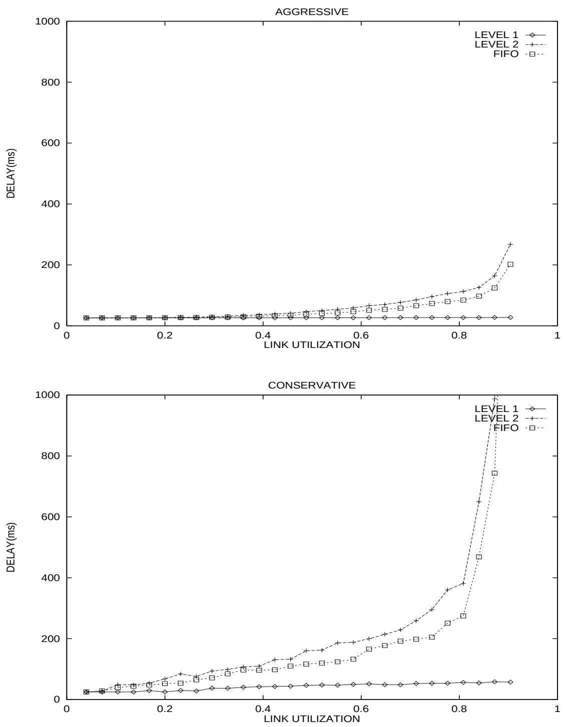

Figure 7: Delay for adaptive applications.

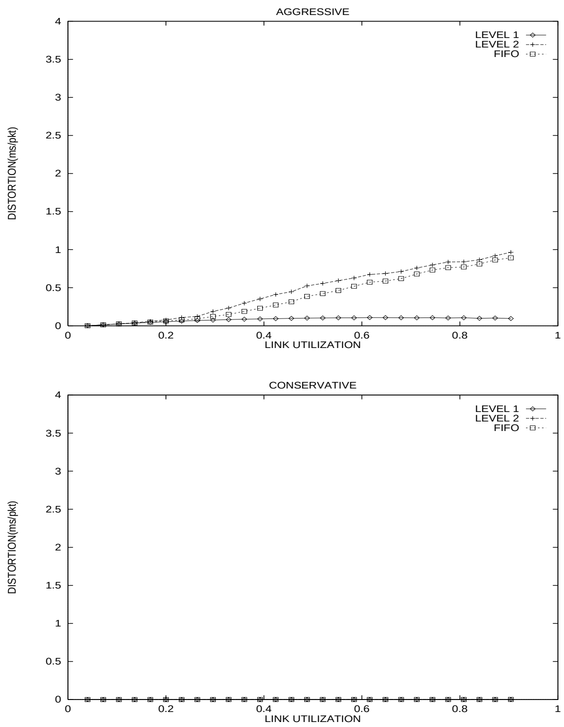

Figure 8: Distortion for adaptive applications.

experiments, computing delay and distortion for each algorithm in the FIFO test and for both levels of service in the Priority case. Delay measures as a function of load are shown in Figure 7 and distortion measures are shown in Figure 8.

These figures demonstrate that the two adaptation algorithms offer tradeoffs of delay for distortion. The aggressive receiver gives lower delays and higher distortion than the conservative algorithm at equivalent load levels. For example, with FIFO service, the aggressive receiver's delay is 51.2 ms while the conservative algorithm yields an average delay of 165.8 ms at 60% utilization (Figure 7). At the same load, the algorithms yield distortion values of .57 and .00071, respectively (Figure 8). However, these figures also show that while adaptation can reduce one quantity at the expense of the other, there is little adaptation can do to reduce both. So, as we shall see, applications that are sensitive to both delay and distortion are the most vulnerable to network service variations.

