A Binary Feedback Scheme for Congestion Avoidance in Computer Networks

K. K. RAMAKRISHNAN and RAJ JAIN
Digital Equipment Corporation

We propose a scheme for *congestion avoidance* in networks using a connectionless protocol at the network layer. The scheme uses a minimal amount of feedback from the network to the users, who adjust the amount of traffic allowed into the network. The routers in the network detect congestion and set a *congestion-indication* bit on packets flowing in the forward direction. The congestion indication is communicated back to the users through the transport-level acknowledgment. The scheme is distributed, adapts to the dynamic state of the network, converges to the optimal operating point, is quite simple to implement, and has low overhead. The scheme maintains *fairness* in service provided to multiple sources. This paper presents the scheme and the analysis that went into the choice of the various decision mechanisms. We also address the performance of the scheme under transient changes in the network and pathological overload conditions.

Categories and Subject Descriptors: C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design—*network communications*; *distributed networks, packet networks, store and forward networks*; C.2.3 [**Computer Communication Networks**]: Network Operations—*network monitoring*; C.4 [**Computer System Organization**]: Performance of Systems—*design studies*

General Terms: Algorithms, Design, Performance

Additional Key Words and Phrases: Computer communication networks, congestion, congestion avoidance, congestion control, congestion indication, connectionless network layer, fairness, network performance, network power

1. INTRODUCTION

Congestion in computer networks is a significant problem due to the growth of networks and increased link speeds. Flow control and congestion control are problems that have been addressed by several researchers in the past [4]. With the increasing range of speeds of links and the wider use of networks for distributed computing, effective control of the network load is becoming more important. The lack of control may result in congestion loss and, with retransmissions, may ultimately lead to *congestion collapse* [11].

The control mechanisms adopted to control the traffic on computer networks may be categorized into two distinct types: flow control and congestion control.

An earlier version of this paper was presented at the "ACM SIGCOMM 88 Symposium on Communications Architectures and Protocols," Stanford, Calif., August 16–19, 1988.

Authors' address: Distributed Systems Architecture and Performance, Digital Equipment Corporation, Littleton, MA 01460-1289.

End-to-end flow control mechanisms are used to ensure that the logical link has sufficient buffers at the destination. It is thus a "selfish" control function. Control mechanisms for congestion, on the other hand, address the "social" problem of having the various logical links in the network cooperating to avoid/recover from congestion of the intermediate nodes that they share. This paper proposes a mechanism for effective avoidance of congestion in connectionless networks.

We distinguish between *congestion control,* which has been studied in the past [2, 8, 14], and *congestion avoidance.* We call the point at which throughput falls off rapidly before the network reaches congestion collapse as the *cliff.* This is also the point at which the response time approaches infinity. The purpose of a congestion control scheme [2, 14] is to detect the fact that the network has reached the cliff, resulting in packet losses, and to reduce the load so that the network can recover to an uncongested state. Congestion avoidance, on the other hand, operates the network at a highly desirable point, which is at the *knee* of the response time (or delay) curve. This is the point beyond which the increase in throughput is small, but the response time increases rapidly with load. This enables the network to reduce significantly the probability of packet loss and prevents the possibility of serious congestion developing and impacting user performance in the network. A more detailed discussion of the differences is presented in [9].

Congestion control mechanisms that detect whether the network has gone beyond the cliff have been proposed [2, 18]. The feedback that indicates congestion in the network results is the loss of packets and the resulting time-out while waiting for the acknowledgment. Other forms of feedback of congestion information have also been used. An example is to send "choke" or "source quench" packets as part of a congestion control mechanism [1, 13, 14]. There are two differences between these mechanisms for *congestion control* and the scheme we propose in this paper. We propose a *congestion avoidance* policy that drives the operation of the network toward the knee of the delay curve, rather than recovering from operation near the cliff. Second, to achieve this operating point, the network provides explicit feedback by using a field in the packet that flows from source to destination to signal congestion. As such, we do not have additional packets, and therefore, we avoid additional processing and transmission overhead to process these packets in the network. The scheme we propose here is designed so that it is suitable for connectionless network services (as in the Digital Network Architecture (DNA) [3], the ISO Standards [7], and the DOD TCP/IP protocol suite [15]).

The interesting feature of the scheme is the use of a minimal amount of feedback from the network to users (transport entities) to enable them to control the amount of traffic allowed into the network. The routers in the network detect congestion and set a single "congestion indication" bit on packets flowing in the forward direction. This indication for achieving "congestion avoidance" is communicated back to the users through the transport-level acknowledgment. The scheme is distributed, adapts to the dynamic state of the network, converges to the *efficient* operating point, and is quite simple to implement, with low overhead for operation. The scheme also addresses a very important aspect that is not often addressed in studies of congestion control mechanisms. This is the issue of *fairness* in the service provided to the various sources utilizing the network. The scheme attempts to maintain fairness in service provided to multiple sources and attempts to allow the various users of the network an equal share of the network resources.

In the next section we describe the policy for congestion avoidance and provide a summary of the policies at the network routers and the users of the network. In Section 3 we describe a model for studying the congestion avoidance problem in connectionless networks and discuss optimization criteria. We then consider the individual policy decisions in detail. To begin with, in Section 4 we describe the policy of generating the feedback signal to the user. In Section 5 we describe the policies adopted by each of the users to control their window size. Subsequently, we observe the behavior of the scheme under a variety of situations, including transients in the network characteristics, random packet size distributions, etc. Finally, we present conclusions.

2. THE BINARY FEEDBACK SCHEME FOR CONGESTION AVOIDANCE

The scheme for congestion avoidance proposed and studied here is applicable to connectionless networks and a transport protocol using window flow control. As such, it is applicable to networks using protocols such as DNA, ISO (connectionless network service and Transport Class 4), and TCP/IP. The end–end transport protocol uses a sliding window for controlling the number of unacknowledged packets each source may have outstanding in the network. The connectionless network layer uses encapsulation of the higher layer protocol data unit with its own header. A router recalculates the CRC while forwarding the packet, after any modifications to the network layer header [18]. Our challenge while designing the congestion avoidance scheme was to use as small an additional field in the network layer header and as little bandwidth for explicit feedback information as possible. Because of the strict layering and for reasons of not generating additional traffic in a congested environment (which has been studied earlier [13, 14]), we do not send additional packets selectively to sources causing congestion, as done with the source quench packet scheme.

To describe the scheme, let us consider Figure 1, which abstractly shows relevant fields of the data packet flowing from source to destination. The packet may flow over multiple hops, one or more of which may be congested. A router that is congested sets a *congestion indication* bit in the network layer header of a data packet that is flowing in the forward direction. Any router that is not congested ignores the congestion indication bit. When the data packet reaches the destination, the congestion indication bit is copied into the transport layer header of the acknowledgment packet. This acknowledgment packet is then transmitted from the destination to the source.

For our purposes, the *user* is the entity that manages the window of end users in order to transmit traffic. This user copies the bit into an appropriate data structure to be used by the congestion avoidance algorithm. When a packet is originally transmitted from a source, it clears the congestion indication bit. Users are required to adjust the traffic they place on the network, by adjusting their window size, based on their interpretation of the congestion indication from the network. Since one bit is used for the explicit feedback of congestion information, we call the scheme the *explicit binary feedback scheme* for congestion avoidance.

