Portions of text in this paper have been extracted from *Quality of Service: Delivering QoS on the Internet and in Corporate Networks*, by Paul Ferguson and Geoff Huston, published by John Wiley & Sons, January 1998, ISBN 0-471-24358-2.

Quality of Service on the Internet: Fact, Fiction, or Compromise?

Paul FERGUSON <ferguson@cisco.com>
Cisco Systems, Inc.
USA

Geoff HUSTON <gih@telstra.net>
Telstra Internet
Australia

Abstract

The Internet has historically offered a single level of service, that of "best effort," where all data packets are treated with equity in the network. However, we are finding that the Internet itself does not offer a single level of service quality, and some areas of the network exhibit high levels of congestion and consequently poor quality, while other areas display consistent levels of high quality service. Customers are now voicing a requirement to define a consistent service quality they wish to be provided, and network service providers are seeking ways in which to implement such a requirement. This effort is happening within the umbrella called "Quality of Service" (QoS). Of course, this is now a phrase which has become overly used, often in vague, nondefinitive references. QoS discussions currently embrace abstract concepts, varying ideologies, and moreover, lack a unified definition of what QoS actually is and how it might be implemented. Subsequently, expectations regarding QoS have not been appropriately managed within the Internet community at large of how QoS technologies might be realistically deployed on a global scale. A more important question is whether ubiquitous end-to-end QoS is even realistic in the Internet, given the fact that the decentralized nature of the Internet does not lend itself to homogenous mechanisms to differentiate traffic. This paper examines the various methods of delivering QoS in the Internet, and attempts to provide an objective overview on whether QoS in the Internet is fact, fiction, or a matter of compromise.

Contents

1 Introduction

It is hard to dismiss the entrepreneurial nature of the Internet today -- this is no longer a research project. For most organizations connected to the global Internet, it's a full-fledged business interest. Having said that, it is equally hard to dismiss the poor service quality that is frequently experienced -- the rapid growth of the Internet, and increasing levels of traffic, make it difficult for Internet users to enjoy consistent and predictable end-to-end levels of service quality.

What causes poor service quality within the Internet? The glib and rather uninformative response is "localized instances of substandard network engineering which is incapable of carrying high traffic loads."

Perhaps the more appropriate question is "what are the components of service quality and how can they be measured?" *Service quality* in the Internet can be expressed as the combination of network-imposed *delay*, *jitter*, *bandwidth* and *reliability*.

Delay is the elapsed time for a packet to be passed from the sender, through the network, to the receiver. The higher the delay, the greater the stress that is placed on the transport protocol to operate efficiently.

For the TCP protocol, higher levels of delay imply greater amounts of data held "in transit" in the network, which in turn places stress on the counters and timers associated with the protocol. It should also be noted that TCP is a "self-clocking" protocol, where the sender's transmission rate is dynamically adjusted to the flow of signal information coming back from the receiver, via the reverse direction acknowledgments (ACKs), which notify the sender of successful reception. The greater the delay between sender and receiver, the more insensitive the feedback loop becomes, and therefore the protocol becomes more insensitive to short term dynamic changes in network load. For interactive voice and video applications, the introduction of delay causes the system to appear unresponsive.

Jitter is the variation in end-to-end transit delay (in mathematical terms it is measurable as the absolute value of the first differential of the sequence of individual delay measurements). High levels of jitter cause the TCP protocol to make very conservative estimates of round trip time (*RTT*), causing the protocol to operate inefficiently when it reverts to timeouts to reestablish a data flow. High levels of jitter in UDP-based applications are unacceptable in situations where the application is real-time based, such as an audio or video signal. In such cases, jitter causes the signal to be distorted, which in turn can only be rectified by increasing the receiver's reassembly playback queue, which effects the delay of the signal, making interactive sessions very cumbersome to maintain.

Bandwidth is the maximal data transfer rate that can be sustained between two end points. It should be noted that this is limited not only by the physical infrastructure of the traffic path within the transit networks, which provides an upper bound to available bandwidth, but is also limited by the number of other flows which share common components of this selected end-to-end path.

Reliability is commonly conceived of as a property of the transmission system, and in this context, it can be thought of as the average error rate of the medium. Reliability can also be a byproduct of the switching system, in that a poorly configured or poorly performing switching system can alter the order of packets in transit, delivering packets to the receiver in a different order than that of the original transmission by the sender, or even dropping packets through transient routing loops. Unreliable or error-prone network transit paths can also cause retransmission of the lost packets. TCP cannot distinguish between loss due to packet corruption and loss due to congestion, and packet loss invokes the same congestion avoidance behavior response from the sender, causing the sender's transmit rates to be reduced by invoking congestion avoidance algorithms even though no congestion may have been experienced by the network. In the case of UDP-based voice and video applications, unreliability causes induced distortion in the original analog signal at the receiver's end.

Accordingly, when we refer to differentiated service quality, we are referring to differentiation of one or more of these four basic quality metrics for a particular category of traffic.

Given that we can define some basic parameters of service quality, the next issue is how is service quality implemented within the Internet?

The Internet is composed of a collection of routers and transmission links. Routers receive an incoming packet, determine the next hop interface, and place the packet on the output queue for the selected interface. Transmission links have characteristics of delay, bandwidth and reliability. Poor service quality is typically encountered when the level of traffic selecting a particular hop exceeds the transmission bandwidth of the hop for an extended period of time. In such cases, the router's output queues associated with the saturated transmission hop begin to fill, causing additional transit delay (increased *jitter* and *delay*), until the point is reached where the queue is filled, and the router is then forced to discard packets (reduced *reliability*). This in turn forces adaptive flows to reduce their sending rate to minimize congestion loss, reducing the available *bandwidth* for the application. Poor service quality can be generated in other ways, as well. Instability in the routing protocols may cause the routers to rapidly alter their selection of the best next hop interface, causing traffic within an end-to-end flow to take divergent paths, which in turn will induce significant levels of *jitter*, and an increased probability of out-of-order packet delivery (reduced *reliability*).

Accordingly, when we refer to the quality of a service, we are looking at these four metrics as the base parameters of quality, and it must be noted that there are a variety of network events which can affect these parameter values.

Also, it should be noted that in attempting to take a uniform "best effort" network service environment and introduce structures which allow some form of service differentiation, the tools which allow such service environments to be constructed are configurations within the network's routers designed to implement one or more of the following:

- Signal the lower level transmission links to use a different transmission servicing criteria for particular service profiles,
- Alter the next hop selection algorithm to select a next hop which matches the desired service levels,
- Alter the router's queuing delay and packet discard algorithms such that packets are scheduled to receive transmission resources in accordance with their relative service level, and
- Alter the characteristics of the traffic flow as it enters the network to conform to a contracted profile and associated service level.

The "art" of implementing an effective QoS environment is to use these tools in a way which can construct robust differentiated service environments.

From this perspective, the concept of *service quality* is important to understand, as opposed to what most people call *quality of service*, or QoS. Service quality can be defined as delivering consistently predictable service, to include high network reliability, low delay, low jitter, and high availability. QoS, on the other hand, can be interpreted as a method to provide preferential treatment to some arbitrary amount of network traffic, as opposed to all traffic being treated as "best effort," and in providing such preferential treatment, attempting to increase the quality level of one or more of these basic metrics for this particular category of traffic. There are several tools available to provide this differentiation, ranging from preferential queuing disciplines to bandwidth reservation protocols, from ATM-layer congestion and bandwidth allocation mechanisms to traffic-shaping, each of which may be appropriate dependent on what problem is being solved. We do not see QoS as being principally concerned about attempting to deliver "guaranteed" levels of service to individual traffic flows within the Internet. While such network mechanisms may have a place within smaller network environments, the sheer size of today's Internet effectively precludes any QoS approach which attempts to reliably segment the network on a flow-by-flow basis. The major technology force which has driven the explosive growth of the Internet as a communications medium is the use of stateless switching systems which provide variable best effort service levels to intelligent peripheral devices. Recent experience has indicated that this approach has extraordinary scaling properties, where the stateless switching architectures can scale easily into scales of gigabits per second, preserving a continued functionality where the unit cost of stateless switching has decreased at a level which is close to the basic scaling rate.

We also suggest that if a network cannot provide a reasonable level of service quality, then attempting to provide some method of differentiated QoS on the same infrastructure is virtually impossible. This is where traditional engineering, design, and network architectural principles play a significant role.

2 QoS and network engineering, design, and architecture

Before examining methods to introduce QoS into the Internet, it is necessary to examine methods of constructing a network that exemplify sound network engineering principles -- scalability, stability, availability, and predictability.

Typically, this exercise entails a conservative approach in designing and operating the network, undertaking measures to ensure that the routing system within the network remains stable, and ensuring that peak level traffic flows sit comfortably within the bandwidth and switching capabilities of the network. Unfortunately, some of these seminal principles are ignored in favor of maximizing revenue potential. For example, it is not uncommon for an ISP (Internet Service Provider) to oversubscribe an access aggregation point by a factor of 25 to 1, especially in the case of calculating the number of subscribers compared to the number of available modem ports, nor is it uncommon to see interprovider exchange points experiencing peak packet drop rates in excess of 20% of all transit traffic. In a single quality best effort environment, the practice of oversubscription allows the introduction of additional load into the network at the expense of marginal degradation to all existing active subscribers. This can be a very dangerous practice, and if miscalculated, can result in seriously degraded service performance (due to induced congestion) for all subscribers. Therefore, oversubscription should not be done arbitrarily.

Network engineering is arguably a compromise between engineering capabilities for the average load levels, and engineering capabilities which are intended to handle peak load conditions. In the case of dimensioning access ports, close attention must be paid to user traffic characteristics and modem pool port usage, based on time-of-day and day-of-week, for an extended period prior to making such an engineering commitment. Even after such traffic analysis has been undertaken, the deployed configuration should be closely monitored on a continuing and consistent basis to detect changes in usage and characteristics. It is very easy to assume that only a fraction of subscribers may be active at any given time, or to assume average and peak usage rates, but without close observation and prior traffic sampling, a haphazard assumption could dramatically affect the survival of your business.

