Dynamic Bandwidth Allocation Using Loss-Load Curves

Carey L. Williamson
Department of Computer Science
University of Saskatchewan
57 Campus Drive
Saskatoon, SK, CANADA
S7N 5A9
`carey@cs.usask.ca`

August 21, 1995

Abstract

Loss-load curves are a recently proposed feedback mechanism for rate-based congestion control in packet-switched networks. The novel feature of loss-load curves is that they explicitly provide a direct mathematical relationship between the offered load to the network and the level of packet loss within the network.

With their feedback information, loss-load curves provide a means for sources to dynamically compete for and share the available bandwidth on a network. Switches within the network advertise their aggregate load information to the sources in the form of a loss-load curve. Sources at the edges of the network have the responsibility of choosing their own transmission rate based on their traffic requirements and on the loss-load tradeoff curves provided by the network. Sources can use loss-load curves to maximize raw throughput, to optimize response time for file transfer activities, or to control their end-to-end level of packet loss within the network.

The focus of this paper is on the mathematical underpinnings of the loss-load model. The paper introduces the loss-load model, describes the mathematical foundations of loss-load curves, presents three key theoretical results for the loss-load model, and then discusses the advantages and disadvantages of the loss-load approach to dynamic bandwidth allocation. Formal proofs of the theoretical results are contained in the appendix of the paper.

1 Introduction

The dynamic bandwidth allocation problem requires the allocation of the available network bandwidth to those sources requiring the bandwidth *right now*. The allocation should be as quick, as fair, and as efficient as possible. For example, if M identical sources require the bandwidth, then each source should get $1/M$ of the available bandwidth. If a new source is added to the network, then existing sources should yield enough of their bandwidth to allow the new source to obtain its share. If an existing source departs from the network, then the remaining active sources should be able to compete for and share the bandwidth that has just been released.

Loss-load curves are a recently proposed congestion control mechanism that can provide exactly this function [25]. That is, loss-load curves allow sources on the network to dynamically compete for and share the available bandwidth on the network.

In the loss-load model, each switch in the network monitors local traffic load and provides a *loss-load curve* as feedback to the sources at the edges of the network. The loss-load curve provides a concise mathematical representation of the aggregate load information at that switch and, more importantly, a mathematical characterization of the switch's packet loss behaviour under overload. In particular, the loss-load curve expresses, for each source, the expected probability of packet loss for that source, as a function of that source's offered load. Sources at the edges of the network have the responsibility of choosing their own transmission rate based on the tradeoff between raw throughput and packet loss provided by the loss-load curve.

There is no notion of "reserved" bandwidth in the loss-load algorithm. At a given point in time, each source has its own instantaneous share of the total network bandwidth, but new sources may arrive at any time to take away some of the currently allocated bandwidth. Similarly, existing sources may depart at any time, effectively increasing the available bandwidth for the remaining sources. Thus the instantaneous share of the total network bandwidth may change at any time. It is the responsibility of the source itself to increase or decrease its bandwidth request in response to dynamic changes in the network traffic.

A unique characteristic of the loss-load approach is its *aggressive* nature. That is, the loss-load algorithm allows the sum of the source bandwidth requests to exceed the available capacity of the network, resulting in a non-negligible (but bounded) level of packet loss within the network. In fact, the algorithm is stable only at levels of slight overload and packet loss.

This approach to dynamic bandwidth allocation is suitable for loss-insensitive, delay-insensitive, or low priority traffic (i.e., traffic that can tolerate the higher levels of packet loss that occur in this algorithm, or can tolerate the delay of end-to-end retransmission to recover from packet loss). Examples of such traffic include low priority fax, background file transfer, or electronic junk mail. Such traffic is often called *reservationless* traffic because of its relaxed quality of service requirements.

Network providers, such as the telecommunications companies, could offer such reservationless service at a lower price to encourage usage of slack capacity in the network, similar to the "transport auction" concept defined by Grover and Hayer [8, 10]. The amount of network capacity allocated to the reservationless traffic can be decided by the network provider, and can in fact be changed dynamically with the time of day. Traffic that requires guaranteed bandwidth or that has stringent requirements on packet delay, loss, and retransmission (e.g., standard voice calls, or viewing an MPEG movie from a video server) should use the conventional resource reservation procedures, and pay the full tariff rate. Traffic with relaxed quality of service requirements can compete for the remaining bandwidth, using the loss-load algorithm.[1]

In other words, the loss-load approach provides a means to dynamically allocate spare capacity in the network to reservationless traffic. In the remainder of this paper, the term network traffic implicitly means reservationless traffic, and the term network capacity means network capacity currently available for reservationless traffic.

The purpose of this paper, then, is to establish the mathematical foundation for loss-load curves, and to summarize the advantages and disadvantages of the loss-load approach to dynamic bandwidth allocation.

The remainder of this paper is organized as follows. Section 2 presents the philosophy behind the loss-load model, and Section 3 the mathematical details of the loss-load formula. Section 4 presents three key theoretical properties of the loss-load algorithm, for which proofs are contained in the appendix. Section 5 discusses the advantages and disadvantages of the loss-load approach with respect to other congestion control or bandwidth allocation strategies. Finally, Section 6 concludes the paper.

2 The Loss-Load Model

2.1 Network Model and Assumptions

The network environment considered by the loss-load algorithm consists of *sources*, *switches*, and *links*. Sources[2] are the generators of traffic on the network, injecting packets that traverse one or more links and switches en route to their destination.

The model assumes a connection-oriented (i.e., virtual circuit or virtual channel) network. The source chooses the path for its traffic at the time of call setup, and specifies a desired bandwidth request r, which is expressed in bits per second or packets per second.

[1] In ATM (Asynchronous Transfer Mode) networks, there are currently three proposed traffic classes: Constant Bit Rate (CBR), Variable Bit Rate (VBR), and Available Bit Rate (ABR). CBR traffic has the most stringent quality of service (QOS) requirements, while ABR has the least stringent QOS requirements. The loss-load approach to dynamic bandwidth allocation is potentially applicable to ABR services in ATM networks.

[2] The loss-load model intentionally does not define what a "source" is. A source could be a host, a user, a transport-level connection, or even another network. The definition of source is an implementation decision.

The traffic generated by sources is based on a fluid flow model. Packets are fixed in size, and each source releases packets in a periodic fashion according to its current rate. Sources use a leaky bucket [2, 24] or similar mechanism to transmit packets at their chosen rate.

Sources can adjust their bandwidth request (i.e., rate of transmission of packets) at any time based on their (changing) traffic requirements and the current network load. Bandwidth requests are carried in packet headers, and are processed by switches when computing load information.

Switches monitor network load and provide explicit feedback to the sources using loss-load curves. Loss-load curve information is maintained on a *per link* basis, with one loss-load curve for each of the switch's outgoing links. Each loss-load curve is derived from the arriving traffic load and the capacity of the outgoing link for forwarding packets. Each switch operates independently of the other switches.

The model assumes a homogeneous network. All switches use the loss-load model to send out loss-load curves, and all sources use loss-load curves to choose their sending rates. Each source acts independently, and each source is *selfish* in the sense that it seeks to maximize its own end-to-end performance in the presence of the traffic generated by other sources. Some sources may be greedy (misbehaving) sources.

2.2 Loss-Load Philosophy

The loss-load curve provides a very general resource allocation mechanism, allowing all sources on the network to dynamically compete for the available network bandwidth and share it, fairly, on a statistical multiplexing packet-switching basis.

The notion of fairness in the loss-load model is that each source should get its *fair share* [4] of the bandwidth, if it wishes, subject to the bandwidth requests of the other sources. If the sum of the bandwidth requests from all active sources is less than or equal to the available network capacity, then each source receives exactly its requested bandwidth allocation. If the sum of the bandwidth requests from all active sources exceeds the available capacity, then partial bandwidth allocations are made to each source.