When two levels of service are available, the higher priority level always gives applications better service at high levels of load. However, the relative difference between Level 1 and Level 2 service depends on the adaptation algorithm. For instance, the difference between Level 1 and Level 2 delays is smaller with the aggressive algorithm than with the conservative algorithm. Conversely, the difference in distortion between Level 1 and Level 2 is small with the conservative algorithm and large with the aggressive algorithm. These differences affect the relative utility applications receive in different service levels. We next look at application utility as a function of the application's performance sensitivities.

Figures 9 and 10 show utility as a function of offered load for different types of applications. Each application uses the adaptation algorithm that is best suited for it.[17] With FIFO service, applications that are sensitive to both delay and distortion receive good service (i.e., high utility) only at low levels of load. Utility starts to decrease at utilization levels of about 40% of the link bandwidth as the adaptation algorithm is unable to meet both the delay and distortion requirements of the application, simultaneously. When high priority service is used, performance does not deteriorate. Applications that are sensitive to delay and not distortion maintain high utility up to 60% utilization with FIFO service, as the adaptation algorithm can optimize the performance measure about which the application cares the most. At higher loads ($> 60\%$), high priority service does improve the performance of these applications. The applications that are only sensitive to distortion use the conservative adaptation algorithm to minimize distortion and achieve high utility at all but the highest levels of offered load with FIFO service. Hence, these applications derive no benefit from priority service, except in extreme conditions. Finally, applications that are insensitive to both delay and distortion also do not benefit from priority service (except at very high loads), given that their performance requirements are such that they are satisfied at most load levels with FIFO service.

[17] That is, the two kinds of applications that are sensitive to delay use the aggressive algorithm and the other two use the conservative algorithm.

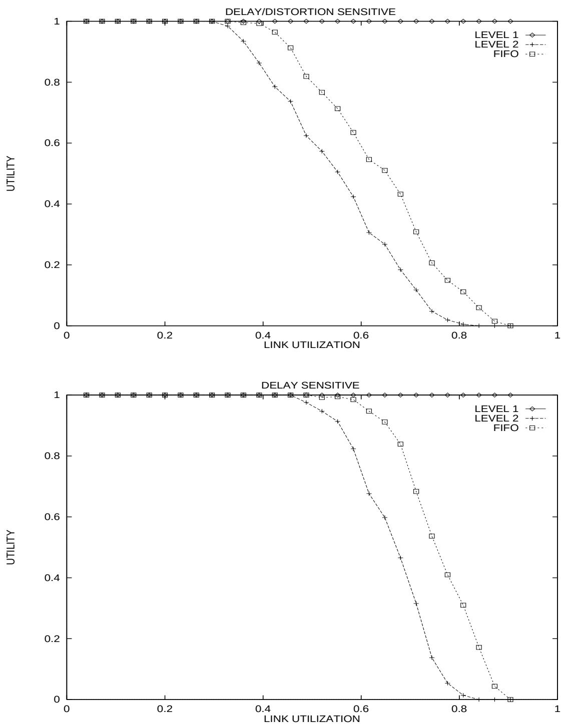

Figure 9: Adaptive application utility for delay/distortion sensitive and delay sensitive applications.

