# Modeling Congestion in Backbone Routers

Jim Gast and Paul Barford
University of Wisconsin - Madison, Computer Science, 1210 West Dayton Street
Madison, WI 53706

{jgast,pb}@cs.wisc.edu

UW-Madison CS Technical Report 1451

**Abstract**

Traffic engineering models based on end-to-end loss probabilities and delays do not scale well to fast backbone links. In this paper, we investigate the nature of congestion events in highly aggregated flows. An examination of congestion events shows distinct phases of queue buildup, packet dropping followed later by TCP reaction, and queue clearing. Evidence of the existence and characteristics of these discrete congestion events is presented using active probe data gathered by the Surveyor project. When connections pass through periodic congestion, the aggregate offered load to neighboring links rises and falls in cadence with the congestion events. We named this group of connections a ”flock” and investigate the implications. Flock formation, distinct from the prior notion of synchronization, can scale to larger numbers of connections depending on the diversity of roundtrip times present. Through simulation we illustrate conditions under which flocking occurs. Unlike prior end-to-end studies, this paper takes the viewpoint of a small number of hops along a long path. A model is presented that predicts important characteristics of the congestion events including the quantity, intensity, and duration. Our results suggest that backbone congestion can be modeled as an interaction between flocks of connections. The model effectively explains the prevalence of discrete congestion events in fast links with high multiplexing factors as well as the phases of congestion.

## I. Introduction

Much of the research in network congestion control focused on the ways in which transport protocols react to packet losses. Prior analyses were frequently conducted in simulation environments with small numbers of competing flows, and along paths that have a single low bandwidth bottleneck. In contrast, modern routers deployed in the Internet easily handle thousands of simultaneous connections along hops with capacities above a billion bits per second.

Since packet loss is still the major mechanism for communicating congestion in the interior of the network, characteristics of losses and bursts of losses remain important. Poisson models were tried and rejected [1], [2]. Fractals or Self-Similarity [3], [4], [5] have been exploited for their ability to explain Internet traffic statistics. These models show that large timescale traffic variability arises from exogenous forces (the composition of the network traffic that arrives) rather than just endogenous forces (reaction of the senders to feedback given to them from the interior). These models fall short in providing insights to improve traffic engineering. In part, the problem is hard simply because the environment has such a rich set of targets to study.

The model presented in this paper is a purely endogenous view. The most significant effort in this regard is RED [6]. Our objective in this paper is provide mechanisms to allow us to investigate the duration, intensity and periodicity of congestion events. Our model is based on identifying distinct portions of a congestion event, predicting the shape of congestion events and the gap between them. Our congestion model is developed from the perspective of queue sizes during congestion events that have a shape we call a “shark fin”. Packets that try to pass through a congested link during a packet dropping episode are either dropped or placed at the end of an (almost) full queue. While this shape is familiar in both analytical and simulation studies of congestion, its characteristics in measurement studies have not been robustly reported.

Research supported by the Anthony C. Klug fellowship in Computer Science
Research supported by NSF ANI-0085984

Data collected with the Surveyor infrastructure shows evidence that these shark fins exist in the Internet. There are distinct spikes at very specific queue delay values that only appear on paths that pass through particular links. This validation required highly accurate one-way delay measurements taken over a four month test spread over a wide geographic scope.

In the presence of regularly spaced congestion events, connections shrink their congestion windows (cWnd) in cadence with the congestion events. The cWnd's slowly grow back between events. In effect, the well-known saw-tooth graphs of cWnd [7] for the individual long-lived connections are brought into phase with each other, forming a "flock". We discuss the distinction between a flock and the common notion of synchronization in Section 2.

From the viewpoint of a neighboring link, a flock will offer an aggregate load that rises together. When it reaches a ceiling (at the original hop) the entire flock will lower its cWnd together. We believe flocking can be used to explain synchronization of much larger collections of connections than any prior research.

We investigate a spectrum of congestion issues related to our model in a series of ns2 simulations. We explore the accuracy of our model over a broad range of offered loads, mixtures of RTT's, and multiplexing factors. In simulation, congestion event statistics are compared to the output of the model and demonstrate an improved understanding of the duration of congestion events.

Future active queue management could make use of the improved understanding of flocking to avoid global synchronization and to mitigate cases where transport protocols get poor performance when passing through multiple congested gateways.

## II. Other Related Work

Packet delay and loss behavior in the Internet has been widely studied. Examples include [8] which established basic properties of end-to-end packet delay and loss based on analysis of active probe measurements between two Internet hosts. That work is similar to ours in terms of evaluating different aspects of packet delay distributions. Paxson provided one of the most thorough studies of packet dynamics in the wide area in [9]. While that work treats a broad range of end-to-end behaviors, the sections that are most relevant to our work are the statistical characterizations of delays and loss. The important aspects of scaling and correlation structures in local and wide area packet traces are established in [3], [1]. Feldmann *et al.* investigate multifractal behavior of packet traffic in [5]. That simulation-based work identifies important scaling characteristics of packet traffic at both short and long timescales. Yajnik *et al.* evaluated correlation structures in loss events and developed Markov models for temporal dependence structures [10]. Recent work by Zhang *et al.* assesses three different aspects of *constancy* in delay and loss rates.

There are a number of widely deployed measurement infrastructures which actively measure wide area network characteristics [11], [12], [13]. These infrastructures use a variety of active probe tools to measure loss, delay, connectivity and routing from an end-to-end perspective. Recent work by Pasztor and Veitch identifies limitations in active measurements, and proposes an infrastructure using the Global Positioning System (GPS) as a means for improving accuracy of active probes [14]. That infrastructure is quite similar to Surveyor [11] which was used to gather data used in our study.

Internet topology and routing characteristics have also been widely studied. Examples include [15], [16], [17]. These studies inform our work with respect to the structural characteristics of end-to-end Internet paths.

A variety of methods have been employed to model network packet traffic including queuing and auto-regressive techniques [18]. While these models can be parameterized to recreate observed packet traffic time series, parameters for these models often do not relate to network properties. Models for TCP throughput have also been developed in [19], [20], [21]. These models use RTT and packet loss rates to predict throughput, and are based on characteristics of TCP's different operating regimes. Our work complements these in terms of providing a more robust perspective on the delay and loss values used to parameterize the models.

The tendency of traffic to synchronize was first reported by Floyd and Jacobson [22]. Their study found resonance at the packet level when packets arrived at gateways from two nearly equal senders. Deterministic queue management algorithms like drop-tail could systematically discriminate against some connections. It formed the earliest arguments in favor of Random Early Detection (RED). RED breaks the pattern at the packet level by probabilistically dropping packets when congestion is imminent. In contrast, our notion of flocking is the alignment of congestion window saw-tooth behavior. Packet level resonance was never shown to extend to more than a few connections.

