

# Evaluating and Modeling Window Synchronization in Highly Multiplexed Flows

Jim Gast and Paul Barford

University of Wisconsin - Madison, Computer Science, 1210 West Dayton Street  
Madison, WI 53706

{jgast,pb}@cs.wisc.edu

**Abstract**—In this paper, we investigate issues of synchronization in highly aggregated flows such as would be found in the Internet backbone. Understanding this phenomenon is important since it leads to reduced network utilization. Our hypothesis is that regularly spaced loss events lead to window synchronization in long lived flows. We argue that window synchronization is likely to be more common in the Internet than previously reported. We support our argument with evidence of the existence and evaluation of the characteristics of periodic discrete congestion events using active probe data gathered in the Surveyor infrastructure. When connections experience loss events which are periodic, the aggregate offered load to neighboring links rises and falls in cadence with the loss events. Connections whose cWnd values grow from  $W/2$  to  $W$  at approximately the same rate as the loss event period soon synchronize their cWnd additive increases and multiplicative decreases. We find that this window synchronization can scale to large numbers of connections depending on the diversity of roundtrip times of individual flows. Through simulation we investigate conditions under which window synchronization occurs. A model is presented that predicts important characteristics of the loss events in window synchronized flows including the quantity, intensity, and duration. The model effectively explains the prevalence of discrete loss events in fast links with high multiplexing factors as well as the queue buildup and queue draining phases of congestion. We then show how the model scales well for common traffic engineering tasks at the multi-hop level.

## I. INTRODUCTION

Much of the research in network congestion control has been 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 from 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 can arise 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).

Traffic engineering tradition, then, was to size links to accommodate mean load plus a factor for large variability. The problem comes in estimating the large variability. Cao *et al.* [6] provides ways to estimate this variability and suggests that old models do not scale well when the number-of-active-connections ( $NAC$ ) is large. As  $NAC$  increases, packet inter-arrival times will tend toward independence. In particular, that study divides time up into equal-length, consecutive intervals and watches  $p_i$ , the packet counts in interval  $i$ . In that study, the coefficient of variation (standard deviation divided by the mean) of  $p_i$  goes to zero like  $\frac{1}{\sqrt{NAC}}$ . The LRD of the  $P_i$  is unchanging in the sense that the autocorrelation is unchanging, but as  $NAC$  increases, the variability of  $p_i$  becomes much smaller relative to the mean. In practical terms, links utilization of 50% to 60% average measured over a 15 to 60 minute period is considered appropriate [7] for links with average  $NAC$  32. Cao's datasets include a link at OC-12 (622 Mbps) with average  $NAC$  above 8,000. Clearly, traffic engineering models that implicitly assume  $NAC$  values below 32 are inappropriate for fast links.

The model presented in this paper is a purely endogenous view. For simplicity, it only explores oscillations caused by the reactions of sources to packet marking or dropping. Each time a packet is dropped (or marked), the sender of that packet cuts his sending rate (congestion window, cWnd) using multiplicative decrease. Because there is an inherent delay while the feedback is in transit, a congested link may have to give drops (or marks) to many senders. If the congestion was successfully eliminated, connections will enjoy a long loss-free period and will grow their cWnd using additive increase. If the connections grow and shrink their cWnd in synchrony, the global synchronization is referred to as window synchronization [8].

The most significant effort to reduce oscillations caused by synchronization is Random Early Detection (RED) [9]. RED tries to break the deterministic cycle by detecting incipient congestion and dropping (or marking) packets probabilistically. On

slow links, this effectively eliminates global synchronization [10]. But a comprehensive study of window synchronization on fast links has not been made.

Key to understanding window synchronization is an understanding of the congestion events themselves. One objective in this paper is to develop a mechanism for investigating 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.

Data collected with the Surveyor infrastructure [11] 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 during a four month test period with a wide geographic scope.

We now explore the implications if congestion events are regularly spaced. Connections shrink their congestion windows (cWnd) in cadence with the congestion events. The cWnds slowly grow back between events. In effect, the well-known saw-tooth graphs of cWnd [12] for the individual long-lived connections are brought into phase with each other, forming a “flock”: a set of connections whose windows are synchronized.

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 study of synchronization phenomena.

Explicit Congestion Notification (ECN) [13] promises to significantly reduce the delay caused by congestion feedback. In this paper, we will assume that marking a packet is equivalent to dropping that packet. In either case, the sender of that packet will (should) respond by slowing down. Whenever we refer to dropping a packet, marking a packet would be preferable.

We investigate a spectrum of congestion issues related to our model in a series of ns2 [14] simulations. We explore the accuracy of our model over a broad range of offered loads, mixtures of RTT’s, and multiplexing factors. Congestion event statistics from simulation are compared to the output of the model and demonstrate an improved understanding of the duration of congestion events.

