Additive Increase Appears Inferior

Sergey Gorinsky Harrick Vin

Technical Report TR2000-18
Department of Computer Sciences, University of Texas at Austin
Taylor Hall 2.124, Austin, Texas 78712-1188, USA
{gorinsky,vin}@cs.utexas.edu

May 17, 2000

Abstract

Feedback-based adjustment of load is a common mechanism for resource allocation in computer networks. This paper disputes the popular beliefs that the additive-increase multiplicative-decrease adjustment policy is optimal or even necessary for convergence to fair resource sharing. We demonstrate that, in the classic synchronous model, additive increase does not guarantee the quickest convergence of fairness. Moreover, not only fairness but also efficiency converges very slowly under additive increase. For an asynchronous model, we show that the additive-increase multiplicative-decrease algorithm fails to converge to optimal fairness. We observe that the TCP congestion control algorithm suffers from the problems detected by our analysis and is unfair.

1 Introduction

Plenty of congestion control schemes rely on feedback to achieve efficient and fair resource allocation among network users. Since networks are large distributed systems with dynamic load on resources, feedback is often binary and indicates whether the user can increase or should decrease its load on the network [7]. One of infinitely many strategies for performing such adjustments is an additive-increase multiplicative-decrease algorithm that raises load by a constant and reduces load to a fraction of the current value.

The additive-increase multiplicative-decrease policy has enjoyed wide deployment. For instance, Transmission Control Protocol (TCP) regulates congestion in the Internet by using a mechanism that behaves similarly to the additive-increase multiplicative-decrease algorithm [2, 6]. This outcome can be partly attributed to common beliefs that the additive-increase multiplicative-decrease policy is optimal or even needed for stability or convergence to fairness:

> *"It is has been shown that additive increase/multiplicative decrease is a necessary condition for a congestion control mechanism to be stable."*
> *Larry L. Peterson and Bruce S. Davie, "Computer Networks: A Systems Approach"*
> *Second Edition, October 1999*

Our paper argues that these assertions are false. A congestion control mechanism is stable if it satisfies the principle of negative feedback: load is decreased when it exceeds the target value, and load is increased when it drops below the target. The magnitudes of the adjustments affect the size of the converged interval, not the fact of convergence. Even in the stricter sense (which was probably implied in the quoted statement) of providing stability and converging to fairness, the additive-increase multiplicative-decrease algorithm is neither necessary nor optimal. This paper shows that, in the traditional theoretic model, a different policy can provide quicker convergence of fairness. Moreover, in more realistic models as well as in real networks, additive increase does not ensure convergence to fairness.

The only theoretical justification for favoring the additive-increase multiplicative-decrease algorithm has been suggested in the context of systems providing a feedback that reflects the efficiency of resource utilization. [1] asserts that additive increase guarantees optimal convergence of fairness in a synchronous model of such systems where the users employ the linear-increase linear-decrease algorithm to adjust their load synchronically in response to the feedback indicating the impact of the previous adjustment. In Section 3, our paper provides an example illustrating that an increase policy with a multiplicative component can give quicker convergence of fairness. We also demonstrate that both fairness and efficiency are slow to converge in synchronous systems controlled by the additive-increase multiplicative-decrease algorithm. Section 4 considers a more realistic asynchronous model where the users obtain the feedback after different delays and adjust their load with different frequencies. We show that having an additive component in the increase policy does not ensure convergence to optimal fairness in asynchronous systems. The experimental results presented in Section 5 confirm that TCP suffers from the problems revealed by our analysis. We demonstrate that the TCP congestion control algorithm is unfair and that slow start does not completely alleviate the problem of slow convergence to efficiency.

Before presenting our main results, we first consider quantitative measures of fairness in Section 2.

2 Measures of Fairness

The fairness index [3] and min-max ratio [4] are two quantitative metrics proposed for fairness. Consider a resource shared by n users. Let x_i be an amount allocated to user i. Then the fairness index is defined as

$$F = \frac{\left(\sum\limits_{i=1}^{n} x_i\right)^2}{n \sum\limits_{i=1}^{n} x_i^2} \tag{1}$$

and varies from $\frac{1}{n}$ (total unfairness) to 1 (total fairness), and the min-max ratio equals

$$M = \min_{i,j}\left\{\frac{x_i}{x_j}\right\} \tag{2}$$

and takes values between 0 (total unfairness) and 1 (total fairness). For $n = 2$, these metrics are linked through the following one-to-one correspondence:

$$F = \frac{1}{2} + \frac{M}{1+M^2} \text{ and } M = \begin{cases} 0 & \text{if } F = \frac{1}{2}, \\ \frac{1-2\sqrt{F(1-F)}}{2F-1} & \text{if } \frac{1}{2} < F \leq 1. \end{cases} \tag{3}$$

While the fairness index represents fairness of the resource allocation in general, the min-max ratio reflects fairness as perceived by individual users. For instance, if the allocation for user k is zero, and all the other users receive an equal allocation $x > 0$, then $M = 0$ and $F = 1 - \frac{1}{n}$. When $n \to \infty$, $F \to 1$. Thus, the fairness index can be infinitely close to its optimal value even though user k obviously views this allocation as extremely unfair. If providing a fair service to individual users is an objective, then the min-max ratio is a more appropriate measure of fairness than the fairness index. This paper considers both metrics since the fairness index is adopted by the related work reviewed in the next section.

3 Synchronous Systems

The problem of sharing a resource among cooperative users in a synchronous distributed system is studied in [1]. The load on the resource is changed on a discrete timescale. At every time instant t, each user i adjusts its load to $x_i(t)$ based on local data and a binary feedback $y(t)$ provided by the system. The users do not have access to such information as the states of the other users, number n of the users, optimal load $X_{goal} > 0$ on the resource, total load $X(t)$, fairness index $F(t)$, or min-max ratio $M(t)$:

$$X(t) = \sum_{i=1}^{n} x_i(t), \;\; F(t) = \frac{(X(t))^2}{n \sum_{i=1}^{n} (x_i(t))^2}, \;\; M(t) = \min_{i,j} \left\{ \frac{x_i(t)}{x_j(t)} \right\}. \tag{4}$$

The feedback indicates whether the previous adjustment led to overload of the resource:

$$y(t) = \begin{cases} 1 & \text{if } X(t-1) > X_{goal}, \\ 0 & \text{if } X(t-1) \leq X_{goal}. \end{cases} \tag{5}$$

[1] examines the following linear adjustment of the load

$$\forall i = 1, \ldots, n, \;\; x_i(t) = \begin{cases} a_I + b_I x_i(t-1) & \text{if } y(t) = 0, \\ a_D + b_D x_i(t-1) & \text{if } y(t) = 1, \end{cases} \tag{6}$$

where a_I, b_I, a_D, and b_D are real constants. After a careful investigation, [1] derives conditions ensuring that, for any nonnegative initial load $x_1(0), \ldots, x_n(0)$, the total load $X(t)$ converges into a finite interval around X_{goal} while the fairness index $F(t)$ converges to the optimal value of 1:

$$a_I > 0, \;\; b_I \geq 1, \;\; a_D = 0, \;\; 0 \leq b_D < 1. \tag{7}$$

Hence, one can argue that the linear-decrease policy should be multiplicative while the linear-increase policy should always have an additive component and may have a multiplicative component with the coefficient no less than one.

