Characterizing the Dynamics of the Loss-Load Curve Congestion Control Algorithm

Carey L. Williamson
Department of Computational Science
University of Saskatchewan
Saskatoon, SK, Canada S7N 0W0

February 7, 1994

Abstract

Loss-load curves are a recently proposed feedback mechanism for rate-based congestion control in packet-switched computer networks. Earlier work has provided the mathematical basis for the loss-load model and provided simulation results demonstrating its responsiveness, fairness, and stability. However, the dynamic behaviour of the algorithm in the presence of dynamic network traffic, varying network load, and non-negligible feedback delays is not yet fully understood.

This paper presents a set of simulation experiments characterizing the dynamics of the loss-load curve congestion control algorithm in these situations. A one-factor-at-a-time experimental design is used to study the behavioural properties of the algorithm as different factors are varied. Three observations are evident from the results. First, the simulation results provide validation for the mathematical properties of fairness, responsiveness, and stability of the loss-load algorithm. Second, the simulation results show that loss-load curves perform well even under dynamic network traffic, heavy network load, heterogeneous traffic flows, and small to medium feedback delays. Third, the performance of the loss-load algorithm deteriorates rapidly when the feedback delays are large. Because the loss-load algorithm is a feedback-based algorithm, loss-load curves are ineffective when the delay-bandwidth product of the network is on the order of thousands of packets.

1 Introduction

The loss-load model [6] is a recently proposed algorithm for feedback-based congestion control in packet-switched computer networks. In the loss-load model, each gateway (or switch) in the network monitors local traffic load and provides a *loss-load curve* as feedback to the senders at the edges of the network. The loss-load curve expresses, for each sender, the expected probability of packet loss for that sender, as a function of that sender's offered load to the network.

The problem with most feedback-based congestion control algorithms, such as *slow-start* [3] or DEC-bit [4], is that they provide only one bit of feedback information. They can indicate only the presence or absence of congestion. They cannot indicate where the congestion is, how heavy the congestion is, or who is causing the congestion. As a result, senders must use ad hoc rules to react to congestion when it occurs. Oscillations, underutilization, and inefficiency often result, especially as the feedback delay increases.

Loss-load curves, on the other hand, provide more than one bit of feedback. They provide enough information for senders to make informed choices about transmission rates when competing for the available network bandwidth. Senders 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.

Senders can use loss-load curves to maximize raw throughput, to minimize response time, or to control their end-to-end level of packet loss within the network. With loss-load curves, senders have enough information to choose a proper initial rate for transmission into the network. The loss-load curve also provides enough information for senders to respond to dynamic changes in network load, both increases and decreases.

Earlier work [6] has established the mathematical foundation for loss-load curves and demonstrated their properties of fairness, robustness, and stability. The loss-load model works well for simple network environments where each source has a large number of packets to transmit, and wishes to maximize raw throughput. The loss-load curve can also be used by senders to optimize response time for file transfer across a network [5]. Yet what remains to be clearly understood are the fundamental behavioural properties of the loss-load algorithm: how well it responds to dynamic network traffic, how well it behaves as the number of senders or the size of the network increases, and how well it performs in the presence of feedback delays.

The purpose of this paper, then, is to characterize the dynamics of the loss-load congestion control algorithm in precisely these types of situations. This paper will demonstrate, once and for all, the performance capabilities and limitations of the loss-load curve congestion control algorithm.

The remainder of this paper is organized as follows. Section 2 provides background material on loss-load curves, summarizing their formulation and some of their important mathematical properties. Section 3 examines the behavioural properties of the loss-load algorithm in a very simple network configuration when all senders are maximizing raw throughput.

Section 4 presents a more detailed simulation designed to study the loss-load algorithm and the response time model in a network with dynamic traffic flows and non-negligible feedback delays. Finally, Section 5 presents conclusions and directions for further research.

2 The Loss-Load Model

2.1 Loss-Load Philosophy

The loss-load curve provides a very general resource allocation mechanism, allowing all senders 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 sender should get its *fair share* of the bandwidth, if it wishes, subject to the bandwidth requests of the other senders. If the sum of the bandwidth requests from all active senders is less than or equal to the available network capacity, then each sender receives exactly its requested bandwidth allocation. If the sum of the bandwidth requests from all active senders exceeds the available capacity, then partial bandwidth allocations are made to each sender.

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

2.2 Network Model

The network environment considered by the loss-load algorithm consists of *senders*, *gateways*, and *links*. Senders[1] are the sources of traffic on the network, injecting packets that traverse one or more links and gateways en route to their destination. The sender chooses a path (virtual channel) at the time of call set up, and specifies a desired bandwidth request r. Senders can adjust their bandwidth request (i.e., rate of transmission of packets) at any time based on their traffic requirements and the current network load. Senders use a leaky bucket or similar mechanism to transmit packets at their chosen rate.

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

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

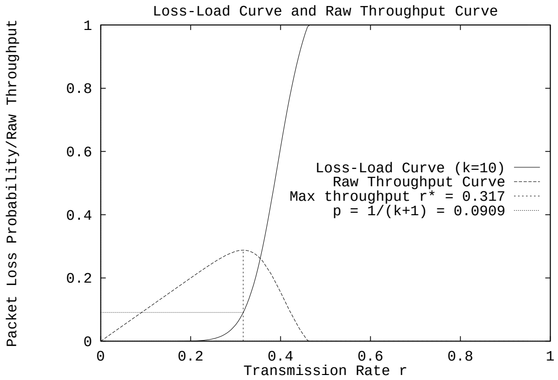

Figure 1: Loss-Load Curve and Raw Throughput Curve

2.3 The Loss-Load Curve

An example of a loss-load curve appears in Figure 1. The curve expresses the tradeoff between the level of packet loss p for a sender at a particular gateway as a function of that sender's offered load. The loss-load curve is a monotonically increasing function of offered load, where offered load (the sender's bandwidth request) is normalized to the capacity C of the gateway, to give a fractional bandwidth request $0 \leq r \leq 1$. The packet loss probability reaches 100% for high offered load in order to protect the network from misbehaving senders [6].

The loss-load curve changes dynamically with time, as load conditions change in the network. Gateways explicitly tell senders the current loss-load curve, and senders 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 2.4. The parameter a represents the aggregate load at the gateway, for all sender traffic streams using the gateway. The parameter b measures the distribution of the aggregate load amongst the senders. A sender can compute the entire loss-load curve for a particular gateway given only that gateway's current values for a and b. This compact representation makes it easy for gateways to exchange loss-load curves with senders across the network.

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

A sample raw throughput curve appears in Figure 1. The throughput curve increases

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

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

2.4 The Loss-Load Formula

Mathematically, the loss-load curve for a single gateway is defined as follows [6]. Assume that the gateway is forwarding traffic from one or more input links onto a single output link of capacity C. Let M be the number of senders sharing the gateway. Let r_i represent the packet arrival rate[3] for the ith sender, and $r_{total} = \sum_{i=1}^{M} r_i$ represent the aggregate traffic load arriving at the gateway. Then the probability of packet loss $P(r)$ for the $(M+1)$st sender when transmitting at rate r is given by

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

where $a = r_{total} - C$ and $b = \sum_{i=1}^{M} r_i^{k+1}$. The parameter a measures the total (excess) traffic load at the gateway, and the parameter b measures the distribution of the total load amongst the competing senders. The parameter $k > 0$ is an integer parameter that can be set as a global constant for the network. The importance of k in determining the behaviour of the loss-load algorithm is summarized in Section 2.5.

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

- If the total arriving load is less than the capacity C of the gateway, then no packets are discarded. That is, $p_i = 0$ for all i.
- If the total load is greater than the capacity, then *all* senders share in packet loss (but not necessarily equally). That is, $p_i > 0$ for all i.
- The p_i's are a non-decreasing function of the sender rates r_i.

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

2.5 Properties of Loss-Load Formula

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

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

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

That is, the level of packet loss at r^*_τ is always bounded. The bound is always the same, regardless of the number of senders, and regardless of the network load. The larger k, the lower the level of packet loss at the throughput optimum rate r^*_τ. Theorem 1 is illustrated in Figure 1, where $k = 10$.

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

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

2.6 Summary

The loss-load curve provides a very general resource allocation mechanism, allowing all senders on the network to dynamically compete for and share the available network capacity. A key assumption in the loss-load model is that it is not unreasonable to operate a packet-switched network at non-negligible levels of packet loss. In fact, the loss-load algorithm is stable only at slight levels of overload and packet loss in the network. This is a provable mathematical property of the loss-load algorithm.

