Fundamental Design Issues for the Future Internet

Scott Shenker[1]
Xerox Palo Alto Research Center
3333 Coyote Hill Road
Palo Alto, CA 94304-1314
415-812-4840
shenker@parc.xerox.com

Abstract

The Internet has been a startling and dramatic success. However, multimedia applications, with their novel traffic characteristics and service requirements, pose an interesting challenge to the technical foundations of the Internet. In this paper we address some of the fundamental architectural design issues facing the future Internet. In particular, we discuss whether the Internet should adopt a new service model, how this service model should be invoked, and whether this service model should include admission control. These architectural issues are discussed in a nonrigorous manner, through the use of a utility function formulation and some simple models. While we do advocate some design choices over others, the main purpose here is to provide a framework for discussing the various architectural alternatives.

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

1 Introduction

There are few technological success stories as dramatic as that of the Internet. As recently as 1985 the Internet had only about 50 sites and 1000 hosts, but now the numbers are well over 40,000 and 3,000,000 respectively and they continue to grow at astonishing rates.[2] While originally built to link a small community of researchers, the Internet has, much to everyone's surprise, grown into a social institution of substantial import; for example, one national newsmagazine has a regular column on the Internet (Newsweek), another featured it on its cover (Time), and many advertisements now include an electronic mail address or web page as a contact point. Even though the Internet is still extremely small compared to the telephone and the cable TV networks in terms of the number of users and the quantity of capital invested, it has clearly joined them as a significant aspect of our telecommunications infrastructure. Thus, the design choices we make for the Internet, far from being the exclusive concern of a small technical community, will have far-reaching implications for the general public. In particular, these design decisions will play an important role, along with many economic and social factors, in determining the nature of our future telecommunications infrastructure.

This paper discusses a few of the fundamental design decisions that the Internet community must address in the near future. After briefly reviewing the current Internet architecture and two approaches of modifying it (in Section 2), we present a criterion for evaluating network designs (in Section 3). We then discuss whether the Internet should adopt a new service model (in Section 4), how this service model should be invoked (in Section 5), and whether this service model should include admission control (in Section 6).

The research literature is replete with proposals for new network designs and analyses of existing ones; we do not attempt to add to this body of knowledge here, nor to review it (see [2, 3, 9, 10, 11, 12, 15, 16, 17, 18, 19, 20, 21, 22, 23, 25, 26, 27, 30, 31, 32, 33, 34, 36, 40, 43, 44] and the references therein for a sampling of the literature). Instead, our focus is on identifying the reasoning and assumptions behind various design approaches in the hope of illuminating some of the fundamental architectural choices facing the Internet. To this end, the treatment here is rather general and illustrative, rather than specific and analytical. While we do argue for some design choices over others, our main purpose here is to provide a conceptual framework for evaluating these design choices rather than advocate a particular design.

A natural question to ask is, given its recent success, why should we contemplate any changes to the Internet architecture at all? To understand this, we must review some aspects of the current Internet and its application base.

2 The current Internet

The Internet offers a single class of "best-effort" service; that is, there is no admission control and the network offers no assurance about when, or even if, packets will be delivered. Current usage of the Internet is dominated by traditional data uses such as remote terminal (e.g., Telnet), file transfer

[2]The number of hosts increased by 81% this year alone. These numbers were taken from [38] and [24].

(e.g., FTP), name service (e.g., DNS), and electronic mail (e.g., SMTP).[3] These applications are rather elastic in nature, in that they tolerate packet delays and packet losses rather gracefully, and so they are rather well served by the current Internet's best effort service.[4] Moreover, because of this elasticity, they can decrease their transmission rate in the presence of congestion;[5] such congestion control algorithms enable the Internet to avoid congestion collapse.

With the emerging widespread use of multimedia applications, computers are now processing voice and video in addition to their more traditional tasks. Therefore, the networks that connect these computers, such as the Internet, must be prepared to cope with the traffic emanating from such applications. "Real-time" applications like video and voice have very different characteristics and requirements[6] than data applications, and thus their emergence is likely to significantly alter the nature of the Internet's traffic load. In particular, as traditionally implemented these real-time applications are typically less elastic – less tolerant of delay variations – than data applications. This lack of elasticity causes two problems.

First, these traditional implementations[7] of real-time applications do not perform adequately when running over the current Internet because the variations in delay are too extreme and there are too many dropped packets. Second, these applications typically do not back off in the presence of congestion; when these real-time applications are contending for bandwidth with traditional data applications, the data applications end up receiving very little bandwidth. Thus, when deployed in the current Internet, traditional real-time applications not only do not always perform adequately but they also often interfere with the data applications.

One can address these problems without changing the basic Internet architecture by improving different aspects of the implementations. The unfairness that results when congestion-avoidant data applications compete with congestion-ignorant real-time applications can be resolved by using the Fair Queueing packet scheduling algorithm (or something roughly equivalent) in routers. These routers would then ensure that every user had access to their "fair share" of bandwidth, and so data applications would be protected from the real-time ones (see [6] for a discussion of such scheduling algorithms). Such a modification does not require any change to the Internet architecture.[8] While this approach solves the second problem mentioned above, it does not solve the first; the service delivered to an individual application can still have substantial delay variance, seriously degrading

[3] We should point out that applications like Mosaic and WWW have made a great impact on Internet usage, but their network usage is almost exclusively file transfer.

[4] We will address the issue of why best effort is the appropriate service for such applications in much greater detail in Section 6.

[5] Of course, this congestion avoidance is typically not done in the application itself but rather in the transport protocol TCP.

[6] We discuss these characteristics at much greater length in [36].

[7] We won't describe implementation issues at all in this paper, except to say that traditional implementations of real-time applications involve either fixed play-back points (see [3, 36]) or, even worse, immediate play-back of incoming data. Both of these implementation styles lead to significant signal degradation under high delay variance.

[8] This approach does not require an architectural modification although it would be greatly enhanced by the inclusion of some sort of flow identifiers and authentication mechanism, because otherwise it is hard to ensure that the "fair share" of bandwidth is allocated to the right granularity (e.g., source, or source-destination, or individual flow, etc.). These modifications to the Internet architecture, especially authentication mechanisms, are probably needed for other reasons as well and we will not discuss them further in this paper.

the performance of these traditional implementations of real-time applications.

One can address this problem by modifying the application implementations rather than the network implementation. In recent years there have been tremendous advances, much born out of necessity, in making such applications much more adaptive to variations in packet delays;[9] such delay adaptive techniques have been highly successful in such Internet applications as *nv* and *vat*. These delay adaptive techniques were first introduced many years ago, see [7, 41], but only recently have these adaptation techniques been seen as a general and enduring solution.

This "change the implementations but not the architecture" approach has several important advantages. No changes are required to any network interfaces, so the change can be incrementally deployed at both the end hosts and the routers. Also, the network mechanisms (Fair Queueing and its relatives) and application mechanisms (delay adaptation) are relatively well understood.[10] However, in this approach the network would deliver the same class of service to all users, with no assurances as to the quality of that service. While the network would protect users from each other, it is up to applications to adjust to the inevitable variations in packet delay and available bandwidth. There are likely to be limitations to this adaptability. Moreover, because there is no admission control the network must be provisioned so that the fair bandwidth shares are not, except in very rare cases, unreasonably small.