The partial allocations are made such that the full bandwidth available is utilized, and all excess traffic is discarded. Excess traffic is discarded by assigning a packet loss probability p_i to each source to reduce source i's requested bandwidth to the allocated bandwidth. Switches are responsible for enforcing this packet loss probability by discarding packets at random from source i's traffic stream, with probability p_i.

The partial bandwidth allocation that a source receives depends on the magnitude of the source's bandwidth request relative to the other sources. A source that requests less than its fair share of the bandwidth receives a low packet loss probability and almost all of its requested bandwidth. A source that asks for more bandwidth may get more bandwidth, but must also expect a higher packet loss probability. Up to a certain point, the more bandwidth that a source asks for, the more bandwidth the source gets. Beyond this point, however, the more bandwidth that a source asks for, the *less* bandwidth the source actually gets. This point of diminishing returns is necessary to prevent greedy or misbehaving sources from monopolizing the network bandwidth.

In other words, greedy sources are "punished" by giving them *less* bandwidth than they would get if they were less greedy, and asked only for their fair share. This approach *encourages* (but does not *require*) cooperation from all sources on the network, and, in the case of a misbehaving source, makes more bandwidth available for the cooperating sources.

It is important to note at this point that "packet loss" in the loss-load model need not necessarily be implemented by discarding packets. Instead, "lost" packets could be "tagged" (as in the virtual leaky bucket control scheme [16]) and forwarded through the network, as long as the network has capacity to accommodate them. In other words, the network only "threatens" to discard packets, but may not actually do so.

Such a "virtual packet loss" approach could be implemented by having the network *lie* about the available capacity for reservationless traffic. If the actual capacity is C_{actual}, the network could advertise a capacity $C_{advertised}$ such that the loss-load approach produces offered traffic exceeding $C_{advertised}$ but still less than C_{actual}. In such a scenario, no packet discarding would need to take place.

This approach, called the *virtual loss-load model* [21, 22], will be revisited briefly in Section 5, once the formal results for the original loss-load model are fully presented. Note, however, that the virtual loss-load approach would work only if the sources did *not* know that the network was lying about the actual capacity. If the sources *did* know that the network was lying, they are less likely to cooperate in their choice of bandwidth requests.

2.3 The Loss-Load Curve

An example of a loss-load curve appears in Figure 1. The curve expresses the relationship between the level of packet loss p for a source at a particular switch as a function of that source's offered load. The loss-load curve is a monotonically increasing function of offered load, where offered load (the source's bandwidth request) is normalized to the capacity C of the switch, to give a fraction $0 \leq f \leq 1$. The packet loss probability reaches 100% for high offered load in order to protect the network from misbehaving sources [25].

The loss-load curve changes dynamically with time, as load conditions change in the network. Switches explicitly tell sources the current loss-load curve, and sources choose their own transmission rate accordingly.

Physically, the loss-load curve can be represented using only two floating point numbers, a and b, defined formally in Section 3.2. A source can compute the entire loss-load curve for a particular switch given only that switch's current values for a and b. This compact representation makes it easy for switches to exchange loss-load curves with sources across the network. These values can be sent on a separate signalling channel (e.g., the D channel in ISDN), piggybacked onto acknowledgment packets returning to the source, or broadcast out to sources at frequent intervals.

Associated with each loss-load curve is a *raw throughput* curve that expresses the fraction

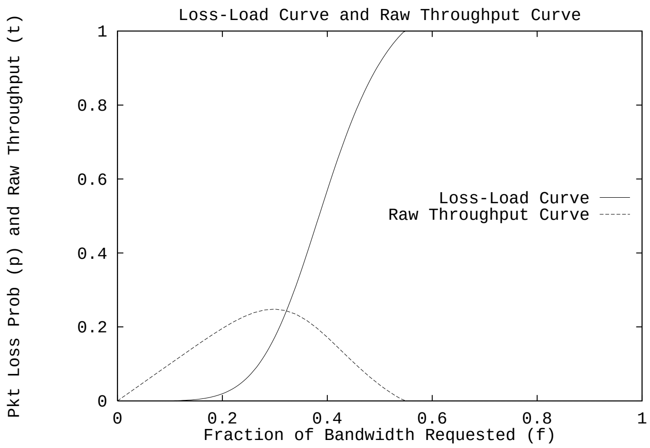

Figure 1: Sample Loss-Load Curve and Raw Throughput Curve

of bandwidth allocated by the switch, for a given source, given the current load conditions. The raw throughput t is computed directly from the loss-load curve, using $t = (1-p)f$ where f is the fraction of bandwidth requested, and p is the probability of packet loss[3].

A sample raw throughput curve appears in Figure 1. The raw throughput curve increases at first, attains some maximum throughput t^*, and then decreases. The maximum raw throughput is achieved at an offered load f^*. Given the loss-load curve, a source can directly compute the raw throughput curve, and can compute f^* (see Section 3.4).

3 Mathematical Details

3.1 Notation

The following notation is used in the ensuing discussion:

C_L – the current capacity (in bits/sec) of link L for carrying reservationless traffic

r_{iL} – the requested bandwidth (in bits/sec) for source i on link L

f_{iL} – the fraction of link L's bandwidth requested by source i ($f_{iL} = r_{iL}/C_L$)

p_{iL} – the packet loss probability enforced by the switch for source i using link L

[3]An analogy may help explain the notion of raw throughput. Consider what happens when you pour water (at rate f) 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 t. In the loss-load model, t is a function of both f and p.

t_{iL} – the raw throughput for source i, expressed as a fraction of link L's bandwidth ($t_{iL} = (1 - p_{iL})f_{iL}$)

τ_{iL} – the raw throughput (in bits/sec) for source i on link L ($\tau_{iL} = t_{iL}C_L$)
(Note that the raw throughput τ_{iL_1} for source i on one link L_1 becomes the effective requested bandwidth r_{iL_2} for the next link L_2 on the path, if any.)

The capacity C_L is needed to translate from absolute to relative bandwidth, or vice versa, at each point in the network (e.g., from r_{iL} to f_{iL}, or from t_{iL} to τ_{iL}). Keeping track of the capacity C_L of each link L makes the loss-load approach to bandwidth allocation work even if the switches and links have different capacities available for reservationless traffic. Where only a single link L is under consideration, the subscript L is dropped from the notation, to improve readability.

3.2 Loss-Load Formula

Mathematically, a single loss-load curve is defined as follows [25]. Assume that a switch S is forwarding traffic from one or more input links onto a single output link L of capacity C. Let M be the number of sources sharing the output link, and let r_i represent the bandwidth request for the ith source, $1 \leq i \leq M$. The switch computes $f_i = r_i/C$ for each source, and $f_{total} = \sum_{i=1}^{M} f_i$. The value f_{total} represents the aggregate traffic load arriving at the switch for link L. If $f_{total} \leq 1$, then all traffic can be accommodated, so the loss probability p_i for each source is set to 0. Otherwise, the switch computes $a = f_{total} - 1$ and $b = \sum_{i=1}^{M} f_i^{k+1}$. The parameter a measures the excess traffic load at the switch, and the parameter b measures the distribution of the total load amongst the competing sources.[4] Given a and b, the switch then computes a packet loss probability p_i for each source such that the excess traffic is discarded (i.e., $\sum_{i=1}^{M} p_i f_i = a$). The packet loss probabilities are assigned in a weighted fashion using $p_i = f_i^k a/b$ for each source i traversing the switch.[5]

The assignment of p_i's in the loss-load algorithm has the following properties [26]:

- If the total offered load is less than or equal to the available capacity, then no packets are discarded. That is, $p_i = 0$ for all i.
- If the total offered load is greater than the capacity, then *all* sources 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 bandwidth requests r_i.

Once the p_i's are determined, then each arriving packet from source i is discarded, at random, with probability p_i, to statistically enforce the desired rate of packet loss.

