A Binary Feedback Scheme for Congestion Avoidance in Computer Networks with a Connectionless Network Layer

K. K. Ramakrishnan and Raj Jain
Distributed Systems Architecture and Performance
Digital Equipment Corporation

Abstract

We propose a scheme for *congestion avoidance* in networks using a connectionless protocol at the network layer. The scheme uses feedback from the network to the users of the network. The interesting challenge for the scheme is to use a minimal amount of feedback (one bit in each packet) from the network to adjust the amount of traffic allowed into the network. The servers 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 acknowledgement.

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 while operational. The scheme also addresses a very important aspect 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.

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 for pathological conditions.

0

1 Introduction

Congestion in computer networks is a significant problem due to the growth of networks and increased link speeds. Flow and congestion control are problems that have been addressed by several researchers in the past [GK80]. 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* [Kle78].

The control mechanisms adopted to control the traffic on computer networks may be categorized into two distinct types: flow control and congestion control. 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 congestion of the intermediate nodes that they share. This paper proposes a mechanism for effective control in connectionless networks.

We distinguish between *congestion control*, which has been studied in the past [BG85], [Nag84], [Jai86], and *congestion avoidance*. Congestion avoidance operates the network at the *knee* of the response time curve. This is the point at which the increase in throughput is small, while the response time increases rapidly with load. This enables the network to significantly reduce the probability of packet loss and preventing the possibility of serious congestion developing and impacting user performance in the network. A more detailed discussion of the differences is made in [JR88].

The congestion avoidance policy we propose here drives the operation of the network toward the knee of the delay curve. To achieve this operating point, the network provides some type of feedback so that the users may control the amount of traffic they place on the network. Congestion control mechanisms have been proposed that detect whether the network has gone beyond the *cliff* [Jai86], [BG85]. The feedback indicating congestion in the network 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 to control congestion [Nag84], [Ahu79], [ea79].

The scheme we propose here is designed so that it is suitable for connectionless network services (as in the Digital Network Architecture (DNA) [DNA82] and the use of a connectionless network layer by transport protocols defined by the ISO Standards [ISO86]). The scheme explicitly feeds back congestion information to the sources of congestion. There are two differences between the feedback mechanism for congestion control using source quench or choke packets and the scheme proposed here. First, we use a field in the packet flowing in the forward direction to signal congestion. As such, we do not have additional packets and therefore avoid additional processing and transmission overhead to process these packets in the network. Second, we use this feedback to achieve *congestion avoidance* rather than *congestion control.*

The interesting feature of the scheme is the use of a minimal amount of feedback from the network to adjust the amount of traffic allowed into the network. The routers in the network detect congestion and set a single 'congestion avoidance' bit on packets flowing in the forward direction. This 'congestion avoidance' 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 *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 will 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 decision maker. In Section 5, we describe the policies that are used at the decision maker to control the window size used by each user of the network. Subsequently, we observe the behavior of the scheme with 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 being proposed and studied here is applicable for connectionless networks using a virtual circuit oriented transport protocol. As such, it is applicable for 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 header and recalculates the CRC while forwarding the packet [Tan81]. Our challenge while designing the congestion avoidance scheme was to use as few additional fields 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 [ea79], [Nag84]), 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

Figure 1: Block Diagram of Bit Scheme

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.

We call the entity that manages the window of end-users to transmit traffic as the *user* for our purposes. 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 user, it clears the congestion indication bit. Users are required to adjust the traffic they place on the network based on their interpretation of the congestion indication from the network. They achieve this by adjusting their window size. Since one bit is used for the explicit feedback of congestion information, we call the scheme the **The Explicit Binary Feedback Scheme** for congestion avoidance.

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, e.g., Digital Network Architecture (DNA), the acknowledgment may move the window forward and also carry the explicit feedback information to the user. Therefore, the user is located at the source generating the packets. 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 routers as well as links) and the users (transport entities) of the network. We summarize the specifics of the policies at the network routers and the users.

