Analysis of the Increase and Decrease Algorithms for Congestion Avoidance in Computer Networks

Dah-Ming CHIU and Raj JAIN
Digital Equipment Corporation, 550 King Street (LKG1-2/A19), Littleton, MA 01460-1289, U.S.A.

Abstract. Congestion avoidance mechanisms allow a network to operate in the optimal region of low delay and high throughput, thereby, preventing the network from becoming congested. This is different from the traditional congestion control mechanisms that allow the network to recover from the congested state of high delay and low throughput. Both congestion avoidance and congestion control mechanisms are basically resource management problems. They can be formulated as system control problems in which the system senses its state and feeds this back to its users who adjust their controls.

The key component of any congestion avoidance scheme is the algorithm (or control function) used by the users to increase or decrease their load (window or rate). We abstractly characterize a wide class of such increase/decrease algorithms and compare them using several different performance metrics. They key metrics are efficiency, fairness, convergence time, and size of oscillations.

It is shown that a simple additive increase and multiplicative decrease algorithm satisfies the sufficient conditions for convergence to an efficient and fair state regardless of the starting state of the network. This is the algorithm finally chosen for implementation in the congestion avoidance scheme recommended for Digital Networking Architecture and OSI Transport Class 4 Networks.

Keywords. Computer Network, Network Performance, Resource Management, Congestion Control, Congestion Avoidance, Flow Control, Fairness.

1. Introduction

1.1. Background

Congestion in computer networks is becoming an important issue due to the increasing mismatch in link speeds caused by intermixing of old and new technology. Recent technological advances

Dah-Ming Chiu received the B.Sc. degree with first class honours from Imperial College of Science and Technology, London University, in 1975, and the M.S. and Ph.D. degrees from Harvard University, Cambridge, MA, in 1976 and 1980 respectively.

From 1979 to 1980, he was with AT&T Bell Laboratories, where he worked on applying queuing theory to network modeling. Since 1980, he has been with Digital Equipment Corporation, where he has worked on performance modeling and analysis of computer systems and networks. Currently, he is a member of the Distributed Systems Architecture group, where he is working on the name service architecture and analyzing various distributed algorithms. His research interests include operation systems, distributed systems, computer networks, performance analysis, and optimization theories.

Dr. Chiu is a member of the ACM and the IEEE.

Raj Jain received the B.E. degree from A.P.S. University, Rewa, India, the M.E. degree from Indian Institute of Science, Bangalore, India, and the Ph.D. degree from Harvard University, Cambridge, MA, in 1972, 1974, and 1978, respectively.

His Ph.D. dissertation, entitled "Control Theoretic Formulation of Operating Systems Resource Management Policies," was published by Garland Publishing, Inc. of New York in their "Outstanding Dissertations in the Computer Sciences" series. Since 1978, he has been with Digital Equipment Corporation, where he has been involved in performance modeling and analysis of a number of computer systems and networks including VAX Clusters, DECnet, and Ethernet. Currently, he is a Consulting Engineer in the Distributed Systems Architecture and Performance Group. He spent the 1983–1984 academic year on a sabbatical at the Massachusetts Institute of Technology doing research on the performance of networks and local area systems. For three years he also taught a graduate course on computer systems performance techniques at MIT and is writing a textbook on this subject, to be published by Wiley-Interscience.

Dr. Jain is a member of the Association for Computing Machinery, and a senior member of IEEE.

North-Holland
Computer Networks and ISDN Systems 17 (1989) 1–14

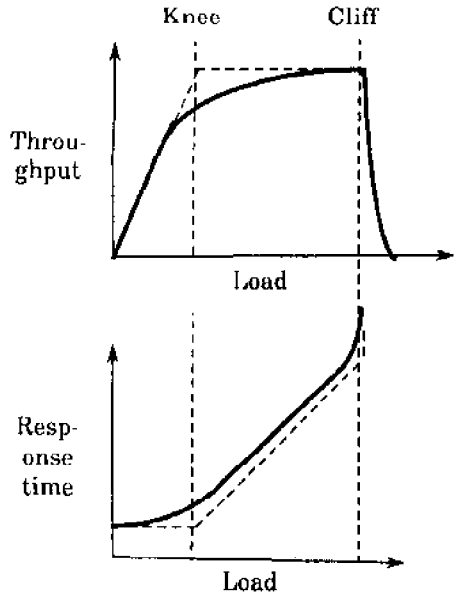

Fig. 1. Network performance as a function of the load. Broken curves indicate performance with deterministic service and interarrival times.

such as local area networks (LANs) and fiber optic LANs have resulted in a significant increase in the bandwidths of computer network links. However, these new technologies must coexist with the old low bandwidth media such as the twisted pair. This heterogeneity has resulted in a mismatch of arrival and service rates in the intermediate nodes in the network causing increased queuing and congestion.

Traditional congestion control schemes help improve performance after congestion has occurred. Figure 1 shows general patterns of response time and throughput of a network as the network load increases. If the load is small, throughput generally keeps up with the load. As the load increases, throughput increases. After the load reaches the network capacity, throughput stops increasing. If the load is increased any further, the queues start building, potentially resulting in packets being dropped. Throughput may suddenly drop when the load increases beyond this point and the network is said to be *congested*. The response-time curve follows a similar pattern. At first the response time increases little with load. When the queues start building up, the response time increases linearly until finally, as the queues start overflowing, the response time increases drastically.

The point at which the packets start getting lost is called a *cliff* due to the fact that the throughput falls off rapidly after this point. We use the term *knee* to describe the point after which the increase in the throughput is small, but when a significant increase in the response time results.

A scheme that allows the network to operate at the knee is called a *congestion avoidance* scheme, as distinguished from a *congestion control* scheme that tries to keep the network operating in the zone to the left of the cliff. A properly designed congestion avoidance scheme will ensure that the users are encouraged to increase their traffic load as long as this does not significantly affect the response time, and are required to decrease them if that happens. Thus, the network load oscillates around the knee.

Both congestion avoidance and congestion control mechanisms are dynamic resource management problems that can be formulated as system control problems in which the system senses its state and feeds this back to its users who adjust their controls. For the congestion problem, the state consists of the load on the network and the control is the number of packets put into the network by the users. Often a window mechanism is used in the transport layer protocols to limit the number of packets put into the network. An alternative mechanism consists of setting a limit on the rate (packets per second or bits per second) that can be sent by a user. In either case, the control (window or rate) can be dynamically adjusted as the total load on the system changes. This control, which we call the increase/decrease algorithm, is at the heart of all congestion avoidance mechanisms.