[4] 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 discussed in Section 4.

[5] Of course, the constraint $p_i \leq 1.0$ is also required.

The resulting loss-load formula is expressed as a function of a source's bandwidth request f and the current load parameters a and b at a switch, as follows:

Loss-Load Formula
The probability of packet loss $P(f)$ for the $(M+1)$st source when requesting a fraction f of the capacity C of a link L shared with M other currently active sources is given by

$$P(f) = \frac{f^k(f+a)}{f^{k+1}+b} \qquad (f_0 \le f \le f_c) \tag{1}$$

where $a = \sum_{i=1}^{M} f_i - 1$ and $b = \sum_{i=1}^{M} f_i^{k+1}$. The term $f+a$ represents what the new excess load would be at the switch if f was requested, and the denominator represents what the new measure of the aggregate load distribution would be. The term f^k determines the $(M+1)$st source's share of the total packet loss required at the switch.

The values f_0 and f_c express the boundary conditions for the loss-load curve. For $f < f_0$, $P(f) = 0$, and for $f > f_c$, $P(f) = 1$. When $f_{total} > 1$, $f_0 = 0$. Otherwise, $f_0 = 1 - f_{total}$. For $a > b$, $f_c = (b/a)^{\frac{1}{k}}$. Otherwise, $f_c = 1$. The resulting loss-load curve $P(f)$ is a strictly increasing function of f in the interval (f_0, f_c) [26]. The curve is also continuous and differentiable in this interval.

3.3 An Example

Consider a single[6] switch S with a single outgoing link L. Assume that the available bandwidth for loss-load traffic is $C = 15$ Mbps, and there are $M = 4$ sources competing for this bandwidth. Suppose that the bandwidth requests r_i for each source are 3 Mbps, 4 Mbps, 4 Mbps, and 5 Mbps, respectively. Assume that the value of k is 2.

The switch S computes the loss-load information for link L as follows. It computes $f_i = r_i/C$ for each source, and then computes $a = \sum_i f_i - 1 = 1.067 - 1 = 0.067$ and $b = \sum_i f_i^{k+1} = 0.083$. Then the packet loss probabilities can be computed using $p_i = f_i^k a/b$. The resulting loss probabilities p_i and bandwidth allocations τ_i are shown in Table 1.

Table 1: Example of Loss-Load Information for Link L, $k = 2$

i	r_i	f_i	p_i	t_i	τ_i
1	3 Mbps	0.200	0.0323	0.1935	2.903 Mbps
2	4 Mbps	0.267	0.0575	0.2516	3.774 Mbps
3	4 Mbps	0.267	0.0575	0.2516	3.774 Mbps
4	5 Mbps	0.333	0.0895	0.3033	4.549 Mbps

[6]While most of the examples given in this paper consider only one switch and link, the loss-load model works equally well with multiple switches and links.

Different choices of k result in different allocations of bandwidth amongst the sources. If one assumes that $k = 10$, then the switch would compute $a = 0.067$ and $b = 6.587 \times 10^{-6}$. The resulting loss probabilities and bandwidth allocations are shown in Table 2.

Table 2: Example of Loss-Load Information for Link L, $k = 10$

i	r_i	f_i	p_i	t_i	τ_i
1	3 Mbps	0.200	0.00104	0.19979	2.997 Mbps
2	4 Mbps	0.267	0.01873	0.26200	3.930 Mbps
3	4 Mbps	0.267	0.01873	0.26200	3.930 Mbps
4	5 Mbps	0.333	0.17054	0.27621	4.143 Mbps

The larger the value of k, the higher the degree of punishment for high rate sources. In the latter case ($k = 10$), source 4 could actually get more bandwidth by reducing its bandwidth request. For example, it could obtain $\tau_4 = 4.178$ Mbps if it chose $r_4 = 4.596$ Mbps. The loss-load curve provides exactly the information that the sources need to make these choices.

Regardless of the value of k, however, the full bandwidth of the link is always allocated. That is, $\sum_i \tau_i = C$. This fact follows directly from the observation that

$$\sum_i t_i = \sum_i (1 - p_i) f_i = \sum_i (1 - \frac{f_i^k a}{b}) f_i$$

$$= \sum_i f_i - \sum_i f_i \frac{f_i^k a}{b} = (a + 1) - \sum_i \frac{f_i^{k+1} a}{b}$$

$$= a + 1 - \frac{a}{b} \sum_i f_i^{k+1} = a + 1 - \frac{a}{b} b$$

$$= a + 1 - a = 1$$

3.4 Source Transmission Strategy

Given the loss-load curve, sources have several options for their choice of transmission rate. They can maximize raw throughput for bulk data transfers, minimize end-to-end response time for file transfers, or attempt to control their average level of packet loss within the network. Note that these are strategies that an *individual* traffic source can choose, given a network that uses the loss-load scheme, regardless of what other traffic sources choose to do.

Maximizing Raw Throughput

A source with a large number of packets to transmit (say, > 1000) would be interested in maximizing raw throughput (i.e., maximizing the delivery rate of successful packets), regardless of the packet loss level at intermediate points in the network. This is the aggressive alternative. Such a source can always retransmit to recover from lost packets.

To maximize raw throughput, a source must maximize the fraction of throughput t at the bottleneck link B along the path. Raw throughput is maximized at a point f^* where $\frac{dt}{df} = 0$. To find f^*, the source differentiates the raw throughput equation ($t = (1-p)f$) with respect to f, sets it equal to zero, and solves for f^*. The resulting equation is $kf^{k+1} + (k+1)af^k - b = 0$ (see the appendix for details). Solving for f^* then requires numerical methods to find the root of a degree $k+1$ polynomial.

> **Example.** Given the four active sources shown in Table 1, and $k = 2$, the maximum raw throughput of $t^* = 0.21049$ for a fifth source is obtained at a bandwidth request of $f^* = 0.31574$, where $p = 0.33333$. This corresponds to an offered rate of 4.736 Mbps, and a raw throughput of 3.157 Mbps. For the example in Table 2, $f^* = 0.26821$, $p = 0.09091$, $t^* = 0.24383$, $r^* = 4.023$ Mbps, and $\tau^* = 3.657$ Mbps.

Where necessary, the subscript τ can be used to clarify that f^*_τ is the offered load that maximizes raw throughput. If the raw throughput context is clear, then the subscript τ can be dropped.

Optimizing Response Time

A source with a smaller number of packets to transmit, or with a longer round trip delay to the destination, may be willing to sacrifice slightly on raw throughput in order to minimize the excessive delay of end-to-end packet retransmission. For such a source, the response time model offers the answer [27].

The response time model attempts to balance the delay required for transmission and retransmission with the delay required for end-to-end retransmission, such that the overall transfer time is minimized. The response time model factors both the number of packets N to be transmitted and the round trip time R into the choice of transmission rate, as well as the effects of packet loss, timeouts, and retransmission. The expected response time for the transfer is given by [27]:

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

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 f, and $q = 1 - p$. The response time curve is U-shaped, with a minimum response time achieved at an offered load f^*. An example of a response time curve is shown in Figure 2.