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 net-

work 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. The 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. This is the frequency at which a decision to update the window size is made.

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% of the bits examined are set, the window size is reduced from its current value of W_c. Otherwise, the window size is increased. This is the signal filtering at the user.

4. Decision Function
When the window size is increased, we increment W_c by 1. When the window size is decreased, it is decreased to $0.875 * W_c$. This is the decision function that the user adopts.

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. The decision makers (users) filter the signals that are received from the network to determine if the traffic placed on the network should be decreased or may be increased. The binary input to the user from the signal filtering component is used to control the amount of traffic placed on the network. The decision function 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 requires a distributed control algorithm. Each individual user controls the amount of traffic placed on the network, based on the feedback received from the network. Multiple decision makers (users) have to coordinate and cooperate in implementing the congestion avoidance policy. Furthermore, 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) is varying 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 decision maker 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, which we call *feedback filtering*. The second is the filtering of the signal fed back by the decision maker, which we call *signal filtering*. The decision maker adjusts the frequency of change in the amount of traffic placed on the network. This allows for the users to see the effect of the change before making another change.

We have approached the analysis of the overall scheme using a detailed simulation, with relevant details of the transport protocols represented. Some of the characteristics of the congestion avoidance policy have been studied analytically, wherever possible. We model the computer network as multiple users generating jobs (packets) in a closed queueing network.

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 (at which power is maximized) is dependent on the service times of the routers in the path between a source and destination. This work has already been reported in [Ram86]. 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 was that 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 transport protocol) 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.).

A variety of network configurations were considered. The multiple hops for communication (routers and links) are represented as service centers with queues for packets awaiting service at these nodes. We use the term 'router' to mean both routers as well as links. Links which can process multiple packets at a time, as we shall see 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 the 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 [JR88].

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 queueing 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, e.g., when the service times of all the hops are identical. We may obtain this approximately using the balanced job bounds analysis [ZSEG82] for 'unbalanced' networks [Ram86]. Practically, we do not know the service times of the individual hops in the network. We use a function called *Power* at each router and use this 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 [Kle79]. Maximizing power has been proposed as an objective for computer networks [GHKP78]. Power at any resource is defined as:

$$Power = \frac{Throughput^{\alpha}}{ResponseTime}, \quad where: \; 0 < \alpha < 1 \tag{1}$$

We note that power has a single maximum as shown in [JR88]. 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 would be that the efficiency is 0% if the throughput is zero, or the response time is infinity and the efficiency is 100% at the maximally efficient operating point. The normalized power defined by the ratio of power to its value at knee satisfies this requirement. The efficiency of a resource's usage is therefore quantified by:

$$\text{Resource Efficiency} = \frac{\text{Resource Power}}{\text{Resource Power at knee}}$$

$$= \frac{(\text{Throughput/Knee throughput})}{(\text{Response time/Knee Response time})}$$

Notice, that the resource is used at 100% efficiency at the knee and as we move away from the knee, the resource is being used inefficiently, i.e., either underutilized (throughput lower than the knee-capacity) or overutilized (high 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, i.e., all users have identical demands say D. The maximally fair allocation then consists of equal allocations to all users, i.e., $A_i = A$ for all i. The fairness of any other (non-equal) allocation to each of the users is measured by the following fairness function [JCH84]:

$$f = \frac{\left(\sum_{i=1}^{N} x_i\right)^2}{\left(n \sum_{i=1}^{N} x_i^2\right)} \quad where, x_i = \frac{A_i}{D} \qquad (2)$$

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

We use user throughputs to measure allocations, A_i and demands D, 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 [JR88]. 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

$$W = \frac{Throughput}{Round\ Trip\ time} \qquad (3)$$

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

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 have used 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 the multiple queues per router if needed. We use the generic term router to represent each individual router or link (whichever is the bottleneck between the two) in the network. Further, we have assumed that all the sources of traffic share the same path for purposes of this study.

