A Virtual Loss-Load Congestion Control Strategy for High Speed Networks

Narayanan Prithviraj*
Carey L. Williamson
Department of Computer Science
University of Saskatchewan

Abstract

This paper evaluates a hybrid congestion control strategy called the Virtual Loss-Load model. The approach combines the leaky bucket traffic shaper (a preventive congestion control mechanism) with the loss-load model (a reactive congestion control mechanism). Simulation is used to evaluate the virtual loss-load model, and to compare its performance to that of other reactive congestion control strategies from the literature. The evaluation is done using a benchmark suite of network scenarios proposed by Kanakia, Keshav and Mishra. The performance metrics used in the evaluation are file transfer time and packet loss probability.

The simulation results show that the virtual loss-load model is an effective congestion control strategy, providing performance comparable to several other reactive congestion control strategies. While transient effects due to dynamic changes in cross traffic result in significantly higher packet loss than expected in the virtual loss-load model, the parameters of the virtual loss-load model make it directly tunable to trade off packet loss and file transfer time.

1 Introduction

Congestion in a network occurs whenever the demand on a network resource, such as link bandwidth or buffer space, is exceeded. Congestion left uncontrolled leads to serious consequences, such as long transfer delays due to queueing within the network, and high packet loss due to buffer overflows. As a result, the network will be unable to meet the stringent quality of service (QOS) requirements of emerging network applications. Controlling congestion is therefore essential in broadband networks [3, 4, 6].

Congestion control strategies can be broadly classified as *preventive* or *reactive* [4]. Preventive approaches require an end-to-end allocation of resources before the commencement of data transfer. In reactive schemes, feedback received from the network or the receiver is used to alter a source's sending strategy while a connection is active.

Preventive congestion control strategies may operate at several different time scales. *Admission control* [4] is a preventive congestion control procedure where the network decides, at the time of

*Now with Wipro Infotech Ltd., Bangalore, India.

call arrival, whether to accept or reject a new connection. In making this decision, the network uses the *traffic descriptor* of the connection that characterizes the nature of the traffic generated by the connection [15]. The source must then ensure that the traffic released into the network conforms to the declared traffic descriptor. *Traffic shaping* [1, 2, 5] is a packet[1] level control mechanism used to achieve this.

The *leaky bucket* [18] (or *token bucket* [15]) is one such traffic shaping approach. Packets are buffered at the source and released into the network at a controlled rate. The scheme smoothes out bursts, helping to reduce network congestion. The leaky bucket strategy shapes traffic effectively and is simple to implement. Furthermore, the parameters determining the operation of the leaky bucket are easy to control and can be set based on the traffic descriptor of a connection. However, since the parameters are set statically at the time of call admission, the basic leaky bucket model tends to operate conservatively [11].

Reactive congestion control strategies adjust source transmission behaviour dynamically based on feedback received from the network. Reactive schemes can be *window-based* or *rate-based* [8]. Window-based mechanisms, which specify the maximum number of packets that can be transmitted and outstanding at a time, have been used in traditional protocols, such as TCP. Rate-based approaches, which specify both the number of packets and the time interval of the transmission, are gaining popularity in broadband networks [13], as they tend to be less bursty than window-based schemes. In addition, since some of the broadband applications such as those based on voice and video are inherently rate-based, the adoption of a rate-based scheme is more natural.

Loss-load curves [19] are an example of a rate-based reactive congestion control scheme where switches provide congestion feedback to sources. Sources dynamically adjust their rate of operation based on the feedback information provided by the loss-load curves. The advantage of loss-load curves is that they provide a concise mathematical representation of aggregate load in the network. The feedback information provided by loss-load curves allows sources to react dynamically to changing network conditions. Sources are provided with precise knowledge of the expected loss rate of their traffic within the network, allowing them to control the load they generate on the network. The loss-load congestion control strategy is, however, aggressive in its operation, and at equilibrium generates a load greater than the network capacity, causing excessive packet loss.

The purpose of this paper is to study a "virtual loss-load model" that combines the conservatism of the preventive leaky bucket scheme with the aggressiveness of the reactive loss-load approach. The idea is to exploit the advantages and to mitigate the disadvantages that each of the schemes possesses in isolation. Simulation is used to evaluate the virtual loss-load model on a set of benchmark scenarios proposed by Kanakia, Keshav, and Mishra [9].

The remainder of this paper is organized as follows. Section 2 presents the virtual loss-load model, including its two key components: the leaky bucket traffic shaper and the original loss-load model. Section 3 describes the experimental methodology used to simulate and evaluate the performance of the virtual loss-load model, and lists the performance metrics used in the evaluation. Section 4 presents the simulation results for two sample network scenarios. Finally, Section 5 concludes the paper.

[1] Traffic shaping is usually described in terms of *cells*. By convention, the term cell means small fixed-size packets. Throughout the rest of this paper, the more general term *packet* is used since the virtual loss-load approach is applicable to both packet-based and cell-based networks.

2 The Virtual Loss-Load Model

The virtual loss-load model proposed in this paper combines the leaky bucket traffic shaper and the loss-load model together, along with some modifications to provide a hybrid congestion control strategy. This section examines the leaky bucket and loss-load models in greater detail.

2.1 Leaky Bucket Model

The leaky bucket model is a traffic shaping approach implemented at the network edges. Sources release packets at a controlled rate into the network, thereby smoothing out bursts.

Figure 1 depicts the operation of a buffered leaky bucket traffic shaper. Arriving packets enter a queue in the traffic shaper. If the queue is full, the arriving packet is dropped. For a packet at the head of the queue to enter the network, the packet must first obtain a token from a token pool. Tokens are generated at a fixed deterministic rate and enter this pool. If the token pool is full, newly generated tokens are discarded.

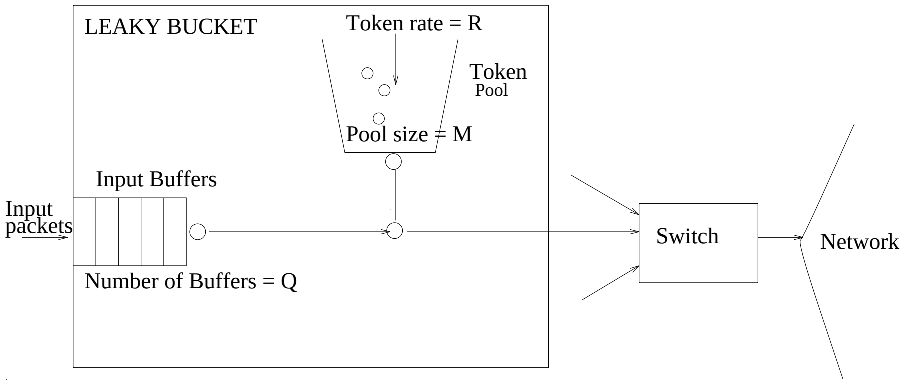


Figure 1: A Buffered Leaky Bucket Traffic Shaper

Several parameters govern the operation of the leaky bucket. The size of the token pool M limits the maximum number of back-to-back packets that can enter the network. The token rate R determines the mean rate at which packets enter the network. Typically, this rate is set to the rate allocated to the connection at the time of call admission. The size of the queue Q represents a tradeoff between packet delay and packet loss. Increasing the queue size decreases the probability of packets being dropped at the shaper. However, the resulting longer queues imply that packets must wait for longer periods before being transmitted.