To minimize response time, a source chooses a rate f^* such that $\frac{d}{df}(Response(f)) = 0$ (by solving for the root of a degree k polynomial). The operating point f^* that

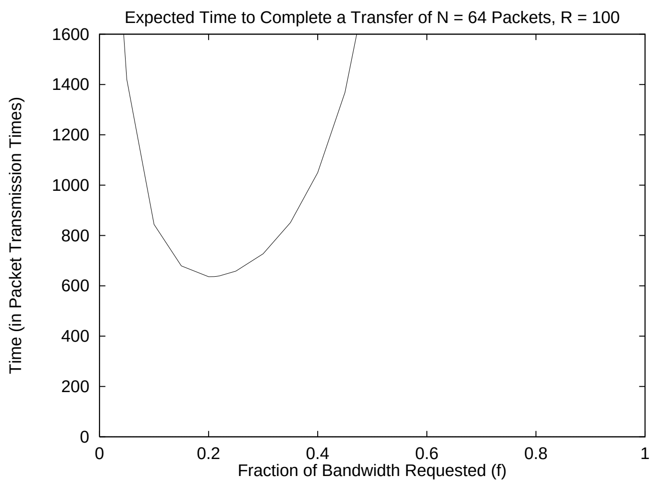

Figure 2: An Example of a Response Time Curve [27]

minimizes response time for an N packet transfer with round trip time R is always less than the raw throughput optimum rate f^*_τ. The response time optimum rate f^* (sometimes written $f^*_{N,R}$) increases with N and decreases with R. For $R = 0$ or $N = \infty$, $f^*_{N,R}$ approaches f^*_τ.

> **Example.** Given the four active sources shown in Table 1, and $k = 2$, the optimum response time for a fifth source with $N = 50$ and $R = 20$ is obtained at a bandwidth request of $f^* = 0.26486$. This corresponds to an offered rate of 3.973 Mbps. The packet loss probability at this point is $p = 0.22919$. For the example in Table 2, $f^* = 0.25357$, $r^* = 3.804$ Mbps, and $p = 0.05131$.

Controlling Packet Loss

A source concerned about its overall level of packet loss can take a conservative approach. That is, a source can intentionally choose a low transmission rate (on the flatter portion of the loss-load curve), with a corresponding lower level of packet loss. By operating on the flatter portion of the loss-load curve, a source is also less sensitive to transient changes in network load than is a source operating at the raw throughput optimum point (at the knee of the loss-load curve).

Given the loss-load curve and an upper bound p_{max} on the desired (or maximally tolerable) level of packet loss, a source determines an offered load f^* such that $P(f^*) \leq p_{max}$. A source determines f^* (also written f^*_p) by solving $P(f) - p_{max} = 0$.

> **Example.** Given the four active sources shown in Table 1, and $k = 2$, suppose that the fifth source desires a packet loss probability that is less than or equal to $p_{max} = 0.01$. The resulting bandwidth request should be less than or equal to

$f^* = 0.07630$ ($r^* = 1.145$ Mbps). For the example in Table 2, $f^* = 0.21719$, and $r^* = 3.258$ Mbps.

3.5 Practical Matters

Several practical matters concerning the loss-load model require further elaboration.

One such matter is the precise definition of the loss-load curve. Note that the loss-load curve for a source (as defined in Equation 1) is always expressed in terms of the aggregate load offered by the *other* active sources on the network. If there are M sources sharing a common switch, then each of those sources sees a *different* loss-load curve: that is, the loss-load curve for source j is determined from the rates r_i of the $M - 1$ other sources, for $i = 1 \ldots M$ and $i \neq j$.

The intuition behind this requirement is simple: the source is not competing against itself for bandwidth. The implication of this requirement is that, technically, the switches must compute and send a different loss-load curve to each source. In practice, it is adequate for the switches to compute and distribute only the aggregate load information (the same for all sources), relying on sources to subtract out their own load contribution when computing loss-load curves. For large M, this distinction becomes unimportant, and only a single aggregate loss-load curve is needed (since each f_i contribution is very small). For small M, however, this requirement is absolutely crucial to the theoretical results discussed in Section 4.

Another such matter is the exchange of loss-load curve information. As described earlier, the loss-load model implies that all switches send all loss-load curves to all sources whenever any load information changes. With this approach, all sources have global network load information and can make informed rate choices for new and continuing traffic flows on any network path. Clearly, however, such an approach is not practical in a large network.

In practice, it is sufficient for switches to send loss-load curves only to traffic sources currently traversing that switch, thereby reducing the volume of loss-load control traffic on the network. Sources initiating traffic flows on new network paths (i.e., unknown load) can obtain up-to-date loss-load information as part of the call setup process, and use that information to choose initial transmission rates.

Furthermore, switches can send loss-load curves periodically, rather than every time the loss-load curve changes. A reasonable choice of update interval (e.g., 10-100 msec) can further reduce control traffic on the network, though at the expense of the timeliness and accuracy of the information exchanged.

Loss-load curve updates can also be sent without requiring additional network packets. For example, the values of a and b (along with the switch/link identifier and the link capacity C) can be piggybacked on acknowledgement packets returning to the traffic source.[7] Indeed, this piggybacking mechanism can be designed so that one switch conditionally overwrites another switch's loss-load information in the acknowledgement packet only if the current

[7]This optimization assumes an acknowledgement-based protocol at the transport layer, and that acknowledgement packets follow the same network path as the data packets, but in the reverse direction.

switch is a more stringent bottleneck than the switch currently represented in the packet. This mechanism would then automatically identify the bottleneck link for a traffic source, reducing the processing and storage of loss-load curves required at sources.

A final practical matter to consider is how the switches find out about source transmission rates. The discussion up to now has assumed that rate information is explicitly carried in all packet headers, and that rate information is updated by the switches whenever the packet is forwarded to reflect the bandwidth assigned on the outgoing link. With this approach, switches have exact rate information for each incoming traffic source.

This approach, however, requires that the switches *trust* the sources to declare their proper rates. One way to circumvent this problem is to have the switches use *rate estimation* to determine source rates. While rate estimation eliminates the need for trusted sources, there is always estimation error and a time lag involved in the estimation process, which reduces the accuracy and responsiveness of the loss-load model. Rate estimation also adds significant overhead to the processing required at the switches. The remainder of this paper assumes explicit declaration of rates in packet headers by trusted sources, and updated rates for each switch and link traversed.

3.6 Summary

The loss-load curve provides enough information for sources to make informed decisions about their choice of transmission rate when competing for bandwidth. By providing the complete loss-load curve to the sources, rather than just f^*_τ, the switches leave it up to sources to choose their own transmission strategy. Regardless of the transmission strategy chosen, a source must always re-evaluate its choice of rate whenever the loss-load curve changes (i.e., in response to the actions and reactions of other sources). The compact mathematical representation of the loss-load curve makes it easy to provide loss-load curves to sources across the network.

4 Theoretical Results

Several theoretic results have been established for loss-load curves when all sources are maximizing raw throughput. These results are reported here.

4.1 Bounded Packet Loss

The first theoretical result for the loss-load model is that it provides a bounded level of packet loss for cooperating sources [26].

Theorem 1 Bounded Packet Loss
At an overloaded switch (i.e., $a > 0$), the packet loss probability when transmitting at the rate

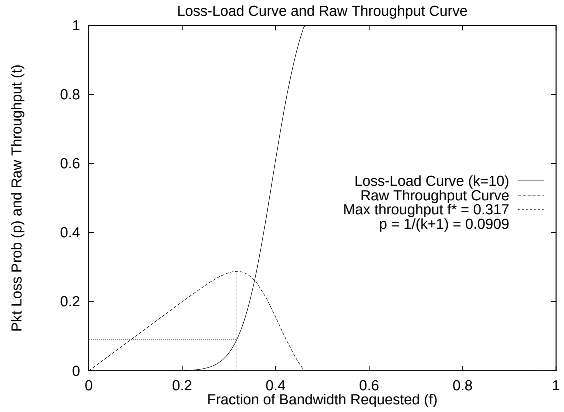

Figure 3: A Graphical Illustration of Theorem 1 for $k = 10$

f_τ^* to maximize raw throughput is always $\frac{1}{k+1}$, where $k > 0$ is a settable integer parameter in the loss-load formula.

Theorem 1 says that the level of packet loss at f_τ^* is always bounded. The bound is always the same, regardless of the number of sources, and regardless of the network load. The bound depends only on k, a settable integer parameter in the loss-load formula. The larger k, the lower the level of packet loss at the raw throughput optimum rate f_τ^*. A proof of Theorem 1 is contained in the appendix of this paper. A graphical illustration of the theorem is shown in Figure 3.