We have investigated a number of congestion avoidance mechanisms, reported in a series of papers, and this paper is a part of that series [7,8,10,11]. The concept of congestion avoidance and several alternatives are described in [7]. We chose a particular alternative called the "binary feedback scheme" which is described in detail in [11]. This scheme is later extended in [10] to include a "selective feedback" mechanism in which the routers monitor different users and permit some users to increase load while requesting others to decrease load. All of our work on congestion avoidance is summarized in [8].

This paper concentrates on a detailed analysis of the increase/decrease algorithms. This analysis resulted in the selection of the increase/decrease algorithms used in the binary feedback scheme proposed in [11] and [10]. However, the analysis presented here is general and applies to many other applications besides congestion avoidance.

Briefly, the binary feedback scheme for congestion avoidance operates as follows. The resources in the network monitor their usage and determine if they are loaded below or above an optimal load level. Depending upon the load level, the resource sends a binary feedback (1 = overloaded, 0 = underloaded) to the users who then adjust their load using an increase/decrease algorithm. This binary feedback is sent by setting a bit in the packet header. The use of a bit in the packet header as a feedback mechanism has been incorporated into the OSI connectionless networking protocol standards [4]. The bit is called a "congestion experienced bit" and is a part of a field called "quality of service" in the network layer header.

The abstract model assumes that all the users sharing the same bottleneck will receive the *same* feedback. Based on this feedback, the users try to adjust their load so that the bottleneck is efficiently used as well as equally shared by all users. In this abstracted context, we assume that the feedback and control loop for all users is synchronous, that is, all users receive the same feedback and react to it; the next feedback is then generated after all users have reacted to the feedback and so on. Also, we concentrate on one bottleneck resource and the users that share it. Because of these abstractions, we are able to demonstrate some of the subtle behavior of this type of algorithm. The results presented here were verified by detailed simulations of real networks [7,10,11].

1.2. Past Work

The algorithms studied here belong to a class of distributed algorithms for managing distributed resources. A spectrum of such distributed algorithms have been studied in the literature. At one end of the spectrum, we have centralized decision-making. In this paradigm, information (about user demands) flows to the resource managers, and the decision of how to allocate the resource is made at the resource. The analysis by Sanders [12] is a good example of this approach. At the other end of the spectrum, we have decentralized decision-making. In this case the decisions are made by the users while the resources feed information regarding current resource usage. Algorithms studied by Jaffe [5] and later extensions by Gafni [2] and Mosely [9] are all good examples of this approach.

In this paper we analyze a class of decentralized decision-making algorithms that are based on a special form of feedback, namely the feedback from the resource is a binary signal. This binary signal indicates whether the resource is currently overloaded or underutilized. A very good reason for considering a binary form of feedback is the motivation of making the controller/manager of the resource as simple and efficient as possible. The requirement of a binary feedback often minimizes the work at the resource in generating the feedback.

1.3. Notations and Definitions

Figure 2 shows the assumed model of the network with n users sharing it. The congestion state of the system is determined by the number of packets in the system. We assume a discrete time operation with time divided into small slots. These slots basically represent intervals at the beginning of which the users set their load level based on the network feedback received during the previous interval. If during time slot t, the ith user's load is $x_i(t)$, then the total load at the bottleneck resource would be $\Sigma x_i(t)$, and the state of the system is denoted by the n-dimensional vector $x(t) = \{x_1(t), x_2(t), \ldots, x_n(t)\}$. Since we are operating at or near the knee, all resources demanded by the users are granted (this is not true at the cliff). Thus, $x_i(t)$ denotes the ith user's

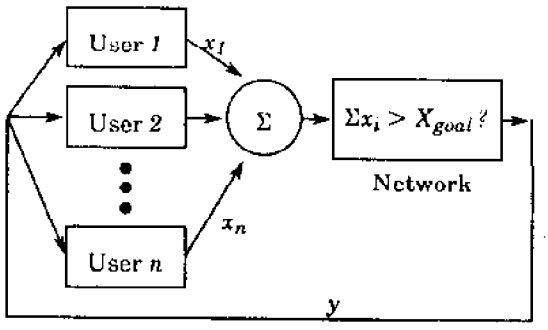

Fig. 2. A control system model of n users sharing a network.

demand as well as allocation of the system's resources. During the interval, the system determines its load level and sends a binary feedback $y(t)$, which is interpreted by the users as follows:

$$y(t) = \begin{cases} 0 \rightarrow \text{Increase load,} \\ 1 \Rightarrow \text{Decrease load.} \end{cases}$$

The users cooperate with the system and change (increase of decrease) their demands by an amount $u_i(t)$. Thus,

$$x_i(t+1) = x_i(t) + u_i(t). \tag{1}$$

The change $u_i(t)$ represents ith user's control. It is a function of the user's previous demand and the system feedback:

$$u_i(t) = f(x_i(t),\ y(t)). \tag{2}$$

In other words,

$$x_i(t+1) = x_i(t) + f(x_i(t),\ y(t)).$$

Notice that the users are not aware of other user's individual demands and, thus, cannot make $u_i(t)$ a function of $x_j(t)$, $j \neq i$. In general, the control function $f(\ )$ can be any linear or nonlinear function. However, we will focus first on *linear* controls. The state equations (1) reduce to

$$x_i(t+1) = \begin{cases} a_I + b_I x_i(t) & \text{if } y(t) = 0 \Rightarrow \text{Increase,} \\ a_D + b_D x_i(t) & \text{if } y(t) = 1 \Rightarrow \text{Decrease.} \end{cases}$$

Here, a_I, b_I, a_D, b_D are constants. The following are some examples of the control functions:

(1) *Multiplicative Increase/Multiplicative Decrease*:

$$x_i(t+1) = \begin{cases} b_I x_i(t) & \text{if } y(t) = 0 \Rightarrow \text{Increase,} \\ b_D x_i(t) & \text{if } y(t) = 1 \Rightarrow \text{Decrease.} \end{cases}$$

Here, $b_I > 1$ and $0 < b_D < 1$. All users increase their demands by multiplying their previous demands by a constant factor. The decrease is also multiplicative.

(2) *Additive Increase/Additive Decrease*:

$$x_i(t+1) = \begin{cases} a_I + x_i(t) & \text{if } y(t) = 0 \Rightarrow \text{Increase,} \\ a_D + x_i(t) & \text{if } y(t) = 1 \Rightarrow \text{Decrease.} \end{cases}$$

Here, $a_I > 0$ and $a_D < 0$. All users increase their demands by adding a constant amount to their previous demands. The decrease is also additive.[1]

(3) *Additive Increase/Multiplicative Decrease*:

$$x_i(t+1) = \begin{cases} a_I + x_i(t) & \text{if } y(t) = 0 \Rightarrow \text{Increase,} \\ b_D x_i(t) & \text{if } y(t) = 1 \Rightarrow \text{Decrease.} \end{cases}$$

The increase is by a constant amount but the decrease is by a constant factor.

