An Edge-to-Edge Overlay Congestion Control Architecture for the Internet

David Harrison[1], Shivkumar Kalyanaraman[2]
[1]Computer Science Department
[2]Electrical, Computer, and Systems Engineering Department
Rensselaer Polytechnic Institute
harrisod@cs.rpi.edu, shivkuma@ecse.rpi.edu

Abstract—

This paper proposes an edge-to-edge overlay congestion control architecture for managing traffic *aggregates*. Motivated by scalability issues, the core rate-based scheme breaks up congestion in the intermediate network(s) and distributes it across overlay edges leading to superior best-effort performance. Consolidation of bottlenecks at overlay edges also enables the creation of *purely edge-based* better-best-effort services, such as an end-to-end *lossless TCP* service illustrated in the paper. Our overlay architecture is unique in that it:

a) does not require any support from interior network nodes or end-systems **(unlike ATM ABR [15], credit-based schemes [19], bit-based schemes [25], [6] or TCP modifications [12], [23])**
b) **operates in a *lossless* manner with bounded buffers unlike TCP which uses packet loss to indicate congestion [11],**
c) **aims to *complement, not substitute* existing end-to-end congestion control (eg: TCP) or traffic management schemes (eg: diff-serv [1]) to achieve scalability and services goals,**
d) **enables new edge-based better-best-effort services across CoS-based [1] or VC-based intermediate network(s),**
e) **is applicable to controlling any type of L4 traffic (TCP, non-TCP), and**
f) **can be deployed on a variety of existing networks (IP, MPLS, frame-relay, ATM or X.25).**

Keywords— **Internet congestion avoidance, congestion control**

I. Introduction

Despite the explosion of bandwidth in the core of the Internet, congestion is commonly observed on end-to-end paths resulting in degradation of perceived best-effort service. This is because congestion continues to exist in key spots such as access links, aggregation points, tail circuits (to remote locations), international circuits and at peering points. The degree of flow multiplexing at these bottlenecks is also increasing, leading to a scalability problem recently articulated by Morris [24]. A minimum of 4-8 packets per TCP flow is required at the bottleneck, combined with a buffer management strategy that works at small time-scales (in hardware) and enforces this minimum [21]. Therefore, even with end-to-end adaptive control, a buffer management scheme at a bottleneck scales only up to the number of flows for which it can effectively enforce this minimum allocation. A key contribution of our paper is a quickly (and incrementally) deployable distributed architecture which scales performance dramatically beyond this limit.

More generally, we propose an *overlay congestion-control architecture* which *transparently* establishes control loops around bottlenecks. We call these control loops *virtual links*. The virtual links operate through CoS-based (DiffServ[1]) or VC-based networks, and rely on LSPs (MPLS), VCs (ATM, Frame Relay), or tunneling (IPSec[17]) to provide high likelihood of single-path routing (symmetric paths are not required). With coordination across ISPs similar to DiffServ[1], virtual links can span multiple networks between arbitrary points.

Our core scheme breaks up and distributes the congestion problem into smaller-scale per-loop congestion problems. Moreover, by moving the congestion problems to the "edge" of the network, more stateful and protocol-aware techniques (eg: FRED [21], TCP-friendly dropping [22], TCP rate-control [16] etc) can be used to further enhance user-perceived end-to-end service. This paper differs from our recent work [8] (Explicit Edge Control (EEC)) which also proposed edge-to-edge control, but was not a true overlay architecture. This was because the core EEC scheme required support from bottlenecks (bit-marking) to achieve congestion distribution. The core scheme proposed in this paper is called "Implicit Edge Control (IEC)" to emphasize this difference. In other words, IEC leads to an architectural as well as a congestion control contribution in this paper.

One of the results of the overlay control architecture is that bottlenecks along a virtual link are now consolidated at the virtual link head. Such consolidation enables the creation of *purely edge-based* better-best-effort services. We illustrate one such service, the end-to-end *lossless TCP* service in this paper. Future work will explore the full edge-based services creation potential of our architecture.

The remainder of this paper is organized as follows. Section II reviews the basic congestion-control mechanisms shared between EEC and IEC, and relates it to congestion control literature. Section III presents the new aspects of the Implicit Edge Control scheme. Section IV presents selected simulation results including IEC integrated with TCP Rate Control. Section V discusses incremental deployment into a Class of Service (CoS) network. We finish the paper with a discussion of future work.

II. Core Congestion Control Concepts

This section discusses the core and novel congestion control concepts introduced in EEC and IEC. Though some of these ideas are introduced in our earlier work[8], this paper presents a fuller picture of the contribution. Our scheme assumes the simple FIFO bottleneck model shown in Figure 1. The FIFO queue could correspond to a single class in a Class-of-Service (CoS) network [1]. While the capacity allocated to the class could vary due to scheduling, we assume that flows not controlled by our scheme are *isolated* into a separate class.

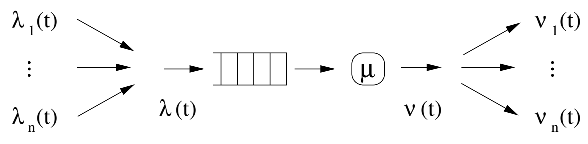


Fig. 1. Traffic Control Model

A. Congestion Epoch

The scheme is based upon the simple observation that *at all times*, the sum of the output rates of flows passing through a particular bottleneck ($\sum \nu_i$) is less than or equal to the bottleneck capacity μ (see Figure 1). Most important, this condition holds during periods of congestion which we call *"congestion epochs"*. We define congestion epoch as the period of full utilization incurred when the mean aggregate load (λ) exceeds capacity (μ).

