Available at
www.ComputerScienceWeb.com
POWERED BY SCIENCE @ DIRECT®

Computer Networks 43 (2003) 227–245

COMPUTER NETWORKS

www.elsevier.com/locate/comnet

Exploiting the efficiency and fairness potential of AIMD-based congestion avoidance and control

Adrian Lahanas [a], Vassilis Tsaoussidis [a,b,*]

[a] College of Computer Science, Northeastern University, Boston, MA 02115, USA
[b] Department of Electrical and Computer Engineering, Democritos University of Thrace, Xanthi 67100, Greece

Received 30 April 2002; received in revised form 20 March 2003; accepted 20 March 2003

Responsible Editor: S.S. Lam

Abstract

Additive increase multiplicative decrease (AIMD) is the dominant algorithm for congestion avoidance and control in the Internet. The major goal of AIMD is to achieve fairness and efficiency in allocating resources. In the context of packet networks, AIMD attains its goal partially. We exploit here a property of AIMD-based data sources to share common knowledge, yet in a distributed manner; we use this as our departing point to achieve better efficiency and faster convergence to fairness.

Our control model is based on the assumptions of the original AIMD algorithm; we show that both efficiency and fairness of AIMD can be improved. We call our approach AIMD with fast convergence (AIMD-FC). We present experimental results with TCP that match the expectations of our theoretical analysis.
© 2003 Elsevier B.V. All rights reserved.

Keywords: TCP; AIMD; Congestion control; Fairness; Efficiency

1. Introduction

One of the most challenging problems in packet networks is resource management. The complexity of multiplexing is accentuated by the variability of packet sizes, resource heterogeneity, protocol autonomy and management anarchy. Consequently, the dynamics of packet networks are frequently associated with a chaotic behavior. In this context, congestion avoidance and control is a vital function of packet networks.

An important characteristic of packet networks is the lack of centralized control and the dominant presence of decentralized authorities (end hosts) that are capable of making decisions pertaining to allocation of resources. Although theoretically all senders are entitled to an ad libitum resource utilization due to lack of a management authority, practically, they are bound by scarce bandwidth resources or high user contention. Therefore, fairness and efficiency are becoming naturally the major design goals of any congestion avoidance and control function.

*Corresponding author. Address: College of Computer Science, Northeastern University, Boston, MA 02115, USA. Tel.: +1-617-373-8169; fax: +1-617-373-5121.
E-mail addresses: ladrian@ccs.neu.edu (A. Lahanas), vtsaoussi@ee.duth.gr, vassilis@ccs.neu.edu (V. Tsaoussidis).

Internet applications are currently governed by the rules of additive increase multiplicative decrease (AIMD), an algorithm proposed by Chiu and Jain in [1] as the most efficient approach to resource management. Jacobson in [2] exploited experimentally the mechanism's potential in TCP [3], integrating AIMD with transmission tactics suitable for congestion avoidance and control. Since then, the algorithm became the major component of TCPs [2] congestion avoidance and control and, in turn, it signified an operational point for the Internet. According to AIMD principles, congestion should trigger a drastic response from the senders (multiplicative decrease) to avoid a congestive collapse, a major concern in packet networks of the last decade. However, the algorithm is designed to be responsive to other network dynamics such as fluctuations of bandwidth availability, and to system characteristics such as the end system autonomy. The former is managed by a continuous probing mechanism through additive increase and the latter is managed by enforcing common behavioral rules to entities that make decisions, namely the data sources.

The goal of each sender is to operate independently but nevertheless to adjust its rate (or window) in a manner that the total bandwidth of the network will be expended *fairly* and *effectively*. From its algorithmic perspective the above problem is challenging because the distributed entities (sources) do not have any prior or present knowledge of the other entities' states; nor do they know the system's capacity and the number of competitors. Hence, the goal of fairness and efficiency appears initially difficult to attain. However, since the system is entitled to a prescribed responsive behavior and the entities agree on common transmission tactics (additive increase when bandwidth is available or multiplicative decrease otherwise), convergence [1] to fairness becomes feasible. We show here that this pattern of behavior contains information that has not been exploited exhaustively in [1].