Equally, it is necessary for the network operator to understand the nature, size, and timing of traffic flows that are carried across the network's transmission systems. While a critical component of traffic analysis is the monitoring of the capacity on individual transmission links, monitoring the dispersion of end-to-end traffic flows allows the network operator to ensure that the designed transmission topology provides an efficient carriage for the data traffic. It also attempts to avoid the situation where major traffic flows are routed sub-optimally across multiple hops, incurring additional cost and potentially imposing a performance penalty through the transit through additional routing points.

The same can be said of maintaining stability in the ISP's network routing system. Failure to create a highly stable routing system can result in destinations being intermittently unreachable, and ultimately frustrating customers. Care should be taken on all similar infrastructure and "critical" service issues, such as DNS (Domain Name System) services. The expertise of the engineering and support staff will be reflected in the service quality of the network, like it or not.

Having said that, it is not difficult to understand that poorly designed networks do not lend themselves to QoS scenarios, due to the fact that if acceptable levels of service quality cannot be maintained, then it is quite likely that adding QoS in an effort to create some level of service differentiation will never be effective. Granted, it may allow the network performance to degrade more "gracefully" in times of severe congestion for some applications operated within the group of elevated QoS customers, but limiting the impact of degradation for some, at the cost of increasing the impact for the remainder of the customer base, is not the most ingenious or sensible use of QoS mechanisms. The introduction of QoS differentiation into the network is only partially effective if those customers who do not subscribe to a QoS service are adversely impacted. After all, if non-QoS subscribers are negatively impacted, they will seek other service providers for their connectivity, or they will be forced to subscribe to the QoS service to obtain an acceptable service level. This last sentence requires a bit of unconventional logic, since not all subscribers can realistically be QoS subscribers -- this violates some of the most fundamental QoS strategies.

The design principles which are necessary to support effective QoS mechanisms can be expressed in terms of the four base service quality parameters noted in the previous section -- delay, jitter, bandwidth, and reliability. In order to minimize delay, the network must be based upon a transmission topology which reflects the pattern of end-to-end traffic flows, and a routing system design which attempts to localize traffic such that minimal distance paths are always preferred. In order to minimize jitter, the routing system must be held in as stable a state as possible. Router queue depths must also be configured so that they remain within the same order of magnitude in size as the delay bandwidth product of the transmission link that is fed by the queue. Also, unconditional preferential queuing mechanisms should be avoided in favor of weighting or similar fair access queue mechanisms, to ensure that all classes of traffic are not delayed indefinitely while awaiting access to the transmission resources. Selection of maximum transfer unit (MTU) sizes should also be undertaken to avoid MTUs which are very much greater than the delay bandwidth product of the link -- again as a means of minimizing levels of network-induced jitter.

In terms of overall reliability, the onus is on the network architect to use transmission media which have a very low intrinsic bit error rate, and to use router components which have high levels of availability and stability. Care should also be taken to ensure that the routing system is configured to provide deterministic outcomes, minimizing the risk of packet reordering. Transmission capacity, or bandwidth, should be engineered to minimize the level of congestion-induced packet loss within the routers. This is perhaps not so straightforward as it sounds, given that transmission capacity is one of the major cost elements for an Internet network service provider, and the network architect typically has to assess the trade-off between the cost performance of the network, and the duration and impact of peak load conditions on the network. Typically, the network architect looks for average line utilization, and "busy hour" to "average hour" utilization ratios to provide acceptable levels of economic performance, while looking at busy hour performance figures to ensure that the network does not revert into a condition of congestion collapse at the points in time where usage is at a maximum.

It is only after these basic design steps have been undertaken, and a basic level of service quality achieved within the network, that the issue of QoS (or service level differentiation) can be examined in any productive manner. The general conclusion here is that you cannot introduce QoS mechanisms to salvage a network which is delivering very poor levels of service. In order to be effective, QoS mechanisms need to be implemented in a network that is soundly engineered and which operates in a stable fashion under all levels of offered load.

3 QoS tools

Now that we have provided a framework definition for QoS, there are several mechanisms (and architectural implementations) which can provide differentiation for traffic in the network. We break these mechanisms into three basic groups, which align with the lower three layers of the OSI reference model -- the Physical, Link, and Network layers [Figure 1].

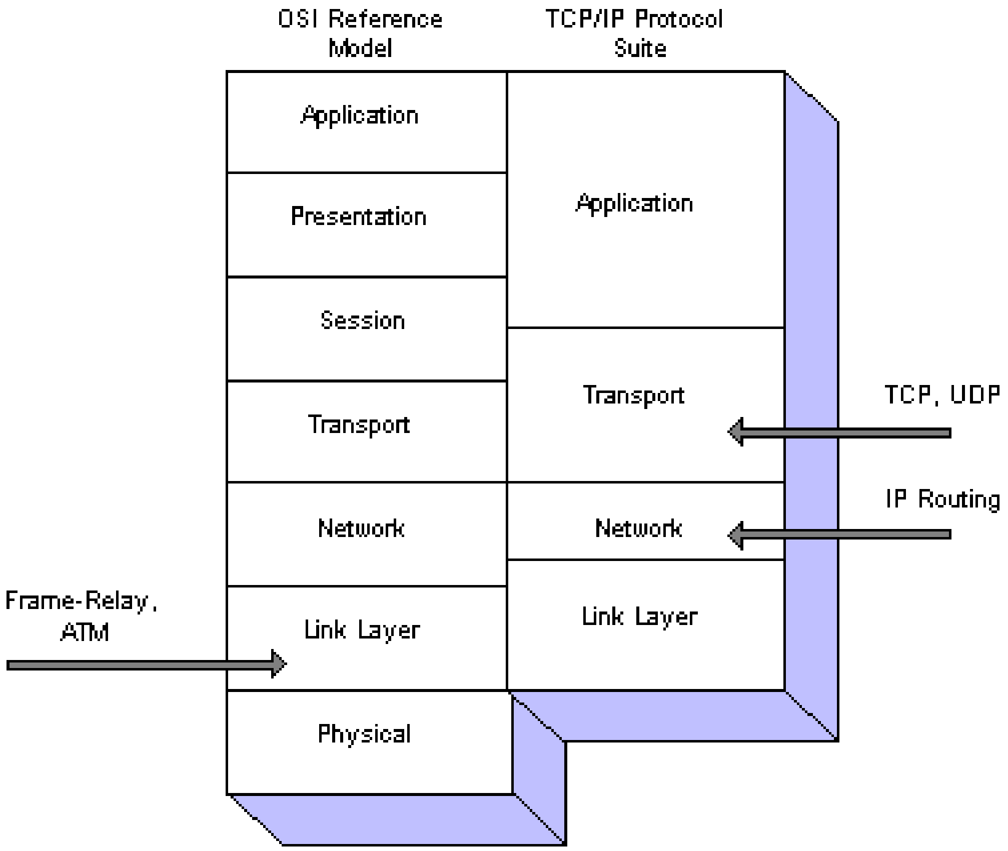

Figure 1

3.1 Physical layer mechanics

The physical layer (also referred to as L1, or Layer 1), consists of the physical wiring, fiber optics, and the transmission media in the network itself. It is reasonable to ask how Layer 1 physical media figures within the QoS framework, but the time-honored practice of constructing diverse physical paths in a network is, perhaps ironically, a primitive method of providing differentiated service levels. In some cases, diverse paths are constructed primarily to be used by network layer routing to provide for redundant availability, should the primary physical path fail for some reason, although often the temptation to share the load across the primary and backup paths is overwhelming. This can lead to adverse performance where, for example, having more than one physical path to a destination can theoretically allow for some arbitrary amount of network traffic to take the primary low-delay, high-bandwidth path, while the balance of the traffic takes a backup path which may have different delay and bandwidth properties. In turn, such a configuration leads to reduced reliability and increased jitter within the network as a consequence, unless the routing profile has been carefully constructed to stabilize the traffic segmentation between the two paths.

3.1.1 Alternate physical paths

While the implementation of provisioning diverse physical paths in a network is usually done to provide for backup and redundancy, this can also be used to provide differentiated services if the available paths each have differing characteristics. In Figure 2, for example, best-effort traffic could be forwarded by the network layer devices (routers) along the lower-speed path, while higher priority (QoS) traffic could be forwarded along the higher-speed path.

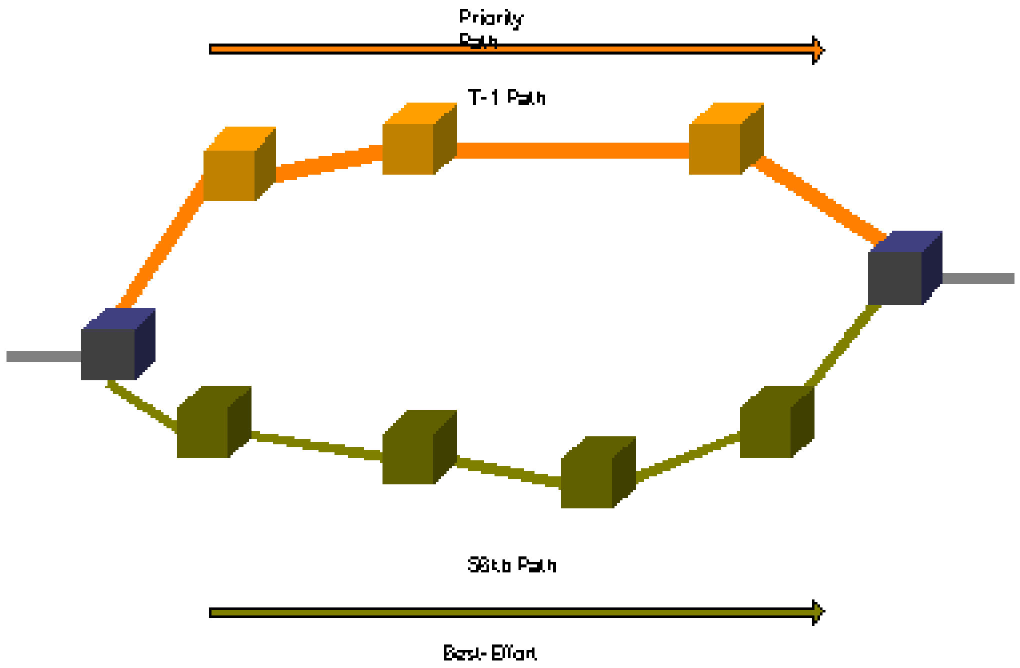

Figure 2

Alternatively, the scenario could be a satellite path complemented by a faster terrestrial cable path. Best effort traffic would be passed along the higher delay satellite path, while priority traffic would be routed along the terrestrial cable system.