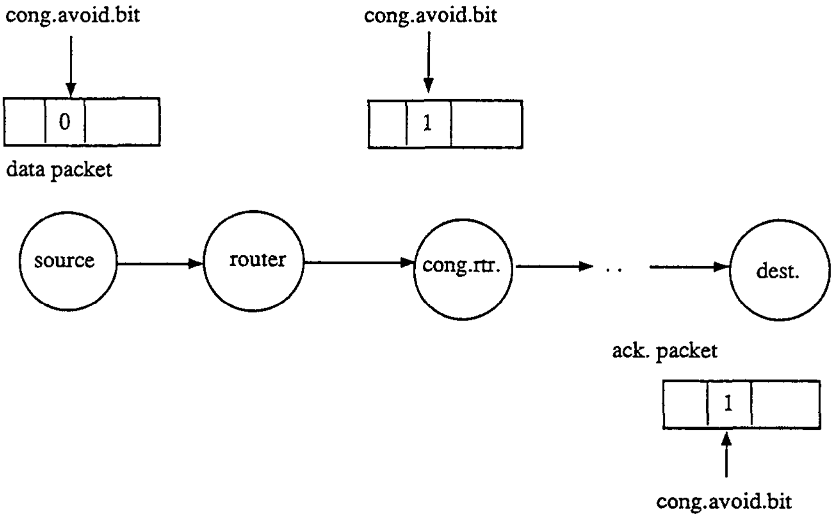


Fig. 1. Information flow in the binary feedback scheme.

There are some differences in the location of the user, based on the network architectures that implement the congestion avoidance scheme. In some network architectures, for example, Digital Network Architecture (DNA), the acknowledgment may move the window forward and also carries the explicit feedback information to the source that controls the window size. Therefore, the user is located at the source generating packets. When multiple packets are acknowledged by the destination using a single acknowledgment, the destination uses the same signal filtering policy as the source. In other architectures (e.g., ISO Transport), the user controlling the window size is at the destination. In this case, there is no need for the communication of the congestion indication bit from the destination to the source.

The feedback control system has two sets of policies for controlling the traffic placed on the network. These are at the network routers (we use the term *router* for the processing entity as well as links) and the users (transport entities) of the network. We summarize the specifics of these policies:

Router Policy

(1) *Congestion detection.* The router sets the congestion avoidance bit in the packet when the average queue length at the router at the time the packet arrives is greater than or equal to one.

(2) *Feedback filter.* The average queue length is determined based on the number of packets in the network router that are queued and in service averaged over an interval T. This interval T is the last (busy + idle) cycle time plus the busy period of the current cycle.

User Policy

(1) *Decision frequency.* The user updates the window size after receiving acknowledgments for a number of packets transmitted. This number is the sum

of the previous window size (W_p) and the current window size (W_c) at which the transport connection is operating. The bits returned in the acknowledgment are stored by the user.

(2) *Use of received information.* Only the bits corresponding to the last W_c packets for which acknowledgments are returned are examined.

(3) *Signal filtering.* If at least 50 percent of the bits examined are set, the window size is reduced from its current value of W_c. Otherwise, the window size is increased.

(4) *Decision function.* When the window size is increased, W_c is increased additively by α. When the window size is decreased, W_c is decreased multiplicatively to $\beta * W_c$. Suggested values for the parameters α and β are 1 and 0.875, respectively.

We address each of these issues in the subsequent sections of this paper.

3. MODEL AND SOLUTION METHODOLOGY

We view the computer network and users as a feedback control system for the purposes of studying the congestion avoidance policies. The feedback signal generation component is achieved by policies used in the network to generate a congestion indication signal back to decision makers (users). The user first filters the signals that are received from the network to determine if the traffic placed on the network should be decreased or may be increased. This binary output from the signal filtering component is used to alter the amount of traffic placed on the network. The decision function component at the user determines the amount of change that is made to the window size of the user so that the overall network operates efficiently.

A wide-area network may span large geographical areas involving considerable communication delay. Therefore, it is infeasible to have a single point in the distributed network for exerting control of the traffic from users, and a distributed control algorithm is required. In the distributed algorithm we propose, each individual user controls the amount of traffic placed on the network, based on the feedback received from the network. Multiple users have to coordinate and cooperate in implementing the congestion avoidance policy. We have approached the analysis of the overall scheme using a detailed simulation, with relevant details of the transport protocol accurately represented. We model the computer network as multiple users generating jobs (packets) in a closed queuing network of servers representing network routers. Some characteristics of the congestion avoidance policy have been studied analytically, wherever possible. One of the first aspects of the policies that we have studied analytically is the determination of the optimal window size given a network configuration. The optimum window size is dependent on the service times of the routes in the path between a source and destination. This work has already been reported in [16].

The instantaneous state of the network (which may be considered to be the queue lengths at the individual servers, such as routers and end-nodes, in the network) varies quite dynamically. Therefore, the communication of feedback information may be subject to considerable noise due to transient effects. Because of imperfect information (noisy or old) at the user, there is a need for filtering of the feedback signal. We achieve this by having two levels of filtering in our model. The first occurs at the point where the feedback signal is generated, to detect congestion; we call this level *feedback filtering.* The second level, which we call *signal filtering,* is the filtering of the feedback signal at the user. The user also adaptively adjusts the frequency of change in the amount of traffic placed on the network so as to allow users to see the effect of the change before making another change. Since feedback delays and correlation between packet arrival times are difficult to represent in an analytical model, we have studied the sensitivity to parameters of the overall policy through simulation. The workload we considered for the purposes of our design is one in which each source is considered to have packets ready to transmit at all times. They are allowed to transmit the packet as long as the transmit window they use (as part of the end–end flow control) allows them to do so. The packet size distribution (and thus the service demand distribution at each node) is allowed to be both deterministic as well as random (exponential, Erlang, uniform, etc.).

The network is represented as a queuing network model. The multiple hops for communication (routers and links) are represented as service centers with queues for packets awaiting service at these nodes. Links that can process multiple packets at a time, as with satellite hops, are represented by an additional delay center accommodating a fixed number of packets for service in a pipelined fashion. We assume that all users generate packets that traverse the same path to the destinations. Details of the simulation model as well as the limitations of the model and assumptions made in the analysis of the congestion avoidance policy have been described in [9].

3.1 Optimization Criteria

The congestion avoidance scheme attempts to operate the network at the knee of the overall response time curve. At this operating point, the response time has not increased substantially because of queuing effects. Furthermore, the incremental throughput gained for applying additional load on the network is small.

We may determine the knee of the delay curve theoretically, given the service times of the individual hops in the path. This is exact in a "balanced" network, for example, when the service times of all the hops are identical. We may obtain this approximately using the balanced job bounds analysis [20] for "unbalanced" networks [16]. We define a function called *Power* at each router. We use this function in order to choose the operating point of the network so that we are at the knee of the delay curve. This is a function that has been studied in considerable detail in the literature [12]. Maximizing power has been proposed as an objective for computer networks [5]. Power at any resource is defined as

$$Power = \frac{Throughput^{\alpha}}{Response\ time}, \qquad \text{where} \quad 0 < \alpha < 1. \tag{1}$$

Note that power has a single maximum [9]. When $\alpha = 1$, the point at which power is maximized is the knee of the delay curve, which is our desired operating point. To use the power at each resource to finally determine the network operating point, we use a function called *Efficiency.* The maximally efficient operating point for the resource is its knee. To compute the efficiency at any other operating point, we need a function that measures the distance of the operating point from the maximally efficient operating point. A desirable characteristic of the function is that the efficiency is 0 percent if throughput is zero (and response time is infinity) and the efficiency is 100 percent at the maximally efficient operating point. The normalized power defined by the ratio of power to its value at the knee satisfies this requirement. The efficiency of a resource's usage is therefore quantified by

$$\begin{aligned} \textit{Resource Efficiency} &= \frac{\textit{Resource power}}{\textit{Resource power at knee}} \\ &= \frac{(\textit{Throughput/knee throughput})}{(\textit{Response time/knee response time})}. \end{aligned}$$

Notice that the resource is used at 100 percent efficiency at the knee, and, as we move away from the knee, the resource is used inefficiently, that is, either underutilized (throughput lower than the knee capacity) or overutilized (higher response time).