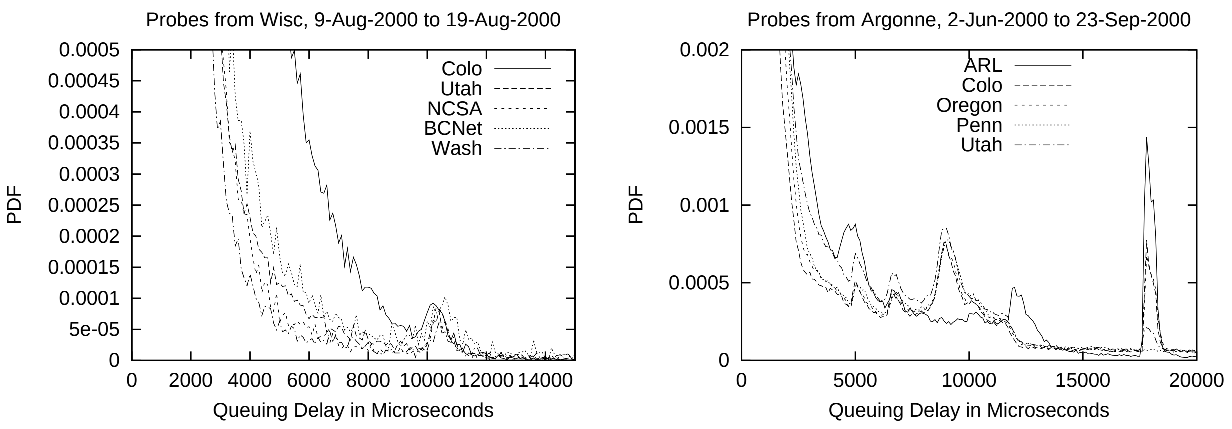


Fig. 1. Left: Paths with a shared initial segment show a peak at 10.3 ms. Right: Peaks also show up in other paths, at other characteristic queue delay times.

Qui, Zhang and Keshav [23] found that global synchronization can result when a small number of connections share a bottleneck at a slow link with a large buffer, independent of the mixture of RTTs. Increasing the number of connections prevents the resonance. Flocking is the opposite. Flocking scales to large numbers of connections, but a broad mixture of RTTs prevents the resonance.

## III. Surveyor Data: Looking for Characteristics of Queuing

Empirical data for this study was collected using the Surveyor [11] infrastructure. Surveyor consists of 60 nodes placed around the world in support of the work of the IETF IP Performance Metrics Working Group [24]. The data we used is a set of active one-way delay measurements taken during the period from 3-June-2000 to 19-Sept-2000. Each of the 60 Surveyor nodes maintains a measurement session to each other node. A session consists of an initial handshake to agree on parameters followed by an long stream of 40 byte probes at random intervals with a Poisson distribution and a mean interval between packets of 500 milliseconds. The packets themselves are Type-P UDP packets of 40 bytes. The sender emits packets containing the GPS-derived timestamp along with a sequence number. See RFC 2679 [25]. The destination node also has a GPS and records the one-way delay and the time the packet was sent. The graphs in Figure 1 show the PDF of queuing delay from 2 sample sites.

Each probe's time of day is reported precise to 100 microseconds and each probe's delay is accurate to $\pm$ 50 microseconds. Data is gathered in sessions that last no longer than 24 hours.

The delay data are supplemented by traceroute data using a separate mechanism. Traceroutes are taken in the full mesh approximately every 10 minutes. For this study, the traceroute data was used to find the sequence of at least 100 days that had the fewest route changes.

### *A. Deriving Propagation Delay*

The Surveyor database contains the entire delay seen by probes. Before we can begin to compare delay times between two paths we must subtract propagation delay fundamental to each path. For each session, we assume that the smallest delay seen by that session is the propagation delay between source and destination along that route. Any remaining delay is assumed to be queuing delay. Sessions were discarded if traceroutes changed or if any set of 500 contiguous samples had a local minimum that was more than 0.4 ms larger than the propagation delay.

### *B. Peaks in the Queuing Delay Distribution*

Figure 1 shows a variety of paths coming from one site. They all share one OC-3 interface (155 Mbps) at the beginning and have little in common after that. The Y-axis of this graph represents the number of probes that experienced the same one-way delay value (adjusted for propagation delay). Counts are normalized so that the size of the curves can be easily compared. Each histogram bin is 100 microseconds of delay wide.

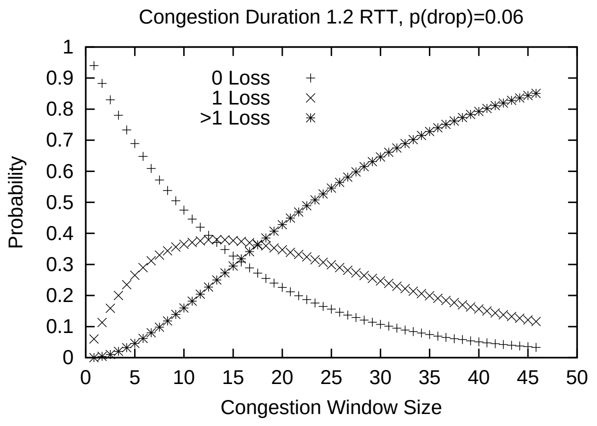


Fig. 2. Showing the probability of losing 0, exactly 1, or more than one packet in a single congestion event as a function of cWnd.

Our conjecture was that a full queue in the out-bound link leaving that site was 10.3 milliseconds long, and that probes were likely to see almost empty queues (outside of congestion events) and almost full queues (during congestion events).

The right graph in Figure 1 shows that this phenomenon is neither unique nor rare. The left graph shows that other paths see other delays and complex paths with many hops see more than one peak in their queuing delay distribution.

## C. Other Potential Causes Of Peaks

Peaks in the PDF might be caused by measurement anomalies other than the congestion events proposed in this paper. Hidden (non-queuing) changes could come from the source, the destination, or along the path. Path hidden changes could be caused by load balancing at layer 2. If the load-balancing paths have different propagation delays, the difference will look like a peak. Or ISPs could be introducing intentional delays for rate limiting or traffic shaping. Or there could be delays involved when link cards are busy with some other task (e.g. routing table updates, called the coffee break effect [26] ). Our data does not rule out the possibility that we might be measuring some phenomenon other than queuing delay.