Earlier work by Liu [11] has shown the following results for the leaky bucket traffic shaper:

- The shaper has been shown to result in very good network-level performance. Traffic flows entering the network after passing through the shaper are smooth, and experience little or no queueing delay at network switches, even when the network load is high.

- Good network-level performance, however, comes at the expense of extra queueing delay and packet loss at the shaper. There is a definite tradeoff to be made between user-level performance and network-level performance, in terms of packet loss and delay.

- The leaky bucket traffic shaper is conservative in its operation, and does not exploit available capacity when the network load changes dynamically.

More recent work by Parekh [14] has established formal end-to-end delay bounds for packet traffic sent using a leaky bucket traffic shaper, providing that the network switches support a generalized processor sharing service discipline.

While the leaky bucket traffic shaper has definite performance advantages, it also possesses the disadvantage of operating at a fixed token generation rate, thereby preventing a connection from possibly exploiting increased available capacity in the network. To overcome this problem, a *virtual leaky bucket* traffic shaper has been proposed [6]. In a virtual leaky bucket, packets arriving when the token pool is empty are tagged *red* prior to being sent into the network, while other packets are marked *green* before being dispatched. Red packets are considered violators of available bandwidth as their transmission implies operation at a rate higher than that allocated. However, because of the statistical nature of arriving traffic, the network may have sufficient capacity to carry the red packets, provided other connections are not adversely affected. If congestion is detected somewhere in the path of the connection, the red packets are discarded so as not to affect the transmission of the green packets. Packet tagging allows sources to operate at higher rates, but provides no guarantees for the delivery of (or number of) red packets in the network.

In summary, the leaky bucket model offers the following advantages:

- *Simplicity* : The leaky bucket model shapes traffic effectively and is simple to implement.
- *Controllability*: The parameters determining the operation of the leaky bucket are easy to control and can be set based on the traffic descriptor of the connection.

The model, however, has certain drawbacks:

- *Conservative Operation*: As the leaky bucket traffic shaper parameters are set statically, it may not be possible to exploit available network bandwidth when network conditions vary dynamically.
- *Limited Sender Control*: Although the virtual leaky bucket traffic shaper does allow more packets into the network, there is no information about the expected loss rate of red packets released into the network. Thus the traffic source has limited knowledge about how to control its overall level of packet loss.

2.2 The Loss-Load Model

The loss-load model is a reactive congestion control mechanism that allocates bandwidth dynamically depending on changing network conditions. Switches monitor network load and provide explicit loss-load feedback information to sources. A loss-load curve exists for each switch's outgoing link. Each loss-load curve is based on the total load on an outgoing link and the capacity of that link, and, for each source, expresses the packet loss probability (i.e., the probability that a packet transmitted by that source is dropped by the network) as a function of the source's offered load.[2]

[2]The exact formula used to compute the loss probability is given later in this section.

Sources select the rate of operation depending on the level of packet loss that they are willing to tolerate.

Switches monitor each source's rate of operation. If the load on an outgoing link is less than the link's capacity, no packets are dropped at the switch. If the capacity of a link is exceeded, a switch drops the excess packets by associating a packet loss probability p_i with each source i using the link. By dropping excess packets, switches restrict the accepted traffic load on a link to operate at or below the link's capacity.

Figure 2 depicts an example of a loss-load curve. The packet loss probability p $(0 \leq p \leq 1)$ is a non-decreasing function of the source's transmission rate f $(0 \leq f \leq 1)$, where f is expressed as a fraction of the switch's output link capacity. The packet loss probability reaches 1.0 for high values of f to protect the network from misbehaving (greedy) sources [19].

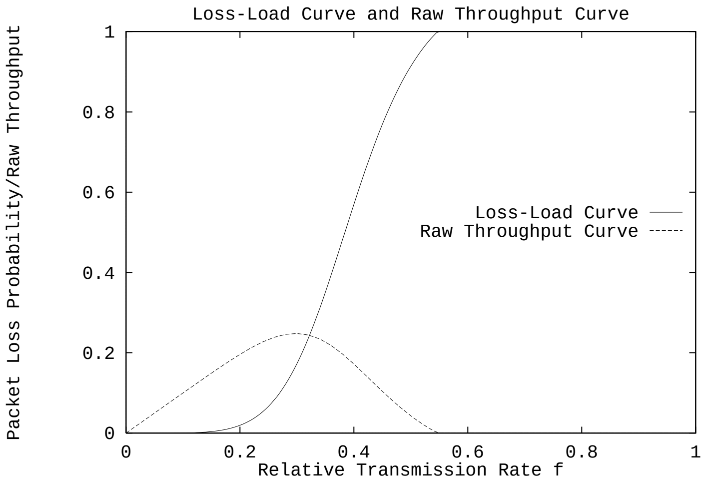


Figure 2: An Example of a Loss-Load Curve for a Source ($k = 5$)

Associated with each loss-load curve is a raw throughput curve that measures the raw throughput (i.e., the rate of successful packet delivery) attainable by a source for a given rate of operation. The raw throughput curve is directly computable from the loss-load curve using the relation $t = (1 - p)f$.

The loss-load curve for a source is always expressed in terms of the aggregate load offered by *other* sources in the network. If M sources share a single outgoing link at a switch, the switch computes the following for the jth source:

- f_i, the rate of operation of the ith source, for $i = 1, 2, \ldots M$
- $f_{total} = \sum_{i=1, i \neq j}^{M} f_i$, the aggregate ambient load on the link of capacity C,
- $a = f_{total} - C$, the excess load on the link generated by other sources,
- $b = \sum_{i=1, i \neq j}^{M} f_i^{k+1}$, a measure of the distribution of the load amongst the other sources.

The computed values of a and b are sent back to the jth source by the switch. The jth source uses the values of a and b to compute the packet loss probability at a load f using the formula [19]:

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

Note that the aggregate loss-load information is represented using only two floating point numbers a and b, which are computed at each switch and fed back to sources. Sources compute the entire loss-load curve $P(f)$ based on the values of a, b, and a constant integer parameter k, which is known globally by all sources and switches in the network.