The second criterion that is of equal importance in the design of the congestion avoidance policy is fairness across all the users of the network. Informally, the fairness criterion is that all the users of the network receive an equal share of the resources of the network. The fairness of an allocation is a function of the amount of the resource demanded as well as the amount allocated. To simplify the problem, let us first consider the case of equal demands; that is, all users have identical demands, say, D. The maximally fair allocation then consists of equal allocations to all users; that is, $A_i = A$ for all i. The fairness of any other (nonequal) allocation to each of the users is measured by the following fairness function [10]:

$$f = \frac{(\sum_{i=1}^{N} x_i)^2}{(n \sum_{i=1}^{N} x_i^2)} \qquad \text{where} \quad x_i = \frac{A_i}{D}. \tag{2}$$

This function has the property that its value always lies between 0 and 1 and that it is 1 (or 100 percent) for a maximally fair allocation.

We use user throughputs to measure allocations, A_i, because of its additivity property: Total throughput of n users at a single resource is the sum of their individual throughputs. We describe this criterion in more detail in [9]. Given that all the users are using the same path, this fairness goal immediately translates to all the users achieving equal window sizes, W, since

$$W = \frac{\textit{Throughput}}{\textit{Round trip time}}. \tag{3}$$

Thus, the goal for the congestion avoidance mechanism is to use the routers in the network efficiently, while achieving fairness across all the users that are using the network.

3.2 Network Configurations Used for Analysis

Although we have studied the behavior of the binary feedback scheme using a variety of configurations, we use a limited number of configurations in this paper to illustrate the characteristics of the scheme. We have modeled each hop by a single service time, to represent the service time of the bottleneck (link or processing entity). The mean service times at each of the hops in the path are different so that we have a nonhomogeneous path. The first configuration is a network path with four hops. The normalized service time of the individual hops is 2, 5, 3, and 4. The path also contains a satellite to reflect the ability of the scheme to accommodate the long delays in such links as well. The satellite hop is represented by a delay of 62.5 time units. As a result, the satellite hop can accommodate multiple packets in flight at the same time. The maximally efficient aggregate window [16] size for this configuration is 15.5. This is the primary configuration we will use.

To represent an overloaded network, we use the same configuration, but without the satellite hop. This results in the maximally efficient aggregate window size being 3.

4. FEEDBACK SIGNAL GENERATION

In this section we study the policy of feedback signal generation from the network when the routers or the links are congested. The model we use for studying the policy for feedback signal generation is one in which each intermediate point in the network is a single service center with a first-come-first-served queue. Our model may be easily extended to accommodate multiple queues per router if needed. Further, for the purposes of this study, we assume that all the sources of traffic share the same path.

The routers use a feedback signal to indicate congestion. A variety of feedback schemes for flow and congestion control has been proposed in the literature. When the destination is overloaded, explicit feedback mechanisms, such as *ON-OFF* schemes [17, 19] have been suggested. For congestion control at intermediate systems, feedback with source-quench packets [13, 14] has been used. All these schemes involve sending additional packets by a congested system. Our explicit feedback scheme involves the router using the *congestion indication* bit in the network layer header of a packet when it is congested, and does not result in additional packets being transmitted.

We monitor each individual router to detect congestion. We may determine that the router is congested when the utilization reaches a certain level or when the queue length achieves a certain value. The utilization of the router depends on the distribution of the service time of the packets. We model the service time of the packet as a function of the packet size. When the packet size distribution is deterministic, then the router may sustain a utilization of almost 100 percent before any performance degradation is seen. When there is considerable variance in the packet size distribution, then the response time degrades even when the utilization is greater than 50 percent and is therefore no longer a good estimate of congestion of the router. The average queue length may also be used to reflect congestion of the router. When the average queue length becomes larger, we know that the delay becomes larger, even though throughput does not increase significantly. The average queue length is less sensitive to the distribution of the service time, since response time degrades with increasing average queue lengths independent of the service time distribution. Therefore, we monitor the average queue length at the router (including the packet currently in service) to detect congestion.

The various algorithms for generating the feedback signal, based on the queue lengths of packets to be forwarded, may be categorized as being either a simple thresholding policy or a hysteresis policy. Consider the case of a single router in isolation, associated with a queue of packets to forward. Two thresholds, T_1 and T_2 ($T_1 \leq T_2$), are defined for the size of the queue. The simple thresholding algorithm is to generate the feedback signal when the queue size is above a threshold, say, T_2. The *hysteresis* algorithm used to generate the feedback signal is slightly more complex. The hysteresis algorithm indicates congestion when the queue size is increasing and crosses a threshold value, for instance, T_2. The feedback signal continues to indicate congestion as the queue size decreases, till it reaches the smaller threshold value T_1.

Hysteresis has been proposed as a scheme to reduce the amount of communication of signals when switching across a single threshold [6]. Using hysteresis at the signal generation point minimizes the overhead of sending congestion *on* and *off* signals. When the explicit feedback signal is transmitted as a separate packet to the sources generating congestion, it may result in additional congestion. Note that with the binary feedback scheme the generation and communication of the feedback signal itself do not consume any significant additional resources, both of the CPU as well as of the link.

We have studied the policy for setting the congestion indication bit by observing the behavior of *global power* by simulation. Multiple users share the path, comprising multiple routers. We considered the policies of using a single threshold as well as hysteresis to set the bit. Figure 2 shows the variation of power with the hysteresis value used by the router to set the bit. The representation for the hysteresis algorithm used in the figure specifies the center C of the range of the hysteresis and also the width of the hysteresis, H. Thus, $T_1 = C - H/2$ and $T_2 = C + H/2$. This representation can therefore be used for both the hysteresis and the single-threshold policies. We find that the power is maximum when the hysteresis is nonexistent, with a threshold value of 1. We have observed this behavior for both deterministic as well as random service times at the individual service centers (i.e., deterministic as well as random packet sizes). Therefore, the algorithm we use for signal generation by the router is for it to set the congestion indication bit on an arriving packet when the average number of packets at the router (including the one in service) is greater than or equal to 1.

4.1 Feedback Filter

To ensure that we operate at the correct point, we do not use the instantaneous queue sizes at the router, but instead use the average queue size. There are several reasons for this. First, there may be significant delays for feeding back congestion information to the users. The effect of such a feedback delay is that the congestion signal generated at a router may not be relevant when it reaches the users. Using the instantaneous queue lengths may also result in signaling congestion prematurely, by reacting to transient changes, and thus potentially increasing the idle time of the router. Second, using the instantaneous queue sizes may result in some users having their bits set and some other users not having their bits set. This results in unfairness in the signals sent to the individual users and, as a result, the throughput they receive. Therefore, we need a low-pass filter function

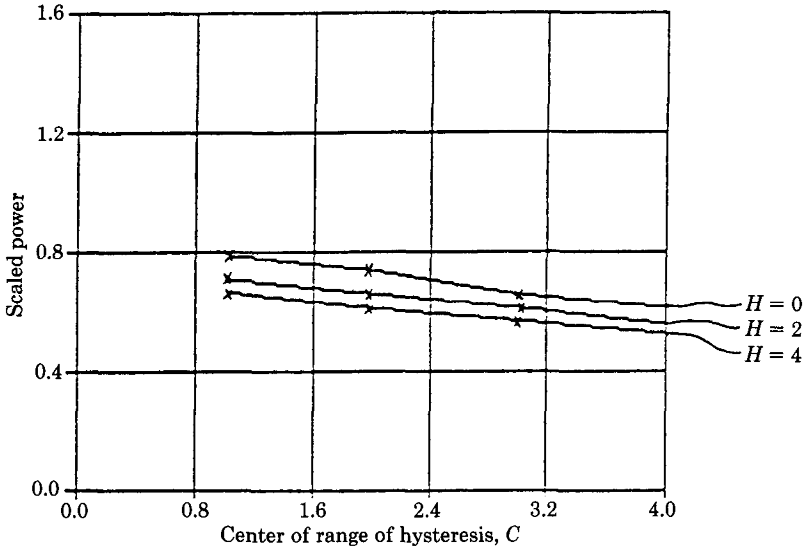


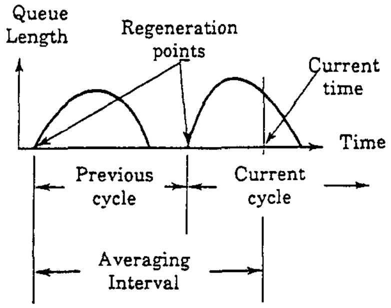
Fig. 2. Behavior of power with hysteresis.

to pass only those states of the routers that are expected to last long enough for the user action to be meaningful. We achieve this by setting the bit on packets flowing through a router that is considered "congested" only when its average queue length is above the threshold of 1. This average queue length includes the packet in service.