(4) *Multiplicative Increase/Additive Decrease*:

$$x_i(t+1) = \begin{cases} b_I x_i(t) & \text{if } y(t) = 0 \Rightarrow \text{Increase,} \\ a_D + x_i(t) & \text{if } y(t) = 1 \Rightarrow \text{Decrease.} \end{cases}$$

In order to evaluate the effectiveness of these controls, we next define a set of criteria explicitly in the next section.

1.4. Criteria for Selecting Controls

The key criteria are: *efficiency*, *fairness*, *distributedness*, and *convergence*. We define them formally as follows:

(1) *Efficiency:* The efficiency of a resource usage is defined by the closeness of the total load on the resource to its knee. If X_{goal} denotes the desired load level at the knee, then the resource is operating efficiently as long as the total allocation $X(t) = \sum x_i(t)$ is close to X_{goal}. Overload ($X(t) > X_{goal}$) or underload ($X(t) < X_{goal}$) are both undesirable and are considered inefficient. We consider both as equally undesirable.

Notice, that efficiency relates only to the total allocations and thus two different allocations can both be efficient as long as the total allocation is close to the goal. The distribution of the total allocation among individual users is measured by the fairness criterion.

(2) *Fairness:* The fairness criterion has been widely studied in the literature. When multiple users share multiple resources, the *maxmin fairness* criterion has been widely adopted [2,3,5,10]. Essentially, the set of users are partitioned into equivalent classes according to which resource is their primary bottleneck. The *maxmin* criterion then asserts that the users in the same equivalent

[1] It is assumed that truncation is applied when $a_D + x_i(t)$ is less than zero, so that $x_i(t)$ will never become negative.

class ought to have the equal share of the bottleneck. Thus, a system in which $x_i(t) = x_j(t)\ \forall i, j$ sharing the same bottleneck is operating fairly. If all users do not get exactly equal allocations, the system is less fair and we need an index or a function that quantifies the fairness. One such index is [6]:

$$\text{Fairness:}\quad F(x) = \frac{(\Sigma x_i)^2}{n(\Sigma x_i^2)}.$$

This index has the following properties:

(a) The fairness is bounded between 0 and 1 (or 0% and 100%). A totally fair allocation (with all x_i's equal) has a fairness of 1 and a totally unfair allocation (with all resources given to only one user) has a fairness of $1/n$ which is 0 in the limit as n tends to ∞.

(b) The fairness is independent of scale, i.e., unit of measurement does not matter.

(c) The fairness is a continuous function. Any slight change in allocation shows up in the fairness.

(d) If only k of n users share the resource equally with the remaining $n-k$ users not receiving any resource, then the fairness is k/n.

For other properties of this fairness function, see [6].

(3) *Distributedness:* The next requirement that we put on the control scheme is that it be distributed. A centralized scheme requires complete knowledge of the state of the system. For example, we may want to know each individual user's demand or their sum. This information may be available at the resource. However, conveying this information to each and every user causes considerable overhead, especially since a user may be using several resources at the same time. We are thus primarily interested in control schemes that can be implemented in real networks and, therefore, we assume that the system does the minimum amount of feedback. It only tells whether it is underloaded or overloaded via the binary feedback bits. Other information such as X_{goal} and the number of users sharing the resource are assumed to be unknown by the users. This restricts the set of feasible schemes. We, therefore, describe the set of feasible schemes with and without this restriction.

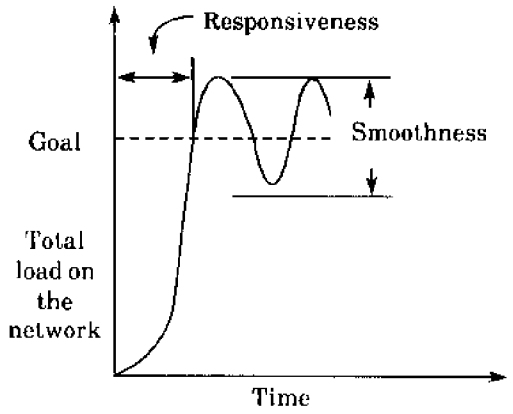

Fig. 3. Responsiveness and smoothness.

(4) *Convergence:* Finally we require the control scheme to converge. Convergence is generally measured by the speed with which (or time taken till) the system approaches the goal state from any starting state. However, due to the binary nature of the feedback, the system does not generally converge to a single steady state. Rather, the system reaches an "equilibrium" in which it oscillates around the optimal state. The time taken to reach this "equilibrium" and the size of the oscillations jointly determine the convergence. The time determines the *responsiveness*, and the size of the oscillations determine the *smoothness* of the control. Ideally, we would like the time as well as oscillations to be small. Thus, the controls with smaller time and smaller amplitude of oscillations are called more responsive and more smooth, respectively, as shown in Fig. 3.

1.5. Outline of this Paper

In this paper, we develop a simple and intuitive methodology to explain when and why a control converges. We address the following questions: *What are all the possible solutions that converge to efficient and fair states? How do we compare those controls that converge?*

The paper is organized as follows. In Section 2 we will characterize the set of all *linear* controls that converge and, thus, identify the set of feasible controls. Then we narrow down the feasible set to a subset that satisfies our distributedness criterion. These subset still includes controls that have unacceptable magnitudes of oscillation or those that converge too slowly. Then in Section 3, we discuss

how to find the subset of feasible distributed controls that represent the optimal trade-off of responsiveness and smoothness, as we defined in convergence. In Section 4, we discuss how the results extend to nonlinear controls. And in the last section we summarize the results and discuss some of the practical considerations (such as simplicity, robustness, and scalability).

2. Feasible Linear Controls

2.1. Vector Representation of the Dynamics

In determining the set of feasible controls, it is helpful to view the system state transitions as a trajectory through an n-dimensional vector space. We describe this method using a 2-user case, which can be viewed in a 2-dimensional space.

As shown in Fig. 4, any 2-user resource allocation $\{x_1(t), x_2(t)\}$ can be represented as a point (x_1, x_2) in a 2-dimensional space. In this figure, the horizontal axis represents allocations to user 1, and the vertical axis represents allocations to user 2. All allocations for which $x_1 + x_2 = X_{goal}$ are efficient allocations. This corresponds to the straight line marked "efficiency line". All allocations for which $x_1 = x_2$ are fair allocations. This corresponds to the straight line marked "fairness line". The two lines intersect at the point $(X_{goal}/2, X_{goal}/2)$ that is the optimal point. The goal of control schemes should be to bring the system to this point regardless of the starting position.