Hidden source or destination changes could be caused by other user level processes or by sudden changes in the GPS reported time. For example, the time it takes to write a record to disk could be several milliseconds by itself. The Surveyor software is designed to use non-blocking mechanisms for all long delays, but occasionally the processes still see out-of-range delays. The Surveyor infrastructure contains several safeguards that discard packets that are likely to have hidden delay. For more information see [27].

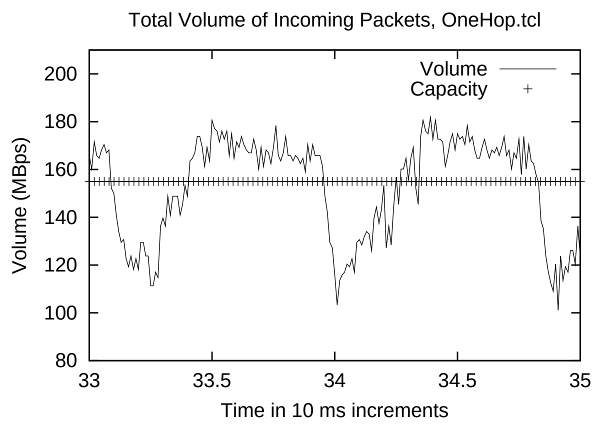
## IV. Window Size Model

We now construct a cWnd feedback model that predicts the reaction of a group of connections to a congestion event.

Assume that a packet is dropped at time $t_0$. The sender will be unaware of the loss until one reaction time, $R$, later. Let $C$ be the capacity of the link. Before the sender can react to the losses, ($C$ / $R$) packets will depart. During that period, packets are arriving at a rate that consistently exceeds the departure rate. It is important to note that the arrival rate has been trained by prior congestion events. If the arrival rate grew slowly, it has reached a level only slightly higher than the departure rate. For each packet dropped, many subsequent packets will see a queue that has enough room to hold one packet. This condition persists until the difference between the arrival rate and the departure rate causes another drop.

Figure 2 shows the probability that a given connection will see $\ell$ losses from a single congestion event. This example graph shows the probabilities when passing packets through a congestion event with 0.06 loss rate, $L$. Here $R$ is assumed to be 1.2 RTT. Each connection with a congestion window, $W$, will try to send $W$ packets per RTT through the congestion event. We now compute the post-event congestion window, $W'$.

With probability $p(NoLoss)$, a connection will lose no packets at all. Its packets will have seen increasing delays during queue buildup and stable delays during the congestion event. Their ending $W'$ will be $W + R/RTT$. This observation contrasts with analytic models of queuing that assume all packets are lost when a queue is “full”.



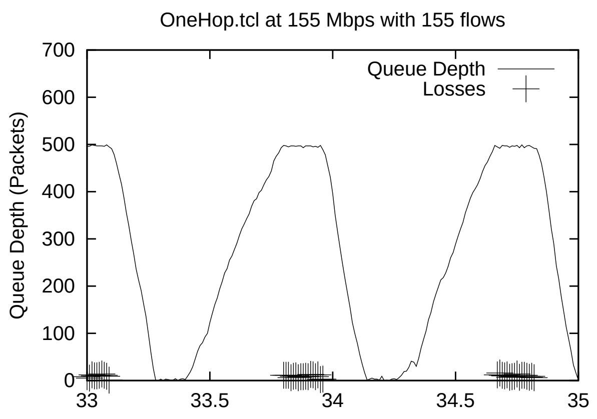


Fig. 3. Mbps of traffic destined for a congested link.

With probability $p(OneLoss)$ a connection will experience exactly 1 loss and will back off. The typical deceleration makes $W'$ be $W/2$.

With probability $p(Many)$, a connection will see more than one loss. In this example, a connection with $cWnd$ 40 is 80% likely to see more than one loss. Some connections react with simple multiplicative decrease (halving their congestion window). TCP Reno connections might think the losses were in separate round trip times and cut their volume to one fourth. Many connections (especially connections still in slow start) completely stop sending until a coarse timeout. For this model, we simply assume $W'$ is $W/2$.

$$W' = p(NoLoss) * (W + \frac{R}{RTT}) + (p(OneLoss) + p(Many)) * \frac{W}{2}$$

## V. Simulations

We use a series of $ns2$ simulations to understand congestion behavior details and have made them available for public download at [28].

### A. One Hop Simulation

We begin with a simulation of the widely used dumbbell topology to highlight the basic features of our model. All of the relevant queuing delay occurs at a single hop. There are 155 connections competing for a 155 Mbps link. We use infinitely long FTP sessions with packet size 1420 bytes and a dedicated 2 Mbps link to give them a ceiling of 2 Mbps each. To avoid initial synchronization, we stagger the FTP starting times among the first 10 ms. End-to-end propagation delay is set to 50 ms. The queue being monitored is a 500 packet drop-tail queue feeding the dumbbell link.

### B. Portions of the Shark Fin

The right graph in Figure 3 shows two and a half complete cycles that look like shark fins. Our model is based on the distinct sections of that fin:

- **Clear**: While the incoming volume is lower than the capacity of the link, Figure 3 shows a cleared queue with small queuing delays. Because the graph here looks like grass compared to the delays associated with congestion, we refer to the queuing delays as "grassy". This situation persists until the total of the incoming volumes along all paths reaches the outbound link's capacity.
- **Rising**: Clients experience increasing queuing delays during the "rising" portion of Figure 3. The shape of this portion of the curve is close to a straight line (assuming acceleration is small compared to volume). The "rising" portion of the graph has a slope that depends on the acceleration and a height that depends on the queue size and queue management policy of the router.
- **Congested**: Drop-tail routers will only drop packets during the congested state. This portion of Figure 3 has a duration heavily influenced by the average reaction time of the flows.

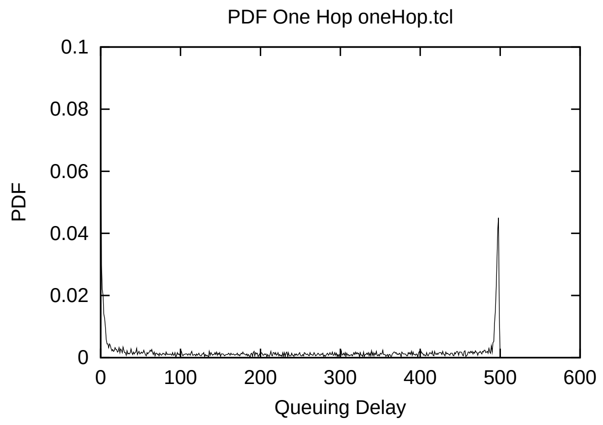


Fig. 4. Mbps of traffic destined for a congested link.