3 Behaviour for Simple Traffic Flows

This section presents simulation results for a very simple traffic model where all senders are maximizing raw throughput. The purpose of this section is to establish the fundamental behavioural properties of the loss-load algorithm prior to the more detailed simulations presented in Section 4.

The network configuration used for this set of experiments is shown in Figure 2. Two senders, S1 and S2, share a single bottleneck link en route to destination D3. All links have the same capacity $C = 1.0$, and all links have the same short propagation delay (one time unit). All times in the simulation are normalized to the transmission time of a packet on the bottleneck link.

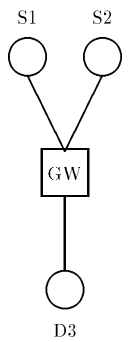

Figure 2: Network Configuration for Simple Traffic Experiments

3.1 Two Senders

Consider two senders optimizing raw throughput when $k = 2$. Both sender connections have a large number of packets to transmit (1000), and both wish to maximize raw throughput. Suppose that the first sender (CONN 0) arrives at time 0, and the second sender (CONN 1) arrives at time 40. CONN 0, arriving first, sees no load at the gateway (a degenerate case of the loss-load curve), and chooses a transmission rate $r_0 = 1.0$. When CONN 1 arrives at time 40, both senders must adjust their rates to dynamically compete for and share the available link capacity.

Figure 3 illustrates the behaviour of the loss-load algorithm for this case. The figure shows three graphs: CONN 0's transmission rate, CONN 1's transmission rate, and the aggregate offered load r_{total} arriving at the gateway GW destined for the bottleneck link (LINK 0). All graphs are plotted as a function of simulation time. Before CONN 1 arrives, CONN 0 uses the full bandwidth of the bottleneck link, transmitting at rate $r_0 = 1.0$. When CONN 1 arrives, CONN 1 uses the current loss-load curve at the gateway to choose a rate $r_1 = 0.7937$ to maximize its raw throughput in the presence of CONN 0's traffic (i.e., since $k = 2$ and there is one sender with $r_0 = 1.0$, CONN 1 computes $a = r_{total} - C = 1.0 - 1.0 = 0.0$, and $b = 1.0^3 = 1.0$, and $r_\tau^* = 0.7937$). Upon seeing the new loss-load curve reflecting CONN 1's rate (namely, $a = r_{total} - C = (1.0 + 0.7937) - 1.0 = 0.7937$, and $b = 1.0^3 + 0.7937^3 = 1.5000$), CONN 0 then reduces its rate to $r_0 = 0.7517$ based on the loss-load curve obtained by subtracting its rate out from the totals given above (i.e., $a = -0.2063$ and $b = 0.5000$, thus $r_\tau^* = 0.7517$). Upon seeing CONN 0's new adjustment, CONN 1 chooses $r_1 = 0.7500$, and then CONN 0 subsequently chooses $r_0 = 0.7500$ as well. At this point, convergence is complete, and the total load is $r_{total} = \frac{k+1}{k} = 1.50$, as predicted by Theorem 2. The bottom plot in Figure 3 shows the total load r_{total} for the bottleneck link, illustrating the new equilibrium. Once equilibrium is reached, no further rate adjustments are required unless the traffic load changes.

3.2 Effect of k

Figures 4 through 6 show the corresponding behaviour for different values of k when two senders are maximizing raw throughput. The network configuration is the same as in Figure 2. The network traffic workload is the same as in the previous experiment (i.e., two senders, each maximizing raw throughput, arriving at time 0 and time 40, respectively).

Figures 4 through 6 show behaviour very similar to Figure 3. When the new connection (CONN 1) arrives at time 40, rate adjustments are made by each sender until the new equilibrium is reached. The convergence process is iterative, requiring two to four rate adjustments by each sender. In all cases, equilibrium is approached monotonically from above. This is a proven property of the loss-load algorithm.

The parameter k has three main effects on the loss-load algorithm. First, it controls the level of packet loss at the raw throughput optimum rate r_r^*. This packet loss probability is always $\frac{1}{k+1}$. Second, it determines the level of load at equilibrium, which is always $\frac{k+1}{k}$. Third, it affects the speed of convergence to the equilibrium load.

The latter two of these effects are clearly illustrated in Figures 3 through 6. The larger k is, the lower the level of overload at equilibrium in each simulation (i.e., the vertical distance between the equilibrium load r_{total} and the capacity C). The equilibrium load diminishes from 1.50 for $k = 2$ (when the probability of packet loss is $p = 0.3333$) to 1.05 for $k = 20$ (when the probability of packet loss is $p = 0.0476$). However, increasing k slows down the convergence to equilibrium. The larger k is, the smaller the rate adjustments by each sender, and the slower the convergence to the new equilibrium. This effect is most noticable in Figure 6, where it takes several rate adjustments by each sender before the new equilibrium is reached.

Values of k larger than $k = 20$ are undesirable for two reasons. First, they are sluggish to respond to changes in load, as shown in Figure 6. Second, they introduce the possibility of mathematical underflow (and thus inaccuracy in the loss-load model) when computing r^{k+1} for small r.

A reasonable tradeoff between the responsiveness of the loss-load algorithm and the level of overload at equilibrium is achieved for $k = 10$. For implementation reasons, values such as $k = 7$ or $k = 15$ may be attractive so that $k + 1$ is a power of 2.

3.3 Effect of Feedback Delay

Figure 7 illustrates the effect of increased feedback delay on the loss-load algorithm. The traffic scenario and the value of k ($k = 20$) in this example are the same as in Figure 6, but the one-way propagation delay for each sender in Figure 2 has been changed from 1 time unit to 10 time units.

Feedback delays make the loss-load algorithm slower to respond to changes in network load, but equilibrium is still achieved. Convergence is an iterative process, as in Figure 6. In fact, the iterative convergence of the loss-load algorithm is even more evident in Figure 7 than in Figure 6. Note the discrete steps, as well as the alternating pattern of adjustments (action and reaction) by each of the senders.

The same equilibrium load is reached as in Figure 6, regardless of the feedback delay. In fact, even the vertical step sizes for rate adjustments are exactly the same. This is another property of the loss-load algorithm.

The same equilibrium would be achieved even if the feedback delays to each sender were not identical. In other words, feedback delays do not change the convergence or stability properties of the loss-load algorithm. Clearly, however, it takes much longer to reach equilibrium when the feedback delay increases.

3.4 Multiple Senders

The preceding experiments considered only two senders sharing the network. However, the loss-load model works just as well for multiple senders. The experiment reported in this subsection presents a traffic scenario where the number of senders ranges from 1 to 6. Initially, one sender (CONN 0) has sole use of the network. Additional senders (CONN 1 through CONN 5) then arrive, one at a time, every 40 time units. The additional connections then depart, one by one, until CONN 0 has sole use of the network once again.

Figure 8 presents the results for this traffic scenario with $k = 2$, and all feedback delays of 1 time unit. The figure shows CONN 0's transmission rate at the top, and then a plot of r_{total} at the bottleneck link, as a function of time.

The arrival of each new connection into the network creates an upward spike in the total load at the bottleneck link. Following a period of action and reaction by the active senders, the equilibrium load is again reached. Note that the size of the vertical spike diminishes with the number of senders already in the network (since new senders will always choose a lower rate than the existing senders are using), and that the time required for convergence back to equilibrium increases only marginally with the number of senders.

Connection departures (at times 244, 286, 326, 345 and 380) create a downward spike in the aggregate load, as r_{total} temporarily drops. With the next loss-load curve update, however, all senders see the "extra" capacity available, and increase their rates. This creates another short upward spike, before diminishing back down to equilibrium as before. Again, the magnitude of the upward and downward spikes depends on the number of senders. The larger the number of active senders, the smaller the fluctuations.

3.5 Summary

The parameter k is the single most important control parameter in the loss-load algorithm. The choice of k presents a tradeoff between the level of overload in the network and the responsiveness of the loss-load algorithm. The larger k is, the less aggressive the loss-load algorithm is in terms of network overload and packet loss.

4 Behaviour for Dynamic Traffic Flows

This section presents simulation results for the loss-load algorithm in a network with highly dynamic traffic flows. Senders at the edges of the network use the response time model [5] to choose their transmission rates based on the loss-load curves provided by the network.

4.1 Network and Traffic Model

The network configuration used for this set of simulation experiments is shown in Figure 9. The configuration is based on a simple topology called the "connected star". A single connected star contains two source nodes (S_i) and two destination nodes (D_i) arranged in a star configuration around a single gateway (G_i). Single star components can be connected together in series to form arbitrarily long networks. A connected star network with N stages has $2N$ sources and $2N$ destinations, and thus $4N^2$ possible source-destination pairs. Connected star networks allow the study of connections that have different numbers of hops and different round trip times. At the same time, the regular topology of the connected star makes routing trivial.