To provide a consistent state of the router, the router needs to use the properly computed value of the average queue length rather than the instantaneous queue length to set the congestion indication bit. Several filtering techniques for the feedback signal at the router were attempted. We attempted to use the average over a fixed interval of time, attractive because of its simplicity. We examined the network behavior with different values for the averaging interval. We found that the signals generated to the users are consistent and result in a fair allocation of the router's resources when the interval is close to the round-trip delay from the users. When the interval is different, we found that the inconsistency in signals to the users increased. We then used a weighted exponential running average of the queue length. This too showed (to a different degree) the problem of having inconsistent signals to the users. This was because the exponential average estimates the queue length over an interval, and when the interval was further off from the round-trip time, the inconsistency arose once again. This indicated a need for an adaptive averaging algorithm, which we describe below.

The adaptive averaging, in effect, determines the busy/idle periods adaptively at the router. This is the *regeneration cycle* at each individual router. Each router adapts to the characteristics of the load placed on it by its users. A regeneration cycle is defined as a *busy* + *idle* interval seen at the router. The beginning of the busy period is called a *regeneration point*. The regeneration point signifies the birth of a new cycle for the evolution of system behavior since the queuing

Fig. 3. A regeneration cycle at the router.



system's behavior after the regeneration point does not depend on the state before it. The average queue length is given by the area under the curve divided by the total time for the regeneration cycle. This average can be used for generating feedback signals for the entire duration of the next cycle.

We adopt some refinements to account for the case where a regeneration cycle may be very long. For example, when the busy period is very long, we need to be able to reflect a more current average queue length than the last cycle's average. The feedback based on the previous cycle may not reflect the current situation. The refinement is achieved by basing the queue average on the previous cycle as well as on the current cycle, which may be incomplete. This is shown in Figure 3, which reflects a hypothetical behavior of the router queue length. The queue average is computed by considering the integral (area under the curve) of the queue length since the beginning of the previous cycle. The averaging is now performed as each packet arrives at the router. Thus, as the length of the current cycle gets longer, the contribution to the average by the current cycle begins to dominate, and the effect of the previous cycle begins to decay. This adaptive averaging generates a consistent signal of congestion to all the users and is seen to work satisfactorily even with a large number of users of the path. The results presented in the subsequent sections are based on this adaptive averaging algorithm for congestion detection at the routers.

5. POLICIES FOR DECISION MAKING

The feedback signals received from the network by the user are used to control the window size. There are several components to the decision-making policy. These are

—decision frequency,

—use of received information,

—signal filtering, and

—increase/decrease algorithms.

The frequency of decision making determines the period between updates to the window size. This may possibly be in terms of the number of packets received or packets for which acknowledgments have been received, depending on the location of the "user." The second component determines the number of congestion indication bits received at the user that are examined to determine the update to

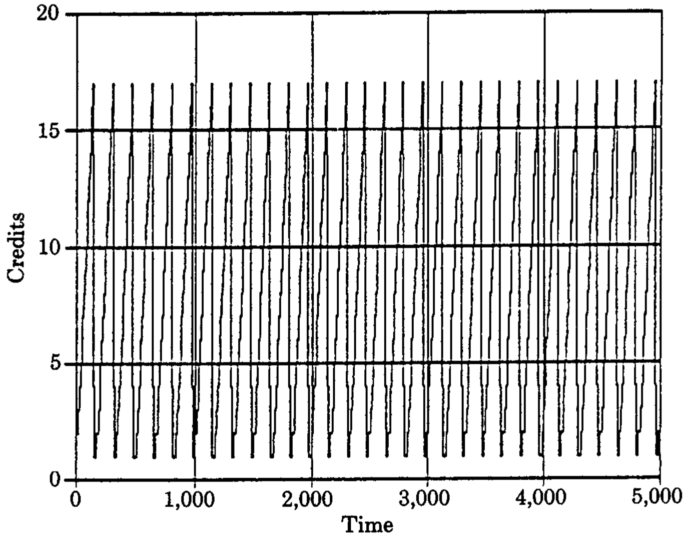


Fig. 4. Behavior of window size updated after every acknowledgment.

the window size. The signal filtering component is used to filter the noise that may be in the congestion indication signals received at the user. The increase/ decrease algorithms determine the extent of change to the current window size at each update. We describe each of the components in the following sections.

5.1 Decision Frequency

The first issue that arises is the frequency of decision making. Let us assume the "user" is the source station. Our initial approach is to make a decision at the instant each acknowledgment is received. We assume that there is no acknowledgment accumulation and that the destination acknowledges every packet that is received. Upon receiving an acknowledgment, the source may determine whether to increase or decrease the window. If the user determines the new value of the window size based on the current signal, the effect of the change to the window size takes a certain amount of time before it alters the state of the network. The state of the routers (uncongested or congested) is aggregated and represented by the single bit received at the user for every packet. It takes several packets to be received at the user before the effect of the change in window size is reflected in the congestion indication bits being fed back to the user. When the window size is altered on every packet, it prematurely alters the window size, causing overcorrection. Altering the window size after every acknowledgment causes considerable oscillation in the user's window size, as shown in Figure 4.

Given a change in the window size to W_c by the source, it is only the first packet in the next window cycle (the $(W_c + 1)$th packet) that will bring a network feedback corresponding to window W_c. A general control function may require us to remember a long history of bits received at the user. A simple control policy is obtained if we keep the window constant for two cycles [9]. Our approach, therefore, has been to introduce a waiting period after every window size update,

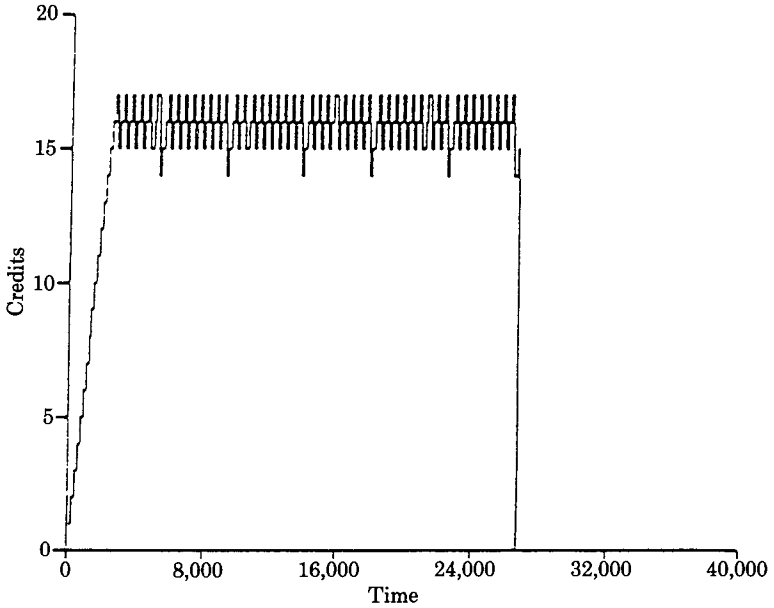


Fig. 5. Behavior of window size updated every two window sizes.

before the next update is performed. If W_p = window size before the update, and W_c = window size after the update, we wait for $(W_p + W_c)$ packets to be acknowledged. Some of these acknowledgments would correspond to those for the window size W_c. If we are operating close to the optimal window size, then W_c of these acknowledgments would be for exactly W_c packets sent with the new window. Figure 5 shows the behavior of the window size with the frequency of update being once every $W_p + W_c$ acknowledgments. The oscillation of the window size is now considerably reduced.