Certainly, this type of approach is indeed primitive, and not without its pitfalls.

Destination-Based Routing and Path Selection

The method in which IP packets are forwarded in the Internet is based on the destination contained in the packet header. This is termed *destination-based routing*, and the byproduct of this mode of packet forwarding is that since packets are generally switched based on a local decision of the best path to the IP destination address contained in the IP packet header, the mechanisms which do exist to forward traffic based on its IP source address are not very robust. Accordingly, it is difficult to perform outbound path selection based on the characteristics of the traffic source. The default form of path selection is based on the identity of the receiver. This implies that a QoS differentiation mechanism using path selection would be most efficiently implemented on selection of incoming traffic, while outgoing traffic would adopt the QoS parameters of the receiver. A destination-based routed network cannot control the QoS paths of both incoming and outgoing traffic to any particular location. Each QoS path will be determined as a destination-based path selection, leading to the observation that in a heterogeneous QoS environment, asymmetric quality parameters on incoming and outgoing data flows will be observed. This is a significant issue for unidirectional UDP-based traffic flows, where it becomes the receiver who controls the quality level of the transmission, not the sender. It is also significant for TCP, where the data flow and the reverse ACK flows may take paths of differing quality. Given that the sender adapts its transmission rate via signaling ,which transits the complete round trip, the resultant quality of the entire flow is influenced by the lower of the two quality levels of the forward and reverse paths.

TCP and symmetric path selection

Reliable traffic transmission (TCP) requires a bidirectional data flow -- sessions which are initiated and established by a particular host (sender) generally require control traffic (e.g., explicit acknowledgments, or the notification that acknowledgments were not processed by the receiver) to be returned from the destination (receiver). This reverse data flow is used to determine the transmission success, out-of-order traffic reception at the receiver, transmission rate adaptation, or other maintenance and control signals, in order to operate correctly. In essence, this reverse flow allows the sender to infer what is happening along the forward path and at the receiver, allowing the sender to optimize the flow data rate to fully utilize its fair share of the forward paths resource level. Therefore, for a reliable traffic flow which is transmitted along a particular path at a particular differentiated quality level, the flow will need to have its return traffic flow traverse the same path at the same quality level if optimal flow rates are to be reliably maintained (this routing characteristic is also known as symmetric paths). Asymmetric paths in the Internet continue to be a troubling issue with regard to traffic which is sensitive to induced delay and differing service quality levels, which effectively distort the signal being generated by the receiver. This problem is predominantly due to local routing policies in individual administrative domains through which traffic in the Internet must traverse. It is especially unrealistic to expect path symmetry in the Internet, at least for the foreseeable future.

The conclusion is that path diversity does allow for differentiated service levels to be constructed from the different delay, bandwidth, and load characteristics of the various paths. However, for reliable transmission applications, this differentiation is relatively crude.

3.2 Link layer mechanics

There exists a belief that traffic service differentiation can be provided with specific link layer mechanisms (also referred to as Layer 2, or L2), and traditionally this belief in differentiation has been associated with Asynchronous Transfer Mode (ATM) and Frame Relay in the wide area network (WAN), and predominantly with ATM in the local area network (LAN) campus. A brief overview is provided here of how each of these technologies provides service differentiation, and additionally, we provide an overview of the newer IEEE 802.1p mechanics which may be useful to provide traffic differentiation on IEEE 802 style LAN media.

3.2.1 ATM

ATM is one of the few transmission technologies which provide data-transport speeds in excess of 155 Mbps today. As well as a high-speed bit-rate clock, ATM also provides a complex subset of traffic-management mechanisms, Virtual Circuit (VC) establishment controls, and various associated QoS parameters for these VCs. It is important to understand why these underlying transmission QoS mechanisms are not being exploited by a vast number of organizations that are using ATM as a data-transport tool for Internet networks in the wide area. The predominate use of ATM in today's Internet networks is simply because of the high data-clocking rate and multiplexing flexibility available with ATM implementations. There are few other transmission technologies which provide such a high speed bit-rate clock.

However, it is useful to examine the ATM VC service characteristics and examine their potential applicability to the Internet environment.

3.2.1.1 Constant bit rate (CBR)

The ATM CBR service category is used for virtual circuits that transport traffic at a consistent bit rate, where there is an inherent reliance on time synchronization between the traffic source and destination. CBR is tailored for any type of data for which the end-systems require predictable response time and a static amount of bandwidth continuously available for the lifetime of the connection. The amount of bandwidth is characterized by a Peak Cell Rate (PCR). These applications include services such as video conferencing, telephony (voice services), or any type of on-demand service, such as interactive voice and audio. For telephony and native voice applications, AAL1 (ATM Adaptation Layer 1) and CBR service is best suited to provide low-latency traffic with predictable delivery characteristics. In the same vein, the CBR service category typically is used for circuit emulation. For multimedia applications, such as video, you might want to choose the CBR service category for a compressed, frame-based, streaming video format over AAL5 for the same reasons.

3.2.1.2 Real-time and non-real-time variable bit rate (rt- and nrt-VBR)

The VBR service categories are generally used for any class of applications that might benefit from sending data at variable rates to most efficiently use network resources. Real-Time VBR (rt-VBR), for example, is used for multimedia applications with *lossy* properties, applications that can tolerate a small amount of cell loss without noticeably degrading the quality of the presentation. Some multimedia protocol formats may use a lossy compression scheme that provides these properties. Non-Real-Time VBR (nrt-VBR), on the other hand, is predominantly used for transaction-oriented applications, where traffic is sporadic and bursty.

The rt-VBR service category is used for connections that transport traffic at variable rates -- traffic that relies on accurate timing between the traffic source and destination. An example of traffic that requires this type of service category are variable rate, compressed video streams. Sources that use rt-VBR connections are expected to transmit at a rate that varies with time (e.g., traffic that can be considered bursty). Real-time VBR connections can be characterized by a Peak Cell Rate (PCR), Sustained Cell Rate (SCR), and Maximum Burst Size (MBS). Cells delayed beyond the value specified by the maximum CTD (Cell Transfer Delay) are assumed to be of significantly reduced value to the application, thus, delay is indeed considered in the rt-VBR service category.

The nrt-VBR service category is used for connections that transport variable bit rate traffic for which there is no inherent reliance on time synchronization between the traffic source and destination, but there is a need for an attempt at a guaranteed bandwidth or latency. An application that might require an nrt-VBR service category is Frame Relay interworking, where the Frame Relay CIR (Committed Information Rate) is mapped to a bandwidth guarantee in the ATM network. No delay bounds are associated with nrt-VBR service.

3.2.1.3 Available bit rate (ABR)

The ABR service category is similar to nrt-VBR, because it also is used for connections that transport variable bit rate traffic for which there is no reliance on time synchronization between the traffic source and destination, and for which no required guarantees of bandwidth or latency exist. ABR provides a best-effort transport service, in which flow-control mechanisms are used to adjust the amount of bandwidth available to the traffic originator. The ABR service category is designed primarily for any type of traffic that is not time sensitive and expects no guarantees of service. ABR service generally is considered preferable for TCP/IP traffic, as well as other LAN-based protocols, that can modify its transmission behavior in response to the ABR's rate-control mechanics.

ABR uses Resource Management (RM) cells to provide feedback that controls the traffic source in response to fluctuations in available resources within the interior ATM network. The specification for ABR flow control uses these RM cells to control the flow of cell traffic on ABR connections. The ABR service expects the end-system to adapt its traffic rate in accordance with the feedback so that it may obtain its fair share of available network resources. The goal of ABR service is to provide fast access to available network resources at up to the specified Peak Cell Rate (PCR).

3.2.1.4 Unspecified bit rate (UBR)

The UBR service category also is similar to nrt-VBR, because it is used for connections that transport variable bit rate traffic for which there is no reliance on time synchronization between the traffic source and destination. However, unlike ABR, there are no flow-control mechanisms to dynamically adjust the amount of bandwidth available to the user. UBR generally is used for applications that are very tolerant of delay and cell loss. UBR has enjoyed success in Internet LAN and WAN environments for store-and-forward traffic, such as file transfers and e-mail. Similar to the way in which upper-layer protocols react to ABR's traffic-control mechanisms, TCP/IP and other LAN-based traffic protocols can modify their transmission behavior in response to latency or cell loss in the ATM network.

3.2.1.5 The misconceptions about ATM QoS

Several observations must be made to realize the value of ATM QoS and its associated complexity. This section attempts to provide an objective overview of the problems associated with relying solely on ATM to provide QoS in the network. However, it sometimes is difficult to quantify the significance of some issues because of the complexity involved in the ATM QoS delivery mechanisms, and their interactions with higher-layer protocols and applications. In fact, the inherent complexity of ATM and its associated QoS mechanisms may be a big reason why many network operators are reluctant to implement those QoS mechanisms.