3.1 Multiplicative Versus Additive

"Additive" and "multiplicative" are often perceived as "conservative" and "aggressive" respectively. This is not always true since, for small $x_i(t)$, additive increase can exceed multiplicative increase. It would be more accurate to refer to multiplicative adjustments as "proportional" while considering additive adjustments as "fixed" or "disproportionate". Choosing an appropriate constant a_I for additive increase is a challenge in dynamic diverse systems such as computer networks. Due to the mix of employed technologies and uneven load patterns in different parts of the network, it is unlikely that the same constant can always supply an acceptable trade-off between convergence time and the size of oscillations after convergence: for some scenarios, the constant can be too small and can lead to slow acquisition of available bandwidth; for other scenarios, the constant amount of additive increase can be too large and can create significant overload.

Let us add Equations (6) for all n users to derive the maximum X_{max} and minimum X_{min} possible total load after convergence of the linear-increase multiplicative-decrease algorithm:

$$X_{max} = n a_I + b_I X_{goal}, \;\; X_{min} = b_D X_{goal} \tag{8}$$

The assumption of arbitrary n and X_{goal} implies a possibility of *unlimited overload* after convergence:

$$Overload = \frac{X_{max} - X_{goal}}{X_{goal}} = \frac{n a_I}{X_{goal}} + b_I - 1 \to \infty \;\text{ when }\; n \to \infty \;\text{ or }\; X_{goal} \to 0. \tag{9}$$

Thus, if the number of users is large or the optimal load value is low, additive increase can create severe overload of the resource. For example, when thousands of Web flows utilize a network link, the per-flow fair share of the link bandwidth can amount to one maximum segment size per round-trip time; under these circumstances, synchronous additive increase in transmission of each flow from one to two maximum segment sizes per round-trip time can burden the link with the load that is twice the link capacity.

On the other hand, the contribution of the multiplicative component to overload is bounded by a constant. No matter how many users share the resource or how low the optimal load value is, the total overload caused by multiplicative increase in the loads of the users stays within the $(b_I - 1)$ factor of the optimal load value.

The threat of unlimited overload could have been prevented by removing the additive component from the increase adjustment. Unfortunately, $a_I > 0$ is required by Conditions (7) to ensure convergence of fairness.

Let us define linear increase to be *multiplicative-additive* if it has an additive component and a multiplicative component with the coefficient greater than one. Below, we compare the additive-increase multiplicative-decrease (AIMD) and multiplicative-additive-increase multiplicative-decrease (MAIMD) algorithms with respect to convergence of fairness, convergence of efficiency, and then convergence of both efficiency and fairness.

In this paper, we refer to a load adjustment policy as *optimal* if it provides the quickest convergence into a target load area. When we reason about convergence of fairness, the target load area is characterized by constraint $F(t) \geq F_{goal}$ or, alternatively, $M(t) \geq M_{goal}$ where F_{goal} and M_{goal} are target values for fairness. When we discuss convergence of efficiency, the target load area is specified by constraints $X_{low} \leq X(t) \leq X_{high}$ where X_{low} and X_{high} are the lower and upper boundaries of a target load interval. The target load area is determined by both the fairness and efficiency constraints when we study convergence to the loaded fair state.

It should be pointed out that if AIMD and MAIMD employ the same values of a_I and b_D, then MAIMD creates bigger oscillations of the total load after convergence. This difference in the converged intervals is unfortunate since it interferes with our desire to provide a completely fair comparison of AIMD and MAIMD. On the other hand, the maximum increment contributed to overload by the multiplicative component is, as we showed above, proportional to the optimal load value and can be contained by an appropriate choice of b_I. In fact, by selecting the multiplicative components b_I^{ma} and b_D^{ma} of MAIMD increase and decrease so that $b_I^{ma} - b_D^{ma} = 1 - b_D^a$ where b_D^a is the multiplicative component of AIMD decrease, the maximum possible magnitudes of load oscillations after convergence of the algorithms can be made equal. Although this setting of the parameters gives the converged intervals of MAIMD and AIMD the same size, it does not provide an unbiased basis for comparison of the algorithms because these converged intervals are positioned differently with respect to X_{goal}. In our paper, we chose to consider MAIMD and AIMD with identical a_I and b_D. Thus, the lower boundaries of the converged intervals coincide while the upper boundary is higher for MAIMD due to the multiplicative component in its increase adjustment. On a side note, having the multiplicative components in both the increase and decrease adjustments is often desirable in order to keep the average total load close to the optimal load value.

3.2 Optimal Convergence of Fairness

[1] asserts that additive increase yields the fastest convergence of fairness. This claim is incorrect. Even though additive increase (i.e., $a_I > 0$ and $b_I = 1$) does maximize $F(t)$ for any load $x_1(t-1), \ldots, x_n(t-1)$ after any single load adjustment step, it does not ensure the best improvement in fairness over a sequence of load adjustments. Under some circumstances, an algorithm with multiplicative-additive increase reaches the target value for the fairness index faster.

Example 1. Consider a system serving two users with initial loads of $x_1(0) = 17$ and $x_2(0) = 0$. Let the optimal load be $X_{goal} = 20$ and the target fairness be $F_{goal} = 99\%$. Then, AIMD with $b_I = 1$, $a_I = 1$, and

$b_D = 0.01$ produces load assignments characterized by fairness

$$F_a(0) = 50\%, \;\; F_a(1) = 55.5\%, \;\; F_a(2) = 60.4\%, \;\; F_a(3) = 60.4\%.$$

At the same time, MAIMD with $b_I = 1.1$, $a_I = 1$, and $b_D = 0.01$ yields

$$F_{ma}(0) = 50\%, \;\; F_{ma}(1) = 55.1\%, \;\; F_{ma}(2) = 55.1\%, \;\; F_{ma}(3) = 99.2\%.$$

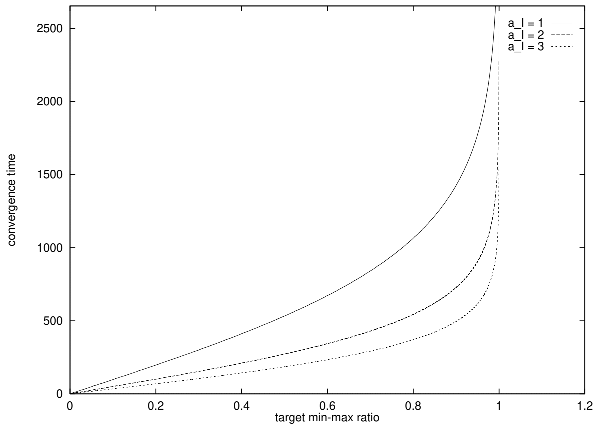
Thus, the policy with multiplicative-additive increase achieves the target fairness $F_{goal} = 99\%$ after three iterations while the policy with additive increase reaches only fairness of 60% . ■

Corollary 1. *Additive increase does not guarantee the quickest convergence of fairness in synchronous systems.*

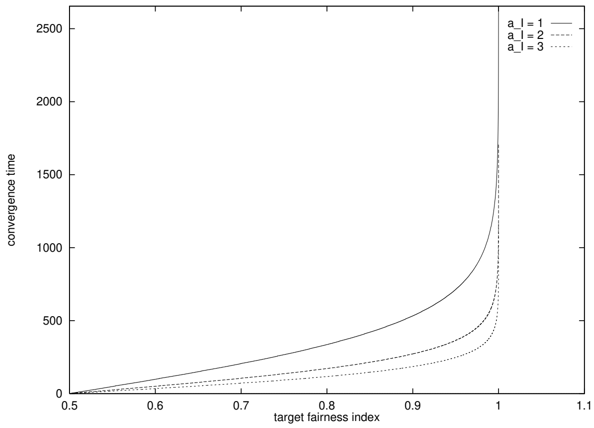
In the presented counterexample to the classic assertion that additive increase provides the fastest convergence of fairness, we used $b_D = 0.01$ to demonstrate that the advantage of MAIMD can be dramatic. Although the improvement in fairness is not so drastic for larger values of b_D or for different settings of other parameters, multiplicative-additive increase still gives quicker convergence in innumerous scenarios. For instance, the choice of $b_D = 0.5$ in the considered counterexample would produce $F_a(3) = 60.4\%$ and $F_{ma}(3) = 62.9\%$. Therefore, by more efficient acquisition of the available resource, multiplicative-additive increase can provide faster convergence of fairness.