The routers use a feedback signal to indicate congestion. This is achieved by setting the *congestion indication* bit in the routing layer header when the router is congested. A variety of feedback schemes for flow and congestion control have been proposed in the literature. When the destination is congested, explicit feedback mechanisms, such as *ON-OFF* schemes, [Rei83], [YY83] and source-quench packets [Nag84], [ea79]. We describe a feedback scheme that the router uses to indicate congestion in a network using a connectionless network layer.

The router may be monitored to detect congestion in the network. This may be performed either by looking at the utilization of the router or the queue length. 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 upon 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% before any performance degradation is seen. When there is considerable variance in the packet size distribution, then the utilization is no longer a good estimate of congestion of the router. The average queue length may also be used to reflect congestion of the router, irrespective of the distribution of the service time. Therefore, we use 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 into two classes as being 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. Figure 2 shows the isolated router. Two thresholds, T_1 and T_2, ($T_1 <= T_2$),

Figure 2: Isolated Router as a single server queue

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.

When the explicit feedback signal is transmitted as a separate packet to the sources generating congestion it may result in additional congestion. Hysteresis has been proposed as a scheme to reduce switching overheads and the communication that results. Using hysteresis at the signal generation point minimizes this overhead of sending congestion '*on*' and '*off*' signals. Such hysteresis policies have been studied in the literature [YY83], [Har84]. We will study the effect of generating the feedback signal using a single threshold as well as determine if there is any benefit to using a hysteresis algorithm in the context of the bit scheme. Note that with the binary feedback scheme, the generation and communication of the feedback signal itself does not consume any significant additional resources, both of the CPU as well as the link.

We studied the policy for setting the bit by observing the behavior of *global power* by simulation. Multiple users share the path, comprising multiple routers, which in the general case need not be simultaneous. We considered the policies of using a single threshold as well as hysteresis to set the bit. Figure 3 shows the variation of power with the hysteresis value used by the router to set the bit. The representation for the hysteresis

Figure 3: Behavior of Power with Hysteresis

algorithm used in the figures specifies the center C of the range of the hysteresis and also the width of the hysteresis, K. Thus, $T_1 = C - K$ and $T_2 = C + K$. This representation can therefore be used for both the hysteresis as well as the single-threshold policies. We find that the power is maximum when the hysteresis is non-existent with the threshold value = 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 bit on an arriving packet when the number of packets at the router is greater than or equal to 1.

4.1 Feedback Filter

The tradeoff being made by setting the congestion avoidance bit when the queue size is 1 is between significant queueing (and higher throughput) versus increased idle time (and lower response time) at the router.

To ensure that we operate at the correct point, we do not use the instantaneous queue sizes, but instead use the average queue size. We set the bit on packets flowing through a router determined to be 'congested' when its average queue length is above the threshold of 1. The problem with using the instantaneous queue lengths is that we may signal congestion prematurely, thus potentially increasing the idle time of the router. When the instantaneous values for the queue sizes at the intermediate resources of the network are used, we find it is possible that some sources have the bits set while some others do not. Using the instantaneous queue lengths leads to the generation of congestion signals that may not be relevant when it reaches the sources (effect of feedback delay) and may also not be fair to the individual users that receive the signal. Therefore, wwe need a low-pass filter function to pass only those states of the routers taht are expected to last long enough for the user action to be meaningful.

