OverQoS: An Overlay based Architecture for Enhancing Internet QoS

Lakshminarayanan Subramanian[*] Ion Stoica[*] Hari Balakrishnan[+] Randy H. Katz[*]

[*]*University of California at Berkeley* {lakme,istoica,randy}@cs.berkeley.edu

[+]*Massachusetts Institute of Technology* hari@nms.lcs.mit.edu

Abstract

This paper describes the design, implementation, and experimental evaluation of *OverQoS*, an overlay-based architecture for enhancing the best-effort service of today's Internet. Using a *Controlled loss virtual link (CLVL)* abstraction to bound the loss rate observed by a traffic aggregate, OverQoS can provide a variety of services including: (a) smoothing packet losses; (b) prioritizing packets within an aggregate; (c) statistical loss and bandwidth guarantees.

We demonstrate the usefulness of OverQoS using two sample applications. First, *RealServer* can use OverQoS to improve the signal quality of multimedia streams by protecting more important packets at the expense of less important ones. Second, *Counterstrike*, a popular multi-player game, can use OverQoS to avoid frame drops and prevent end-hosts from getting disconnected in the presence of loss rates as high as 10%. Using a wide-area overlay testbed of 19 hosts, we show that: (a) OverQoS can simultaneously provide statistical loss guarantees of 0.1% coupled with statistcal bandwidth guarantees ranging from 100 Kbps to 2 Mbps across international links and broadband end-hosts; (b) OverQoS incurs a low bandwidth overhead (typically less than 5%) to achieve the target loss rate, and (c) the increase in the end-to-end delay is bounded by the round-trip-time along the overlay path.

1 Introduction

Over the past decade, there have been many efforts to provide QoS in the Internet. Most notably, the Intserv and Diffserv service architectures have been proposed to offer a large array of services ranging from per flow and delay guarantees to per aggregate guarantees and priority services. Despite these efforts, today's Internet still continues to provide only a best-effort service. One of the main reasons is the requirement of these proposals that all network elements between a source and a destination implement QoS mechanisms. The inherent difficulty in changing the IP infrastructure coupled with the natural lack of incentives for ISPs to coordinate their deployment has rendered this requirement infeasible, and ultimately hurt the adoption of IntServ and DiffServ.

In this paper, rather than trying to achieve traditional QoS guarantees such as the ones offered by Intserv and Diffserv, we ask the following question: *are there any meaningful QoS enhancements that can be provided in the Internet without requiring support from the IP routers*? To answer this question we turn our attention to overlay networks as an alternative for introducing new functionality that is either too cumbersome to deploy in the underlying IP infrastructure, or that requires information that is hard to obtain at the IP level. Examples of successful overlay networks include application-layer multicast [12, 21], Web content distribution networks, and resilient overlay networks (RONs) [7].

To this end, we propose *OverQoS*, an overlay based QoS architecture for enhancing Internet QoS. The key building block of OverQoS is the *controlled-loss virtual link* (CLVL) abstraction. CLVL provides statistical loss guarantees to a traffic aggregate between two overlay nodes in the face of varying network conditions. In addition, it enables overlay nodes to control the bandwidth and loss allocations among the individual flows within a CLVL. While OverQoS cannot provide the spectrum of service guarantees offered by IntServ [10], it can still provide useful QoS enhancements to applications. Examples of such enhancements are:

Smoothing losses: Bursty network losses can have a negative impact on many applications such as multi-player games. OverQoS can reduce or even eliminate the loss bursts by smoothing packet losses across time.

Packet prioritization: OverQoS can allow applications to express the importance of the packets within a stream, and protect important packets at the expense of less important ones. For example, OverQoS can protect I-frames in an MPEG stream over B-frames or P-frames.

Statistical Bandwidth and Loss Guarantees: Besides statistical loss guarantees, OverQoS can provide statistical bandwidth guarantees to a small fraction of its traffic.

To understand the tradeoffs and the limitations of the OverQoS architecture, we present its design and implementation, and perform an extensive evaluation. Across a wide-area testbed of 19 diverse nodes (spanning US, Europe, and Asia), we show that OverQoS can simultaneously provide statistical loss guarantees on the order of 0.1% and and bandwidth guarantees ranging from 100 Kbps to 2 Mbps. In addition, by simultaneously running multiple competing CLVLs along with long-lived TCPs on a lossy access network, we show that OverQoS is fair to cross-traffic and can co-exist with other competing OverQoS links.

We additionally demonstrate how *multiplayer games* and *streaming media* can benefit from using OverQoS. In the multi-player game example, an end-user can use OverQoS to interactively play a game like *Counterstrike* in highly lossy environments (experiencing a loss rate as high as 10%) without observing any skips or getting disconnected. In the streaming media example, we demonstrate how *RealPlayer* can use OverQoS to preferentially drop and recover specific packets to enhance the quality of a stream *without consuming any additional bandwidth*. OverQoS achieves this by simply redistributing the losses among the packets within the stream. The increase in the end-to-end delay is bounded by the end-to-end RTT.

The rest of the paper is organized as follows. In Section 2 we describe the basic OverQoS architecture and describe the construction of CLVLs in Section 3. In Section 4, we provide the details of our OverQoS implementation. In Section 5, we show two real-world applications that can benefit by using OverQoS. In Section 6, we evaluate the performance of OverQoS in the wide area Internet. We present related work in Section 7 and conclusions in Section 8.

2 OverQoS Architecture

Figure 1 illustrates an OverQoS network with overlay nodes spanning different routing domains and flows routed within this network. We make no assumptions about the placement of overlay nodes in the Internet. Rather, we assume that a placement of overlay nodes is pre-specified. In this paper, we will assume that the end-to-end path on top of an overlay network is fixed and we will attempt to enhance the QoS along this path in the presence of varying levels of network congestion. We can use existing approaches like RON [32] to determine the overlay path between a pair of end-hosts.

In the remainder of this paper, we will use the term *virtual link* to refer to the IP path between two overlay nodes and *bundle* to refer to a stream of application data packets carried across the virtual link. A bundle typically includes packets from multiple transport-layer flows across different sources and destinations. The following constraints and requirements make the design of any overlay-based QoS challenging:

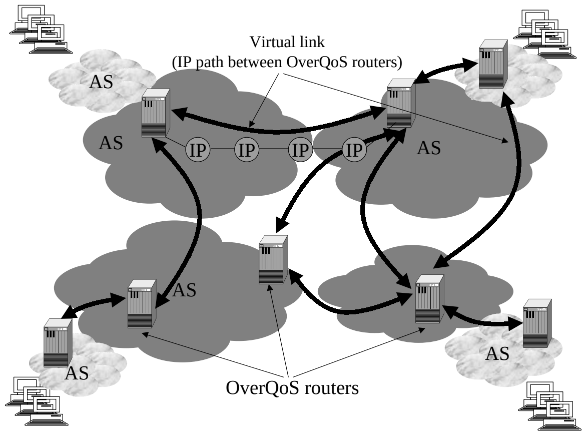


Figure 1: The OverQoS system architecture. OverQoS nodes in different AS's communicate with each other over virtual links using the underlying IP paths.

1. *Node Placement and Cross Traffic:* Overlay nodes will usually span different routing domains and will not be directly connected to the congested links. Hence, one cannot avoid losses or delays along virtual links. Additionally, the losses incurred due to cross traffic is time-varying and can be hard to predict.
2. *Fairness:* Overlays should not offer QoS at the expense of hurting cross traffic. Therefore, the overlay traffic at an aggregate level should be congestion sensitive and not use more than its fair share. One standard metric for determining fair share is based on *TCP-friendliness* [27].
3. *Stability:* Multiple overlay networks independently offering QoS with many virtual links overlapping on congested physical links in the underlying network should be able to co-exist.

To address these challenges we propose a solution that builds on two design principles:

Bundle loss control: Overlay nodes should bound the loss rate experienced by a bundle along a virtual link in the presence of time-varying cross traffic. We propose a *controlled-loss virtual link* (CLVL) abstraction to achieve this loss bound and characterize the service received by a bundle.

Resource management within a bundle: An overlay node can control the loss and bandwidth allocations of each flow and/or application within a bundle.