This result illustrates the aggressive nature of the loss-load algorithm. A source achieves its maximum raw throughput by sending *more* traffic than can fit in the network. The intuition here is subtle. Sending less than can fit wastes bandwidth, and does not maximize raw throughput. Sending far too much results in low throughput, since the network discards most or all of the packets from a misbehaving source. In between is an optimum rate f_τ^*, with a non-zero probability of packet loss. The surprising result is that the level of packet loss at f_τ^* is constant: the amount of extra traffic to send is always bounded. In other words, the loss-load approach automatically *controls* the level of packet loss, rather than eliminating it.

The significance of this result is two-fold. First, it provides a *bounded* and *predictable* probability of packet loss for cooperating sources. The level of packet loss at f_τ^* is always known, regardless of the network capacity, regardless of the network load, and regardless of what each other source is doing. Second, the level of packet loss at f_τ^* can be made as low as desired simply by changing k, the control parameter of the algorithm. The larger k is, the less aggressive sources are.

4.2 Convergence

The second theoretical result addresses the convergence and stability properties of the loss-load model [26].

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

Theorem 2 says that the loss-load model always converges to a stable equilibrium. The equilibrium load achieved is independent of the number of sources, independent of the initial rates chosen by the sources, and independent of the feedback delay to each source. A proof of Theorem 2 is contained in the appendix. If all sources are identical, and all are maximizing raw throughput, then in equilibrium each source ends up with the same f^*_τ, the same packet loss probability, and the same bandwidth allocation. Note that convergence is to a stable *overload* of $\frac{k+1}{k}C$. The larger k, the smaller the overload at equilibrium.

The convergence process is iterative, involving action and reaction between sources. Convergence is fast, typically requiring two to four iterations. The speed of convergence depends on k. The larger k, the slower the convergence.

> **Example.** Consider two sources sharing a common bottleneck link, initially empty. Each source has an infinite number of packets to send, and wants to maximize raw throughput. The value of k is 2, and $C = 1$.
>
> Suppose (without loss of generality) that source 1 starts transmitting first. Seeing no load on the bottleneck link, source 1 chooses $f_1 = 1.0$ with throughput $t_1 = 1.0$. When source 2 is ready to send, it sees the loss-load curve representing source 1's offered load. Source 2 then chooses $f_2 = 0.7937$, with a loss probability $p_2 = 0.3333$ and a raw throughput of $t_2 = 0.5291$. The arrival of source 2's traffic causes source 1 (with $f_1 = 1.0$) to lose part of its previous bandwidth allocation. Source 1 now experiences $p_1 = 0.5291$ and $t_1 = 0.4719$.
>
> When source 2 starts transmitting, the load on the link changes, and the switch sends out new loss-load curves. Source 1 sees the new loss-load curve representing source 2's load, and reacts by choosing a new rate $f_1 = 0.7518$. At this transmission rate, $p_1 = 0.3333$ and $t_1 = 0.5012$. This choice of rate causes p_2 to increase to 0.3715, and t_2 to drop to 0.4988.
>
> Following this adjustment, source 2 sees the new loss-load curve for source 1, and selects $f_2 = 0.7500$. Source 1 sees this, and also chooses $f_1 = 0.7500$. At this point, convergence is complete. Equilibrium is achieved at a load of $\frac{k+1}{k} = 1.5$, and each source receives an equal throughput allocation $t_i = 0.5$.

The convergence property holds true for any number of sources and any value of k. For a fixed number of sources, all maximizing raw throughput, the convergence process always approaches the equilibrium load monotonically from above. Intuitively, this is because each source adjusts its rate downward only enough to not be the highest rate source. Convergence is slower for larger values of k because the rate adjustments made at each step are smaller.

If not all sources are maximizing raw throughput, the total offered load at equilibrium will be less than $\frac{k+1}{k}$. As long as at least one source is maximizing raw throughput, equilibrium will be achieved, and a slight level of overload will occur. Simulation results reported in [29] show that in the presence of highly dynamic network traffic, the loss-load model produces network loads that are at or below the equilibrium bound predicted by Theorem 2 the vast majority of the time, depending on the value of k (e.g., 98.2% of the time for $k = 2$, and 90.6% of the time for $k = 10$).

4.3 Multi-Hop Paths

The third theoretical result for the loss-load model concerns its fairness across multi-hop paths in the network [26].

Theorem 3 Multi-Hop Paths
A source traversing a multi-hop path in the network may experience packet loss at more than one congested switch or link along the path. However, as long as the source's transmission rate is chosen based on the loss-load curve for the bottleneck link*, the level of end-to-end packet loss for that source is approximately $\frac{1}{k+1}$, for large k (i.e., $k \geq 10$).*

In other words, only the bottleneck link matters when sources are transmitting over a multi-hop path. The bottleneck link is the link whose loss-load curve and capacity give the lowest value for $r_\tau^* = f_\tau^* C_L$. As long as the source's transmission rate is chosen based on the load at the bottleneck link, the end-to-end level of packet loss is approximately $\frac{1}{k+1}$. A proof of Theorem 3 is contained in the appendix.

Loss at switches other than the bottleneck is negligible since the bandwidth request will be on the flat portion of the loss-load curves for those switches. Furthermore, the effective packet arrival rate (and thus the loss probability) diminishes with each lossy hop traversed, making subsequent loss less and less likely. Simulation experiments [25] show that packet loss at points other than the bottleneck link along a multi-hop path is negligible for $k \geq 10$.

4.4 Setting k

The previous three subsections have shown that the setting of k has a dramatic impact on the behaviour of the loss-load model. A natural question that arises then is: what value of k should be used?[8]

[8] Recall that the value of k in the loss-load model must be a global constant in the network, used by all switches and all sources in their loss-load computations. It does not make sense to use different values of

Clearly, the choice of k involves tradeoffs. Setting k too low (e.g., $k = 1$ or $k = 2$) will result in a high degree of network overload and extremely high levels of packet loss (e.g., 33-50%, see Theorem 1), and possibly significant packet loss bias over multi-hop paths (see Theorem 3). Setting k too high (e.g., $k = 100$) will make the model extremely unresponsive, since the larger the value of k, the slower the convergence to equilibrium (see discussion of Theorem 2). Furthermore, large values of k (e.g., $k > 30$) may be impractical because of the possibility of numerical underflow (particularly on 32-bit machines), and thus imprecision in loss-load computations.

Extensive simulation experience with the loss-load model (reported in [25, 27, 29]) suggests that a value of $k = 10$ provides the most reasonable tradeoff. This setting keeps packet loss for raw throughput optimization below 10% (and even lower when sources use response time optimization), keeps aggregate load within 5-10% of the desired capacity, provides reasonably fast convergence to equilibrium for dynamic traffic flows, and produces minimal packet loss cumulation effects for sources traversing multi-hop paths in the network. Thus the recommended setting for k in the loss-load model is $k = 10$.

4.5 Summary

The loss-load algorithm offers a novel way to dynamically allocate bandwidth to reservationless traffic. Despite the dynamic nature of the bandwidth allocation problem, the loss-load algorithm has formally provable mathematical properties for bounded packet loss, bounded overload, and convergence to a stable equilibrium. 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.

Simulation results [25, 27, 29] have further established the fairness, responsiveness, robustness, and stability of the loss-load algorithm for dynamic network traffic. The response time model is particularly effective in providing equitable service to sources with different traffic workloads and different round trip times [27].