Several filtering techniques for the feedback signal at the router were attempted. To provide a consistent state of the router, the router needed to use some form of the average queue length rather than the instantaneous queue length to set the congestion avoidance bit. We attempted to use the average over a fixed interval of time, and examined the behavior with different values of 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 find that the inconsistency 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 cycle seen by the individual routers of the load placed on them by users. The cycle time T is determined at the router adaptively. A cycle is defined as a $busy + idle$ interval seen at the router. This interval is also called a 'regeneration cycle' and the begining of the busy period is called a 'regeneration point'. The word 'regeneration' signifies the birth of a 'new' system since the queueing system's behaviour after the rengeneration point does not depend upon that before it. The average queue length is given by the area under the curve divided by the total cycle time. This average will be used for feedback 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. This is achieved by basing the queue average on the previous cycle as well as the current, though incomplete cycle. This is shown in Figure 4 reflecting a hypothetical behavior of the router queue length. The

Figure 4: A Regeneration Cycle at the Router

queue average is computed by considering the integral (area under the curve) of the queue length since the begining of the last cycle. The averaging is now performed as each packet arrives at the router. Thus, we find that as the length of the current cycle gets longer, the average due to the current cycle begins to dominate and the effect of the average 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 (the decision maker) 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
- Increase/Decrease Algorithms

The frequency of decision making determines the period in terms of the number of packets that have been received (or for which acknowledgments have been received, depending on where the decision maker is located), between updates to the window size. The second component determines the number of fed back congestion indication bits that are used to determine the update to the window size. The signal filtering component is used to filter the noise that may be received in the signal. 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 performed. Our initial approach was to make a decision at the instant each acknowledgment was received. We assume that to there is no acknowledgment accumulation and thus, 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 decision maker 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. We consider the state space to be a boolean value (uncongested or congested), which is represented by the single bit received at the decision maker for every packet. Therefore, prematurely altering the window size before receiving the signals indicating the effect of the new window size on the network, may cause over-correction. Thus, altering the window size after every acknowledgment causes considerable oscillation, as shown in Figure 5.

To demonstrate the feedback delay, consider an example of a new source of traffic deciding to join the network with a large starting window of w_1. As shown in Figure 6, this is a case of the

Figure 5: Behavior of Window size updated every Acknowledgement

source changing its window from w_0 to w_1 with $w_0 = 0$. Let us assume that this happens at time $t = 0$. The effect of this window change will not be felt immediately. In fact, the first few packets will find the network response to be the same as before the source came on. The first network feedback to the source will come with the first packet at time $t = r_0$, where r_0 is the round-trip delay corresponding to the old control (zero window from this source). It is only the first packet in the next window cycle ($(w_1+1)^{th}$ packet) that will bring a network feedback corresponding to window w_1. This packet would enter the network at time $t = r_0$ and come back at time $t = r_0 + r_1$, where r_1 is the round-trip delay corresponding to window w_1. The key point to notice is that it takes at least [1] two round-trip delays for the effect of a window change to be observed. The feedback signals $\mathbf{y}(n)$ (a vector) observed in the

[1]The delay may be more if the network feedback signals are based on the state of the network in the previous cycle rather than this cycle.

Figure 6: Decision Frequency. After the window w is changed from w_0 to w_1, the feedback $\mathbf{f}$ received during the second round-trip delay interval is a function of w_1.

n^{th} cycle correspond to the windows during cycles $n-1$ and $n-2$.

$$\mathbf{y}(n) = f\{w(n-1), w(n-2)\}$$

where $w(n)$ is the feedback signals corresponding to window in cycle n. $w(n)$ may be determined as a function of all past feedback and window history:

$$w(n+1) = fn\{w(n-j), \mathbf{y}(n-i), i = 0, 1, 2, \ldots, j = 0, 1, 2, \ldots\}$$

The most general control functions may require us to remember a long history. A simple control policy is obtained if we keep the window constant for two cycles, so that $w(n-1) = w(n)$, where for n is an even integer, and use only the feedback for the last cycle. That is, for even values of n:

$$\mathbf{y}(n) = f\{w(n-1)\}$$

$$w(n+1) = f\{w(n), \mathbf{y}(n)\}$$

Our approach, therefore, has been to introduce a waiting period after every window size update, before the next update is performed. Consider

the situation at each of the sources, where W_p = window size before the update, and W_c = window size after the update.
We wait for $(W_p + W_c)$ acknowledgments. Part of these acknowledgments would correspond to those for the window size W_c. Further, 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 7 shows the behavior of the window size with the frequency of update being changed to once every $W_p + W_c$ acknowledgments. The oscillation of the window size is now considerably reduced.