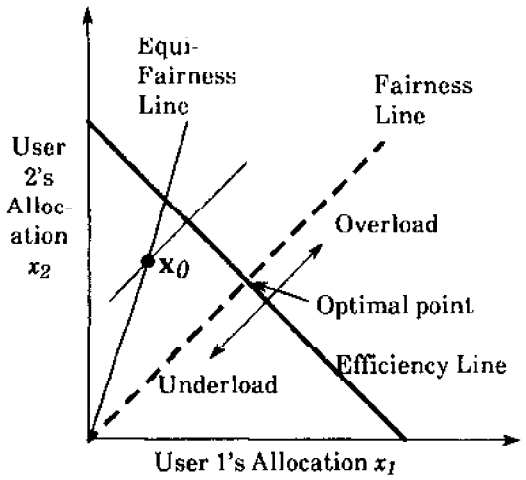

Fig. 4. Vector representation of a two-user case.

All points below the efficiency line represent an "underloaded" system and ideally the system would ask users to increase their load. Consider, for example, the point $x_0 = (x_{10}, x_{20})$. The additive increase policy of increasing both users' allocations by a_1 corresponds to moving along a 45° line. The multiplicative increase policy of increasing both users' allocations by a factor b_1 corresponds to moving along the line that connects the origin to the point. Similarly, all points above the efficiency line represent an "overloaded" system and additive decrease is represented by a 45° line, while multiplicative decrease is represented by the line joining the point to the origin.

The fairness at any point (x_1, x_2) is given by

$$\text{Fairness} = \frac{(x_1 + x_2)^2}{2(x_1^2 + x_2^2)}.$$

Notice that multiplying both allocations by a factor b does not change the fairness. That is, (bx_1, bx_2) has the same fairness as (x_1, x_2) for all values of b. Thus, all points on the line joining a point to origin have the same fairness. We, therefore, call a line passing through the origin a "equi-fairness" line. The fairness decreases as the slope of the line either increases above or decreases below the fairness line.

Figure 5 shows a complete trajectory of the two-user system starting from point x_0 using an additive increase/multiplicative decrease control policy. The point x_0 is below the efficiency line and so both users are asked to increase. They do so additively by moving along at an angle of 45°. This brings them to x_1 which happens to be above the efficiency line. The users are asked to decrease and they do so multiplicatively. This corresponds to moving towards the origin on the line joining x_1 and the origin. This brings them to point x_2, which happens to be below the efficiency line and the cycle repeats. Notice that x_2 has higher fairness than x_0. Thus, with every cycle, the fairness increases slightly, and eventually, the system converges to the optimal state in the sense that it keeps oscillating around the goal.

Similar trajectories can be drawn for other control policies. Although not all control policies converge. For example, Fig. 6 shows the trajectory for the additive increase/additive decrease control

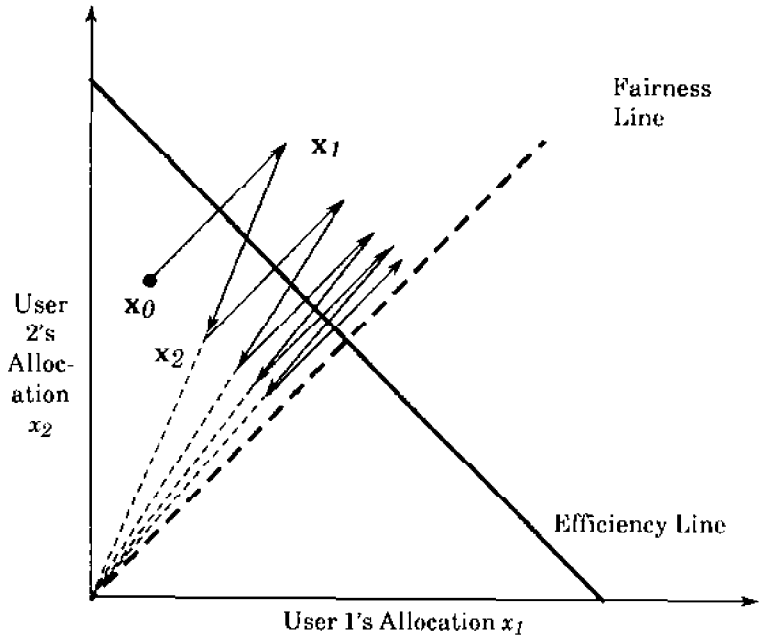

Fig. 5. Additive Increase/Multiplicative Decrease converges to the optimal point.

policy starting from the position x_0. The system keeps moving back and forth along a 45° line through x_0. With such a policy, the system can converge to efficiency, but not to fairness. The conditions for convergence to efficiency and fairness are derived algebraically in the next section.

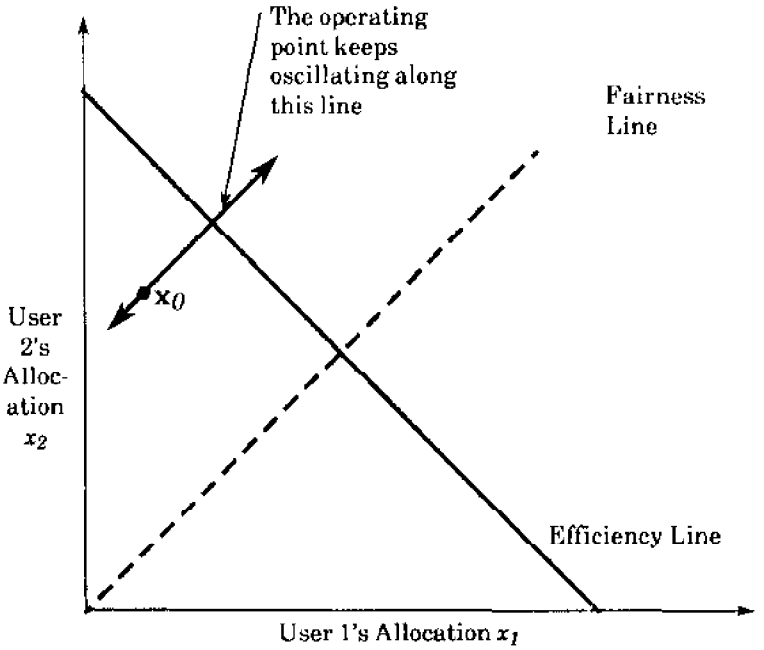

Fig. 6. Additive Increase/Additive Decrease does not converge.

