Optimizing File Transfer Response Time Using the Loss-Load Curve Congestion Control Mechanism

Carey L. Williamson
Department of Computational Science
University of Saskatchewan

January 8, 1993

Abstract

Loss-load curves are a recently proposed feedback mechanism for rate-based congestion control in datagram computer networks. In the loss-load model, packet loss inside the network is a direct function of sender transmission rates, and senders have the responsibility of choosing their own transmission rate based on the loss-load tradeoff curve provided by the network. Earlier work [17] has provided the mathematical basis for the loss-load model and provided preliminary simulation results demonstrating its responsiveness, fairness, and stability. The loss-load model works well for simple network environments where each source has a large number of packets to transmit, and wishes to maximize raw throughput.

In this paper, we show how to extend the applicability of loss-load curves to senders with more general network traffic requirements, such as file transfer activities. In particular, we consider senders wishing to minimize the response time for a transfer of N packets. When N is small, a sender chooses a low transmission rate, to minimize the probability of packet loss. When N is large, a higher transmission rate is chosen to increase raw throughput. The loss-load curve mechanism provides necessary and sufficient information for senders to choose the proper transmission rate based on their traffic requirements and the current network load.

Simulation results show that the loss-load mechanism is effective in achieving high network utilization and minimizing response time for file transfers. The loss-load model also does a good job of providing equitable service to senders with different traffic workloads and different round trip times. Furthermore, in an aggregate network of users, the overall level of packet loss is actually lower than the bound predicted by the basic loss-load model.

1 Introduction

Optimizing the response time for file transfer activities in packet-switched networks is a difficult problem. The bursty nature of data traffic makes it hard to predict resource requirements within the network, and the presence of unregulated bursty traffic often makes the network subject to congestion.

The end-to-end performance seen by users of a packet-switched network is thus strongly dependent on the congestion control strategy used within the network. The goal of a congestion control algorithm is to regulate traffic flow in the network to prevent, reduce, or control the level of congestion. Congestion control algorithms must strive to simultaneously achieve several desirable features: responsiveness, fairness, robustness, stability, and low overhead [8]. Many algorithms for congestion control are proposed in the literature [4, 6, 7, 8, 10, 13].

There are two main approaches to congestion control: *reactive* approaches and *preventive* approaches [5]. Reactive approaches are *feedback-based*: they rely on the network to provide feedback signals, either explicit or implicit, indicating the presence or absence of congestion. Sources then dynamically adjust their rate of transmission into the network. Preventive approaches are *reservation-based*: they rely on *admission control* (a new call is accepted into the network only if its specified resource requirements can be accommodated without jeapordizing other calls) and *bandwidth enforcement* (ensuring that a call remains within its reserved resources). Preventive approaches seek to eliminate congestion at the network level, by making (possibly) conservative use of network resources. Feedback-based approaches usually allow higher network utilization, but at the risk of some level of congestion and packet loss within the network.

The problem with most feedback-based approaches is that they don't provide enough feedback information. Some approaches provide only one bit of feedback information [13], indicating either the presence or absence of congestion along a network path. There is no means to indicate where the congestion is, how heavy the congestion is, or who is causing it. Senders must adjust their transmission rates based on incomplete information about changes in network load. Oscillations and low network utilization often result, particularly as the feedback delay increases. For this reason, preventive approaches are often favoured for high-speed networks.

While preventive approaches will most likely be required as part of the congestion control framework for high-speed networks [5], it is still necessary to identify what the fundamental performance limits of feedback-based approaches really are (i.e., to understand how well feedback-based approaches could perform if more complete feedback information was provided). One possible means for providing this feedback information is with the *loss-load model* [17].

The loss-load model is a recently proposed algorithm for feedback-based congestion control in datagram computer networks. In the loss-load model, each gateway (or switch) in the network monitors local traffic load and provides a *loss-load curve* as feedback to each sender. The loss-load curve expresses, for each sender, the expected probability of packet loss for that sender, as a function of that sender's offered load. Senders have the responsibility of choosing their own transmission rate based on the tradeoff between throughput and packet loss provided by the loss-load curve.

The advantage of the loss-load curve over other feedback mechanisms [7, 13] is that it provides more than one bit of feedback. Senders have enough information to choose a proper initial rate for transmission into the network. The loss-load curve also provides enough information for senders to respond to changes in network load, both increases and decreases.

Earlier work [17] has established the mathematical foundation for loss-load curves and demonstrated their properties of fairness, robustness, and stability. The loss-load model works well for simple network environments where each source has a large number of packets to transmit, and wishes to maximize raw throughput. What remains to be seen is how well the model works for more general network scenarios: dynamically changing network traffic, senders with differing amounts of traffic to send, and senders with different round trip times. The purpose of this paper, then, is to study how senders can use the loss-load model to optimize their data transfer response time in the presence of dynamic network traffic.

The remainder of this paper is organized as follows. Section 2 provides more background material on loss-load curves, summarizing their formulation and some of their important properties. Section 3 describes how to make use of the loss-load curve at the sender to optimize response time for file transfer activity, rather than just raw throughput. A simple model for file transfer activity is proposed, and validated through simulation. Section 4 presents a more detailed simulation to study the response time model in a network with dynamic traffic flows. Results show that the loss-load model can provide a 50% improvement in response time compared to a traditional feedback-based approach [7]. Finally, Section 5 presents conclusions and directions for further research.

2 The Loss-Load Model

The main premise of the loss-load model is that it is not unreasonable to operate a packet-switched network at non-negligible levels of packet loss (e.g., 1-10%). Some network applications, such as voice or video, are insensitive to such levels of packet loss, depending on the coding strategy used [12, 14]. Some applications, such as file transfer, can recover from packet loss by retransmitting lost packets. Some applications may even be explicitly designed to tolerate such levels of packet loss [1, 15]. For these reasons, then, the loss-load model allows the network to operate at levels of very high utilization, or even overload, with non-negligible levels of packet loss. In fact, the aggressive nature of the loss-load model makes it stable only at levels of slight overload.

In such an overloaded network, the level of packet loss assigned to each sender by the loss-load model is a direct function of that sender's transmission rate. Gateways in the network dynamically compute the packet loss function for each sender from their current local traffic load, and provide the loss function (in the form of a loss-load curve) as explicit

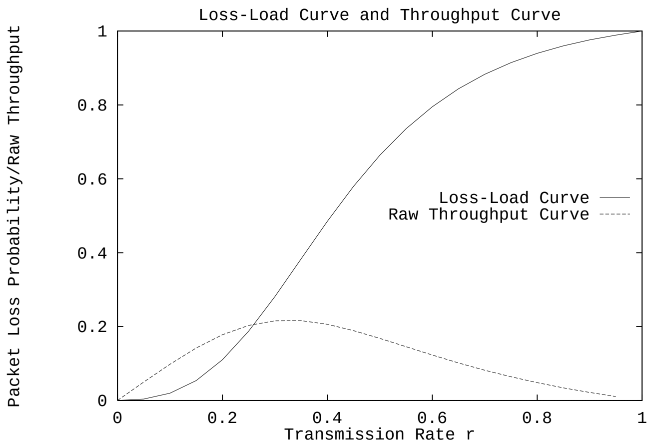

Figure 1: Sample Loss-Load Curve and Raw Throughput Curve

feedback to the senders. Senders use the curve to adjust their transmission rates.