Figure 7: Behavior of Window size updated every Two window sizes

5.2 Use of Received Information

The next issue is whether the decision maker maintains information (the bits returned in the acknowledgments) after a decision is made. We make the observation that maintaining bits used in the previous decision caused over-correction, using the single cut-off filtering algorithm at the decision maker. Using past history results in domination of that history of bits received for a long period. This period may be longer than the duration during which the network was congested. For example, the window size is reduced to below the optimal value before the current state of the network, and the signals generated as a result dominate and cause a correction in the right direction. For reasons of simplicity, old information is erased after a window size is changed. In fact, we discard any of the history that is maintained in the network itself, after a decision 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. As we described in the previous subsection on the decision frequency, we update the window after every $W_p + W_c$ acknowledgments. Therefore, we have received $W_p + W_c$ congestion avoidance bits. 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 to update the window size. This is based on simplicity and a motivation to avoid any additional state being maintained to relate the bits received to the appropriate window size.

5.3 Signal Filtering

The feedback signals that are generated by the router in the network (routers) are received at the decision maker. Let us consider initially for the purposes of this discussion, that the decision maker is at the source. We have one bit of information fed back from the router for each packet that is transmitted by the source. The decision maker filters 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 source.

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 cut-off 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 cut-off values, then the filter may specify that the source increase its window size by increasing amounts, based on the percentage of the received bits being set is below cut-off factors 1, 2, ...,n. In the same way, cut-off 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 cut-off factors results in greater complexity of the increase/decrease algorithms.

The algorithm that we have adopted uses a single cut-off 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 cut-off factor is dependent on the policy used by the routers in the network to set the bit in order 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 inter-arrival and service times at that router are exponentially distributed.
Let λ = the mean inter-arrival time.
Let μ = the mean service time.
Let $\rho = \frac{\lambda}{\mu}$ utilization of the bottleneck router.
Then we can express Power as:

$$Power = \frac{(1 - \rho)}{\lambda/\mu} \tag{4}$$

We may then show that the value of ρ, at which power at the router is maximized, is 0.5.

Now using this value of power, we consider the probability of having the congestion avoidance bit set for different values of the threshold at the router.

Let M = the threshold at which the congestion indication bit is set.
Let $P(n)$ = Probability of n customers at the router, including the one in service.
The Probability(bit set by the router) = $1 - (P(0) + P(1) + ... + P(M-1))$
when
$M = 1,\ Prob(bit\ set) = 1 - P(0) = \rho = 0.5,$

when $M = 2,\ \ Prob(bit\ set) = 0.25.$

Thus, when the threshold (M) at which the congestion avoidance bit being set by the router is 1, then the percentage of bits that are set by the router is = 50%. When the service times are deterministic. power is maximized when the utilization $\rho = 1$. With a threshold M of 1, the $Prob(bit\ set) = 1.0$.

Thus, the relationship between M 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 M at the router. The service time distribution depends on the packet size distribution, since we primarily model the service time at each of the service centers as a function of the packet size. Figure 8 shows the variation of the value of global power with the variation of the cut-off factor for the signal filter, for various values of the router threshold M for a network with multiple nodes. The users follow the increase/decrease algorithm described in the next section. We find that power is maximum when the router threshold M is 1. and power is maximum when the cut-off factor for the percentage of received bits being set is 50%. When M is 2, the power is maximum when the cut-off factor is 25%. We have used a value of M = 1, and used a cut-off factor value of 50%.

Figure 8: Behavior of Power with varying cutoff values

5.4 Increase/Decrease Algorithms for the Window Size

When all the users share the same path, the algorithms followed by the servers in the network ensure that all the users receive the same signal of congestion from the network. The signal filter at the decision maker provides a binary signal to increase or decrease the window size. Here, we present some justification and primarily the results of using the additive increase/multiplicative decrease decision function discussed in [JR88].