The specific network configuration used for the simulations is a connected star with five stages, as shown in Figure 9. There are 10 source nodes, 10 destination nodes, and 5 gateways. Each gateway has 100 buffers. All links are bidirectional, and all links have the same capacity $C = 1.0$. The one-way propagation delays for the links are also shown in Figure 9. The round trip delay for different source-destination pairs ranges from a minimum of $R = 4$ to a maximum of $R = 52$.

The traffic workload presented to the network consists of 2000 transport level connections. Connections arrive according to a Poisson process with rate $\lambda = 0.1$. Each connection specifies a source node, a destination node, and a number of packets to transmit. Each possible source-destination pair in the network is equally likely to be chosen as the source and destination for each connection.

Each connection represents one sender's file transfer activity. The number of bytes per connection is generated from an empirical distribution for ftp traffic [1], using the *tcplib* package [2]. Approximately 66% of the connections transfer fewer than 1000 bytes. The simulated network uses fixed-size 48-byte packets for transmission. All times in the simulation are normalized to the transmission time of one packet.

4.2 Experimental Methodology

The simulation experiment is conducted using a *one-factor-at-a-time* experimental design. The factors chosen for the experiment are the loss-load parameter k, the number of active senders, and the magnitude of the feedback delay. For each factor, four levels were chosen and simulated.

The performance impact of these factors are assessed by studying the dynamic load behaviour at the bottleneck link in the simulated network, both qualitatively and quantitatively. In all simulations, the bottleneck links in Figure 9 are the two horizontal links between G2 and G3 and between G3 and G4. The link between G2 and G3 (hereafter called LINK 9) is arbitrarily chosen to illustrate the dynamic behaviour of the loss-load algorithm.

The dynamic behaviour of the algorithm is represented using a *load profile* graph. The load profile shows the aggregate offered load at the link as a function of simulation time.

4.3 Effect of Dynamic Traffic

Figure 10 shows the load profile graph for link 9 during the simulation for $k = 2$. The topmost graph presents the aggregate offered load at link 9 for the entire duration of the simulation. The bottom three graphs present the same load information for smaller and smaller selected subintervals of the simulation. The mean number of active connections on link 9 during the simulation was 4.951.

The load profile graph shows significant fluctuations in load at the bottleneck link. The fluctuations are caused primarily by the dynamic nature of the traffic model assumed (i.e., connection arrivals and departures), and not by the loss-load algorithm itself. The bottom graph, for instance, shows that the loss-load algorithm handles individual connection arrivals and departures in the same manner as in Figure 8.

The total offered load at link 9 is nearly always between $C = 1.0$ and $\frac{k+1}{k} = 1.50$. The aggregate offered load is nearly always[4] above 1.0 because of the aggressive nature of the loss-load algorithm. Stability is achieved only at non-negligible levels of packet loss. The aggregate offered load is nearly always less than the predicted bound of 1.50 because most senders are using the response time model (a more conservative approach) to choose their transmission rate, instead of the raw throughput model. All in all, the total offered load at link 9 for $k = 2$ is at or below the predicted bound 98.18% of the time.

In the remaining sections, only the load profile between time 50000 and time 100000 will be used to illustrate the performance impact of different factors on the loss-load algorithm. There are two reasons behind this choice of interval. First, link 9 is a heavily loaded bottleneck link during this interval. Second, this choice of interval eliminates startup and end effects from the discussion of the results.

4.4 Effect of k

The effect of the parameter k on the performance of the loss-load algorithm is illustrated in Figure 11. Four different load profile graphs are presented: $k = 2$, $k = 5$, $k = 10$, and $k = 20$. The traffic workload (i.e., arrival of connections) presented to the network is identical for

[4]The load could drop below 1.0 if some other link becomes the bottleneck in the network. In this simulation run, for example, the link between G3 and G4 becomes the bottleneck between times 125000 and 140000.

each value of k. The mean number of active connections on link 9 during each simulation is approximately 4.5.

In all cases, the total offered load remains at or below the predicted bound of $\frac{k+1}{k}$ most of the time (e.g., 98.18% for $k = 2$, 95.27% for $k = 5$, 90.59% for $k = 10$, and 86.78% for $k = 20$). Larger values of k give a tighter bound on the predicted level of overload, but the magnitude of deviations from the predicted bound increases, and the time spent above the predicted bound also increases.

The average level of packet loss in the simulation (not shown in the figure) decreases with increasing k, as predicted by the loss-load model (e.g., 26.93% for $k = 2$, 11.66% for $k = 5$, 6.48% for $k = 10$, and 4.52% for $k = 20$). The actual levels of packet loss are always less than the predicted level $\frac{1}{k+1}$ because most senders are using the response time model. However, the magnitude of the difference between the predicted and actual packet loss diminishes for large k. In other words, the advantages of using the more conservative response time model diminish for larger k.

Values of k larger than $k = 20$ are unattractive for the reasons mentioned above. A reasonable tradeoff between responsiveness and packet loss is obtained for $k = 10$.

4.5 Effect of Number of Senders

The effect of the number of active senders is illustrated in Figure 12. These four plots all use $k = 10$, but use different connection arrival rates to produce different levels of network load. From top to bottom, the mean number of active connections on link 9 during each simulation were 4.248, 4.928, 7.064, and 32.736.

As the number of active connections increase (from the top graph to the bottom graph), the smaller the fluctuations in the total load at the bottleneck link. In other words, the dynamics of the loss-load algorithm improve as the number of active senders increase. This makes sense, intuitively, since the more active senders there are, the smaller the contribution of each sender to the total load, and the lower the rate chosen by new senders starting up. As a result, the magnitude of the fluctuations diminish, and the loss-load algorithm consistently spends about 90% of the time at or below the predicted equilibrium load.

4.6 Effect of Feedback Delay

Figure 13 illustrates the effect of feedback delay on the dynamics of the loss-load algorithm. Different feedback delays are simulated by applying a scalar multiplier D to all the link propagation delays in Figure 9. In this experiment, D is varied from $D = 1$ to $D = 1000$ in multiples of 10, generating round trip time ranges from 4 to 52 packet times for $D = 1$ and from 4000 to 52000 packet times for $D = 1000$. In all four graphs, the value of k is 10, and the presented traffic workload is the same as in Figure 11.

The performance of the loss-load algorithm deteriorates rapidly as the feedback delay increases. As D increases, the magnitude of displacements from the predicted load increases, and the time spent above the predicted load increases. For example, the time spent at or below the predicted load bound of 1.10 is 90.59% for $D = 1$, 80.65% for $D = 10$, 57.38% for $D = 100$, and 0.14% for $D = 1000$. In the latter case, connections arrive far too quickly for the loss-load algorithm to respond, and the network becomes irretrievably overloaded.

The loss-load algorithm is highly effective when the feedback delays are small (i.e., $D = 1$ and $D = 10$), is tolerable for $D = 100$, and completely ineffective for $D = 1000$. Because the loss-load algorithm is a feedback-based control scheme, it can not possibly work for feedback delays on the order of thousands of packets.

5 Conclusions

This paper presents a set of simulation experiments characterizing the dynamics of the loss-load curve congestion control algorithm in a simple network configuration. A one-factor-at-a-time experimental design is used to study the behavioural properties of the algorithm as different factors are varied. Three conclusions are evident from the results.

First, the simulation results provide validation for the mathematical properties of fairness, responsiveness, and stability of the loss-load algorithm. The algorithm provides bounds on the average level of packet loss in the network, and a bound on the aggregate offered load to the network at equilibrium. The actual level of packet loss is often significantly lower than the bound predicted by the loss-load model because of the use of the response time model. A separate analysis of per connection performance [5] shows that the loss-load algorithm is remarkably fair for connections with a different number of packets and different round trip times.

Second, the simulation results show that loss-load curves perform well even under dynamic network traffic, heavy network load, heterogeneous traffic flows, and small to medium feedback delays. While dynamic traffic can create transient load changes that displace the network from equilibrium, the loss-load algorithm reacts quickly, and maintains an equilibrium load within the bounds predicted by the loss-load algorithm a large percentage of the time.

Third, the performance of the loss-load algorithm deteriorates rapidly when the delay-bandwidth product of the network is on the order of thousands of packets. Because the loss-load algorithm is a feedback-based algorithm, loss-load curves are ineffective when the feedback delays are large. While this effect is less pronounced when the number of active connections is large, it nevertheless places an ultimate limit on the effectiveness of the loss-load algorithm in high-speed wide-area packet-switched networks.