The loss-load curve is dynamic; it changes with time as senders dynamically adjust their rates. Nevertheless, loss-load curves are inherently stable, since they depend primarily on the aggregate network traffic, and not as much on what any particular sender is doing. Each sender can have only a bounded effect on the loss-load curve of other senders [16]. Furthermore, as the number of senders increases, the contribution of any single sender diminishes, increasing stability.

2.1 The Loss-Load Curve

Conceptually, the loss-load curve appears as shown in Figure 1. The curve expresses the tradeoff between the level of packet loss for a sender at a particular gateway as a function of that sender's offered load. The loss-load curve is a monotonically increasing function of sender rate[1]. The packet loss probability approaches 100% for high transmission rates in order to protect the network from misbehaving senders [17].

Physically, the loss-load curve can be represented using only two 32-bit floating point numbers, a and b, defined formally in Section 2.2. The parameter a represents the aggregate load at the gateway, for all sender traffic streams using the gateway. The parameter b measures the distribution of the aggregate load amongst the senders. A sender can compute the entire loss-load curve for a particular gateway given only that gateway's current values for a and b. This compact representation makes it easy for gateways to exchange loss-load curves with senders across the network.

[1]For convenience, all transmission rates r are normalized to the capacity C of the gateway, to give $0 \leq r \leq 1$. In this particular figure, $C = 1$. Throughout the paper rates and capacities are treated as dimensionless quantities. Rates and capacities could be expressed in bits/sec or packets/sec, if desired.

Associated with each loss-load curve is a *raw throughput* curve that expresses the rate at which packets are successfully forwarded by the gateway, for a given sender, given the current load conditions. The raw throughput τ is computed directly from the loss-load curve, using $\tau = r * (1 - p)$ where r is the rate of transmission, and p is the probability of packet loss[2].

A sample raw throughput curve appears in Figure 1. The throughput curve increases at first, attains some maximum throughput τ^*, and then decreases. The maximum raw throughput is achieved at a rate r^*_τ. Given the loss-load curve, a sender can directly compute the raw throughput curve, and can compute r^*_τ.

2.2 The Loss-Load Formula

Mathematically, the loss-load curve for a single gateway is defined as follows [17]. Assume that the gateway is forwarding traffic from one or more input links onto a single output link of capacity C. Let M be the number of senders sharing the gateway. Let r_i represent the packet arrival rate[3] for the ith sender, and $r_{total} = \sum_{i=1}^{M} r_i$ represent the aggregate traffic load arriving at the gateway. Then the probability of packet loss for the $(M+1)$st sender when transmitting at rate r is given by

$$P(r) = \frac{r^{k+1} + ar^k}{r^{k+1} + b}$$

where $a = r_{total} - C$ and $b = \sum_{i=1}^{M} r_i^{k+1}$. The parameter a measures the total (excess) traffic load at the gateway, and the parameter b measures the distribution of the total load amongst the competing senders. The parameter $k > 0$ is an integer parameter that can be set as a global constant for the network. The importance of k in determining the behaviour of the loss-load algorithm is summarized in Section 2.3.

With the loss-load formula defined above, the gateway can dynamically determine a packet loss probability p_i for each sender such that the gateway's excess traffic is discarded (i.e., $\sum_{i=1}^{M} p_i r_i = r_{total} - C$). The assignment of p_i's has the following properties [16]:

- If the total arriving load is less than the capacity of the gateway, then no packets are discarded. That is, $p_i = 0$ for all i.

- If the total load is greater than the capacity, then *all* senders share in packet loss (but not necessarily equally). That is, $p_i > 0$ for all i.

- The p_i's are a non-decreasing function of the sender rates r_i.

[2] An analogy may help explain the notion of raw throughput. Consider what happens when you pour water (at rate r) into a funnel (of capacity C). If you pour the water too quickly, some fraction (p) of the water spills out the top of the funnel, and some of the water gets through successfully. The water that gets through successfully is the raw throughput (τ). In the loss-load model, τ is a function of both r and p.

[3] Assume that the rates for each sender are established at connection setup time, and renegotiated while the connection is active. Alternatively, the rate information could be explicitly carried in each packet, or determined by the gateway using rate estimation.

Once the p_i's are determined, then each packet from sender i is discarded, at random, with probability p_i, to statistically enforce the desired rate of packet loss[4]. In normal operation, packets are discarded well before the buffers at the gateway fill up. By setting the capacity parameter C, a reasonable average buffer occupancy can be achieved.

2.3 Properties of the Loss-Load Formula

Several theoretic results have been established for loss-load curves when all senders are maximizing raw throughput. These results are summarized here. Complete details are reported elsewhere [16].

Theorem 1 Bounded Packet Loss *The packet loss probability when transmitting at the rate* r_τ^* *to maximize raw throughput is always* $\frac{1}{k+1}$*, where* $k > 0$ *is a settable integer parameter in the loss-load formula.*

That is, the level of packet loss at r_τ^* is always bounded. The bound is always the same, regardless of the number of senders, and regardless of the network load. The larger k, the lower the level of packet loss at r_τ^*.

Theorem 2 Convergence *If* M *identical senders sharing a single bottleneck gateway of capacity* C *are all maximizing raw throughput, then the network converges to a stable equilibrium operating point with each sender at their own* r_τ^**. The total offered load at equilibrium is* $\frac{k+1}{k}C$*.*

In other words, the loss-load model always converges to a stable equilibrium. The equilibrium achieved is independent of the number of senders, and independent of the round trip delay to each sender. The convergence process is iterative, involving action and reaction between senders. Convergence is fast, typically requiring two to four iterations. The speed of convergence depends on k. The larger k, the slower the convergence. Note that convergence is to a stable *overload* of $\frac{k+1}{k}C$. The larger k, the smaller the overload at equilibrium.

Theorem 3 Multi-Hop Paths *A sender traversing a multi-hop path in the network may experience packet loss at more than one congested gateway along the path. However, as long as the sender's transmission rate is chosen based on the loss-load curve for the* bottleneck gateway*, the level of end-to-end packet loss for that sender asymptotically approaches* $\frac{1}{k+1}$ *as* k *increases.*

[4] "Packet loss" in the loss-load model need not necessarily be implemented by discarding packets, as implied here. Instead, "lost" packets could be "tagged" (as in the virtual leaky bucket control scheme [9]) and forwarded through the network, as long as the network has a different priority or pricing structure for delivering tagged packets that provides a deterrent to misbehaving senders. The same control principles would still apply, and a bounded level of tagged packets would result. The presentation in this paper assumes only the simplest form of the loss-load model, where packets are simply discarded according to the loss-load curve.