The considerations the decision maker must have for the decision function are:

- 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.
- Minimize the time to achieve steady state.

Some of these criteria are quantified by defining the individual window sizes, a fairness measure defined in [JCH84], and our global power metric.

Consider the simple *additive increase/additive decrease* function to start with. This decision function is described by the following equations: Let W_i^t = window size at decision epoch t of source i.

- Additive Increase and additive decrease (algorithm A)
  Increase: $W_i^{t+1} = W_i^t + b, \quad b \geq 0$
  Decrease: $W_i^{t+1} = W_i^t - d, \quad d \geq 0$

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 9. The unfairness appears to arise from the fact that all the participating sources increase or decrease by equal amounts. In [JR88], we provide the justification to consider 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: $W_i^{t+1} = W_i^t + b, \quad b \geq 0$
  Decrease: $W_i^{t+1} = cW_i^t, \quad 0 < c \leq 1$

We show in Figure 10 and that we can achieve fairness by using algorithm B. As described in [JR88], although the control placed on the network is discrete since the window sizes are integer values, we use the real values to maintain the window sizes 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.

Figure 9: Behavior of Window Size with Additive Increase/Decrease

Figure 10: Behavior of Window Size with Additive Increase/Multiplicative Decrease =0.9

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. The users increase additively by 1 and decrease multiplicatively by a factor of 0.8. The optimal window for the configuration, as before was 15.5. If the two users start at different times, we find that the two users stabilize 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 (using a factor of 0.8) to 8 and 4 (0.8(6)=4.8 truncated to 4). The total window is less than w_{knee} and hence both users are asked to go up. They go up by 1 to 9 and 5. The total window is still less than w_{knee} and the users go up to 10 and 6. After this, the cycle repeats and the second user gets 6/10th of the first user's throughput. For the same configuration, when real values are used, we find that the allocations to the users are fair, as shown in Figure 10. 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 window is adequate.

There are several other considerations that influence the choice of parameters for the increase factor and the decrease factor (b and c for algorithm B). By using a small decrease factor (for instance, reducing the window size to 50% of the current value, compared to 90% 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 reduces the amount of throughput degradation because of the increase/decrease policies. We choose a value of 0.875 for the decrease factor based on the ease of implementation, while minimizing oscillations.

6 Testing of the Binary Feedback Scheme

In this section we discuss the behavior of the binary feedback scheme for some of conditions that have been outlined in [JR88] for a scheme to be acceptable. In the previous sections, We have already seen the capability of the scheme to to operate at the maximally efficient point and be fair across multiple users.

6.1 Behavior with Random Packet Size Distributions

We consider the behavior of the scheme with randomly distributed packet size distributions. Figure 11 shows the behavior of the window size of two sources with exponentially distributed packet sizes. The mean service time at each of the network routers in the path are different so that we have a non-homogeneous path. The path also contains a satellite to reflect the ability of the bit scheme to accommodate the long delays in such links as well. The maximally efficient aggregate window size for this configuration 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. We also see that the two sources start at different times. The result of the multiplicative decrease/additive increase algorithms show that the two sources reach a fair value of the network resources allocated to them, as evidenced by their window size.

Figure 11: Behavior of Window Size with Exponential Packet Size Distribution

6.2 Behavior of Scheme Under Transients

Any scheme that 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 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 optimum window size changing during this transient period.

Figure 12 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. We find that after a small initial undershoot, the overall window size recovers and the network operates at its new maximally efficient point. When

Figure 12: Behavior of Window Size with Transient in Bottleneck Service Time

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 we see in Figure 12.

The other transient condition that we typically see in a network is the injection of additional load by users who start up. 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 10 we showed 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.