These design principles enable OverQoS to provide a range of useful services to Internet applications. Example of such services are: (1) packet prioritization, (2) smoothing losses (i.e., eliminate the bursts of losses by spreading losses in time), and (3) statistical bandwidth and loss guarantees, though this service can be typically offered only to a small

b	Maximum sending rate on a virtual link
c	CLVL available/ aggregate bandwidth
q	CLVL target loss rate / statistical bound on the CLVL loss-rate
r	CLVL redundancy factor
c_{min}	Minimum statistical bandwidth guarantee
u	Probability of not meeting the bandwidth guarantee c_{min}

Table 1: OverQoS Notation table

fraction of a bundle's traffic. We next elaborate on our two design principles.

2.1 Bundle Loss Control

The basic building block for enabling OverQoS to achieve loss control over a bundle is the Controlled-loss Virtual Link (CLVL) abstraction. The CLVL abstraction provides a bound, q, on the loss rate seen by the bundle over a certain period of time regardless of how the underlying network loss rate varies with time. Overlays can achieve this bound by recovering from network losses using a combination of Forward Error Correction (FEC) and packet retransmissions in the form of ARQ. By setting q to an arbitrarily low value (close to 0), a CLVL provides the notion of a near-loss free pipe across a virtual link. Therefore, a CLVL *isolates* the losses experienced by the bundle from the loss-rate variations in the underlying IP network path. The biggest challenge in constructing a CLVL is to achieve the loss bound q in the presence of time-varying cross traffic and network conditions. Additionally, the amount of bandwidth overhead should be minimized. In Section 3.2, we present a hybrid FEC/ARQ solution which minimizes the amount of redundancy required to provide a CLVL abstraction for a given value of q.

The total traffic between two overlay nodes consists of: (a) the traffic of the bundle; (b) the redundancy traffic required to achieve the target loss rate, q. The fairness and stability constraints limits the maximum rate (inclusive of the redundancy traffic) at which OverQoS can transmit across a virtual link. Let $b(t)$ denote this traffic bound at time t (Section 3.1 elaborates on how b is computed). Let $r(t)$ denote the fraction of redundancy traffic required by OverQoS to achieve q. Then, the *available bandwidth* for the flows in the bundle is $c(t) = b(t) \times (1 - r(t))$. Thus, the service provided by a CLVL to the bundle is: *As long as the arrival rate of the bundle at the entry node does not exceed $c(t)$, the packet loss rate across the virtual link will not exceed q, with high probability.*

2.2 Resource Management within a Bundle

The CLVL abstraction provides the bundle an available bandwidth, c, which varies with time and guarantees the entire bundle a target loss rate, q. If the traffic arrival rate of the bundle is larger than c, the extra traffic is dropped at the entry overlay node. The overlay node can employ any QoS scheduling discipline to distribute c and the losses across the flows in the bundle. In particular, in a Diffserv-like model, if every packet is associated with a priority, then the overlay node can use these priorities to preferentially drop packets and allocate bandwidth to different flows.

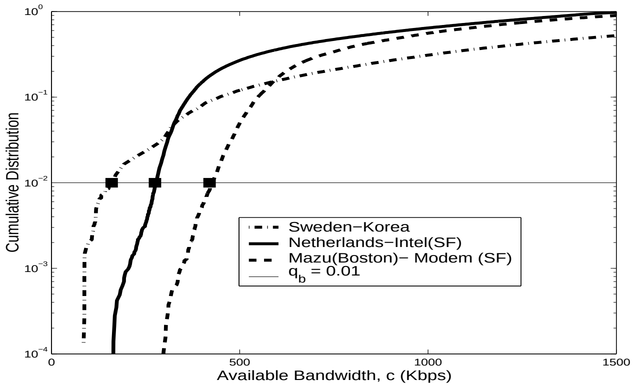


Figure 2: The cumulative distribution of c across three separate CLVLs is measured on Jan 20, 2003 by transmitting 1,500,000 packets over each virtual link(each with 250 bytes payload). The intersection point between $u = 0.01$ and the CDF curves represent the values of c_{min} along the three links.

While in general the available bandwidth, c, of a CLVL bundle varies with time, it might be possible to statistically bound the minimum bandwidth of the bundle to offer bandwidth guarantees to a fraction of OverQoS traffic. Given a small probability value, u, one can capture the variations of the available bandwidth on a CLVL using a distribution and determine a value c_{min} such that the probability, $P(c < c_{min}) = u$ where u represents the probability of not meeting the bandwidth guarantee, c_{min}. If the corresponding c_{min} is a significant fraction of c, then OverQoS can provide statistical bandwidth guarantees by allocating bandwidth to flows within a CLVL as long as the total allocated bandwidth is less than c_{min}. Table 1 tabulates all the variables we use in expressing the properties of a CLVL.

In practice, we notice that the value of c_{min} across overlay links can be reasonably high implying that OverQoS can indeed be used to provide meaningful statistical bandwidth guarantees to applications. Figure 2 shows the distribution of c for three different overlay links traversing international links and broadband networks: Lulea (Sweden)-Korea, Mazu (Boston)- Cable Modem (SF), Netherlands-Intel (SF). The values of c_{min} across these links to provide a $u = 0.01$ guarantee are 160 Kbps, 420 Kbps, and 269 Kbps respectively. Statistical bandwidth guarantees can be provided only to a subset of the OverQoS flows, potentially at the expense of other flows. Flows requiring guarantees should be given a higher priority over other flows at an

OverQoS node. The remaining bandwidth $c - c_{min}$ is distributed among the other flows.

2.3 Overall picture

An OverQoS network (Figure 1) comprises of a collection of overlay links where each link is associated with a CLVL abstraction. Individual CLVLs along different OverQoS links are stitched together to generate an end-to-end path along which a flow may be routed and guaranteed a specific amount of QoS. In this paper, we demonstrate that an overlay network can indeed be useful in enhancing Internet but do not address the issue of how to route flows on top of an OverQoS network. We rely on an overlay routing service like RON [32] to specify an end-to-end path across an OverQoS network. Given one such path, OverQoS determines the level of QoS that can be provided along the path.

Application-OverQoS Interface: A legacy application intending to use OverQoS is required to perform two functionalities. First, it needs to tunnel its packets through the overlay network using an OverQoS proxy. The proxy node functionality can reside either at the first OverQoS node along the path or within the same host as the application. Second, the proxy is responsible for signaling the application specific requirements to OverQoS. For example, if OverQoS offers the service of smoothing losses or packet prioritization, the proxy is required to mark the priority of packets within the flows. Our current implementation of an OverQoS proxy is application specific in that it infers the priorities of the packets of an application flow without modifying the application. However, in the case of statistical loss or bandwidth guarantees, an application is required to clearly signal its QoS requirements (loss,bandwidth) to the OverQoS proxy. For this particular service, the proxy is additionally responsible for undergoing an admission control test to determine whether OverQoS can indeed satisfy the application's QoS requirements. The signaling aspects of the admission control as well as the issue of how to route flows within OverQoS are out of the scope of this paper.

2.4 Discussion

End-to-end Recovery vs Overlay CLVL: An alternative to applying the CLVL abstraction on an overlay network is to apply loss control on an end-to-end per flow basis. There are several arguments against end-to-end loss control: First, using FEC to apply end-to-end loss control is far more expensive than applying it on an aggregate level. For example, in order to provide a 0.1% loss guarantee to a 64 Kbps stream (like game console traffic or IP telephony stream) over a bursty channel with an average loss rate of say 2%, the minimum amount of FEC required can be as high as 32 Kbps. However, if 10 such flows are aggregated at an overlay node, the per-flow FEC requirement can drop to lower than 5 Kbps. Second, with a better distribution of overlay nodes, we expect the overlay links to have much smaller RTTs than end-to-end RTTs. Hence, overlay-level recovery using ARQ has better delay characteristics than end-to-end recovery. Finally, aggregation of flows within an overlay provides the ability to trade resources across different flows (or within packets of the same flow) which is fundamentally necessary to provide better QoS properties.

Delay guarantees: Overlay networks have no control over variations in queuing delays along virtual links and hence cannot offer delay assurances. On the other hand, overlay networks have been used to route around congestion [33, 7]. Such techniques can be embedded into an overlay to improve the end-to-end delay characteristics of a path.