However, our motivation on the problem is not purely the algorithmic challenge but mainly the mechanism's potential in packet networks. In this context, we note that the description of the above system matches the major properties of packet networks. Consider for example a number of TCP flows m that compete for a limited bandwidth B. The flows' data rates are gradually increasing (additive increase) and the network eventually becomes congested. The network signals the senders about the change of state (from "available" to "congested") and the senders reduce their rate multiplicatively. The ensuing phase of additive increase leads again naturally to congestion and the cycle goes on for as long as the applications have data to send. Indeed, this behavior of TCP conforms [2] to the rules and principles described in [1].

The objective of this behavior is to control the fairness/efficiency tradeoff effectively. That is, the system should allow the active flows to share the channel's bandwidth. The real question therefore, is how to maximize bandwidth utilization and allocate resources fairly. If the protocol windows could grow sufficiently large, bandwidth will be wasted during convergence; convergence here is associated with a multiplicative rate decrease at half the previous window.

By and large, fairness and efficiency (i.e., bandwidth utilization) involve a tradeoff which AIMD attempts to control. For example, an easy way to improve utilization could be to apply a more conservative multiplicative decrease; however, this will cause the system to reach an equilibrium at a later stage, thus degrading fairness. We present an improvement of the AIMD algorithm that impacts positively both efficiency and fairness: efficiency is improved up to 14% and fairness is achieved faster and smoother. However, our contribution does not lie in a new algorithm but rather in an optimization of AIMD during the convergence procedure that enables the algorithm to converge faster *and* achieve higher efficiency.

[1] Convergence to fairness should be perceived in this paper as the procedure which enables different flows that consume different amount of resources each, to balance their resource usage.

[2] Some TCP operations do not strictly follow the AIMD scheme. We discuss them in Section 5.

We call our approach additive increase multiplicative decrease with fast convergence (AIMD-FC).

In the Section 2 we outline the system parameters and metrics that enable a theoretical analysis of the problem in the context of packet networks, and we discuss the results of [1]. In Section 3 we present our proposal and provide proofs for our claims algebraically. In Section 4 we extend the modifications further trying to exploit the system's potential to the maximum extent possible. We also investigate the potential of alternative schemes from the perspective of the dynamics of real networks. We demonstrate the impact of the proposed algorithm on standard TCP in Section 5 where we also present the results from experiments with TCP. In Section 6 we discuss other potential approaches as well as their practical impact. Finally, in Section 7 we highlight some derived conclusions.

2. System model

Our system's model is initially characterized by a synchronous generation of responses, in congruity with [1]. The system response is 1 when bandwidth is available and 0 when bandwidth is exhausted. The instruction to the system entities (sources) is to increase or decrease their data rate, respectively. Note that in real networks, the responsive behavior of the system is not administered by any centralized authority with additional knowledge of the network dynamics—it is simply a packet drop due to congestion that naturally happens when bandwidth is exceeded. Fig. 1 describes the model assumed here and is based on the assumptions of [1].

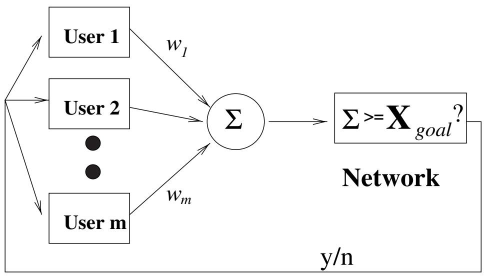


Fig. 1. A control system model of *m* users sharing a network [1].