Ongoing research focusses on a *virtual loss-load* congestion control protocol that is much less aggressive (i.e., loses fewer packets) than the original loss-load model, while still maintaining the desirable mathematical properties of the loss-load algorithm. The dynamic loss-load curve principle is also being used to characterize delay for round-robin queueing in packet-switched networks.

References

[1] R. Caceres, P. Danzig, S. Jamin, and D. Mitzel, "Characteristics of Individual Application Conversations in TCP/IP Wide-Area Internetworks", Proceedings of ACM SIGCOMM '91, pp. 101-112, September 1991.

[2] P. Danzig and S. Jamin, "tcplib: A Library of TCP Internetwork Traffic Characteristics", Technical Report, Computer Science Department, University of Southern California, September 1991.

[3] V. Jacobson, "Congestion Avoidance and Control", Proceedings of ACM SIGCOMM '88", pp. 314-329, August 1988.

[4] K.K. Ramakrishnan and R. Jain, "A Binary Feedback Scheme for Congestion Avoidance in Computer Networks with a Connectionless Network Layer", Proceedings of ACM SIGCOMM '88, pp. 303-313, August 1989.

[5] C. Williamson, "Optimizing File Transfer Response Time Using the Loss-Load Curves Congestion Control Algorithm", Proceedings of ACM SIGCOMM '93, pp. 117-126, September 1993.

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

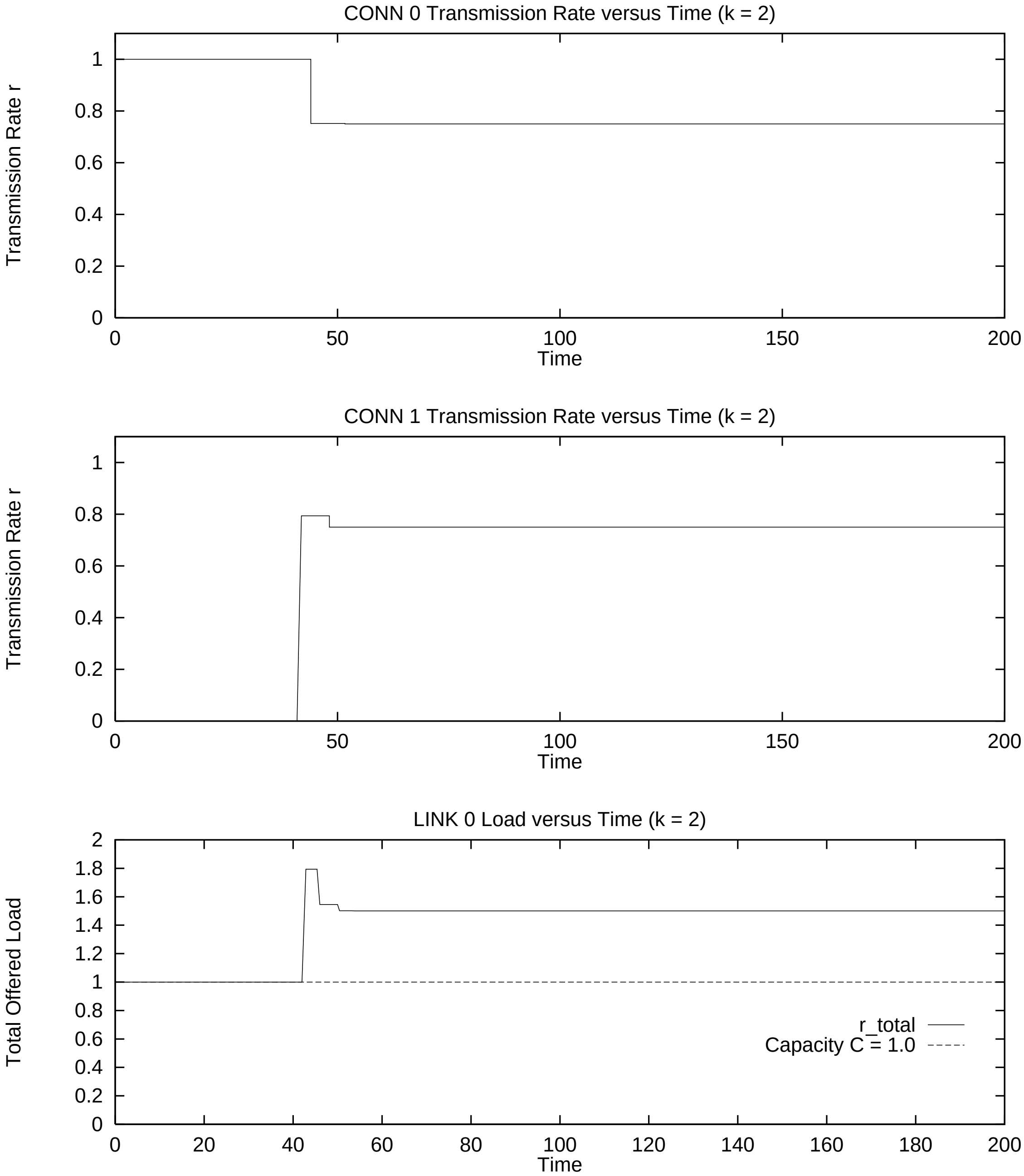

Figure 3: Two connections maximizing raw throughput, $k = 2$. Connection 0 starts up at time 0. Connection 1 starts up at time 40. Both senders compete for the shared bottleneck link (link 0). After a period of action and reaction, both senders converge to a stable rate of 0.75. At equilibrium, the aggregate offered load at the bottleneck link is $\frac{k+1}{k} = 1.50$.

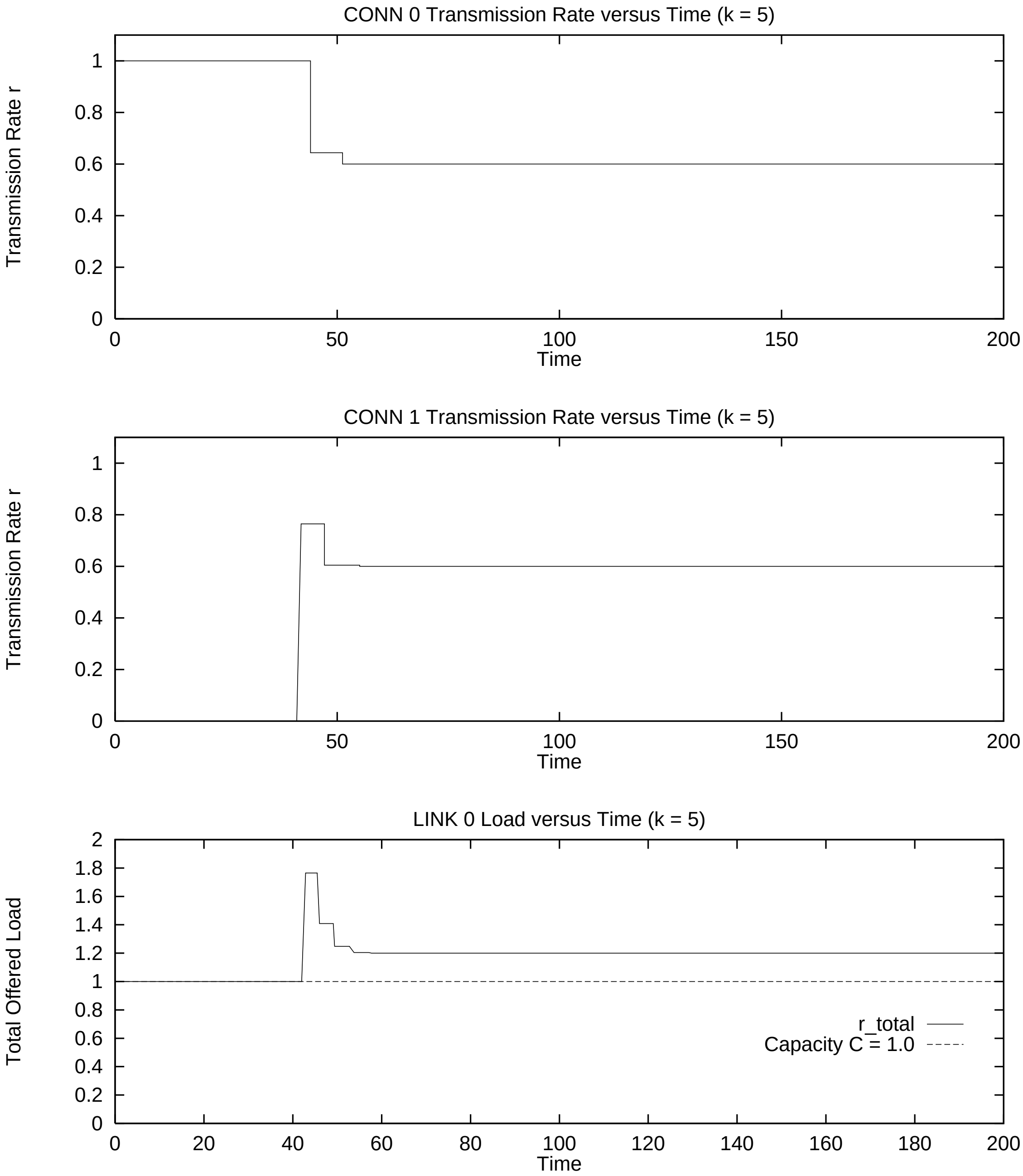

Figure 4: Two connections maximizing raw throughput, $k = 5$. At equilibrium, the aggregate offered load at the bottleneck link is $\frac{k+1}{k} = 1.20$.