5 Discussion and Related Work

There are numerous other congestion control strategies proposed in the literature that are also capable of providing dynamic bandwidth allocation among competing traffic sources in packet-switched networks [3, 7, 9, 11, 12, 13, 15, 17, 18, 19, 23]. Among these, the five most relevant to this paper are TCP Slow-Start [12], the DECbit binary feedback scheme [23], the packet-pair scheme [15], hop-by-hop flow control [17], and Random Early Detection (RED) gateways [7].

k for different entities in the network, and it does not make sense to vary k dynamically (unless you can guarantee that all switches and sources change to the new k value simultaneously and atomically).

This section discusses the advantages and disadvantages of the loss-load approach with respect to these five strategies. A simulation comparison of the loss-load approach to these strategies, using a suite of benchmark scenarios proposed by Kanakia, Keshav, and Mishra [14], is available in [21].

5.1 Strengths of the Loss-Load Approach

The primary advantage of the loss-load model is that it provides more complete load information to the traffic sources. Approaches such as TCP slow-start or binary feedback can indicate the presence or absence of congestion along a network path, but they cannot indicate where the congestion is, how heavy the congestion is, or who is causing it. With a loss-load curve from each switch in the network, traffic sources in the loss-load model can determine exactly where the congestion is, and how to react to it. There is no need to use additive increase or multiplicative decrease to try and "guess" the proper sending rate: the loss-load curve allows a source to precisely compute its proper sending rate.

Related to this advantage is the fact that loss-load curves provide much greater "control" for traffic sources. Some sources may wish to maximize raw throughput for bulk data transfers; some sources may wish to minimize response time for a finite file transfer; and some sources may wish to achieve a sufficiently low level of packet loss. The network and the switches have no way of knowing what strategy an individual source may prefer. Thus, rather than providing a single number to a source to assign bandwidth, switches instead provide the *rule* that they will use for bandwidth allocation. With the loss-load curve, sources then have the freedom to make their own transmission rate choices, and the knowledge that they need to make these choices properly. No other congestion control strategy has this feature.

The second key advantage of the loss-load model is that it does *not* require cooperation from traffic sources. Almost every other congestion control strategy, including the five mentioned above, require cooperation from all sources in order to work properly. Greedy sources that misbehave (in TCP, for example) can acquire arbitrarily large fractions of the total network bandwidth. In fact, this observation was one of the original motivations for the loss-load approach to congestion control. If all sources cooperate, then a bounded level of load and a bounded level of packet loss result. If not all sources cooperate, then the bound no longer applies: the packet loss for the misbehaving source can be arbitrarily high, and the remaining (cooperative) sources get *more* than their usual share of the bandwidth.

A third advantage of the loss-load model is that elaborate queueing and buffering strategies are not required. The loss-load model works with FIFO queueing, unlike the packet pair scheme, which requires round robin scheduling. In short, there is no need to design a scheduling algorithm that allocates bandwidth on a per-source basis, with per-source queues, and that gives appropriate feedback for sources to discover bandwidth availability. Furthermore, since packet discard decisions are based on source rates, not buffer occupancy (i.e., packets can be thrown away long before the buffer fills up), the loss-load model can work with nominal buffering (e.g., 100 buffers). This approach can potentially offer lower delay for packets traffic traversing the switch, while at the same time eliminating the unfairness, unpredictability, and complicated phasing effects observed in Drop-Tail, Random-Drop, and RED gateways [5, 6, 7].

Finally, the loss-load model offers a high degree of flexibility and extensibility, making it attractive as a bandwidth allocation strategy for current and future high speed networks (e.g., as a means for dynamic bandwidth allocation for ABR traffic in ATM networks). For example, it is straightforward to extend the model to support the notion of priority (or pricing). One approach is to predefine a set of traffic subclasses, with a separate capacity C for each subclass. Sources can then compete for their share of the bandwidth in their assigned subclass, and traffic in each subclass is independent of traffic in other subclasses. The second approach is to associate a fixed priority d_i with each source bandwidth request r_i. A source with priority $d_i = 2$ would have twice the priority of a source with $d_i = 1$. The effect of this approach is simply to add a scalar multiplier $1/d_i$ to all source rates in the loss-load formula, making for an effective bandwidth request of $\hat{r_i} = r_i/d_i$. All other properties of the loss-load algorithm remain the same (i.e., convergence, bounded packet loss, etc.). With this notion of priority, a source with $d_i = 2$ would get twice as much bandwidth at equilibrium as a source with $d_i = 1$, but both would experience the same packet loss probability $p_i = \frac{1}{k+1}$, if both are maximizing raw throughput.

5.2 Weaknesses of the Loss-Load Approach

The main drawback of the loss-load approach is its implementation overhead. The loss-load model requires that each switch do the following for each of its outgoing links: (1) keep track of the transmission rate r_i of each source traversing the link; (2) compute and record the packet loss probability p_i for each source traversing the link; (3) (re)compute the aggregate loss-load curve (and the p_i's) each time any of the source rates change; (4) provide the updated loss-load curve to all traffic sources in the network (or at least the traffic sources traversing the link); and (5) enforce the loss-load curve by discarding, at random, each incoming packet from source i, with probability p_i. Clearly, this is an imposing set of tasks for a switch to perform. For thousands of traffic sources, the storage and processing costs associated with the algorithm may not make it feasible.

Traffic sources also incur overhead. Each source must: (1) receive loss-load curves from all switches in the network (or at least those switches that the source's traffic flow is traversing); (2) determine which of the switches represents the bottleneck link for the traffic flow; (3) compute the optimal sending rate (which may involve numerical methods, such as Newton's Method for root finding); and (4) shape outgoing packet traffic to match the chosen rate. These tasks are much more complicated than those in other congestion control strategies.

The second disadvantage of the loss-load model is its high degree of packet loss (e.g., 5-10% packet loss, for a practical value of k, such as $k = 10$ or $k = 20$). There are very few network applications that can tolerate such a high level of packet loss, even for ABR traffic.

The remedy to this problem is the virtual loss-load model [21, 22]. Such an approach could be implemented by having the network lie about the available capacity for reservationless traffic. If the actual capacity is C_{actual}, the network could advertise a capacity $C_{advertised}$ such that the loss-load approach produces offered load $\frac{k+1}{k}C_{advertised} \leq C_{actual}$. Packet loss could then be effectively reduced to zero.

This approach would work as long as the sources did not know that the network was lying about the actual capacity. If the sources did know that the network was lying, they are less likely to cooperate in their choice of bandwidth requests. In other words, a single greedy source could acquire more bandwidth than a cooperating source. Thus there is no longer an incentive to cooperate, and the bounds on load and packet loss no longer apply. If cooperating sources are assumed, the virtual loss-load model could work very well, with minimal packet loss.

The final problem with the loss-load model lies in the fact that it is a feedback-based control scheme. As with any feedback-based approach (e.g., [1]), it cannot react to congestion problems faster than the time required for the feedback delay. Thus the approach will not necessarily scale well to high speed or wide area networks. The convergence and stability properties of the loss-load approach suggest that it will scale *better* than most feedback-based approaches (since it provides more information to sources, and is less susceptible to oscillatory behaviour, or "ringing"), but simple control theory says that it cannot possibly work well in a wide area network environment. This dependence on feedback delay will be the ultimate limiting factor on the applicability of the loss-load model.

6 Conclusions

Loss-load curves are a novel feedback mechanism for rate-based congestion control in packet-switched networks. Switches in the network explicitly advertise loss-load curves to the sources on the network, and sources have the responsibility of choosing their own transmission rate.

Loss-load curves provide the necessary and sufficient information for sources to dynamically compete for and share the available bandwidth on the network. New sources know what transmission rate to choose. Existing sources know how to respond to load changes in the network, both increases and decreases. Sources can use loss-load curve information to maximize raw throughput, minimize response time, or control their level of packet loss in the network.

Several mathematical properties of loss-load curves make them an attractive mechanism for dynamic bandwidth allocation. Loss-load curves guarantee a bounded level of offered load to the network at equilibrium, as well as a bounded level of packet loss for cooperating sources. Furthermore, the bound can be made as tight as desired simply by changing a single control parameter in the algorithm.