The EEC scheme [8] detects this state by marking all departing packets when the instantaneous queue length exceeds a carefully designed threshold provisioned to be above steady-state queue fluctuations. The goal of IEC is to perform congestion epoch detection without involving the participation of the bottleneck. Our concept of the congestion epoch is differentiated from contemporary work in the following respects:

- The congestion epoch concept is primarily a *rate-based* concept because our goal is to match aggregate input *rate* to the bottleneck rate. Buffers are used merely to absorb statistical fluctuations and not considered part of the "capacity" of the network even if buffer space is unbounded. This is different from traditional window-based [11], [25] or credit-based [19] approaches which attempt to match aggregate window size to the bandwidth-RTT product of the network where buffers are typically considered part of the bandwidth-RTT product.
- Since our notion of congestion epoch implies bandwidth saturation and not buffer saturation, *packet loss is not required in the congestion detection process*. Morever, as described later in this paper, such *congestion detection is possible without any participation of the bottleneck* leading to overlay implementation capabilities and deployment advantages. It is the *combination* of a lossless detection of congestion and no support requirements from intermediate bottlenecks which is unique. Note that our notion of "lossless" is only in an edge-to-edge sense. End-to-end performance implications are profound and will be discussed later in the paper.

In comparison, schemes based upon the bandwidth-RTT product paradigm which do not expect bottleneck participation have not been very successful in developing lossless congestion detection algorithms. For example, end-to-end delay-based schemes (Eg: see [20], [13]) have proven hard to design. This is because the definition of congestion period is intertwined with (possibly unbounded) buffer size. Unbounded buffer size leads to unbounded queue lengths and large RTT variability. RTT variability means that the bandwidth-RTT estimate is prone to variability, which in turn means that there is not a stable reference for congestion detection (i.e. the beginning and end of a congestion period). Consequently many schemes use loss-based congestion detection [11].
Congestion *avoidance* approaches [25] almost inevitably require some form of bottleneck participation, either for early congestion detection (Eg: RED,Decbit [7], [25]) or for per-flow isolation (Eg: packet-pair scheme [18]). Credit-based schemes [19] and explicit rate-based schemes [15], [14] developed for ATM ABR service represent other types of schemes which can achieve zero-loss operation

but which require complex data-path support from the bottleneck.

Despite the above advantages, we believe that further work is necessary to adapt this congestion epoch concept for end-to-end congestion control efforts (eg: ECM work at IETF [5]) primarily because of partial deployment concerns. This is because our work still requires class-based (not per-flow) isolation of traffic aggregates controlled using the congestion epoch concept. Such isolation of traffic aggregates is not a tenable assumption for general end-to-end implementation. Without such isolation, flows controlled using our concept are beaten down by TCP flows which use loss-based (late) congestion detection techniques. However, the concept is well-suited for ISP networks where a class can be set aside for edge-to-edge controlled traffic aggregates.

B. Edge-to-edge Control Increase/Decrease Policy

This section outlines the edge-to-edge control increase/decrease policy assuming that there exists a method for congestion epoch detection. Referring again to Figure 1, note that output rates of flows (ν_i) can be measured at the receiver (egress edge) and fed back to the sender. During congestion epochs each sender (ingress edge) imposes a rate limit r_i such that $r_i \leftarrow \beta\nu_i$ where $\beta < 1$. If each sender *consistently* constrains its input rate (λ_i) such that $\lambda_i \leq r_i$ during the congestion epoch, the epoch will eventually terminate.

In other words, the forward path requires mechanisms to detect congestion epochs and the reverse direction involves the explicit feedback of the measured output rates (ν_i). This positions the implementation requirements of our scheme in-between purely implicit congestion detection schemes (eg: TCP [11]) and explicit feedback schemes ([25], [6], [15], [19]). Moreover, our rate increase/decrease policy differs from the well known additive-increase multiplicative-decrease (AIMD) policy [2].

We call our increase/decrease policy "AIMD-ER." The steady state dynamics are demonstrated in Figure 2. In particular, the multiplicative decrease factor is applied to the explicitly fed back output rate ("ER"), and not to the input rate limit as in AIMD. This allows us to choose a backoff parameter much larger than 0.5 (the backoff used by TCP) and leads to the rate being held relatively flat (below output rate) during the congestion epoch as shown in the figure. NETBLT[3] is the only other scheme to the best of our knowledge that uses a policy similar to AIMD-ER, i.e., renegotiating input rate based upon egress measurements. However, NETBLT negotiates rates only once per block transfer and congestion is detected based on rate difference which is prone to instability when rate differences are small. In addition to AIMD-ER, the ingress edge in EEC and IEC implements the following details:

- A rate-based slow start analogous to TCP slow start in order to come up quickly to a fully-utilized link.
- Each ingress shapes traffic entering the network using a leaky bucket shaper, to conform to a Linearly Bounded Arrival Process (LBAP) [4], i.e.:

$$\int_t^{t+\tau} \lambda_i(T)dT \leq r_i\tau + \sigma \tag{1}$$

where r_i is the ith virtual link's rate limit and σ is the maximum burst size.

- During additive increase, the rate increments by $\frac{\sigma}{\mathsf{T}}$ at the beginning of each edge-to-edge round-trip time denoted T.
- During congestion, the egress continues to feedback smoothed output rate measurements once per T until it detects the end of the congestion epoch.

III. IMPLICIT EDGE CONTROL

Implicit Edge Control (IEC) and Explicit Edge Control (EEC) largely share the same edge system behavior. They differ in the manner in which each infers the beginning and ending of congestion epochs. In particular, IEC introduces two new techniques, one for detecting the beginning of the epoch and one for detecting the end of an epoch. The *beginning* of a congestion epoch is detected whenever the virtual link's contribution ("accumulation"), q_i, to the queue length exceeds σ, the allowed burstiness. The *end* of the congestion epoch is detected when the queuing delay at the bottleneck is eliminated, i.e., the one-way delay seen by the egress comes within $\epsilon = \frac{\sigma}{2\nu_i}$ of the minimum one-way delay. The section elaborates on the details behind these techniques.