5.2 Use of Received Information

The next issue is whether the user maintains information (the bits returned in the acknowledgments) after a decision is made. We make the observation that maintaining congestion indication bits used in the previous decision causes overcorrection, if we use a simple filtering algorithm at the user. Using arbitrary amounts of past history can result in the history, say, of congestion encountered dominating the decisions for a long period, which may be longer than the duration during which the network was congested. Therefore, an appropriate amount of old information should be erased after a window size update. In fact, we propose discarding any of the history that is maintained in the network itself, after a decision is made to alter the window. There would typically be packets in the network transmitted by the user at the previous window size whose acknowledgments would be received after an update. Since we update the window after every $W_p + W_c$ acknowledgments, we receive $W_p + W_c$ congestion indication bits between updates. Of these, W_p bits correspond to the packets transmitted with the previous window size. We ignore these when we decide to examine the received bits to update the window. We examine only the last W_c bits in order to update the window size. This is primarily to keep the policy simple and is motivated by a desire to avoid any additional state being maintained to relate the bits received to the appropriate window size.

5.3 Signal Filtering

We have one bit of information fed back from the router for each packet that is transmitted by the user. The user needs to filter any noise in the signals received between successive decision points. We call this *signal filtering.* The output of the filter initiates a change in the window size used by the user.

In general, the filtering performed at the decision maker on the bits that are received by the source, with varying information content (bit set or not set), may be used to provide different types of information to the increase/decrease algorithms. For instance, we may have the filter specify only the direction of the change (either increase or decrease) by using a single cutoff value for the determination of the output of the filter. In the general case, the filter algorithm may be such that it not only provides the direction of the change in the window size required, but also the relative magnitude of such a change. If, in the general case, there are n cutoff values, then the filter may specify that the source increase its window size by increasing amounts, based on the percentage of the received bits that are being set below cutoff factors 1, 2, ..., n. Cutoff factors may also be used to indicate a reduction in the window size similar to those used for an increase in the window size. The consequence of using a larger number of cutoff factors results in greater complexity of the increase/decrease algorithms.

The algorithm that we have adopted uses a single cutoff factor for the filtering of the signal at the decision maker. The primary motivation is for simplicity of this component of the decision-making policy. The value of the cutoff factor is dependent on the policy used by the routers in the network to set the bit to indicate the existence of congestion and also the distribution of the service times at the router. This may be shown by considering a simple example. Consider the case of the bottleneck resource in isolation. To start with, assume that the interarrival and service times at that router are exponentially distributed. The value of ρ, at which power at the router is maximized, is 0.5 [12]. Using this value of power, we can consider the probability of having the congestion indication bit set for different values of the threshold at the router. Let C = the threshold at which the congestion indication bit is set. This is the average value of the number of packets at the router, at which the congestion avoidance policy sets the bit.

Let $P(n)$ = the steady-state probability of n packets at the router, including the one in service.

$$\textit{Probability}(\textit{bit set by the router}) = 1 - (P(0) + P(1) + \cdots + P(C-1));$$

when $C = 1$,

$$\textit{Prob}(\textit{bit set}) = 1 - P(0) = \rho = 0.5,$$

and when $C = 2$,

$$\textit{Prob}(\textit{bit set}) = 0.25.$$

Thus, when the threshold, C, at which the congestion indication bit being set by the router is 1, then the percentage of bits that are set by the router is equal to 50 percent. However, when the interarrival and service times are deterministic, power is maximized when the utilization $\rho = 1$. With a threshold C of 1, *Prob*(*bit set*) = 1.0. But, when $\rho < 1$, for $C \geq 1$, *Prob*(*bit set*) = 0.0. Thus, using a threshold C equal to 1, *Prob*(*bit set*) ≥ 0.5, when maximizing power for both the M/M/1 and D/D/1 cases.

Thus, the relationship between C and the probability of receiving the congestion avoidance bit set is dependent on the service time distribution at the routers in the network and the size of the threshold C at the router. The service time distribution depends on the packet size distribution, since we model the service time at each of the service centers in the queuing network model as a function of the packet size. Figure 6 shows the variation of the value of global power with the variation of the cutoff factor for the signal filter, for various values of the router threshold C for a network with multiple nodes. The users follow the increase/decrease algorithm described in the next section. When the router threshold C is 1, power is maximum when the cutoff factor for the percentage of received bits being set is equal to 50 percent. When C is 2, power is maximum when the cutoff factor is equal to 25 percent. We have used a value of $C = 1$ and a cutoff factor value of 50 percent. We observe in fact that for $C = 1$ the curve is relatively flat. This is desirable as it reflects little sensitivity to this parameter, in keeping with our goal of the parameters of the scheme being applicable over a broad range of network characteristics.

5.4 Increase/Decrease Algorithms for the Window Size

When all the users share the same path, the algorithms followed by the routers in the network ensure that all users receive the same signal of congestion from the network. The signal filter at the user provides a binary signal to increase or decrease the window size. Here, we present some justification for and primarily the results of using the additive increase/multiplicative decrease decision function discussed in [9].

The user must consider the following criteria for the decision function:

—maintain the overall global window size as close to the maximally efficient value as possible,

—maintain fairness across multiple sources,

—minimize oscillations in the window sizes, and

—minimize the time to achieve steady state.

Some of these criteria are quantified by defining, for the individual window sizes, a fairness measure defined in [10], as well as our global power metric.

To begin with, consider the simple *additive increase/additive decrease* function. This decision function is described by the following equations: If W_p^i = window size after the previous decision epoch of source i and W_c^i = window size after the current decision at source i,

Additive increase and additive decrease (*Algorithm A*)

Increase: $W_c^i = W_p^i + b, \quad b \geq 0;$
Decrease: $W_c^i = W_p^i - d, \quad d \geq 0.$

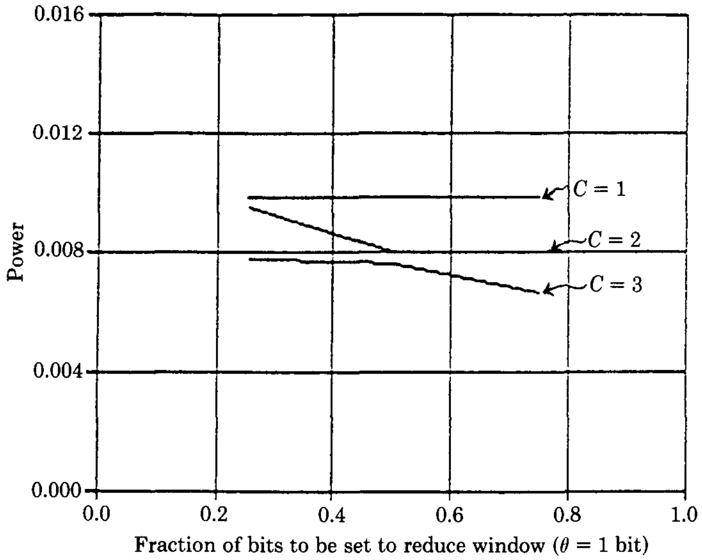


Fig. 6. Behavior of power with varying cutoff values.