While the underlying recovery mechanism of ATM cell loss is signaling, the QoS structures for ATM VCs are excessively complex, and when tested against the principle of Occam's Razor (a popular translation frequently used in the engineering community for years is, "All things being equal, choose the solution that is simpler"), ATM by itself would not be the choice for QoS services, simply because of the complexity involved, compared to other technologies that provide similar results. Having said that, however, the application of Occam's Razor does not provide assurances that the desired result will be delivered -- instead, it simply expresses a preference for simplicity.

ATM enthusiasts correctly point out that ATM is complex for good reason -- in order to provide predictive, proactive, and real-time services, such as dynamic network resource allocation, resource guarantees, virtual circuit rerouting, and virtual circuit path establishment to accommodate subscriber QoS requests, but ATM's complexity is unavoidable. The underlying model of ATM is a heterogeneous client population where the real time service models assume simple clients which are highly intolerant of jitter, while the adaptive models assume very highly sophisticated clients which can opportunistically tune their data rates to variations in available capacity which may fluctuate greatly within time frames well inside normal end-to-end round trip times.

It also has been observed that higher-layer protocols, such as TCP/IP, provide the end-to-end transportation service in most cases, so that although it is possible to create QoS services in a lower layer of the protocol stack (namely ATM in this case), such services may cover only part of the end-to-end data path. This gets to the heart of the problem in delivering QoS with ATM, when the true end-to-end bearer service is not pervasive ATM. Such partial QoS measures often have their effects masked by the effects of the traffic distortion created from the remainder of the end-to-end path in which they do not reside, and hence the overall outcome of a partial QoS structure often is ineffectual. In other words, if ATM is not pervasively deployed end-to-end in the data path, efforts to deliver QoS using ATM can be unproductive. There is traffic distortion introduced into the ATM landscape by traffic-forwarding devices which service the ATM network and upper-layer protocols such as IP, TCP, and UDP, as well as other upper-layer network protocols. Queuing and buffering introduced into the network by routers and non-ATM-attached hosts skew the accuracy with which the lower-layer ATM services calculate delay and delay variation (jitter).

On a related note, some have suggested that most traffic on ATM networks would be primarily UBR or ABR connections, because higher-layer protocols and applications cannot request specific ATM QoS service classes, and therefore cannot fully exploit the QoS capabilities of the VBR service categories. A cursory examination of deployed ATM networks and their associated traffic profiles reveals that this is indeed the case, except in the rare instance when an academic or research organization has developed its own native "ATM-aware" applications that can fully exploit the QoS parameters available to the rt-VBR and nrt-VBR service categories. Although this certainly is possible, and has been done on several occasions, real-world experience reveals that this is the proverbial exception and not the rule.

It is interesting to note the observations published by Jagannath and Yin [1], which suggest that "it is not sufficient to have a lossless ATM subnetwork from the end-to-end performance point of view," especially in the case of ABR services. This observation is due to the fact that two distinct control loops exist -- ABR and TCP [Figure 3]. Although it generally is agreed that ABR can effectively control the congestion in the ATM network, ABR flow control simply pushes the congestion to the edges of the network (i.e., the routers), where performance degradation or packet loss may occur as a result. Jagannath and Yin also point out that "one may argue that the reduction in buffer requirements in the ATM switch by using ABR flow control may be at the expense of an increase in buffer requirements at the edge device (e.g., ATM router interface, legacy LAN to ATM switches)." Because most applications use the flow control provided by TCP, one might question the benefit of using ABR flow control at the subnetwork layer, because UBR (albeit with Early Packet Discard) is equally effective and much less complex. ABR flow control also may result in longer feedback delay for TCP control mechanisms, and this ultimately exacerbates the overall congestion problem in the network.

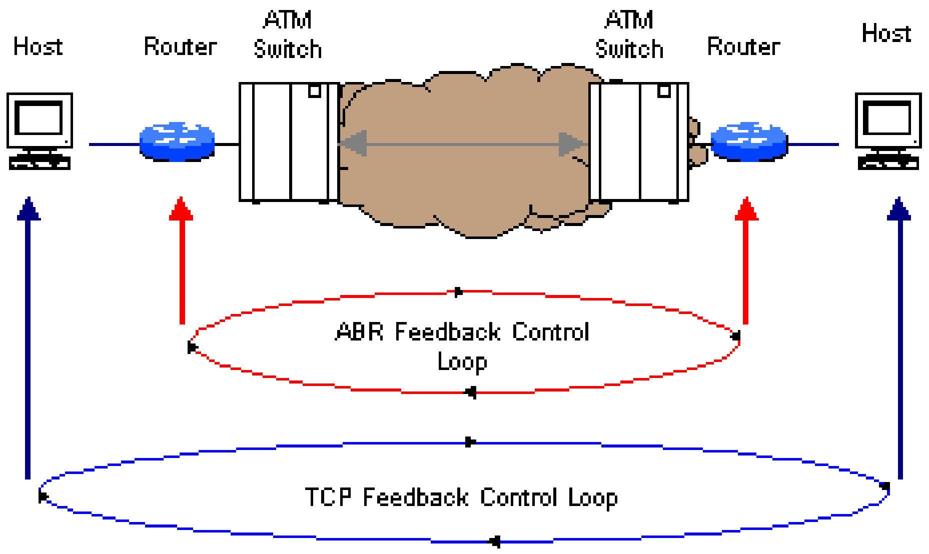

Figure 3

Aside from traditional data services that may use UBR, ABR, or VBR services, it is clear that circuit-emulation services which may be provisioned using the CBR service category can certainly provide the QoS necessary for telephony communications. However, this becomes an exercise in comparing apples and oranges. Delivering voice services on virtual digital circuits using circuit emulation is quite different from delivering packet-based data found in local-area and wide-area networks. Providing QoS in these two environments is substantially different -- it is substantially more difficult to deliver QoS for data, because the higher-layer applications and protocols do not provide the necessary hooks to exploit the QoS mechanisms in the ATM network. As a result, an intervening router must make the QoS request on behalf of the application, and thus the ATM network really has no way of discerning what type of QoS the application may truly require. This particular deficiency has been the topic of recent research and development efforts to address this shortcoming and investigate methods of allowing the end-systems to request network resources using RSVP, and then map these requests to native ATM QoS service classes as appropriate.

The QoS objective for networks similar in nature to the Internet lies principally in directing the network to alter the switching behavior at the IP layer, so that certain IP packets are delayed or discarded at the onset of congestion or delay, or completely avoid if at all possible, the impact of congestion on other classes of IP traffic. When looking at IP-over-ATM, the issue (as with IP-over-Frame Relay) is that there is no mechanism for mapping such IP-level directives to the ATM level, nor is it desirable, given the small size of ATM cells and the consequent requirement for rapid processing or discard. Attempting to increase the complexity of the ATM cell discard mechanics to the extent necessary to preserve the original IP QoS directives by mapping them into the ATM cell is arguably counterproductive.

Thus, it appears that the default IP QoS approach is best suited to IP-over-ATM. It also stands to reason that if the ATM network is adequately dimensioned to handle burst loads without the requirement to undertake large-scale congestion avoidance at the ATM layer, there is no need for the IP layer to invoke congestion-management mechanisms. Thus, the discussion comes full circle to an issue of capacity engineering, and not necessarily one of QoS within ATM.

3.2.2 Frame relay

Frame Relay's origins lie in the development of ISDN (Integrated Services Digital Network) technology, where Frame Relay originally was seen as a packet-service technology for ISDN networks. The Frame Relay rationale proposed was the perceived need for the efficient relaying of HDLC framed data across ISDN networks. With the removal of data link-layer error detection, retransmission, and flow control, Frame Relay opted for end-to-end signaling at the transport layer of the protocol stack to undertake these functions. This allows the network switches to consider data-link frames as being forwarded without waiting for positive acknowledgment from the next switch. This in turn allows the switches to operate with less memory and to drive faster circuits with the reduced switching functionality required by Frame Relay.

3.2.2.1 Frame relay rate management control structures

Frame Relay is a link layer protocol which attempts to provide a simple mechanism for arbitration of network oversubscription. Frame Relay decouples the characteristics of the network access link from the characteristics of the virtual circuits which connect the access system to its group peers. Each virtual circuit is configured with a traffic committed information rate (CIR), which conforms to a commitment on the part of the network to provide traffic delivery. However, any virtual circuit can also accept overflow traffic levels -- bursts which may transmit up to the rate of the access link. Such excess traffic is marked by the network access gateway using a single bit indicated in the Frame Relay frame header, termed the "Discard Eligible" (DE) bit.

The interior of the network uses three basic levels of threshold to manage switch queue congestion. At the first level of queue threshold, the network starts to mark frames with Explicit Congestion

Notification (ECN) bits. Frame relay congestion control is handled in two ways -- congestion avoidance and congestion recovery. Congestion avoidance consists of a Backward Explicit Congestion Notification (BECN) bit and a Forward Explicit Congestion Notification (FECN) bit, both of which are also contained in the Frame Relay frame header. The BECN bit provides a mechanism for any switch in the frame relay network to notify the originating node (sender) of potential congestion when there is a build-up of queued traffic in the switch's queues. This informs the sender that the transmission of additional traffic (frames) should be restricted. The FECN bit notifies the receiving node of potential future delays, informing the receiver to use possible mechanisms available in a higher-layer protocol to alert the transmitting node to restrict the flow of frames. The implied semantics of the congestion notification signaling is to notify senders and receivers to reduce their transmission rates to the CIR levels, although this action is not forced upon them. At the second level of queue threshold, the switch discards packets which are marked as DE, honoring its commitment to traffic which conforms to committed information rates on each circuit.

The basic premise within Frame Relay networks is that the switching fabric is dimensioned at such a level that it can fulfill its obligations of committed traffic flows. If this is not the case, and discarding of all discard eligible traffic fails to remove the condition which is causing switch congestion, the switch passes the third threshold of the queue, where it discards frames which form part of committed flow rates.