The system has *m* users (flows) and the instantaneous throughput for the *i*th flow is w_i. The system's goal is to operate at an optimal point X_{goal}. Note that this point is not necessarily the bandwidth B since throughput might decrease before we reach B. We assume that responses are synchronous and consequently the duration of RTTs is common [3] for all flows. Hence, the sources respond uniformly by decreasing their windows in response to a 0 signal; they increase their windows by one in response to a signal of 1. We use the notation a_{I} for the additive increase factor and b_{D} for the multiplicative decrease factor. In the standard AIMD scheme which is also used in TCP, a_{I} is 1 and b_{D} is $1/2$.

The limitations of the system are derived from the dynamics of packet networks: bandwidth B is limited; Each flow is not aware of the throughput rates (window sizes) of other flows; Each flow is not aware of the number of competitors in the channel; no flow is aware of the size of bandwidth B.

In the context of our system behavior we define the following measurement units:

A *cycle* is the phase starting immediately after a system response 0 and ending at the next event of congestion when the system response is again 0. Hence, a cycle includes exactly one multiplicative decrease phase and involves a number of steps through additive increase.

A *step* reflects each window adjustment towards convergence in response to a system instruction (0 or 1). Hence, a step during additive increase involves an increment of one ($a_{\text{I}} = 1$) resource unit (i.e. packet) per flow, and each increase step involves *m* packets more than the previous step (*m* is also the number of flows). In the context of our system, the number of steps matches the number of RTTs.

The convergence behavior of a two flow AIMD system is depicted by vectors in a 2-dimensional space oscillating around the efficiency line in Fig. 2. Upon each multiplicative decrease, the two windows x_1 and x_2 move closer to the fairness line ($x_1 = x_2$). More details on the convergence of AIMD can be found in [1].

[3] The impact of these assumptions has been evaluated in the Internet for several years. We use TCP simulations similarly in the present work.

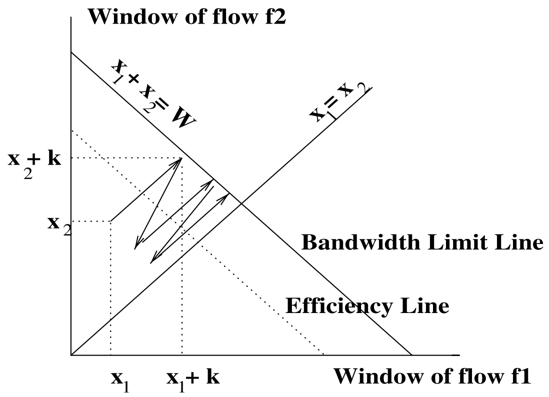


Fig. 2. Vectorial representation of two-flow convergence to fairness. The figure is based on [1]. A point on the quadrant between axes represent the sum of the windows. 'k' denotes the length of the projection of the vector on the x and y axis. W is the value of X_{goal} in terms of packets.

2.1. A pseudocode for AIMD

Based on Fig. 2 we will make a numerical presentation of that convergence behavior and design a possible pseudocode for the AIMD algorithm.

Assume a two-flow system with bandwidth B (or X_{goal}) and let $W = B/\text{MSS}$ (where maximum segment size (MSS) is the packet size) be the maximum number of packets that the system can transmit per step or RTT. Let the flows f_1 and f_2 have x_1 and x_2 initial resources [4] $(x_1, x_2 \in N)$, [5] respectively (see Fig. 2). Without loss of generality, we assume that $x_1 < x_2$ and $x_1 + x_2 < W$ (W coincides with the *bandwidth limit* line in Fig. 2). A simple convergence scenario follows:

$$\begin{array}{ll}
\text{Flow } f_1 & \text{Flow } f_2 \\
x_1 & x_2 \\
x_1 + 1 & x_2 + 1 \\
x_1 + 1 + 1 & x_2 + 1 + 1 \\
x_1 + \underbrace{1 + \cdots + 1}_{k_1} & x_2 + \underbrace{1 + \cdots + 1}_{k_1}
\end{array}$$