As we will explore in this paper, it is not clear that such an approach is desirable; there are other approaches to supporting real-time applications that modify the basic Internet architecture. These modifications usually involve extending the Internet's service model – the set of delivery services – from the single class of best-effort service to include a wider variety of service classes. In addition, one can augment this service model with admission control, which is the ability to turn some flows[11] away when the network is overloaded. In this paper we ask whether or not these architectural modifications are appropriate. Surprisingly, this question has received rather little explicit attention in the literature. The approach which involves only implementation enhancements and not architectural modification, while advocated by several researchers, has not been adequately described in the literature. The approaches which entail major architectural changes have been fully described (see [2, 3, 9, 10, 11, 12, 15, 16, 17, 18, 19, 20, 21, 22, 23, 25, 26, 27, 30, 31, 32, 33, 34, 36, 40, 43, 44] and references therein for examples). Most of these papers implicitly assume that such modifications are necessary and hence give little justification for the basic approach (this author also pleads guilty to this crime; see [3]); instead the focus in these papers is on the details of the design and comparison with other similar architectures. The purpose of this paper is to articulate some of the basic issues in what has been an underarticulated disagreement about basic architectural assumptions. We hope to provide a framework for considering the various architectural tradeoffs and identify some of the critical assumptions that lead to one choice or another. We do this by presenting a rather abstract formulation and a few rather simple models; this approach is not intended to model reality, but to be the simplest formulation which illustrates the real issues at stake. Central to this endeavor are questions about the nature of future computer applications, the

[9]By making the play-back point be adjustable, the application can adapt to the current delays and can tolerate fairly large variations in delay without significant performance degradation.

[10]Although we have much less experience with rate adaptation, it is also needed in this approach to limit the application's sending rate to its fair share. We will discuss rate adaptation again in Section 6.

[11]We will use the term flow to refer to the traffic stream representing a particular user or application.

cost of additional network mechanism, the future supply of bandwidth and other matters which are not amenable to precise analysis. Consequently, our discussion here is nonrigorous and intuitive; our goal is to articulate these issues, not provide analytical resolution of them. Of course, all of this analysis must start by asking what is the goal of network design.

3 What is the goal of network design?

By what criteria do we evaluate a particular network architecture? The Internet was designed to meet the needs of users, and so any evaluative criteria must reduce, in essence, to the following question: how happy does this architecture make the users? Network performance must not be measured in terms of network-centric quantities like utilization, dropped packets, or power, but rather should be evaluated solely in terms of the degree to which the network satisfies the service requirements of each user's applications.[12] For instance, if a particular application cares more about throughput than delay, or vice-versa, the network service to that application should be evaluated accordingly.

We can formalize such a notion of network performance as follows. Let the vector s_i describe the service delivered to the i'th application or user; s_i contains all relevant measures (delay, throughput, packet drops, etc.) of the delivered service. We then let the utility functions U_i map the service delivered into the performance of the application; increasing U_i reflects increasing application performance. The utility function describes how the performance of an application depends on the delivered service.[13] Later, in Section 6, we will discuss the shapes of these utility functions for some common application classes but for now we introduce them merely for definitional purposes.

The goal of network design is to maximize the performance of the resident applications. With this formalism, this goal can be restated as being, quite simply, to maximize the sum of the utilities. We will call this quantity V, the *efficacy* or total utility of an architecture: $V \equiv \sum_i U_i(s_i)$. Much of our subsequent analysis bears on how various design choices affect V.

4 Why do we need to extend the service model?

One can always increase the efficacy of an architecture by supplying more bandwidth; faster speeds mean lower delays and therefore higher utility values. However, one can also increase V by keeping the bandwidth fixed and delivering a wider variety of services than just a single class of best-effort service. Such an extension in the service model, which is the set of services offered by the network, is a fundamental aspect of many of the recently proposed network architectures. Such an extension would allow different applications to get different qualities of service; the general intuition is that the closer the services are aligned with application needs, the higher the efficacy. For instance, if

[12] Here I am equating the happiness of users with the performance of their applications and will use the terms application and user somewhat interchangeably in this context.

[13] We will assume we are considering *transferable* utility; that is, we can make comparisons between the performance of applications so quantities like the sums of utilities make sense. The utility can then be considered a measure of how much a user would be willing to pay for the service.

different applications have widely different sensitivities to delay, then offering two priority classes will likely increase the efficacy of the network (when compared to a single class of service).

We can illustrate this by the following simple example, where we compare providing two priority classes with a single class of FIFO service. 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 $U_1 = 4 - 2d_1$ and $U_2 = 4 - d_2$ where d_i represents the average queueing delay delivered to client i.[14] 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 $V^{FIFO} = 2$. 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 $V^{priority} = 8/3$. 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.

We should note that this example is misleading in that a slightly overprovisioned FIFO network ($\mu = 17/16$) has the same value of V as the priority network with $\mu = 1$. If we consider more realistic examples, where the delay preferences are more varied (e.g., depending on higher moments of the delay or having a "knee" in the utility function at some value of delay), a much greater degree of overprovisioning is needed to make a FIFO network match the efficacy of a priority network.

All other things being equal, one should definitely offer a more varied set of services than just the single class of best-effort service. However, there is a tradeoff between the cost of adding bandwidth and the cost of adding the extra mechanism needed to extend the service model. Both of these costs are impossible to gauge precisely. The mechanistic aspects are costly not just in control overhead and in the extra complexity required in network components, but also in the disruption caused when changing such a basic aspect of the architecture (later in this section we will explore some of the implications of these changes). The cost of the bandwidth will depend greatly on the nature of the competitive and regulatory environment, as well as future technological developments, none of which we can accurately foretell. Moreover, the amount of bandwidth needed to offset the benefits of extending the service model depends in detail on the utility functions of applications and the service model being offered. Evaluating this tradeoff requires making judgements about future developments about which little is known and opinions vary widely.

Nonetheless, despite this uncertainty, at the core of this bandwidth vs. mechanism tradeoff is the central fact that the timescales of the service requirements of real-time applications are much smaller, by several orders of magnitude, than those of, say, FAX or electronic mail. One cannot operate a network at reasonable utilizations while delivering to all traffic a service suitable for real-time applications; yet the extreme elasticity of FAX and email would be able to utilize the significant amount of leftover bandwidth if the service model could just keep it out of the way of the real-time traffic. For this reason alone it seems plausible, if not probable, that the payoff in terms of the bandwidth saved from offering these multiple classes of service will more than outweigh the costs of the extra mechanism.

[14] 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 $d_2 = \frac{\mu}{(\mu - r_1)(\mu - r_1 - r_2)}$.

One might ask if we should just use separate networks for the various applications, each with its own single service class; this is similar to what we do now with separate telephony, video, and data network infrastructures. We can use our simple example to examine this possibility. If we double the linespeed, $\mu = 2$, and double the number of applications, the performance numbers become: $V^{priority} = 32/3$ and $V^{FIFO} = 10$. Now consider two networks, with separate bandwidths $\mu_1 + \mu_2 = 2$, each carrying two applications. If we divide the clients so that each network carries one application of each type, and the bandwidths are split evenly (which is optimal here), then we revert back to our original case and the most efficient thing is to use priority on each link and achieve a total efficacy of 16/3. If we partition the clients so that one network carries two delay-sensitive applications and the other carries the two less sensitive applications, then the optimal arrangement is to use FIFO on each network and to split the bandwidth[15] as $\mu_1 = 5/2 - \sqrt{2} = 1.086...$ and $\mu_2 = \sqrt{2} - 1/2 = .914...$, which yields an efficacy of $10 - 4\sqrt{2} = 4.343....$ Thus, combining the networks into a single infrastructure yields a much higher value for V than using separate networks; in fact, in this simple example the value of V is doubled by combining the networks. When considering separate networks, there is greater efficacy when the application types are mixed. Thus, the *least* efficient network design is to build separate networks each with a different application class.