A. Detection of Congestion Epoch Beginning

Our earlier work EEC [8] detected congestion epochs by having the bottleneck promiscuously mark packets (and signal all participating loops) whenever the queue length is larger than the sum of the allowed burstiness for each loop (i.e. threshold = $N\sigma$) . In IEC, we aim to emulate the same behavior but purely edge-to-edge. Specifically, the threshold $N\sigma$ can be interpreted in two ways: a) as an accumulation of σ packets per virtual link, and b) as a queuing delay of $\frac{N\sigma}{\mu}$. Our edge-based strategy uses the

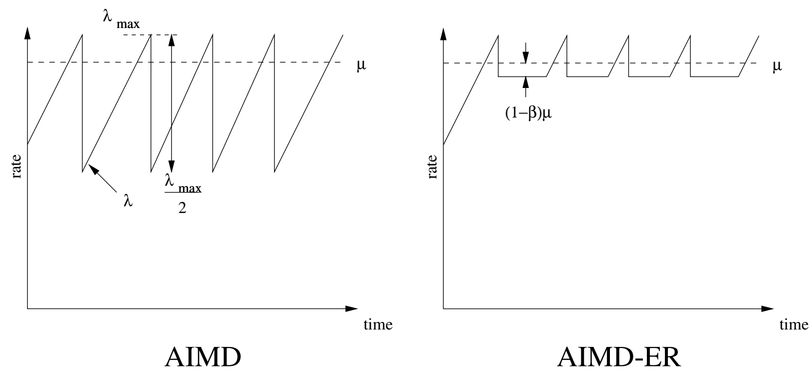


Fig. 2. Steady-State Dynamics

first interpretation to detect the epoch beginning and the second interpretation to detect the epoch end.

In particular, if each loop (or virtual link) can track its contribution ("accumulation"), q_i, to the queue length, then each loop can independently detect congestion when q_i exceeds σ, its *allowed burstiness*. Recall that each loop is rate-limited by a leaky bucket shaper which has a burstiness parameter, σ. Also observe that the EEC detection procedure is similar to the IEC procedure in the following sense. That is, when all N virtual links contribute an accumulation of σ, the total accumulation is $N\sigma$ which is the minimum mark threshold used in the EEC scheme. In this sense, accumulation is similar to the "credit" concept in credit-based schemes [19], but we try to estimate it in a rate-based framework.

The accumulation q_i can be calculated using the following observation. Assume a sufficiently large interval τ. If the average input rate during this period τ is $\overline{\lambda}_i$, and the average output rate is $\overline{\nu}_i$, the accumulation caused by this flow during the period τ is $(\overline{\lambda}_i - \overline{\nu}_i) \times \tau$. The accumulation measured during this period can be added to a running estimate of accumulation q_i which can then be compared against the virtual link's maximum burst size imposed by its leaky bucket shaper.

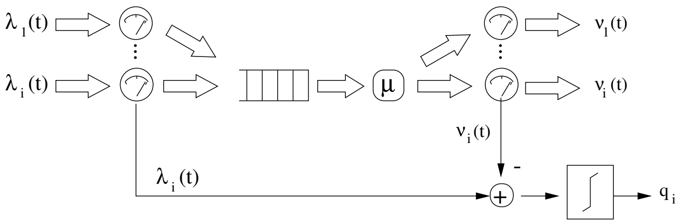


Fig. 3. Measuring Virtual Link's Contribution to Queue Length

The ingress node sends two control packets in each edge-to-edge RTT T (but no faster than the real data rate). We choose two control packets/T in an attempt to emulate the Nyquist sampling criterion. τ is the interdeparture time of control packets *at the ingress*. In each control packet, the ingress inserts a timestamp and the measured average input rate (measured over the last τ seconds), $(\overline{\lambda}_i)$. The average output rate $\overline{\nu}_i$ is measured over the time interval between *arrivals* of consecutive control packets at the egress. The egress node now has all three quantities required to do the computation: $(\overline{\lambda}_i - \overline{\nu}_i) \times \tau$ and add it to a running estimate of accumulation. The running estimate of accumulation is also reset at the end of each congestion epoch to avoid propagation of measurement errors.

B. Detection of Congestion Epoch End

As introduced in the previous section, recall that our detection strategy is based upon emulating EEC concepts. Specifically, for detecting the end of the congestion epoch, we need to detect when the bottleneck queue drops below the EEC threshold $N\sigma$. For this purpose, we interpret the threshold as an equivalent queuing delay of $\frac{N\sigma}{\mu}$. Since the egress edge does not have access to the estimate of the bottleneck rate divided by N, μ/N, it uses an approximation instead, ν_i. Therefore, our goal is to estimate when the queueing delay drops below $\frac{\sigma}{\nu_i}$. To allow for estimation errors, we choose to formulate our condition more weakly, i.e. detect congestion epoch end when the queueing delay drops below $\frac{\sigma}{2\nu_i}$. The egress rate ν_i is an effective approximation of μ/N because higher-rate virtual links (i.e. with higher ν_i) which contribute more to congestion wait for a longer period before they detect the end of the congestion epoch.

The next problem in this method is to estimate bottleneck queuing delay at the egress edge. For this, we use estimates of one-way edge-to-edge delay and compare the current delay estimates (during a congestion epoch) to the *minimum one-way delay* seen in the past. We observe that delay is a one-sided distribution, i.e., delay samples will not drop below persistent delay such as propagation, transmission, processing, and *persistent queueing delays*. Because delay samples provide a reliable upper bound on persistent delay, delay samples are better for detecting the *end* rather than the beginning of a congestion epoch. As each control packet arrives at the egress, the egress updates the minimum one-way delay seen so far. If the delay is within $\epsilon = \frac{\sigma}{2\nu_i}$ of the minimum one-way delay, the egress flags the end of the epoch and stops sending negative feedback.

Further, since low delay indicates lack of congestion, the scheme does not attempt to detect the beginning of a congestion epoch until a control packet has a delay greater than ϵ above the min delay. Note that the clocks do not need to be synchronized since we are interested only in delay relative to the minimum delay. Other measurement and robustness issues (including multiple bottlenecks and the minor effect of clock skews over such time-scales) are discussed in our technical report [9]. An important robustness issue concerns packet loss. If the network class does not have sufficiently provisioned buffers or if available capacity changes dramatically without corresponding adjustment in buffer allocation, packet losses may occur. As a fallback procedure, the IEC scheme backs off by $\nu_i/2$ if packet loss is detected. Packet loss is detected at the egress by examining aggregate counts of packets sent (by ingress) and received (at egress) between control packets.