Even when AIMD outperforms MAIMD, its convergence of fairness can be very slow. Consider a system controlled by AIMD when $n = 2$, $X_{goal} = 100$, and $b_D = 0.9$. Figure 1 shows time of convergence for the min-max ratio and fairness index from the loaded unfair state, where $x_1(0) = 100$ and $x_2(0) = 0$, for different values of a_I. When $a_I = 1$, the second user attains 90% of the load imposed by the first user (i.e., the min-max ratio and fairness index reach their target values $M_{goal} = 90\%$ and $F_{goal} = 99.7\%$) after 1427 adjustments. It takes 2564 adjustments to achieve the 99% share ($M_{goal} = 99\%$, $F_{goal} = 99.9975\%$).

In the context of networks, such slow convergence of AIMD and MAIMD means that minutes can pass before a flow reaches a relatively fair share of the bottleneck bandwidth. While neither additive increase nor multiplicative-additive increase is optimal in theory, they can fail to support fairness in reality where most of the flows are short-lived.



(a) convergence time for the min-max ratio



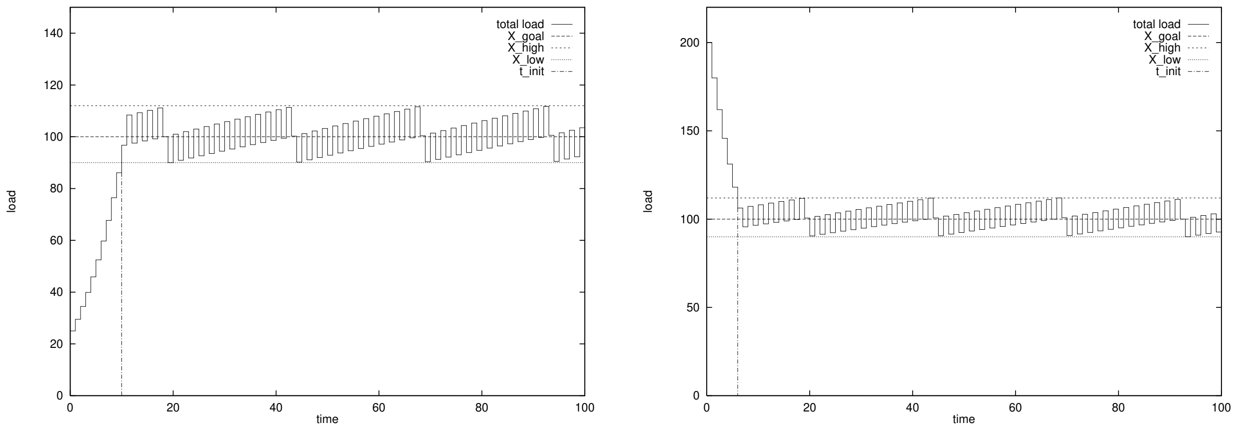
(b) convergence time for the fairness index

Figure 1: Convergence of fairness from the loaded unfair state in a synchronous system controlled by AIMD when $n = 2$, $b_D = 0.9$, $X_{goal} = 100$, $x_1(0) = 100$, and $x_2(0) = 0$.

3.3 Optimal Convergence of Efficiency

While not guaranteeing the fastest convergence of fairness, additive increase gives inferior convergence of the total load. Linear-increase multiplicative-decrease algorithms converge not to the optimal load value X_{goal} but into an interval around X_{goal}. Let us characterize the target load interval by its lower X_{low} and upper X_{high} boundaries: $X_{low} < X_{goal} < X_{high}$. Similarly to [1], we can derive convergence time t_{init} for reaching the target interval from the initial load (see Figure 2):

$$t_{init} = \begin{cases} \left\lceil \log_{b_I} \frac{na_I + (b_I - 1)X_{low}}{na_I + (b_I - 1)X(0)} \right\rceil & \text{if } X(0) < X_{low} \text{ and } b_I > 1, \\ \left\lceil \frac{X_{low} - X(0)}{na_I} \right\rceil & \text{if } X(0) < X_{low} \text{ and } b_I = 1, \\ \left\lceil \log_{b_D} \frac{X_{high}}{X(0)} \right\rceil & \text{if } X(0) > X_{high}, \\ 0 & \text{if } X_{low} \leq X(0) \leq X_{high}. \end{cases} \tag{10}$$



(a) convergence from underload

(b) convergence from overload

Figure 2: Convergence of efficiency for the MAIMD algorithm.

Our goal is to minimize t_{init} under the constraint that the oscillations of the total load after convergence stay within the target load interval:

$$X_{high} \geq X_{max}, \quad X_{low} \leq X_{min}. \tag{11}$$

Taking into account Equations (8), Constraints (11) can be rewritten as:

$$X_{high} \geq na_I + b_I X_{goal}, \quad X_{low} \leq b_D X_{goal}. \tag{12}$$

Let us represent the target load interval by parameters d_I, c_I, and c_D such that