Our analysis of this simple example also reveals two other important points about extending the service model. First, not every client gains directly from the increase in efficacy; that is, if we compare two service allocations $\vec{s}^1$ and $\vec{s}^2$, $V(\vec{s}^1) > V(\vec{s}^2)$ does not imply that $U_i(s_i^1) > U_i(s_i^2)$ for all i. For instance, in the simple example with just two clients, $U_2^{FIFO} = 2$ but $U_2^{priority} = 4/3$ even though $V^{priority} > V^{FIFO}$. Efficacy in heterogeneous networks is gained by shifting resources from applications that are not extremely performance-sensitive to those that are; the performance-sensitive clients gain from offering more sophisticated service models, but the less-performance sensitive clients lose.

Second, in order to achieve the additional efficacy with an extended service model, the mapping between service classes and applications must reflect the application requirements. In our simple example, the increased efficiency of priority service can only be realized if the network can recognize which client is more delay-sensitive and assign it the appropriate service class. Assigning flow 1 to the lower priority class and flow 2 to the higher priority class would result in a lower value for V; in fact, it yields $V = 4/3$ (which is 1/2 of the optimal value). We will return to these two points in Section 5.

5 Who chooses the service for a flow?

At[16] this point, we have argued that the network should offer a service model that includes more than just the single class of best-effort service. This service model could be as simple as two priority levels, or as complicated as the multiple delay-bounded service classes described in [3]. However, we

[15] Using the formulae for delay, $V = \{8 - \frac{2}{\mu_1 - 0.5}\} + \{8 - \frac{2}{\mu_2 - 0.5}\}$ and the fact that $\mu_1 + \mu_2 = 2$, we can solve for the optimal value of μ_1 and μ_2.

[16] While I have freely stolen the insights of Dave Clark and Lixia Zhang throughout this paper, in this section the larceny is especially egregious.

have yet to address one fundamental question which applies to all of these possible service models: how does the architecture decide which service to give a particular flow? There are really two possible answers to this question: the flow can pick the service, or the network can pick the service. We now contrast these two options.

5.1 Two options

If the network chooses the service class, then we say that the service is *implicitly* supplied; the application sends its packets without saying anything about its service requirements and the network then classifies these packets into some service class and handles them accordingly. For instance, the network might divide all traffic into the categories of asynchronous bulk, interactive bulk, interactive burst, and real-time by inspecting the port number or identifying a packet transmission "signature"[17], and then deliver to each category an appropriate service.

This approach has some important advantages. Most notably, this approach does not require any change to the service interface. Currently, applications can just send their packets without any negotiation with the network, and this will continue to be true with an extended service model if the service is supplied implicitly; applications would not need to specify their desired service to the network, and the network would, in turn, not need to describe the service-to-be-delivered to the application. Also, since there is no explicit commitment to a given service level, the mapping from application to service class, and the nature of the service delivered to each service class, need not be uniform across all routers nor stable over time. Since applications would not have to change, and the service given by the routers would not have to be standardized, such an architecture could be deployed immediately and then modified incrementally.

Weighing against these powerful advantages are some practical disadvantages. The implicit approach entails a fixed set of application classes; at any given time a router only knows about a certain set of applications and cannot properly service those applications about which it does not know. Moreover, this approach cannot accommodate individual or situational variations within a single application. For instance, if one uses the same audio or video application for both interactive (which needs low delay and low jitter) and lecture (which can tolerate large delays and jitter) settings, then there is no way for the network to distinguish between these two cases; this leads to inefficiencies because the network must give service appropriate for the interactive mode and thus the application gets better service than it needs when it is being used in lecture mode.[18]

These two practical problems are symptoms of a fairly basic architectural flaw in the implicit approach. The implicit approach can only work if the network knows something about the service requirements of each application. New applications cannot get the service they require if the network does not know what their service requirements are. Embedding application information into the network layer information violates one of the central design tenets of the Internet. The internetwork layer – the IP layer – was designed to provide a clean interface between networks

[17]The actual method by which the categorization is done is not important here, we care only that the categorization is under the control of the network and not the application.

[18]This example is a situational variation. An individual variation would be when one user is much more sensitive to delay or jitter than another user and would, if given the option, prefer a different service class. If the service classes are rather broad then this is unlikely to be much of a problem.

and applications. The fact that any application capable of running on top of IP could run on any network that supported IP encouraged diversity both above and below the IP layer. A wide variety of networking (or, more correctly, subnetworking technologies) have emerged (e.g., Ethernet, Token Ring, ATM, etc.) and a wide and ever-increasing variety of applications have flourished.[19] This clean separation is markedly different from the telephony and cable infrastructures, which are more focused on a single application and a uniform network technology. Experience suggests that violating the clean separation embodied in the IP layer by embedding application information in the network would inhibit the creation of new applications. Thus, the implicit approach, while certainly attractive in the short-term, has serious long-term deficiencies.

The alternative is to have applications *explicitly* request the service they desire. That is, the network offers a set of service classes and the applications indicate to the network which service class they want. This approach has the advantage that it maintains the clean architectural interface between applications and networks, so any future application can be serviced in the desired service class. However, this approach does have some unfortunate disadvantages. The first disadvantage is the incentive issues that are raised, and the second is the lack of flexibility in the service model. We address these two problems separately.

5.2 Incentives

Recall that in Section 4 we made two observations. First, we observed that the optimal efficacy – the maximal V – was only achieved when low priority service was given to the flow with less stringent delay requirements. Second, we observed that not all applications benefit directly from this increase in V; those asking for lower priority service received worse service. In the explicit approach, this mapping between between the application and service class is under control of the applications themselves and therefore ultimately under control of the user. We can achieve optimal efficacy only if some applications ask for lower quality service. What will motivate users to do this; why won't users always ask for the highest quality service no matter what their application requirements?[20] Certainly when the Internet had a small user population with a strong sense of tradition and comraderie, informal social conventions would have been sufficient to induce users to behave properly. However, in the public Internet of the future, as with any heavily utilized public facility, informal social conventions will not be sufficient to discourage selfish behavior. Thus, the network must provide some other system of incentives to encourage users to request the proper service classes for their applications. Pricing of network services is one approach. Charging more for the higher quality service will ensure that only the extremely performance-sensitive applications

[19] Note that there are two aspects of the separation provided by the internetworking layer. Separating the underlying network technology from the internetworking layer allows networks to interwork in a general fashion; separating the internetworking layer from applications above allows the network to support a more general set of applications. One can imagine designs which adopted one of these separation principles without the other, leading to either interoperable but application-specific networks, or incompatible networks which support a variety of applications; neither of these are attractive design choices.

[20] Our discussion here assumes that the service request made by the application is under user control. One could imagine a scenario where the service requests are embedded into the application itself (much as the port numbers used by FTP and Telnet can be used as quality of service signals) in a way that makes it rather difficult for users to manipulate them. Such an approach would certainly limit the incentive issues, but would then make it impossible for users to make adjustments to the video quality or sound quality (by tuning the network service request), and so this approach, while attractive in the short term, is unlikely to be adopted as an architectural principle.

will request that service. As discussed in [4, 35], pricing can be used to spread the benefits of increasing V to all applications; for some applications, the reduction in the quality of service is compensated by the reduction in price, and for others the increase in price is compensated by the increase in the quality of service.