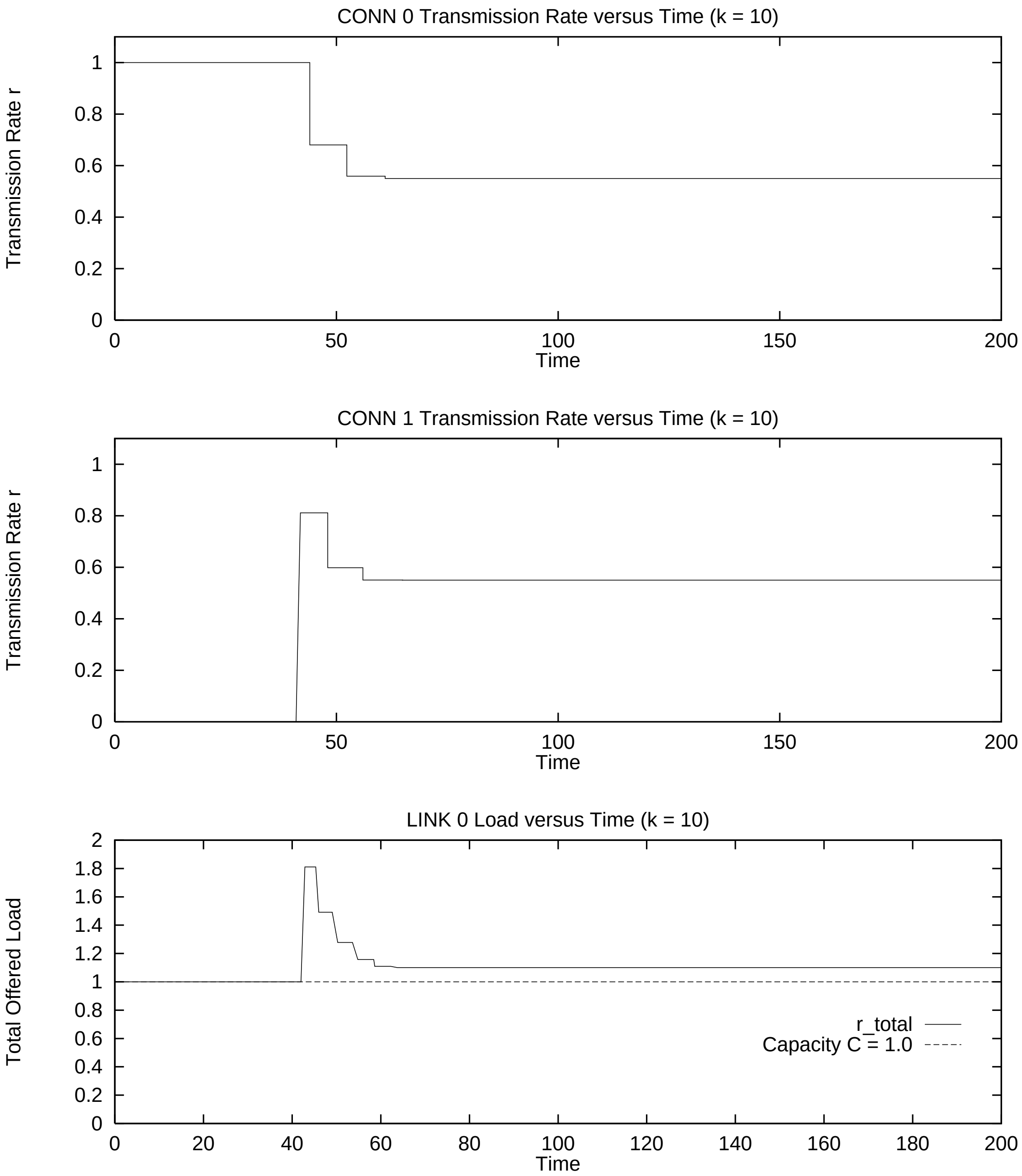

Figure 5: Two connections maximizing raw throughput, $k = 10$. At equilibrium, the aggregate offered load at the bottleneck link is $\frac{k+1}{k} = 1.10$.

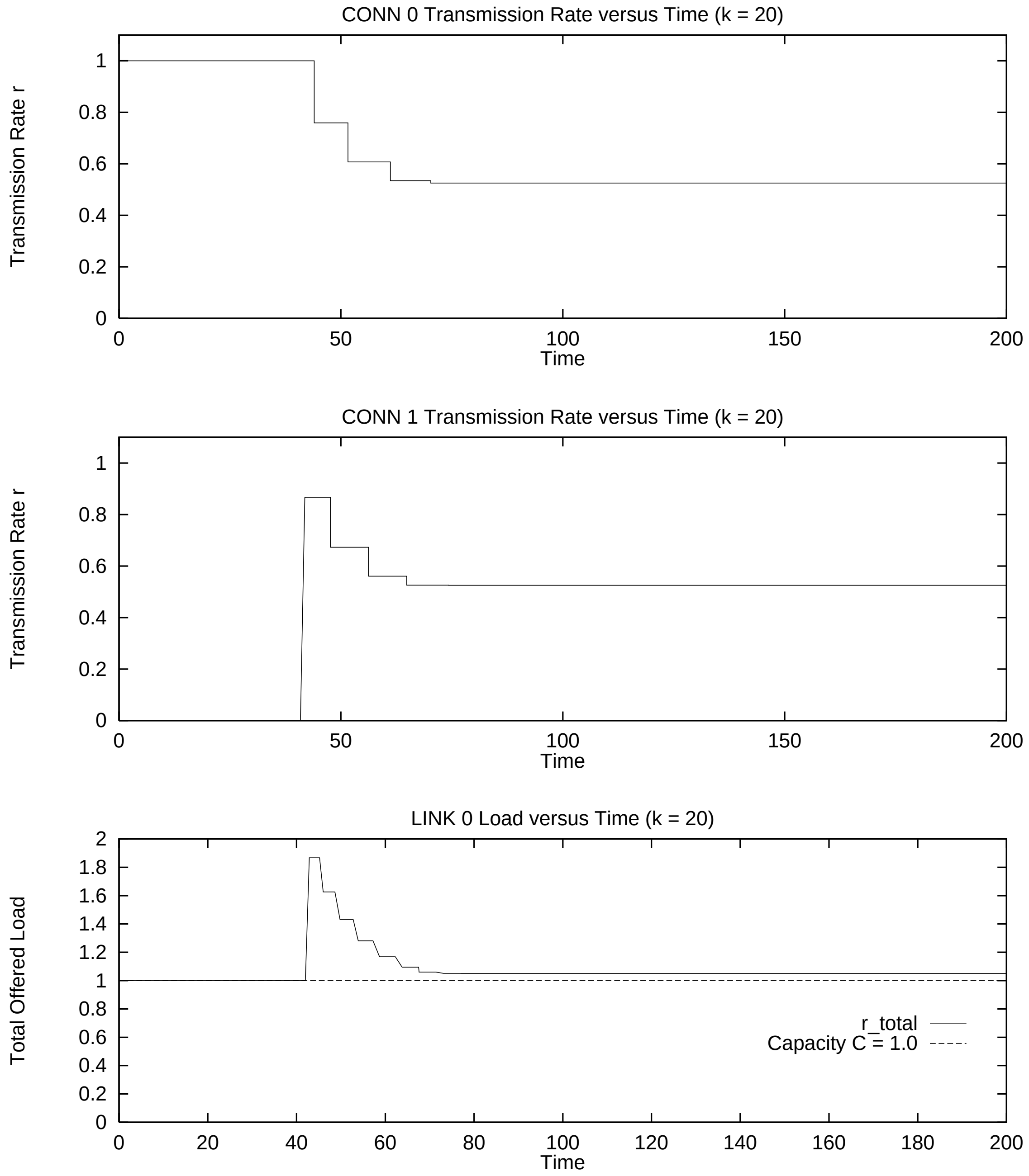

Figure 6: Two connections maximizing raw throughput, $k = 20$. At equilibrium, the aggregate offered load at the bottleneck link is $\frac{k+1}{k} = 1.05$. Note that the time taken to reach equilibrium is longer for larger values of k since the size of the rate adjustments at each step are smaller. More steps are required before equilibrium is reached.