Over-provisioning: Recent measurement studies have shown that Internet backbones are over-provisioned and have low levels of congestion [19, 17]. This questions the basic need for Internet QoS. We contend that over-provisioning is not necessarily a permanent feature of the Internet, but a reflection of the big disparity between the poor connectivity at edges, and the backbone capacity. As more homes and enterprises become connected over faster, multi-megabit/s or higher, links with optical fibers, we expect that at least some parts of the Internet such as small ISPs to become more congested. This trend is already evident in countries like Japan where ISPs offer 10 Mbps broadband connections to homes [4]. In addition, many ISPs already provide aggregate QoS within their networks using MPLS technologies [26]. We believe that overlays are the right platform for translating these aggregate intra-domain QoS to meaningful end-to-end QoS guarantees.

3 Controlled-Loss Virtual Link (CLVL)

In this section, we describe the realization of the CLVL abstraction. In particular, we describe: (a) how to compute, b, the maximum sending rate across an OverQoS link; (b) how to achieve the target loss rate q for the flows in the bundle; (c) the architecture of the OverQoS node.

3.1 Estimating b

OverQoS tunes the maximum output rate, b, depending on network congestion in order to be both fair to cross traffic as well as achieve stability in the presence of other competing OverQoS traffic. One way of achieving this is to set b based on an N*-TCP pipe* abstraction which provides a bandwidth which is N times the throughput of a single TCP connection on the virtual link. We set N to be equal to the number of flows in the bundle.

We use MulTCP [29] to emulate the behavior of N TCP connections. MulTCP uses a TCP-like congestion control mechanism with $\alpha = N/2$ and $\beta = \frac{1}{2N}$ as the increment and decrement parameters. While MulTCP may react quickly to congestion, it may not provide smooth variations in the sending rate. To obtain smoother variations, we may prefer to choose an alternate operating point with a lesser value of α and β without altering the net steady state throughput as determined by the TCP equation [27]. If we set $\alpha = \sqrt{N}$, the corresponding value of β can be calculated using the TCP equation as equal to $4/(3 \times N^{1.5} + 2)$. Across most of our evaluations, we use the standard parameters of MulTCP. Alternatively, we can also use an equation-based approach to emulate the behavior of N TFRC connections [16].

3.2 Achieving target loss rate q

We will describe a hybrid solution which uses a combination of FEC and ARQ to construct a CLVL. Recall that a CLVL abstraction aims to bound the bundle loss rate to a small value q. Since burstiness of cross-traffic is usually unpredictable, we define q as a statistical bound on the average loss rate observed over some larger period of time (on the order of seconds).

FEC vs ARQ trade-off: The main distinction between FEC and ARQ is in the trade-off between bandwidth overhead and packet recovery time. While FEC can help in quickly recovering from packet losses, the bandwidth overhead can be high especially over virtual links experiencing bursty losses [22]. On the other hand, an ARQ based solution will have a high packet recovery time if the RTT between two overlay nodes is large. To strike a balance between these two approaches, we present a hybrid approach that uses the best features of both these mechanisms.

We will first briefly describe how one will construct a CLVL using purely ARQ or FEC and combine these approaches to obtain a hybrid CLVL construction.

ARQ-based CLVL: A purely ARQ-based solution for building CLVLs is easy to construct. In a reliable transmission ($q = 0$), a packet is repeatedly retransmitted until the sender receives an acknowledgment from the receiver. Similarly, to achieve a non-zero target loss rate, q, it is enough to retransmit any lost packet $L = \log_{\bar{p}} q - 1$ times, where $\bar{p}$ represents the average loss rate over the interval over which we want to bound q. However, if $L > 1$, a pure-ARQ based CLVL is unattractive since it uses multiple RTTs to achieve the bound q.

FEC-based CLVL: In an FEC-based approach, we divide time into windows of period, T_r, where a window is a unit of encoding/decoding. We consider an *erasure code* such as Reed-Solomon, characterized by (n, k), where k is the number of packets arriving at the entry node during the window, and $(n - k)$ represents the number of redundant packets added. Let r denote the redundancy factor, where $r = (n - k)/n$. The FEC problem reduces then to determining a minimum redundancy factor, r, such that the target loss rate q is achieved. Since the hybrid approach (*i.e.,* FEC+ARQ based CLVLs) presented below outperforms the FEC based CLVLs in most of the cases, we skip the description of our algorithm for computing the ideal value of r.

FEC+ARQ based CLVL: Due to delay constraints for loss recovery, we restrict the number of retransmissions to at most one. We divide packets into windows and add an FEC redundancy factor of r_1 for each window in the first round. In the second round, if a window is non-recoverable, the entry node retransmits the lost packets with a redundancy factor r_2.

We need to estimate the parameters, r_1 and r_2. Let $f(p)$ denote the PDF of the loss rate p, where each value of p is measured over an encoding/decoding window. FEC offers loss protection within a window if the fraction of packets lost in a window, p, is less than the amount of redundancy added for that window. Given a redundancy factor, r, the expected packet loss rate after recovering from FEC is given by:

$$G(r) = \int_r^1 p f(p) dp$$

Hence the expected packet loss rate after the two rounds in the hybrid approach is equal to $L(r_1, r_2) = G(r_1) \times G(r_2)$. Given a target loss-rate, q, we require:

$$L(r_1, r_2) \le q$$

For a given window, r_1 is the FEC overhead in the first round, $G(r_1)$ is the expected number of retransmitted packets and $G(r_1) \times r_2$, the expected overhead in the second round. The expected bandwidth overhead is given by

$$O(r_1, r_2) = r_1 + G(r_1)(1 + r_2)$$

This yields the following optimization problem: Given a target loss rate q, determine the redundancy factors r_1 and r_2 that minimizes the expected overhead, $O(r_1, r_2)$ subject to the target loss constraint: $L(r_1, r_2) \le q$.

For many loss distributions that occur in practice, the optimal solution for this problem is when $r_1 = 0$. This solution implies that it is better not to use FEC in the first round, and use FEC only to protect retransmitted packets. When $r_1 = 0$ and $r_2 = 0$, an FEC+ARQ CLVL reduces to a pure ARQ based CLVL. This happens when $q < p_{avg}^2$ where $p_{avg} = G(0)$ is the average loss-rate along the virtual link. An FEC+ARQ CLVL can be made adaptive to sudden variations in the loss characteristics by always applying a minimal amount of FEC ($r_2 > 0$), to the retransmitted packets in a window.

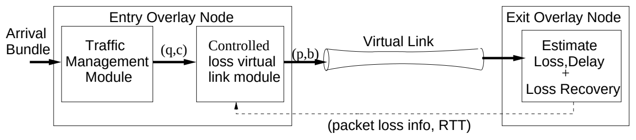

Arrival Bundle
Entry Overlay Node
Traffic Management Module
(q,c)
Controlled loss virtual link module
(p,b)
Virtual Link
Exit Overlay Node
Estimate Loss,Delay + Loss Recovery
(packet loss info, RTT)

Figure 3: Components of entry and exit OverQoS nodes

We made a simplistic assumption in the above calculation. We used the same distribution $f(p)$ to model the fraction of losses during both the first and second round. Since the number of packets in a retransmitted window may be much smaller than the original window, the same distribution $f(p)$ may not apply. To overcome this problem, we estimate a table of loss distributions (rather than one $f(p)$) across different time-scales and apply the appropriate distribution based on the number of retransmitted packets.

3.3 Node Architecture

Figure 3 captures the interactions between the various components in the entry and exit overlay nodes. The entry node consists of two modules: one that implements the CLVL abstraction, and another that performs per-aggregate or per-flow traffic management. The first module communicates with the exit OverQoS node to estimate the link loss rate and delay. It uses this information to adapt the data traffic to conform to the CLVL abstraction. The second module allocates the capacity of the CLVL among competing traffic aggregates or flows. The exit OverQoS node is responsible for measuring the loss and delay characteristics and reconstructing lost packets if necessary. If the CLVL abstraction uses ARQ for loss recovery, the exit node propagates individual packet loss information to the entry node.

The entry node exerts control on the traffic in the bundle at two levels of granularity: on the bundle as a whole, and on a per-flow basis within the bundle. At both these levels, the entry node can control either the sending rate or the loss rate. The CLVL management module at the entry node first determines the sending rate of the bundle, b, using MulTCP [29] to emulate the aggregate behavior of N virtual TCPs. Next, it determines the level of redundancy r required to achieve a certain target loss-rate q based on the loss-characteristics determined by the window. The resulting available bandwidth c is estimated to be $b(1-r)$. The traffic management module at the entry node then distributes the available bandwidth c among the individual flows. If the net input traffic is larger than c, the entry node drops the extra traffic and exercises control in distributing the losses amongst the flows.