Frame Relay allows a basic level of oversubscription of basic point-to-point virtual circuits, where individual flows can increase their transfer rate, with the intent of occupying otherwise idle transmission capacity which is not being utilized by other virtual circuits which share the same transmission paths. When the sender is not using all of the committed rate within any of the configured virtual circuits, other VCs can utilize the transmission space with discard eligible frames.

Frame Relay indicates that it is possible to provide reasonable structures of basic service commitment, together with the added capability of provision of overcommitment using a very sparse link level signaling set -- the DE, FECN, and BECN bits.

3.2.2.2 Frame relay and Internet QoS

Frame Relay is certainly a good example of what is possible with relatively sparse signaling capability. However, the match between Frame Relay as a link layer protocol, and QoS mechanisms for the Internet, is not a particularly good one.

Frame Relay networks operate within a locally defined context of using selective frame discard as a means of enforcing rate limits on traffic as it enters the network. This is done as the primary response to congestion. The basis of this selection is undertaken without respect to any hints provided by the higher-layer protocols. The end-to-end TCP protocol uses packet loss as the primary signaling mechanism to indicate network congestion, but it is recognized only by the TCP session originator. The result is that when the network starts to reach a congestion state, the method in which end-system applications are degraded matches no particular imposed policy, and in this current environment, Frame Relay offers no great advantage over any other link layer technology in addressing this.

One can make the observation that in a heterogeneous network that uses a number of link layer technologies to support end-to-end data paths, the Frame Relay ECN and DE bits are not a panacea -- they do not provide for end-to-end signaling, and the router is not the system that manages either end of the end-to-end protocol stack. The router is more commonly performing IP packet into Frame Relay encapsulation. With this in mind, a more functional approach to user-selection of DE traffic is possible, one that uses a field in the IP header to indicate a defined quality level via a single discard eligibility field, and allow this designation to be carried end-to-end across the entire network path. With this facility, it then is logical to allow the ingress IP router (which performs the encapsulation of an IP datagram into a Frame Relay frame) to set the DE bit according to the bit setting indicated in the IP header field, and then pass the frame to the first-hop Frame Relay switch, which then can confirm or clear the DE bit in accordance with locally configured policy associated with the per-VC CIR.

The seminal observation regarding the interaction of QoS mechanisms within various levels of the model of the protocol stack is that without coherence between the link layer transport signaling structures and the higher-level protocol stack, the result, in terms of consistency of service quality, is completely chaotic.

3.2.3 IEEE 802.1p

It should be noted that an interesting set of proposed enhancements is being reviewed by the IEEE 802.1 Internetworking Task Group. These enhancements would provide a method to identify 802-style frames based on a simple priority. A supplement to the original IEEE MAC Bridges standard [2], the proposed 802.1p specification [3] provides a method to allow preferential queuing and access to media resources by traffic class, on the basis of a "priority" value signaled in the frame. The IEEE 802.1p specification, if adopted, will provide a way to transport this value (called *user priority*) across the subnetwork in a consistent method for Ethernet, token ring, or other MAC-layer media types using an extended frame format. Of course, this also implies that 802.1p-compliant hardware may have to be deployed to fully realize these capabilities.

The current 802.1p draft defines the user priority field as a 3-bit value, resulting in a variable range of values between zero and seven decimal, with seven indicating the highest relative priority and zero indicating the lowest relative priority. The IEEE 802.1p proposal does not make any suggestions on how the user priority should be used by the end-system or by network elements -- it only suggests that packets may be queued by LAN devices based on their relative user priority values.

While it is clear that the 802.1p user priority may indeed prove to be useful in some QoS implementations, it remains to be seen how it will be most practically beneficial. At least one proposal exists [4] which suggests how the 802.1p user priority values may be used in conjunction with the Subnet Bandwidth Manager (SBM), a proposal that allows LAN switches to participate in RSVP signaling and resource reservation objectives [5]. However, bearing in mind that the widespread deployment of RSVP in the global Internet is not wholly practical (as discussed in more detail below), it remains to be seen how QoS implementations will use this technology.

3.3 Network and transport layer mechanics

In the global Internet, it is undeniable that the common bearer service is the TCP/IP protocol suite, therefore, IP is indeed the common denominator. (The TCP/IP protocol suite is commonly referred to simply as *IP* -- this has become the networking vernacular used to describe IP, as well as ICMP, TCP, and UDP.) This thought process has several supporting lines of reason. The common denominator is chosen in the hope of using the most pervasive and ubiquitous protocol in the network, whether it be Layer 2 or Layer 3 (the network layer). Using the most pervasive protocol makes implementation, management, and troubleshooting much easier and yields a greater possibility of successfully providing a QoS implementation that actually works.

It is also the case that this particular technology operates in an end-to-end fashion, using a signaling mechanism that spans the entire traversal of the network in a consistent fashion. IP is the end-to-end transportation service in most cases, so that although it is possible to create QoS services in substrate layers of the protocol stack, such services only cover part of the end-to-end data path. Such partial measures often have their effects masked by the effects of the signal distortion created from the remainder of the end-to-end path in which they are not present, or introduce other signal distortion effects, and as mentioned previously, the overall outcome of a partial QoS structure is generally ineffectual.

When the end-to-end path does not consist of a single pervasive data-link layer, any effort to provide differentiation within a particular link-layer technology most likely will not provide the desired result. This is the case for several reasons. In the Internet, for example, an IP packet may traverse any number of heterogeneous link-layer paths, each of which may (or may not) possess characteristics that inherently provide methods to provide traffic differentiation. However, the packet also inevitably traverses links that cannot provide any type of differentiated services at the link layer, rendering an effort to provide QoS solely at the link layer an inadequate solution.

It should also be noted that the Internet today carries three basic categories of traffic, and any QoS environment must recognize and adjust itself to these three basic categories. The first category is *long held adaptive reliable traffic flows,* where the end-to-end flow rate is altered by the end points in response to network behavior, and where the flow rate attempts to optimize itself in an effort to obtain a fair share of the available resources on the end-to-end path. Typically, this category of traffic performs optimally for long held TCP traffic flows. The second category of traffic is a boundary case of the first category, *short duration reliable transactions,* where the flows are of very short duration, and the rate adaptation does not get established within the lifetime of the flow, so that the flow sits completely within the startup phase of the TCP adaptive flow control protocol. The third category of traffic is an *externally controlled load unidirectional traffic flow,* which is typically a result of compression of a real time audio or video signal, where the peak flow rate may equal the basic source signal rate, and the average flow rate is a byproduct of the level of signal compression used, and the transportation mechanism is an unreliable traffic flow with a UDP unicast flow model. Within most Internet networks today, empirical evidence indicates that the first category of traffic accounts for less than 1% of all packets, but as the data packets are typically large, this application accounts for some 20% of the volume of data. The second category of traffic is most commonly generated by World Wide Web servers using the HTTP/1.0 application protocol, and this traffic accounts for some 60% of all packets, and a comparable relative level of volume of data carried. The third category accounts for some 10% of all packets, and as the average packet size is less than one-third of the first two flow types, it currently accounts for some 5% of the total data volume.

In order to provide elevated service quality to these three common traffic flow types, there are three different engineering approaches that must be used. To ensure efficient carriage of long held, high volume TCP flows requires the network to offer consistent signaling to the sender regarding the onset of congestion loss within the network. To ensure efficient carriage of short duration, TCP traffic requires the network to avoid sending advance congestion signals to the flow end-points. Given that these flows are of short duration and low transfer rate, any such signaling will not achieve any appreciable load shedding, but will substantially increase the elapsed time that the flow is held active, which results in poor delivered service without any appreciable change in the relative allocation of network resources to service clients. To ensure efficient carriage of the externally clocked UDP traffic requires the network to be able to, at a minimum, segment the queue management of such traffic from adaptive TCP traffic flows, and possibly replace adaptation by advance notification and negotiation. Such a notification and negotiation model could allow the source to specify its traffic profile in advance, and have the network respond with either a commitment to carry such a load, or indicate that it does not have available resources to meet such an additional commitment.

As a consequence, it should be noted that no single transport or network layer mechanism will provide the capabilities for differentiated services for all flow types, and that a QoS network will deploy a number of mechanisms to meet the broad range of customer requirements in this area.

3.3.1 TCP congestion avoidance

As noted above, there are two major types of traffic flow in the Internet today. One is an adaptive-rate, reliable transmission, control-mediated traffic flow using TCP. The other is externally clocked, unreliable data flows which typically use UDP. Here, we look at QoS mechanisms for TCP, but to preface this it is necessary to briefly describe the behavior of TCP itself in terms of its rate control mechanisms.

TCP uses a rate control mechanism to achieve a sustainable steady state network load. The intent of the rate control mechanism is to reach a state where the sender injects a new data packet into the network at the same time that the receiver removes a data packet. However, this is modified by a requirement to allow dynamic rate probing, so that the sender attempts to inject slightly more data than is being removed by the receiver, and when this rate probing results in congestion, the sender will reduce its rate and again probe upwards to find a stable operating point. This is accomplished in TCP using two basic mechanisms.

The first mechanism used by TCP is *slow start*, where the connection is initialized with the transmission of a single segment. Each time the sender receives an ACK from the receiver, the congestion window (*cwnd*) is increased by one segment size. This effectively doubles the transmission rate for each round trip time (*RTT*) cycle -- the sender transmits a single packet and awaits the corresponding ACK. A TCP connection commences with an initial *cwnd* value of one, and a single segment is sent into the network. The sender then awaits the reception of the matching ACK from the receiver. When received, the *cwnd* is opened from one to two, allowing two packets to be sent. When each ACK is received from these two segments, the congestion window is incremented. The value of *cwnd* is then four, allowing four packets to be sent, and so on (receivers using delayed ACK will moderate this behavior such that the rate increase will be slightly less than doubled for each *RTT*).