2.2. Convergence to Efficiency

In order to guarantee convergence to efficiency we need to first make sure that at each step the system react correctly to the feedback by moving in the right direction. That is, when the system asks the users to decrease, we should ensure that the total load will not increase and when the system asks the users to increase, the total load will not decrease This is the principle of *negative feedback*.[2] Algebraically:

$$y(t) = 0 \Rightarrow \Sigma x_i(t+1) > \Sigma x_i(t),$$

$$y(t) = 1 \Rightarrow \Sigma x_i(t+1) < \Sigma x_i(t).$$

In terms of the policy parameters, this means that the parameter values should be

$$na_I + (b_I + 1)\Sigma x_i(t) > 0 \quad \forall n \text{ and } \forall \Sigma x_i(t),$$

$$na_D + (b_D - 1)\Sigma x_i(t) < 0 \quad \forall n \text{ and } \forall \Sigma x_i(t),$$

or, equivalently,

$$b_I > 1 - \frac{na_I}{\Sigma x_i(t)}$$

$$b_D < 1 - \frac{na_D}{\Sigma x_i(t)} \quad \forall n \text{ and } \forall \Sigma x_i(t). \tag{3}$$

2.3. Convergence to Fairness

Convergence to fairness is defined as moving towards the fairness index of one, i.e.,

$$F(x(t)) \to 1 \quad \text{as } t \to \infty.$$

The linear control policies affect the fairness as follows:

$$F(x(t+1)) = \frac{(\Sigma x_i(t+1))^2}{n(\Sigma x_i^2(t+1))} \tag{4}$$

$$= \frac{(\Sigma a + bx_i(t))^2}{n\Sigma(a + bx_i(t))^2} \tag{5}$$

$$\text{where } c = a/b \tag{6}$$

$$= F(x(t)) + (1 - F(x(t))) \times \left(1 - \frac{\Sigma x_i^2(t)}{\Sigma(c + x_i(t))^2}\right). \tag{7}$$

The last expression in the above equation is an increasing function of c. Thus, it is sufficient to ensure that $c \geq 0$ to guarantee non-decrease of fairness. Note that $c = 0 \Rightarrow F(x(t+1)) = F(x(t))$, i.e., the fairness stays the same. To ensure convergence to fairness, we require $c > 0$ for either increase or decrease policy. In terms of increase/decrease parameters, this implies

$$\frac{a_I}{b_I} \geq 0 \quad \text{and} \quad \frac{a_D}{b_D} > 0 \tag{8}$$

or

$$\frac{a_I}{b_I} > 0 \quad \text{and} \quad \frac{a_D}{b_D} \geq 0. \tag{9}$$

In (8), the fairness goes up during decrease and either goes up or stays the same during increase. Similarly, (9) ensures that fairness goes up during increase and either goes up or stays the same during decrease. This is sufficient to ensure convergence to fairness. We do not need the fairness to go up during both increase and decrease.

Equations (8) and (9) basically state that a_I and b_I should not be of opposite signs. Similarly, a_D and b_D should not be of opposite signs.

To satisfy (8) or (9), it follows that all four parameters a_I, b_I, a_D, and b_D must be positive, for otherwise $x_i(t)$ can become negative. Also, since n, $\Sigma x_i(t)$, and a_D are all positive, from (3) we know b_D must be less than 1. So

$$a_I \geq 0, \quad b_I \geq 0,$$
$$a_D \geq 0, \quad 0 \leq b_D < 1 \tag{10}$$

where a_I and b_I cannot be both zero, else it would imply zero increase; and a_I and a_D cannot be both zero, else it would imply c is always zero.

2.4. Distributedness

The requirement of having no information about system state other than the feedback $y(t)$ further limits the set of feasible linear controls. Since the fairness requirements (Equation (8) or (9)) do not involve any system state, it already satisfies the distributedness criterion. The ef-

[2] Note that satisfying the negative feedback condition alone only guarantees that the system will oscillate about the efficiency point, but says nothing about the size of oscillation. So this is strictly speaking weaker than the efficiency condition. We will, however, explore how the oscillation size can be minimized when we talk about the optimality of a policy in the next section.

ficiency convergence conditions stated in (3), however, require knowledge of $\Sigma x_i(t)$ and n at each user. In the absence of such knowledge, each user must try to satisfy the negative feedback condition by itself. This means a stronger condition to guarantee convergence to efficiency:

$$y(t) = 0 \Rightarrow x_i(t+1) > x_i(t) \ \forall i,$$
$$y(t) = 1 \Rightarrow x_i(t+1) < x_i(t) \ \forall i. \tag{11}$$

Which translates into

$$a_I + (b_I - 1)x_i(t) > 0 \quad \forall x_i(t) \geq 0,$$
$$a_D + (b_D - 1)x_i(t) < 0 \quad \forall x_i(t) \geq 0.$$

This implies further constraining equation (10) to be

$$\begin{aligned} &a_I > 0, \qquad b_I \geq 1, \\ &a_D = 0, \qquad 0 \leq b_D < 1. \end{aligned} \tag{12}$$

We shall demonstrate these constrains graphically later, using the vector representations.

There is, however, a simple variation for us to make the conditions in (12) less restrictive for parameters b_I and a_D. If each user i truncates its control whenever the conditions in (11) would otherwise be violated, as below

$$x_i(t+1) = \begin{cases} \max(a_I + b_I x_i(t),\ x_i(t)) \\ \quad \text{if } y(t) = 0 \Rightarrow \text{Increase}, \\ \min(a_D + b_D x_i(t),\ x_I(t)) \\ \quad \text{if } y(t) = 1 \Rightarrow \text{Decrease}, \end{cases} \tag{13}$$

then (10) can guarantee both convergence to efficiency with the distributed requirements. There is one catch, however, that is all users could truncate at the same time (thus stopping any progress). To prevent this possibility, let's consider the following conditions:

$$a_I + (b_I - 1)X_{max} > 0,$$
$$N_{max} a_D + (b_D - 1)X_{min} < 0 \tag{14}$$

for some X_{min} and X_{max} satisfying

$$X_{min} \leq X_{goal} \leq X_{max}. \tag{15}$$

Here, N_{max} is the upper bound on the number of users that would share the resource. The claim is that when (14) and (15) are satisfied, it is impossible for $\Sigma x_i(t+1) = \Sigma x_i(t)$.

Let us suppose the contrary is true for the case $y = 0$. This means that

$$a_I + b_I x_i(t) < x_i(t) \quad \forall i$$

which means

$$na_I + (b_I - 1)\Sigma x_i(t) < 0.$$

Since a_I, b_I, and all x_i's are positive, the above inequality is possible only if $b_I - 1$ is negative and if so, substituting X_{max} which is more than $\Sigma x_i(t)$ will make the left-hand side even more negative:

$$na_I + (b_I - 1)X_{max} < 0.$$

This violates (14); thus a contradiction with our assumption.