The model is easily scaled to paths with multiple congested hops and the interactions between traffic that comes from distinct congestion areas. Extending the model to large networks promises to give better answers to a variety of Traffic Engineering problems in capacity planning, performance analysis and latency tuning.

The rest of this paper is organized as follows. In Section 2, we describe related work. In Section 3, we present the Sur-

veyor data that enabled our evaluation of queue behavior in-situ. Section 4 introduces the notion of an aggregate window for a group of connections and shows how the aggregate reacts to a single congestion event. Section 5 presents ns2 simulations that show how window synchronization can bond many connections into flocks. Each flock then behaves as an aggregate and can be modeled as a single entity. In Section 6, we present our model that accurately predicts the interactions of multiple flocks across a congested link. Outputs include the queue delays, congestion intensities and congestion durations. Sample applications in traffic engineering are enumerated. Finally, Section 7 presents our conclusions and suggests future work.

## II. RELATED WORK

Packet delay and loss behavior in the Internet has been widely studied. Examples include [15] 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 [16]. 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 [17]. Recent work by Zhang *et al.* [18] 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], [19], [20]. 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 [21]. 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 [22], [23], [24]. 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 [25]. 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 [26], [27], [28]. These models use RTT and packet loss rates to predict throughput, and are based on characteristics of TCP’s different operating regimes. Our work uses simpler parameters that are more directly tuned by traffic engineering.

### A. Fluid-Based Analysis

Determining the capacity of a network with multiple congested links is a complex problem. Misra proposed fluid-based analysis [29] employing stochastic differential equations to model flows almost as though they were water pressure in water pipes. Bu used a fixed-point approach [30] that focuses on predicting router average queue lengths. Both methods are fast enough to use in “what if” scenarios for capacity planning or performance analysis. Both methods take, as input parameters, a set of link capacities, the associated buffer capacities, and a set of sessions where each session takes a path that includes an ordered list of links. Our model uses essentially the same input parameters. We expect that the results of these models would be complementary to our results and suggest that traffic engineers use one fluid-based analysis to compare to our window synchronization model. Our expectation is that fluid-based analyses might overstate capacity when our model would understate.

### B. Other Forms of Global Synchronization

The tendency of traffic to synchronize was first reported by Floyd and Jacobson [31]. 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. This paper formed the earliest arguments in favor of RED. This form of global synchronization is the synchronization of losses when a router drops many consecutive packets in a short period of time. Fast retransmit was added to TCP to mitigate the immediate effects. The next form of global synchronization was synchronization of retransmissions when the TCP senders retransmit dropped packets virtually in unison.

In contrast, window synchronization is the alignment of congestion window saw-tooth behavior. Packet level resonance was never shown to extend to more than a few connections. Qui, Zhang and Keshav [18] 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. Window synchronization is the opposite. Window synchronization 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 [32]. 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 a long stream of 40 byte probes at random intervals with a Poisson distribution and a mean interval between

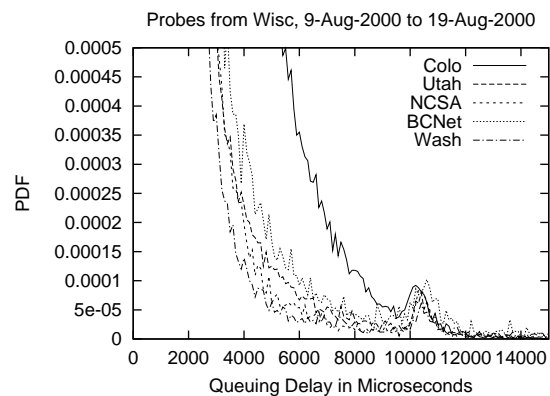


Fig. 1. Probability density of queuing delays of 5 paths

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 [33]. The destination node also has a GPS and records the one-way delay and the time the packet was sent.

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 with a common source. 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.

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).

Figure 2 is included here to put the PDF in context. The cumulative distribution function (CDF) shows that the heads of these distributions differ somewhat. The paths travel through

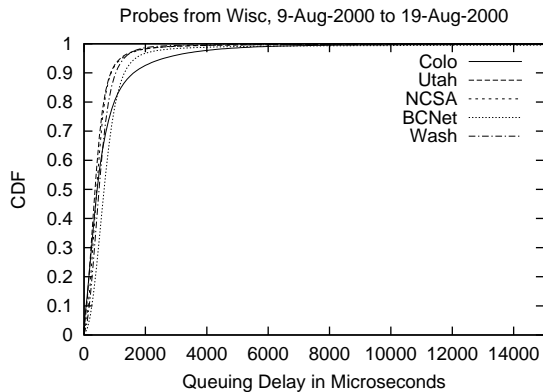


Fig. 2. Cumulative distribution of queuing delays experienced along the 5 paths.