Currently, most users of the Internet (those with direct connections to a regional provider) do not have to worry about their usage of the network incurring additional costs (users connected to the Internet through a public access providers sometimes do have usage based charging already). The introduction of pricing, especially usage-based pricing, into the Internet will involve major changes in both design and culture. While most popular user interfaces hide the details of network activity, once charging is in place the user interface will need to reveal the costs to the user. Also, the basic network architecture must incorporate sufficient capabilities to do the requisite authentication, accounting, and billing. Perhaps most importantly, such charging could alter the "gift economy" and browsing mentality that exists in the Internet today. The Internet is such an exciting development largely because of the cornucopia of information and resources freely available; unfortunately, once there are charges for network usage users will be less likely to disseminate information widely. Moreover, many users spend hours browsing through the Internet, much as one browses through a bookstore; if network usage were charged, just as if you were charged to take books off of a shelf in a bookstore, such browsing would be seriously curtailed. Thus it appears that neither the user interfaces nor the basic architecture are ready to support such pricing, and it is also not clear that the current Internet culture could survive its introduction.

Do these dire consequences mean that we should not extend the service model? This is certainly a debatable point, and one that should be debated more extensively than it has. It should be noted that while extending the service model raises the issues of incentives when deciding *how* to send data, even in the case of a single class of best-effort service we need to address the incentives of *whether* to send data. Currently, there are no limits on usage except the bandwidth of the access line, but rough adherence to rules of etiquette, along with adequate provisioning, have kept the Internet in relatively good shape. However, once "junk mailing" becomes commonplace and automatic browsers (agents automatically browsing the Internet and retrieving anything that looks interesting) are widely deployed, the Internet will suffer.[21] Even in the absence of multiple qualities of service, we need some incentive system to discourage or at least prioritize use; MacKie-Mason and Varian have explored this issue in [29]. Thus, while usage-based pricing may be a terrible thing, we need to confront its existence regardless of our decisions about extending the service model.

We should also note that there are pricing mechanisms which would have less of a negative impact, and that there are incentive schemes which do not rely primarily on pricing. For instance, quotas could be applied to an institution at its access point to the Internet, and then the issue of allocating service within those quotas becomes a local problem for that institution which, in many cases, could be handled through informal social conventions. Also, if the network offered reservations for real-time applications, then usage pricing could be centered on the high quality real-time services and not applied, at least in the near term, to the lower quality services. Moreover, much of the authentication and accounting infrastructure for this charging could be added along with the reservation mechanism, and so the best-effort architecture could be left relatively intact. This

[21] Of course, this assumes, with some justification, that one cannot provision the Internet so that if everyone is sending at the same time the service is acceptable.

would leave undisturbed the cultural aspects of the current best-effort Internet while charging a premium for high quality video and audio connections.

Clearly, these incentive issues are extremely important and many issues remain unresolved. See [4, 8, 29, 35] for further discussion of pricing and incentive issues.

5.3 Stability

The other implication of the explicit approach is that the network service offerings must be known to applications.[22] Applications must know the set of services in order to ask for the service, and they must know the characterization of the delivered service – if any – in order to decide which service best meets their needs.[23] Since knowledge of the service model must be embedded in applications, the service model must remain stable, though extensible. That is, it can be extended further but the services that are already in place cannot be easily altered because this would interfere with the installed application base.

The service model serves as the abstraction of network service that applications can be programmed against. Because of this, it is the most fundamental, and most stable, aspect of the network architecture. The underlying network technologies can change, and even IP can change, without disturbing applications, but the service model cannot. This inflexibility of the service model is both an advantage and a disadvantage. It is an advantage because the service model then provides a useful abstraction of network service. It is a disadvantage because if the initial service offerings are not well designed it is much more painful and disruptive to change them; in the implicit approach, such incremental adjustments would be invisible to applications.

In the explicit approach, the service model is not only stable, it is uniform. That is, there is single IP service model, as opposed to the implicit case where each router could use a different set of service offerings. This uniformity imposes a standardization requirements on network routers and subnets. A set of router and subnet standards, call them network element requirements, must be developed so that any concatenation of routers and subnets obeying these standards can support the end-to-end service offering advertised by IP. Note that the service model is a set of end-to-end services and it is up to the network to ensure that the services delivered at each link along the data path combine to support the offered end-to-end service.[24]

One of the aspects of the Internet architecture that contributed to its dramatic success is its ability to accommodate heterogeneous subnet technologies. Virtually any subnet technology can support IP, because IP does not require any particular performance; i.e., it requires protocol correctness but does not mandate when or even if packets arrive. If the service model is extended in any nontrivial

[22] Actually, it is more accurate to say the the service offerings must be known by the operating system; one could imagine some intermediary process requesting service on behalf of an application. We will just refer to applications below as the entity needing to know the service offerings, since this additional complication does not change the points we are making.

[23] And, in some cases, they need to know the characterization of that service in order to use it effectively. For instance, for a delay bounded service, applications with a fixed play-back point need to know the delay bound in order to set their play-back point.

[24] In some cases, like priority service, the mapping between the end-to-end service and the appropriate network element service is trivial. In others, like bounded delay service, this mapping is much more difficult.

manner, then we could end up in a situation where many subnet technologies (e.g., Ethernets) could no longer support IP which would seriously hinder the extension of IP connectivity and fragment the Internet. We can avoid this by designing the network element requirements so that (1) they only impose demanding performance standards for services that require admission control, and (2) they allow network elements to merely reject requests for services they cannot support. In this way, we can retain the property that virtually all subnet technologies can support IP and thus widespread IP connectivity can be maintained. More demanding services can be gradually deployed as the Internet infrastructure is slowly upgraded.

However, the uniformity of the service model does mean that the various communities responsible for standardizing subnet technologies (e.g., 802.X, ATM, etc.) must come together with the Internet community and agree on a single set of network element requirements. While not a technical challenge, this is indeed a severe organizational and political challenge considering the vastly different design traditions and assumptions in the various communities.[25]

5.4 Link-Sharing and other implicit services

Our analysis above, which focused on the service received by individual applications, suggests that the service will be requested explicitly by applications. However, there are exceptions to this general rule because there are some cases where we are more concerned with the service given to an aggregate of flows. For instance, a company may want to ensure that its aggregate traffic always has access to a certain amount of bandwidth along a path between two different locations. Or a university may want to make sure that bandwidth on the access line to the Internet is evenly split between various departments. One can use packet scheduling algorithms to accomplish this form of *link-sharing*; see [5, 13, 36, 39] for examples.

The point here is that one cannot request access to these link shares, access is implicitly given; your membership in a department is something over which you have no control.[26] This kind of service is indeed implicitly supplied. Because it is not designed to satisfy the detailed quality of service requirements of the individual users,[27] but rather is intended to meet the needs of organizations and other collective entities, this service is not subject to the arguments we made above that led us to conclude that the explicit approach was better. Such implicit services can also be used as a form of network management, for example dividing bandwidth between different protocol families. Because organizational and other collective service requirements are important, we expect that link-sharing and other implicitly supplied services will eventually play an important role in the future Internet architecture. These implicitly supplied services can be incrementally developed and deployed, and so it is not crucial that these parts of the architecture be immediately addressed.

[25] In [37] we argue that not only do the IP subnet technologies need to agree on the service model, but that competitors to IP, like IPX, DECnet, and ATM also need to come together and agree on a service model.

[26] An individual may have access to different credentials (e.g., is a member of two departments) and can choose to which one to associate her traffic, but that is different than being able to arbitrarily assert membership in any department.

[27] In particular, these implicit services cannot, by themselves, provide adequate service to real-time flows.