Observe that we use two methods for detecting the beginning and end of the epoch. This is because the delay-based technique has a reliable minimum reference basis (minimum one-way delay) and the accumulation-based technique has a reliable maximum reference basis (maximum per-VL accumulation). Note that it is harder to formulate a reliable maximum reference basis for the delay-based scheme and a minimum reference basis for the accumulation scheme. The former is because of the one-sided nature of the delay distribution and the latter because of the accumulation of measurement error. This especially holds under variable demand and variable capacity conditions. This is why our scheme is not purely delay-based or purely credit (i.e. accumulation)-based, but a hybrid of the two approaches.

IV. Simulation Results

We have performed several simulations investigating the performance of IEC in a technical report [9]. For reasons of space, we report a subset of the simulations in this paper. This section uses simple, but non-trivial simulations to illustrate the behavior of IEC in the following dimensions:

- **Basic dynamics**: uses a number of persistent end-to-end flows (2-100) to show how IEC distributes congestion.
- **TCP scalability**: evaluates the dramatic improvement in end-to-end TCP performance as a result of distribution and divide-and-conquer of congestion at the edges. We show that the performance is superior to stateful queue management schemes applied to the single bottleneck (FRED [21]). We examine up to 1000 flows in our simulation examples.
- **Edge-based services**: illustrates a simple end-to-end TCP (or L4) service which achieves zero-loss, zero-timeout and fair capacity sharing. We use Packeteer's TCP rate control technology [16] at the edges to implement a "perfect callback", i.e., admit TCP packets into the network only when the network has capacity to accept it. We stress-test this service by simulating bottlenecks having less than one packet per-TCP flow. None of the conventional buffer management schemes can provide such a service. Such edge-based private network services could be used as a building block in VPNs.
- **Deployment**: Shows how a simple class-based network architecture can be leveraged to implement our overlay methods. In particular, we show that setting aside a single class for edge-to-edge controlled aggregates is sufficient. This allows ISPs to incrementally deploy this technology on a customer-by-customer basis, without needing to have a single flag day for upgrades.

The technical report [9] further illustrates the capabilities of IEC:

- To protect well-behaved TCP connections from rogue sessions (UDP streams or hacked TCP implementations) which attempt bandwidth-based denial of service attacks. This can be thought of as a "distributed penalty-box" following the terminology introduced in RED [7].
- To be robust to delay heterogeneity, multiple bottleneck cases, variable bottleneck capacity etc.
- To show equal or better performance compared to EEC.

All simulations use `ns-2.1b5`. Except where noted the simulations use the parameters in figure 4. As mentioned elsewhere, our earlier paper also [8] also develops analytical steady-state queue length bounds ($4N\sigma$).

We use one bandwidth-RTT product worth of buffer as recommended in [27]. The choice of data packet size is chosen to represent the Maximum Transmission Unit for some arbitrary datalink. Similarly, the maximum burst size is set to accomodate one MTU. Larger values for σ simply

bottleneck buffer size	1 bandwidth-RTT product
ingress buffer size	1 bandwidth-RTT product
data packet size	1000 bytes
max burst, σ	8000 bits
backoff, β	0.98
mark threshold[a]	$N\sigma$

[a] applies only to EEC

Fig. 4. Simulation parameters

result in larger queue lengths. β was set empirically as a tradeoff between utilization and drain time. The system has been tested with values of β as high as 0.99 and as low as 0.5 with no effect other than the aforementioned tradeoff.

The EEC mark threshold requires that we provision for N active virtual links and allow for one maximum burst to arrive from all N virtual links.

A. Basic Congestion Control Behavior

These simulations demonstrate the basic congestion control characteristics of EEC and IEC schemes. The purpose of the section is to show that IEC closely emulates EEC behavior and also demonstrates impressive absolute performance figures. We use persistent, long-lived (UDP) flows to overload the bottleneck. The results are shown in Figure 6. The table shows that the bottleneck queue length (max and average) in both cases is small because the queues and congestion have been distributed to the edges. In particular the queue lengths stay within one σ per virtual link of the analytically-derived steady-state queue length bound $4N\sigma$ presented in our earlier paper [8]. The queue length bound was derived for EEC but we note that IEC still remains remarkably close to the analytic bound. The other columns in the table show that both IEC and EEC have similar mean bottleneck queue lengths and high utilization. Observe that given the critical size of the buffer, the steady state packet loss rate at the bottleneck is *zero!* Moreover, the distribution of rates is fair to within a standard deviation that is 6% of the mean (i.e., coefficient of variation < 0.05). Further, these properties hold true across a wide range (2-100) in the number of virtual links passing through the bottleneck showing that the method is scalable.

B. TCP Scalability and Edge-based Services

Recall that edge-to-edge control essentially breaks up core congestion and moves them to the edges. Given such

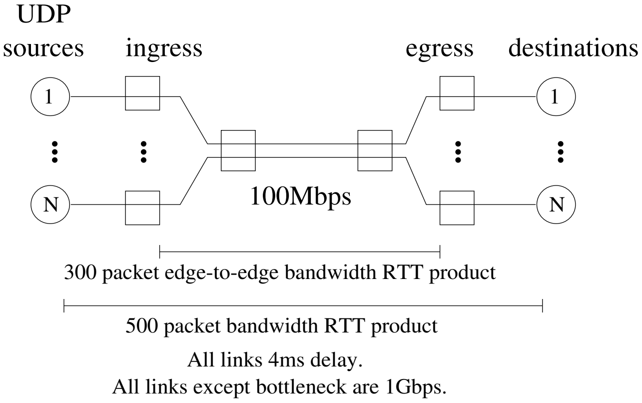


Fig. 5. Topology used in simulations of many IEC and EEC virtual links

behavior, we can place sophisticated control mechanisms at the edge thus avoiding inflicting such complexity on intermediate network nodes, and yet achieving superior performance. We demonstrate this capability by simulating two stateful buffer management schemes at the edge: a) a dropping scheme, FRED[21], i.e. (IEC+FRED), and, b) a rate-control scheme, TCPR (TCP Rate control[16]), i.e. (IEC+TCPR). As with IEC alone, neither IEC+FRED nor IEC+TCPR requires upgrading end-systems or intermediate routers provided intermediate routers support per-class queueing in order to separate edge-controlled and non-edge-controlled traffic. The buffer management integration is done only at the edge.