We find that Algorithm A is unfair. This is because the state of unfairness of the system (e.g., when one of the sources is at a lower window size than another) is preserved by the additive increase and the additive decrease functions. This is shown in Figure 7 for our primary configuration with normalized service times of 2, 5, 3, and 4, and including a satellite hop (optimal window $W_{knee} = 15.5$). The unfairness arises from the fact that all the participating sources increase or decrease by equal amounts.

In [9] we provide the justification for considering decision functions that alter the window size proportional to the current window size. We call this a *multiplicative algorithm*. This algorithm was argued as being fair. It may be represented as follows:

Additive increase and multiplicative decrease (Algorithm B)

Increase: $W_c^i = W_p^i + b, \quad b \geq 0;$
Decrease: $W_c^i = dW_p^i, \quad 0 < d \leq 1.$

We show in Figure 8 that we can achieve fairness by using Algorithm B for the primary configuration. As described in [9], although the control placed on the network is discrete since the window sizes are integer values, we maintain the window sizes as real values at the individual sources. The actual window size, which is the number of packets that may be outstanding in the network, is obtained by rounding the real value of the window size to the nearest integer value.

If only integer values are maintained for the window, *additive increase* and *multiplicative decrease* may also stabilize to unfair values, although this may not be the case for all values of increase amounts and decrease factors. Consider an example where two users increase additively by 1 and decrease multiplicatively by a factor of 0.8 and the optimal window size 15.5. If the two users start at

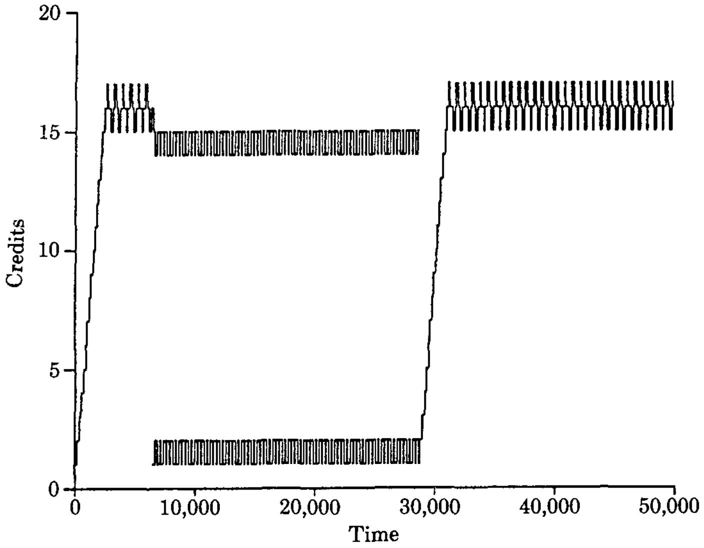


Fig. 7. Behavior of window size with additive increase/decrease.

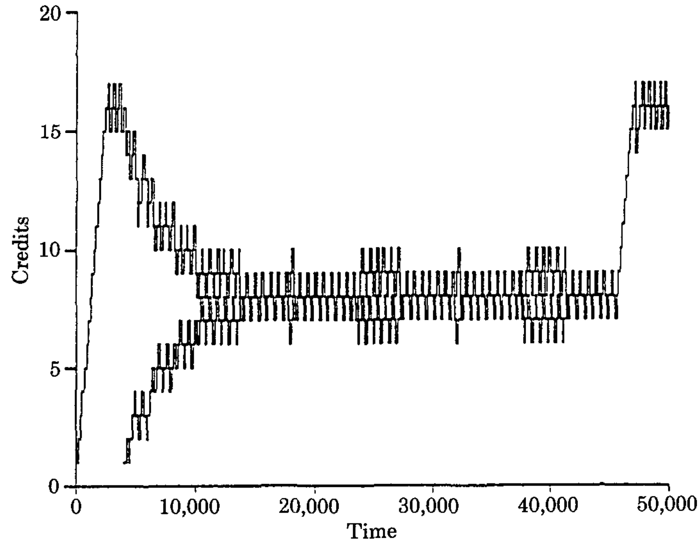


Fig. 8. Behavior of window size with additive increase/multiplicative decrease.

different times, they may reach a stable point such that user 1 has a window of 10 and user 2 has a window of 6. The sum is more than $W_{knee} = 15.5$, and therefore both users are asked to reduce. They come down (say, using a factor of 0.8) to 8 and 4 (after truncation). The total window is less than W_{knee}, and hence, both users are allowed to increase. They go up by 1 to 9 and 5. The total window is still less than W_{knee}, and the users then go up to 10 and 6. After this, the cycle repeats. For the same configuration, when real values are used, we find that the allocations to the users are fair, as shown in Figure 8. By exhaustively searching the parameter space, we verified the fairness of the additive increase and multiplicative decrease algorithm when the implemented window size is obtained by rounding the computed window. We found that, generally, single precision floating-point representation of the window is adequate.

There are several other considerations that influence the choice of parameters for the increase factor and the decrease factor (b and d for Algorithm B, respectively). By using a small decrease factor (e.g., reducing the window size to 50 percent of the current value, compared to 90 percent of the current value), we achieve fairness more rapidly, compared to a larger decrease factor. On the other hand, once convergence of the two window sizes is reached, we would like to minimize the oscillations around the maximally efficient window size as much as possible. This is achieved by using as large a multiplicative factor for the decrease of the window size as possible. We chose to give precedence to minimizing the oscillations once the system has reached the point of maximum efficiency. Although the amount of time that it takes to reach a fair value may be longer, we find that the reduced oscillations improve the throughput because of the increase/decrease policies. We choose a value of 0.875 ($\frac{7}{8}$) for the decrease factor based on the ease of implementation, while minimizing oscillations.

The overall scheme is also efficient. Consider Figure 8, where each of the two users is sending 5,000 packets through the configuration with a bottleneck router whose per-packet service time is 5 units. The time to complete sending the total work load is approximately 50,000 time units, indicating that the binary feedback scheme is efficient.

6. TESTING THE BINARY FEEDBACK SCHEME

In this section we discuss the behavior of the binary feedback scheme for several of the conditions outlined in [9] as essential for an acceptable scheme. We have already seen the capability of the scheme to operate at the maximally efficient point and to be fair across multiple users.

6.1 Behavior with Random Packet Size Distributions

Figure 9 shows the behavior of the window size of a single source with exponentially distributed packet sizes. Recall that the mean service times at each of the network routers in the path are different so that we have a nonhomogeneous path. The maximally efficient aggregate window size is 15.5. The average aggregate window size of the source in this experiment was 14.3. We find that the dynamic behavior of the window size is reasonable.

6.2 Behavior of Scheme under Transients

Any scheme that is proposed for control of congestion in the network must exhibit good response to transient changes in the network. We consider two types of transients in the network: in the first, additional users enter a network that is already operating, injecting additional traffic; in the second, changes in the network (such as topology changes) result in the service times of packets being

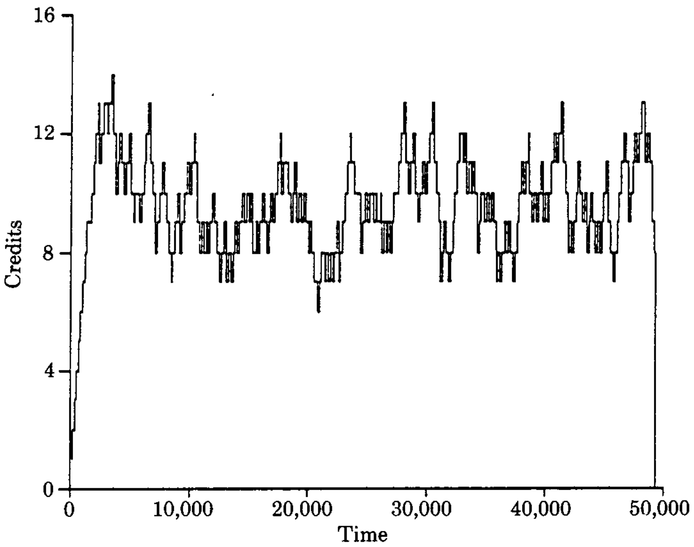


Fig. 9. Behavior of window size with exponential packet size distribution.

different during the transient. We have simulated the latter situation by having a transient change in the service time of the bottleneck router, which would result in the optimal window size, W_{knee}, changing during this transient period.