In other words, only the bottleneck gateway matters when senders are transmitting over a multi-hop path. The bottleneck gateway is the gateway whose loss-load curve has the lowest value for r^*_τ. As long as the transmission rate is chosen based on the bottleneck gateway, the end-to-end level of packet loss is approximately $\frac{1}{k+1}$. Packet loss at gateways other than the bottleneck gateway is negligible for $k \geq 5$.

2.4 Summary

The parameter k is the most important control parameter in the loss-load formula. The choice of k presents a tradeoff between the level of packet loss in the network and the responsiveness of the loss-load algorithm. A recommended value for k is $k = 10$ [16].

The theoretic results highlighted above show that the loss-load model may work well for bulk transfer activities: senders with a large number of packets to send, interested in maximizing raw throughput. What remains to be seen is how well the model works for more general network traffic scenarios: senders with differing amounts of traffic to send, and senders with different round trip times. The remainder of this paper addresses exactly this issue.

3 Optimizing File Transfer Response Time

Consider a sender with N packets that need to be reliably transferred to a destination. The sender is interested in minimizing the elapsed time to complete the transfer. Should the sender transmit at a high rate, to maximize throughput, or at a low rate, to minimize the probability of losing packets?

The loss-load curve provides enough information for senders to answer this question. This section presents an abstract model for the response time optimization problem, and shows how to use loss-load curves to choose a transmission rate. This model is suitable for common data transfer applications such as file transfer, net news, or electronic mail.

3.1 Protocol Model

The analysis assumes a simple transport protocol that supports positive/negative acknowledgement and selective retransmission. Retransmission of lost packets is done using conventional retransmission mechanisms. No forward error correction techniques are used.

The protocol also supports rate-based flow control. A sender injects packets into the network at a rate r, $0 \leq r \leq 1$. For simplicity, assume that one packet is sent (if available) every $1/r$ time units. Thus the transmission of N packets takes time N/r. The value of r is chosen by the transport layer based on information obtained from the network and application layers. The rate r remains constant for the duration of the transfer.

3.2 Response Time Model

The elapsed time to complete a file transfer depends only on the time at which the last packet is successfully delivered to the destination. It does not matter what order the packets arrive in, or how many packets are lost and retransmitted during the transfer. For file transfer activities, then, response time can be modelled as follows.

A reliable transfer of N packets requires several *rounds*. In the first round, the N packets are transmitted. One round trip time after the transmission of the Nth packet, an acknowledgement is received[5] indicating the delivery status (received or not received) of each of the N packets. If all N packets have been correctly received the transfer is complete. Otherwise, the N' packets requiring retransmission are sent in the next round. If the loss probability is p, then the expected value for N' is Np. In i rounds, the expected number of packets transmitted is

$$E[pkts] = N + Np + Np^2 + \ldots + Np^{i-1} = \frac{N(1-p^i)}{1-p}$$

When p^i is small, $E[pkts] \approx N/(1-p)$. Packet loss errors in this model are assumed to be statistically independent.

The response time for a transfer of N packets then consists of two components: the time spent transmitting (or retransmitting) packets in each round, and the time spent waiting for acknowledgements at the end of each round. If the transmission rate r remains constant for the duration of the transfer, then the time spent in transmission and retransmission is N/qr where $q = (1-p)$. The time spent waiting for acknowledgements is given by $R * E[rounds]$, where R is the round trip time to the destination[6], and $E[rounds]$ denotes the expected number of rounds to complete the transfer. If no errors occur, $E[rounds] = 1.0$, and the only wait is for the final (and successful) acknowledgement. When errors occur, $E[rounds] > 1$, and additional rounds are required for error recovery.

Figure 2 depicts the expected number of rounds $E[rounds]$ to complete an N packet transfer for various values of N and p. The values shown here were obtained analytically and verified by simulation. $E[rounds]$ is a monotonically increasing function of N and p. As p increases, the expected number of rounds required increases sharply (e.g., for $p > 10^{-2}$). The variance in the number of rounds required (not shown in the figure) also increases.

Mathematically, the probability that a transfer succeeds in at most i rounds is given by

$$Pr[rounds \leq i] = (1-p^i)^N$$

since each packet succeeds in at most i attempts, and there are N packets in total, each failing independently [11]. When $Np \ll 1$, the expected number of rounds is approximately $1 + Np/q$ [11]. For $Np \gg 1$, $E[rounds] \approx 1 - (log\ N/log\ p)$, since it takes $i-1$ rounds

[5] It is assumed that acknowledgement packets are never lost.

[6] In this simple model, R is assumed to be constant. R includes the network propagation delay, as well as the processing and queueing delay. To avoid dependency on any particular network speed or size, R is expressed in units of the packet transmission time.

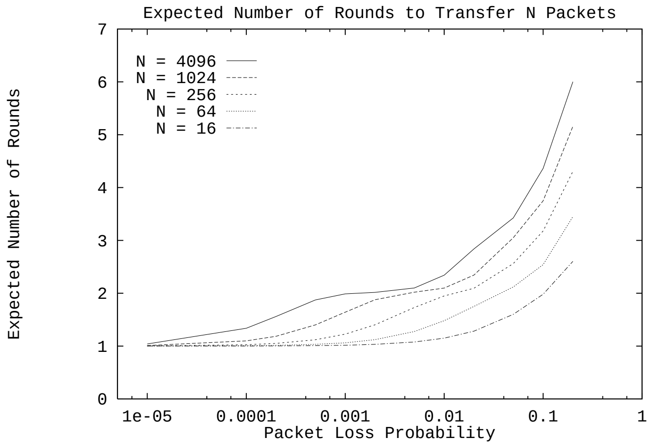

Figure 2: Expected Number of Rounds to Complete an N Packet Transfer

until the number of remaining packets $N' = Np^{i-1}$ satisfies $N' \leq 1$. The last round is always successful, by definition, and thus $i \approx 1 - (log\ N/log\ p)$.

The expected response time $Response(r)$ for the transfer is then:

$$Response(r) = \frac{N}{qr} + R * (1 - \frac{log\ N}{log\ p})$$

where N is the number of packets to be transmitted, R is the round trip time to the destination, p is the packet loss probability when transmitting at rate r, and $q = 1 - p$.

3.3 The Response Time Curve

The response time optimization problem is depicted graphically in Figure 3. Consider a host A that has N packets to transfer (reliably) to host B. In this example, $N = 64$, and the round trip time R between hosts A and B is $R = 100$. The downward-sloping line in the graph depicts the time N/r to transmit N packets at rate r. To minimize transmission time, host A clearly wants r to be as large as possible. However, as r increases, the probability of packet loss increases[7], and more packets are dropped. Packets that are lost must be retransmitted, and may again experience loss.