The parameter k is an important parameter in the loss-load formula as it determines how severely sources with high transmission rates are punished relative to slower sources. The larger the value of k, the greater is the packet loss probability for greedy sources. Suggested values for k in the original loss-load model range from $k = 1$ to $k = 20$, with $k = 10$ being a reasonably good choice [19, 20, 21]. The choice of a reasonable value for k in the virtual loss-load model is discussed in Section 4.

Once provided with the loss-load curve, sources can select their rate of operation based on one of two possible strategies [19, 20]:

- **Raw Throughput Optimization**:

  Sources may base their rate of operation on the raw throughput curve. Raw throughput is maximized at a rate f_t, obtained by solving for f in the equation $\frac{dt}{df} = 0$.

  When all sources choose to maximize raw throughput, the loss-load model has been shown to have two important properties [19]:

  - *Bounded Packet Loss*: When a source maximizes raw throughput, the packet loss probability for a source at f_t is $\frac{1}{k+1}$. This result is important as it provides a bounded and predictable packet loss probability for a source at f_t.
  - *Convergence*: If M identical sources sharing a single bottleneck link of capacity C all maximize raw throughput, the network converges to a stable equilibrium operating point, with each source operating at its individual f_t. The total offered load in the network is $\frac{k+1}{k}C$. The property of convergence is important as it shows that the loss-load model is stable. The equilibrium reached is independent of a source's initial rate and feedback delay. Smaller values of k provide faster convergence than larger values. However, since the total offered load at equilibrium is $\frac{k+1}{k}C$, larger values of k result in lower overload.

- **File Transfer Response Time Optimization**:

  Instead of operating at a maximum raw throughput rate, sources may wish to operate at a rate that minimizes the total time to complete a file transfer connection. The expected response time for a transfer of N packets when the round trip time is R is given by [20]:

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

  where p is the packet loss probability at rate f.

To minimize response time, a source solves for f in the equation $\frac{d(Response(f))}{df} = 0$. The minimum response time is obtained at a rate f_r. For $R \to 0$, or $N \to \infty$, f_r approaches f_t.

In summary, the loss-load model offers the following advantages:

- *Concise Representation*: The loss-load model provides a concise mathematical representation of aggregate load in the network, as opposed to schemes that just indicate congestion in the network without numeric quantification.
- *Sender Control*: Since the loss-load model captures more information about load conditions in the network, sources have precise knowledge of the expected loss rate at different rates of operation. Sources thus have control in determining the rate at which they need to operate.

The model, however, has the drawback that it operates aggressively. When all sources maximize raw throughput, the overall load offered to the network can be much higher than the capacity of the network, depending on the value of k. The packet loss probability is, in many cases, far too high (e.g., 9.1% for $k = 10$) to support the QOS requirements of most high speed network applications.

2.3 Virtual Loss-Load Model Description

Sources in the virtual loss-load model combine the leaky bucket traffic shaper with the loss-load model to create an augmented leaky bucket traffic shaper (see Figure 3).

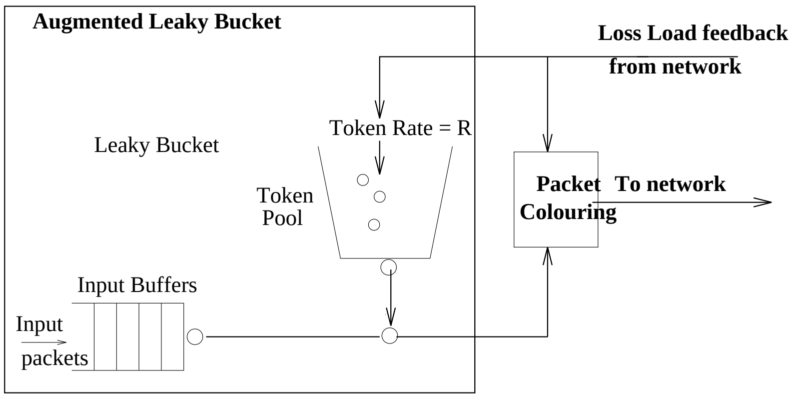


Figure 3: Augmented Leaky Bucket Shaper in the Virtual Loss-Load Model

The leaky bucket shaper used in the virtual loss-load model contains two major changes. First, the token generation rate is not maintained at a fixed rate as is the case in the basic and virtual leaky bucket traffic shapers. Similar to other rate-based congestion control strategies [13, 17], sources can adjust the token generation rate (and hence the packet release rate) based on the feedback that they receive from the network. In particular, sources make use of the feedback information provided by the loss-load curve. Sources can use either raw throughput optimization or response time optimization to adjust the token rate. Second, the packet colouring strategy employed in the virtual loss-load model differs from that used in the virtual leaky bucket shaper. Sources colour a fraction of the packets departing from the leaky bucket. The fraction of packets coloured is chosen so as to match the packet loss probability associated with the current rate of packet release, as calculated from the loss-load feedback. The result is a bounded percentage of red packets in the network.

The switches in the virtual loss-load model also contain several modifications from those in the original loss-load model. First, the switches associate a *virtual capacity* C_v with each outgoing link connected to the switch. If the actual capacity of a link is C, then the virtual capacity $C_v = vC$ where v $(0 \leq v \leq 1)$ is a parameter called the virtual factor. All loss-load feedback computations made by a switch are based on C_v rather than C, so that sources perceive operation over links of virtual capacity. Switches, however, discard packets from sources based on the *actual* capacity C of a link, similar to the loss-load model.[3] Second, switches give higher priority to the transmission of green packets than red packets. Whenever a packet must be discarded at a switch, the switch discards the arriving packet if it is red. If the arriving packet is green, the switch scans its buffers looking to drop a red packet belonging to that source. If no red packet is found, then the arriving green packet is discarded.

2.4 Advantages of the Virtual Loss-Load Model

Combining the leaky bucket shaper with the original loss-load model offers the following advantages:

- *Simplicity and Controllability*: The leaky bucket shaper advantages of simplicity and controllability are preserved in the virtual loss-load model. The virtual factor v is an additional controlling parameter in the virtual loss-load model. By choosing appropriate values of v (see Section 4), the aggressiveness of the congestion control strategy can also be controlled.

- *Higher Network Utilization*: Since the token generation rate of the leaky bucket shaper can be modified using loss-load feedback, the conservative nature of the basic leaky bucket shaper is overcome in the virtual loss-load model. The virtual loss-load model can therefore exploit available network capacity during dynamic network conditions, increasing network utilization.