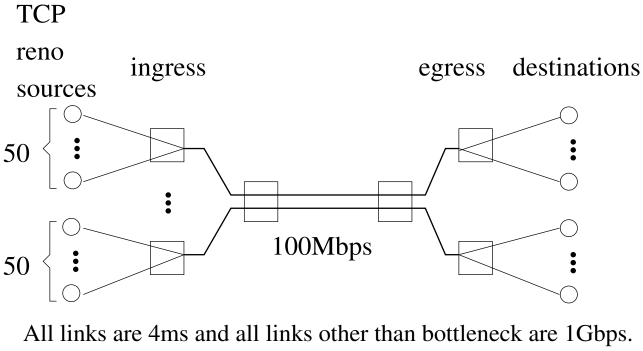


Fig. 7. Topology used in comparing Edge Mechanisms

The IEC+TCPR is an interesting combination because it implements what is known as a "perfect callback", i.e. have sources send packets only when the network has capacity to accept them. Such a perfect callback leads to a *zero-loss, zero-timeout end-to-end TCP service.* In particular, this perfect callback manifests in two stages: first IEC pushes the congestion from the core to the edge and then TCPR pushes the congestion from the edge back to

Virtual[a] Links, N	IEC or EEC	Utilization	Queue Length[b] (avg/max pkts)	Drops[c] (pkts)	C.O.V. Throughput[d]
2	EEC	98.8%	0.874/6	0	0.0348
2	IEC	99.0%	1.01/9	0	0.0098
10	EEC	98.2%	9.28/32	0	0.0453
10	IEC	99.0%	11.3/39	0	0.0157
100	EEC	95.6%	162.7/416	0	0.0583
100	IEC	98.1%	197.5/461	0	0.0252

[a] Statistics collected starting at 10 seconds into a 30 second simulation.
[b] Bottleneck queue length. We ignore ingress queue length for now.
[c] Only drops in intermediate network during steady state. UDP incurs substantial loss at the edges.
[d] Coefficient of Variation of Throughput=$\frac{\text{standard-deviation}}{\text{mean}}$

Fig. 6. Single 100Mbps Bottleneck with Persistent (UDP) Flows

the source. To accomplish this the edge-to-edge virtual link ascertains the available capacity at the bottleneck and provides a (variable) rate-limit to the TCP rate controller (TCPR). The TCP rate controller [16] then converts the rate-limit to the appropriate TCP window size and stamps the window size in the receiver advertised window field in each acknowledgement heading back to the source. It is these aspects of TCPR which does not require packet loss to trigger TCP congestion control [16].

The performance of these options is illustrated in Figure 8 which has four rows. The first two rows illustrates TCP/IP performance when edge-to-edge control is not implemented, but FIFO (first row) or FRED (second row) is implemented at the bottleneck. The third row shows IEC+FRED (FRED at the edge) and the fourth row shows IEC+TCPR (TCPR at the edge). In the last two rows, the intermediate node implements FIFO. Note that the last two rows tradeoff an increased ingress edge queue (fourth column) for total performance in terms of other metrics.

The third row in Figure 8 shows that IEC+FRED provides better best-effort service than the first two rows, i.e., FIFO or FRED implemented in the core. A larger end-to-end delay (58.3 ms vs 39.4 ms) and ingress packet drops (3508 vs 0) is traded off to achieve superior bottleneck queuing and TCP service performance (i.e. lower retransmissions and timeouts). Notice that utilization (third column) is not a differentiating metric in these simulations.

The last row in Figure 8 shows IEC+TCPR operating better for nearly every crucial metric. In particular, all the four latter column entries are zero, i.e., there is *zero packet loss, zero TCP retransmissions and zero TCP timeouts!* This is phenomenal service by any best-effort measure, especially for TCP which traditionally expects packet loss to signal congestion. Also note that the average end-to-end delays (fifth column) which includes transmission and queueing is 22.6 ms. In other words, the total *average queuing delay* at the edge and the core combined is a mere 2.2 ms, i.e. 90% lower than the equivalent value in the first row (canonical case). As a caveat this IEC+TCPR solution has a limited deployment space because the edges must have access to the TCP acknowledgement stream. However, this can be ideally done in a CPE-based VPN deployment.

In summary, our overlay architecture not only enables edge-based better-best-effort services, but also simplifies buffer management requirements at intermediate network nodes.

C. IEC+TCPR with many TCP connections

Using the same topology shown in figure 7, we vary the number of TCP Reno connections and the number of ingress edges. In all simulations, the TCP Reno connections are evenly distributed across the ingress edges. The results are graphed in Figure 9. Note in all of these simulations *IEC+TCPR achieves zero loss, zero retranmissions, and zero retransmission timeouts* in the steady state and the utilization is near 100%.

As noted in [24], with schemes like FIFO or FRED at the core, TCP's fairness begins to degrade significantly once the number of connections exceeds the number of packets in the bandwidth-RTT product (> 500). We show similar degradation for FIFO and FRED cases in our graph plotting coefficient of variation of goodput[1] (figure 9(a)).

[1] Goodput is the number of useful payload bits transmitted through the bottleneck per second, i.e., not including retransmitted payload that has already been received.