The expected number of errors when transmitting N packets with loss probability p is Np. Over all rounds, the expected number of errors before the transfer is successfully completed is Np/q, where $q = 1 - p$. Figure 3 illustrates the expected number of errors Np/q as a function of r. The number of errors increases with r. The resulting transmission time $T(r)$ for the N original packets and the Np/q retransmissions is $N/r + Np/qr = N/qr$, and is also shown in Figure 3.

[7]This example uses the loss-load curve from Figure 1. There are $M = 5$ senders, sending at rates 0.1, 0.2, 0.3, 0.4, and 0.5. The value of k is 2.

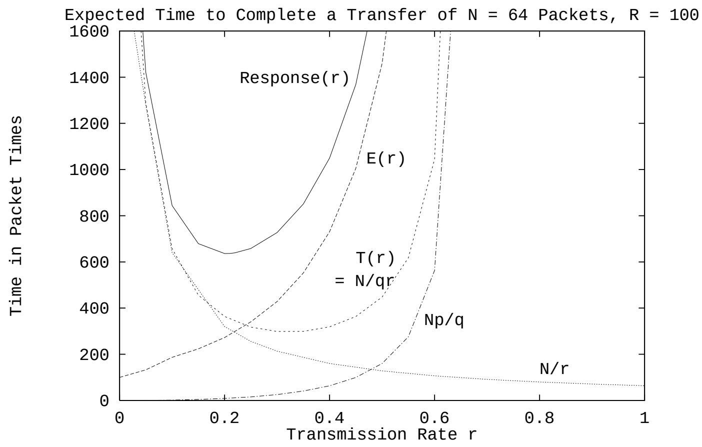

Figure 3: The Structure of the Response Time Curve $Response(r)$

Errors can also increase the number of rounds required to successfully complete the transfer. Each round requires a timeout period of size R, the round trip time from A to B, adding to the response time for the transfer. Figure 3 illustrates the error recovery cost $E(r) = R * E[rounds]$ for our example.

The total response time $Response(r)$ to successfully transfer N packets at rate r depends on the time $T(r)$ spent in transmission (and retransmission), as well as the time $E(r)$ spent in error recovery. The sum of the two functions $T(r)$ and $E(r)$ defines the response time curve $Response(r)$, also shown in Figure 3. The response time curve $Response(r)$ is concave up with a minimum at some rate r^*.

To minimize response time, a host chooses a rate r^* such that $\frac{d}{dr}(Response(r)) = 0$. The operating point r^* that minimizes response time for an N packet transfer with round trip time R is always less than the throughput optimum rate r^*_τ. The response time optimum rate r^* increases with N and decreases with R. For $R = 0$ or $N = \infty$, r^* approaches r^*_τ.

Given the loss-load curve, a sender can directly compute the response time curve, and can compute r^*. The exact value of r^* depends on the shape of the loss-load curve, as well as on N and R.

3.4 Validation of Response Time Model

There are several sources of inaccuracy in the response time model due to the simplifying assumptions made. In particular:

- The round trip time R is assumed to be constant. Variations in queueing delay are ignored. This assumption is reasonable only for short transfers, or if the propagation delay is the dominant component of the round trip time.

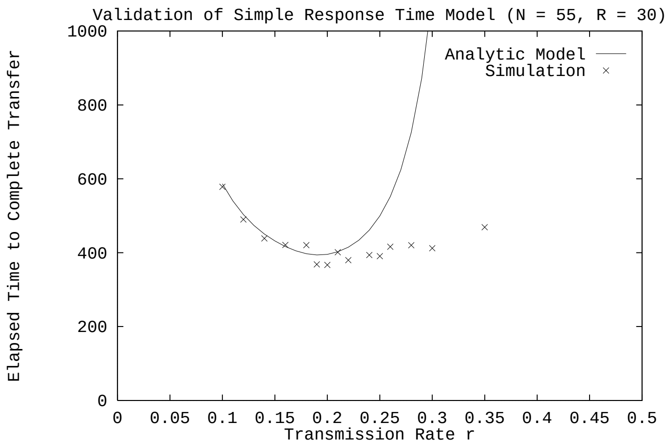

Figure 4: Simulation Validation of Response Time Model

- Senders wait a full round trip time R before beginning the next round of retransmission. In networks with per-packet acks and nacks, retransmission of lost packets could start much earlier. With this assumption, our model overestimates the time required for a reliable N packet transfer.

- Our model uses the exact value of R for timeout and retransmission. In real networks, senders use a retransmit timer which must, to be effective, be set to a value *larger* than the round trip time R. With this assumption, our model underestimates the time for a reliable N packet transfer.

- The transmission rate r and loss probability p are assumed to be constant for the duration of the transfer. The rate r represents the peak rate when transmitting the packets of each round. In real networks, r may be adjusted dynamically based on changes in network load.

In spite of these simplifying assumptions, the simple response time model works well in practice. For small R, the transmission time dominates, and the error in the approximation is low. For R very large (relative to N), the round trip time R dominates the response time, and the relative error is again small. For intermediate values of R, the relative error is usually 20-30% [16].

Figure 4 shows a simulation experiment validating the response time model. One sender is attempting to transfer $N = 64$ packets to a receiver across the network. The base round trip time for the network path is 20. Several other senders are active in the network, creating a dynamic flow of competing traffic. The solid line shows the response time curve derived using the response time model. The x's (obtained by simulation) represent the actual response time for the transfer for various choices of the sender rate r. The response time values obtained by simulation match well with those predicted by the response time model. For

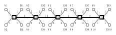

Figure 5: Network Configuration for Response Time Simulations

our purposes, then, the response time model is accurate enough. The model is also attractive since it presents a tractable model for analysis.

4 Simulation Results for the Response Time Model

Earlier work [17] has presented simulation results for the loss-load model for very simple network and traffic models. This section presents the results of a much more elaborate simulation, complete with dynamic network traffic.

4.1 Network and Traffic Model

The network configuration used for the response time simulation experiments is shown in Figure 5. The configuration is based on a simple topology called the "connected star". A single connected star contains two source nodes (S_i) and two destination nodes (D_i) arranged in a star configuration around a single gateway (G_i). Single star components can be connected together in series to form arbitrarily long networks. A connected star network with N stages has $2N$ sources and $2N$ destinations, and thus $4N^2$ possible source-destination pairs. Connected star networks offer an opportunity to study connections that have different numbers of hops and different round trip times. At the same time, the regular topology of the connected star makes routing trivial, and makes the simulation results easier to understand.

The specific network configuration used for the simulations is a connected star with 5 stages, as shown in Figure 5. All links have the same capacity, and each gateway has 100 buffers. The propagation delays for the links are also shown in Figure 5. The round trip propagation delay between the farthest pair of hosts[8] is $R = 52$.