4 OverQoS Implementation

We implemented the OverQoS node architecture in about 5000 lines of C code. The communication between overlay nodes uses the UDP socket interface. For loss recovery, we use the FEC software library built by Rizzo *et al.* [30]. Our implementation works on both Linux and FreeBSD platforms.

Figure 4 illustrates the structure of a single OverQoS node along a given path. An OverQoS node listens on a UDP socket for the arrival bundle and tunnels the traffic to the exit node as a UDP stream. The CLVL Encoder and Decoder modules implement the CLVL abstraction on top of the overlay link by adding the necessary level of redundancy to recover from packet losses.The decoder also provides loss feedback to the encoder for computing the optimal redundancy factor. The Traffic Management module implements per-flow or per-packet resource management. Different QoS schedulers and buffer management schemes like priority scheduling and smoothing losses is performed by this module. The rate estimator computes the CLVL parameters b,c and r while the link estimator provides feedback to the transmitting OverQoS node about the virtual link characteristics comprising: (a) loss feedback for computing the loss distribution; (b) RTT, the round trip time.

CLVLs along an overlay path can be stitched together (or *cascaded*) to provide end-to-end services. Cascaded CLVLs can introduce artificial losses at an overlay node if the available bandwidth on the incoming links is larger than the available bandwidth in the outgoing links. In order to avoid any artificial packet losses at an intermediary node in an overlay path, an OverQoS node uses b_{max} to signal the maximum sending rate to its predecessor. This is illustrated in Figure 4.

4.1 Other Implementation Issues

We will now briefly discuss some of the salient implementation issues:

Application-dependent proxy: An important aspect of interfacing with legacy applications is to use an application proxy that can signal an application's requirements to the OverQoS network. In the case of MPEG streaming, the application proxy interprets the packets in the stream and marks the priority of recovery for each packet. For smoothing losses, all packets in a stream are associated with the same priority. For obtaining bandwidth guarantees, the proxy needs to use a signaling mechanism like RSVP [10] to reserve the resources along an overlay path.

Choosing parameters: The parameters N, RTT and p_{avg} need to be estimated for determining the sending rate b. While N can be estimated as the instantaneous number of flows, we set N as the average number of flows observed over a larger period of time (certain flows have a very short lifetime). This is to reduce the variations in the sending rate induced by N. Only flows that generate a minimum

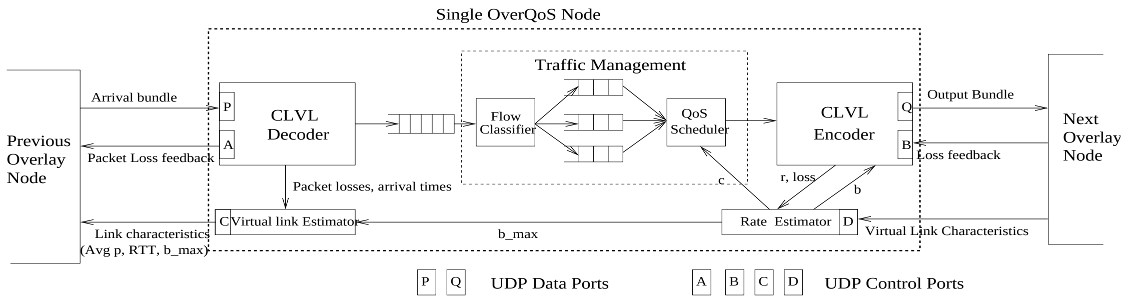


Figure 4: Structure of a single OverQoS node along a path.

number of packets, are used in calculating N. We leverage the techniques used in equation based congestion control [16] for estimating the RTT and p_{avg} between two OverQoS nodes. We choose a reasonably low value of the target loss-rate, $q = 0.1\%$, for most of our experiments. For FEC+ARQ based CLVLs, we choose the packet recovery time, T_r to be $2 \times RTT$.

Startup phase: During periods of no usage (i.e. when N=0), we do not send additional traffic to estimate the virtual link parameters. After such a phase, OverQoS nodes need to determine an initial value of b along a virtual link. Like TCP, we use a *slow-start* phase to estimate the initial value of b. During the slow-start phase, OverQoS does not use loss recovery.

FEC implementation: Our implementation can perform FEC encoding and decoding (for a redundancy factor as high as 50%) at over 300 Mbps on a Pentium III 866 MHz running Linux 2.4.18 kernel. Since we operate on small window sizes, ($n < 1000$), Reed Solomon coding is not a bottleneck. For example, on a virtual link with an $RTT = 100$ ms, the window size is bounded by 1000 for sending rates less than 40 Mbps. Other coding techniques like Tornado codes [23] while faster, may not provide the same level of error correction for small window sizes.

5 Two Sample Applications

In this section, we will describe two real applications that can leverage the QoS enhancements offered by OverQoS. The first application shows how *RealServer*, a streaming media application can improve the signal quality of multimedia streams by using OverQoS to preferentially recover important packets at the expense of less important ones *without using any additional network bandwidth*. The second application is *Counterstrike*, a popular online multiplayer game with a user base of over 1 million players [1]. For this application, we show how OverQoS can smooth out losses and enable players to play the game under high-loss environments.

5.1 Streaming Media Applications

Streaming media applications are typically more sensitive to network losses than delay since delay variations can be masked by using a buffer at the client. OverQoS is an ideal platform for providing different forms of enhancements for such applications. Two such forms of enhancements are:

1. The quality of streaming audio can be enhanced by converting bursty losses into smooth losses.
2. By preferentially recovering packets in an MPEG stream, one can improve the quality of the video stream.

Given that delay variations is not a primary issue for these applications, OverQoS primarily uses an ARQ-based CLVL for these applications. For both streaming audio and video, OverQoS does *not* consume any additional bandwidth. It achieves this by performing the following operation: Whenever an important packet is lost in the network, OverQoS retransmits this packet and drops a later lesser important packet to compensate for the retransmission. In the process, the application observes the same end-to-end loss-rate as it would in the normal Internet and will experience an occasional increase in the end-to-end delay which is bounded by the RTT along the overlay path.

5.1.1 QoS Enhancements for Streaming Media

Streaming Audio: Bursty errors in a streaming audio application can either cause interruptions to an audio stream or cause gaps in an audio stream for periods of time easily perceptible by the human ear. We consider the case where a RealServer streams a *.wav/.mp3* audio file to an end-host using RTP. The audio stream can use OverQoS to smooth out bursty losses *i.e.*, spread a bursty loss over time.

MPEG Streaming: An MPEG video stream consists of a Group of Pictures (GOP) each comprising of I-frames, P-frames and B-frames [6]. Among these, I-frames are the most important since they represent the start of a video sequence in a GOP while P-frames and B-frames are inter-coded frames. Each frame is typically larger than a packet and a frame is sent across multiple consecutive packets. All packets corresponding to an I-frame occur in succession. A single bursty network loss can eliminate an I-frame completely which can cause an MPEG player like Mplayer [5] to disconnect since a GOP cannot be reconstructed. The B-frame and P-frame of a GOP are useless without the corresponding I-frame.

Using OverQoS, one can associate packets belonging to I-frames with higher priority and recover packets within an I-frame at the expense of B or P-frame packets. Additionally, bursty dropping of B and P frame packets affects the quality of one GOP in an MPEG stream, smoothly dropping B and P packets can affect the quality of multiple GOPs. The type of frame of a packet is embedded in the MPEG Video-Specific header within the payload of a packet.

5.1.2 Evaluation

Network Setup: We use the Helix server version 9.0.2 [2] as our streaming media server and use Mplayer [5] as the streaming media client. All streaming media requests are issued using the Real Time Streaming Protocol (RTSP) to stream packets using UDP. We built a client proxy and a server proxy to interpret the streaming media packets and associate them with different priorities. Using these proxies, we tunnel a media stream from RealServer to an Mplayer client along an overlay path along which we replay sample bursty loss traces collected along different overlay links. For the purpose of illustration, we consider two such loss traces: (a) Mazu (Boston)-Korea with an average loss rate of 2%; (b) Intel (San Francisco) - Lulea (Sweden) with an average loss trace of 3%. Each trace is 20 minutes long. To emulate the behavior without OverQoS, we consider the OverQoS nodes to act as packet forwarders. If the length of a media stream is shorter than the length of the trace, we repeat the analysis for different portions of the trace.