$$\sum(w + k) \geqslant X_{\text{goal}} \ \text{State}: \ x_1 + x_2 + 2k_1 \geqslant W$$
$$\text{Action}: \text{Multiplicative Decrease}$$

$$\begin{array}{ll}
\frac{x_1 + k_1}{2} & \frac{x_2 + k_1}{2} \\
\frac{x_1}{2} + \frac{k_1}{2} + 1 & \frac{x_2}{2} + \frac{k_1}{2} + 1 \\
\frac{x_1}{2} + \frac{k_1}{2} + \underbrace{1 + \cdots + 1}_{k_2} & \frac{x_2}{2} + \frac{k_1}{2} + \underbrace{1 + \cdots + 1}_{k_2}
\end{array}$$

$$\sum(w + k) \geqslant X_{\text{goal}} \ \text{State}: \ \frac{x_1}{2} + \frac{x_2}{2} + k_1 + 2k_2 \geqslant W$$
$$\text{Action}: \text{Multiplicative Decrease}$$

$$\begin{array}{ll}
\frac{x_1}{4} + \frac{k_1}{4} + \frac{k_2}{2} & \frac{x_2}{4} + \frac{k_1}{4} + \frac{k_2}{2} \\
\vdots & \vdots \\
\frac{x_1}{2^{\lg x_2}} + \frac{k_1}{2^{\lg x_2}} + \frac{k_2}{2^{\lg \frac{x}{2}}} + \cdots + k_j & \frac{x_2}{2^{\lg x_2}} + \frac{k_1}{2^{\lg x_2}} + \frac{k_2}{2^{\lg \frac{x}{2}}} + \cdots + k_j
\end{array}$$

It can be seen from this numerical example that two flows running the AIMD algorithm will converge to fairness after $1 + \lg x_2$ cycles (the unknown terms $x_1/2^{1+\lg x_2}$ and $x_2/2^{1+\lg x_2}$ become practically insignificant). [6] In general, if the multiplicative decrease parameter is $b_D = 1/\beta$ and x_1 and x_2 are the initial windows of two flows then these flows will converge to fairness in $\log_\beta(\max(x_1, x_2)) + 1$ cycles [7] or $O(W \log_\beta W)$ steps because of the linear increase.

Based on the above example, below we present a pseudocode for AIMD algorithm and an example of a distributed algorithm for an AIMD-based system of m flows. A new feature of this pseudocode is that it distinguishes the amount by which the window of the flow has increased during additive increase phase. We symbolize this amount as 'k' and it can be easily noticed in Fig. 2 and in the above example. Resources consumed by the flows (i.e. congestion window) are represented by the vector/tuple $(w_1, w_2, \ldots, w_m)$, where the ith element of the vector represents the congestion window of flow i. Note that the AIMD-System's pseudocode

[4] In the context of TCP this is the congestion window [2].

[5] The example is based on the assumption that the window value is an integer and there is a minimum divisible unit (i.e., a packet, a segment, or a byte).

[6] Practically, it can happen that $x_2 \simeq x_1$ in which case the difference of the unknown terms is insignificant. Alternatively, we consider the case where $x_2 \gg x_1$ or vice versa. In this case multiplicative decrease will be executed several more times (i.e. $\log_\beta(x_2) + 1$ cycles) which leads again to an insignificant difference. Also note that the windows of the corresponding flows take only integer values.