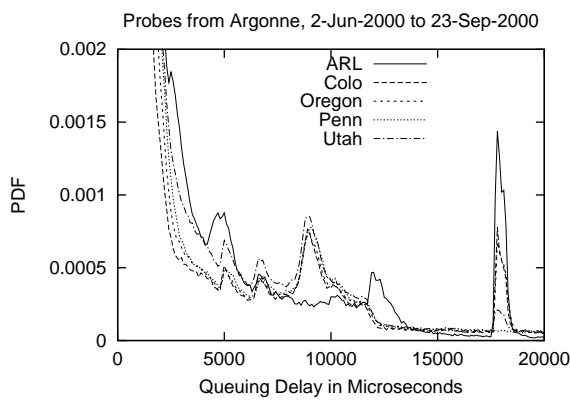


Fig. 3. Probability density of queuing delays on 5 paths that share a long prefix with each other.

different numbers of queues and those routers have different average queue depths and link speeds. But 99% of the queue delay values are below 5 ms. From the CDF alone, we would not have suspected that the PDF showed peaks far out on the tail that were similar width and height.

Figure 3 shows that a distinctive peak in the PDF tail is a phenomenon that is neither unique nor rare. These paths traverse many congested hops, so there is more than one peak in their queuing delay distribution. The path from Argonne to ARL clearly shows that it diverges from the other paths and does pass through the congested link whose signature lies at 9.2 ms. Note that these paths from Argonne do not show any evidence of the peak shown in figure 1, presumably because they do not share the congested hop that has that characteristic signature.

### 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. ISPs could be introducing intentional delays for rate limiting or traffic shaping. There could be delays involved when link cards are busy with some other task (e.g. routing

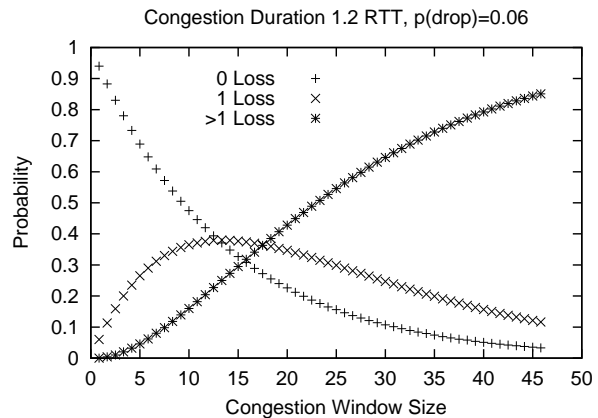


Fig. 4. Showing the probability of losing 0, exactly 1, or more than one packet in a single congestion event as a function of cWnd.

table updates, called the coffee break effect [34]). Our data does not rule out the possibility that we might be measuring some phenomenon other than queuing delay, but our intuition is that those phenomenon would manifest themselves as slopes or plateaus in the delay distribution rather than peaks.

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 [35].

## IV. WINDOW SIZE MODEL

We construct a cWnd feedback model that predicts the reaction of a group of connections to a congestion event. This model simplifies an aggregate of many connections into a single flock and predicts the reaction of the aggregate when it passes through congestion.

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 4 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  $p(\text{NoLoss})$ , 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”.

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

With probability  $p(\text{Many})$ , a connection will see more than one loss. In this example, a connection with  $cW_{nd}$  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  $W/2$ .

If an aggregate of many connections could be characterized with a single  $cW_{nd}$ ,  $W$ , a reaction time,  $R$ , and a single  $RTT$ , the aggregate would emerge from the congestion event with  $cW_{nd}$   $W'$ .

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

This change in  $cW_{nd}$  predicts the new value after the senders learn that congestion has occurred. In section 6, we will incorporate a simple heuristic to include a factor that represents the quiet period if the losses were heavy enough to cause coarse timeouts.

## V. SIMULATIONS

We use a series of *ns2* simulations to understand congestion behavior details and have made them available for public download at [36].