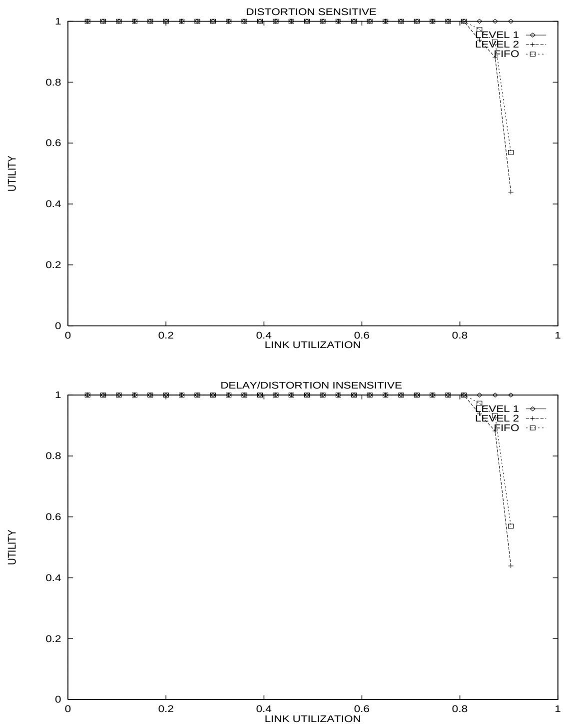

Figure 10: Adaptive application utility for distortion sensitive and delay/distortion insensitive applications.

3.2.4 One or Two Levels

The previous results showed, not surprisingly, that some applications achieve higher utility with high priority service than without it, and the magnitude of the difference depends on the characteristics of the application (its performance sensitivities) and on the level of the ambient traffic. However, by itself, this does not provide an answer to the question of the number of service levels that should be offered in the network. After all, better service always helps some applications, but at the same time it gives worse service to other applications. Hence, the answer depends on how much better two levels of service makes the overall network service. There is not a single best way to answer this question. We consider two alternatives.

The first, and perhaps most obvious approach is to consider the impact of multiple levels of service on total network utility. This method is fraught with problems. For instance, it depends on the mix of different kinds of applications in the network, the absolute value of utility achievable by each application, and on an incentive mechanism that impacts the mapping of application to service level. Nonetheless we proceed forward using the results of our previous experiment. We assume that there are equal amounts of each kind of application, that only the applications that are sensitive to both delay and distortion use the higher priority service, and that all applications have the same maximum utility.[18] Figure 11 shows the average utility per application as a function of offered load. Below 40% utilization, two levels of service offers no increase in utility. Above 40% load, service priority does offer modest advantages, with the benefit increasing with load. In contrast, Figure 12 shows total utility for rigid applications. Relative to the adaptive applications, these results present a stronger, but possibly misleading, case for multiple levels of service.

In addition to looking at the impact on total network utility, it is important to ask what effect service levels has on particular classes of applications. That is, independent of total utility, it may be important for a network to make sure that it serves all classes of applications adequately. In this case, the relevant question to ask is, for a given service discipline, at what level of offered load is an application no longer well-served by the network. We can ask this question in the context of our earlier results (see Figure 9). When only a single FIFO service class is offered, applications that are sensitive to delay and distortion start to suffer a loss in utility at utilization levels of 40%. When these applications use priority service, the network provides them with useful service at very high load. Assuming the other application types use the lower priority service, it is important to consider the levels of utilization at which the network no longer satisfies these applications. As is evident from the previous graphs, the delay insensitive applications do not suffer by using the lower priority service. The delay sensitive and distortion in-

[18] The actual mapping of applications to service levels depends on the incentives of using each class, which are not included in our analysis. However, given our previous results, we assume that if priority service is available, it will be used only by applications sensitive to both delay and distortion, since these are the applications that derive the most benefit from it.

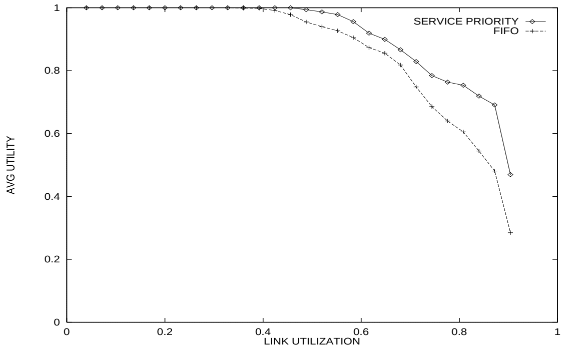

Figure 11: Average utility for adaptive applications.