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 avoidance bit (if it is not already set). This information is then returned to the user by the destination which receives the packet. This information is utilized 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 this model. We studied the policies that need to be used in each of these components through analysis as well as simulation.

The network servers 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* cycle time seen at the server. The decision makers (source or destination, based on the 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 used by a signal filter at the decision maker. When at least 50% of these bits are set, the window size is reduced from its current value of W_c to 87.5% of its value. Otherwise, it is increased by 1.

We showed 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 also addressed an important 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.

We 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.

References

[Ahu79] V. Ahuja. Routing and flow control in systems network architecture. *IBM Systems Journal*, 18(2):293–314, 1979.

[BG85] Werner Bux and Davide Grillo. Flow control in local-area networks of interconnected token rings. *IEEE Transactions on Communications*, COM-33(10):1058–1066, October 1985.

[DNA82] Digital Equipment Corporation. *DECnet Digital Network Architecture (Phase IV) General Description*, 1982. Order No. AA-N149A-TC.

[ea79] J. C. Majithia et al. Experiments in congestion control techniques. In *Proceedings of the International Symposium on Flow Contol in Computer Networks*, pages 211–234, February 1979.

[GHKP78] A. Giessler, J. Haanle, A. Konig, and E. Pade. Free buffer allocation - an investigation by simulation. *Computer Networks*, 1(3):191–204, July 1978.

[GK80] Mario Gerla and Leonard Kleinrock. Flow control: A comparative survey. *IEEE Transactions on Communications*, COM-28(4):553–574, April 1980.

[Har84] Peter G. Harrison. An analytic model for flow control schemes in communication network nodes. *IEEE Transactions on Communications*, COM-32(9):1013–1019, September 1984.

[ISO86] International Organization for Standardization. *ISO 8073: Information Processing Systems - Open Systems Interconnection - Connection Oriented Transport Protocol Specification*, July 1986. (Ref.no. ISO 8073-1986 (E)).

[Jai86] Raj Jain. A timeout-based congestion control scheme for window flow-controlled networks. *IEEE Journal on Selected Areas in Communications*, October 1986.

[JCH84] R. K. Jain, Dah-Ming Chiu, and William R. Hawe. A quantitative measure of fairness and discrimination for resource allocation in shared systems. DEC Technical Report TR-301, Digital Equipment Corporation, September 1984.

[JR88] Raj Jain and K. K. Ramakrishnan. Congestion avoidance in computer networks with a connectionless network layer: Concepts, goals and methodology. In *Proceedings of the Computer Networking Symposium*, pages 134–143, April 1988.

[Kle78] Leonard Kleinrock. On flow control in computer networks. In *Proceedings of the International Conference on Communications*, June 1978.

[Kle79] L. Kleinrock. Power and deterministic rules of thumb for probabilistic problems in computer communications. *Proceedings of the International Conference on Communcations*, June 1979.

[Nag84] John Nagle. Congestion control in tcp/ip internetworks. *Computer Communication Review*, 14(4), October 1984.

[Ram86] K. K. Ramakrishnan. Analysis of a dynamic window congestion control protocol in heterogeneous environments including satellite links. In *Proceedings of the Computer Networking Symposium*, November 1986.

[Rei83] M. Reiser. Queueing and delay analysis of a buffer pool with resume level. In A. K. Agrawala and S. K. Tripathi, editors, *Performance '83*, pages 25–32, May 1983.

[Tan81] Andrew S. Tanenbaum. *Computer Networks*. Prentice-Hall, Englewood Cliffs, N. J., 1981.

[YY83] Takshing P. Yum and Hung-Ming Yen. Design algorithm for a hysteresis buffer congestion control strategy. In *Proceedings of the IEEE International Conference on Communications*, pages 499–503, June 1983.

[ZSEG82] J. Zahorjan, K. C. Sevcik, D. L. Eager, and B. Galler. Balanced job bound analysis of queueing networks. *Communications of the ACM*, 25(2), February 1982.