Figure 10 shows the behavior of the overall window size when the service time of the router changes to double its initial value after the network has achieved a steady operating point. After a small initial undershoot, the overall window size recovers, and the network operates at its new maximally efficient point. When the router's service time once again goes back to its initial value (possibly simulating recovery of the original lower cost path), the overall window size goes back to the original operating point. The amount of time taken to recover to the original operating point is minimal, as shown in Figure 10.

The other transient condition that we typically see in a network is the injection of an additional load by users who start up at arbitrary points in time. The users who are already on the network are operating at the efficient window size. Thus, when a new user arrives into the system, the resources of the network are shared between the two users. In Figure 8 we show the case where two users start at different times in the network while sharing the same path. We found that, ultimately, the two sources converge to a fair value so that their window sizes are nearly equal.

6.3 Behavior of the Scheme under Special Cases

Often, a scheme designed to work in a configuration and workload that is typical may not work satisfactorily (in terms of efficiency or fairness) when the dynamic range for the algorithms is exceeded. In the following subsections, we illustrate the robustness of the binary feedback scheme to such special circumstances.

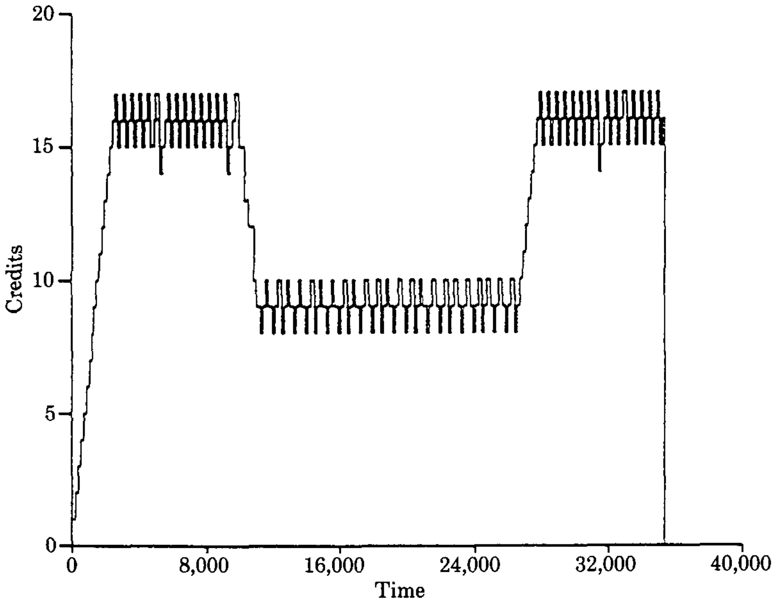


Fig. 10. Behavior of window size with transient in bottleneck service time.

6.3.1 *Starting Users at Arbitrary Initial Values for the Window Size.* Figure 11 shows the dynamic behavior of the overall window size with one user who initially starts at a large window size. By setting the congestion indication bit on the packets sent by this user, the network brings down the window size to the correct value so that the network is operating at the desired operating point of the knee in a relatively small period of time. Thus, even if the users of the network actually start at any arbitrary window size, the congestion avoidance policy adapts dynamically and adjusts the window size to its correct value.

6.3.2 *Source Bound Network.* It is important that the window size not increase arbitrarily when the source is the bottleneck. The network, because it is not congested in such a case, does not set the congestion indication bit. Therefore, the window size could increase to a large value, far beyond the efficient value, if the processing time at the source is also taken into account. To avoid this particular condition, we include as part of the user's policy, a check to see if the window size increased in the previous step. We ensure that the window size is not increased if the source has not been able to implement an increase of the previous epoch. Figure 12 shows the variation of the window size over time with a configuration that is source bound. The increase in the window size is limited by the source.

6.3.3 *Behavior in an Overloaded Network.* We test the binary feedback avoidance mechanism in an *overloaded* network using the same configuration as before, except that we do not have the satellite hop. The maximally efficient aggregate window size is 3. In this situation the number of users placing a load on the

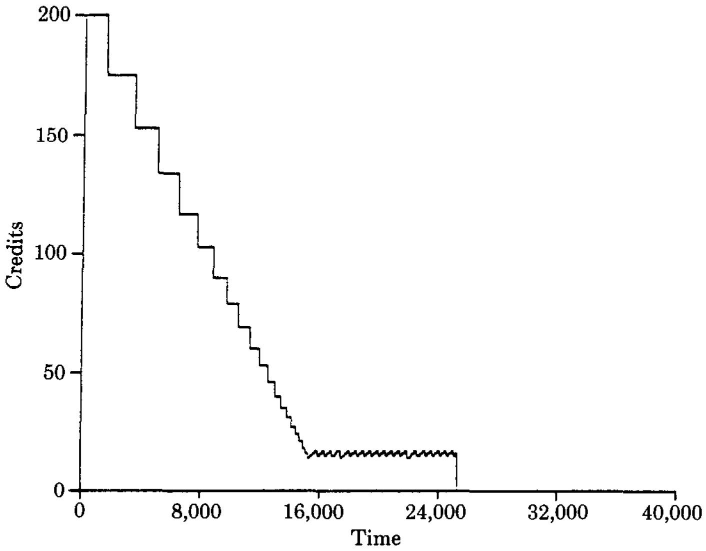


Fig. 11. Behavior of window size when starting at an arbitrary large value.

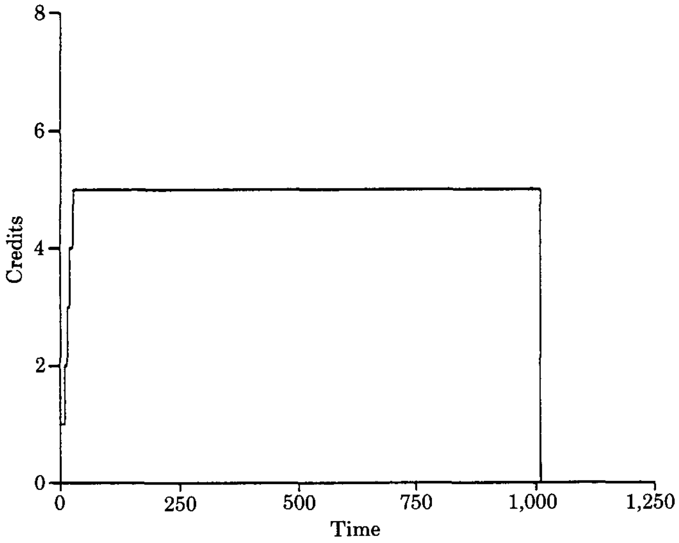


Fig. 12. Behavior of window size when source bound.

network is greater than the optimal window size. This results in a situation where the window size of each of the sources is limited to one, which is the minimum for each user's window. Figure 13 shows the case where there are nine sources that begin to use the network, arriving one after the other. The figure shows the overall aggregate window size, as well as the individual window sizes. The aggregate window initially oscillates because of the relatively fewer number of