- *Lower Network Overload*: There are two parameters that control network load at equilibrium in the virtual loss-load model. By extending a result of the loss-load model, the expected load in the virtual loss-load model when all sources maximize raw throughput is $\frac{k+1}{k}C_v = \frac{k+1}{k}vC$. The choice of k in the original loss-load model determines how aggressive the model is (i.e., by how much the advertised network capacity is exceeded). For a given value of k, v can be set to reduce or nullify network overload. The virtual loss-load model thus provides an additional parameter that can help reduce network overload, while preserving the convergence property of the original loss-load model.

The remainder of this paper presents an evaluation of the performance of the virtual loss-load model, using simulation.

[3] Note that this scheme requires source cooperation in order to work properly, unlike the original loss-load model. In the virtual loss-load model, for example, non-cooperating greedy sources that are aware of the network 'lying' about its capacity may choose to operate at a higher rate to exploit available capacity, thereby obtaining more than their fair share of the network bandwidth.

3 Experimental Methodology

Simulation experiments were conducted to evaluate the performance of the virtual loss-load model. This section describes the experimental methodology used for the simulation experiments.

3.1 Simulation Scenarios

The simulation experiments for the virtual loss-load model were primarily based on a benchmark suite developed by Kanakia, Keshav and Mishra [9]. The scenarios forming that benchmark suite are described as being representative of typical high delay-bandwidth product networks, and attempt to capture a variety of network conditions.

Kanakia *et al.* propose this benchmark suite as a standardized test suite for the evaluation of reactive congestion control strategies. The experiments conducted in this research use the same benchmark scenarios, the same workload models, and the same assumptions as in the proposed benchmark suite. This approach facilitates comparison of the virtual loss-load congestion control strategy with the other congestion control strategies evaluated in [9].

3.2 Workload Model

An important factor determining the performance of a congestion control algorithm is the workload used. In this work, as in Kanakia, Keshav, and Mishra [9], the workload at each source is assumed to be a finite file transfer application. According to [9], the file transfer application is a very general model of a network application: applications as diverse as network news, database access and shared whiteboards all employ the file transfer paradigm, which is the bulk transfer of data at the fastest rate possible. The study [9] argues that a realistic evaluation of a congestion control scheme can be made only on the basis of finite file transfers, unlike many existing studies that assume infinite data sources, so as to study performance at stability. Also, according to the study, as networks become faster, file transfer durations will decrease, so that it is the transient network conditions that are most relevant in studying the performance of a congestion control scheme.

3.3 Performance Metrics

In this paper, two performance metrics are used to evaluate the virtual loss-load model. Performance metrics are measured for a specific connection in the network, called the *control connection.* The metrics are file transfer time and packet loss (drop) probability.

- *File Transfer Time* is the performance metric used in this study to measure network throughput (goodput). File transfer time is defined as the time from when the first byte of data is available at the source for transmission, to the time when the source receives the acknowledgment that the destination received the last byte of the file. Sources retransmit lost packets as many times as necessary until all packets (bytes) of the file are successfully delivered. This metric is thus a sum of the time taken to transmit all the bytes of the file, plus the time taken to retransmit lost packets.

  The file transfer time is the primary metric used by Kanakia *et al.* [9]. It directly measures the effective throughput achieved by a congestion control scheme: the lower the file transfer time, the higher the effective network throughput, and vice versa. According to [9], the file transfer time allows for the simultaneous representation of several other metrics, such as bottleneck utilization, fairness, and packet loss probability.

- *Packet Loss Probability (P.L.P.)* measures the fraction of packets lost (i.e., dropped) by a connection. That is, it is the ratio of the total number of dropped packets to the total number of transmitted packets, which, by definition, includes all the original transmitted packets, plus any and all retransmissions of dropped packets. Clearly, low values of this performance metric are desirable.

  Although the P.L.P. metric is not used in [9], it is used in this paper for comparing the performance of the virtual loss-load model with the original loss-load model. The metric is also useful in assessing the efficiency of file transfers having comparable file transfer times, but different P.L.P.'s.

3.4 Experimental Design

The simulation experiments were conducted using a two factor experimental design, with the two factors being k and v. For all experiments, the factor v was varied from 0.5 to 1.0 in steps of 0.1. For each value of v, values of k at 2, 5, 10 and 20 were used, similar to [19]. For each value of k and v, the performance metrics described in Section 3.3 were used to assess how well the virtual loss-load model performed. The simulation results follow in the next section.

4 Simulation Results

This section presents simulation results for two selected network scenarios. The complete simulation results on all simulation scenarios tested are available in [16].

4.1 Scenario 0: Two Sources

In order to evaluate the effectiveness of the virtual loss-load model, a preliminary experiment was conducted to study a simple two-sender case as shown in Figure 4. This scenario does not appear in Kanakia *et al.* [9], but does provide a useful basis for understanding the virtual loss-load model.

In this scenario, two hosts H1 and H2 share a bottleneck link of capacity 40 Mbps and transmit to a host H3. Each source transmits over a link of capacity 200 Mbps. Transmitted packets are of a fixed size of 500 bytes. Both sources start at time 0 and transmit 2 Mbytes of data. The leaky bucket traffic shaper in the virtual loss-load model operates with a buffer size of 10, and a token pool size of 5. Switches use a buffer size of 500 packets. The value of C is normalized to a capacity value of 1, and all links have a propagation delay of 1 ms. The round trip time (RTT) for both sources is 4 ms. All performance metrics are measured for the control connection, Connection 1. Both sources transmit at a rate to *maximize raw throughput*, and colour the packets randomly based on the loss probability calculated from loss-load feedback.

The objective behind this experiment was to study the operation of the virtual loss-load model under a simple network scenario. Two main advantages are gained from this study. First, it provides a validation mechanism for the virtual loss-load model by allowing a comparison between

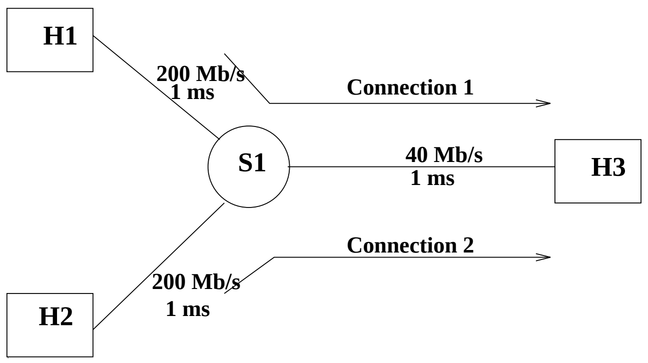


Figure 4: Network Configuration for Scenario 0 (Two Sources)

the theoretical predicted performance of the virtual loss-load model with those obtained through simulation results. Second, it provides a reference point for comparison with the more complicated benchmark scenarios proposed by Kanakia *et al.*

The simulation experiments use two factors k and v. As both sources use the raw throughput model, the expected load at equilibrium is $\frac{k+1}{k}C_v = \frac{k+1}{k}vC$. When $v = \frac{k}{k+1}$, the load at equilibrium should equal 1, so that sources maximize network utilization without causing network overload. At this value of v, therefore, sources should expect the best possible performance. The following two subsections describe the experimental results obtained.