For the case $y = 1$, if all users truncate, then it means

$$na_D + b_D x_i(t) > x_i(t) \quad \forall i,$$

thus

$$na_D + (b_D - 1)\Sigma x_i(t) > 0.$$

Since b_D is less than 1, the second term in the left-hand side of the above equation is negative. $y = 1$ implies $\Sigma x_i(t)$ is greater than X_{goal}, hence X_{min}. Substituting N_{max} in place of n, and X_{min} in place of $\Sigma x_i(t)$, should maintain the inequality. That is, we must have

$$N_{max} a_D + (b_D - 1)X_{min} > 0.$$

This leads to a violation of (14); thus a contradiction.

So the linear controls with truncation leave us with a set of conditions weaker than (12) and stronger than (10):

$$a_I > 0, \qquad b_I > 1 - \frac{a_I}{X_{max}},$$
$$0 \leq a_D < (1 - b_D)\frac{X_{min}}{N_{max}}, \qquad 0 \leq b_D < 1. \tag{16}$$

Notice that in the case that we do not have any knowledge to bound X_{goal} or n, that simply corresponds to $N_{max} = \infty$, $X_{min} = 0$ and $X_{max} = \infty$. Then the conditions on linear control with truncation reduce to the same ones as those on the strictly linear control. We have essentially proven the following propositions:

Proposition 1. *In order to satisfy the requirements of distributed convergence to efficiency and fairness without truncation, the linear decrease policy should be multiplicative, and the linear increase policy should always have an additive component, and optionally it may have a multiplicative component with the coefficient no less than one.*

Proposition 2. *For the linear controls with truncation (as defined in Equation* (13)*), the increase and decrease policies can each have both additive and multiplicative components, satisfying the constraints in Equations* (16) *and* (15).

The vectorial representation in the next section should help illustrate these results further.

2.5. Vectorial Representation of Feasibility Conditions

The constraint on the control imposed by the efficiency and fairness convergence conditions are depicted in Fig. 7 for the 2-user case. Let us first consider a point in the overloaded region. As shown in Fig. 7(a), the users start at the point $x^H = (x_1^H, x_2^H)$, which is above the efficiency line. The system asks the users to decrease. The line $x_1 + x_2 = x_1^H + x_2^H$ represents an "equi-efficiency" line. All points on this line have the same efficiency as x^H. For convergence to efficiency it is sufficient to ensure that the next decrease moves into the shaded area.

The requirement of linear controls and distributedness puts additional restrictions. Linear controls imply that the new state vector $x(t+1)$ is a sum of two vectors corresponding to a and $bx(t)$. In two dimensions, a vector is represented by a 45° line through $x(t)$. This is shown in Fig. 7(b) by the line marked $b = 1$. All future states corresponding to $b = 1$ lie on this line. Points to the left of the line can be reached if and only if we choose $b > 1$. Similarly, points to the right of the line can be reached if $b < 1$. The second vector corresponding to $bx(t)$ is represented by the line marked $a = 0$ in Fig. 7(b). If we choose $a = 0$, the state $x(t+1)$ will lie on this line. Points to the left of this line can be reached by choosing $a < 0$. Similarly, points to the right of this line can be reached by $a > 0$. Depending upon the values of a and b, the set of reachable states will lie in one of the four regions formed by the two lines $a = 0$ and $b = 1$. Only one of these four regions, the one corresponding to $a \leq 0$ and $b \leq 1$, is completely below the equi-efficiency line. This region is shown shaded in Fig. 7(b). If we choose parameter values corresponding to other regions, the next state can not be guaranteed to be always below the equi-efficiency line.

For fairness, we note that the points between the fairness line and the line passing through x^H have higher fairness than x^H (see Fig. 7(c)). If we locate the mirror image of x^H— the point $x^{H'} = (x_2^H, x_1^H)$—this point has the same fairness as x^H, and all points between the fairness line and the line joining $x^{H'}$ have higher fairness than $x^{H'}$. Thus, for convergence to fairness, it is sufficient that the next point be in the region bounded by the two lines joining origin to the points x^H and $x^{H'}$.

Combining the effect of all the restrictions, the region for distributively converging to efficiency and fairness is given by the intersection of the regions shown in Fig. 7(b), and (c), i.e., by the line joining x^H to the origin as shown in Fig. 7(d). Thus the only policies that would distributively satisfy the fairness and efficiency convergence conditions are those that move the operating point along this line. In other words, the decrease must be multiplicative.

Similarly, starting with a point $x_L = (x_1^L, x_2^L)$ in the underloaded region, the region for distributively converging to efficiency and fairness is given by the region shown in Fig. 7(e).

Equations (12), (16), and (15) are basically the algebraic statement of these conditions.

3. Optimizing the Control Schemes

Having established the feasible control region, the next step is to determine the optimal policy — a policy that takes the system to the goal quickly. In this section, therefore, we discuss the selection of control parameters to minimize the time to convergence and to minimize the oscillations.

3.1. Optimal Convergence to Efficiency

In this subsection, we deal exclusively with the tradeoff of time to converge to efficiency, t_e, with the oscillation size, s_e. More figuratively, we also refer to these two metrics as *responsiveness* and *smoothness*, respectively.