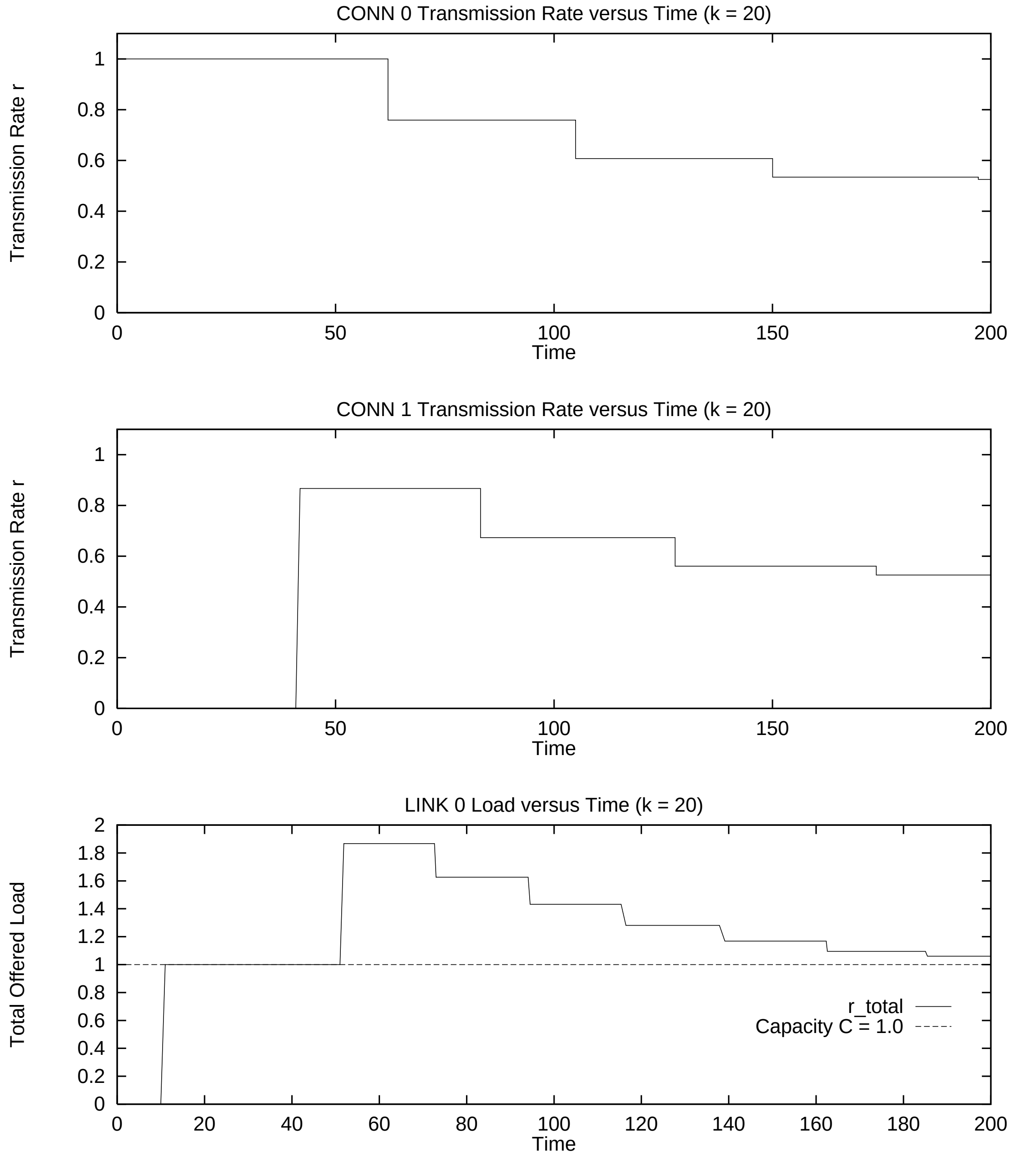

Figure 7: Two connections maximizing raw throughput when the feedback delay is non-negligible, for $k = 20$. The alternating pattern of action and reaction by each connection is evident in this figure, as each sender reacts to the other sender's choice of rate. The bottleneck link's perception of load (the bottom graph) is always one feedback delay (10 time units) out of date with respect to the sender's current choice of rate (the top two graphs). The vertical step sizes and the equilibrium achieved are the same as in Figure 6, but it obviously takes longer to reach equilibrium because of the feedback delays.

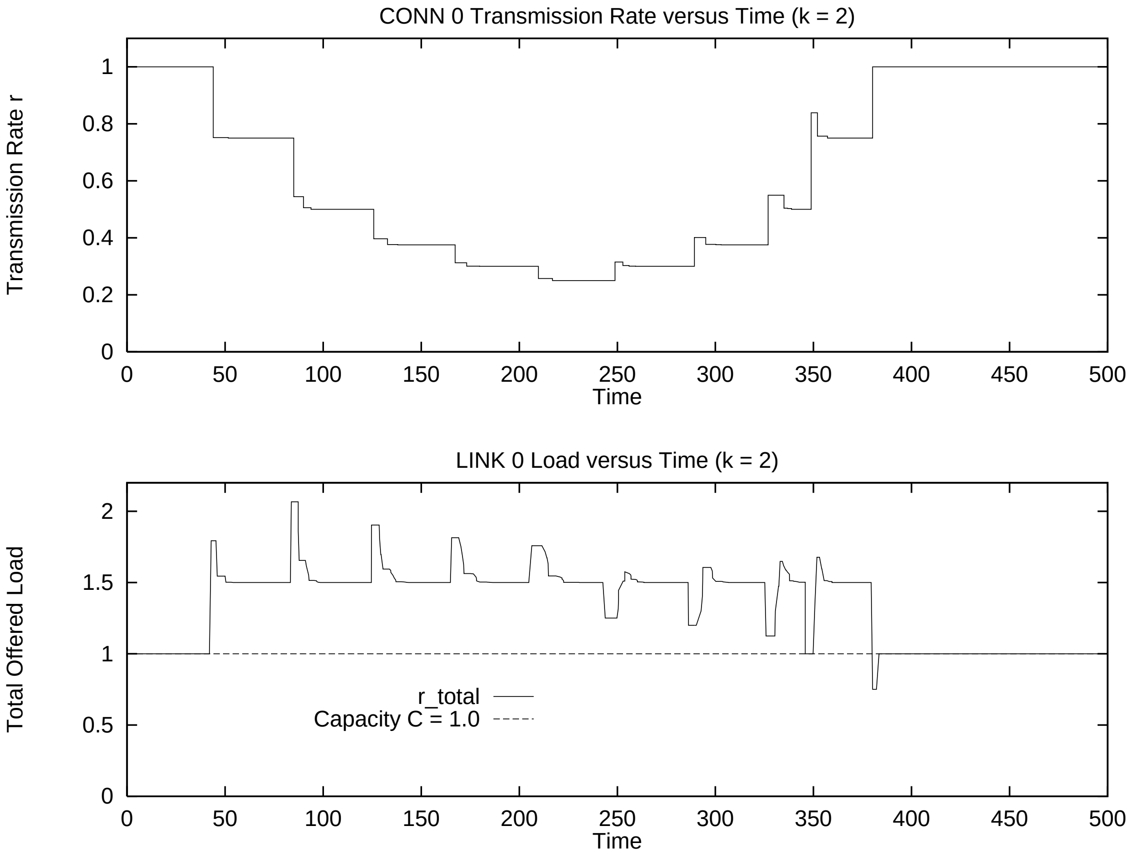

Figure 8: Simulation results for multiple senders, $k = 2$. The first connection starts at time 0, and new connections are added at times 40, 82, 123, 163, and 204, respectively. Connections depart (in reverse order) at times 244, 286, 326, 345, 380, and 1248, respectively. The top graph shows connection 0's reaction to each arrival and departure. The bottom graph shows the aggregate load at the bottleneck link at each point in time.