4.1.1 File Transfer Time

The file transfer time provides a measure of network goodput. The theoretical optimum transfer time for each connection is 800 ms, assuming both connections share the bottleneck link equally. The purpose of this experiment was to determine whether the file transfer time was minimized at $v = \frac{k}{k+1}$.

Figure 5(a) shows how the file transfer time changes with values of v for Connection 1 for different values of k. For small values of v, the file transfer time is large. This result is expected since at small values of v, the entire network capacity is not exploited. As the value of v increases, the file transfer time drops as the virtual capacity of the bottleneck link approaches the actual capacity. From the graph, it may be seen that the optimum file transfer time can be achieved for any value of k, but the value of v at which this occurs, namely at $v = \frac{k}{k+1}$, depends on k. It can be seen that file transfer time is indeed minimized at a value of v close to $\frac{k}{k+1}$. Values of $v > \frac{k}{k+1}$ result in increased overload, more packet loss and retransmissions, and no improvement in the file transfer time.

4.1.2 Packet Loss Probability

The P.L.P. measures the overall packet loss for a connection, and also provides an indication of network overload. Ideally, the P.L.P. for a connection should be close to 0. Again, theoretically, at a value of $v = \frac{k}{k+1}$, there should be no overload since the equilibrium load predicted for this

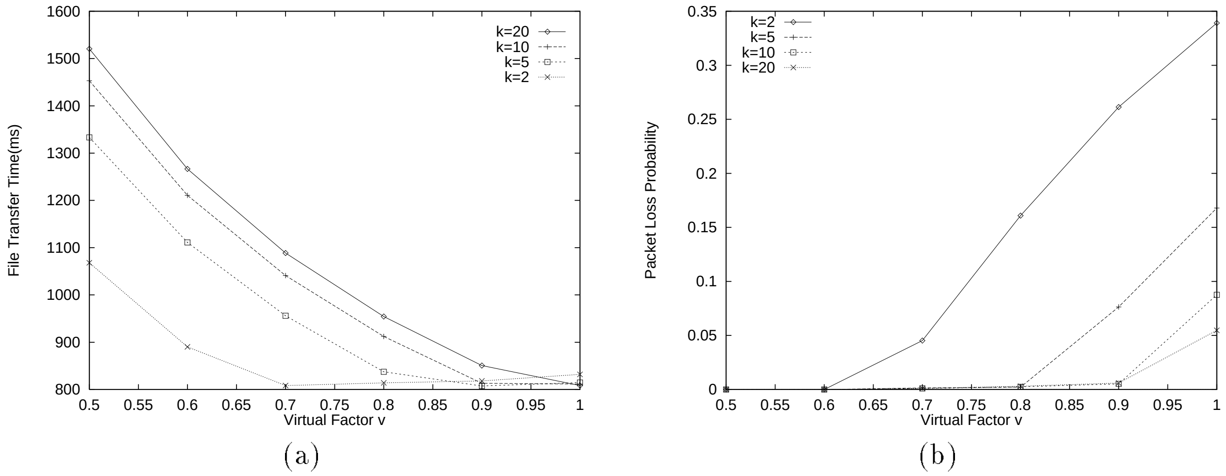


Figure 5: Simulation Results for Connection 1 in Scenario 0 (Two Sources): (a) File Transfer Time; (b) Packet Loss Probability

value of v is 1. The P.L.P. is expected to increase beyond this value of v, and the purpose of the experiment was to determine whether this indeed was the case.

Figure 5(b) depicts the packet loss probability for Connection 1 for different values of k and v. The simulation results match those predicted by theory. As in the earlier cases, the P.L.P. is negligible for values of $v \leq \frac{k}{k+1}$, since network overload is negligible. Above this value, however, overload in the network results in increased packet loss. A value of $v = 1$ implies that the virtual loss-load model becomes equivalent to the original loss-load model. The resulting loss at this point matches closely with that predicted theoretically (i.e., $\frac{1}{k+1}$), yielding confidence in the simulation results obtained.

4.1.3 Summary

Experiments with the two sender case show the importance of both controlling parameters k and v. The parameter k determines the aggressiveness of the virtual loss-load model (i.e., how much 'overshoot' is generated at equilibrium). The virtual factor v sets the 'speed limit' of this overshoot.[4] By choosing both parameters appropriately, it is possible to obtain a controllable and predictable model that optimizes the performance metrics.

It is to be noted that these results hold only for this simple experiment with stable operating conditions and small feedback delays. The next simulation experiment studies the virtual loss-load model in a more dynamic network scenario with transient network conditions and large feedback delays, using a benchmark test proposed by Kanakia *et al.* [9].

[4] A simple highway analogy is useful here. Since many drivers on a freeway drive 10 mph faster than the posted speed limit, a clever police officer can achieve an effective maximum speed of 60 mph by posting a speed limit of 50 mph. In the context of the virtual loss-load model, the parameter v is the posted speed limit, and the parameter k determines by how much the 'speeders' exceed the speed limit.

4.2 Scenario 1: Slow Changes in Cross Traffic

This scenario intends to measure the performance of the congestion control strategy in response to a gradual increase and decrease of traffic in the network. The configuration used for this experiment is shown in Figure 6.

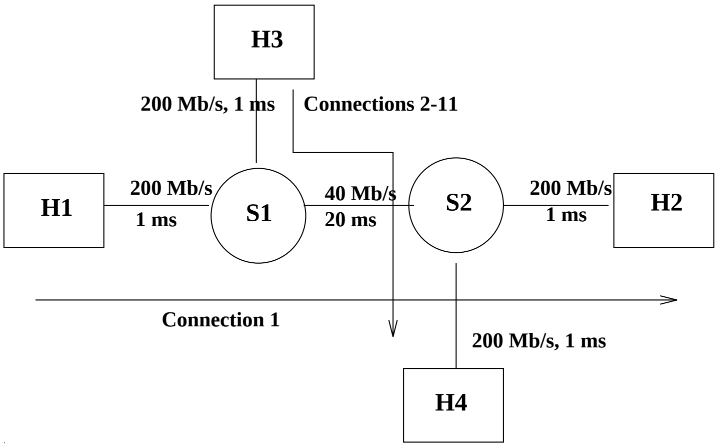


Figure 6: Network Configuration for Scenario 1 (Slow Changes in Cross Traffic)

In this scenario, the cross traffic load seen by the control connection (Connection 1) changes gradually. Connection 1 starts up at time 0 and has 2 Mbytes of data to send. Connection 2 commences after 100 ms. The remaining cross traffic connections, Connection 3 through Connection 11, begin at intervals of 44 ms (the round trip time for Connection 1) following Connection 2. Each of the ten cross traffic streams sends 500 Kbytes of data. All sources use *response time optimization* for choosing transmission rates.