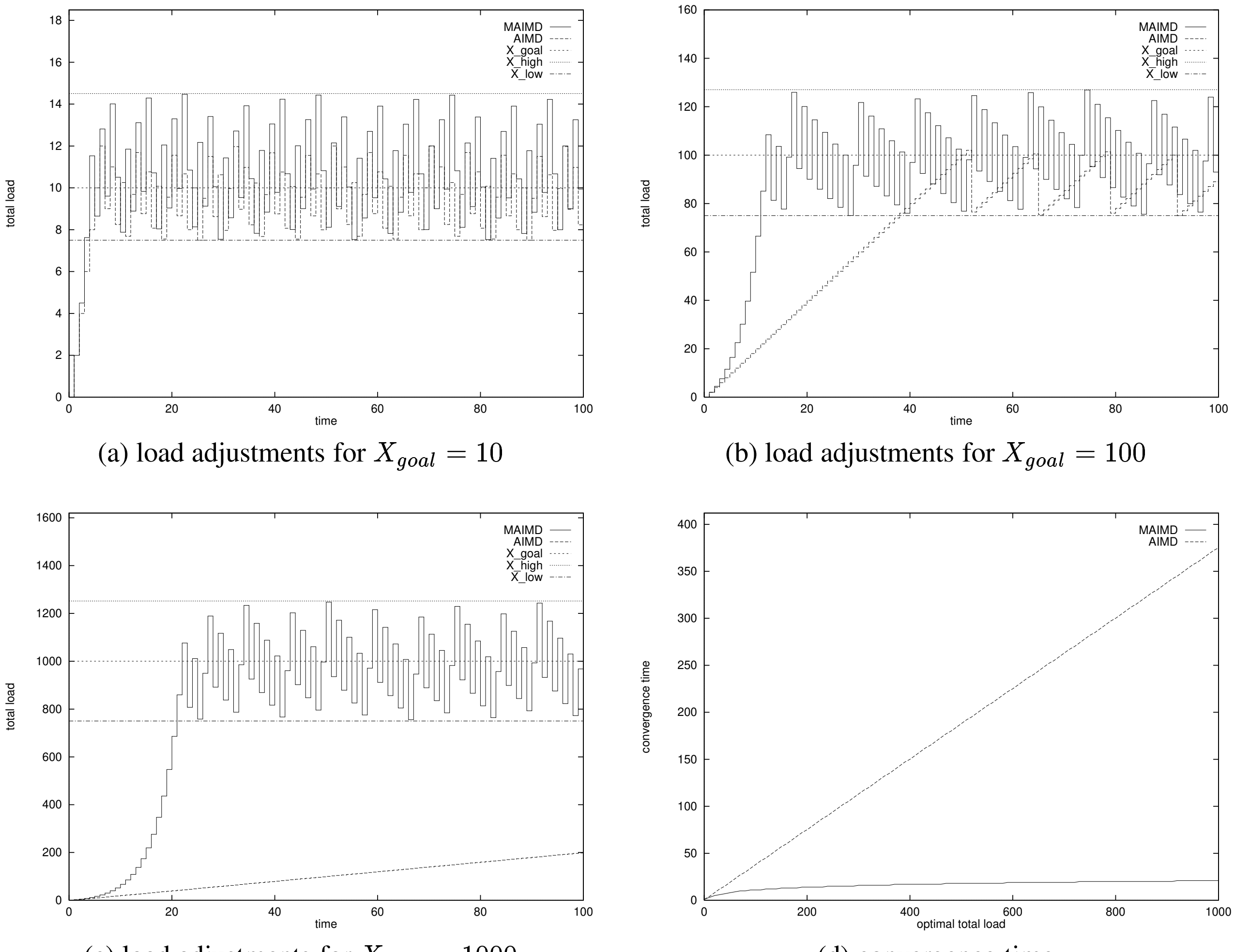
$$X_{high} = nd_I + c_I X_{goal} \text{ and } X_{low} = c_D X_{goal} \tag{13}$$

where $0 \leq c_D < 1$, $d_I > 0$, and $c_I \geq 1$.

The following conclusions can be drawn from Equations (10). First, multiplicative-additive increase provides quicker convergence from underload than additive increase with the same value of a_I: convergence time of MAIMD is logarithmic instead of linear. The difference in performance becomes dramatic when the gap between $X(0)$ and X_{low} is large. Second, b_D should be minimized, subject to Constraints (12), to minimize time of convergence from overload. Taking into account Equations (13), this goal can be achieved by setting b_D equal to c_D. Third, larger a_I and b_I, if conforming to Constraints (12), yield faster convergence from underload. Because either of n and X_{goal} can be arbitrarily high, Conditions (12) and (13) imply $a_I \leq d_I$ and $b_I \leq c_I$. Hence, $a_I = d_I$ and $b_I = c_I$ should be selected.

Proposition 1. *For optimal convergence of efficiency in synchronous systems, the decrease policy should be multiplicative with $b_D = c_D$, and the increase policy should be multiplicative-additive with $b_I = c_I$ and $a_I = d_I$ where c_D, c_I, and d_I are parameters describing the target load interval.*

The following example considers a system where the target load interval is characterized by $c_D = 0.75$, $c_I = 1.25$, and $d_I = 1$. Compare MAIMD employing $b_I = 1.25$, $a_I = 1$, $b_D = 0.75$ and AIMD characterized by $a_I = 1$, $b_D = 0.75$. Figures 3(a), 3(b), and 3(c) show load adjustments performed by these algorithms for different values of X_{goal} when the system serves two users with initial total load $X(0) = 0$. For $X_{goal} = 100$, convergence time for AIMD is four times larger than one for MAIMD. When $X_{goal} = 1000$, AIMD reaches only 25% of X_{goal} after 100 adjustments while MAIMD converges into the optimal interval after 20 adjustments. Figure 3(d) demonstrates that, as the optimal load value grows, linearly increasing convergence time of AIMD greatly exceeds logarithmically increasing convergence time of MAIMD.



(a) load adjustments for $X_{goal} = 10$

(b) load adjustments for $X_{goal} = 100$

(c) load adjustments for $X_{goal} = 1000$

(d) convergence time

Figure 3: Convergence of efficiency for MAIMD and AIMD when $n = 2$, $X(0) = 0$, $b_I = 1.25$, $a_I = 1$, and $b_D = 0.75$.

The oscillations of the total load after convergence can also be described by the maximum time t_{over} of continuous overload and maximum time t_{under} of continuous underload:

$$t_{over} = \left\lceil \log_{b_D} \frac{X_{goal}}{na_I + b_I X_{goal}} \right\rceil, \quad t_{under} = \begin{cases} \left\lceil \log_{b_I} \frac{na_I + (b_I - 1)X_{goal}}{na_I + (b_I - 1)b_D X_{goal}} \right\rceil & \text{if } b_I > 1, \\ \left\lceil \frac{(1-b_D)X_{goal}}{na_I} \right\rceil & \text{if } b_I = 1. \end{cases} \tag{14}$$

Since t_{under} is linear with respect to X_{goal} in the case of AIMD while the dependence for t_{over} is logarithmic, the ratio of underload and overload durations varies widely for different X_{goal}. When the optimal load value is large in comparison with the amount of additive increase, the resource is underloaded most of the time. If the optimal load value is small in comparison with the amount of additive increase, spells of continuous overload prevail.

3.4 Optimal Convergence of Efficiency and Fairness

We showed above that MAIMD is superior to AIMD with respect to time of convergence to efficiency. Ideally, we would like to optimize convergence of the system to the loaded fair state. Figures 4, 5, 6, 7, and 8 compare times of such convergence for MAIMD and AIMD algorithms in a system serving two users. The target load area (denoted on these graphs as *optimal area*) is specified by constraints $F(t) \geq F_{goal}$ and $b_D X_{goal} \leq X(t) \leq 2a_I + b_I X_{goal}$. Figure 4 shows that as the target fairness increases, MAIMD outperforms AIMD for a smaller portion of the initial loads. Figure 5 demonstrates that larger values of b_I do not bring benefits to MAIMD: even though multiplicative-additive increase gives quicker acquisition of the available resource, AIMD gains superior convergence from some lightly loaded states (and the target load area expands). According to Figure 6, changes in a_I affect the balance between MAIMD and AIMD insignificantly: MAIMD still provides quicker convergence only from fair or lightly loaded initial states. Figure 7 manifests an interesting exception from this common pattern: for low values of b_D, MAIMD outperforms AIMD also in the case of loaded unfair initial states. On the other hand, smaller values of b_D decrease the lower load boundary of the target area. Figure 8 shows that the split between the areas of superiority of MAIMD and AIMD remains qualitatively the same over a wide range of the optimal load values.