[7] Recall that a cycle consists of several additive increase steps and one multiplicative decrease step.

is used to describe the system behavior and is not executed by one single entity.

```
AIMD($w$)
1  $k \leftarrow 0$
2  while (feedback is 1)
3     process($w + k$)
4     $k \leftarrow k + a_I$
5  end
6  $w \leftarrow b_D(w + k)$
7  return $w$
END
AIMD-System($(x_1, x_2, \dots, x_m), m, n$)
1  forall $i = 1$ to $m$ do
2     $w_i \leftarrow x_i$
3  end
4  for $j = 1$ to $n$ do
5     forall $i = 1$ to $m$ do
6        $w_i \leftarrow$ AIMD($w_i$)
7     end forall
8  end
9  return $(w_1, w_2, \dots, w_m)$
END
```

We use the following notation:

x_i	Initial window of flow i
k	Resources consumed by additive increase
w	The value of the window immediately after the multiplicative decrease
$w + k$	Current window of a flow ($k \geqslant 0$)
n	Integer. Represents the number of cycles towards convergence
m	Integer. Represents the number of flows
a_I	The additive increase rate ($a_I = 1$)
b_D	The multiplicative decrease ratio ($b_D = 1/2$)

The function 'process($w + k$)' can perform any transport protocol related function (e.g. transmit $w + k$ bytes/packets/segments). The AIMD-based system converges to fairness when the elements of the returned tuple $(w_1, w_2, \dots, w_m)$ become equal. Based on that observation, we define fairness in the context of the present work.

Definition 1. A system of m flows $S(f_1, f_2, \dots, f_m)$, where f_i is the flow i and w_i is its corresponding instantaneous throughput, converges to fairness in n cycles if $w_1, w_2, \dots, w_m$ become equal exactly at the nth cycle.

The proof of the AIMD pseudocode follows from the numerical convergence example shown previously.

2.2. Observations on the dynamics of AIMD

Extending the above discussion, we highlight four observations and we arrive at one conclusion that constitutes the foundation of the present work:

1. When the flows f_1 and f_2 are in additive increase phase they move parallel to the 45° axis ($x_1 = x_2$) or fairness line; during that phase, equal amount of system resources is being allocated to the flows. This amount ('k') is a public or *common knowledge* (i.e., it is known to every flow in the system) [4].
2. Both the initial windows and the amount of system resources (k) that has been fairly allocated during additive increase are affected by multiplicative decrease. That is, the original AIMD scheme calls for adjustments of the current windows—not the initial windows. Note that the manipulation of the initial (and unknown) windows is the real target for achieving fairness.
3. The distance between the bandwidth limit line and the efficiency line when the system is in equilibrium [8] depends only on the multiplicative decrease factor [1]. The closer the efficiency line to the bandwidth limit line, the higher the bandwidth *utilization* of the algorithm [1]. In general, when the decrease factor is $b_D = 1/\beta$ the bandwidth utilization is
$$\frac{\beta + 1}{2\beta}. \tag{1}$$
4. Two algorithms may need the same number of cycles to converge to fairness: for example, two variants of AIMD with different additive increase rate but the same multiplicative decrease ratio. In such case, the number of steps determines the relative efficiency of the algorithm to converge to fairness. In the context of packet networks, this is associated with the

[8] Due to the system dynamics of continuous adjustments even after fairness has been achieved, the term "steady state" is avoided.

frequency at which we meet the bandwidth limit line and adjust backwards.

Practically, fairness is achieved with AIMD gradually by releasing (through multiplicative adjustments of the windows) the (unknown to other flows) initial resources x_1 and x_2; during the additive increase phase the flows increase their resource consumption uniformly.