The simulation experiments compare two feedback-based approaches to congestion control: the loss-load approach (for $k = 10$), and the slow-start approach [7]. Slow-start is the

[8]This maximum round trip time is still small compared to the delay-bandwidth products expected for wide-area broadband networks (e.g., tens of milliseconds, and thousands of packets). However, the round trip delay considered here is still large enough to demonstrate interesting performance characteristics of feedback-based control algorithms. The bounded pipe size also helps to keep the size and duration of simulation runs under control.

window-based congestion control algorithm currently implemented in the 4.3 BSD TCP networking code. The slow-start algorithm has two phases: *slow start* and *congestion avoidance*. New connections start in the slow start phase with a window size *cwnd* (the *congestion window*) of one packet, and increase *cwnd* by one for each successful acknowledgement received, permitting the connection to increase its rate of transmission. The window size continues to increase in an exponential fashion, doubling each round trip time, until a packet loss occurs. Packet loss is an implicit signal of network congestion. Following a packet loss, a sender restarts in the slow start phase, but remembers the threshold $ssthresh = cwnd/2$ beyond which the congestion occurred. The window size is opened quickly (exponentially) until it reaches *ssthresh*, and is then opened slowly (linearly) from that point on. The window increases by one packet per round trip time. This latter phase is the congestion avoidance phase.

The traffic workload presented to the network is identical for each of the congestion control strategies considered. The traffic consists of 800 transport level connections. Connections arrive according to a Poisson process with rate $\lambda = 0.006$. Each connection specifies a source node, a destination node, and a number of packets to transmit. Each host in the network is equally likely to be chosen as the source or destination for each connection.

Each connection represents one sender's file transfer activity. The number of bytes per connection is generated from an empirical distribution for ftp traffic [2], using the tcplib package [3]. Approximately 70% of the connections transfer fewer than 1000 bytes. The network uses fixed-size 48-byte packets (cells) for transmission. All times in the simulation are normalized to the transmission time of 1 cell.

4.2 Performance Metric: Normalized Response Time

The performance metric for the simulation experiment is *normalized response time*. The normalized response time (NRT) is defined to be the ratio of the actual elapsed time to complete a reliable N packet transfer to the theoretical minimum possible elapsed time for the same transfer (i.e., the elapsed time it would take to complete the transfer if it were the sole active connection in the network, going full speed). Mathematically,

$$\mathrm{NRT} = \frac{actual\ elapsed\ time\ for\ the\ transfer}{minimum\ possible\ elapsed\ time\ for\ the\ transfer}$$

The NRT represents the degree to which the actual response time for a connection is inflated by the presence of other traffic on the network. An optimal transfer would result in NRT = 1.0. Two senders sharing the network equally with large transfers would both obtain NRT = 2.0. Since the NRT metric represents the actual elapsed time for a transfer, the measure automatically includes the effects of packet loss, retransmission, and queueing delays within the network. The larger the NRT, the slower the response time, and the less happy users are.

4.3 Simulation Results

The results from the simulation are presented graphically in Figures 6 to 13. Each of these figures is a scatter plot showing the NRT measure for each of the connections in the simulation. The exact NRT value for each connection depends on many factors: the number of packets to transmit, the round trip time to the destination, the source and destination chosen, and the dynamic load on the network while the connection is active.

On each page, the topmost figure presents the results for the loss-load approach, and the bottom figure presents the results for slow-start. Both figures are plotted on the same scale to facilitate direct visual comparison. Note that the vertical axis in all cases is logarithmic. For the loss-load model, all NRT values observed were less than 100. For slow-start, 7.6% of the NRT values exceeded 100.

Figures 6 and 7 show the connections in order of their arrival into the network. There is no discernible pattern to these plots. After some initial start-up phase (say, 200 connections), the network becomes heavily loaded, and remains that way for the duration of the simulation.

Figures 8 and 9 show the same NRT data for each connection, but sorted by the number of packets in each connection. Here again, the plot looks mostly random, although there are several interesting features. First, most connections send very few packets (a consequence of the ftp model used). Second, senders that have exactly 2 packets to transmit always take at least 2 round trip times using slow-start. The loss-load model does not have this restriction. Finally, in both figures, there is a general upward trend from left to right. That is, the more packets that a connection has to transmit, the higher its NRT value tends to be.

Figures 10 and 11 show the NRT data sorted by the round trip time of each connection. The round trip time shown is the base round trip propagation delay between the two connection endpoints. It does not include dynamic factors, such as queueing delay. As a result, there are only a finite number of possible round trip times, ranging from 4 to 52. The additional line in the graph connects the means of each of these vertical columns (i.e., the mean NRT value for all connections with the same round trip time).

The slow-start algorithm shows a definite bias against senders with longer round trip times (see Figure 11). Senders with long delay paths, on average, see a higher normalized response time than senders with shorter paths. This is not surprising since the slow-start algorithm, in the congestion avoidance phase, increases the window size by approximately one packet every round trip time. Thus senders with longer round trip times increase their window size at a slower rate. Similarly, when recovering from a packet loss, senders with long round trip times spend more time in the slow start phase. This result accounts for the behaviour shown in Figure 11. The loss-load algorithm shows much less dependence on round trip time (although there is some).

Finally, Figures 12 and 13 show the connections sorted by the number of hops traversed, where the number of hops (links) traversed ranges from 2 to 6. Again, the slow-start algorithm shows a bias against senders with longer paths. The loss-load approach (as claimed in Theorem 3) shows little dependence on the number of hops traversed.

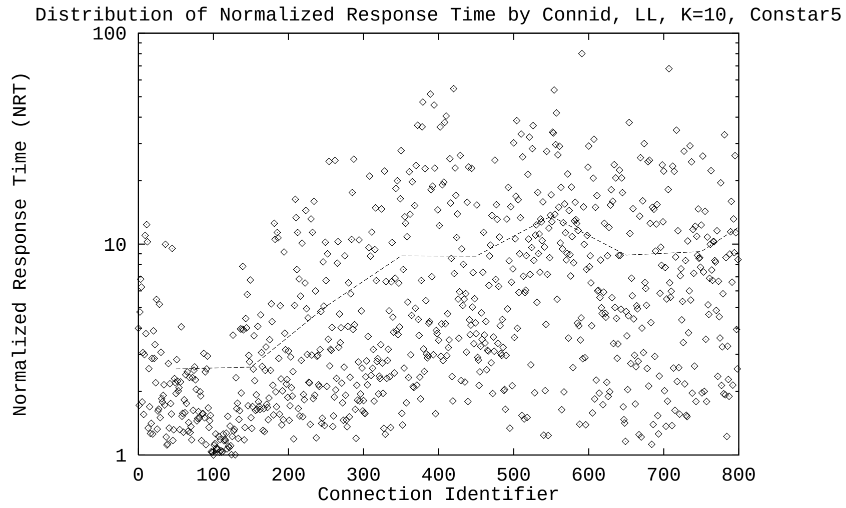

Figure 6: Distribution of NRT by Connection Identifier (Loss-Load)