Because the congested state is long, many connections had time to receive negative feedback (packet dropping). Because the congested state is relatively constant duration, the amount of negative feedback any particular connection receives is relatively independent of the multiplexing factor, outbound link speed, and queue depth. The major factor determining the number of packets a connection will lose is its congestion window size.

- **Falling**: After senders react, the queue drains. If an aggregate flow contains many connections in their initial slow start phase, those connections will, in the aggregate, show a quiet period after a congestion event. During this quiet period, many connections have slowed down and a significant number of connections have gone completely silent waiting for a timeout.

## *C. PDF of Queuing Delay for One Hop Simulation*

Figure 4 shows the PDF of queue depths during the One Hop simulation. This graph shows distinct sections for the grassy portion ($d_0$ to approximately $d_{50}$), the sum of the rising and falling portions (histogram bars for the equiprobable values from $d_{50}$ to $d_{480}$) and the point mass at 36.65 ms when the delay was the full 500 packets at 1420 bytes per packet feeding a 155 Mbps link.

By adding or subtracting connections, changing the ceiling for some of the traffic or introducing short-term connections, we can change the length of the period between shark fins, the slope of the line rising toward the congestion event, or the slope of the falling line as the queue empties. But the basic shape of the shark fin remains over a surprisingly large range of values and the duration of intense packet dropping (the congestion event) remains most heavily influenced by the average round trip time of the traffic.

## *D. Two Hop Simulation*

To further refine our model and to understand our empirical data in detail, we extend our simulation environment to include an additional core router between the ingress and the egress. Both links are 155 Mbps and both queues are drop-tail. To make it easy to distinguish between the shark fins, the queue from ingress to core holds 100 packets but the queue from core to egress holds 200 packets. Test traffic was set to be 15 long-term connections passing through ingress to egress. We also added cross traffic composed of both web traffic and longer connections. The web traffic is simulated with NS2's PagePool application WebTraf. The cross traffic introduced at any link exits immediately after that link.

The left graph in Figure 5 shows the sum of the two queue depths as the solid line. Shark fins are still clearly present and it is easy to pick out the fins related to congestion at the core router at queue depth 200 as distinct from the fins that reach a plateau at queue depth 100.

The stars along the bottom of the graph are dropped packets. Although the drops come from different sources, each congestion event maintains a duration strongly related to the reaction time of the flows. In this example, one fin (at time $t_{267}$) occurred when both the ingress and core routers were in a rising delay regime. Here the dashed mid-delay line shows the queue depth at the ingress router. At most other places, the mid-delay is either very nearly zero or very nearly the same as the sum of the ingress and core delays.

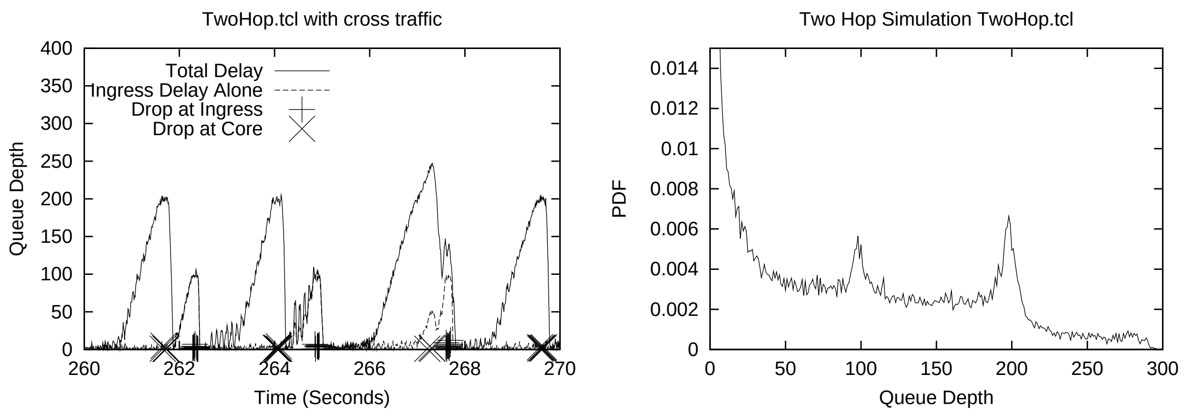


Fig. 5. Both signatures appear when queues of size 100 and 200 are used in a 2-hop path.

The right graph of Figure 5 shows the PDF of queue delays. Peaks are present at a queue depths of 100 packets and 200 packets. This diagram also shows a much higher incidence of delays in the range 0 to 100 packet times due to the cross traffic and the effect of adding a second hop. In terms of our model, this portion of the PDF is almost completely dictated by the packets that saw grassy behavior at both routers. The short, flat section around 150 includes influences from both rising regime at the ingress and rising regime at the core. The falling edges of shark fins were so sharp in this example that their influence is negligible. The peak around queue size=100 is 7.6 ms. It is not as sharp as the One Hop simulation in part because its falling edge includes clear delays from the core router. For example, a 7.6 ms delay might have come from 7.3 ms spent in the ingress router plus 0.3 ms spent in the core. The next flat area from 120 to 180 is primarily packets that saw a rising regime at the core router. A significant number of packets (those with a delay of 250 packet times, for example) were unlucky enough to see rising regime at the ingress and congestion at the core or a rising regime at the core and congestion at the egress.

## *E. Flocking*

In the absence of congestion at a shared link, individual connections would each have had their own saw-tooth graph for cWnd. A connection's cWnd (in combination with it's RTT) will dictate the amount of load it offers at each link along it's path. Each of those saw-tooth graphs has a ceiling, a floor, and a period. Assuming a mixture of RTT's, the periods will be mixed. Assuming independence, each connection will be in a different phase of its saw-tooth at any given moment. If $N$ connections meet at an uncongested link, the $N$ saw-tooth graphs will sum to a comparatively flat graph. As $N$ gets larger (assuming the $N$ connections are independent) the sum will get flatter.

During a congestion event, many of the connections that pass through the link receive negative feedback at essentially the same time. If (as is suggested in this paper) congestion events are periodic, that entire group of connections will tend to reset to their lower cWnd in cadence with the periodic congestion events. Connections with saw-tooth graphs that resonate with the congestion events will be drawn into phase with it and with each other.

This phenomenon is very similar to global synchronization. Global synchronization affects all connections passing through a common congestion point regardless of RTT but depends on the buffer (plus any packets resident in the link itself) being large enough to hold 3 packets per connection. Keshav, et al. [23] study global synchronization from the viewpoint of the packets still in the queue when the drops occur. In global synchronization increasing the number of connections would eliminate the synchronization. Flock formation theory does not depend on large buffers or slow links, but rather it depends on a mixture of RTTs that are close enough to be compatible.