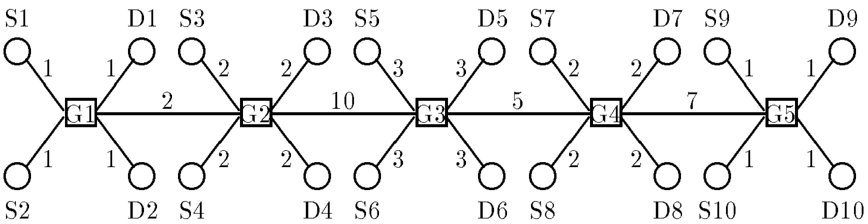

Figure 9: Network Configuration for Dynamic Traffic Experiments

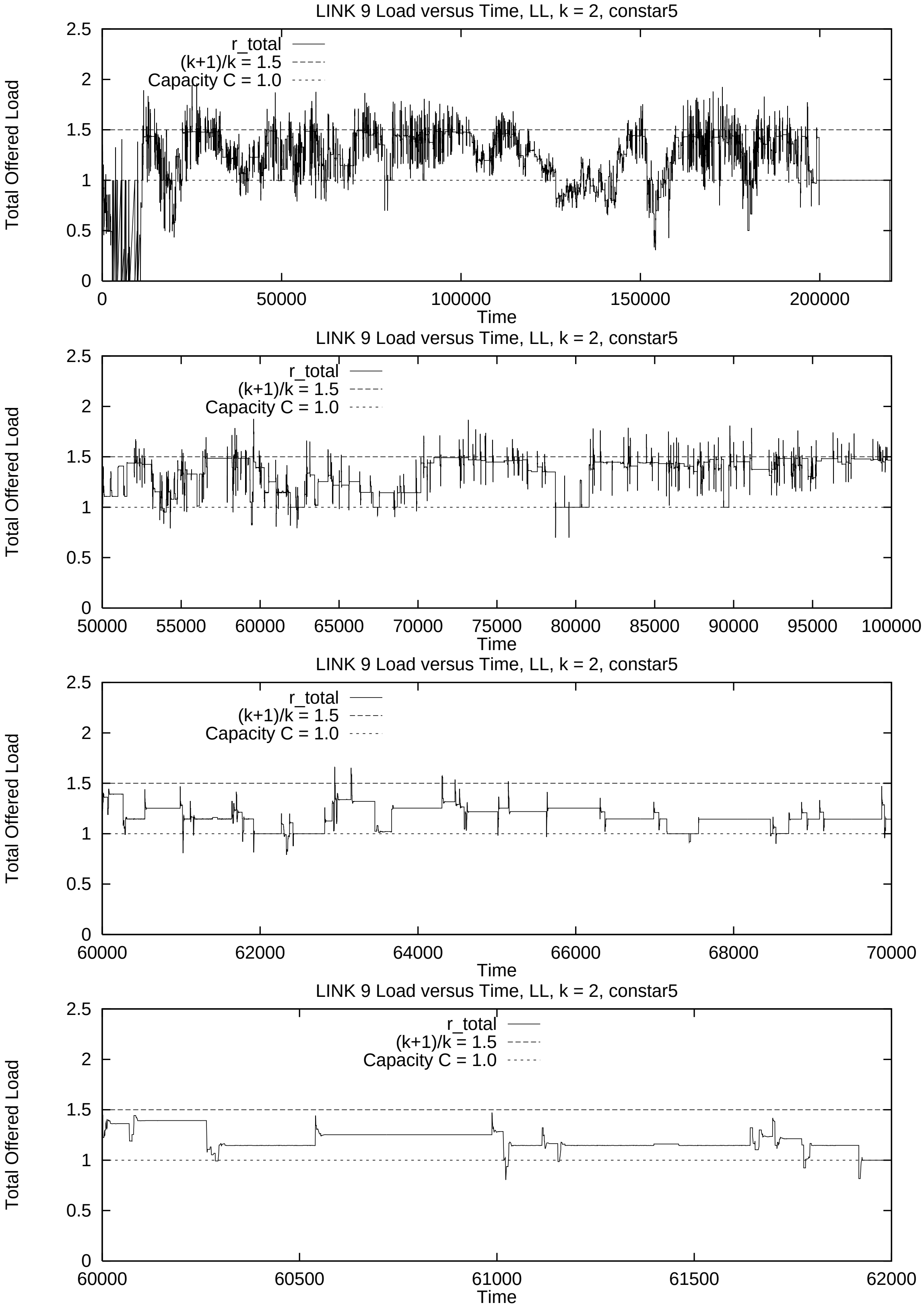

Figure 10: Load profile for link 9 for dynamic traffic, $k = 2$. The top graph shows the aggregate offered load at link 9 (between gateway G2 and gateway G3) for the entire duration of the simulation. The bottom three graphs show greater detail for selected subintervals of the simulation. The offered load remains at or below the predicted bound 98% of the time.

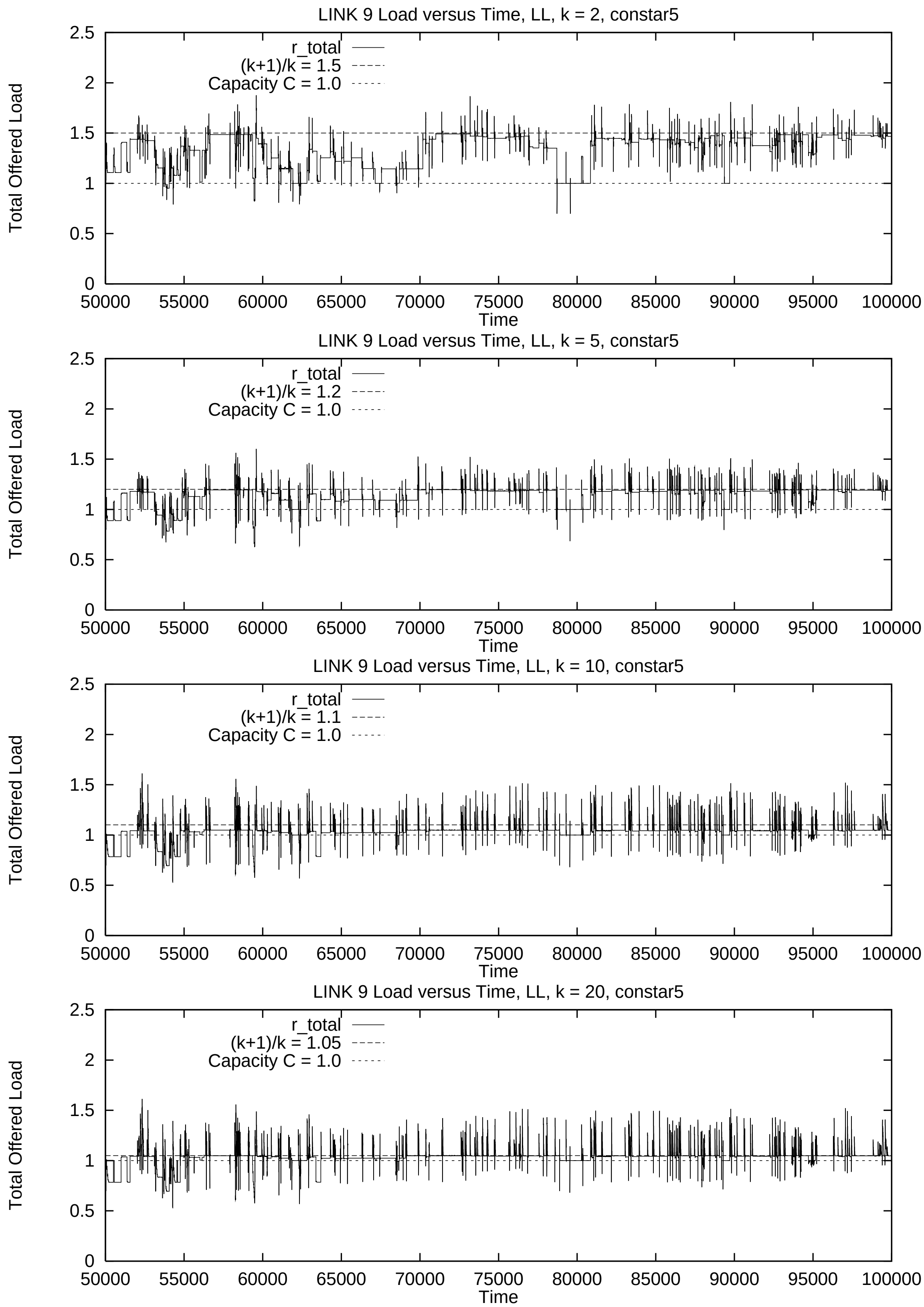

Figure 11: Load profile for link 9 for dynamic traffic when $k = 2, 5, 10, 20$. Larger values of k give a tighter predicted bound on load, but the simulation shows that slightly less time is spent at or below the predicted bound.