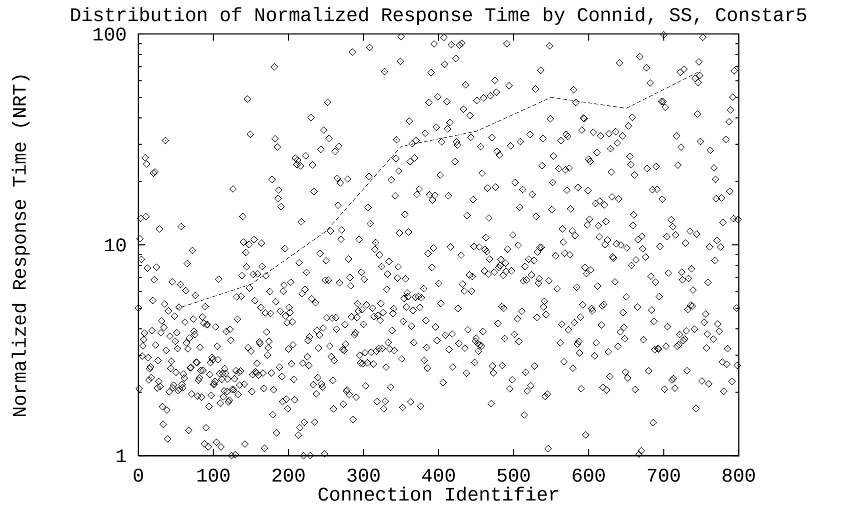

Figure 7: Distribution of NRT by Connection Identifier (Slow-Start)

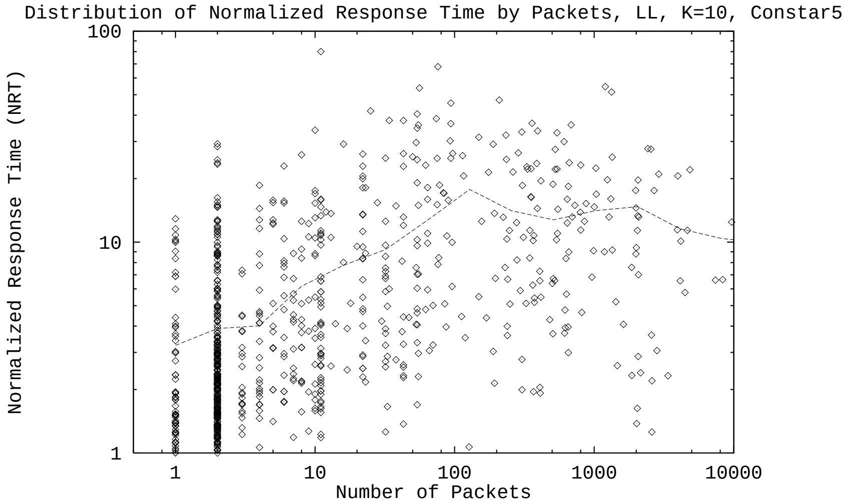

Figure 8: Distribution of NRT by Number of Packets (Loss-Load)

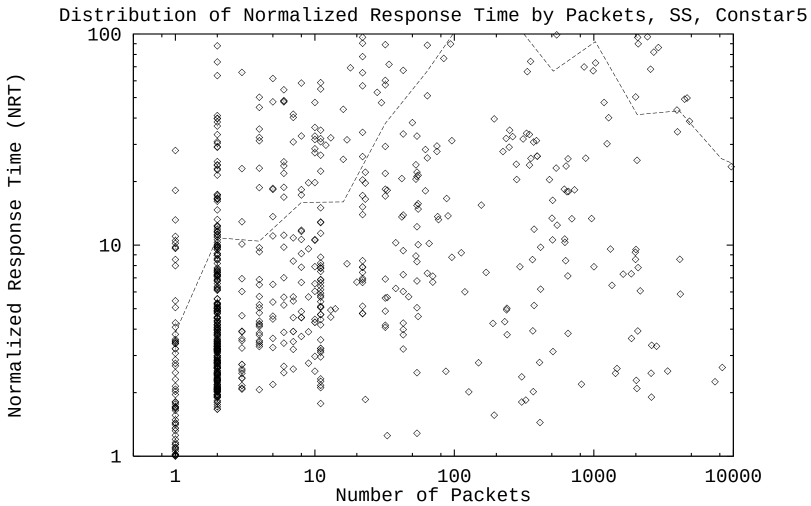

Figure 9: Distribution of NRT by Number of Packets (Slow-Start)

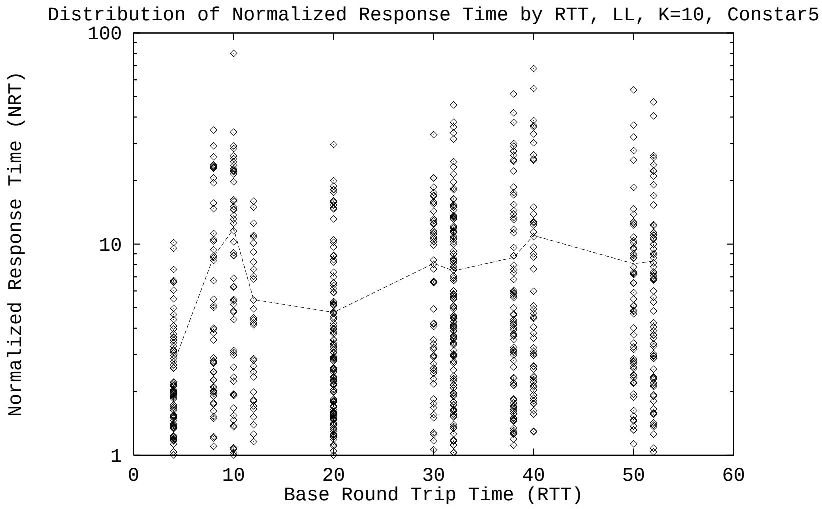

Figure 10: Distribution of NRT by Round Trip Time (Loss-Load)

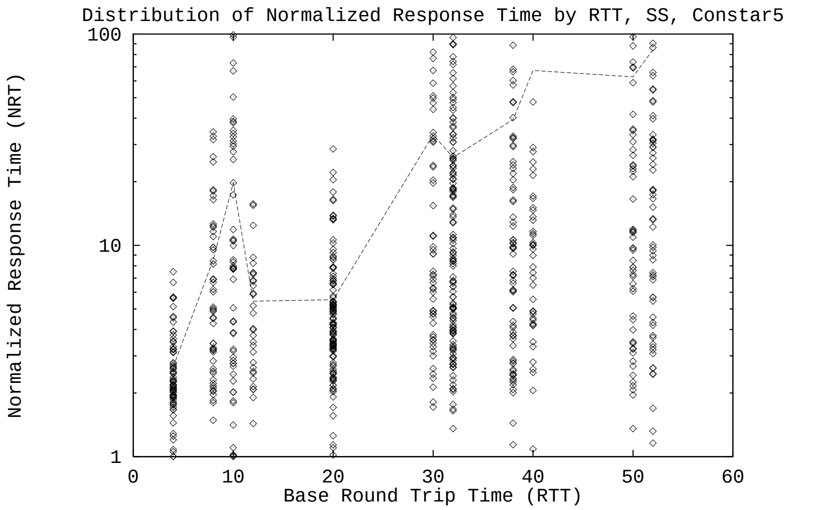

Figure 11: Distribution of NRT by Round Trip Time (Slow-Start)