3. AIMD-fast convergence algorithm

3.1. AIMD-FC goals and metrics

The question of efficiency is associated with the utilized bandwidth; at a first glance, the system dynamics suggest that the higher the oscillation the less the utilization. It also appears [9] that the higher the oscillation, the faster we approach fairness. Some recent versions of AIMD-based algorithms [5–7] attempt to take advantage of this property. More precisely, it has been observed that streaming applications could benefit from modest oscillations since these reflect the smoothness of adjusting the transmission rate backwards. Such protocols are characterized as TCP-friendly because they consume the same amount of bandwidth as TCP$(1, 1/2)$ does [8]. However, an undesirable property of these algorithms is that, a system of TCP-friendly flows reaches equilibrium later than AIMD. For example, a system where all flows use the GAIMD [5] algorithm with $\beta = 8/7$, ($b_D = 1/\beta$) converges to fairness in $O(W \log_{1.14} W)$ steps, where W is the capacity of the system in terms of packets; the SIMD [7] system with $b_D = 15/16$ converges in $O(W \log_{1.06} W)$ steps; the SQRT [6] system with $\beta = 1/(1 - \sqrt{1/W})$, converges in $O(W^2)$; so does the IIAD [6]. The performance of AIMD algorithm (with parameters $a_I = 1$ and $b_D = 1/2$) and in general of TCP is studied in [9,10]. In terms of throughput, the performance of TCP is $(\mathrm{MSS}/\mathrm{RTT})(3/4)W$ [10], which implies 75% bandwidth utilization when the system is in equilibrium, or in terms of a *periodic* packet loss probability p the average throughput is

$$\frac{\mathrm{MSS}}{\mathrm{RTT}} \frac{C_1}{\sqrt{p}} \tag{2}$$

where $C_1 = \sqrt{3/2}$ [10]. Several congestion control algorithms, along with an analysis of their convergence behavior are discussed in [11].

Although system utilization is a well-defined metric, it cannot characterize a congestion control algorithm's performance alone. From our perspective, such an algorithm needs to be evaluated in three more aspects. The first represents the question whether the algorithm converges or not. The second characterizes the speed to reach convergence and is frequently called *responsiveness* [1]. The third involves the question whether the level of window oscillations is high during the convergence procedure. This is commonly called *smoothness* [1].

Therefore, the metrics that we use to evaluate the performance of the algorithms are defined as follows:

Efficiency which is the average flows' throughput per step (or RTT) when the system is in equilibrium over the system theoretical throughput.

Responsiveness which is measured by the number of steps (or RTTs) to reach an equilibrium (i.e. to equate the windows).

Smoothness is reflected by the magnitude of the oscillations during multiplicative decrease.

The modifications presented here do not favor efficiency at the cost of fairness; nor do they favor smoothness at the cost of responsiveness, or vice versa. Therefore, we do not attempt here to balance trade of a_I and b_D within the frame of TCP limitations (i.e. efficiency) and application requirements (i.e. smoothness) but instead we attempt to improve efficiency of TCP *without* degrading its fairness potential. In this context, the goal of the present work is in marked distinction with the TCP-friendly protocols which take an application-centric perspective, since they favor smoothness at the expense of responsiveness in order to satisfy the requirements of multimedia applications without damaging TCP flows. Indeed, the properties we exploit here can be beneficial for TCP-friendly protocols as well.

[9] Both statements have been made initially in [1].

3.2. AIMD-FC

The rationale of AIMD-FC is based on the four observations of Section 2.2. During additive increase the system resources are both well-known and fairly allocated. Hence, at the phase of multiplicative decrease these resources need not be affected.

Assume that two flows (Flow f_1 and Flow f_2) at time t enter the system with windows x_1 and x_2 ($x_1 < x_2$ and $x_1 + x_2 < W$—see Fig. 2). The flows start consuming resources (additively) from the system and at time $t + \delta t$, the system notifies the flows to release resources ($x_1 + x_2 + 2k \geqslant W$). Since both flows f_1 and f_2 evolve with the same additive increase parameter, from time t to time $t + \delta t$ they consume exactly k resource units, each. When the system resources are exhausted the flows essentially release resources (i.e. multiplicative decrease) from the initial windows x_1 and x_2 which were allocated unfairly. So, our algorithm suggests to decrease multiplicatively (to half the previous size) the windows x_1 and x_2 alone.