7 Acknowledgments

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

References

[1] L. Benmohamed and S. Meerkov, "Feedback Control of Congestion in Packet Switching Networks: The Case of a Single Congested Node", *IEEE/ACM Transactions on Networking*, Vol. 1, No. 6, pp. 693-708, December 1993.

[2] M. Butto, E. Cavallero, and A. Tonietti, "Effectiveness of the Leaky Bucket Policy Mechanism in ATM Networks", *IEEE Journal on Selected Areas in Communication*, Vol. 9, No. 3, pp. 335-342, April 1991.

[3] I. Cidon, I. Gopal, and R. Guerin, "Bandwidth Management and Congestion Control in plaNET", *IEEE Communications Magazine*, Vol. 30, No. 10, pp. 54-64, October 1991.

[4] A. Demers, S. Keshav, and S. Shenker, "Analysis and Simulation of a Fair Queueing Algorithm", *Proceedings of ACM SIGCOMM*, pp. 1-12, September 1989.

[5] S. Floyd, "Connections with Multiple Congested Gateways in Packet-Switched Networks Part 1: One-way Traffic", *Computer Communication Review*, Vol. 21, No. 5, pp. 30-47, October 1991.

[6] S. Floyd and V. Jacobson, "On Traffic Phase Effects in Packet-Switched Gateways", *Internetworking: Research and Experience*, Vol. 3, No. 3, pp. 115-156, September 1992.

[7] S. Floyd and V. Jacobson, "Random Early Detection Gateways for Congestion Avoidance", *IEEE/ACM Transactions on Networking*, Vol. 1, No. 4, pp. 397-413, August 1993.

[8] W. Grover, "Transport Auction: A New Service Concept", TR*Labs* Project Proposal, 1991.

[9] I. Habib and T. Saadawi, "Controlling Flow and Avoiding Congestion in Broadband Networks", *IEEE Communications Magazine*, Vol. 30, No. 10, pp. 46-53, October 1991.

[10] J. Hayer, "Transport Auction: A New Service Concept", Technical Report TR-93-05, TR*Labs*, November 1993.

[11] D. Hong and T. Suda, "Congestion Control and Prevention in ATM Networks", *IEEE Network Magazine*, Vol. 5, No. 4, pp. 10-15, July 1991.

[12] V. Jacobson, "Congestion Avoidance and Control", *Proceedings of ACM SIGCOMM*, pp. 158-181, August 1988.

[13] R. Jain, "Congestion Control in Computer Networks: Issues and Trends", *IEEE Network Magazine*, Vol. 4, No. 3, pp. 24-30, May 1990.

[14] H. Kanakia, S. Keshav and P. Mishra, "A Comparison of Reactive Host Based Flow Control Schemes in Reservationless Networks", Draft Report, AT & T Bell Laboratories, August 1993.

[15] S. Keshav, "A Control Theoretic Approach to Flow Control", *Proceedings of ACM SIGCOMM*, pp. 3-15, September 1991.

[16] W. Liu, I. Cidon, I. Gopal, and M. Sidi, "Congestion Control Through Input Rate Regulation", *Proceedings of IEEE GLOBECOM*, pp. 1764-1768, November 1989.

[17] P. Mishra and H. Kanakia, "A Hop-by-Hop Rate Based Congestion Control Scheme", *Proceedings of ACM SIGCOMM*, pp. 112-123, August 1992.

[18] A. Parekh and R. Gallager, "A Generalized Processor Sharing Approach to Flow Control in Integrated Services Networks: The Single Node Case", *IEEE/ACM Transactions on Networking*, Vol. 1, No. 3, pp. 344-357, June 1993.

[19] A. Parekh and R. Gallager, "A Generalized Processor Sharing Approach to Flow Control in Integrated Services Networks: The Multiple Node Case", *IEEE/ACM Transactions on Networking*, Vol. 2, No. 2, pp. 137-150, April 1994.

[20] C. Partridge, *Gigabit Networking*, Addison-Wesley, Reading MA, 1994.

[21] N. Prithviraj, "A Virtual Loss-Load Congestion Control Strategy for High Speed Networks", M.Sc. Thesis, Department of Computer Science, University of Saskatchewan, July 1994. Available by anonymous ftp to `ftp.cs.usask.ca` as `pub/discus/thesis-np.ps.Z`

[22] N. Prithviraj and C. Williamson, "A Virtual Loss-Load Congestion Control Strategy for High Speed Networks", Technical Report 95-5, DISCUS Research Group, Department of Computer Science, University of Saskatchewan. Available by anonymous ftp to `ftp.cs.usask.ca` as `pub/discus/paper.95-5.ps.Z`, March 1995. Submitted to *ACM Computer Communication Review*.

[23] K.K. Ramakrishnan and R. Jain, "A Binary Feedback Scheme for Congestion Avoidance in Computer Networks", *ACM Transactions on Computer Systems*, Vol. 8, No. 2, pp. 158-181, 1990.

[24] J. Turner, "New Directions in Communications (or Which Way to the Information Age?)", *IEEE Communications Magazine*, Vol. 24, No. 10, pp. 8-15, October 1986.

[25] C. Williamson and D. Cheriton, "Loss-Load Curves: Support for Rate-based Congestion Control in Datagram Computer Networks", *Proceedings of ACM SIGCOMM*, pp. 17-28, September 1991.

[26] C. Williamson, "Minimizing Round-Trip Times for High-Performance Transport Communication", Technical Report STAN-CS-92-1409 (Ph.D. Thesis), Computer Science Department, Stanford University, October 1991.

[27] C. Williamson, "Optimizing File Transfer Response Time Using the Loss-Load Curve Congestion Control Mechanism", *Proceedings of ACM SIGCOMM*, pp. 117-126, September 1993.

[28] C. Williamson, "Optimizing File Transfer Response Time Using Loss-Load Curves: A Simulation Study", to appear in *Internetworking: Research and Experience*, 1995.

[29] C. Williamson, "Characterizing the Dynamics of the Loss-Load Curve Congestion Control Algorithm", Technical Report 94-3, DISCUS Research Group, Department of Computer Science, University of Saskatchewan. Available by anonymous ftp to `ftp.cs.usask.ca` as `pub/discus/paper.94-3.ps.Z`, February 1994.

[30] G. Woodruff, R. Rogers and P. Richards, "A Congestion Control Framework for High-Speed Integrated Packetized Transport", *Proceedings of IEEE GLOBECOM*, pp. 203-207, November 1988.

Appendix

This appendix presents proofs for Theorems 1, 2, and 3. For the purposes of proofs, assume there is at least one source with $f_i > 0$. Thus $b > 0$. The proofs also assume $k > 0$, and that $f_0 < f < f_c$. For additional details of the proofs, the interested reader is referred to [26].

Proof of Theorem 1 (Bounded Packet Loss)

From the definition of raw throughput, $t(f) = f\,(1 - P(f))$, and $\frac{dt}{df} = 1 - P(f) - f\frac{dP(f)}{df}$. Thus,

$$\frac{dt}{df} = 1 - \frac{f^{k+1} + af^k}{f^{k+1} + b} - f\left[\frac{-af^{2k} + (k+1)bf^k + abkf^{k-1}}{(f^{k+1} + b)^2}\right]$$

$$= \frac{(f^{k+1} + b)^2 - (f^{k+1} + af^k)(f^{k+1} + b) + af^{2k+1} - (k+1)bf^{k+1} - abkf^k}{(f^{k+1} + b)^2}$$

$$= \frac{f^{2k+2} + 2bf^{k+1} + b^2 - f^{2k+2} - af^{2k+1} - bf^{k+1} - abf^k + af^{2k+1} - (k+1)bf^{k+1} - abkf^k}{(f^{k+1} + b)^2}$$

$$= \frac{(2b - b - (k+1)b)f^{k+1} - (ab + abk)f^k + b^2}{(f^{k+1} + b)^2}$$
$$= \frac{-bkf^{k+1} - (k+1)abf^k + b^2}{(f^{k+1}+b)^2}$$
$$= \frac{-b(kf^{k+1} + (k+1)af^k - b)}{(f^{k+1}+b)^2} \tag{2}$$

So $\frac{dt}{df} = 0$ when $b = 0$ or when $kf^{k+1} + (k+1)af^k - b = 0$. But $b > 0$ when any $f_i > 0$, so only the latter case is of interest.