Note that the round trip time for the control connection is 44 ms. For fairness of comparison to other reactive congestion control policies (see Table 1), the virtual loss-load model uses end-to-end feedback from the destination to the source with this full round trip delay. Load information gleaned from the switches on the forward (data) path is sent back to the source on the reverse (acknowledgement) path by the destination.[5]

4.2.1 Simulation Dynamics

Before studying the performance of the virtual loss-load model for this scenario, it is instructive to study the dynamics of the virtual loss-load model in this scenario, and to understand how the values of k and v affect performance. Figure 7 shows the aggregate offered load arriving at switch S1 (destined for the 40 Mbps output link of switch S1) as a function of time for k = 2, 5, 10, and 20. The virtual factor v is fixed at a value of 1.0, so that the performance of the virtual loss-load model essentially mimics that of the original loss-load model.

As may be seen from the four graphs, the value of k affects the dynamics, the responsiveness, and the level of the overload in the virtual loss-load model. In general, the value of k determines

[5] Note that this assumption differs from that in the original loss-load model [19, 20], which assumes direct feedback from switches to the sources.

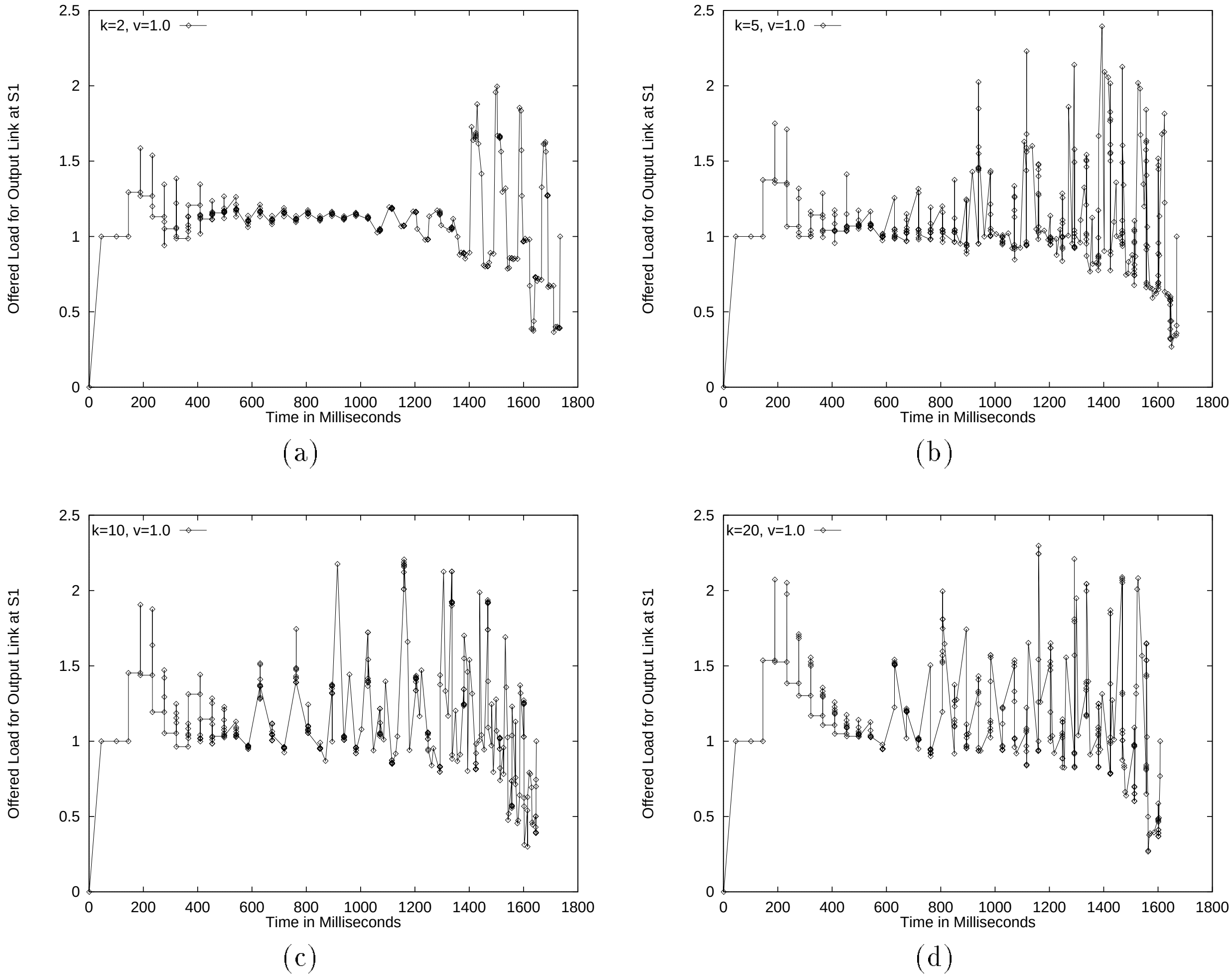


Figure 7: Traffic Load at Switch S1 for Different Values of k, with $v = 1.0$ in Scenario 1 (Slow Changes in Cross Traffic): (a) $k = 2$; (b) $k = 5$; (c) $k = 10$; (d) $k = 20$

the level of overshoot produced over the "speed limit" set by the value of v (which is 1.0 in these four graphs). The effect of decreasing the value of v (not shown in these graphs) would be to shift the entire graph vertically downward, so that the load level can be maintained closer to capacity.