The n state equations corresponding for n users are

$$x_i(t+1) = a + bx_i(t), \quad i = 1, 2, \ldots, n.$$

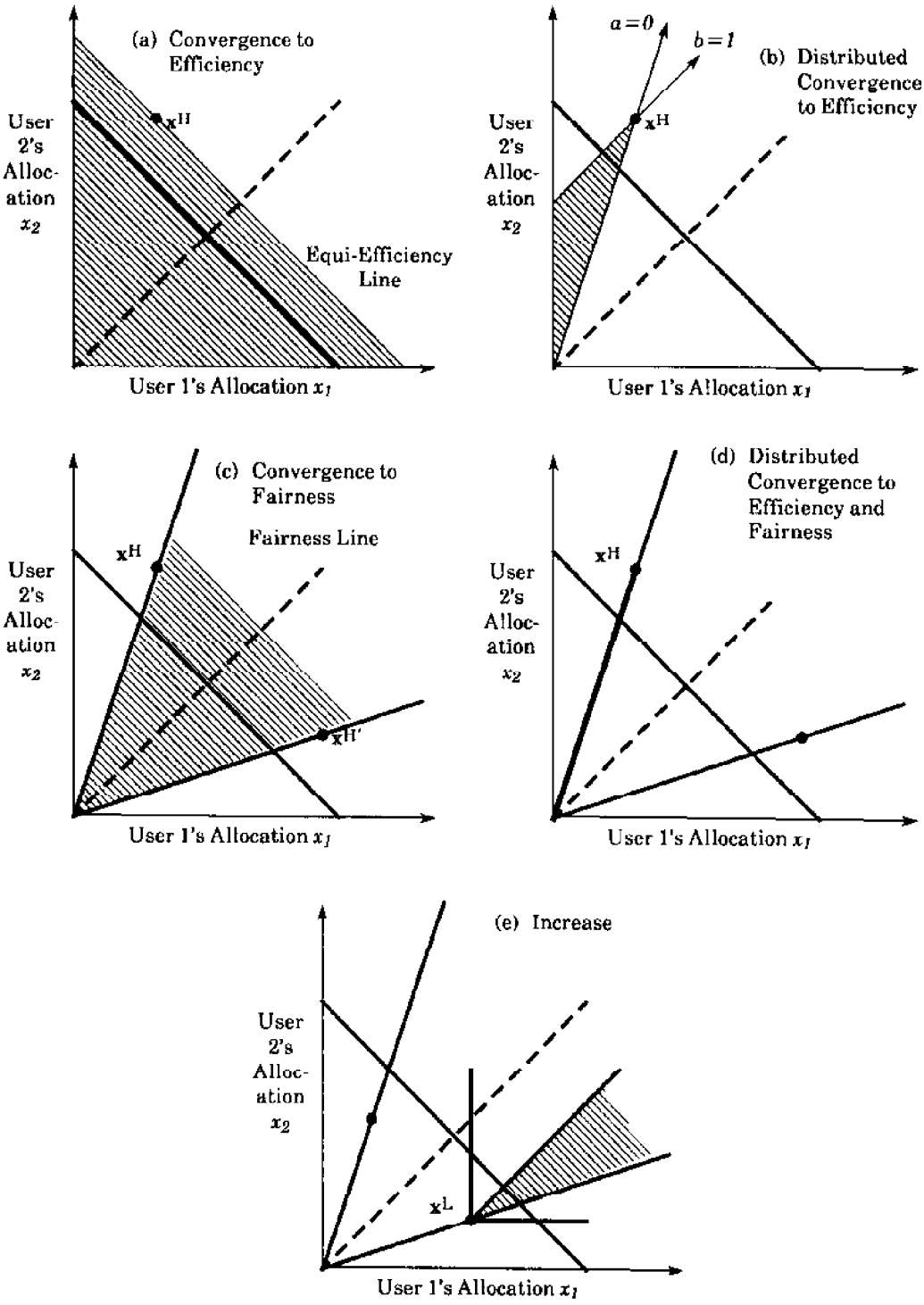

Fig. 7. Vectorial representation of efficiency and fairness feasibility conditions.

These n equations can be added to form a single state equation:

$$\Sigma x_i(t+1) = na + b\Sigma x_i(t)$$

or,

$$X(t+1) = na + bX(t) \quad \text{where } X = \Sigma x_i.$$

Given initial state $X(0)$, the time to reach X_{goal} is

$$t_e = \begin{cases} \dfrac{\log\left(\dfrac{an + (b-1)X_{goal}}{an + (b-1)X(0)}\right)}{\log(b)}, & b > 0, \\ \dfrac{X_{goal} - X(0)}{an}, & b = 0. \end{cases}$$

After converging to X_{goal} there will be a maximum overshoot of

$$s_e = |an + (b-1)X_{goal}|.$$

Notice that t_e is a monotonically decreasing function of a and b, while s_e is a monotonically increasing function of a and b. Thus, any attempt to increase responsiveness (decrease t_e) also results in decreased smoothness (increased s_e), and vice versa.

3.2. Optimal Convergence to Fairness

Equation (7) shows that the per step improvement in fairness $F(x(t-1)) - F(x(t))$ is a monotonically increasing function of $c = a/b$. Thus, larger values of a and smaller values b give quicker convergence to fairness.

For the case of strict linear controls, this leads to an elegantly simple conclusion. For decrease, feasibility conditions required $a_D = 0$. Thus, the fairness remains the same at every decrease step and the parameter b_D has no effect on time to converge to a fair state. For increase, smaller b_I results in quicker convergence to fairness. Thus, the optimal value of b_I is its minimum value—one. (See Equation (12).) Choosing $b_I = 1$ is equivalent to saying that additive increase gives us the quickest convergence to fairness. This result can be formally stated as:

Proposition 3. *For both feasibility and optimal convergence to fairness, the increase policy should be additive and the decrease policy should be multiplicative.*

This is in fact the form of the control we are proposing for our congestion avoidance schemes [10,11].

4. Nonlinear Controls

In this section, we explore the behaviour of certain nonlinear controls. In particular, we show how they can be represented by the vector diagrams as in the case for linear controls. This technique again gives an intuitive feeling about how nonlinear controls work. The detailed analysis of nonlinear controls is beyond the scope of this paper. However, we will explain why we consider such nonlinear controls not suitable for practical purposes.

Let us consider in general state transition equations that are expressible as a power of the state:

$$x_i(t+1) = x_i(t) + \alpha(x_i(t))^k,$$

or, in terms of control,

$$u_i(t) = \alpha(x_i(t))^k$$

where k can be any integer (positive, negative or zero), and α is a normalization constant that defines the step size and sign. Note that $k = 0$ gives the additive policy and $k = 1$ gives the multiplicative policy.

Now let us consider the two user vector representation for these controls. In Fig. 8 we first show the efficiency and fairness lines as before. Consider the point $x^{L'}$. Let $\theta(k)$ be the slope of $u(t)$. Then we know $\theta(0)$ is 45° and $\theta(1)$ is the same as the slope for the initial state $x(t)$. As k tends to infinity, $\theta(k)$ tends to 0° and as k tends to negative infinity, $\theta(k)$ tends to 90°.

Since fairness requires that the slope of the new state $x(t+1)$ be closer to 45° than that of the initial state $x(t)$, we must have $k \leqslant 1$ (negative k is fine).

Slopes for other three possibilites x^L, x^U, and $x^{H'}$ are shown in the figure and can be similarly explained. Considering all four possibilities we see that the feasibility condition requires $k \leqslant 1$ for increase and $k \geqslant 1$ for decrease (with at least one inequality being strict), and appropriate values for α so the sign and step size are correct. The additive increase and multiplicative decrease control, for example, clearly satisfies this general condition.

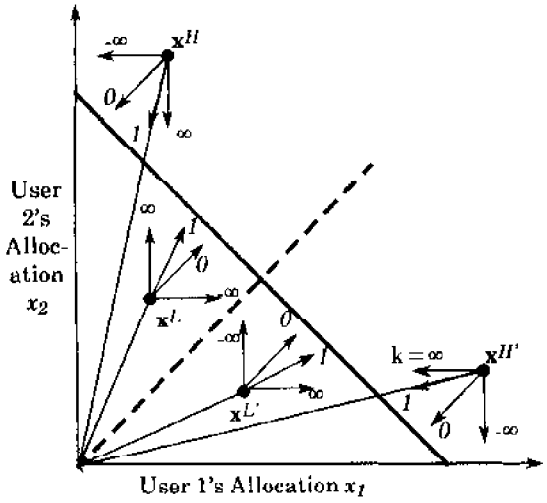

Fig. 8. Vectorial representation for nonlinear controls.