Streaming Audio: To demonstrate the effect of smooth dropping on streaming audio, we concatenated several speech samples provided by International Telecommunication Union (ITU-T) to produce two test samples of length 84 sec and 82 sec respectively. Perceptual Evaluation of Speech Quality(PESQ) [3] is one metric to evaluate the quality of voice. We measured the PESQ score for the received stream in comparison to the original stream. A PESQ score of 5 is considered to be ideal implying that the received audio stream has not degraded in quality.[1]

Table 2 compares the PESQ scores of streaming audio with and without OverQoS for two benchmark speech samples. We observe that smoothing the losses does help in increasing the quality of the audio stream. Using OverQoS we are able to increase the PESQ score of the output stream by roughly $0.15 - 0.2$. To demonstrate that $0.15 - 0.2$ is indeed a reasonable improvement in the audio quality, we experimented with several artificial bursty loss patterns while maintaining the same average loss-rate of the traces (*i.e.,* 2% and 3%) and measured the PESQ scores for each of them. For an average loss rate of 2%, we found the PESQ scores to vary between 4.2 and 4.3 across a variety of bursty loss patterns. For these cases, we again found that smooth dropping performs better than bursty drops. Hence, we find that smoothing losses using OverQoS uniformly outperforms different types of bursty network losses.

[1]The PESQ measure is applicable only for pure speech samples and not for arbitrary audio streams. Hence our analysis is limited to only these standardized samples.

		Sample 1	Sample 2
Mazu-Korea	Without OverQoS	4.25 ± 0.3	4.27 ± 0.5
Mazu-Korea	With OverQoS	4.46 ± 0.4	4.45 ± 0.3
Intel-Lulea	Without OverQoS	4.04 ± 0.2	4.13 ± 0.3
Intel-Lulea	With OverQoS	4.19 ± 0.3	4.31 ± 0.3

Table 2: PESQ scores for speech samples with and without OverQoS for both the Mazu-Korea and Intel-Lulea loss traces. This table also shows the standard deviation of these scores across different loss traces.

		5% PSNR	Median PSNR
Mazu-Korea	Without OverQoS	15.27	22.33
Mazu-Korea	Using OverQoS	17.4	24.95
Intel-Lulea	Without OverQoS	14.68	21.59
Intel-Lulea	Using OverQoS	16.21	24.7

Table 3: This table shows the 5% and median values from the PSNR distribution of the received stream. 5% value indicates the minimum PSNR value observed by 95% of the images in the stream.

MPEG streaming: Peak Signal-to-Noise ratio (PSNR) is a standard metric used to measure the quality of the video images in a stream. Given an MPEG stream received at the client, we use the "-yuv4mpeg" utility in Mplayer to convert the stream into a stream of images. For every image, we compute the PSNR value of the received image in comparison to the video image in the original MPEG stream. We quantify the quality of the received MPEG stream using a distribution of PSNR values for the individual images. We consider a sample MPEG-1 stream which is 37 seconds for this analysis.

Table 3 compares the 5% and median values of the PSNR values of the received MPEG stream with and without OverQoS across both the loss samples. We make the following observations. First, in the case when an entire I-frame was lost due to a burst, Mplayer stopped playing the video stream since an entire GOP cannot be reconstructed. This occurred in both the loss traces when a burst coincided with the packets of an I-frame. However, OverQoS was able to recover from the burst so that the stream could progress. Second, OverQoS is able to improve both the 5% and the median PSNR values of the stream by preferentially dropping B and P packets in a burst when compared to the quality of the stream without OverQoS. We illustrate the 5% PSNR value mainly to show that OverQoS not only improves the quality of the stream in the average case but also the minimum quality of a stream. To summarize, OverQoS can improve the quality of a media stream without consuming any additional network resources.

5.2 Counterstrike application

Counterstrike is a team-based multi-player game where online players are grouped into competing teams where each team is assigned a specific goal. The environment of the game is pre-loaded and clients exchange game state over the network using small UDP packets. Bursty losses can have an adverse effect on the progress of this game. First, during the initiation phase, the client generates important control packets which if lost can render the client unable to connect to the server. Second, a burst of packet losses during the middle of a game can either cause skips or cause a player to get disconnected. A skip can arise because the game state messages received immediately after a congestion provides a context jump in the game. Third, in a multi-player game, problems observed by one player will affect other players in the game. For example, disconnection of a single player can sometimes halt the progress of a game.

OverQoS can alleviate the problem of bursty losses by performing the following operations:

1. Recover from bursty network losses by using an FEC+ARQ based CLVL abstraction between overlay links along the path.
2. Smoothly drop data packets equivalent to the size of the burst at the overlay node.
3. Identify control packets based on packet size and not drop these packets.

By both recovering lost packets as well as smoothly dropping an equivalent amount of data packets at an overlay node, OverQoS achieves three objectives: (a) OverQoS provides the Counterstrike client with critical updates to continue the progress of the game. For example, many UDP packets generated by Counterstrike merely contain the coordinates of different players. If OverQoS can deliver even a fraction of these packets, the client will still be able to reconstruct the movement and position of other players. (b) OverQoS uses minimal amount of additional bandwidth to support this application. The additional bandwidth which is not compensated by OverQoS is the FEC portion of the redundant traffic. Our wide-area experiments over realistic overlay links show that this additional bandwidth is negligible (refer to Section 6). (c) The application observes the same loss-rate as it would in the normal Internet yet not experience any skips in a game. In the event of a bursty loss, the application experiences an additional delay equal to the loss recovery time of a CLVL. With a reasonable distribution of overlay nodes, we expect this recovery time to be much smaller than end-to-end recovery.

Figure 5: Snapshot from a Counterstrike game at a 10% loss rate.

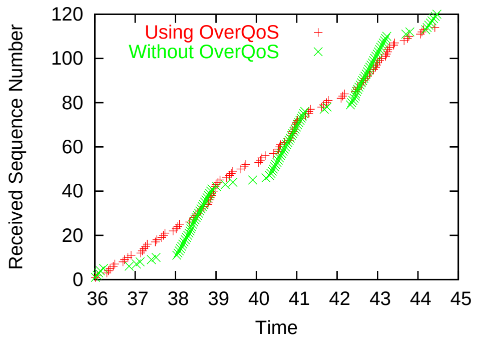


Figure 6: Sequence number plot illustrating smoothing of packet losses using OverQoS.

Counterstrike Proxy: By reverse engineering the traffic characteristics of Counterstrike, we built a client and server proxy to interpret the Counterstrike packets. We chose a proxy-based implementation for two reasons: First, Counterstrike client and server codes are proprietary and we do not modify the code. Second, it is a simple way of capturing different application specific traffic and tunneling them through OverQoS.

Example Scenario: We consider a cable modem loss trace with an high loss-rate of 10% and compare the effect of losses on the Counterstrike game under two scenarios: (a) with OverQoS; (b) without OverQoS. Figure 5 illustrates a snapshot of a Counterstrike game where OverQoS converts bursty losses into smooth losses and the client does not observe any skips. Figure 6 better illustrates the smoothing of losses using OverQoS. In the case without OverQoS, we observed many short periods of time where the network losses was as high as $70 - 80\%$ followed by periods with no congestion. The OverQoS node compensates the additional bandwidth consumed for loss recovery by smoothly dropping packets during non-lossy periods.

We make two additional observations. First, smoothing losses works well only when the bursty loss-periods are relatively short by compensating. When burst periods last for longer periods of time, OverQoS will not be able to smoothly drop packets in the absence of any non-lossy periods. Second, in this scenario, the CLVL abstraction is unable to achieve the target loss-rate due to congestion periods with very high loss-rates. However, the loss reduction provided by OverQoS during bursty periods is sufficient for the Counterstrike game to progress.

6 Evaluation

In this section, we answer several questions relating to the practical viability of OverQoS in the wide area Internet using implementation results and measurements on a wide-area network comprising of 19 diverse nodes. Additionally we use ns-2 based simulations [25] to answer specific questions that a wide area evaluation may not be able to address. The specific questions we address are:

1. Can OverQoS provide statistical bandwidth guarantees and loss assurances to flows? In particular:
   (a) *Loss Guarantees:* When can a CLVL abstraction provide loss guarantees along a virtual link?
   (b) *Bandwidth Guarantees:* What bandwidth guarantees are realizable on a virtual link?
   (c) *OverQoS Cost:* What is the bandwidth overhead and delay cost of using OverQoS?
2. *Fairness/Stability:* Is OverQoS fair to cross traffic and stable in the presence of multiple competing OverQoS networks?