In each of the four figures, there are three regions of interest:

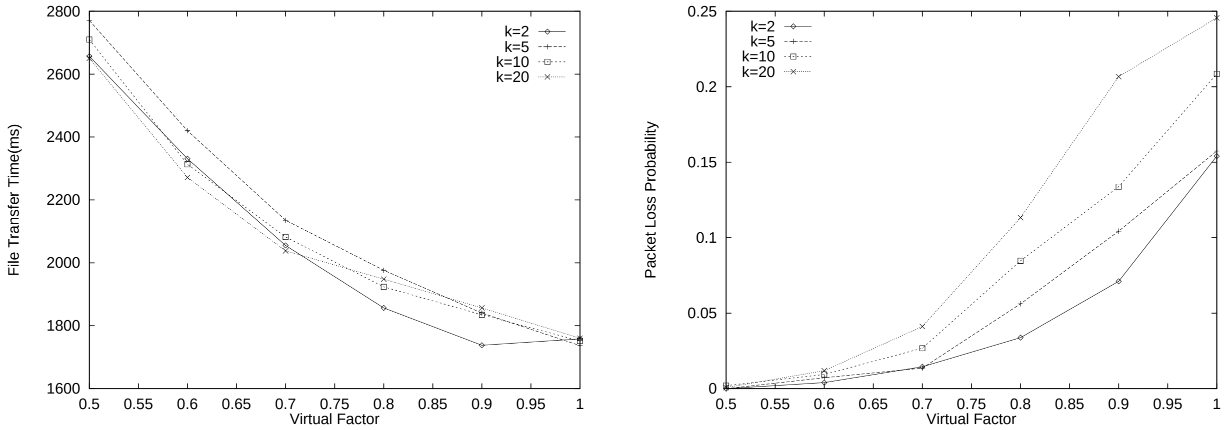
- In the first region, approximately from 0 ms to 500 ms, connections start to arrive at the switch. It can be seen that when Connection 1 begins, the load at the output link of the switch rises to 1.0, indicating that the entire bandwidth of the link is made available to that connection. As further connections arrive, the load starts to fluctuate as different connections compete for the bandwidth and adjust their load based on the response time model. Each new arrival results in a sudden increase (the vertical spike) in the offered load at S1, followed by a decrease in the load as sources react to the new load. The magnitude of the spikes and the responsiveness of the sources depends on k. For a value of $k = 2$, for example, the load rises to a maximum of about 1.58. For $k = 20$, the load rises to as much as 2.07.

- In the second region, approximately from 500 ms to 1000 ms, connections attempt to reach network equilibrium. The equilibrium achieved should always be greater than 1.0 because of the aggressiveness of the loss-load model, and always less than $\frac{k+1}{k}$ because of the more conservative nature of the response time model [20]. It is seen from the four graphs, however, that true equilibrium is never attained, and there is always some oscillation. Larger values of k take more time to reach stability, and the oscillation is much worse. A value of $k = 2$ provides the most stable operating conditions, as convergence to network equilibrium has been shown to be fastest for small values of k [19]. The overall load for this value of k varies from about 1.11 to 1.15. If the raw throughput model had been employed instead of the response time model, the expected load in this region for a value of $k = 2$ would have been $\frac{k+1}{k} = 1.5$.

- In the third region, spanning 1000 ms to the end of the simulation, connections start departing. As each connection departs, the other connections obtain the information about increased availability of network bandwidth, and try to operate at a higher rate. This causes a period of overload and oscillation once again as rate adjustments occur. Larger values of k cause much greater overload and oscillation.

Larger values of k are clearly undesirable in all three regions. Large values of k are less responsive to dynamic load, and much more susceptible to large transient effects and oscillations, especially when there are large feedback delays in the network. In all three regions, lower values of k perform better in controlling the amount of overload produced. In other words, lower values of k have much lower packet loss rates. Note that this is the *exact opposite* of the behaviour in Scenario 0. It thus appears that lower values of k are more suitable in networks where network load conditions are dynamic.

There are several reasons for this unexpected behaviour. First, and most important, is the large end-to-end feedback delays in this network scenario. Large feedback delays cause a large difference between the loss rate expected by a source, and the actual packet loss rate for a source at the switch. This difference is worsened in the presence of dynamic network conditions, where network conditions change within short intervals of time. Second, the convergence and stability properties of



(a) (b)

Figure 8: Simulation Results for Connection 1 in Scenario 1 (Slow Changes in Cross Traffic): (a) File Transfer Time; (b) Packet Loss Probability

the loss-load model depend upon at least one source maximizing raw throughput. In this scenario, all sources use the response time model in selecting their rate of operation, as opposed to the raw throughput model. Thus convergence is no longer guaranteed to hold. Finally, the number of sources in this scenario is quite small. The dynamic behaviour of the loss-load algorithm is always at its most extreme when the number of sources is small [21].

The following two subsections present the file transfer time and packet loss probability simulation results obtained for Scenario 1.

4.2.2 File Transfer Time

Figure 8(a) shows the file transfer time for Connection 1 for various values of k and v for Scenario 1. The theoretically optimal file transfer time for Connection 1 is 1444 ms.

The file transfer time decreases with increasing values of v. For example, at k =2, as v increases from 0.5 to 0.6, the file transfer time falls from 2656 ms to 2330 ms. This decline is understandable, since more bandwidth is made available to connections as v increases. However, the decline in the file transfer time levels off with larger v. This occurs as the connections use all available bandwidth, and with increased rates of operation, no further gains are possible since the network has already been exploited to capacity. In fact, with increased values of v, the file transfer time performance actually worsens for $k = 2$ (the file transfer time in this case increases from 1737 ms at $v = 0.9$ to 1758 ms at $v = 1.0$), since packet loss worsens. It can also be seen that the best file transfer time no longer occurs at a value of $v = \frac{k}{k+1}$.

Clearly, the file transfer time depends more strongly on v than it does on k. Large settings of v (e.g., $v = 0.8$ to $v = 1.0$) provide reasonable file transfer times for all the k values considered in this simulation configuration.

4.2.3 Packet Loss Probability

Figure 8(b) shows the overall packet loss probability for Connection 1 for various values of k and v for Scenario 1. As expected, operating at low values of v results in lower packet loss than at larger values of v. Also, for a given value of v, lower values of k result in lower packet loss. For example, at $v = 0.9$, the packet loss probability is 0.10 for $k = 5$, but is 0.20 for $k = 20$. It may be noted that the packet loss probability at $v = 1.0$, for a given value of k, differs considerably from the value of $\frac{1}{k+1}$ predicted by the raw throughput model at $v = 1.0$.

The response time model used in the current scenario, although different from the raw throughput model, should also yield P.L.P.'s that are inversely proportional to k. However, the exact opposite behaviour is observed in the current scenario. For example, at $v = 1.0$ for $k = 2$, the P.L.P. is 0.15, while for $k = 20$, it is 0.25. The main reason for the discrepancy is that the value of $\frac{1}{k+1}$ holds true only when the system reaches equilibrium. As was seen earlier, the period of equilibrium is small compared to the transient periods for the current scenario. It is these transient periods that dominate the packet losses encountered by a connection, and the transient periods are much worse for large values of k.