### 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

Figure 5 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 6 shows a cleared queue with small queuing delays. Because the graph here looks like grass compared to the delays associated with congestion,

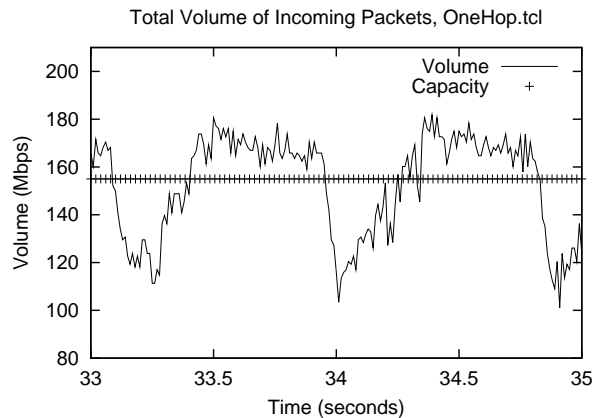


Fig. 5. Mbps of traffic arriving at the ingress router in the one hop simulation.

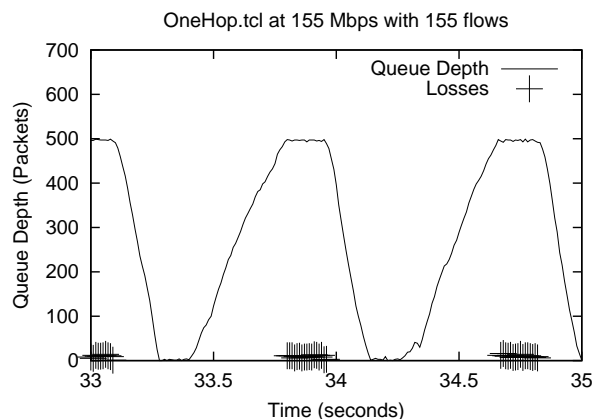


Fig. 6. Probability density of queue depths in packets in the one hop simulation.

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 6. 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 6 has a duration heavily influenced by the average reaction time of the flows. 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

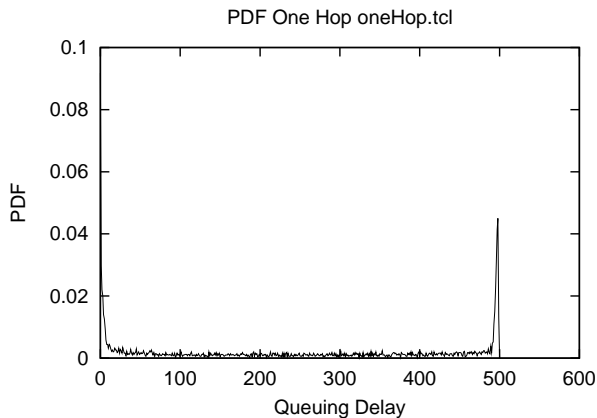


Fig. 7. Mbps of traffic destined for a congested link.

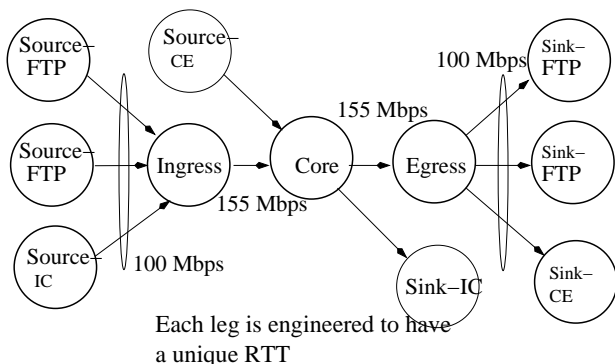


Fig. 8. Simulation layout for two-hop traffic

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 7 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 equi-probable 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

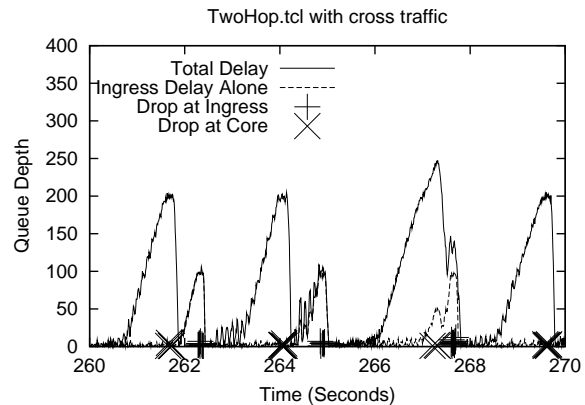


Fig. 9. Both signatures appear when queues of size 100 and 200 are used in a 2-hop path.

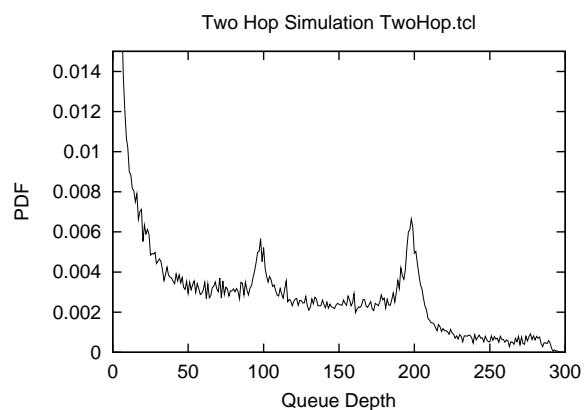


Fig. 10. The distinctive signature of each queue shows up as a peak in the PDF.

an additional core router between the ingress and the egress as shown in figure 8. 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.