6.1 Evaluation Methodology

Our evaluation methodology is two-fold: (1) we use wide area experiments to evaluate how OverQoS performs in practice, and (2) we use simulations to get a better understanding of the OverQoS performance over a wider range of network conditions.

Wide-Area Evaluation Testbed: Using resources available in two large wide-area test-beds namely RON [32] and PlanetLab [28], we construct a network of 19 nodes in *diverse* locations: 6 university nodes in Europe, 1 site in Korea, 1 in Canada, 3 company nodes, 8 behind access networks (Cable, DSL). Our main goal in choosing these nodes is to test OverQoS across wide-area links which we believe are lossy. For this reason, we avoided nodes at US universities connected to Internet2 which are known to have very few losses [7].

Simulation Environment: We built all the functionalities of our OverQoS architecture on top of the *ns-2* simulator

Background Traffi c	Average Loss(%)	FEC+ARQ Achieved Loss (%)
100 TCPs(SACK)	1.84	0.06 %
9 Mbps Self Similar	1.91	0.08%
400 Web sessions	0.68	0.03 %

Figure 7: Simulations: Achieved loss rate by a CLVL across three types of background traffic. We set $q = 0.1\%$ and the bottleneck link is 10 Mbps using RED queue.

version 2.1b8. Unless otherwise specified, most of our simulations use a simple topology consisting of a single congested link of 10 Mbps where we vary the background traffic to realize different types of traffic loss patterns. We use three commonly used bursty traffic models as background traffic: (a) long lived TCP connections; (b) Self similar traffic [36]; (c) Web traffic [15]. In addition, we use publicly available loss traces to test the performance of a CLVL.

6.2 Statistical Loss Guarantees

In this section, we answer the question: Under what network conditions, can OverQoS achieve a CLVL abstraction across an overlay link? For all the scenarios described in the section, we choose a target loss-rate to be a small value 0.1%, *i.e.,*, $q = 0.001$. To compute the available bandwidth, b, we use N-TCP with a value of $N = 10$.

Simulations: We first test whether the FEC+ARQ CLVL construction can achieve the target loss-rate across a variety of bursty loss models. Our key conclusion from the simulations is that in *all* cases, we meet the target loss rate $q = 0.1\%$, despite bursty losses and the average loss-rate varying between 0.5% and 3.3%. Furthermore, this conclusion is true not just for the means, but for the tails of the distribution as well. Figure 7 shows the achieved loss rate for the FEC+ARQ based CLVL for three different background traffic scenarios. In addition, our recovery algorithm achieves the target loss irrespective of whether the IP routers along the virtual link use FIFO or RED queues. These results demonstrate that our CLVL algorithm is robust over a range of dynamic traffic conditions and works even when the underlying loss rate is 30 times larger that the target loss rate, q.

Wide Area Evaluation: Given our specific choice of overlay nodes, we found 83 virtual links in our overlay testbed to be lossy. A link is characterized as lossy if the loss-rate along the link is at least 0.5%. Across each link, we ran a CLVL abstraction for time-ranges varying from 20 minutes to 1 hour. In order to measure the system under stress, the sending rate as determined by N-TCP averaged between 120 Kbps (across Cable modems and DSL lines) to 2 Mbps from other nodes. [2]

[2]Given this high sending rate, we did not run our experiments for continued periods of time. Additionally, bandwidth is an ex-

c_{avg}	Mean $\frac{c_{min}}{c_{avg}}$	Variation
$100 - 200$ Kbps	0.41	$0.32 - 0.49$
$200 - 400$ Kbps	0.48	$0.29 - 0.75$
$400 - 800$ Kbps	0.41	$0.19 - 0.81$
$800 - 1600$ Kbps	0.41	$0.16 - 0.86$
> 1600 Kbps	0.49	$0.04 - 1.0$

Figure 9: Variation of $\frac{c_{min}}{c_{avg}}$ as a function of c_{avg}

The FEC+ARQ based CLVL achieved the target loss-rate over 80 of the 83 virtual links. Our FEC+ARQ algorithm failed to achieve the target loss rate of 0.1% only across 3 of the overlay links. Upon closer investigation, we found the causes to be : *short outages* and *bi-modal loss distributions*. A short outage refers to a period of time when all packets transmitted along a virtual link are lost. Within our testbed, we noticed non-recoverable losses along two links: PDI-NBG and Unibo-Media. These non-recoverable losses lasted for short periods of time (< 5 s). Short outages can occur due to a variety of problems such as routing changes or link resets. A loss distribution is said to be bi-modal if the losses experienced in every window is zero or very high. Links with very bursty losses have a bi-modal distribution. An FEC+ARQ based CLVL cannot recover a large portion of a window of packets from a bimodal loss distribution if a long burst affects both the FEC window, and the ARQ transmissions. During our experiments, Mazu-Cba1 experienced a bimodal loss distribution.

6.3 Statistical Bandwidth Guarantees

In this section, we answer the question: What bandwidth guarantees are realizable on a virtual link?

Recall that the statistical bandwidth guarantee achievable along a virtual link is given by c_{min} such that $P(c<c_{min}) = u$, where c represents the instantaneous bandwidth along the virtual link, and u represents the probability with which the guarantee is not met. The Rate Estimator module updates c once every window of packets (O(RTT) sec) based on the feedback information received from the next OverQoS hop.

Across the 171 pairs of nodes between the 19 end-hosts in our testbed, we monitored 83 unique virtual links over a period of 7 working days. Figures 8(a) and (b) show the distribution of c_{min} for $u = 0.01$ and $u = 0.005$. We make two observations. First, the value of c_{min} is greater than 100 Kbps for more than 80% of the links. 20% of the links are predominantly connected to broadband hosts. Second, in many cases, c_{min} is at least 25% of the average throughput along the virtual link. In specific cases, c_{min} is as large as 90% of the average throughput. The median value of c_{min}/c_{avg} is 0.4 and 0.35 for $u = 0.01$ and $u = 0.005$ respectively. Figure 9 shows the variation of c_{min}/c_{avg} as a function of c_{avg}. As c_{avg} increases, we notice that the maximum value of c_{min}/c_{avg} increases while the minimum value decreases. The minimum decreases because we notice *self-induced losses* across some of the links thereby causing MulTCP to drastically reduce its sending rate and thereby reducing c_{min}.

pensive resource for RON and PlanetLab which we did not want to misuse.

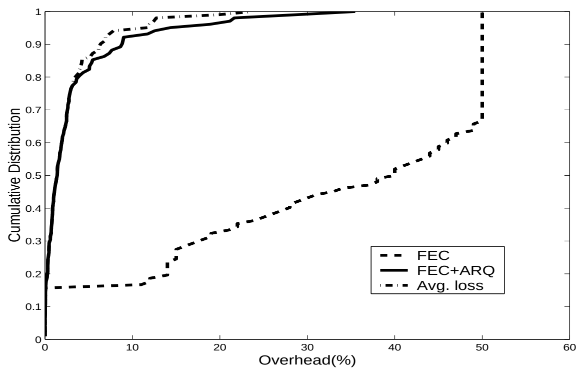


Figure 10: Overhead Characteristics in the wide-area testbed: Compares overhead of FEC+ARQ with FEC and Average loss rate across the links, p_{avg}.

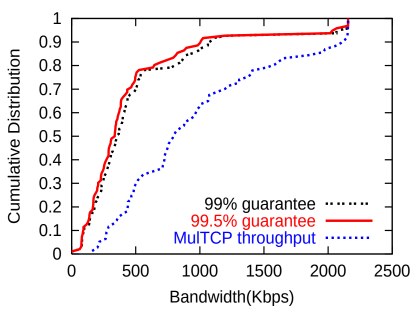
Stability of c_{min}: If the underlying distribution of c is stable, the estimated value of c_{min} will roughly be a constant. However under dynamic conditions, we need to continuously re-estimate c_{min} and flows need to renegotiate their bandwidth reservations. For a given value of u, we estimate c_{min} using $O(1/u)$ samples of c. As an example, given $RTT = 100$ msec and $u = 0.01$, we can calculate c_{min} based on the last $20/u$ samples (representing a history of 200 seconds). In this scenario, flows renegotiate their bandwidth requirements every few minutes.