## *F. Flock Formation*

To demonstrate a common situation in which flocks form we construct the dumbbell environment shown in Figure 6. Each of the legs feeding the dumbbell runs at 100 Mbps, while the dumbbell itself is a 155 Mbps link.

We gave each leg entering node 0 a particular propagation delay so that all traffic going through the first leg has a RoundTrip Time of 41 ms. The second leg has traffic with RTT 47 ms, and the final leg has traffic at 74 ms RTT.

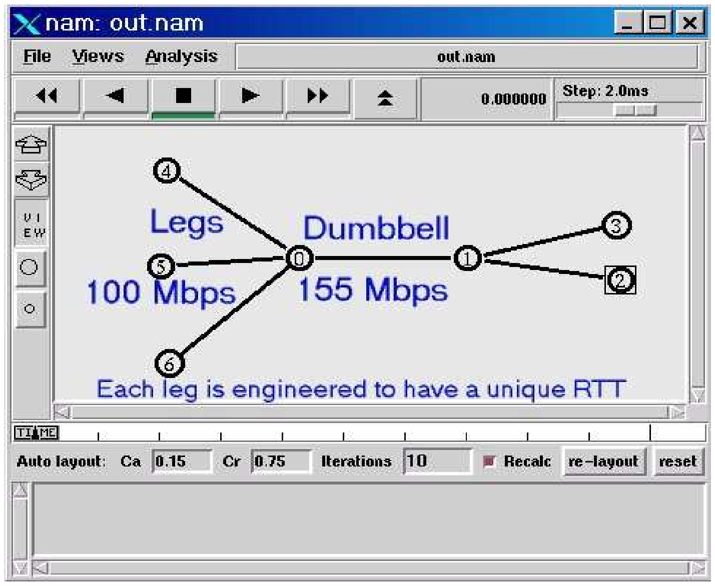


Fig. 6. Simulation environment to foster formation of flocks.

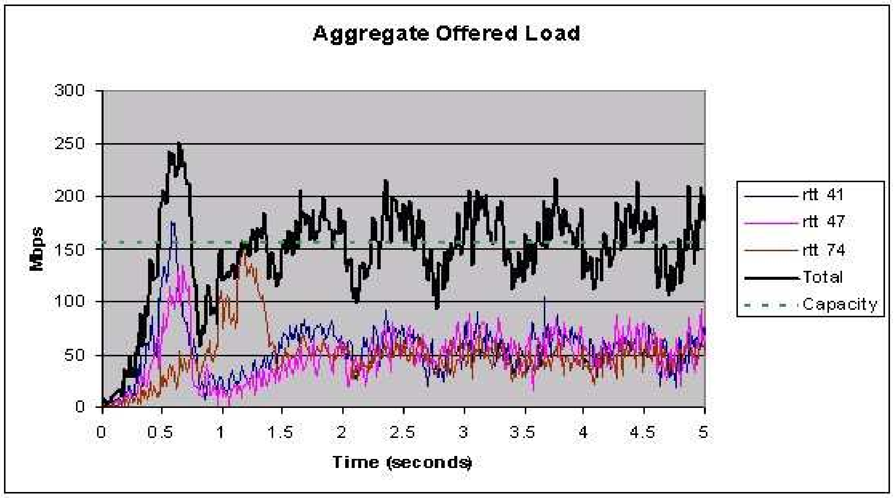


Fig. 7. Connections started at random times synchronize cWnd decline and buildup after 2 seconds.

Figure 7 shows the number of packets coming out of the legs and the total number of packets arriving at the ingress. Congestion events happen at 0.6 sec, 1.3 sec and 1.8 sec. As a result of those congestion events, almost all of the connections, regardless of their RTT, are starting with a low cWnd at 2.1 seconds. After that, the dumbbell has a congestion event every 760 milliseconds, and the traffic it presents to subsequent links rises and falls at that cadence.

Not shown is the way in which packets in excess of 155 Mbps are spread (delayed by queuing) as they pass through. The flock at RTT 74 ms is slow to join the flock, but soon falls into cadence. Effectively, the load the dumbbell passes on to subsequent links is a flock, but with many more connections and a broader range of RTT's.

## *G. Range of RTT Values in a Flock*

Next we investigate how RTT values affect flocking. We use the same experimental layout shown in Figure 6 except that a fourth leg has been added that has an RTT too long to participate in the flock formed at the dumbbell. Losses from the dumbbell come far out of phase with the range that can be accommodated by a connection with a 93 millisecond RTT.

Figure 8 shows the result of 240 simulation experiments. Each run added one connection in round-robin fashion to the various legs. When there is no contention at the dumbbell, each connection gets goodput limited only by the 100 Mbps leg. The graph plots the total goodput and the goodput for each value of RTT.

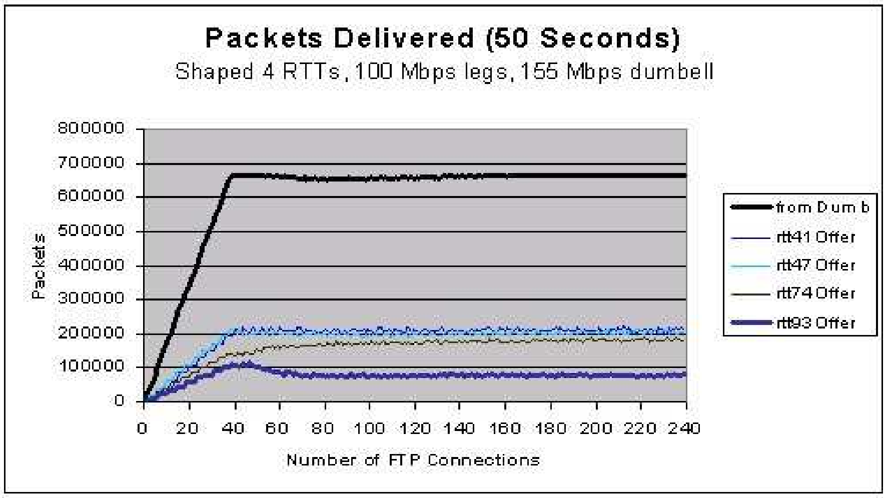


Fig. 8. Connections with RTT slightly too long to join flock.