A nonlinear control could generally include more components with different slopes:

$$u_i(t) = \sum_{k=-\infty}^{\infty} \alpha_k (x_k(t))^k.$$

Then the sum of the components must have a slope satisfying the above condition.

Although nonlinear controls offer us far more flexibility in trying to direct towards fairness, it also complicates the task of finding the right scaling factors, represented by α_k in the above equation. These parameters usually must be chosen relative to system parameters, such as the capacity X_{goal} and maximum number of users N_{max}. Being too sensitive to system parameters reduces the *robustness* of the control. For this reason we spent less effort in exploring nonlinear controls. We will discuss the robustness question more in the next section.

5. Practical Considerations

The problem studied in this report is a generic, but also highly abstracted problem. In order to apply the results to solve the decentralized congestion control problem in real networks, many practical issues must be taken into considerations. We briefly discuss some of them here.

One general principle in choosing an algorithm in a general architecture is to be independent of hardware and software *scales* or *parameters* as much as possible. The reason being that in a complex system, scaling parameters are not easily gathered in an automatic fashion and, thus, require human help to *configure*. Having algorithms depend on system scales would complicate the configuration task and make the algorithm more vulnerable to human error. While in this report we have discussed optimizing the control scheme according to certain criteria, practical consideration may dictate that the control be chosen for the widest range of values of system parameters. As we indicated earlier, the nonlinear controls tend to be more sensitive to the system parameters and, thus, are less likely to be useful in practice.

Another possible constraint is that the resource and thus the allocations must be integral. For example buffers and windows are all measured in integers. Simple rounding off to the nearest integer may cause violation of the various convergence conditions.

Ease of implementation could also affect the choice of the controls. For example, the number of multiplications or exponentiations required to implement a control would impact on the minimal hardware required.

5.1. Further Questions

There are many further questions worth exploring in conjunction to this problem. In particular, the following are important:

(1) *How does delayed feedback affect the control?* In practice there is invariably some delays before the feedback arrives at the controller. As the delay lengthens, the feedback becomes less and less useful, and the performance worsens. It would be valuable to have quantitative assessment of how the performance degrades.

(2) *What is the marginal utility of increased bits of feedback?* The binary feedback is the simplest. Adding additional feedback signals may help to cut down the oscillations. A formal analysis would allow an assessment of the tradeoff of performance versus complexity.

(3) *Is it worthwhile to guess the current number of users n?* Users come and go dynamically and the number changes by integral values. A sequence of increase signals may indicate the reduction in the number of users from n to $n-1$. If n were known or bounded, sources could predict

$n-1$ and request that level of resources right away. A wrong guess may make the system less stable. This also deserves further study.

(4) *What is the impact of asynchronous operation?* The algorithm described in the paper is intrinsically a synchronous algorithm. In other words, it is assumed that all users start from an initial state and follow through common phases of computation and feedback. When the frequency of feedback is different or when the time delay of feedback is drastically different, then the convergence properties cannot be guaranteed. This topic is currently under further study.

6. Summary of Results

Congestion avoidance schemes allow a network to operate in the optimal region of low delay and high throughput. This is achieved via the network monitoring its load level and asking the users to increase or decrease the load as appropriate. In this paper, we examined the user increase/decrease policies under the constraints that the feedback from the system is limited to a single bit, which tells whether the current load is above or below the goal.

We formulated a set of conditions that any increase/decrease policy should satisfy to ensure convergence to efficient and fair state in a distributed manner. In particular, we showed that the decrease must be multiplicative to ensure that at every step the fairness either increases or stays the same as that the the current operating point. We derived the sufficient conditions analytically and then explained them using a vector representation.

Optimization considerations require that the change in efficiency and fairness be maximized in every feedback cycle. Using these considerations we showed that additive increase with multiplicative decrease is the optimal policy. This is the policy finally chosen for implementation in the congestion avoidance scheme recommended for Digital Networking Architecture and OSI Transport Class 4 Networks.

Acknowledgment

Many members of the Digital's networking architecture and implementation groups contributed to the congestion avoidance project. The analysis in the paper benefited in particular from the support and input from Tony Lauck, John Harper, and William Hawe.

References

[1] D.P. Bertsekas, Distributed Asynchronous Computation of Fixed Points, *Mathematical Programming* **27** (1983) 107–120.

[2] E. Gafni and D. Bertsekas, Dynamic Control of Session Input Rates in Communication Networks, in: *Proc. MILCOM '82*, Boston, MA, MIT Technical Report LIDS-P-1228, 1982.

[3] H. Hayden, Voice Flow Control in Integrated Packet Networks, MIT, M.S. Thesis, MIT Technical Report LIDS-TH-1152, 1981.

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

[5] J.M. Jaffe, Bottleneck Flow Control, *IEEE Transactions on Communications* **29** (7) (1981) 954–962.

[6] R. Jain, D.M. Chiu and W. Hawe. A Quantitative Measure of Fairness and Discrimination for Resource Allocation in Shared Systems, Technical Report, Digital Equipment Corporation, DEC-TR-301, 1984.

[7] R. Jain and K.K. Ramakrishnan, Congestion Avoidance in Computer Networks with a Connectionless Network Layer: Concepts, Goals and Methodology, in: *Proc. Computer Networking Symposium*, Washington, DC (1988) 134–143.

[8] R. Jain, K.K. Ramakrishnan and D.M. Chiu, Congestion Avoidance in Computer Networks with a Connectionless Network Layer, Technical Report DEC-TR-506, Digital Equipment Corporation, 1987. Also in: C. Partridge, ed., *Innovations in Internetworking* (Artech House, Norwood, MA, 1988) 140–156.

[9] J. Mosely, Asynchronous Distributed Flow Control Algorithms, MIT, Ph.D. Thesis, MIT Technical Report LIDS-TH-1415, 1984.

[10] K.K. Ramakrishnan, D.M. Chiu and R. Jain, Congestion Avoidance in Computer Networks with a Connectionless Network Layer, Part IV-A Selective Binary Feedback Scheme for General Topologies, Technical Report DEC-TR-509, Digital Equipment Corporation, 1987.

[11] K.K. Ramakrishnan and R. Jain, A Binary Feedback Scheme for Congestion Avoidance in Computer Networks with a Connectionless Network Layer, in: *Proc. ACM SIGCOMM'88* (1988).

[12] B.A. Sanders, An Incentive Compatible Flow Control Algorithm for Fair Rate Allocation in Computer/Communication Networks, *6th International Conference on Distributed Computing Systems*, Cambridge, MA (1986).