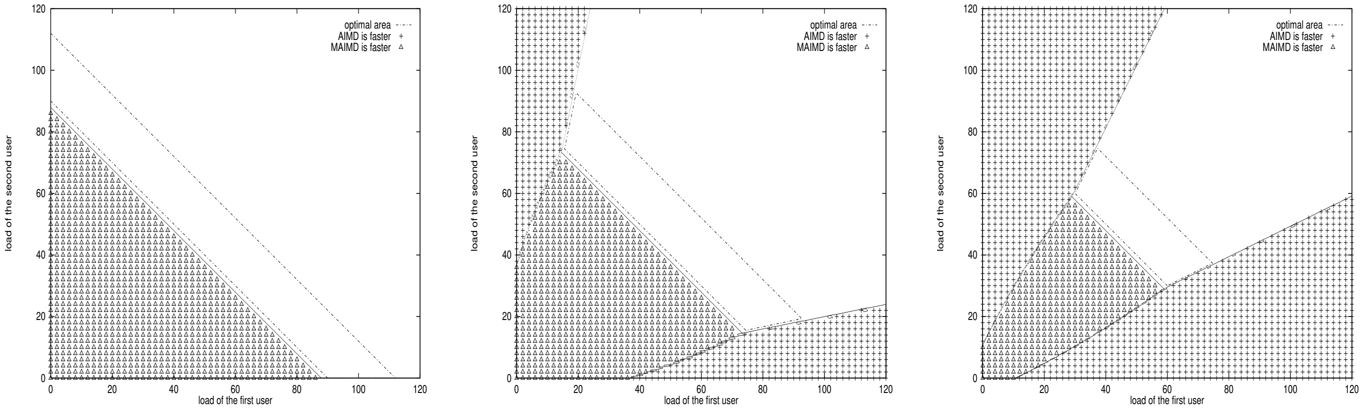
In general, MAIMD provides quicker convergence from fair or lightly loaded initial states while AIMD converges faster (though, very slowly) from loaded unfair initial states.

3.5 Summary

In this section, we compared performance of AIMD and MAIMD algorithms in the classical synchronous model. We discovered that, despite a common belief, AIMD does not guarantee the quickest convergence of fairness. While neither AIMD nor MAIMD is optimal among the linear-increase linear-decrease policies with respect to time of their convergence to fairness, both algorithms converge to fairness very slowly. On the other hand, MAIMD provides the quickest convergence of efficiency. Section 3.1 demonstrated that having an additive component in the increase policy opens a possibility for unlimited overload and thus is undesirable. In terms of the fastest convergence of fairness and efficiency, neither MAIMD nor AIMD is optimal: AIMD usually converges quicker from loaded unfair initial states while MAIMD converges faster from fair or lightly loaded initial states.

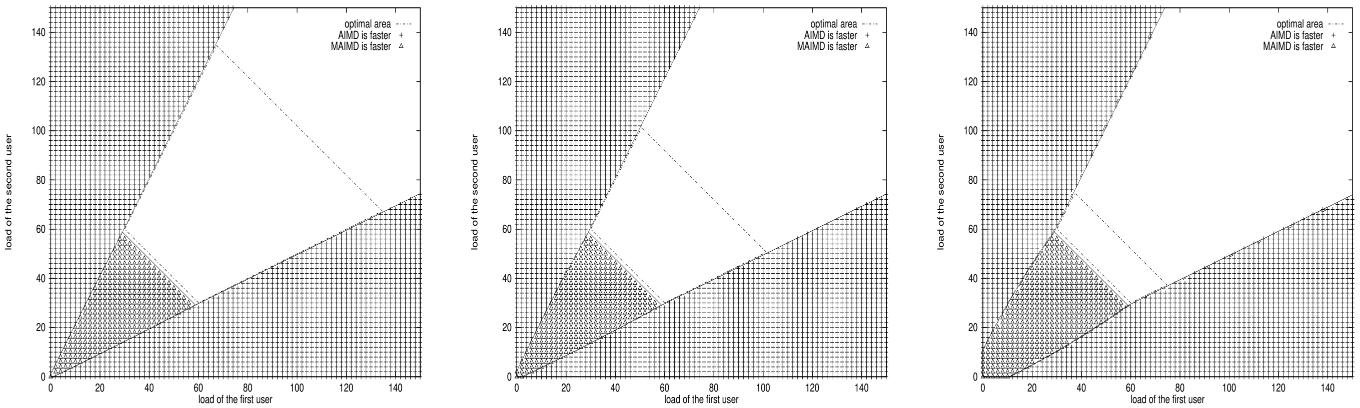
4 Asynchronous Systems

The synchronous model considered in the previous section is not an accurate representation of computer networks. In reality, feedback reaches different users with different delays. Similarly, load adjustments do not affect actual load on the resource simultaneously. Below, we examine a more realistic asynchronous



(a) $F_{goal} = 0.5$ (b) $F_{goal} = 0.7$ (c) $F_{goal} = 0.9$

Figure 4: The influence of the initial load on relative time of convergence for MAIMD and AIMD when $n = 2$, $b_I = 1.1$, $a_I = 1$, $b_D = 0.9$, $X_{goal} = 100$.



(a) $b_I = 2$ (b) $b_I = 1.5$ (c) $b_I = 1.1$

Figure 5: The influence of the initial load on relative time of convergence for MAIMD and AIMD when $n = 2$, $a_I = 1$, $b_D = 0.9$, $X_{goal} = 100$, $F_{goal} = 0.9$.

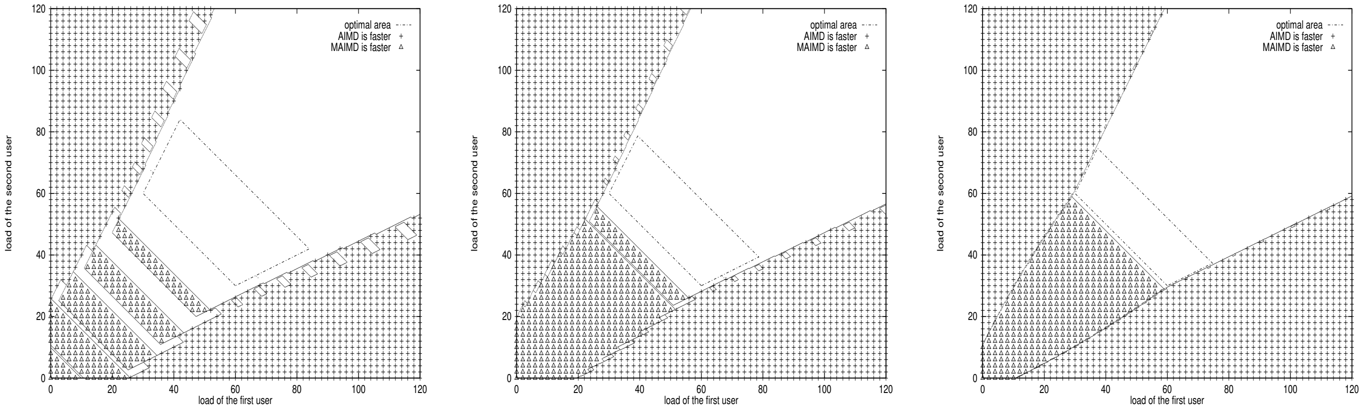
model and demonstrate that, in this model, additive increase does not guarantee convergence to optimal fairness.