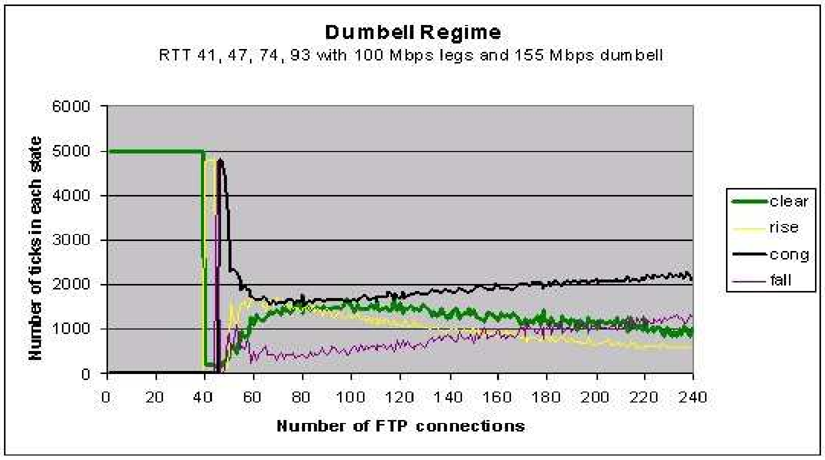


Fig. 9. Proportion of time spent in each queue regime.

The result is that the number of packets delivered per second by the 93 ms RTT connections is only about half that of the 74 ms group. In some definitions of fair distribution of bandwidth, each connection would have delivered the same number of packets per second, regardless of RTT.

This phenomenon is similar to the TCP bias against connections with long RTT reported by Floyd, et al. [29], but encompasses an entire flock of connections.

It should also be noted that turbulence at an aggregation point (like the dumbbell in this example) causes incoming links to be more or less busy based on the extent to which the traffic in the leg harmonizes with the flock formed by the dumbbell. In the example in Figure 8, the link carrying 93 millisecond RTT traffic had a capacity of 100 Mbps. In the experiments with 20 connections per leg (80 connections total), this link only achieved 21 Mbps. Increasing the number of connections did nothing to increase that leg's share of the dumbbell's capacity.

## *H. Formation of Congestion Events*

The nature of congestion events can be most easily seen by watching the amount of time spent in each of the queuing regimes at the dumbbell. We now examine the proportion of time spent in each portion of the shark fin using the same simulation configuration as in the prior section.

Figure 9 shows the proportion of time spent in the clear (no significant queuing delay), rising (increasing queue and queuing delay), congested (queuing delay essentially equal to a full queue), and falling (decreasing queue and queuing delay).

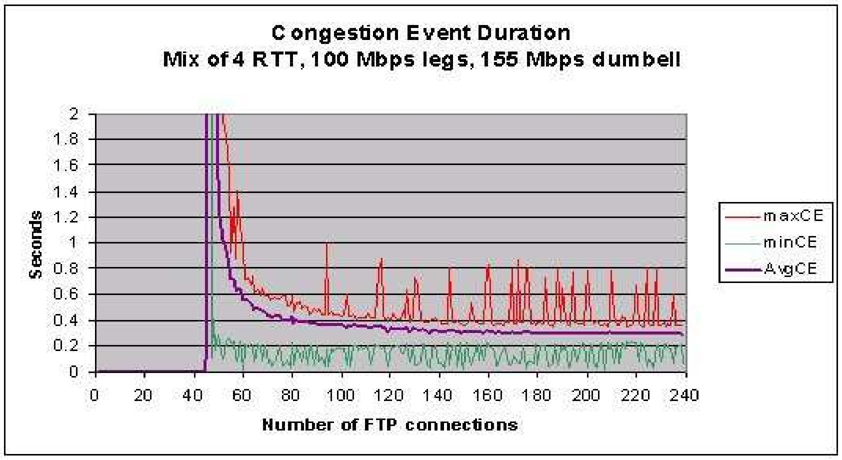


Fig. 10. Congestion Event Duration approaches reaction time.

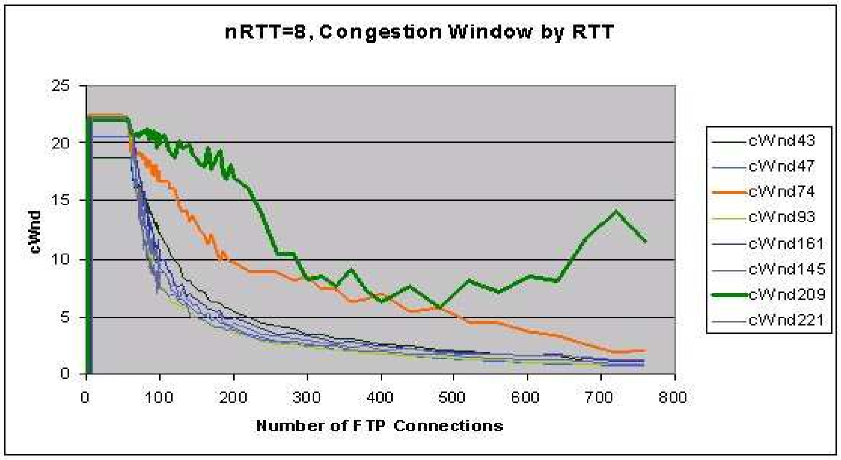


Fig. 11. As flocks at each RTT drop below cWnd 4, they lose much of their share of bandwidth.

As the number of connections builds up, the queue at the bottleneck oscillates between congested and grassy. In this experiment, the dumbbell spent a significant portion of the time (20%) in the grassy area of the shark fin, even though there were 240 connections vying for it's bandwidth.

## I. Duration of a Congestion Event

Figure 10 shows the duration of a congestion event in the dumbbell. As the number of connections increases, the shark fins become increasingly uniform, with the average duration of a congestion event comparatively stable at approximately 280 milliseconds.

## J. Chronic Congestion

Throughout this discussion, TCP kept connections running smoothly in spite of changes in demand that spanned 2 to 200 connections. We searched for the point at which TCP has well-known difficulties when a connection's congestion window drops below 4. At this point, a single dropped packet cannot be discovered by a triple duplicate ACK and the sender will wait for a timeout before re-transmitting.

Figure 11 shows what happened when we increased the number of connections to 760 and spread out the RTT values. When the average congestion window on a particular leg dropped below 4, the other legs were able to quickly absorb the bandwidth released. In this example, the legs with 74 millisecond RTT and 209 millisecond RTT stayed above cWnd 4 and were able to gain a much higher proportion of the total dumbbell bandwidth. When each leg had 90 connections (total 720), the 209 ms leg had an average cWnd of 14.1, compared to 1.9 for it's nearest competitor, RTT 74. Subsequent runs with other values always had the 209 ms leg winning and the 74 ms leg coming in second.