This behavior will continue until the transfer is completed, or the sender's buffer space is exhausted, in which case the sender is transmitting at its maximal possible rate across the network's selected path, given the delay bandwidth product of the path, or an intermediate router in the path experiences queue exhaustion, and packets are dropped. Given that the algorithm tends to cluster transmission events at epoch intervals of the *RTT,* such overload of the router queue structure is highly likely when the path delay is significant.

The second TCP rate control mechanism is termed *congestion avoidance*. In the event of packet loss, as signaled by the reception of duplicate ACKs, the value of *cwnd* is halved, and this value is saved as the threshold value to terminate the *slow start* algorithm (*ssthresh*). When *cwnd* exceeds this threshold value, the window is increased in a linear fashion, opening the window by one segment size in each *RTT* interval. The value of *cwnd* is bought back to one when the end-to-end signaling collapses, and the sender times out on waiting for any ACK packets from the receiver. Since the value of *cwnd* is below the *ssthresh* value, TCP also switches to *slow start* control mode, doubling the congestion window with every *RTT* interval.

It should be noted that the responsiveness of the TCP congestion avoidance algorithm is measured in intervals of the *RTT*, and the overall intent of the algorithm is to reach a steady state where the sender injects a new segment into the network at the same point in time when the receiver accepts a segment from the network.

It should be noted that this algorithm works most efficiently when the spacing between ACK packets as received by the sender matches the spacing between data packets as they were received at the remote end. It should also be noted that the algorithm works optimally when congestion-induced packet loss happens just prior to complete queue exhaustion. The intent is to signal packet loss through duplicate ACK packets, allowing the sender to undertake a fast retransmission and leave the congestion window at the *ssthresh* level. If queue exhaustion occurs, the TCP session will stop receiving ACK signals and retransmission will only occur after timeout, and at that point the congestion window is bought back to a single segment, and the slow start algorithm is recommenced. An equal performance problem is network congestion events which occur within very short time intervals compared to the *RTT* (as may be the case in a mixed traffic load across a common ATM transport substrate, where short ATM switch cell queues may lead to very short interval cell loss events). In this case, a TCP session may switch from the aggressive window expansion of *slow start* into a much slower window expansion of *congestion avoidance* at a traffic rate well below the true long term network availability level.

One major drawback in any high-volume IP network is that when there are congestion hot spots, uncontrolled congestion can wreak havoc on the overall performance of the network to the point of congestion collapse. When a high volume of TCP flows are active at the same time, and a congestion situation occurs within the network at a particular bottleneck, each flow conceivably could experience loss at approximately the same time, creating what is known as *global synchronization*. *Global* refers to all TCP flows in a given network that traverse a common path. Global synchronization occurs when hundreds or thousands (or perhaps hundreds of thousands) of flows back off their transmission rates and revert to TCP slow start mode at roughly the same time. Each TCP sender detects loss and reacts accordingly, going into slow-start mode, shrinking its transmission window size, pausing for a moment, and then attempting to retransmit the data once again. If the congestion situation still exists, each TCP sender detects loss once again, and the process repeats itself over and over again, resulting in a network form of gridlock [6].

Uncontrolled congestion is detrimental to the network -- behavior becomes unpredictable, system buffers fill up, packets ultimately are dropped, and the byproduct is a large number of retransmits which could ultimately result in complete congestion collapse.

3.3.2 Preferential congestion avoidance at intermediate nodes

Van Jacobson discussed the basic methods of implementing congestion avoidance in TCP in 1988 [7]. However, Jacobson's approach was more suited for a small number of TCP flows, which is much less complex to manage than the volume of active flows in the Internet today. In 1993, Sally Floyd and Van Jacobson documented the concept of RED (Random Early Detection), which provides a mechanism to avoid congestion collapse by randomly dropping packets from arbitrary flows in an effort to avoid the problem of global synchronization and, ultimately, congestion collapsed [8]. The principal goal of RED is to avoid a "queue tail drop" situation in which all TCP flows experience congestion at the same time, and subsequent packet loss, thus avoiding global synchronization. RED also attempts to create TCP congestion signals using duplicate ACK signaling, rather than through sender timeout, which in turn produces a less catastrophic rate backoff by TCP. RED monitors the mean queue depth, and as the queue begins to fill, it begins to randomly select individual TCP flows from which to drop packets, in order to signal the receiver to slow down [Figure 4]. The threshold at which RED begins to drop packets is generally configurable by the network administrator, as well as the rate at which drops occur in relation to how quickly the queue fills. The more it fills, the greater the number of flows selected, and the greater the number of packets dropped. This results in signaling a greater number of senders to slow down, thus resulting in a more manageable congestion avoidance.

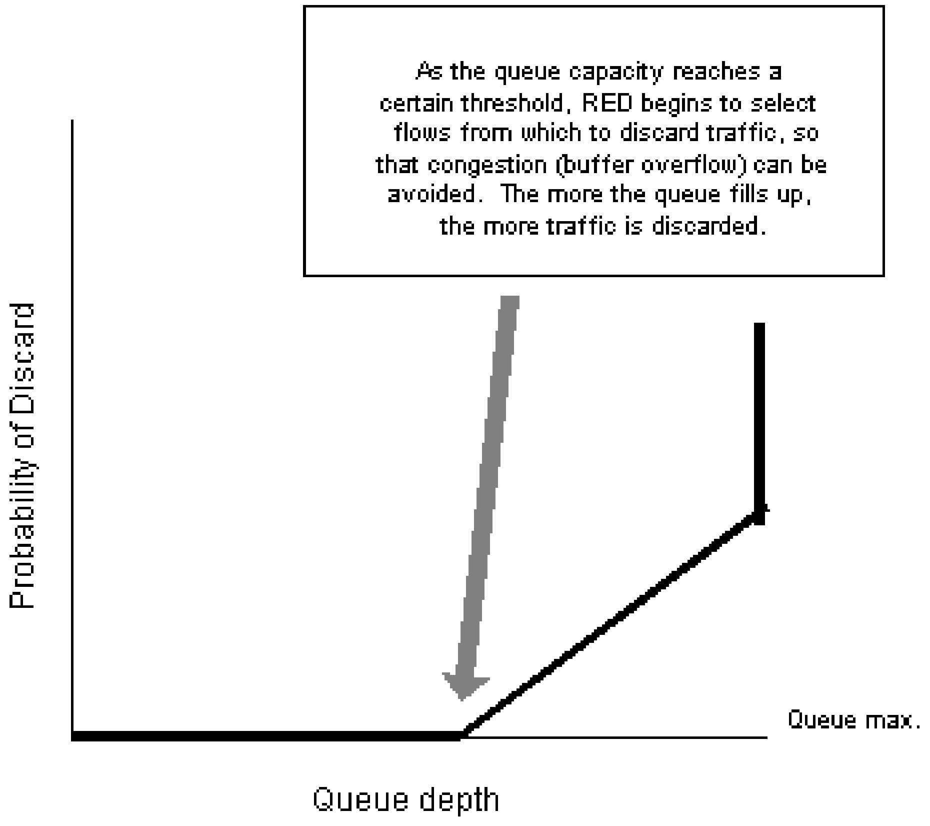

Figure 4

The RED approach does not possess the same undesirable overhead characteristics as some non-FIFO (First In, First Out) queuing techniques (e.g., simple priority queuing, class based queuing, weighted fair queuing). With RED, it is simply a matter of who gets into the queue in the first place -- no packet reordering or queue management takes place. When packets are placed into the outbound queue, they are transmitted in the order in which they are queued. Queue-based scheduling mechanisms, such as priority, class-based, and weighted-fair queuing, however, require a significant amount of computational overhead due to packet reordering and queue management. RED requires much less overhead than these non-FIFO queuing mechanisms, but then again, RED performs a completely different function.

RED can be said to be fair -- it chooses random flows from which to discard traffic in an effort to avoid global synchronization and congestion collapse, as well as to maintain equity in which traffic actually is discarded. Fairness is all well and good, but what is really needed for differentiated QoS structures is a tool that can induce unfairness -- a tool that can allow the network administrator to predetermine what traffic is dropped first (or last, as the case may be) when RED starts to select flows from which to discard packets. You can't differentiate services with fairness.

There are several proposals in the IETF which have suggested using the IP precedence subfield of the TOS (Type of Service) byte contained in the IP packet header to indicate the relative priority, or discard preference, of packets and to indicate how packets marked with these relative priorities should be treated within the network. As precedence is set or policed when traffic enters the network (at ingress), a weighted congestion avoidance mechanism implemented in the core routers determines which traffic should be discarded first when congestion is anticipated due to queue-depth capacity. The higher the precedence indicated in a packet, the lower the probability of discard. The lower the precedence, the higher the probability of discard. When the congestion avoidance is not actively discarding packets, all traffic is forwarded with equity.

Of course, for this type of operation to work properly, an intelligent congestion-control mechanism must be implemented on each router in the transit path. A least one currently deployed mechanism is available that provides an unfair, or weighted, behavior for RED. This deviation of RED yields the desired result for differentiated traffic discard in times of congestion and is called *Weighted Random Early Detection* (WRED). A similar scheme, called *enhanced RED,* is documented in a paper authored by Feng, Kandlur, Saha, and Shin [9].

3.3.3 Scheduling algorithms to implement differentiated service

There are other approaches to implement differential QoS within the network which use the routers queuing algorithm (or *scheduling discipline*) as the enabling mechanism. Considering that queuing is perhaps the optimal point to introduce QoS differentiation mechanisms, this is an area of considerable interest.

FIFO Queuing

The base of best effort, single quality, network environments is that of a FIFO queue, where there is no inherent differentiation undertaken by the router's transmission scheduler. Every packet which is scheduled to be transmitted on an output interface must await all previously scheduled packets before transmission. All such packets occupy slots in a single per-interface queue, and when the queue fills, all subsequent packets are discarded until the queue becomes available once more. As with the basic RED algorithm, this is a fair algorithm, given that it allocates the transmission resource fairly and imposes the same delay on all queued packets. For differentiated service levels, it is necessary to alter this fairness and introduce mechanisms to trigger preferential outcomes for classes of traffic.