6 Do we need admission control?

We have argued that the Internet should extend its service model, but we have not yet discussed what services should be added. Some services, like best effort priority levels do not carry any quantitative characterization of delay or bandwidth and allow all traffic to be admitted much like in today's Internet. Other services, like bounded delay, do provide quantitative characterizations. Such quantitative services require explicit resource reservation and admission control; that is, the network must turn away additional flows when admitting them would lead to violating its quantitative service commitments. Thus, a key architectural decision to make is: should the network ever deny access to flows? In keeping with our assertion in Section 3, the answer depends on which choice maximizes the efficacy V of the network.

Admission control is typically used to prevent the network from becoming overloaded. One can also prevent overloading by overprovisioning the network. We first discuss what overloading means in the context of our utility function formulation, and then we address the extent to which overprovisioning could be used as a substitute for admission control.

6.1 Overloading

We typically think a network is overloaded if the delays are large and packets are being dropped. However, this perspective only addresses the service seen by individual users. Recall that our goal is to maximize the sum of the utilities. While allowing an additional flow into the network always decreases the utility of the existing flows, the question we need to ask is whether adding this additional flow decreases V. If so, we should not let the flow in.[28] Thus, one can define a network to be *overloaded* if the value of V goes up when one remove a flow.[29]

Let us consider the following *Gedanken* experiment. Consider an unlimited set of potential network users each using the same application on a network with a single congested link with bandwidth B. Since all applications are identical, and there is a single link, the service delivered to each application is merely a function of the bandwidth share allocated to each user; thus, we can use the simplifying notation $U(\frac{B}{n})$. We are interested in the behavior of the function $V(n) = nU(\frac{B}{n})$. The question of admission control, which is often mired in ideological differences, here reduces to the following simple mathematical question: for which value of n is the function $V(n)$ maximized? If $V(n)$ is always increasing and takes on its maximal value at $n = \infty$ then the network is never overloaded and we need not include admission control in the architecture. If $V(n)$ is maximized at some finite $\bar{n}$ then the network can overload; one then needs to decide whether to use admission control or overprovisioning as a method of avoiding such overloads.

Assuming that $U(x)$ is a nondecreasing function, we can make the following two statements about

[28]This is somewhat oversimplified, in that allowing a temporary decrease in V might lead to a later increase in V because of the effect on later admission control decisions, but we do not address that level of complexity in our qualitative treatment.

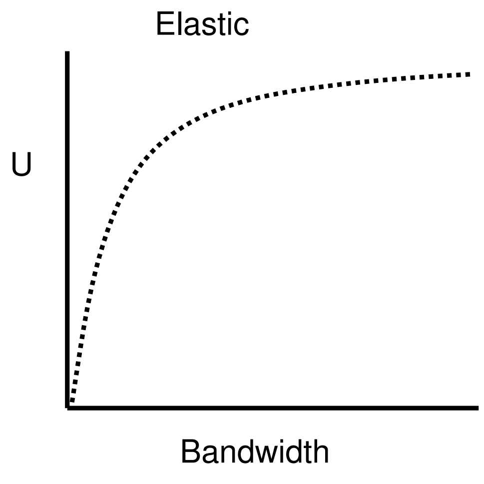
[29]A network which is not overloaded according to our technical definition can still be significantly underprovisioned; that is, even though the network is delivering lousy service to all clients, the total V is decreased if a single flow is removed. Our definition of overloading does not address the question of what loading levels offer the best cost/performance tradeoff.

$V(n)$. First, if there exists some $\epsilon > 0$ such that the function $U(x)$ is convex but not concave (i.e., not linear) in the neighborhood $[0, \epsilon)$, then there exists some $\bar{n}$ such that $V(\bar{n}) > V(n)$ for all $n > \bar{n}$. In this case, the network is overloaded whenever $n > \bar{n}$. Second, if the function $U(x)$ is everywhere strictly concave, then $V(n)$ is a strictly monotonically increasing function of n; in this case, the network is never overloaded. For example, if $U(x) = x^p$ then $V(n) = nU(\frac{B}{n}) \approx n^{1-p}$. For $p > 1$, $V(n)$ is maximized at $n = 1$ whereas for $p < 1$, $V(n)$ is maximized at $n = \infty$. Thus, the issue of overloading depends in detail on the shape of the utility curves. We now discuss a few sample application classes and their utility functions.

6.1.1 Utility Functions

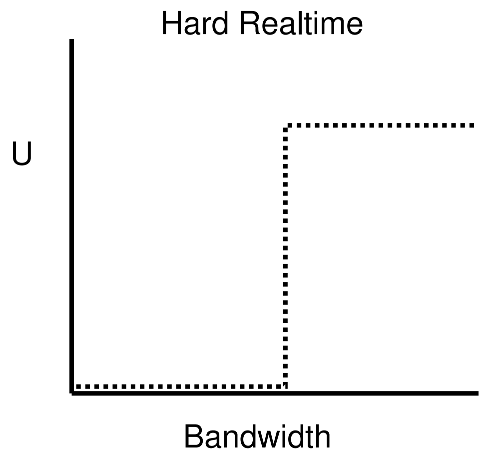
Recall that for our simple *Gedanken* experiment we are describing the service solely in terms of the bandwidth share. The functions U_i are then functions of a single variable and can be more easily analyzed. Our question is: what do typical utility functions look like? Since we care only about the qualitative properties of these functions, and there is little hard evidence for their shape (see [42]), we will make only guesses about their qualitative properties.

Traditional data applications like file transfer, electronic mail, and remote terminal are rather tolerant of delays. On an intuitive level they also would appear to have decreasing marginal degradation due to incremental increases in delays (i.e., adding an additional second of delay hurts much more when the delays are small than when the delays are big). We will call such applications elastic applications, and their utility functions look something like that below:



Here there is a diminishing marginal rate of performance enhancement as bandwidth is increased, so the function is strictly concave everywhere. V is always maximized when *no* users are denied access. For this class of applications, admission control has no role. This analysis reaffirms the original design choice of best-effort service for the Internet's architecture.

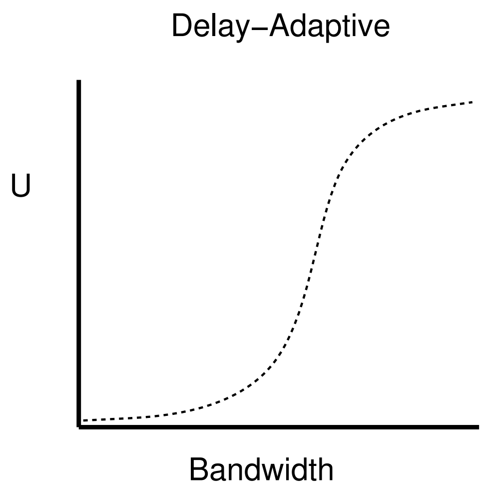
At the other extreme of delay sensitivity are applications with hard real-time requirements. These applications need their data to arrive within a given delay bound; the application does not care if packets arrive earlier, but the application performs very badly if packets arrive later than this bound. Examples of such applications are link emulation, traditional telephony, and other applications which expect circuit-switched service. For applications with hard real-time requirements, the utility curves look like the one below:



While the delay bounds are being met the application performance is constant, but as soon as the bandwidth share drops below that needed to meet the required delay bounds, the performance falls sharply to zero. A system with such applications becomes overloaded as soon as the bandwidth share falls below the critical level.

Traditionally video and audio applications have been designed with hard real-time requirements. However, as the current experiments on the Internet have dramatically shown, most audio and video applications can be implemented to be rather tolerant of occasional delay bound violations and dropped packets. However, such applications have an intrinsic bandwidth requirement because the data generation rate is independent of the network congestion. Thus, the performance degrades badly as soon as the bandwidth share becomes smaller than the intrinsic generation rate.