4.2.4 Comparison to Other Strategies

Kanakia *et al.* [9] present simulation results for five different reactive congestion control schemes. These schemes are: the TCP slow-start scheme (tcp) [7], the DECbit scheme [17], the Packet Pair scheme (pp) [10], the hop-by-hop congestion control scheme (hbh) [12], and the explicit end-to-end feedback scheme (e2e-ef) [12]. Slow-start is the prevalent window-based congestion control strategy used in TCP. It uses packet loss within the network as an implicit signal of congestion, and dynamically adjusts a source's congestion window based on packet drops experienced. DECbit is a binary feedback congestion control strategy that allows switches to indicate the presence or absence of congestion along a network path. This information is fed back to traffic sources so that they can dynamically adjust their transmission rates or flow control window size. Packet pair is a rate-based control strategy based on using pairs of "probe" packets to estimate the packet processing rate of the bottleneck switch in the network. This scheme is dependent on a round robin service discipline at network switches, while the other schemes assume only FIFO (First-In-First-Out) service. The hop-by-hop and end-to-end explicit feedback schemes use dynamic adjustment of sender rates to control buffer occupancy at network switches, either on a per-hop or end-to-end basis.

The performance results for these five strategies on Scenario 1 are summarized in Table 1. Performance results for the virtual loss-load model on Scenario 1 are summarized in Table 2.

The results show that the virtual loss-load model provides file transfer time performance comparable to several existing reactive congestion control algorithms. On Scenario 1, the virtual loss-load model is far superior to the tcp and decbit scheme, but is not nearly as good as the pp, hbh, and e2e-ef schemes. Reducing the value of the virtual factor v improves the performance of the virtual loss-load model slightly, reducing both the packet loss probability and the file transfer time.

Similar results are obtained from a performance comparison of all strategies on a fuller set of benchmark scenarios [16].

Table 1: Performance for Scenario 1 (Slow Changes in Cross Traffic) (from [7])

Scheme	Conn 1 Transfer Time (ms)	Mean Queueing Delay (ms)	Std. Dev.
Optimal	1444	0	0
hbh	1534	0.99	0.41
pp	1569	0.31	0.17
e2e-ef	1621	0.85	0.42
tcp	4430	na	na
decbit	5566	na	na

Table 2: Virtual Loss-Load Model Performance for Scenario 1 (Slow Changes in Cross Traffic)

Scheme	Conn 1 Transfer Time (ms)	Mean Queueing Delay (ms)	Std. Dev.	Overall P.L.P.
$k = 2$, $v = 1.0$	1758	0.41	0.09	0.15
$k = 2$, $v = 0.9$	1737	0.45	0.48	0.07

4.3 Summary

The simulation results show that the virtual loss-load model is an effective congestion control strategy, providing performance comparable to several other reactive congestion control strategies. While transient effects due to dynamic changes in cross traffic result in significantly higher packet loss than expected in the virtual loss-load model, the parameters of the virtual loss-load model make it directly tunable to trade off packet loss and file transfer time.

5 Concluding Remarks

Based on the experiments that were conducted, the following conclusions may be drawn:

- *Effect of factors in the virtual loss-load model*: Low values of k provided the best overall performance for the benchmark scenarios considered in this research. Larger values of v improve file transfer time, but do so at the expense of increased packet loss and delay.

- *Comparison with the loss-load model*: The virtual loss-load model improves on the original loss-load model by substantially reducing packet loss. The virtual loss-load model differs from the original loss-load model primarily through the incorporation of the virtual factor v. By setting v to small values, packet loss can be decreased considerably.

- *Comparison with other congestion control strategies*: The performance of both the original loss-load model and the virtual loss-load model is comparable to that of several other reactive congestion control strategies.

The experiments that were conducted show the importance of characterizing and understanding the dynamics of congestion control strategies in transient network conditions, with dynamic cross traffic flows and large feedback delays. Although these conditions are aimed specifically at modeling future broadband networks, transient traffic conditions are certainly present even in the existing LAN and WAN environment. The performance of many current congestion control strategies, on the other hand, have been evaluated under simple traffic scenarios and/or under stable operating conditions. Further research needs to be carried out to see if these performance studies are realistic enough, and whether the performance results of these strategies provide a true characterization of the effectiveness of these congestion control approaches. Also, more research needs to be carried out to design more and better benchmark scenarios to sufficiently capture the performance characteristics of different congestion control strategies.

Acknowledgements

Funding for this research was provided by TR*Labs* (Telecommunications Research Laboratories) in Saskatoon, and by NSERC Research Grant OGP0120969. The authors are grateful to Leanne Breker for her careful reading of this paper, as well as for asking lots of challenging and insightful questions about the loss-load model. The authors also thank the anonymous referees for their constructive comments and suggestions, which helped to improve the clarity of the final paper.

Contact information for authors: Dr. Carey L. Williamson, Department of Computer Science, University of Saskatchewan, 57 Campus Drive, Saskatoon, SK, Canada, S7N 5A9.
Email: `carey@cs.usask.ca`, `np@wipinfo.soft.net`

References

[1] 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.

[2] 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.

[3] A. Eckberg, "B-ISDN/ATM Traffic and Congestion Control", *IEEE Network Magazine*, Vol. 6, No. 5, pp. 28-37, September 1992.

[4] H. Gilbert, O. Aboul-Magd, and V. Phung, "Developing a Cohesive Traffic Management Strategy for ATM Networks", *IEEE Communications Magazine*, Vol. 30, No. 10, pp. 36-45, October 1991.

[5] 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.

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

[7] V. Jacobson, "Congestion Avoidance and Control", *Proceedings of ACM SIGCOMM '88*, pp. 158-181, August 1988.

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

[9] 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, July 1992.

[10] S. Keshav, "A Control Theoretic Approach to Flow Control", *Proceedings of ACM SIGCOMM '91*, pp. 3-15, September 1991.

[11] H. Liu, "Traffic Shaping for Congestion Control in High Speed ATM Networks", Research Report 92-6 (M.Sc. Thesis), Department of Computer Science, University of Saskatchewan, August 1992.

[12] P. Mishra and H. Kanakia, "A Hop-by-Hop Rate Based Congestion Control Scheme", *Proceedings of ACM SIGCOMM '92*, pp. 112-123, August 1992.

[13] H. Ohsaki, M. Murata, H. Suzuki, C. Ikeda, and H. Miyahara, "Rate-Based Congestion Control for ATM Networks", *ACM Computer Communication Review*, Vol. 25, No. 2, pp. 60-72, April 1995.

[14] 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.

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

[16] N. Prithviraj, "A Virtual Loss-Load Congestion Control Strategy for High Speed Networks", Research Report 94-7 (M.Sc. Thesis), Department of Computer Science, University of Saskatchewan, July 1994.

[17] 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.

[18] 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.

[19] C. Williamson and D. Cheriton, "Loss-Load Curves: Support for Rate-Based Congestion Control in Datagram Networks", *Proceedings of ACM SIGCOMM '91*, pp. 17-28, September 1991.

[20] C. Williamson, "Optimizing File Transfer Response Using the Loss-Load Curve Congestion Control Mechanism", *Proceedings of ACM SIGCOMM '93*, pp. 117-126, August 1993.

[21] 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.