To find the optimal f^*, find the roots of the polynomial $g(f) = kf^{k+1} + (k+1)af^k - b$. (For $k > 1$, this may require numerical methods.) If $a > \frac{b-k}{k+1}$, there must be at least one root in the interval [0,1], since $g(0) = -b < 0$ and $g(1) = k + (k+1)a - b > 0$. There is at most one root since $\frac{dg}{df} = (k+1)kf^k + (k+1)kaf^{k-1} = (k+1)kf^{k-1}(f+a) > 0$. Let this root be f^*.

For the purposes of the proof, we do not need to know f^* itself. Instead, we use only the fact that f^* satisfies

$$f^{k+1} = \frac{b - (k+1)af^k}{k}$$

Plugging the above into Equation 1 gives

$$P(f^*) = \frac{{f^*}^{k+1} + a{f^*}^k}{{f^*}^{k+1} + b}$$
$$= \frac{\frac{b-(k+1)a{f^*}^k}{k} + a{f^*}^k}{\frac{b-(k+1)a{f^*}^k}{k} + b}$$
$$= \frac{b - (k+1)a{f^*}^k + ak{f^*}^k}{b - (k+1)a{f^*}^k + kb}$$
$$= \frac{b - a{f^*}^k}{(k+1)b - (k+1)a{f^*}^k}$$
$$= \frac{1}{k+1}$$

as desired. □.

Proof of Theorem 2 (Convergence)

The proof of Theorem 2 relies on the following two lemmas, which are stated here without proof. Complete details are reported in [26].

Lemma 1 *If each source is transmitting at its own f_i, and the Mth source increases its rate f_M by a small abount $\Delta f_M > 0$ (all other sources remaining the same), then:*

- *if f_M is small, then the loss-load curve $P(f)$ for the $(M+1)$st source increases (i.e., shifts upwards)*
- *if f_M is large, then the loss-load curve $P(f)$ for the $(M+1)$st source decreases (i.e., shifts downwards)*

More precisely, the (lengthy) proof in [26] shows that for each rate f used by the $(M+1)$st source there exists a turning point t_f, $0 \leq t_f \leq 1$, such that $\frac{dP}{df_M} = 0$ at t_f. If $f_M < t_f$, then increasing f_M increases $P(f)$. If $f_M > t_f$, then increasing f_M decreases $P(f)$.

Lemma 2 *The special value t_f is simply f_τ^* for the Mth source, given that source $(M+1)$ is using rate f.*

With these lemmas, it is now possible to prove Theorem 2, as follows:

Suppose some source i is initially sending at a rate $f_i > f_\tau^*$. Then source i can increase its throughput by reducing f_i to f_τ^*. So source i moves to f_τ^* with $P(f_\tau^*) = \frac{1}{k+1}$. But by Lemma 1, this increases the loss-load curve for other sources, so they in turn reduce their rates, which in turn increases the loss-load curve for the ith source. Several rounds of adjustments occur, decreasing in magnitude each time until equilibrium is reached. Convergence is always from above the equilibrium load. $\Box$.

Proof of Theorem 3 (Multi-Hop Paths)

The proof of Theorem 3 is even more complicated. The proof relies on constructing a simple polynomial approximation for the portion of the loss-load curve from the origin to the raw throughput optimum point. For the proposed loss-load formula, the loss-load curve for $f < f_\tau^*$ can be approximated by a polynomial of the form $g(f) = cf^k$. The development of the approximation, and proof of the bound, begins with the following lemma.

Lemma 3 *Given a loss-load curve $P(f)$ with load parameters a and b, and optimum throughput rate f_τ^*, the function $g(f) = (f+a)/(f^{k+1}+b)$ attains its maximum at $f = f_\tau^*$.*

Proof of Lemma 3. Take the derivative of $g(f)$.

$$\frac{dg}{df} = \frac{(f^{k+1}+b)1 - (f+a)(k+1)f^k}{(f^{k+1}+b)^2}$$

$$= \frac{f^{k+1} + b - (k+1)f^{k+1} - (k+1)af^k}{(f^{k+1}+b)^2}$$

$$= \frac{-kf^{k+1} - (k+1)af^k + b}{(f^{k+1}+b)^2}$$

Thus $\frac{dg}{df} = 0$ when $kf^{k+1} + (k+1)af^k - b = 0$. Comparison to the equation derived in the proof of Theorem 1 reveals that this equation is identical to that derived for throughput optimization. Thus $\frac{dg}{df} = 0$ when $f = f_\tau^*$. $\Box$.

This lemma leads directly to the next result:

Lemma 4 *For the loss-load curve $P(f)$ defined in Equation 1, $P(f) \leq \frac{f^k}{(k+1){f_\tau^*}^k}$ for all $0 \leq f \leq 1$.*

Proof of Lemma 4. The relation is clearly true for $f = 0$, since $P(0) = 0$. So now consider $f > 0$. Let $p = P(f)$ be the packet loss value given for rate f on the loss-load curve. Let $h(f) = \frac{f^k}{(k+1){f_\tau^*}^k}$ be the approximation. We prove the result by contradiction. Suppose to the contrary that for some f, $p > h(f)$. Then from the definition of $P(f)$ and $h(f)$ we get:

$$\frac{f^k(f+a)}{f^{k+1}+b} > \frac{f^k}{(k+1){f_\tau^*}^k}$$

Dividing both sides by $f^k > 0$ leaves:

$$\frac{f+a}{f^{k+1}+b} > \frac{1}{(k+1){f_\tau^*}^k} \tag{3}$$

From the result of Theorem 1 we know that

$$P(f_\tau^*) = \frac{{f_\tau^*}^k(f_\tau^*+a)}{{f_\tau^*}^{k+1}+b} = \frac{1}{k+1}$$

Thus,

$$\frac{1}{(k+1){f_\tau^*}^k} = \frac{f_\tau^*+a}{{f_\tau^*}^{k+1}+b}$$

Substituting this into Equation 3 above yields

$$\frac{f+a}{f^{k+1}+b} > \frac{f_\tau^*+a}{{f_\tau^*}^{k+1}+b}$$

which is a contradiction of Lemma 3. Thus our original supposition that $p > h(f)$ for some f must be in error. Hence for all f, the loss-load curve $P(f)$ is upper bounded by $h(f)$. $\square$.

It is now possible to prove Theorem 3 directly, as follows:

By Theorem 1, it is known that the packet loss probability at the bottleneck link is $\frac{1}{k+1}$. It is thus sufficient to show that packet loss at points other than the bottleneck link are negligible.

By the definition of the bottleneck link, the value of f_τ^* for the bottleneck link is lower than the corresponding value for any other switch or link traversed. Thus the sending rate f chosen by the source (based on the bottleneck link) will be explicitly less than the f_τ^* value (which is different for each switch) for any other switch, regardless of whether it is traversed before or after the bottleneck switch. Thus the value $\frac{f}{f_\tau^*}$ is explicitly less than 1.

From Lemma 4, it is known that packet loss at such a switch is *at most* $\frac{1}{k+1}\left(\frac{f}{f_\tau^*}\right)^k$. Clearly, this packet loss probability approaches zero for large k (say, $k \geq 10$). The proof of Theorem 3 is thus complete. $\square$.