Figure 9 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.

Figure 10 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 its RTT) will dictate the amount of load it offers at each link along its 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.

Contrast this with another form of global synchronization reported by Keshav, et al. [18] in which all connections passing through a common congestion point regardless of RTT synchronize. The Keshav study depends on the buffer (plus any packets resident in the link itself) being large enough to hold 3 packets per connection. In that form, increasing the number of connections would eliminate the synchronization. Window synchronization theory does not depend on large buffers or slow links, but rather it depends on a mixture of RTT's that are close enough to be compatible.

### F. Flock Formation

To demonstrate a common situation in which cWnd sawtooth graphs fall into phase with each other, we construct the dumbbell environment shown in Figure 11. Each of the legs feeding

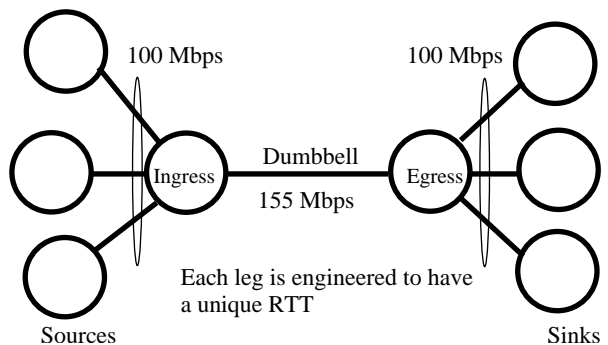


Fig. 11. Simulation environment to foster window synchronization.

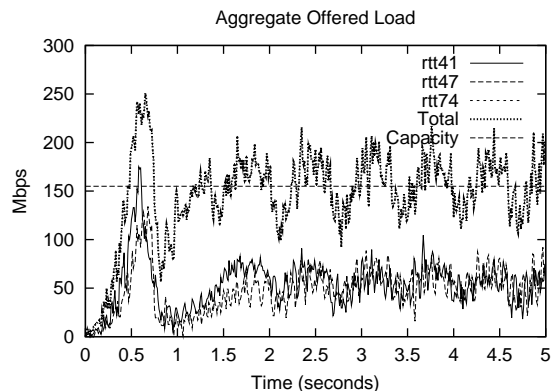


Fig. 12. Connections started at random times synchronize cWnd decline and buildup after 2 seconds.

the dumbbell runs at 100 Mbps, while the dumbbell itself is a 155 Mbps link.

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

Figure 12 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 at time  $t_2.1$ . 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 11 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.

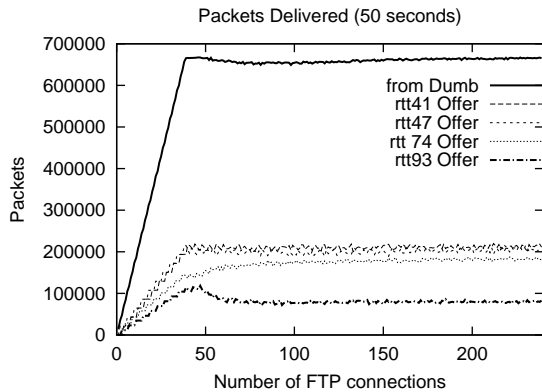


Fig. 13. Connections with RTT slightly too long to join flock.

Figure 13 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.

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. [37], 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 13, 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 next examine the proportion of time spent in each portion of the shark fin using the same simulation configuration as in the prior section.

Figure 14 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).

When there are fewer than 40 connections, the aggregate offered load reaching the dumbbell is less than 155 Mbps, and no packets need to be queued. There is a fascinating anomaly from 40 to 45 connections that happens in the simulations, but is likely to be transient in the wild. In this situation, the offered load coming to the dumbbell is reduced (one reaction time later) by an amount that exactly matches the acceleration of the TCP window growth during the reaction time. This results in a state

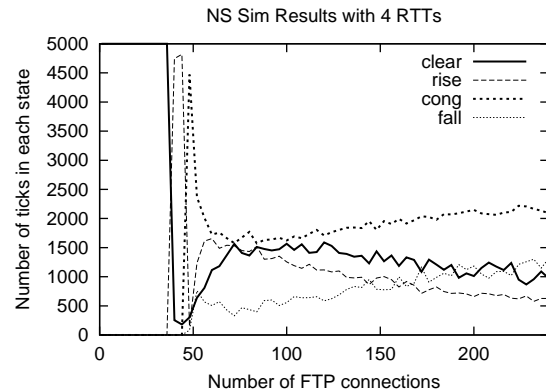


Fig. 14. Proportion of time spent in each queue regime.