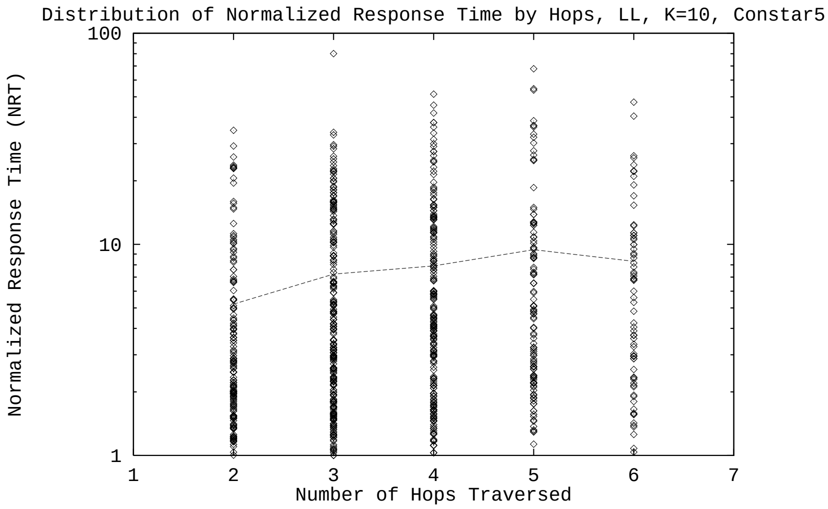

Figure 12: Distribution of NRT by Number of Hops (Loss-Load)

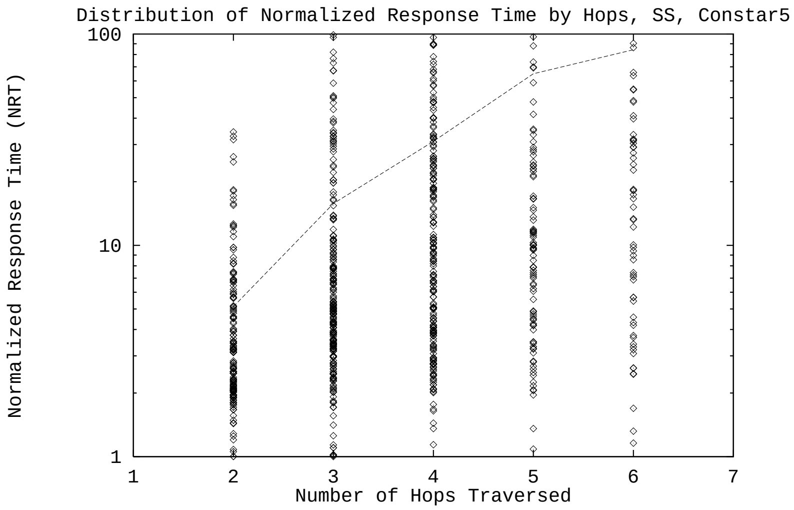

Figure 13: Distribution of NRT by Number of Hops (Slow-Start)

Table 1: Normalized Response Time (NRT) Simulation Results

Control Strategy	Normalized Response Time	
	Mean	Variance
Loss-Load, $k = 10$	9.05	94.06
Slow-Start	18.99	1433.08

4.4 Analysis of Results

Table 1 summarizes the results for the response time simulation experiments. The results reported here consider only the last 600 connections from the 800 connections simulated. The first 200 connections are excluded from the analysis to eliminate start-up effects. The average number of simultaneously active connections during the simulation for the loss-load model is approximately 20. For slow-start, this number was significantly higher.

On average, the loss-load approach provides a 50% lower NRT than the slow-start approach. The advantage of the loss-load approach is clear. New senders starting up know exactly what transmission rate to choose, rather than having to find this out over the span of several round trip times. Furthermore, when load changes occur, senders know how to react and adjust to conditions in the network.

Tables 2 and 3 further classify the NRT simulation data by connection size and round trip time (chosen arbitrarily). Each entry in the table shows the number of connections in that category, the mean NRT for those connections, and their variance (in parentheses).

These tables provide confirmation for several of the observations made in Section 4.3. First, the slow-start approach shows much more bias against senders with long round trip times than the loss-load approach does. For example, the mean NRT values across each horizontal row of Table 3 increase significantly. In Table 2, these increases are much less dramatic.

Second, the loss-load approach performs significantly better than the slow-start approach for connections exchanging a medium to large number of packets. Slow-start, interestingly enough, performs better for connections with either a very small or a very large number of packets than it does on connections with a large number of packets. A possible explanation for this phenomenon is as follows. For a small number of packets, the connection requires only a few round trip times, so slow-start performs satisfactorily. For a very large number of packets, the connection usually lasts long enough for slow-start to open up a fairly large window. If the connection operates most of the time in the congestion avoidance phase with a reasonable window size, the connection will achieve a reasonable level of throughput. For a medium to large number of packets, the slow-start algorithm must spend more time in the slow-start phase than in the congestion avoidance phase. By the time the window is finally opened to a value large enough to achieve reasonable throughput, the connection has almost completed its transfer.

Third, the loss-load approach provides much more consistent performance across all clas-

Table 2: NRT Simulation Results for Loss-Load, $k = 10$

Number of Packets	Base Round Trip Time			Overall Total
	Short (< 15)	Medium	Long (> 35)	
Small (< 10)	85 5.76 (56.57)	142 4.53 (13.93)	123 5.69 (15.56)	350 5.24 (25.20)
Medium ($10 - 99$)	35 12.67 (216.22)	58 11.08 (81.28)	37 18.59 (241.87)	130 13.64 (173.46)
Large ($100 - 999$)	26 9.81 (46.24)	28 14.90 (62.48)	22 23.36 (111.03)	76 15.61 (100.09)
Very Large (≥ 1000)	15 10.51 (52.89)	18 12.28 (17.15)	11 23.26 (256.08)	44 14.42 (115.69)
Overall Total	161 8.36 (97.63)	246 7.82 (51.50)	193 11.18 (138.47)	600 9.05 (94.06)