The basic modification of the single level FIFO algorithm to enable differentiated QoS is to divide traffic into a number of categories, and then provide resources to each category in accordance with a predetermined allocation structure, implementing some form of proportional resource allocation.

Of course, the *Law of Conservation* holds here, such that the sum of the mean queuing delays per traffic category, weighted by their share of the resources they receive, is limited, with the corollary that in reducing the mean queuing delay for one category of traffic will result in the increase in mean queuing delay for one or more of the remaining categories of traffic [10]. Accordingly, you can't improve the performance profile of one class of traffic without adversely affecting the performance profile of one or more of the other classes of traffic, and the level of degradation will be similar in quantity to the level of improvement that was effected.

Priority Queues

A basic modification to the base FIFO structure is to create a number of distinct queues for each interface and associate a relative priority level with each. Packets are scheduled from a particular priority queue in FIFO order only when all queues of a higher priority are empty. In such a model, the highest priority traffic receives minimal delay, but all other priority levels may experience resource starvation if the highest precedence traffic queue remains occupied. To ensure that all traffic receives some level of service, it is a requirement that the network use admission policies which restrict the amount of traffic which is admitted at each elevated priority, or that the scheduling algorithm is adjusted to ensure that every priority class receives some minimum level of resource allocation. Accordingly, this simple priority mechanism does not scale well, although it can be implemented with relatively little cost, and more sophisticated (and more robust) scheduling algorithms are required within the Internet for QoS support.

Generalized Processor Sharing

The ideal approach is to associate a relative weight (or precedence) with each individual traffic flow and at every router, segment each traffic flow into an individual FIFO queue, and configure the scheduler to service all queues in a bit-wise round-robin fashion, allocating service to each flow in accordance with the relative weight. This is an instance of a *Generalized Processor Sharing* (*GPS*) discipline [11 - pp. 234-236].

Weighted Round-Robin and Deficit Weighted Round-Robin

Various scheduling techniques can approximate this model. A basic approach is to use a packet's marked precedence to place the packet into a precedence-based queue, and then use a *weighted round-robin* scheduling algorithm to service each queue. If all packets are identically sized, this is a relatively good approximation of *GPS*, but when packet sizes vary, this algorithm can exhibit significant deviation from a strict relative resource allocation strategy. This can be partially addressed using a *deficit weighted round-robin* [11 - pp. 237-238] algorithm which modifies the round-robin algorithm to use a service quantum unit. A packet is scheduled from the head of a weighted queue only if the packet size minus the per-queue deficit counter is less than the weighted quantum value, and the next packet in the queue is tested using a weighted quantum value which has been reduced by the size of the scheduled packet. When the test fails, the remaining weighted quantum size is added to the per-queue deficit counter, and the scheduler moves to the next queue. This algorithm performs with an average allocation which corresponds to the relative weights of each queue, but still exhibits unfairness within time frames which are commensurate to the maximum packet service time.

Weighted Fair Queuing

Weighted Fair Queuing (*WFQ*) [12] attempts to provide fairer resource allocation measures which protect "well behaved" traffic sources from uncontrolled sources. *WFQ* attempts to compute the finish time of each queued packet if a bit-wise weighted *GPS* scheduler had been used, and then schedules for service the packet with the smallest finish time which would've been receiving service in the corresponding GPS scheduler model. *WFQ* is both a scheduling and packet drop policy, where packet drop is based on a preference for dropping packets with the greatest finish time in response to an incoming packet which requires a queue slot. While *WFQ* requires a relatively complex implementation, it has a number of desirable properties. The scheduling algorithm does undertake fair allocation which does indeed ensure that different categories of traffic are not capable of resource starving other categories. The algorithm also bounds the queue delay per service category, which also provides a lever to create delay-bounded services without the need for resource reservation.

3.3.4 Traffic shaping non-adaptive flows

One of the more confounding aspects of providing differentiated services at the network and transport layers of the TCP/IP protocol suite is that of dealing with non-adaptive flows, or in other words, traffic flows which do not adapt their transmission rates in response to loss in the network. The most offensive of this category appear to be applications which use UDP as their transport protocol. This is somewhat ironic in that long-standing traditional thinking has assumed that applications which use UDP are generally thought to be designed to be "intelligent enough" to recognize loss, since UDP does not provide any error correction itself. The resulting observation is that this is a fundamentally flawed assumption, since it is generally recognized that applications which use UDP are generally not very "network friendly."

Having said that, the subsequent action is to rate-shape UDP flows as they enter the network, limiting their transmission rate to a specified bit rate. This method is arguably a compromise -- it's not pretty, but we understand how to do this, and it works. There are a couple of proposed methods to enhance the basic RED mechanism to provide some relief in the face of non-adaptive flows; however, the validity and practicality of these schemes are still being evaluated. One such proposal is discussed in [13].

3.4 Integrated services and RSVP

It has been suggested that the Integrated Services architecture [14] and RSVP [15] are excessively complex and possess poor scaling properties. This suggestion is undoubtedly prompted by the existence of the underlying complexity of the IP layer signaling requirements. However, it also can be suggested that RSVP is no more complex than some of the more advanced routing protocols. An alternative viewpoint might suggest that the underlying complexity is required because of the inherent difficulty in establishing and maintaining path and reservation state information along the transit path of data traffic. The suggestion that RSVP has poor scaling properties deserves additional examination, however, because deployment of RSVP has not been widespread enough to determine the scope of this assumption.

As discussed in [16], there are several areas of concern about the wide-scale deployment of RSVP. With regard to concerns of RSVP scalability, the resource requirements (computational processing and memory consumption) for implementing RSVP on routers increase in direct proportion to the number of separate RSVP reservations, or sessions, accommodated. Therefore, supporting a large number of RSVP reservations could introduce a significant negative impact on router performance. By the same token, router-forwarding performance may be impacted adversely by the packet classification and scheduling mechanisms intended to provide differentiated services for reserved flows. These scaling concerns tend to suggest that organizations with large, high-speed networks will be reluctant to deploy RSVP in the foreseeable future, at least until these concerns are addressed. The underlying implications of this concern also suggest that without deployment by Internet service providers, who own and maintain the high-speed backbone networks in the Internet, the deployment of pervasive RSVP services in the Internet will not be forthcoming.

Another important concern expressed in [16] deals with policy-control issues and RSVP. Policy control addresses the issue of who is authorized to make reservations and encompasses provisions to support access control and accounting. Although the current RSVP specification defines a mechanism for transporting policy information, it does not define the policies themselves, because the policy object is treated as an opaque element. Some vendors have indicated that they will use this policy object to provide proprietary mechanisms for policy control. At the time of this writing, however, the IETF has chartered a new working group, called the RSVP Admission Policy (rap) working group [17], to develop a simple policy-control mechanism to be used in conjunction with RSVP. There is ongoing work on this issue in this working group. Several mechanisms already have been proposed to deal with policy issues; however, it is unclear at this time whether any of these proposals will be implemented or adopted as a standard.

The key recommendation contained in [16] is that given the current form of the RSVP specification, multimedia applications run within smaller, private networks are the most likely to benefit from the deployment of RSVP. The inadequacies of RSVP scaling, and lack of policy control, may be more manageable within the confines of a smaller, more controlled network environment than in the expanse of the global Internet. It certainly is possible that RSVP may provide genuine value and find legitimate deployment utility in smaller networks, both in the peripheral Internet networks and in the private arena, where these issues of scale are far less critical. Therein lies the key to successfully delivering QoS using RSVP. After all, the purpose of the Integrated Services architecture and RSVP is to provide a method to offer quality of service, not to degrade the service quality.

4.0 Conclusions

A number of dichotomies exist within the Internet that tend to dominate efforts to engineer possible solutions to the quality of service requirement. Thus far, QoS has been viewed as a wide-ranging solution set against a very broad problem area. This fact often can be considered a liability. Ongoing efforts to provide "perfect" solutions have illustrated that attempts to solve all possible problems result in technologies that are far too complex, have poor scaling properties, or simply do not integrate well into the diversity of the Internet. By the same token, and by close examination of the issues and technologies available, some very clever mechanisms are revealed under close scrutiny. Determining the usefulness of these mechanisms, however, is perhaps the most challenging aspect in assessing the merit of any particular QoS approach.

In the global Internet, however, it becomes an issue of implementing QoS within the most common denominator -- this is clearly the TCP/IP protocol suite -- because a single link-layer media will never be used pervasively end-to-end across all possible paths. What about the suggestion that it is certainly possible to construct a smaller network of a pervasive link-layer technology, such as ATM? Although this is certainly possible in smaller private networks, and perhaps in smaller peripheral networks in the Internet, it is rarely the case that all end-systems are ATM-attached, and this does not appear to be a likely outcome in the coming years. In terms of implementing visibly differentiated services based on a quality metric, using ATM only on parts of the end-to-end path is not a viable answer. The ATM subpath is not aware of the complete network layer path, and it does not participate in the network or transport layer protocol end-to-end signaling.

The simplistic answer to this conundrum is to dispense with TCP/IP and run native cell-based applications from ATM-attached end-systems. This is certainly not a realistic approach in the Internet, though, and chances are that it is not very realistic in a smaller corporate network, either. Very little application support exists for native ATM. Of course, in theory, the same could have been said of Frame Relay transport technologies in the recent past, and undoubtedly will be claimed of forthcoming transport technologies in the future. In general, link layer technologies are similar to viewing the world through plumber's glasses -- every communications issue is seen in terms of point-to-point bit pipes. Each wave of transport technology attempts to add more features to the shape of the pipe, but the underlying architecture is a constant perception of the communications world as a set of one-on-one conversations, with each conversation supported by a form of singular communications channel.