The presented asynchronous model is a slight generalization of the synchronous model from Section 3. Two parameters f_i and d_i are added to represent differences in delays and adjustment frequencies: user i adjusts its load $x_i(t)$ with delay f_i after the resource sends feedback; this feedback reflects the previous adjustment conducted d_i time units before the feedback is sent. Thus, user i adjusts its load once per $(d_i + f_i)$ based on the feedback that reflects the efficiency of resource utilization as it was f_i time units before the adjustment. Then, the total load, fairness index, and min-max ratio at time t are as follows:

$$X(t) = \sum_{i=1}^{n} x_i(t - d_i), \quad F(t) = \frac{(X(t))^2}{n \sum_{i=1}^{n} (x_i(t - d_i))^2}, \quad M(t) = \min_{i,j} \Big\{ \frac{x_i(t - d_i)}{x_j(t - d_j)} \Big\}, \tag{15}$$

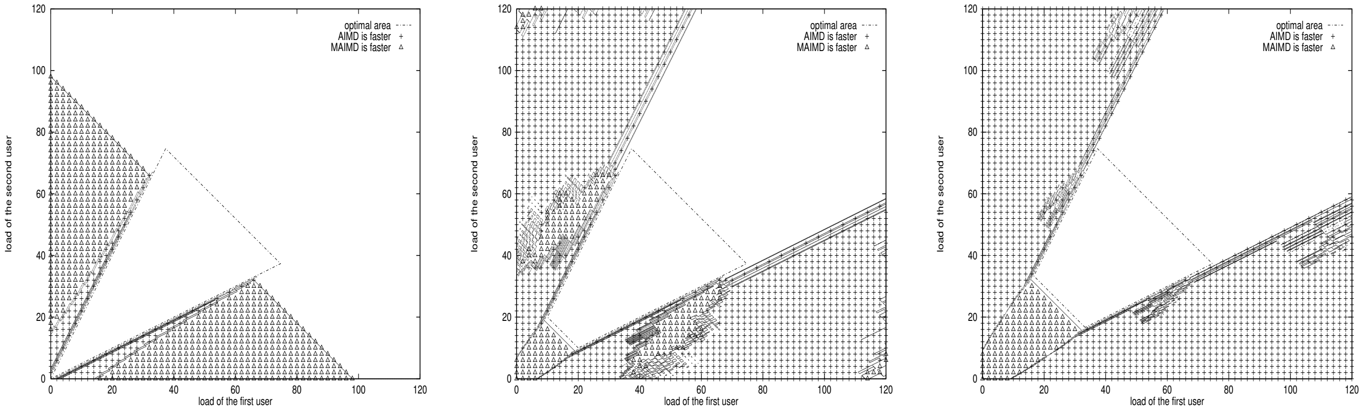
while the feedback indicates whether the resource is overloaded:

$$y(t) = \begin{cases} 1 & \text{if } X(t) > X_{goal}, \\ 0 & \text{if } X(t) \le X_{goal}. \end{cases} \tag{16}$$



(a) $a_I = 8$ (b) $a_I = 4$ (c) $a_I = 1$

Figure 6: The influence of the initial load on relative time of convergence for MAIMD and AIMD when $n = 2$, $b_I = 1.1$, $b_D = 0.9$, $X_{goal} = 100$, $F_{goal} = 0.9$.



(a) $b_D = 0$ (b) $b_D = 0.3$ (c) $b_D = 0.5$

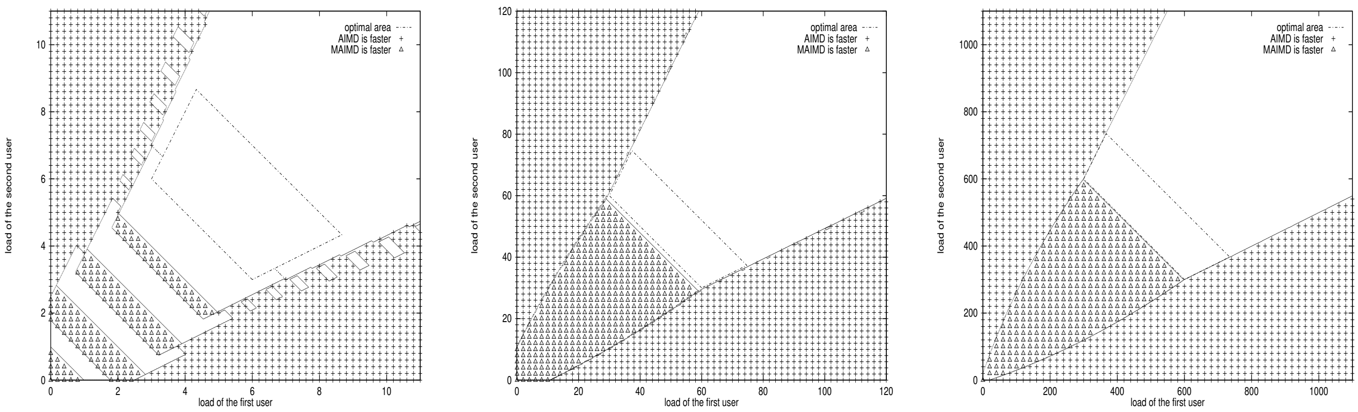
Figure 7: The influence of the initial load on relative time of convergence for MAIMD and AIMD when $n = 2$, $b_I = 1.1$, $a_I = 1$, $X_{goal} = 100$, $F_{goal} = 0.9$.

We examine linear-increase multiplicative-decrease adjustments of the load:

$$\forall i = 1, \dots, n, \;\; x_i(t) = \begin{cases} a_I + b_I x_i(t - d_i - f_i) & \text{if } \; y(t - f_i) = 0, \\ b_D x_i(t - d_i - f_i) & \text{if } \; y(t - f_i) = 1. \end{cases} \tag{17}$$

Example 2. Let us consider an asynchronous system controlled by AIMD. The system has $X_{goal} = 100$ and serves two users. The second user adjusts its load four times more frequently than the first user: $d_1 = 4$, $f_1 = 4$, $d_2 = 1$, $f_2 = 1$. Figure 9(a) shows load adjustments when the initial state is unloaded ($x_1(0) = 0$, $x_2(0) = 0$). In this scenario, the second user captures, due to the higher frequency of its adjustments, a larger share of the resource and keeps it after the system stabilizes. On the other hand, Figure 9(b) shows that, when the initial state is loaded unfair ($x_1(0) = 100$, $x_2(0) = 0$), the first user preserves its bigger portion of the resource even though the second user adjusts its load four times more frequently. Figure 9(c) provides an adjustment diagram which is similar to one introduced in [1]. The diagram shows that, in both scenarios, the system oscillates far from the fair state after stabilization. Moreover, as Figure 9(d) demonstrates, the ranges of the fairness oscillations differ for different initial states. ■

Corollary 2. *The presence of the additive component in the increase policy does not guarantee conver-*



(a) $X_{goal} = 10$ (b) $X_{goal} = 100$ (c) $X_{goal} = 1000$

Figure 8: The influence of the initial load on relative time of convergence for MAIMD and AIMD when $n = 2$, $b_I = 1.1$, $a_I = 1$, $b_D = 0.9$, $F_{goal} = 0.9$

gence to optimal fairness in asynchronous systems.