Ingress[a] Type	Bneck Type	Util-ization	Ingress Queue[b] (avg/max pkts)	Bneck Queue[c] (avg/max pkts)	Avg Delay[d]	Ingress Drops[e]	Bneck Drops	TCP RTX[f]	TCP RTO[g]
FIFO	FIFO	99.99%	small/1[h]	284.1/500	42.8ms	0	3682	3700	0
FIFO	FRED[i]	100.0%	small/1[h]	243.2/266	39.4ms	0	6389	7863	724
IEC+FRED	FIFO	99.00%	237.3/267	0.861/8	58.3ms	3508	0	4214	230
IEC+TCPR	**FIFO**	**99.06%**	**14.54/54**	**0.731/9**	**22.6ms**	*0*	*0*	*0*	*0*

[a] Statistics gathered for the last 20 seconds of a 30 second simulation
[b] Max and mean queue length across both ingress edges
[c] Bottleneck queue length
[d] Average end-to-end delay incl. propagation and transmission delays. Propagation delay is 20ms. Transmission delays total 0.4ms
[e] Sum drops from both ingress edges.
[f] Total TCP retransmissions
[g] Total TCP retransmission timeouts
[h] Ingress is not saturated and does not implement edge control. So we expect low queue lengths
[i] In all simulations we use FRED parameters as specified in section 4.5 of [21].

Fig. 8. Edge mechanisms for controlling TCP: "FRED at the edge" and "TCPR at the edge"

This occurs because each TCP connection must have at least a window size of four packets to survive a packet loss without incurring a timeout. When this is impossible the rate of timeouts increases (see figure 9(c)).

Since TCP rate control does not rely on loss to control TCP connections, it avoids incurring timeouts or unfairness by evenly reducing each TCP's advertised window size until all TCP connections send with window size 1. At this point, our implementation of TCP rate control does not reduce TCP send rate further. This is a modeling limitation of our simulation which is not reflective of Packeteer's implementation. Thus once the number of flows increases above the bandwidth-RTT product (> 500), our implementation of TCP rate control introduces a persistent backlog and the end-to-end delay begins to increase (see figure 9(c)).

As in the previous section, for a saturated bottleneck and a small number of connections, edge-to-edge control plus TCP rate control achieves an end-to-end delay that is comparable or better than FIFO. However as the number of virtual links increases, so does the delay. This is can at least partially be attributed to increasing bottleneck queue length as number of virtual links increases.

V. Incremental Deployment Strategies

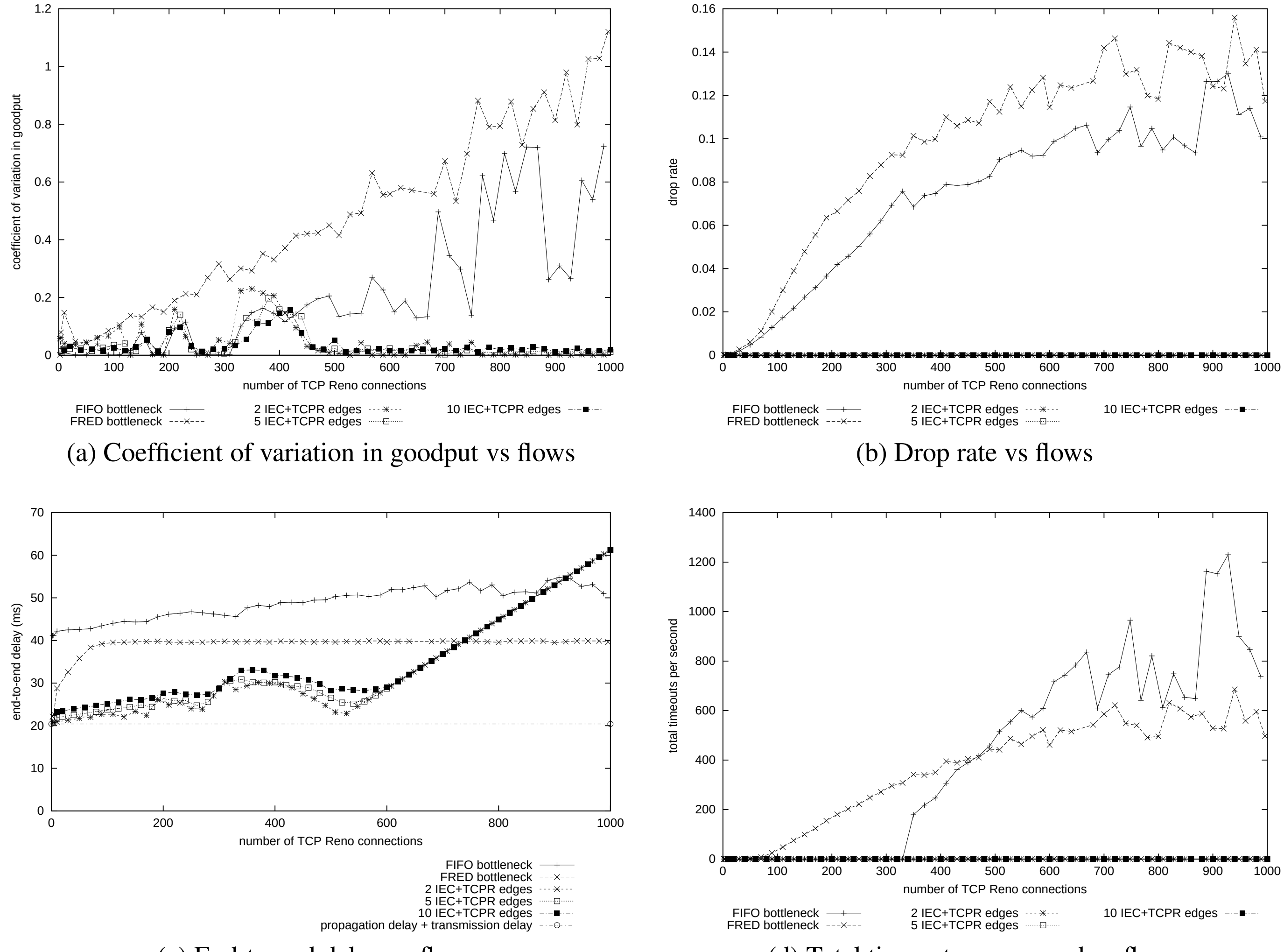
Implementation and deployment flexibility is an important concern in modern protocol design. Questions of interest to ISPs and vendors in our work include:

- the applicability domain of the technology,
- the potential for incremental deployment and what technological aspect drives it.
- the value-proposition for ISP customers

Our overlay architecture is generic, which means that it can be applied to a variety of underlying networks such as: IP, ATM, frame relay, MPLS, X.25 or even over leased lines or SONET circuits. The goal would be to enable novel edge-based services on top of these networks. The list also shows that the technology can fit at either layer 2 or layer 3 in the protocol hierarchy. The architecture can be also leveraged to build customized services for a wide variety of Layer 4 (L4) and application traffic types. This allows a potentially large application domain from a technological viewpoint. Next we show that the architecture allows flexibility in terms of deployment strategies (for ISPs) and implementation options (for vendors).