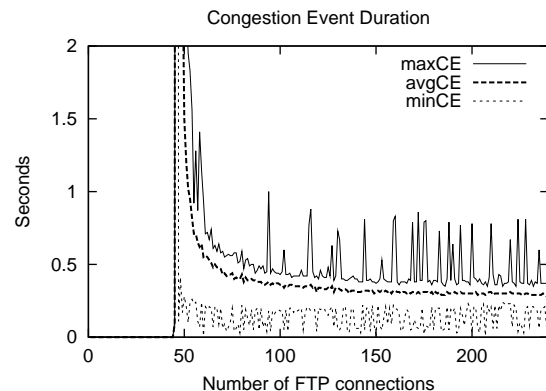


Fig. 15. Congestion Event Duration approaches reaction time.

of continuous congestion with a low loss rate. We believe this anomaly is the result of the rigidly controlled experimental layout. Further study would be appropriate.

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 its bandwidth.

#### I. Duration of a Congestion Event

Figure 15 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 16 shows what happened when we increased the number of connections to 760 and spread out the RTT values. When

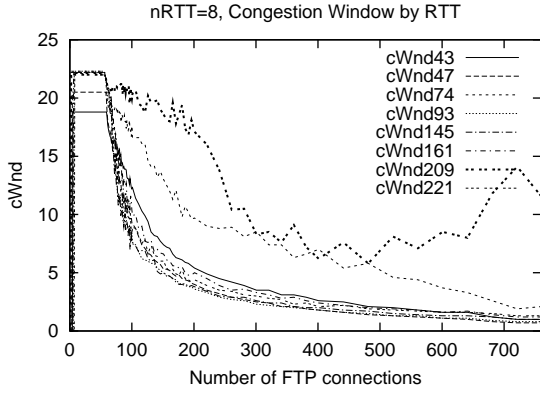


Fig. 16. As flocks at each RTT drop below  $cW_{nd} 4$ , they lose much of their share of bandwidth.

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  $cW_{nd} 4$  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  $cW_{nd}$  of 14.1, compared to 1.9 for its nearest competitor, RTT 74. Subsequent runs with other values always had the 209 ms leg winning and the 74 ms leg coming in second.

In this case, TCP connections with 209 ms RTT actually fared better than many connections with shorter RTT. This is directly contradicts the old adage, “TCP hates long RTT”. We speculate that the sawtooth graph for  $cW_{nd}$  for those connections is long (slow) enough so that the loss risk is low for two consecutive congestion events. Perhaps the new adage should be “TCP hates  $cW_{nd}$  below 4”.

### K. Short-Lived Flows and Non-Responsive Flows

Next we considered simulations that added a variety of short-lived connections. It is common for the majority of connections seen at an Internet link to be short-lived, while the majority of packets are in long-lived flows. For our purposes, we consider a connection short-lived if its lifetime is shorter than the period between congestion events. The short-lived connections have no memory of any prior congestion event. They neither add to nor subtract from the long-range variance in traffic. As the number of active short-lived connections increases, more bandwidth is added to the mean traffic. At high bandwidth (and therefore a high number of active connections), both short-lived flows and non-responsive flows (typically a sub-class of UDP flows that do not slow down in response to drops) simply add to mean traffic.

## 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 flocks,  $F$ , 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 floor,  $Floor_f$ . The values of  $Ceil_f$  and  $Floor_f$  are measured in packets per tick and chosen to represent the bandwidth flock  $f$  will achieve if it is unconstrained at the dumbbell and only reacts to its 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 flock to react to feedback. Let  $A_{f,t}$  be the acceleration rate in packets per tick per tick at which a flow increases its volume in the absence of any negative feedback. Let  $W_{f,t}$  be the average congestion window.