Unlike in the synchronous model, fairness of the resource sharing in the asynchronous model does not increase monotonically and, if converges, converges into an interval but not to the optimal value. Consider a system in the loaded fair state where each user imposes load $\frac{X_{goal}}{n}$ on the resource. If only one of the users increases its load, the min-max ratio and fairness index drop from their optimal value of 1 to:

$$M = \frac{1}{\frac{na_I}{X_{goal}} + b_I} \text{ and } F = \frac{\left(n - 1 + \frac{na_I}{X_{goal}} + b_I\right)^2}{n\left(n - 1 + \left(\frac{na_I}{X_{goal}} + b_I\right)^2\right)}. \tag{18}$$

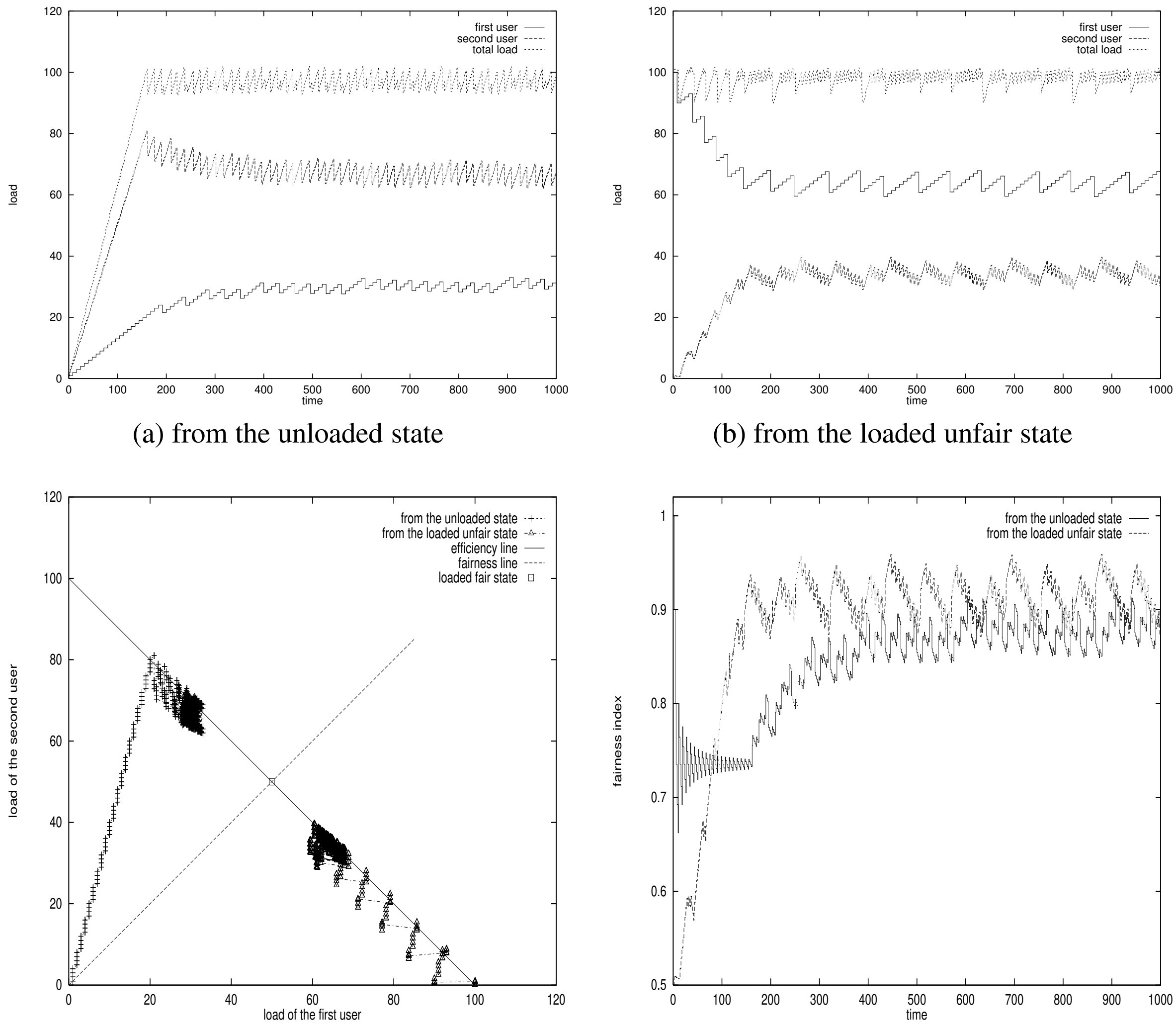
Since $M \to 0$ and $F \to 0$ when $n \to \infty$, $X_{goal} \to 0$, and $a_I > 0$, the assumption of arbitrary n and X_{goal} implies a possibility of *unlimited oscillations of fairness* in the presence of the additive component.

Due to the possibilities of unlimited overload and unlimited oscillations of fairness, it is undesirable to have an additive component in the increase policy. Because the presence of the additive component, as we showed in this section, does not ensure convergence to optimal fairness in asynchronous systems, neither AIMD nor MAIMD appears to be an appropriate mechanism for providing fairness in computer networks. The next section experimentally confirms that the TCP congestion control suffers from the problems detected by our analysis.

5 Additive Increase and TCP Congestion Control

There exists a belief that additive increase and multiplicative decrease of the congestion window during the congestion avoidance mode enable TCP to provide fair bandwidth sharing. The findings of Section 4 suggest that this opinion is incorrect. Indeed, we show below that TCP is not fair. Besides, its reliance on additive increase leads to slow convergence of efficiency in the congestion avoidance mode.

To expedite convergence to efficiency, TCP connections go through a special initial phase called slow start [2]: a new connection increases its congestion window multiplicatively until the first detection of overload; after that, the connection switches to the congestion avoidance mode governed by AIMD. Being a step in the right direction, slow start does not completely alleviate the problem of slow convergence to efficiency. While it accelerates convergence after the arrival of a new connection, slow start can fail to speed up convergence of efficiency in many scenarios when additional bandwidth becomes available. These scenarios include termination of other connections, release of reserved bandwidth, addition of parallel links.



(a) from the unloaded state

(b) from the loaded unfair state

(c) Chiu-Jain adjustment diagram

(d) fairness

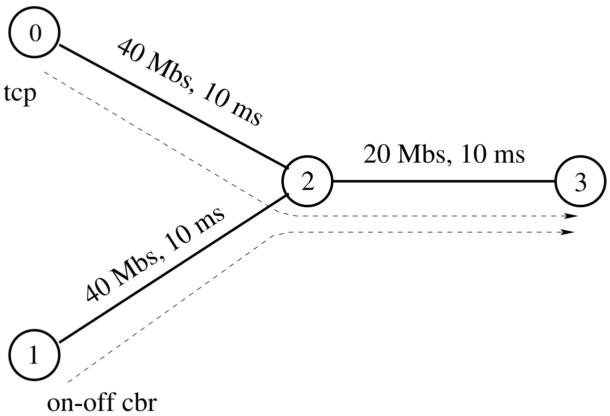
Figure 9: Load adjustments by AIMD in the asynchronous system where the second user adjusts its load four times more frequently than the first user, $n = 2$, $d_1 = 4$, $f_1 = 4$, $d_2 = 1$, $f_2 = 1$, $X_{goal} = 100$, $a_I = 1$, and $b_D = 0.9$.

If such events occur when the existing connections are in the congestion avoidance mode, the connections continue to rely on additive increase, and acquisition of the available bandwidth can be very slow.

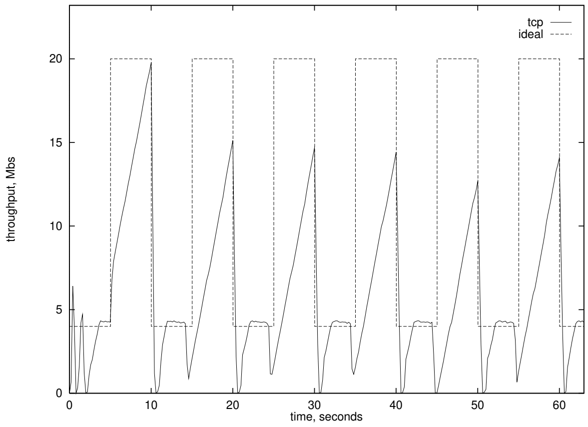
Before presenting our simulation studies, we want to remark that the numerous interdependent mechanisms (such as packet acknowledgments, retransmission timeout, fast retransmit, fast recovery, exponential backoff of the retransmission timer, etc) make the TCP congestion control algorithm quite complex, and it is difficult to single out the behavioral aspects caused exclusively by additive increase. Besides, TCP does not actually employ AIMD but rather behaves, on the round-trip timescale, in a manner that resembles the behavior of AIMD (for instance, the actual increase of the congestion window during the congestion avoidance mode is inverse increase upon receiving an acknowledgment). What this section tries to show is that fairness, provisioning of which is commonly attributed to additive increase, is not a feature of TCP.