## VI. Congestion Model

The simulation experiments in Section 5 provide the foundation for modeling queue behavior at a backbone router. In this section we present our model and initial validation experiment.

### A. Input Parameters

For a fixed size time tick, $t$, let $C$ be the capacity of the dumbbell link in packets per tick and $Q$ be the maximum depth the link's output queue can hold. The set of fbcks, $F$, has members, $f$, each with a round trip time in ticks, $RTT_f$, a number of connections, $N_f$, a ceiling, $Ceil_f$, and a fbor, $Floor\ _f$. The values of $Ceil_f$ and $Floor_f$ are measured in packets per tick and chosen to represent the bandwidth fbck $f$ will achieve if it is unconstrained at the dumbbell and only reacts to it's worst bottleneck elsewhere.

### B. Operational Parameters

Let $B_t$ be the number of packets buffered in the queue at tick $t$. Let $D_t$ be the number packets dropped in tick $t$, and $L_t$ be the loss ratio. Let $V_{f,t}$ be the volume in packets per tick being offered to the link at time, $t$. Reaction Time, $R_f$, is the average time lag for the fbck to react to feedback. Let $A\ _{f,t}$ be the acceleration rate in packets per tick per tick at which a fbw increases its volume in the absence of any negative feedback. Let $W\ _{f,t}$ be the average congestion window.

2
Initially,

$$V_{f,0} = Floor_f$$

$$W_{f,0} = \frac{V_{f,0} * RTT_f}{N_f}$$

$$R_f = RTT_f * 1.2$$

$$A_{f,0} = ComputeAccel(W_{f,0})$$

$$B_0 = 0$$

For each tick,

$$AvailableToSend_t = B_t + \sum_{f=0}^{F} V_{f,t}$$

$$Sent_t = min(C, AvailableToSend_t)$$

$$Unsent_t = AvailableToSend_t - Sent_t$$

$$B_{t+1} = min(Q, Unsent_t)$$

$$D_t = Unsent_t - B_{t+1}$$

$$L_t = \frac{D_t}{D_t + Sent_t}$$

$$RememberFutureLoss(L_t)$$

For each flock, prepare for the next tick:

$$W_{f,t+1} = ReactToPastLosses(f, L, R_f, W_{f,t})$$

$$A_{f,t} = ComputeAccel(f, W_{f,t})$$

$$V_{f,t+1} = V_{f,t} + A_{f,t}$$

RememberFutureLoss retains old loss rates for future flock adjustments.

ReactToPastLosses looks at the loss rate that occurred at time $t - R_f$ and adjusts the congestion window accordingly. If the loss rate is 0.00, $W_{f,t}$ is increased by $1.0/RTT_f$. If the loss rate is between 0.00 and 0.01, $W_{f,t}$ is unchanged. If the loss rate is higher than 0.01, $W_{f,t}$ is decreased by $W_{f,t}/(2.0 * RTT_f)$. If $Ceiling_f$ has been reached, $W_{f,t}$ is adjusted so $V_{f,t+1}$ will be $Floor_f$. To represent limited receive window, $W_{f,t}$ is limited to $min(46, W_{f,t})$.

In ComputeAccel, if $W_{f,t}$ is below 4.0, acceleration is set to $N_f$ packets per second per second (adjusted to ticks per second). Otherwise ComputeAccel returns $N_f/RTT_f$.

## C. *Outputs of the model*

The model totals the number of ticks spent in each of the queue regimes: Clear, Rising, Congested, or Falling. The queue is in Clear until it rises above 30%, Rising until it reaches 95%, Falling when it drops to 90%, and Clear again at 20%. False rising leads to Clear if, while Rising, the queue drops to 20%. False falling leads to Congested if, while Falling, the queue grows to 95%.

## D. *Calibration*

Figure 12 shows what the model predicts for the simulation in Figure 9. Improvements will be needed in the model to more accurately predict the onset of flocking, but the results for moderate cWnd sizes are appropriate for traffic engineering models. The model correctly approximated the mixture of congested and clear ticks through a broad range of connection loads. Even though the model has simple algorithms for the aggregate reaction to losses, it is able to shed light on the way in which large flocks interact based on their unique RTTs.

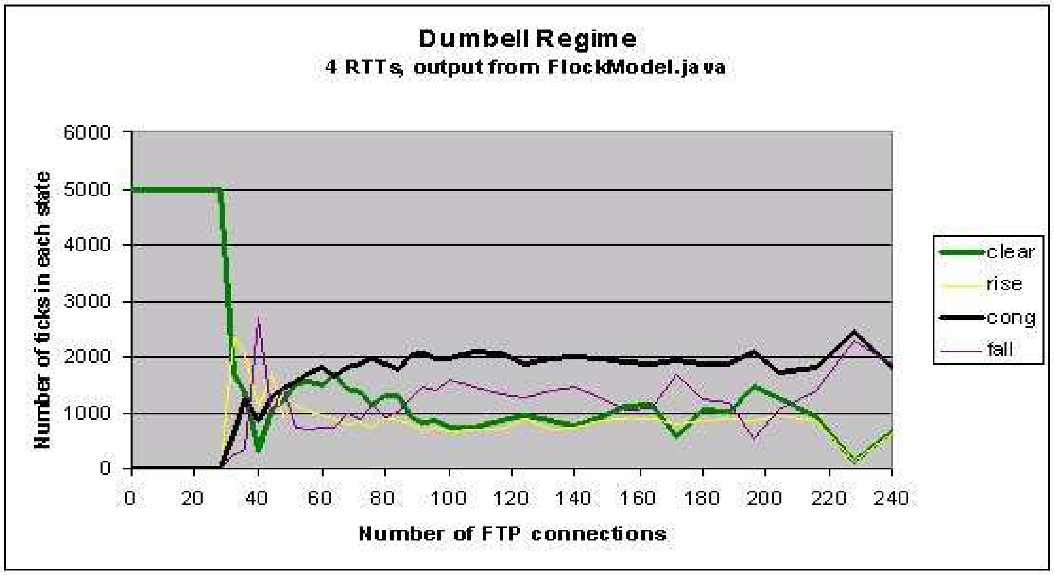


Fig. 12. Queue regimes predicted by the model roughly match the simulation.

## VII. SUMMARY

The study of congestion events is crucial to an understanding of packet loss and delay a multi-hop Internet with fast interior links and high multiplexing. We propose a model based on fbcking as an improved means for explaining periodic traffic variations.