Incremental deployment is possible on a customer-by-customer basis. Assume a set of networks set aside a class for our architecture. The deployment is then straighforward with virtual circuit based technologies like MPLS, ATM, frame-relay etc. In these cases, the new site-to-site virtual circuits can be mapped to the class set aside, and the edge-to-edge control can be operated over the VCs. With connectionless technologies like IP (diff-serv), there arises the problem of mapping end-to-end (micro-flows) to edge-to-edge virtual links (or loops). In a VPN deployment that uses tunneling (IPSec), the mapping is already done for us. Furthermore, ISPs are unlikely to apply multi-path routing within a tunnel because of the adverse affects that packet reordering has on TCP, thus this also satisfies the single-path requirement.

From an implementation (vendor) perspective, the technology can be implemented either in the data-plane (i.e.



(a) Coefficient of variation in goodput vs flows

(b) Drop rate vs flows

(c) End-to-end delay vs flows

(d) Total timeouts per second vs flows

Fig. 9. IEC plus TCP Rate Control under heavy loads (10Mbps bottleneck)

forwarding path of information flows) or in the control-plane (i.e. on management servers interfacing to data-plane components). A data plane solution could be a new "services box" or an integration of this technology with "access" or "edge" products. The latter solution avoids duplication of hardware platforms. The control-plane solution would interface to data-plane components through command-line interfaces (CLI). The performance and service capabilities of this alternative would depend upon the richness and responsiveness of the CLI.

Regarding the value-proposition, our core technology addresses the problem of congestion control scalability and provides a basis for edge-based bandwidth services. While ISPs have abundant core bandwidth there are still key niches (access links, aggregation points, tail circuits, international circuits and peering points) where the congestion problem exist and affect end-user performance. The goal of this technology is to overlay multiple loops around these points and move the congestion away to manageable points (edges) as illustrated in the paper. For example, congestion at all of the key locations can be captured by virtual links spanning Customer Premise Equipment (CPE) boxes.

From a services perspective, ISPs still face the problem of packaging their abundant bandwidth resources into high-value end-to-end services (SLAs). Current QoS and services technology (eg: diffserv [1] etc) requires fairly long development and deployment cycles. Our technology provides the foundation for edge-based services which can be deployed and updated rapidly. For example, for better best effort and L4 (eg: TCP) services, ISPs need only set aside a queue at potential bottlenecks. ISPs could then

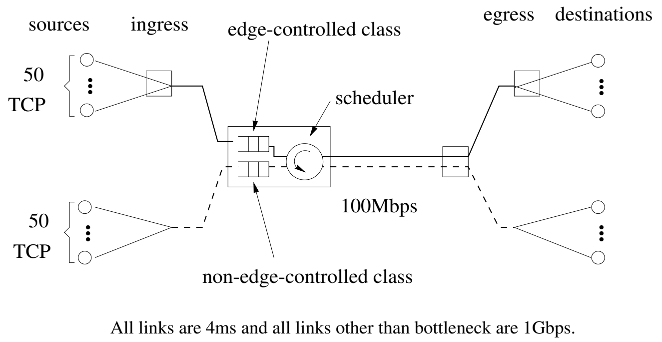


Fig. 10. Class of Service Topology

extend and develop a larger base of services, possibly including dynamically contracted services, in the future (eg: [26]).

A. *Validation of the Basic Deployment Approach*

This section provides a sample simulation result to validate the deployment approach of requiring class-based intermediate network(s) where the edge-to-edge control architecture can co-exist with traditional TCP traffic. Figure 10 show a bottleneck with two classes (queues) serviced by a round robin scheduler. One class is for the edge-controlled traffic and the other for best effort. In this simulation our fair scheduler provides 50% of the bandwidth to each traffic class.

The results in figure 11 demonstrate that the distribution of capacity between the two classes is reasonably fair (46.8 Mbps edge-controlled versus 53.2 Mbps non-edge-controlled). The reason the split is not accurate is because the edge-to-edge control sometimes leads to zero queues and the capacity is captured by the uncontrolled class. During such a saturation period, the edge-to-edge control also backs off based upon measured egress rates.

In other words, the mapping of edge-controlled traffic aggregates on top of a class leads to a slight beat down of the capacity allocated to the class. Simulations show that a 10% change in configuration parameters (53% vs 47% allocation) compensates for this situation. More importantly, observe that edge-to-edge controlled TCPs can co-exist (though isolated class-wise) with non-edge-to-edge controlled TCPs on a single bottleneck. Moreoever, when compared against the non-edge-controlled traffic, the edge-controlled class obtains *97% less excess delay*, *98.7% lower coefficient of variation in goodput* between TCP connections, as well as zero retransmissions and zero transmission timeouts, i.e., superior best effort performance.

VI. Future Work

The edge-to-edge overlay control architecture is more than congestion control and is the basis for a future *service platform*. Another key contribution of this work is to achieve these benifits without requiring support from intermediate network nodes or requiring upgrades to end-systems. In this paper we have only demonstrated its usefulness for providing a better-best-effort service. We intend to broaden the service menu in future work. For example, we have initial results which suggest that we can provision the equivalent of a simple assured service (i.e. minimum bandwidth assurance) completely edge-based using this architecture. In the future, schedulers at the ingress might distribute a virtual link's capacity according to application-level policy, or the schedulers might provide managed delay differentiation for interactive traffic. In the context of DiffServ, edge-control might inter-operate with the Assured Forwarding Per-Hop-Behavior [10] by performing all drop-preference at the ingress edge.