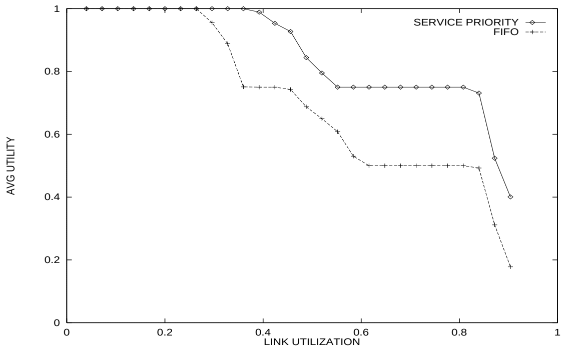

Figure 12: Average utility for rigid applications.

sensitive applications do suffer a bit when using the lower priority service: the level of utilization at which they no longer achieve utility of 1 decreases from over 60% to 50%. Nonetheless, when looking at all application classes, multiple levels of service increases the load levels at which the network can satisfy all of them. This suggests that multiple service levels may be worth deploying according to this criterion.

4 Discussion

Previous studies of application adaptivity have compared algorithms, focusing on such performance measures as the percentage of late packets for a given packet stream and adaptation algorithm.[8, 10] We believe ours is the first study to incorporate this adaptive behavior into consideration of a network design question. Our study provides no definitive answer to the question of whether multiple service priorities should be provided for real-time traffic in the Internet. Our results showed that with a single level of service, performance of some applications starts to degrade at utilization levels below 50%. These applications would benefit from two levels of service, and other applications are able to tolerate a lower priority service, yielding higher total network utility. However, the ultimate answer to our question depends on several characteristics of the future Internet, about which we are uncertain. First, how bursty will aggregate traffic be? We have presented results for a single kind of background traffic produced by multiplexing a moderate number of low frame rate video sources. Results from additional experiments, not shown here due to space limitations, showed that the burstiness of background traffic can be viewed as a knob that can be varied. With smoother background traffic and FIFO service, performance does not degrade until higher levels of utilization are reached. Therefore, the relative benefits of priority service are smaller and only occur at higher levels of utilization, making a weaker case for multiple levels of service. The converse is true with burstier background traffic. One should not take too seriously the absolute values of the utilization levels presented here; by making the traffic even burstier, one could make the levels of utilization at which performance degrades, arbitrarily small. We do not yet know if future Internet traffic will be smooth enough everywhere to obviate the need for multiple levels of service, or will the levels of burstiness be such that multiple service priorities are unambiguously desirable. In any event, additional simulations of either less bursty or more bursty traffic will not resolve this.

Second, what will the ratio of best-effort to real-time traffic be in the network? For example, if future network traffic consists of 90% best-effort traffic then the relevant utilization levels for real-time traffic in our simulations would only be 10%. At this level of load, for all but the burstiest traffic imaginable, real-time traffic always receives low delay and distortion, even with a single level of service. Delay and distortion would be absorbed by best-effort applications, which are well-suited to handle the performance degradation. Just as a few years ago one would not have predicted the tidal wave of web traffic, we cannot, at this point, predict the extent to which the Internet will be used for real-time applications.

Finally, what are the nature of utility functions of real-time applications? While we believe we have captured the essential characteristics of these functions, actual thresholds and functions may be significantly different. Ultimately, our simulation model can provide an answer to the larger question given an adequate set of parameters, but the model itself cannot resolve these questions. Hence, a more definitive answer requires a much better understanding of network applications, traffic mix and utility functions.

While we do not provide an unambiguous answer to the initial question we posed, our results do yield other key observations. First, application adaptivity is not a panacea. Our simulations showed that adaptive algorithms can very successfully remove distortion at the receiver, or they can reduce delay. However, achieving both low delay and low distortion is difficult under moderate load with bursty traffic. Hence, if there are applications that are sensitive to both performance measures, then under certain traffic conditions, adaptivity may not be enough. In this case, service discrimination inside the network is needed to provide these applications an acceptable level of performance.