We use NS-2 [5] to simulate Reno version of TCP. In these experiments, TCP connections transfer files using the maximum segment size of 536 bytes and compete for the bandwidth of link 2-3. This link has a buffer with capacity that covers the bandwidth-delay products of the simulated TCP connections. If the buffer is full when a new packet arrives, this packet is dropped. The packets are served in the order of their arrival. Computation of throughput is conducted over 200 ms intervals and considers only the packets that reach the destination for the first time.

First, we examine efficiency of TCP. Figure 10 shows the simulated topology and achieved throughput for a TCP connection (denoted as tcp) that competes for bandwidth with an on-off session (denoted as on-off cbr). The bottleneck link 2-3 has a buffer for 100 KB. The on-off session intermits five-second periods of transmission with five-second periods of silence. When on-off cbr transmits, it sends 500-byte packets with constant rate 16 Mbs. Figure 10(b) shows the throughput of tcp as well as the difference (denoted as ideal) between the bottleneck bandwidth and current transmission rate of on-off cbr. By 1.2 seconds into the simulation, tcp abandons its slow start. Since then, the connection operates in the congestion avoidance mode. Because the transmission increases additively in this mode, tcp captures the available bandwidth slowly. The slow convergence of efficiency is especially conspicuous after on-off cbr turns silent – when the session resumes its transmission five seconds later, the throughput of the TCP connection still does not reach the link bandwidth. Hence, additive increase makes convergence of TCP to efficiency slow.



(a) simulated topology

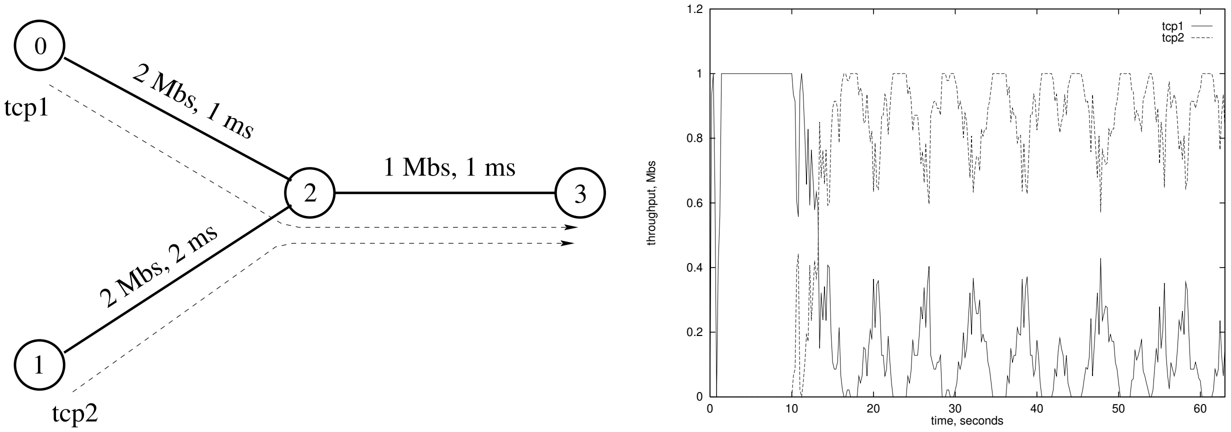


(b) throughput

Figure 10: TCP captures the available bandwidth slowly.

Now, we consider fairness of TCP. Figure 11(a) depicts the simulated scenario: two TCP connections share the bottleneck link 2-3 which has a buffer for 20 KB. The first connection (denoted as tcp1) begins its transmission 10 seconds earlier than the second connection tcp2. By 1.2 seconds into the simulation, tcp1 acquires all the bottleneck bandwidth and switches to the congestion avoidance mode. When tcp2 starts its transmission, link 2-3 is fully utilized by tcp1. By 11.6 seconds into the simulation, tcp2 switches to the congestion avoidance mode. From this moment on, both connections stay in the congestion avoidance mode. Nevertheless, additive increase and multiplicative decrease do not provide convergence to fair bandwidth sharing. Figure 11(b) shows that tcp2 captures and keeps most of the bandwidth on link 2-3 even though the round-trip propagation time for this connection is larger. Thus, TCP is unfair.

This section showed that AIMD does not ensure convergence of TCP to fairness. At the same time, reliance on additive increase makes convergence of TCP to efficiency slow. Since the additive-increase multiplicative-decrease algorithm fails to provide quick convergence of fairness and efficiency in the considered systems, different mechanisms are needed to achieve these goals.



(a) simulated topology (b) throughput

Figure 11: TCP is unfair.

6 Conclusions

This paper disputed the popular beliefs that the additive-increase multiplicative-decrease algorithm is optimal or even necessary for fairness convergence and stability of feedback-based congestion control. We showed that, in a synchronous model of systems where a feedback reflects the efficiency of resource utilization, additive increase does not guarantee the quickest convergence of fairness of resource sharing. This result is interesting because it contradicts an assertion from the classic work by Chiu and Jain. Moreover, both fairness and efficiency are slow to converge in synchronous systems controlled by the additive-increase multiplicative-decrease algorithm. For an asynchronous model, we demonstrated that additive increase does not ensure convergence to optimal fairness. Besides, having an additive component in the increase policy can lead to such undesirable effects as unlimited overload and unlimited oscillations of fairness. The presented experimental results showed that the TCP congestion control algorithm is unfair (despite the common belief that AIMD supplies TCP with convergence to fairness) and that slow start does not assure faster convergence of TCP to efficiency. The fundamental reason for slow convergence of fairness in synchronous systems and for the failure of asynchronous systems to converge to optimal fairness is the attempt to reach fairness relying exclusively on the feedback about the total load. In future, we are planning to design a scalable congestion control scheme that promptly converges to the fair efficient state based on a feedback that reflects both the efficiency and fairness of resource utilization.

References

[1] D. Chiu and R. Jain. Analysis of the increase and decrease algorithms for congestion avoidance in computer networks. *Journal of Computer Networks and ISDN*, 17(1):1–14, June 1989.

[2] V. Jacobson. Congestion avoidance and control. In *Proceedings ACM SIGCOMM'88*, August 1988.

[3] R. Jain, D. Chiu, and W. Hawe. A quantitative measure of fairness and discrimination for resource allocation in shared computer systems. Technical Report TR-301, DEC, September 1984.

[4] M.A. Marsan and M. Gerla. Fairness in local computing networks. In *Proceedings IEEE ICC'82*, June 1982.

[5] Ucb/lbnl/vint network simulator ns-2. http://www-mash.cs.berkeley.edu/ns, January 2000.

[6] L.L. Peterson and B.S. Davie. *Computer Networks: A Systems Approach*. Morgan Kaufmann, second edition, October 1999.

[7] K.K. Ramakrishnan and R. Jain. A binary feedback scheme for congestion avoidance in computer networks with connectionless network layer. In *Proceedings ACM SIGCOMM'88*, August 1988.