For delay-adaptive audio and video applications, the utility function curves look something like this.

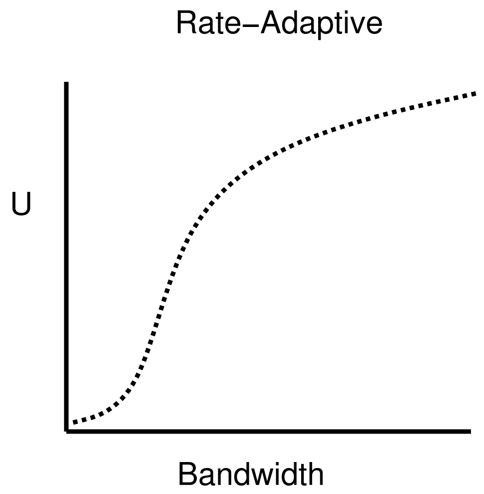


Note that the dropoff in performance is not nearly so sharp as with hard real-time applications, but the general shape of the curve is very similar. In particular, this utility function is convex but not concave in a neighborhood around zero. Thus, the network can become overloaded with such applications; the exact location of the overloading point will depend on the exact shape of the utility curve around the inflection point.

This analysis suggests why the Internet community and the telephony community have been at an impasse for years over the issue of best-effort vs. real-time service. The Internet community built a network to support data transfer applications that fall into the elastic category. For them, the decision to offer best-effort service was the natural and correct one. It is only now that video and other real-time applications are being widely used on the Internet, and with their appearance are coming calls for admission control.

Similarly, the telephony community built a network around an application (voice) with hard real-time requirements. When the community designed ATM to service these applications, the decision to offer real-time service and use admission control was again the natural and correct choice. Now that data services are being contemplated for ATM networks, the idea that ATM should offer best-effort has properly arisen.[30]

There is another class of real-time applications. Rate-adaptive applications adjust their transmission rate[31] in response to network congestion. This adjustment keeps the delays moderate no matter what the bandwidth share. Thus, the performance of the application depends completely on the quality of the signal. Certainly at high bandwidths the marginal utility of additional bandwidth is very slight because the signal quality is much better than humans need. It also appears that at very small bandwidths, the marginal utility is very slight because the signal quality is unbearably low (see [42] for some human factors studies). The curves take the following shape:



Similar to the regular video utility functions, these utility functions are convex but not concave in a neighborhood around zero so the network can become overloaded with such applications.

[30]Of course, one can say that the real mistake was to design the application (voice) to have such rigid behavior and that if voice applications were made rate-adaptive then admission control need not have been introduced.

[31]Whether or not the adjustment is actually done at the source, or done by the network dropping a subset of packets, is immaterial here. We are including in this class hierarchically encodings where the network preferentially drops the higher levels of the encodings when the network is congested. In addition, we are assuming that all rate-adaptive applications are also delay-adaptive.

However, the overloading point is much smaller than those of regular delay-adaptive applications. The overloading point of delay-adaptive applications is tied to the bandwidth consumption in the normal case, whereas the overloading point of rate-adaptive applications is tied to the bandwidth consumption of the minimally acceptable signal quality.

We should note that our simple *Gedanken* experiment can be generalized to mixtures of different types of applications with the same conclusions; the treatment is complicated by the need to describe the relative allocation of bandwidth to the various different applications.

This *Gedanken* experiment suggests that for a network with only traditional data applications, efficacy is maximized by accepting all flows. However, when there are real-time applications, whether hard real-time, or delay-adaptive, or even rate-adaptive, then efficacy is maximized when some flows are turned away. We now address the question: should one build an architecture that includes admission control, or should one overprovision the network so that overloading rarely occurs?

6.2 Overprovisioning

If one could cost-effectively overprovision the network so that under natural conditions the offered load rarely exceeded the overloading point, then one might choose to do that rather than include admission control in the architecture. We don't require that overloading never occurs, we merely require that overloading occur on a link no more often than the mean time between failures on that link; then, overloading is just another failure mode. We are not asking if there are individual links that can be overprovisioned; undoubtedly there are. Rather, since the IP architecture is uniform, we are asking if it is cost-effective to overprovision the entire network.

This question can be addressed on two different time scales: short-term and long-term. The analysis of both cases is based on speculations about future developments and is therefore inherently unreliable.

In the short-term, say in the next five years or so, demand for high-bandwidth video will increase rapidly as fast LAN's are deployed and workstation video technology improves. Multimegabit video streams will become commonplace in many environments and pockets of high bandwidth users are likely to develop. However, the access lines between these pockets and the rest of the Internet are likely to remain comparatively slow and will thus frequently be overloaded. This isn't a technical problem about providing bandwidth, but merely an economic one. The cost of a high speed LAN will be much less than the cost of a high speed access line into the Internet, and for workgroups it is much more important to have high speed connectivity to coworkers than to the outside world. Clearly in the short term there is no hope of overprovisioning everywhere.

The long-term analysis is considerably less clear. By long-term we mean when networking is a mature and competitive industry and workstation technology has progressed to a point where getting bandwidth onto networks is no longer a significant bottleneck.

The phone network in the U. S. is an example of a network which, in its mature state, has been successfully overprovisioned; the rate of call blocking is extremely low in most areas. This overprovisioning is now part of users' expectations; users in the U. S. would probably be extremely dissatisfied with a telephone network which had significant levels of blocking. If one provisioned the Internet so that admission control rejected requests very rarely, then one needn't have implemented admission control in the first place. Can the Internet follow the example of the telephone company?

There are some important differences between the Internet and the phone network. A user of the phone network can do one thing: place a phone call. The bandwidth usage is fixed, and the invocation of the call usually requires human action. Both of these factors limit the variability in aggregate telephone usage. In contrast, the future Internet load will be much more variable. Video flows will range in bandwidth from a few 10's of kbps to perhaps as high as 100 mbps, and some other applications such as data collection from remote sensors may reach even higher bandwidth rates. Recall that in the central limit theorem, which says that the sum of individual distributions tends toward a Gaussian, the variance of the aggregate is proportional to the variance of the individual distributions. Thus, this variability in each individual use means that the aggregate usage will be more variable. In addition, because computer communications often do not involve humans (whose locations are fairly stable) on both ends, we expect that their usage pattern to be much more unpredictable; for instance, the migration of a popular on-line video repository could cause a major shift in traffic patterns. The higher variation of Internet traffic loads is the key reason why we think overprovisioning of the Internet will not be a cost-effective solution.[32] To see this more clearly, we will make a somewhat artificial distinction between *normal* usage and *leading edge* usage.

The term *normal* usage will refer to those flows of moderate bandwidth; let's say something less than about 1mbps. This will encompass digital telephony, relatively low quality video, and many other applications. The demand from such uses will probably have a variance that is small compared to the average; this is what the telephone network has. For such usage patterns, provisioning the link so that overloads are rare requires a relatively small percentage increase in the capacity. Moreover, when so provisioned, the utilization is still moderate.[33] We expect most normal users would willingly pay the extra expense to overprovision in return for having a very low call blockage rate.

The term *leading edge* usage will refer to those flows with extremely high bandwidth usage. Here, the variance in demand is large compared to the average demand. Overprovisioning requires a large percentage increase in the capacity of the link, and would result in low average utilization levels. Since we expect that a leading edge user might consume as much as 1000 times the bandwidth of the average user, these leading edge users will always be able to make a large impact on the network even if they make up a small fraction of the total population.