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

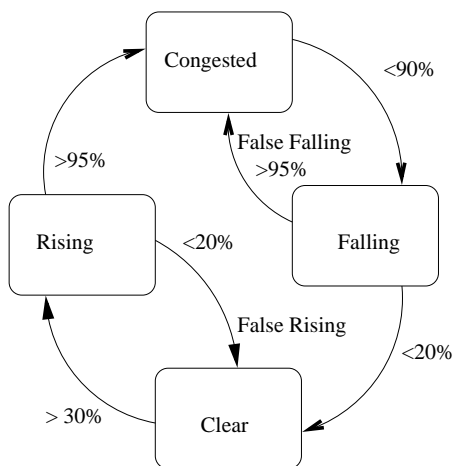


Fig. 17. Finite State Machine for tracking the duration of congestion based on queue occupancy.

loss rate is 0.00,  $W_{f,t}$  is increased by  $1.0/RTT_f$ , representing normal additive increase window growth. If the loss rate is between 0.00 and 0.01,  $W_{f,t}$  is unchanged, modeling an equilibrium state where window growth in some connections is offset by window shrinkage in others. The factor 0.01 is somewhat arbitrarily chosen. Future work should either justify the constant or replace it with a better formula. 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  $\min(46, W_{f,t})$ . The constant here is 46 because 46 packets of 1500 bytes each fill a 64K Byte receive window. Early implementations of LINUX actually used 32K Byte receive windows, but memory became cheap. Without using window scaling, receive windows are limited to 64K Bytes.

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  $N_f/RTT_f$ . Notice that the computation of the acceleration and  $W_{f,t+1}$  given differs from the formula for  $W'$  given in section 3. The compromise was deployed when the model failed to accurately predict the quiet time after a congestion event. The quiet time is primarily caused by connections that suffer a coarse timeout.

The Reaction Time,  $R_f$ , should actually depend on one  $RTT_f$  plus a triple, duplicate ACK. We use the simplification here of taking  $1.2 * RTT_f$  because we did not want to model the complexities of ACK compression and the corresponding effect on clocking out new packets from the sender.

### 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 Finite State Machine is shown in Figure 17. 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%.

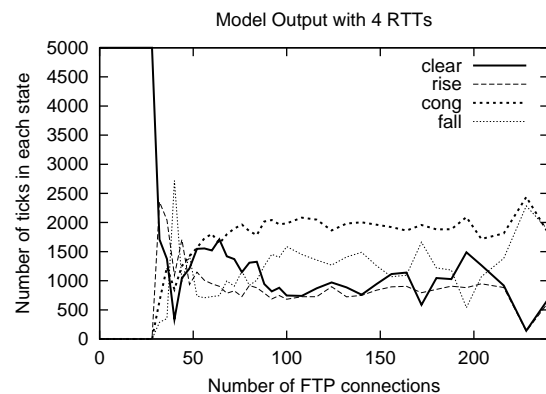


Fig. 18. Queue regimes predicted by the congestion model

### D. Calibration

Figure 18 shows what the model predicts for the simulation in Figure 14. 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.

### E. Extending the Model to Multi-Hop Networks

The ultimate value of the model is its ability to scale to traffic engineering tasks that would typically be found in an Internet Service Provider. Extending the model to a network involves associating with each flock,  $f$ , a sequence of  $h$  hops,  $hop_{h,f} \in Links$ . Each link,  $link \in Links$ , has a capacity,  $C_{link}$ , and a buffer queue length,  $Q_{link}$ .

1) *Headroom Analysis*: The model predicts the number, duration, and intensity of loss events at each link,  $link$ . It is easy to iteratively reduce the modeled capacity of a link,  $C_{link}$ , until the number of loss events increases. The ratio of the actual capacity to the needed capacity indicates the link's ability to accommodate more traffic.

2) *Capacity Planning*: By increasing the modeled capacity of individual links or by adding links (and adjusting the appropriate hop sequences,  $hop_{h,f}$ ), traffic engineers can measure the improvement expectation. Similarly, by adding flocks to model anticipated growth in demand, traffic engineers can monitor the need for increased capacity on a per-link basis. It is important to note that some links with relatively high utilization can actually have very little stress in the sense that increasing their capacity would have minimal impact on network capacity.

3) *Latency Control*: Because the model realistically reflects the impact of finite queue depths at each hop, it can be used in sensitivity analyses. A link with a physical queue of  $Q_{link}$  can be configured using RED to act exactly like a smaller queue. The model can be used to predict the benefits (shorter latency and lower jitter) of smaller queues at strategic points in the network.

4) *Backup Sizing*: After gathering statistics on a normal baseline of flocks, the model can be run in a variety of failure simulations with backup routes. To test a particular backup route, all flocks passing through the failed route need to be assigned a new set of hops,  $hop_{h,f}$ . Automatic rerouting is beyond the scope of the current model, but would be possible to add if multiple outages needed to be modeled.

## 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 flocking as an improved means for explaining periodic traffic variations.

A primary conclusion of this work is that congestion events 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 flocks.