Future research will also have to address issues such as implementing edge-to-edge control and end-to-end SLAs across multiple administrative domains. Also, to broaden the appeal of edge-to-edge control as a traffic management tool, we might study hierarchies of overlaid virtual links. We will also investigate issues surrounding end-to-end deployment, though we do not see that as our core focus. Our current efforts include prototyping this architecture on a testbed and work with Internet2 on deployment and testing initiatives.

References

[1] Blake, S., Black, D., Carlson, M., Davies, E., Wang, Z., and Weiss, W. An architecture for differentiated services. IETF RFC 2474, December 1998.

[2] Chiu, D., and Jain, R. Analysis of the increase/decrease algorithms for congestion avoidance in computer networks. *Journal of Computer Networks 17*, 1 (June 1989), 1–14.

[3] Clark, D. D., Lambert, M. L., and Zhang, L. NETBLT: A high throughput transport protocol. In *SIGCOMM Symposium on Communications Architectures and Protocols* (August 1987), pp. 353–359.

[4] Cruz, R. L. *A Calculus for Network Delay and a Note on Topologies of Interconnection Networks.* PhD thesis, University of Illinois, July 1987.

[5] Endpoint congestion management working group. http://www.ietf.org/html.charters/ecm-charter.html.

[6] Floyd, S. TCP and Explicit Congestion Notification. *ACM Computer Communication Review 24*, 5 (October 1994), 10–23.

[7] Floyd, S., and Jacobson, V. Random Early Detection Gateways for Congestion Avoidance. *IEEE/ACM Transactions on Networking 1*, 4 (August 1993), 397–413.

Class[a]	Throughput (Mbps)	Ingress Queue (avg/max pkts)	Bneck Queue (avg/max pkts)	Avg Delay[b]	Excess Delay [c]	TCP RTX[d]	TCP RTO[e]	C.O.V. Goodput
IEC[f]+ TCPR	46.8	11.4/55	0.38/5	22.31ms	1.91ms	0	0	0.00223
BE[g] TCP	53.2	n/a	45.3/500[h]	85.99ms	65.59ms	849	67	0.166

[a] Statistics for the last 20 seconds of a 30 second simulation.
[b] End-to-end delay
[c] Sum of queueing delays at ingress and bottleneck
[d] Total TCP retransmissions within class
[e] Total TCP retransmission timeouts within class
[f] Edge-controlled class using IEC
[g] Best effort, i.e., non-edge-controlled class
[h] The buffer is 500 packets

Fig. 11. Class of Service Results

[8] HARRISON, D., AND KALYANARAMAN, S. Edge-to-edge traffic control for the internet. Submitted to ICNP 2000, November 2000.

[9] HARRISON, D., AND KALYANARAMAN, S. Edge-to-edge traffic control for the internet. Tech. rep., Rensselaer Polytechnic Institute, http://networks.ecse.rpi.edu/~harrisod/, April 2000.

[10] HEINANEN, J., FINLAND, T., BAKER, F., WIESS, W., AND WROCLAWSKI, J. Assured forwarding phb group. Internet RFC 2597, June 1999. Defines per-hop behaviors building blocks for assured services.

[11] JACOBSON, V. Congestion avoidance and control. In *Proceedings of the SIGCOMM'88 Symposium* (August 1988), pp. 314–332.

[12] JACOBSON, V., AND BRADEN, R. Tcp extensions for long-delay paths. IETF RFC 1072, October 1988. Discusses TCP SACK.

[13] JAIN, R. A delay based approach for congestion avoidance in interconnected hereogeneous computer networks. *Computer Communications Review, ACM SIGCOMM* (October 1989), 56–71.

[14] KALYANARAMAN, S. Traffic Management for the Available Bit Rate (ABR) Service in Asynchronous Transfer Mode (ATM) network. *Ph.D. Dissertation, Dept. of Computer and Information Sciences, The Ohio State University* (August 1997).

[15] KALYANARAMAN, S., JAIN, R., FAHMY, S., GOYAL, R., AND VANDALORE, B. The ERICA switch algorithm for ABR traffic management in ATM networks. *IEEE/ACM Transactions on Networking 8*, 1 (February 2000), 87–98.

[16] KARANDIKAR, S., KALYANARAMAN, S., BAGAL, P., AND PACKER, B. TCP rate control. *Computer Communication Review 30*, 1 (2000).

[17] KENT, S., AND ATKINSON, R. Security architecture for the internet protocol. RFC 2401, November 1998.

[18] KESHAV, S. A control-theoretic approach to flow control. In *Proceedings of ACM SIGCOMM '91* (September 1991).

[19] KUNG, H., BLACKWELL, T., AND CHAPMAN, A. Credit-based flow control for ATM networks: Credit update protocol, adaptive credit allocation, and statistical multiplexing. In *Proceedings of ACM SIGCOMM '94* (September 1994), pp. 101–114.

[20] L. BRAKMO, S. O., AND PETERSON, L. TCP Vegas: New Techniques for Congestion Detection and Avoidance. *SIGCOMM '94* (August 1994), 24–35.

[21] LIN, D., AND MORRIS, R. Dynamics of Random Early Detection. *SIGCOMM'97* (August 1997).

[22] LO MONACO, G., FEROZ, A., AND KALYANARAMAN, S. Tcp friendly marking for scalable and "better" best-effort services on the internet. Submitted to ICNP 2000, November 2000.

[23] MATHIS, M., AND MAHDAVI, J. Forward acknowledgement: Refining tcp congestion control. In *Proceedings of ACM SIGCOMM '96* (1996).

[24] MORRIS, R. TCP behavior with many flows. In *Proceedings of ICNP '97* (October 1997).

[25] RAMAKRISHNAN, K. K., AND JAIN, R. A binary feedback scheme for congestion avoidance in computer networks with a connectionless network layer. In *Proceedings of ACM SIGCOMM '88* (August 1988), pp. 303–313.

[26] SINHA, R., KALYANARAMAN, S., AND RAVICHANDRAN, T. Dynamic capacity contraction: A framework for congestion-sensitive pricing. Submitted to ICC'2000, 2000.

[27] VILLAMIZAR, C., AND SONG, C. High performance tcp in ansnet. *ACM Computer Communication Review* (1995).