A simple example can illustrate the difference in variance between these two kinds of usage. We consider only real-time traffic, so each flow has to establish a reservation. The demand will be modeled by a Poisson stream of flows with arrival rate λ (this is the arrival rate of newly established flows, not the packet arrival rate). Each flow has an exponentially distributed holding time with

[32]There has been recent work showing that many aspects of computer network traffic exhibits *self-similar* behavior [1, 14, 28]. While it is not clear what the nature of the future real-time traffic will be, self-similarity would make it more likely that the demand would have sizable variance.

[33]We are using vague terms like "moderate" and "relatively small" because our main point is to compare this case to the leading edge case below.

average holding time μ. Define $\rho \equiv \frac{\lambda}{\mu}$. To model normal usage, we assume that each flow consumes a unit amount of bandwidth. Thus, the probability that the aggregate bandwidth consumption is greater than some capacity C (assumed, for convenience, to be an integer) is given by ρ^C. To model leading edge usage, we use a different arrival rate $\tilde{\lambda}$ but the same holding time distribution; define $\tilde{\rho} \equiv \frac{\tilde{\lambda}}{\mu}$. We further assume that each leading edge flow consumes L units of bandwidth. Define the capacity $\tilde{C}$ such that the probability of overflow (above capacity $\tilde{C}$) in this leading edge system is the same as the probability of overflow (above capacity C) in the normal usage system. Then, $\tilde{C} = LC\frac{\ln \rho}{\ln \tilde{\rho}}$. Note that when we fix the leading edge usage to be some fraction r of normal usage, $\tilde{\rho} = \frac{r}{L}\rho$, and we let the size of the leading edge jobs grow, then asymptotically $\tilde{C} \approx C\frac{L}{\ln L}$. If we use the values $\rho = 0.5$, $L = 100$, and $r = 0.1$, then $\frac{\tilde{C}}{C} \approx 9.12$. Thus, the capacity needed to overprovision the leading edge system (even without mixing the two classes of users together) is much larger than the capacity needed to overprovision the normal system, even though the average use in the leading edge system is much less than the average use in normal system.

The key question is: who will pay for the increased capacity needed to overprovision? It is the leading edge users that lead to the tremendous variance in demand. If the costs of expanding capacity are spread equally among all users then, in some sense, the normal users are subsidizing the leading edge users. Because we expect telecommunications to be a fully competitive market in the future, such cross-subsidizations cannot be maintained.[34] Thus, the cost-benefit tradeoff for these leading edge users is to either accept a significant call blockage rate or to bear the burden for the bandwidth needed for overprovisioning. The bandwidth needed to overprovision is many times larger than the average demand of these users, so this extra expense will be quite large. Since these users do not have entrenched service expectations (in contrast, users of the U. S. phone system do have such expectations), we expect that accepting the increased call blockage rate to be far preferable to paying the exorbitant cost of overprovisioning.

This very rough line of reasoning suggests that the Internet will be provisioned so that the call blockage rate for normal uses is extremely low, much like the phone network, but that the call blockage rate for extraordinary uses, like very high quality video or massive real-time data streams, is significant. We do not expect that overprovisioning will be cost-effective, and that networks that attempt to overprovision in the place of using admission control will lose in the market place to networks which provision less and use admission control to avoid overloading.

6.3 Discussion

According to the arguments above, networks with real-time applications do indeed overload, and the overprovisioning of such networks would not be cost-effective because of the high variance in demand. This conclusion could be modified if one assumes that almost all future real-time applications will be rate-adaptive.[35] Recall that the overloading point for such applications is at rather low bandwidth. If this overloading point is in the regime of *normal* usage, it may be possible to provision the network so that the overloading point is never reached. We do not yet

[34] By this we mean that competitors will offer alternative pricing schemes which focus the cost of expansion on the leading edge usage. Such a pricing scheme would attract the normal users away from a network which subsidized.

[35] Actually, we need only assume that the leading edge real-time applications will be rate-adaptive, since the network is likely to be overprovisioned for normal usage.

have sufficient experience with such rate-adaptivity to know if it will indeed become the dominant paradigm in real-time applications.

7 Concluding Comments

We have argued that the Internet should extend its service model, that service should be explicitly requested by applications, and that the service model should incorporate admission control. This leaves many other design decisions unresolved. For instance, the admission-controlled services could either be quantitatively characterized by, say, a delay bound or they could merely reflect a level-of-effort (e.g., priority levels with admission control but no delay bound). We do not address these more detailed issues here. Instead, our intent was to open a discussion about some of the more fundamental issues. Our formulation, based on utility functions and the goal of maximizing V, provides a framework for analyzing some of the design decisions. The detailed nature of these utility functions, as well the cost of the necessary mechanism to extend the service model, are issues that need more extensive analysis.

The discussion here focused on the Internet. However, our current telecommunications infrastructure is comprised of three different networks. Data is carried digitally over the Internet, while voice and video are carried analog over the telephone and cable networks respectively. Improvements in computer technology have led to the so-called digital convergence, where all of these media can now be processed digitally, and these improvements have also lead to the corresponding emergence of multimedia computer applications. These two developments have different implications. The emergence of multimedia applications means that the Internet, as the network designed to connect computers, should be capable of carrying such multimedia traffic. Digital convergence means that we could combine our separate network infrastructures into a single digital telecommunications network. The question of how to design IP so that the Internet could carry multimedia computer traffic is quite different than the question of how one would design IP so that it could serve as the unifying foundation of a national telecommunications infrastructure. The quality expectations of telephony and TV users are quite different than users of computer multimedia applications, and the cost tradeoffs of buffering and adapting are quite different if we assume the end device is a set-top box rather than a general purpose computer. For both cases, the design issues combine purely technical considerations with social and economic ones. In our discussion here we have tried to identify some of the key technical considerations, but claim no expertise on the social and economic ones.

8 Acknowledgements

This paper reflects insights gleaned from an enjoyable and fruitful collaboration with Dave Clark and Lixia Zhang. Ongoing discussions with my many other colleagues in the Integrated Service Internet Project – which is an informal collaboration involving researchers from Xerox PARC, MIT, USC and ISI – have also been crucial in shaping my thinking; I would specifically like to thank Steve Berson, Bob Braden, Lee Breslau, Deborah Estrin, Shai Herzog, Sugih Jamin, Danny Mitzel, John Wroclawski, and Daniel Zappala. In addition, I wish to thank Sally Floyd and Bryan Lyles for extremely useful conversations. I also want to express my gratitude to my colleagues Steve Deering, Christian Huitema, and Van Jacobson; while they think I am completely wrong on many of the technical points discussed here, their informed opposition has forced me to think more carefully about the issues involved.

References

[1] J. Beran, R. Sherman, M. S. Taqqu, and W. Willinger. *Variable Bit Rate Video Traffic and Long Range Dependence* In **IEEE/ACM Transactions on Networking**, to appear.

[2] R. Braden, D. Clark, and S. Shenker. *Integrated Services in the Internet Architecture: an Overview*, RFC 1633.

[3] D. Clark, S. Shenker, and L. Zhang. *Supporting Real-Time Applications in an Integrated Services Packet Network: Architecture and Mechanism* In **Proceedings of SIGCOMM '92**, pp 14-26, 1992.

[4] R. Cocchi, D. Estrin, S. Shenker, and L. Zhang. *Pricing in Computer Networks: Motivation, Formulation, and Example*, In **IEEE/ACM Transactions on Networking**, 1(6), pp. 614-627, 1993.

[5] J. Davin and A. Heybey. *A Simulation Study of Fair Queueing and Policy Enforcement*, In **Computer Communication Review**, 20(5), pp 23-29, 1990.