One of the major enduring aspects of the communications industry is that no such thing as a ubiquitous single link layer technology exists. Hence, there is an enduring need for an internetworking end-to-end transport technology that can straddle a heterogeneous link layer substrate. Equally, there is a need for an internetworking technology that can allow differing models of communications, including fragmentary transfer, unidirectional data movement, multicast traffic, and adaptive data flow management.

This is not to say that ATM itself, or any other link layer technology for that matter, is not an appropriate technology to install into a network. Surely, ATM offers high-speed transport services, as well as the convenience of virtual circuits. However, what is perhaps more appropriate to consider is that any particular link layer technology is not effective insofar as providing QoS in the Internet for reasons that have been discussed thus far.

To quote a work in progress from the Internet Research Task Force, "The advantages of [the Internet Protocol's] connectionless design, flexibility and robustness, have been amply demonstrated. However, these advantages are not without cost -- careful design is required to provide good service under heavy load." [18]. Careful design is not exclusively the domain of the end-system's protocol stack, although good end-system stacks are of significant benefit. Careful design also includes consideration of the mechanisms within the routers that are intended to avoid congestion collapse. Differentiation of services places further demands on this design, because in attempting to allocate additional resources to certain classes of traffic, it is essential to ensure that the use of resources remains efficient, and that no class of traffic is totally starved of resources to the extent that it suffers throughput and efficiency collapse.

For QoS to be functional, it appears to be necessary that all the nodes in a given path behave in a similar fashion with respect to QoS parameters, or at the very least, do not impose additional QoS penalties other than conventional best effort into the end-to-end traffic environment. The sender (or network ingress point) must be able to create some form of signal associated with the data that can be used by downstream routers to potentially modify their default outbound interface selection, queuing behavior, and/or discard behavior.

The insidious issue here is attempting to exert "control at a distance." The objective in this QoS methodology is for an end-system to generate a packet that can trigger a differentiated handling of the packet by each node in the traffic path, so that the end-to-end behavior exhibits performance levels in line with the end-user's expectations and perhaps even a contracted fee structure.

This *control-at-a-distance* model can take the form of a "guarantee" between the user and the network. This guarantee is one in which, if the ingress traffic conforms to a certain profile, the egress traffic maintains that profile state, and the network does not distort the desired characteristics of the end-to-end traffic expected by the requestor. To provide such absolute guarantees, the network must maintain a transitive state along a determined path, where the first router commits resources to honor the traffic profile and passes this commitment along to a neighboring router that is closer to the nominated destination and also capable of committing to honor the same traffic profile. This is done on a hop-by-hop basis along the transit path between the sender and receiver, and yet again from receiver back to sender. This type of state maintenance is viable within small-scale networks, but in the heart of large-scale public networks such as the global Internet, the cost of state maintenance is overwhelming. Because this is the mode of operation of RSVP, this presents some serious scaling considerations and is inappropriate for deployment in large networks.

RSVP scaling considerations present another important point, however. RSVP's deployment constraints are not limited simply to the amount of resources it might consume on each network node as per-flow state maintenance is performed. It is easy to understand that as the number of discrete flows increases in the network, the more resources it will consume. Of course, this can be somewhat limited by defining how much of the network's resources are available to RSVP -- everything in excess of this value is treated as best-effort. What is more subtle, however, is that when all available RSVP resources are consumed, all further requests for QoS are rejected until RSVP-allocated resources are released. This is similar in functionality to how the telephone system works, where the network's response to a flow request is commitment or denial, and such a service does not prove to be a viable method to operate a data network where better-than-best-effort services arguably should always be available.

The alternative to state maintenance and resource reservation schemes is the use of mechanisms for preferential allocation of resources, essentially creating varying levels of best-effort. Given the absence of end-to-end guarantees of traffic flows, this removes the criteria for absolute state maintenance, so that "better-than-best-effort" traffic with classes of distinction can be constructed inside larger networks. Currently, the most promising direction for such better-than-best-effort systems appears to lie within the area of modifying the network layer queuing and discard algorithms. These mechanisms rely on an attribute value within the IP packet's header, so these queuing and discard preferences can be made at each intermediate node. First, the ISP's routers must be configured to handle packets based on their IP precedence level, or similar semantics expressed by the bit values defined in the IP packet header. There are three aspects to this. First, you need to consider the aspect of using the IP precedence field to determine the queuing behavior of the router, both in queuing the packet to the forwarding process and queuing the packet to the output interface. Second, consider using the IP precedence field to bias the packet discard processes by selecting the lowest precedence packets to discard first. Third, consider using any priority scheme used at Layer 2 that should be mapped to a particular IP precedence value.

Several methods have been proposed within the IETF which may yield robust mechanisms and semantics for providing these types of differential services (*diffserv*) [19].

The generic *diffserv* deployment environment assumes that the network uses ingress traffic policing, where traffic passed into the network is passed through traffic shaping profile mechanisms, which bind their average and peak data rates, and their relative priority and discard precedence in accordance with the traffic profile and the administrative agreement with the customer. These ingress filters can be configured to either discard out-of-profile packets, or the ingress filter may mark them with an elevated discard priority so that they are carried within the network only when there are adequate resources available. Within the core of the network, *WFQ* (or similar proportional scheduling algorithms) can be used to allocate network resources according to the marked priority levels, allowing the high speed and high volume switching component of the network to operate without per-flow state being imposed.

The cumulative behavior of such stateless, local-context algorithms and corresponding deployment architectures can yield the capability of distinguished and predictable service levels, and hold the promise of excellent scalability. You still can mix best-effort and "better-than-best-effort" nodes, but all nodes in the latter class should conform to the entire QoS selected profile or a compatible subset (an example of the principle is that it is better to do nothing than to do damage).

In conclusion, QoS is possible in the Internet, but it does come at a price of compromise -- there are no perfect solutions. In fact, one might suggest that expectations have not been appropriately managed, since guarantees are simply not possible in the Internet, at least not for the foreseeable future. What is possible, however, is delivering differentiated levels of best effort traffic in a manner which is predictable, fairly consistent, and which provides the ability to offer discriminated service levels to different customers and to different applications.

5.0 References

[1] IETF Internet Draft, "End-to-End Traffic Issues in IP/ATM Internetworks," draft-jagan-e2e-traf-mgmt-00.txt, S. Jagannath, S. Yin, August 1997.

[2] "MAC Bridges," ISO/IEC 10038, ANSI/IEEE Std. 802.1D, 1993.

[3] "Supplement to MAC Bridges: Traffic Class Expediting and Dynamic Multicast Filtering," IEEE P802.1p/D6, May 1997.

[4] IETF Internet Draft, "Integrated Service Mappings on IEEE 802 Networks," draft-ietf-issll-is802-svc-mapping-01.txt, M. Seaman, A. Smith, E. Crawley, November 1997.

[5] IETF Internet Draft, "SBM (Subnet Bandwidth Manager): A Proposal for Admission Control over IEEE 802-style Networks," draft-ietf-issll-is802-bm-05.txt, R. Yavatkar, F. Baker, D. Hoffman, Y. Bernet, November 1997.

[6] "Oscillating Behavior of Network Traffic: A Case Study Simulation," L. Zhang, D. Clark, Internetwork: Research and Experience, Volume 1, Number 2, John Wiley & Sons, 1990, pages 101-112.

[7] "Congestion Avoidance and Control," V. Jacobson, Computer Communication Review, vol. 18, no. 4, pp. 314-329, Aug. 1988.

[8] "Random Early Detection Gateways for Congestion Avoidance," S. Floyd, V. Jacobson, IEEE/ACM Transactions on Networking, v.1, n.4, August 1993.

[9] "Understanding TCP Dynamics in an Integrated Services Internet," W. Feng, D. Kandlur, D. Saha, K. Shin, NOSSDAV '97, May 1997.

[10] "Queuing Systems, Volume 2: Computer Applications," L. Kleinrock, Wiley Interscience, 1975.

[11] "An Engineering Approach to Computer Networking," S. Keshav, Addison-Wesley, 1997.

[12] "Design and Analysis of a Fair Queuing Algorithm," A. Demera, S. Keshav, and S. Shenker, ACM SIGCOMM'89, Austin, September 1989.

[13] "Dynamics of Random Early Detection," D. Lin and R. Morris (Harvard University), a proposal & overview of Fair Random Early Drop (FRED). Presented at ACM SIGCOMM 1997.

[14] RFC1633, "Integrated Services in the Internet Architecture: An Overview," R. Braden, D. Clark, S.

Shenker, June 1994.

[15] RFC2205, "Resource ReSerVation Protocol (RSVP) - Version 1 Functional Specification," R. Braden, L. Zhang, S. Berson, S. Herzog, S. Jamin, September 1997.

[16] RFC2208, "Resource ReSerVation Protocol (RSVP) Version 1 Applicability Statement, Some Guidelines on Deployment," A. Mankin, F. Baker, R. Braden, S. Bradner, M. O'Dell, A. Romanow, A. Weinrib, L. Zhang, September 1997.

[17] The IETF RSVP Admission Policy (rap) working group charter can be found at: http://www.ietf.org/html.charters/rap-charter.html

[18] Internet Research Task Force draft, "Recommendations on Queue Management and Congestion Avoidance in the Internet," draft-irtf-e2e-queue-mgt-recs.ps, R. Braden, D. Clark, J. Crowcroft, B. Davie, D. Estrin, S. Floyd, V. Jacobson, G. Minshall, C. Partridge, L. Peterson, K. Ramakrishnan, S. Shenker, J. Wroclawski, L. Zhang, March 1997.

[19] Differential Service for the Internet, http://diffserv.lcs.mit.edu/

Acknowledgments

Portions of text in this paper have been extracted from *Quality of Service: Delivering QoS on the Internet and in Corporate Networks*, by Paul Ferguson and Geoff Huston, published by John Wiley & Sons, January 1998, ISBN 0-471-24358-2.