### A. Applicability to Traffic Engineering

When the model is applied to multi-hop networks, it can be used for capacity planning, backup sizing and headroom analysis. We expect that networks in which every link is configured for 50% to 60% utilization may be grossly over-engineered when treated as a multi-hop network. It is clear that utilization on certain links can be high even though the link is not a significant bottleneck for any flows. Such links would get no appreciable benefit from increased bandwidth because the flocks going through them are constrained elsewhere.

### B. 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 improved. We also need to expand the model so it more accurately predicts the onset of chronic congestion.

The traffic engineering applications for the model are particularly interesting. Improvements are needed to automate the gathering of baseline statistics (as input to the model) and to script commonly used traffic engineering tasks so the outputs of the model could be displayed in near real time.

Further validation of the model's scalability and accuracy would be important and interesting.

## REFERENCES

- [1] V. Paxson and S. Floyd, "Wide-area traffic: 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] J. Cao, W. Cleveland, D. Lin, and D. Sun, "Internet traffic: Statistical multiplexing gains," *DIMACS Workshop on Internet and WWW Measurement, Mapping and Modeling*, 2002, 2002.
- [7] J. Cao, W. Cleveland, D. Lin, and D. Sun, "The effect of statistical multiplexing on the long range dependence of Internet packet traffic," *Bell Labs Tech Report*, 2002.
- [8] L. Zhang, S. Shenkar, and D. Clark, "Observations on the dynamics of a congestion control algorithm: The effects of two-way traffic," in *Proceedings of ACM SIGCOMM*, 1991.
- [9] 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.
- [10] M. May, T. Bonald, and J-C. Bolot, "Analytic evaluation of RED performance," in *Proceedings of IEEE INFOCOM 2000*, Tel Aviv, Israel, March 2000.
- [11] The Surveyor Project, "http://www.advanced.org/surveyor," 1998.
- [12] W. Stevens, *TCP/IP Illustrated, Volume 1: The Protocols*, Addison-Wesley, 1994.
- [13] K. Ramakrishnan and S. Floyd, "A proposal to add explicit congestion notification ECN to IP," IETF RFC 2481, January 1999.
- [14] UCB/LBNL/VINT Network Simulator - ns (version 2), "http://www.isi.edu/nsnam/ns/," 2000.
- [15] J. Bolot, "End-to-end packet delay and loss behavior in the Internet," in *Proceedings of ACM SIGCOMM '93*, San Francisco, September 1993.
- [16] V. Paxson, *Measurements and Analysis of End-to-End Internet Dynamics*, Ph.D. thesis, University of California Berkeley, 1997.
- [17] M. Jain, 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.
- [18] 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.
- [19] NLNR Active Measurement Program - AMP, "http://moat.nlanr.net/AMP,"
- [20] W. Matthews and L. Cottrell, "The PINGer Project: Active Internet Performance Monitoring for the HENP Community," *IEEE Communications Magazine*, May 2000.
- [21] A. Pasztor and D. Veitch, "A precision infrastructure for active probing," in *PAM2001, Workshop on Passive and Active Networking*, Amsterdam, Holland, April 2001.
- [22] M. Allman and V. Paxson, "On estimating end-to-end network path properties," in *Proceedings of ACM SIGCOMM '99*, Boston, MA, September 1999.
- [23] 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.

- [24] V. Paxson, "End-to-end routing behavior in the Internet," in *Proceedings of ACM SIGCOMM '96*, Palo Alto, CA, August 1996.
- [25] D. Jagerman, B. Melamed, and W. Willinger, *Stochastic Modeling of Traffic Processes*, Frontiers in Queuing: Models, Methods and Problems, CRC Press, 1996.
- [26] 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.
- [27] 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, September 1998.
- [28] N. Cardwell, S. Savage, and T. Anderson, "Modeling TCP latency," in *Proceedings of IEEE INFOCOM '00*, Tel-Aviv, Israel, March 2000.
- [29] V. Misra, W. Gong, and D. Towsley, "Fluid-based analysis of a network of AQM routers supporting TCP flows with an application to RED," in *SIGCOMM*, 2000, pp. 151–160.
- [30] T. Bu and D. Towsley, "Fixed point approximations for TCP behavior in an AQM network," in *Proceedings of ACM SIGMETRICS '01*, 2001.
- [31] 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.
- [32] Internet Protocol Performance Metrics, "<http://www.ietf.org/html.charters/ippm-charter.html>," 1998.
- [33] G. Almes, S. Kalidindi, and M. Zekauskas, "A one-way delay metric for ippm," RFC 2679, September 1999.
- [34] 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.
- [35] S. Kalidindi, "OWDP implementation, v1.0, <http://telesto.advanced.org/kalidindi>," 1998.
- [36] Wisconsin Advanced Internet Lab, "<http://wail.cs.wisc.edu>," 2002.
- [37] 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.