Figure 8(c) shows the variation as a function of time across four separate virtual links from Europe to North America. We make two observations: First, the value of c_{min} is very stable compared to variations in the available bandwidth, c. Across these links, c_{min} does not deviate more than 10% around its mean value. Second, an on-line algorithm for estimating c_{min} based on past history is a reasonable approach. While we set $P(c < c_{min})$ to be 1%, the actual value of c is less than the estimated c_{min} in no more than 1.3% of the cases across all four virtual links.

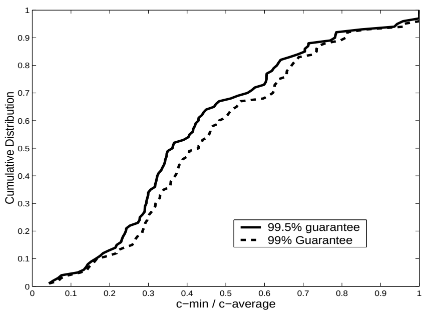
6.4 OverQoS Cost

6.4.1 Overhead Characteristics

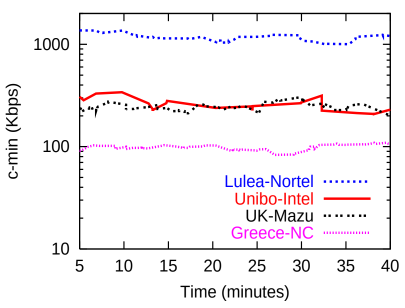
Figure 10 shows the cumulative distribution of the overhead for an FEC+ARQ based CLVL across the 83 overlay links over which we performed our measurements. For each link, we ran an $N-$TCP pipe for $N = 10$ and measured the overhead required to achieve a target loss rate of



(a)



(b)



(c)

Figure 8: (a) Cumulative distribution of the bandwidth guarantee c_{min} across 100 separate measurements over 83 unique overlay links measured across 7 different days from Jan14th - Jan28th. For each run along a single overlay link, we generated between 100,000 - 300,000 packets. All measurements are taken on weak-days (many of them during working hours). (b) Distribution of the fraction c_{min}/c_{avg} across all the links. (c) Variation of c_{min} across 4 different virtual links between Europe and North America. c_{min} is measured as an on-line estimate over a maximum previous history of 5 minutes (time to collect $20/u$ samples for $u = P(c < c_{min}) = 0.01$).

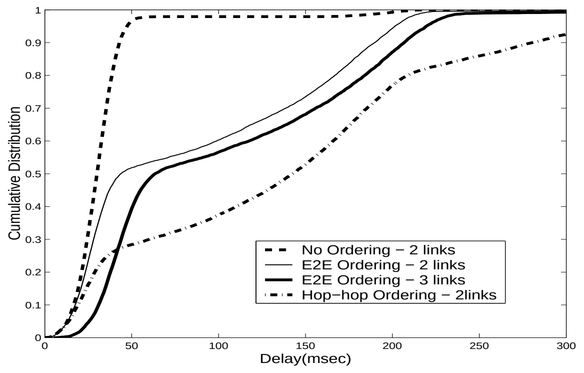


Figure 11: Cascaded CLVL scenario using FEC+ARQ CLVLs: End-to-end ordering within OverQoS network has much better delay characteristics than hop-by-hop ordering.

$q = 0.1\%$. We notice that the overhead of FEC+ARQ is very close to the average loss-rate along the overlay links. The difference between the two is the amount of FEC used in the second round to protect the retransmitted packets. In comparison, a pure FEC based CLVL construction far higher bandwidth. This is primarily due to the network loss characteristics: the burstier the background traffic (i.e., the longer the tail of the loss-rate distribution), the higher the amount of FEC required to recover from these losses [22].

6.4.2 *Delay Characteristics*

This section answers the question: What is the delay cost of using OverQoS? A potential criticism of our algorithm is that it increases the delay observed by packets.[3] There are two reasons for this increase in delay. First, if one or more packets in a window are lost, the recovery process will cause additional delays. Second, if OverQoS is required to support in-sequence delivery of packets, the loss of one packet can increase the delay of other packets. Our implementation showed that the additional delay incurred at a node due to processing overhead is negligible.

In OverQoS, we can support three different models for packet delivery: (a) No packet ordering; (b) End-to-end (E2E) ordering between first and last OverQoS node in a path; (c) Hop-by-hop ordering. We consider a simple scenario where an overlay path traverses multiple overlay nodes with each link having an RTT of 100 msec and experiencing frequent losses ($p_{avg} = 4\%$). Figure 11 shows the distribution of the additional delay incurred due to loss recovery for each of the three packet delivery models. We consider a path consisting of up to three overlay links. We make three observations. First, end-to-end packet recovery has much better delay characteristics than hop-by-hop delay characteristics. Second, the additional delay incurred by adding new OverQoS nodes along a path is limited. Third, the additional delay is also dependent on the loss rate. The loss-rate dictates how frequently the loss recovery process is being invoked.

6.5 Fairness and Stability

The N-TCP pipe abstraction is built using MulTCP which inherently is TCP-friendly in the aggregate with both cross traffic and other OverQoS traffic. Figure 12 illustrates this fact using a real-world experiment on a link between a university node and NBG, a node behind an access network. Three OverQoS bundles (with N=2, N=4,N=8) compete on this shared bottleneck under two different scenarios: (a) no cross-traffic, and (a) cross-traffic consisting of five long lived TCPs (*wget* downloading content in parallel).

[3]Note that this is a legitimate concern only for OverQoS packets and not for other flows sharing links on a path.

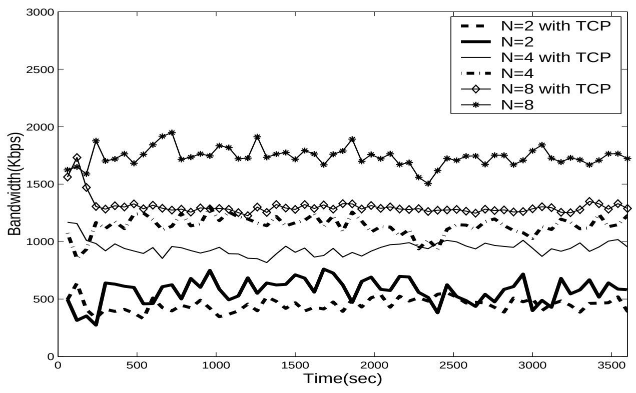


Figure 12: Three independent OverQoS links compete for bandwidth on a shared bottleneck where all CLVLs are established between a university node and NBG, a node behind an access network in Oregon. To make the graph readable, the value of b is averaged over every minute.

We make two observations. First, the three OverQoS bundles co-exist with each other and with the background traffic. Second, the ratio of throughputs of the three OverQoS bundles is preserved across both scenarios.

7 Related Work

We classify related work into: (a) QoS architectures; (b) overlay-based techniques; (c) loss recovery mechanisms.

QoS architectures: OverQoS differs from previously proposed QoS architectures because it does not require QoS mechanisms in all routers in the network. IntServ [10] requires each IP router to implement per-flow admission control on the control path, and per-flow classification, buffer management and scheduling on the data path. Similarly, DiffServ [8, 24] requires edge routers to perform per-flow or per-aggregate classification, buffer management and scheduling, and core routers to perform per-class operations.

OverQoS can leverage the service provided by the underlying network to enhance its services. For instance, within a DiffServ domain, OverQoS may use Expedited Forwarding (or premium service [24]) and provide per-flow bandwidth (and perhaps delay) guarantees. In addition, OverQoS can use techniques like the one proposed in the SCORE architecture [35] to improve its scalability, by having only the first OverQoS node on a flow's path maintain state.

To address the scalability problems of providing end-to-end services, several recent papers have advocated the idea of using endpoint measurement-based admission control (EMBAC) [11, 20, 14]. With EMBAC, an end-host measures the network characteristics of a path and accepts a flow only if the flow's requirements can be satisfied by the path. However, unlike OverQoS, all EMBAC solutions assume that all routers implement some mechanism to isolate the admission-controlled traffic from the best-effort traffic.

Overlay-based Techniques: Several papers have proposed the use of overlay-based approaches for deploying multicast [12, 21] and improving routing functionality (e.g., resilience, as in RON [7]). These systems are motivated in large part by the difficulty of modifying the IP layer both in terms of deployment and in terms of system robustness.