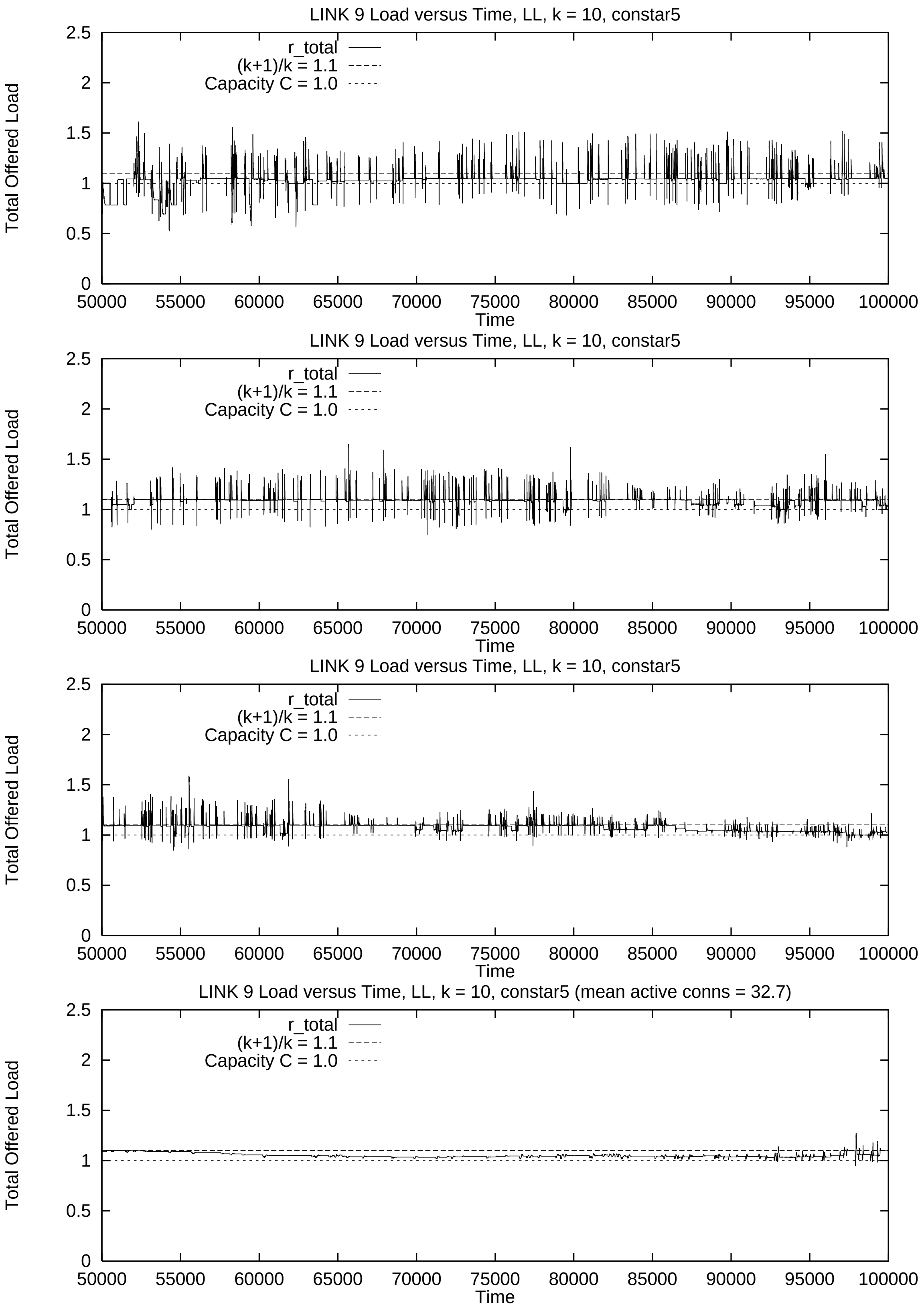

Figure 12: Load profile as number of active connections increases, $k = 10$. These results show the offered load on link 9 for four different connection arrival models. The larger the number of active connections, the smaller the effect of transient load changes when connections arrive and depart. The loss-load algorithm is stablest when the number of active senders is large.

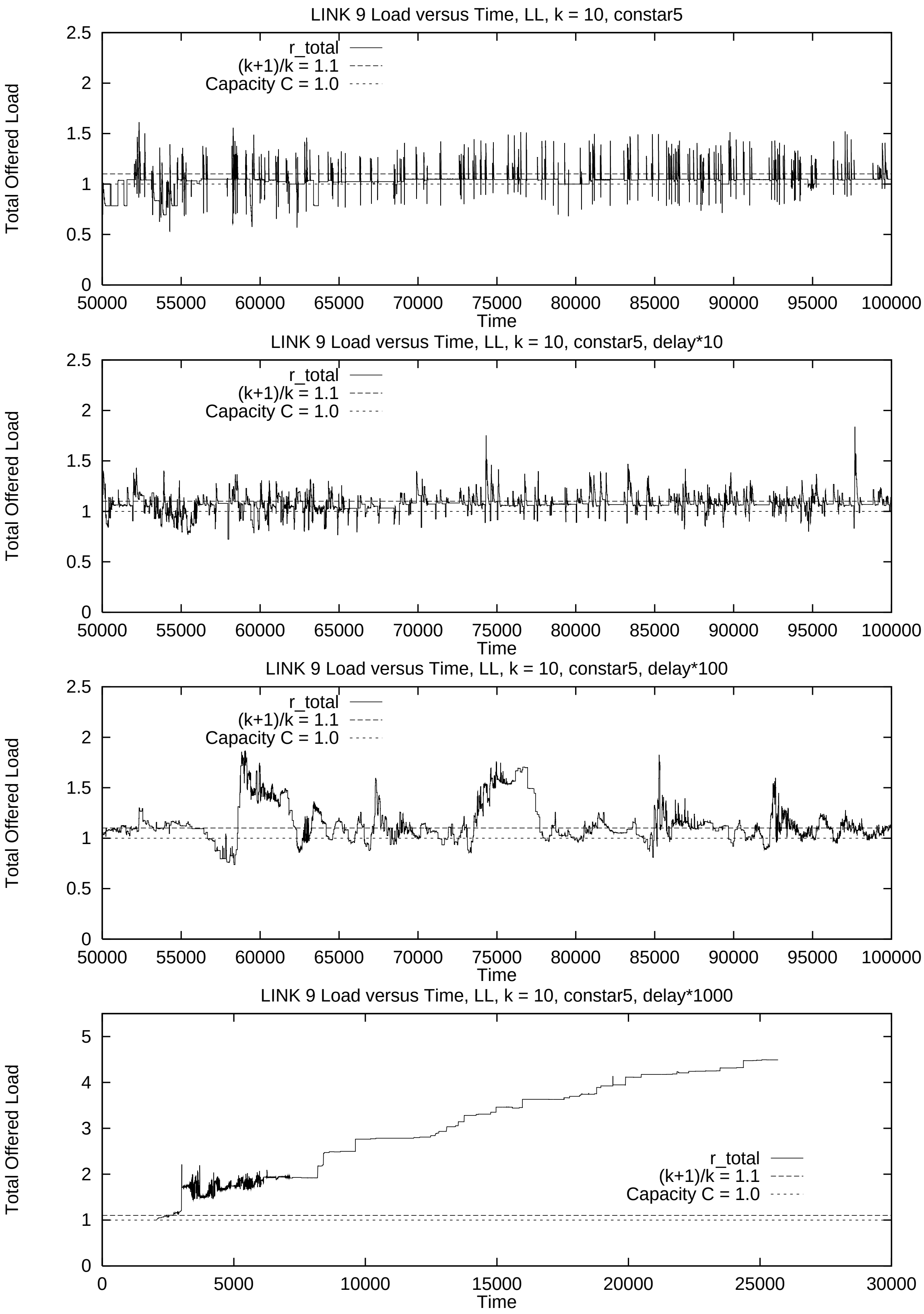

Figure 13: Load profile on link 9 as feedback delay increases, $k = 10$. Feedback delays are all expressed relative to Figure 9. The longer the feedback delay, the larger the transient load effects are, and the less time that the offered load remains at or below the predicted bound.