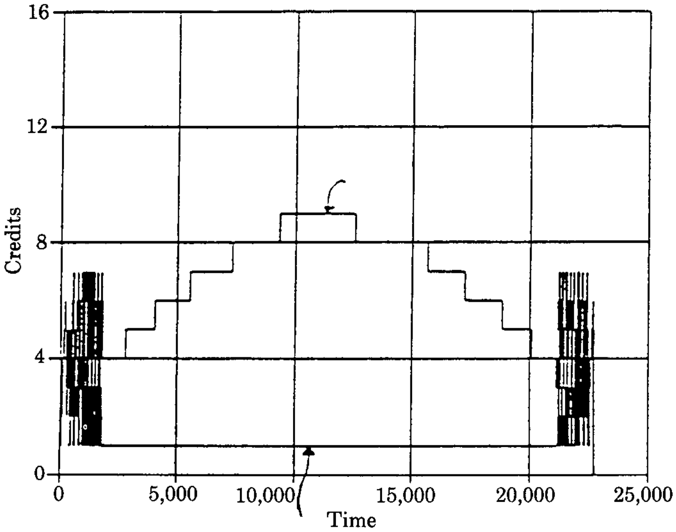


Fig. 13. Behavior of window size in an overloaded network.

users on the network to start with. As the number of users gets to be larger than the maximally efficient window size, the overall aggregate window size is flat, since all of the users see the congestion avoidance bit set and therefore are all limited to a window of 1. As each of the sources complete, the aggregate window size changes accordingly, till we see a sufficiently small number of sources and the oscillation of the window size begins again. This demonstrates the stability of the scheme, even under the situation of an overloaded network, with the individual window sizes limited to 1.

7. CONCLUSIONS

In this paper we have proposed a scheme for congestion avoidance for networks using connectionless protocols at the network layer. The scheme uses a minimal amount of feedback from the network, with just one bit in the network layer header to indicate congestion. Each network server that is congested (routers or links) sets the congestion indication bit (if it is not already set). This information is then returned to the user by the destination that receives the packet. This feedback information is examined by the user to control the amount of traffic that is placed on the network. We modeled the network as a feedback control system and identified the various components of the scheme in the context of such a model. We studied the policies that need to be used in each of these components through analysis, as well as through simulation.

The routers detect their state as being congested and set the congestion indication bit when the average queue length is greater than or equal to one. We described the averaging algorithm at the server, which is based on the busy + idle regeneration cycle time seen at the server. The users (source or destination, based on the network architecture) receive these bits and determine the correct window size to use. The update to the window size is performed when the number of bits received is the sum of the previous window (W_p) and the current window size (W_c). The last W_c bits are examined by a signal filter at the user. When at least 50 percent of these bits are set, the window size is reduced from its current value of W_c to 87.5 percent of its value. Otherwise, it is increased by 1. We showed that these policies result in an effective congestion avoidance scheme.

We have also shown that the scheme is distributed, adapts to the dynamic state of the network, converges to the efficient operating point, and is quite simple to implement, with low overhead while operational. We have addressed the important issue of fairness in the service provided to the various users of the network. The scheme maintains fairness in service provided to multiple users, independent of their starting points or their initial window sizes.

We have addressed the performance of the scheme under transient changes in the network. We have also ensured that the scheme operates the network at a stable point when the network is overloaded, when users start at arbitrary initial values for their window size, and when the source is a bottleneck in the path.

ACKNOWLEDGMENTS

We would like to thank the members of the Distributed Systems Architecture and Performance group for participating in discussions and for contributing in the refinement of the ideas presented here. In particular, we would like to thank Tony Lauck, Linda Wright, and Bill Hawe for their feedback and their encouragement throughout this work. We also thank Radia Perlman, Art Harvey, Kevin Miles, and Mike Shand for their support in incorporating the mechanisms proposed here in architectures they were responsible for. We also thank Dah-Ming Chiu for his close cooperation during this work and in refining it.

REFERENCES

1. AHUJA, V. Routing and flow control in systems network architecture. *IBM Syst. J. 18*, 2 (1979), 293–314.
2. BUX, W., AND GRILLO, D. Flow control in local-area networks of interconnected token rings. *IEEE Trans. Commun. COM-33*, 10 (Oct. 1985), 1058–1066.
3. DIGITAL EQUIPMENT CORPORATION. DECnet digital network architecture (phase IV) general description. Order AA-N149A-TC, Digital Equipment Corporation, Maynard, Mass., 1982.
4. GERLA, M., AND KLEINROCK, L. Flow control: A comparative survey. *IEEE Trans. Commun. COM-28*, 4 (Apr. 1980), 553–574.
5. GIESSLER, A., HAANLE, J., KONIG, A., AND PADE, E. Free buffer allocation—An investigation by simulation. *Comput. Networks 1*, 3 (July 1978), 191–204.
6. HARRISON, P. G. An analytic model for flow control schemes in communication network nodes. *IEEE Trans. Commun. COM-32*, 9 (Sept. 1984), 1013–1019.
7. INTERNATIONAL ORGANIZATION FOR STANDARDIZATION. ISO 8073: Information processing systems—Open systems interconnection—Connection oriented transport protocol specification. ISO 8073-1986 (E), International Organization for Standardization, July 1986.
8. JAIN, R. A timeout-based congestion control scheme for window flow-controlled networks. *IEEE J. Sel. Areas Commun. SAC-4*, 7 (Oct. 1986), 1162–1167.
9. JAIN, R., AND RAMAKRISHNAN, K. K. Congestion avoidance in computer networks with a connectionless network layer, part I—Concepts, goals and alternatives. DEC Tech. Rep. TR-507, Digital Equipment Corporation, Littleton, Mass., Apr. 1987.
10. JAIN, R. K., CHIU, D.-M., AND HAWE, W. R. A quantitative measure of fairness and discrimination for resource allocation in shared systems. DEC Tech. Rep. TR-301, Digital Equipment Corporation, Littleton, Mass., Sept. 1984.

11. KLEINROCK, L. On flow control in computer networks. In *Proceedings of the International Conference on Communications* (June 1978), pp. 27.2.1–27.2.5.
12. KLEINROCK, L. Power and deterministic rules of thumb for probabilistic problems in computer communications. In *Proceedings of the International Conference on Communications* (June 1979), pp. 43.1.1–43.1.10.
13. MAJITHIA, J. C., IRLAND, M., GRANGE, J. L., COHEN, N., AND O'DONELL, C. Experiments in congestion control techniques. In *Proceedings of the International Symposium on Flow Control in Computer Networks* (Feb. 1979), pp. 211–234.
14. NAGLE, J. Congestion control in IP/TCP internetworks. *Comput. Commun. Rev. 14*, 4 (Oct. 1984), 11–17.
15. POSTEL, J. B. Transmission control protocol. Tech. Rep. RFC 793, Information Sciences Institute, Sept. 1981.
16. RAMAKRISHNAN, K. K. Analysis of a dynamic window congestion control protocol in heterogeneous environments including satellite links. In *Proceedings of the Computer Networking Symposium* (Nov. 1986). IEEE, New York, 1986, pp. 94–101.
17. REISER, M. Queueing and delay analysis of a buffer pool with resume level. In *Performance '83, Proceedings of the 9th International Symposium on Computer Performance Modelling, Measurement, and Evaluation* (College Park, Md., May 1983). A. K. Agrawala and S. K. Tripathi, Eds. North-Holland, New York, 1983, pp. 25–32.
18. TANENBAUM, A. S. *Computer Networks.* Prentice-Hall, Englewood Cliffs, N.J., 1981.
19. YUM, T. P., AND YEN, H.-M. Design algorithm for a hysteresis buffer congestion control strategy. In *Proceedings of the IEEE International Conference on Communications* (June 1983). IEEE, New York, 1983, pp. 499–503.
20. ZAHORJAN, J., SEVCIK, K. C., EAGER, D. L., AND GALLER, B. Balanced job bound analysis of queueing networks. *Commun. ACM 25*, 2 (Feb. 1982), 134–141.

Received October 1988; revised October 1989; accepted October 1989