Within the context of QoS, edge-to-edge congestion control [18], a proposal to support a limited range of bandwidth services using an overlay framework, also requires modifications at all edge routers in a domain to achieve its functionality. Service Overlay Network [13], is a recent proposal that purchases bandwidth with certain QoS guarantees from network domains using SLAs and stitches them to provide end-to-end QoS guarantees. Such an architecture would still rely on the underlying domains to meet their specified QoS requirements. For streaming audio and video, multimedia proxies offer the services of smoothing losses [34] and selective discard/recovery of packets [37]. While OverQoS can leverage many of these techniques, two issues differentiate these works from OverQoS: (a) OverQoS can apply the same QoS enhancements within the network as opposed to end-to-end; (b) streaming media flows in OverQoS can be shaped as part of a larger aggregate as opposed to being treated as separate flows.

Loss Recovery: FEC and ARQ based approaches have been investigated in the context of packet audio, video and Internet telephony [9]. Since the FEC constraints are different in these applications (recovering a fraction of packets may be sufficient), we may not be able to apply these results directly to our setting. However, classical coding mechanisms used in wireless networks can potentially be applied to our problem [22, 31, 38].

8 Conclusions

In this paper, we show that it is possible to use overlay networks to enhance Internet QoS without any support from the underlying IP network. Using two real-world applications and experiments over a wide-area testbed we demonstrate three such QoS enhancements: (a) smoothing losses; (b) prioritization of packets within an aggregate; (c) statistical loss and bandwidth guarantees. OverQoS is able to achieve all these enhancements with little (*i.e.*,5%) or no extra bandwidth overhead.

While our results suggest that OverQoS can be a viable architecture to enhance the Internet QoS, more remains to be done. Our current solution assumes that the flows' paths at the OverQoS level are predetermined. A natural extension would be to combine admission control and path selection, e.g., to have the entry OverQoS node compute the "best" path that satisfies a flow's requirements at the admission time. One possibility would be to use RON [32] to find paths with better performance characteristics and to recover from network failures. Another interesting problem would be to determine the "optimal" placement of the OverQoS nodes in the network. We intend to address these issues as part of future work.

Acknowledgments

We thank our anonymous reviewers and our shepherd, Jennifer Rexford, for their invaluable feedback which helped substantially in improving the quality of the paper. We owe our gratitude to Dave Andersen, Tom Anderson, Jon Crowcroft and Scott Shenker for their detailed technical feedback at various stages of this work. We thank Nick Feamster and Matthew Caesar for providing us with MPEG, PESQ processing scripts and several students at UC Berkeley who read earlier drafts of this paper and provided invaluable feedback.

References

[1] CounterStrike. `http://www.counter-strike.net`.

[2] Helix Universal Server - Basic version. `http://www.real.com`.

[3] ITU-T P.862: Perceptual evaluation of speech quality (PESQ). `http://www.itu.int/rec/recommendation.asp?type=items&lang=E&parent=T-RE%C-P.862-200102-I`.

[4] Japanese broadband statistics. `http://www.johotsusintokei.soumu.go.jp/whitepaper/eng/WP2003/Chapter1-1%.pdf`.

[5] Movie Player for Linux. `http://www.mplayerhq.hu`.

[6] MPEG-1 Specification. `http://www.chiariglione.org/mpeg/standards/mpeg-1/mpeg-1.htm`.

[7] D. Andersen, H. Balakrishnan, M. Kaashoek, and R. Morris. Resilient Overlay Networks. In *Proc. ACM SOSP*, Oct. 2001.

[8] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss. An Architecture for Differentiated Services, Oct. 1998. RFC 2475.

[9] J. Bolot, S. Fosse-Parisis, and D. Towsley. Adaptive FEC-based error control for Internet telephony. In *Proc. of IEEE INFOCOM*, Mar. 1999.

[10] R. Braden, D. Clark, and S. Shenker. Integrated Services in the Internet architecture: An overview, June 1994. Internet RFC 1633.

[11] L. Breslau, E. W. Knightly, S. Shenker, I. Stoica, and H. Zhang. Endpoint admission control: Architectural issues and performance. In *Proc. of ACM SIGCOMM*, Sept. 2000.

[12] Y. Chu, S. G. Rao, S. Seshan, and H. Zhang. Enabling conferencing applications on the Internet using an overlay multicast architecture. In *Proc. of ACM SIGCOMM*, Aug. 2001.

[13] Z. Duan, Z. Zhang, and Y.T.Hou. Service Overlay Networks: SLA, QoS and bandwidth provisioning. In *Proc. of ICNP*, Nov. 2002.

[14] V. Elek, G. Karlsson, and R. Ronngren. Admission control based on end-to-end measurements. In *Proc. of IEEE INFOCOM*, Mar. 2000.

[15] A. Feldmann, P. Huang, A. C. Gilbert, and W. Willinger. Dynamics of IP traffic: A study of the role of variability and the impact of control. In *Proc. of ACM SIGCOMM*, Aug. 1999.

[16] S. Floyd, M. Handley, J. Padhye, and J. Widmer. Equation-based congestion control for unicast applications. In *Proc. of ACM SIGCOMM*, Aug. 2000.

[17] C. Fraleigh, F. Tobagi, and C. Diot. Provisioning IP backbone networks to support latency sensitive traffic. In *Proc. of IEEE INFOCOM*, Mar. 2003.

[18] D. Harrison, S. Kalyanaraman, and S. Ramakrishnan. Overlay Bandwidth Services: Basic framework and an edge-to-edge closed-loop building block, Jan. 2001. Preprint.

[19] `http://ipmon.sprintlabs.com`.

[20] S. Jamin, P. Danzig, S. Shenker, and L. Zhang. A measurement-based admission control algorithm for Integrated Services packet networks. In *Proc. of SIGCOMM*, 1995.

[21] J. Jannotti, D. K. Gifford, K. L. Johnson, M. F. Kaashoek, and J. O'Toole. Overcast: Reliable multicasting with an overlay network. In *Proc. USENIX OSDI*, Oct. 2000.

[22] S. Lin and D. Costello. Error Control coding: Fundamentals and applications. In *Prentice Hall*, Feb. 1983.

[23] M. Luby, M. Mitzenmacher, A. Shokrollahi, D. Spielman, and V. Stemann. Practical loss resilient codes. In *Proc. of ACM STOC*, 1998.

[24] K. Nichols, V. Jacobson, and L. Zhang. An approach to service allocation in the Internet, Nov. 1997. Internet Draft.

[25] Ucb/lbnl/vint network simulator - ns (version 2). http://www-mash.cs.berkeley.edu/ns/.

[26] E. Osborne and A. Simha. Traffic Engineering with MPLS. In *Cisco Press*, July 2002.

[27] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose. Modeling TCP throughput: A simple model and its empirical validation. In *Proc. of ACM SIGCOMM*, Oct. 1998.

[28] `http://www.planet-lab.org`.

[29] P.Oechslin and J.Crowcroft. Weighted proportionally fair differentiated service tcp. In *Proc. of ACM Computer Communications Review*, 1998.

[30] L. Rizzo. `http://info.iet.unipi.it/~luigi/fec.html`.

[31] L. Rizzo and L. Vicisano. RMDP: An FEC-based reliable multicast protocol for wireless environments. *Mobile Computing and Communications Review*, 2(2), 1998.

[32] Resilient Overlay Networks. `http://nms.lcs.mit.edu/ron/`, 2001.

[33] S. Savage, A. Collins, E. Hoffman, J. Snell, and T. Anderson. The end-to-end effects of Internet path selection. In *Proc. of ACM SIGCOMM*, Aug. 1999.

[34] S. Sen, J. Rexford, J. Dey, J. Kurose, and D.Towsley. Online smoothing of variable-bit-rate streaming video. In *IEEE Trans. on Multimedia*, Mar. 2000.

[35] I. Stoica and H. Zhang. Providing Guaranteed Services without per flow management. In *Proc. of ACM SIGCOMM*, Sept. 1999.

[36] W. Willinger, M. S. Taqqu, R. Sherman, and D. V. Wilson. Self-similarity through high-variability: Statistical analysis of Ethernet LAN traffic at the source level. In *Proc. of ACM SIGCOMM*, Aug. 1995.

[37] Z. Zhang, S. Nelakuditi, R. Aggarwal, and R. Tsang. Efficient selective frame discard algorithms for stored video delivery across resource constrained networks. In *Proc. of IEEE INFOCOM*, 1999.

[38] M. Zorzi. Performance of FEC and ARQ in bursty channels under delay constraints. In *Proc. of VTC'98*, May 1998.