Finally, we obtained different results for rigid and adaptive applications, further emphasizing the importance of including realistic application behavior in the analysis. While we demonstrate this point in the context of one specific network design question, we believe it has widespread applicability. For example, when considering other questions, such as whether or not the network should provide multiple levels of dropping priority, appropriate models of applications are needed. Conversely, if nothing else, this study has shown us that previous research (such as [14]) that models applications as rigid and does not take application adaptivity into account can lead to incorrect or misleading results. In the Internet, design analyses must incorporate the adaptive nature of applications.

References

[1] Jean-Chrysostome Bolot. End-to-end packet delay and loss behavior in the Internet. In *Proceedings of ACM Sigcomm*, pages 289–298, San Francisco, California, September 1993. ACM. also in *Computer Communication Review* 23 (4), Oct. 1992.

[2] R. Braden, D. Clark, and S. Shenker. Integrated services in the Internet architecture: an overview. RFC 1633, Internet Engineering Task Force, June 1994.

[3] R. Braden, Ed., L. Zhang, S. Berson, S. Herzog, and S. Jamin. Resource ReSerVation protocol (RSVP) – version 1 functional specification. Technical Report RFC 2205, Internet Engineering Task Force, September 1997.

[4] David D. Clark, Scott Shenker, and Lixia Zhang. Supporting real-time applications in an integrated services packet network: Architecture and mechanism. In *Proceedings of ACM Sigcomm*, pages 14–26, August 1992.

[5] Domenico Ferrari, Anindo Banerjea, and Hui Zhang. Network support for multimedia: A discussion of the Tenet approach. *Computer Networks and ISDN Systems*, 10:1267–1280, July 1994.

[6] Sally Floyd. Comments on measurement-based admissions control for controlled-load services. submitted to CCR, July 1996.

[7] Sugih Jamin, Peter B. Danzig, Scott J. Shenker, and Lixia Zhang. A measurement-based admission control algorithm for integrated services packet networks. *IEEE/ACM Transactions on Networking*, 5(1):56–70, February 1997.

[8] Sue B. Moon, Jim Kurose, and Don Towsley. Packet audio playout delay adjustment algorithms: performance bounds and algorithms. Research report, Department of Computer Science, University of Massachusetts at Amherst, Amherst, Massachusetts, August 1995.

[9] Abhay K. Parekh and Robert G. Gallager. A generalized processor sharing approach to flow control in integrated services networks: The single-node case. *IEEE/ACM Transactions on Networking*, 1(3):344–357, June 1993.

[10] Ramachandran Ramjee, Jim Kurose, Don Towsley, and Henning Schulzrinne. Adaptive playout mechanisms for packetized audio applications in wide-area networks. In *IEEE Infocomm*, pages 680–688. IEEE, 1994.

[11] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson. RTP: A transport protocol for real-time applications. RFC 1889, SRI Network Information Center, January 1996.

[12] S. Shenker, C. Partridge, and R. Guerin. Specification of guaranteed quality of service. RFC 2212, Internet Engineering Task Force, September 1997.

[13] S. Shenker and J. Wroclawski. Network element service specification template. Technical Report RFC 2216, Internet Engineering Task Force, September 1997.

[14] Scott Shenker. Fundamental design issues for the future internet. *IEEE Journal on Selected Areas in Communications*, 13(7), September 1995.

[15] C. Topolcic. Experimental internet stream protocol, version 2 (ST-II). RFC 1190, SRI Network Information Center, October 1990.

[16] J. Wroclawski. Specification of the controlled-load network element service. RFC 2211, Internet Engineering Task Force, September 1997.

[17] J. Wroclawski. The use of RSVP with IETF integrated services. Technical Report RFC 2210, Internet Engineering Task Force, September 1997.

[18] Lixia Zhang, Steve Deering, Deborah Estrin, Scott Shenker, and Daniel Zappala. RSVP: A new resource reservation protocol. *IEEE Network Magazine*, 7(5):8–18, September 1993.