A primary conclusion of this work is that congestion event are either successful or unsuccessful. A successful congestion event discourages enough future traffic to drain the queue to the congested link. Throughout their evolution, transport protocols have sought to make congestion events more likely to be a success. Each new innovation in congestion control widened the portion of the design space in which congestion events are successful. The protocols work well across long RTTs, a broad range of link capacities and at multiplexing factors of thousands of connections. Each innovation narrowed the gaps in the design space between successful congestion events and unsuccessful ones. The result is that a substantial fraction of the congestion events in the Internet last for one reaction time and then quickly abate enough to allow the queue to drain. Depending on the intensity of the traffic and the traffic's ability to remember the prior congestion, the next congestion event will be sooner or later.

The shape of a congestion event tells us 2 crucial parameters of the link being protected: the maximum buffer it can supply and the RTT of the traffic present compared to our own. We hope this study helps ISPs engineer their links to maximize the success of congestion events.

The identification of 4 named regimes surrounding a congestion event may lead to improvements in active queue management that address the impact local congestion events have on neighbors. The result could be a significant improvement in the fairness and productivity of bandwidth achieved by fbcks.

### *A. Future Work*

We plan to extend the model to cover the portion of the design space where congestion events are unsuccessful. By exploring the limits of multiplexing, RTT mixtures, and window sizes with our model we should be able to find the regimes where active queue management or transport protocols can be improvemed. We also need to expand the model so it more accurately predicts the onset of chronic congestion.

Flocking can potentially be exploited as a statistic used in capacity planning. It is clear that link utilization on certain links can be very high even though the link could handle an order of magnitude more connections. Other links would get no benefit from increased bandwidth because the fbcks going through them are constrained elsewhere. We also intend to investigate this possibility.

## REFERENCES

[1] V. Paxson and S. Floyd, "Wide-area traffi c: The failure of poisson modeling," *IEEE/ACM Transactions on Networking*, vol. 3(3), pp. 226–244, June 1995.

[2] S. Floyd and V. Paxson, "Why we don't know how to simulate the Internet," in *Proceedings of the 1997 Winter Simulation Conference*, December 1997.

[3] W. Leland, M. Taqqu, W. Willinger, and D. Wilson, "On the self-similar nature of Ethernet traffic (extended version)," *IEEE/ACM Transactions on Networking*, pp. 2:1–15, 1994.

[4] A. Feldmann, A. Gilbert, W. Willinger, and T. Kurtz, "The changing nature of network traffic: Scaling phenomena," *Computer Communications Review*, vol. 28, no. 2, April 1998.

[5] A. Feldmann, P. Huang, A. Gilbert, and W. Willinger, "Dynamics of IP traffic: A study of the role of variability and the impact of control," in *Proceedings of ACM SIGCOMM '99*, Boston, MA, September 1999.

[6] S. Floyd and V. Jacobson, "Random early detection gateways for congestion avoidance," *IEEE/ACM Transactions on Networking*, vol. 1(4), pp. 397–413, August 1993.

[7] W. Stevens, *TCP/IP Illustrated, Volume 1: The Protocols*, Addison-Wesley, 1994.

[8] J. Bolot, "End-to-end packet delay and loss behavior in the Internet," in *Proceedings of ACM SIGCOMM '93*, San Francisco, Setpember 1993.

[9] V. Paxson, *Measurements and Analysis of End-to-End Internet Dynamics*, Ph.D. thesis, University of California Berkeley, 1997.

[10] M. Yajnik, S. Moon, J. Kurose, and D. Towsley, "Measurement and modeling of temporal dependence in packet loss," in *Proceedings of IEEE INFOCOM '99*, New York, NY, March 1999.

[11] The Surveyor Project, "http://www.advanced.org/surveyor," 1998.

[12] NLANR Acitve Measurement Program - AMP, ," http://moat.nlanr.net/AMP.

[13] W. Matthews and L. Cottrell, "The PINGer Project: Active Internet Performance Monitoring for the HENP Community," *IEEE Communications Magazine*, May 2000.

[14] A. Pasztor and D. Veitch, "A precision infrastructure for active probing," in *PAM2001, Workshop on Passive and Active Networking*, Amsterdam, Holland, April 2001.

[15] M. Allman and V. Paxson, "On estimating end-to-end network path properties," in *Proceedings of ACM SIGCOMM '99*, Boston, MA, September 1999.

[16] R. Govindan and A. Reddy, "An analysis of internet inter-domain topology and route stability," in *Proceedings of IEEE INFOCOM '97*, Kobe, Japan, April 1997.

[17] V. Paxson, "End-to-end routing behavior in the Internet," in *Proceedings of ACM SIGCOMM '96*, Palo Alto, CA, August 1996.

[18] D. Jagerman, B. Melamed, and W. Willinger, *Stochastic Modeling of Traffic Processes*, Frontiers in Queuing: Models, Methods and Problems, CRC Press, 1996.

[19] M. Mathis, J. Semke, J. Mahdavi, and T. Ott, "The macroscopic behavior of the TCP congestion avoidance algorithm," *Computer Communications Review*, vol. 27(3), July 1997.

[20] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose, "Modeling TCP throughput: A simple model and its empirical validation," in *Proceedings of ACM SIGCOMM '98*, Vancouver, Canada, Setpember 1998.

[21] N. Cardwell, S. Savage, and T. Anderson, "Modeling TCP latency," in *Proceedings of IEEE INFOCOM '00*, Tel-Aviv, Israel, March 2000.

[22] S. Floyd and V. Jacobson, "Traffic phase effects in packet-switched gateways," *Journal of Internetworking:Practice and Experience*, vol. 3, no. 3, pp. 115–156, September, 1992.

[23] L. Qiu, Y. Zhang, and S. Keshav, "Understanding the performance of many TCP flows," *Computer Networks (Amsterdam, Netherlands: 1999)*, vol. 37, no. 3–4, pp. 277–306, 2001.

[24] Internet Protocol Performance Metrics, "http://www.ietf.org/html.charters/ippm-charter.html," 1998.

[25] G. Almes, S. Kalidindi, and M. Zekauskas, "A one-way delay metric for ippm," RFC 2679, September 1999.

[26] K. Papagiannaki, S. Moon, C. Fraleigh, P. Thiran, F. Tobagi, and C. Diot, "Analysis of measured single-hop delay from an operational backbone network," in *Proceedings of IEEE INFOCOM '02*, March 2002.

[27] S. Kalidindi, "OWDP implementation, v1.0, http://telesto.advanced.org/ kalidindi," 1998.

[28] Anonymized for blind review, "Unanonymize," .

[29] S. Floyd, "Connections with multiple congested gateways in packet-switched networks part 1: One-way traffic," *ACM Computer Communications Review*, vol. 21, no. 5, pp. 30–47, Oct 1991.