[6] A. Demers, S. Keshav, and S. Shenker. *Analysis and Simulation of a Fair Queueing Algorithm*, In **Journal of Internetworking: Research and Experience**, 1, pp. 3-26, 1990. Also in Proc. ACM SIGCOMM '89, pp 3-12.

[7] J. DeTreville and D. Sincoskie. *A Distributed Experimental Communications System*, In **IEEE JSAC**, Vol. 1, No. 6, pp 1070-1075, December 1983.

[8] D. Estrin and L. Zhang. *Design considerations for usage accounting and feedback in internetworks*, In **ACM Computer Communication Review**, 20(5), pp 56-66, October 1990.

[9] D. Ferrari. *Client Requirements for Real-Time Communication Services*, In **IEEE Communications Magazine**, 28(11), November 1990.

[10] D. Ferrari. *Distributed Delay Jitter Control in Packet-Switching Internetworks*, In **Journal of Internetworking: Research and Experience**, 4, pp. 1-20, 1993.

[11] D. Ferrari, A. Banerjea, and H. Zhang *Network Support for Multimedia*, preprint, 1992.

[12] D. Ferrari and D. Verma. *A Scheme for Real-Time Channel Establishment in Wide-Area Networks*, In **IEEE JSAC**, Vol. 8, No. 3, pp 368-379, April 1990.

[13] S. Floyd. *Link-sharing, resource management, and the future internet*, preprint, 1993.

[14] M. W. Garrett and W. Willinger. *Analysis, Modeling and Generation of Self-Similar VBR Video Traffic*, In **Proceedings of SIGCOMM '94**, 1994.

[15] S. J. Golestani. *A Stop and Go Queueing Framework for Congestion Management*, In **Proceedings of SIGCOMM '90**, pp 8-18, 1990.

[16] S. J. Golestani. *Duration-Limited Statistical Multiplexing of Delay Sensitive Traffic in Packet Networks*, In **Proceedings of INFOCOM '91**, 1991.

[17] I. Gopal and R. Guérin. *Network Transparency: The plaNET Approach*, In **IEEE/ACM Transactions on Networking**, 2(3), pp. 226-239, 1994.

[18] R. Guérin and L. Gün. *A Unified Approach to Bandwidth Allocation and Access Control in Fast Packet-Switched Networks*, In **Proceedings of INFOCOM '92**.

[19] R. Guérin, H. Ahmadi, and M. Naghshineh. *Equivalent Capacity and Its Application to Bandwidth Allocation in High-Speed Networks*, In **IEEE JSAC**, Vol. 9, No. 9, pp 968-981, September 1991.

[20] J. Hyman and A. Lazar. *MARS: The Magnet II Real-Time Scheduling Algorithm*, In **Proceedings of SIGCOMM '91**, pp 285-293, 1991.

[21] J. Hyman, A. Lazar, and G. Pacifici. *Real-Time Scheduling with Quality of Service Constraints*, In **IEEE JSAC**, Vol. 9, No. 9, pp 1052-1063, September 1991.

[22] J. Hyman, A. Lazar, and G. Pacifici. *Joint Scheduling and Admission Control for ATS-based Switching Nodes*, In **Proceedings of SIGCOMM '92**, 1992.

[23] J. Hyman, A. Lazar, and G. Pacifici. *A Separation Principle Between Scheduling and Admission Control for Broadband Switching*, In **IEEE JSAC**, Vol. 11, No. 4, pp 605-616, May 1993.

[24] Internet Society Press Release, August 4 1994.

[25] S. Jamin, S. Shenker, L. Zhang, and D. Clark. *An Admission Control Algorithm for Predictive Real-Time Service*, In **Proceedings of the Third International Workshop on Networking and Operating System Support for Digital Audio and Video**, 1992.

[26] C. Kalmanek, H. Kanakia, and S. Keshav. *Rate Controlled Servers for Very High-Speed Networks*, In **Proceedings of GlobeCom '90**, pp 300.3.1-300.3.9, 1990.

[27] J. Kurose. *Open Issues and Challenges in Providing Quality of Service Guarantees in High-Speed Networks*, In **Computer Communication Review**, 23(1), pp 6-15, 1993.

[28] W. E. Leland, M. S. Taqqu, W. Willinger, and D. V. Wilson. *On the Self-Similar Nature of Ethernet Traffic (Extended Version)*, In **IEEE/ACM Transactions on Networking**, 2(1), pp. 1-15, 1994.

[29] J. K. MacKie-Mason and H. Varian. *Pricing the Internet*, In **Public Access to the Internet**, B. Kahin and J. Keller (Eds.), Prentice-Hall, Englewood Cliffs, New Jersey, forthcoming.

[30] R. Nagarajan and J. Kurose. *On Defining, Computing, and Guaranteeing Quality-of-Service in High-Speed Networks*, In **Proceedings of INFOCOM '92**, 1992.

[31] A. Parekh and R. Gallager. *A Generalized Processor Sharing Approach to Flow Control- The Single Node Case*, In **IEEE/ACM Transactions on Networking**, 1(3), pp. 344-357, 1993.

[32] A. Parekh and R. Gallager. *A Generalized Processor Sharing Approach to Flow Control- The Multiple Node Case*, In **IEEE/ACM Transactions on Networking**, 2(2), pp. 137-150, 1994.

[33] A. Parekh. *A Generalized Processor Sharing Approach to Flow Control in Integrated Services Networks*, In **Technical Report LIDS-TR-2089**, Laboratory for Information and Decision Systems, Massachusetts Institute of Technology, 1992.

[34] H. Schulzrinne, J. Kurose, and D. Towsley. *Congestion Control for Real-Time Traffic*, In **Proceedings of INFOCOM '90**.

[35] S. Shenker *Service Models and Pricing Policies for an Integrated Services Internet*, In **Public Access to the Internet**, B. Kahin and J. Keller (Eds.), Prentice-Hall, Englewood Cliffs, New Jersey, forthcoming.

[36] S. Shenker, D. Clark, and L. Zhang. *A Scheduling Service Model and a Scheduling Architecture for an Integrated Services Packet Network*, preprint.

[37] S. Shenker, D. Clark, and L. Zhang. *Services or Infrastructure: Why We Need a Network Service Model*, In **Proceedings of Workshop on Community Networking**, pp 145-149, 1994.

[38] L. Smarr and C. Catlett. *Life after Internet: Making room for new applications*, In **Building Information Infrastructure**, edited by B. Kahin, 1992.

[39] M. Steenstrup. *Fair Share for Resource Allocation*, preprint, 1993.

[40] D. Verma, H. Zhang, and D. Ferrari. *Delay Jitter Control for Real-Time Communication in a Packet Switching Network*, In **Proceedings of TriCom '91**, pp 35-43, 1991.

[41] C. Weinstein and J. Forgie. *Experience with Speech Communication in Packet Networks*, In **IEEE JSAC**, Vol. 1, No. 6, pp 963-980, December 1983.

[42] F. Wilson, I. Wakeman, and W. Smith. *Quality of Service Parameters for Commercial Application of Video Telephony*, In **Human Factors in Telecommunication Symposium**, pp139-148, 1993.

[43] D. Yates, J. Kurose, D. Towsley, and M. Hluchyj. *On Per-Session End-to-End Delay Distribution and the Call Admission Problem for Real Time Applications with QOS Requirements*, In **Proceedings of SIGCOMM '93**, 1993.

[44] L. Zhang. *VirtualClock: A New Traffic Control Algorithm for Packet Switching Networks*, In **ACM Transactions on Computer Systems**, Vol. 9, No. 2, pp 101-124, May 1991. Also in Proc. ACM SIGCOMM '90, pp 19-29.