Table 3: NRT Simulation Results for Slow-Start

Number of Packets	Base Round Trip Time			Overall Total
	Short (< 15)	Medium	Long (> 35)	
Small (< 10)	85 3.62 (15.61)	142 5.47 (35.08)	123 10.99 (317.19)	350 6.90 (136.38)
Medium ($10 - 99$)	35 8.49 (80.63)	58 24.42 (574.36)	37 50.79 (4211.08)	130 30.43 (2119.30)
Large ($100 - 999$)	26 11.63 (157.03)	28 31.65 (1325.32)	22 96.77 (7341.99)	76 48.96 (4511.45)
Very Large (≥ 1000)	15 12.69 (221.83)	18 35.51 (1418.75)	11 60.31 (3268.24)	44 36.10 (1951.05)
Overall Total	161 6.51 (78.70)	246 14.30 (512.86)	193 33.56 (3114.99)	600 18.99 (1433.08)

sifications of connections than the slow-start approach. The reason for this is that senders using the response time model factor both the round trip time and the number of packets into their choice of a transmission rate. Senders with long round trip times are less aggressive in their choice of transmission rate (i.e., operate at a rate $r < r^*_\tau$ that is on a flatter part of the loss-load curve), making them less likely to lose packets, and less vulnerable to variations in network load. Senders that have less than one round trip time's worth of data to transmit are also likely to be less aggressive, and lose fewer packets. Thus the loss-load model provides fairer service for all types of senders. The variances within each class (shown in parentheses) are also significantly lower for the loss-load model.

Finally, the loss-load model provides much more control over the average level of packet loss in the network. Setting the parameter k in the loss-load formula bounds the level of packet loss in the network to $\frac{1}{k+1}$. In an aggregate network of users, the overall level of packet loss in the network may be even lower than the bound predicted by the loss-load model In the simulation results presented here, for example, the observed level of packet loss in the network for the loss-load model was 7.1%, slightly below the predicted value of 9.09% for $k = 10$. Since senders optimizing file transfer response time are operating at a rate $r < r^*_\tau$, they experience a packet loss probability $p < \frac{1}{k+1}$.

5 Conclusions

There are several conclusions that follow from this work. First, it is possible to construct a simple mathematical model for file transfer activity in a lossy network environment. This response time model incorporates the number of packets N to transmit, the round trip time R to the destination, the transmission rate r, and the probability of packet loss p. There is some degree of inaccuracy in the model due to its simplifying assumptions, but it is still accurate enough to be useful.

Second, the response time model makes loss-load curves particularly useful for senders that wish to optimize the elapsed time for file transfer activities. A sender uses the response time model to construct a response time curve from the loss-load curve, and chooses an appropriate transmission rate r to minimize response time. The choice of rate depends on the number of packets to be transmitted, as well as the round trip time. For senders with long round trip times, the choice of rate r may be significantly lower than the maximum raw throughput rate r^*_τ in the loss-load model.

Third, the loss-load approach to congestion control provides more predictable performance for end-user file transfer activities than other feedback-based approaches. Simulation results show that the normalized response time metric is reasonably consistent across connections with different round trip times and different numbers of packets to transmit. Slow-start has a natural bias against senders with long delay paths. With the loss-load response time optimization model, this bias is much less pronounced.

Finally, the loss-load model is potentially a very powerful mechanism for feedback-based congestion control in datagram computer networks. The loss-load curve provides the necessary and sufficient information for senders to choose a proper transmission rate based on their traffic requirements and the current network load. The feedback information provided by the loss-load curve also provides enough information for senders to dynamically respond to changes in the network load, both increases and decreases.

Further work is required to identify the true limits of feedback-based approaches to congestion control in high-speed networks. The work reported here is only one step toward this goal. Work continues on characterizing the full dynamics of a loss-load congestion control strategy, particularly as the feedback delays increase. Applying the loss-load principles to characterize other aspects of network service, such as delay, is also under investigation.

6 Acknowledgements

Funding for this research was provided by the Natural Sciences and Engineering Research Council of Canada (NSERC), through research grant OGP0121969.

References

[1] E. Biersack.
Performance evaluation of forward error correction in ATM networks.
In *Proceedings of SIGCOMM'92*, pages 248–257. ACM, August 1992.

[2] R. Caceres, P. Danzig, S. Jamin, and D. Mitzel.
Characteristics of individual application conversations in TCP/IP wide-area internetworks.
In *Proceedings of SIGCOMM'91*, pages 101–112, September 1991.

[3] P. Danzig and S. Jamin.
tcplib: A library of TCP internetwork traffic characteristics.
Technical report, Computer Science Department, University of Southern California, September 1991.

[4] S. Floyd and V. Jacobson.
Random Early Detection gateways for congestion avoidance.
Technical report, Lawrence Berkeley Laboratory, University of California, September 1992.

[5] H. Gilbert, O. Aboul-Magd, and V. Phung.
Developing a cohesive traffic management strategy for ATM networks.
IEEE Communications Magazine, 30(10):36–45, October 1991.

[6] I. Habib and T. Saadawi.
Controlling flow and avoiding congestion in broadband networks.
IEEE Communications Magazine, 30(10):46–53, October 1991.

[7] V. Jacobson.
Congestion avoidance and control.
In *Proceedings of SIGCOMM '88*, pages 314–329. ACM, August 1988.

[8] R. Jain.
Congestion control in computer networks: Issues and trends.
IEEE Network Magazine, pages 24–30, May 1990.

[9] W. Liu, I. Cidon, I. Gopal, and M. Sidi.
Congestion control through input rate regulation.
In *Proceedings of GLOBECOM '89*, pages 1764–1768. IEEE, November 1989.

[10] P. Mishra, H. Kanakia, and S. Keshav.
A hop by hop rate-based congestion control scheme.
In *Proceedings of SIGCOMM'92*, pages 112–123. ACM, August 1992.

[11] A. Mukherjee, L. Landweber, and J. Strikwerda.
Simultaneous analysis of flow and error control strategies with congestion-dependent errors.

In *1990 ACM SIGMETRICS Conference on Measurement and Modeling of Computer Systems*, pages 86–95. ACM, May 1990.

[12] B. Mukherjee.
Integrated voice-data communication over high-speed fiber optic networks.
IEEE Computer, 24(2):49–58, February 1991.

[13] K.K. Ramakrishnan and R. Jain.
A binary feedback scheme for congestion avoidance in computer networks with a connectionless network layer.
In *Proceedings of SIGCOMM '88*, pages 303–313. ACM, August 1989.

[14] D. Reininger and D. Raychaudhuri.
Transport and error concealment for MPEG-2.
Technical report, David Sarnoff Research Center, for the CCITT Experts Group on ATM Video Coding, Draft, February 1992.

[15] C. Turner and L. Peterson.
Image transfer: An end-to-end design.
In *Proceedings of SIGCOMM'92*, pages 258–268. ACM, August 1992.

[16] C. Williamson.
Minimizing round-trip times for high-performance transport communication.
Technical report, Computer Science Department, Stanford University, October 1991.
Ph.D. Thesis.

[17] C. Williamson and D. Cheriton.
Loss-load curves: Support for rate-based congestion control in datagram computer networks.
In *Proceedings of SIGCOMM '91*, pages 17–28. ACM, September 1991.