The algorithm of an AIMD-FC-based system can be described as follows:

```
AIMD-FC($w$)
1  $k \leftarrow 0$
2  while (feedback is 1)
3     process($w + k$)
4     $k \leftarrow k + a_I$
5  end
6  $w \leftarrow b_D(w) + k$
7  return $w$
END
```

```
AIMD-FC-System($(x_1, x_2, \ldots, x_m), m, n$)
1  forall $i = 1$ to $m$ do
2     $w_i \leftarrow x_i$
3  end
4  for $j = 1$ to $n$ do
5     forall $i = 1$ to $m$ do
6        $w_i \leftarrow$ AIMD-FC($w_i$)
7     end forall
8  end
9  return $(w_1, w_2, \ldots, w_m)$
END
```

It is becoming apparent that the distinctive difference of AIMD and AIMD-FC is centered on the portion of the congestion window that is affected by multiplicative decrease. We call this portion *decrease window*.

Definition 2. Decrease window is that portion of the congestion window that is multiplied by the decrease coefficient b_D. Example: w is the decrease window and $(w + k)$ is the congestion window, respectively, of AIMD-FC; $(w + k)$ is both the decrease window and the congestion window of AIMD function.

3.3. Correctness, efficiency and system limitations

3.3.1. Correctness

Theorem 1 (AIMD-FC convergence theorem). *Let x_1, x_2 ($x_1, x_2 \in N$) denote the initial states of resources/windows of two flows and n ($n \in N, n > 0$) the number of cycles completed towards convergence to fairness.*

If AIMD-System($(x_1, x_2), 2, n$) converges to fairness, then AIMD-FC-System($(x_1, x_2), 2, n$) converges to fairness.

Proof by contraction. We prove the correctness for two flows, but it can be generalized for many flows similarly. Let n be such that AIMD-System($(x_1, x_2), 2, n$) converges to fairness in exactly n cycles and assume that AIMD-FC-System($(x_1, x_2), 2, n$) does not converge after n cycles. Unrolling the window values returned by AIMD-FC after n cycles we have

$$\psi_1 = \frac{x_1}{2^n} + \frac{k_1}{2^{n-1}} + \cdots + \frac{k_{n-1}}{2^2} + k_n, \tag{3}$$

$$\psi_2 = \frac{x_2}{2^n} + \frac{k_1}{2^{n-1}} + \cdots + \frac{k_{n-1}}{2^2} + k_n. \tag{4}$$

Performing the same unrolling for the windows returned by AIMD algorithm we will have

$$\phi_1 = \frac{x_1}{2^n} + \frac{k_1}{2^n} + \cdots + \frac{k_{n-1}}{2^2} + \frac{k_n}{2}, \tag{5}$$

$$\phi_2 = \frac{x_2}{2^n} + \frac{k_1}{2^n} + \cdots + \frac{k_{n-1}}{2^2} + \frac{k_n}{2}. \tag{6}$$

The terms on the right of x_1- and x_2-fractions of both series are equal. AIMD-System converges to fairness if and only if $x_1/2^n$ and $x_2/2^n$ are insignificant to the total sum of ϕ_1 and ϕ_2 (we consider this to happen when $n \geqslant 1 + \lg(\max(x_1, x_2))$). That is, if AIMD-FC-System$((x_1, x_2), 2, n)$ has not converged after n cycles then the fractions $x_1/2^n$ and $x_2/2^n$ have a considerable value. Consequently, the x_1- and x_2-fractions of AIMD series (i.e. ϕ_1 and ϕ_2 series) have a considerable value. This contradicts the hypothesis that AIMD-System$((x_1, x_2), 2, n)$ converges in n cycles. □

From Theorem 1 we conclude that the AIMD-FC algorithm converges to fairness in O(lg W) cycles or O(W lg W) steps, where W is the system's capacity in packets.

3.3.2. *Efficiency of AIMD-FC*

In Fig. 3 we present the window evolution of AIMD-FC algorithm and below we calculate its average throughput (per step or RTT). First we calculate the minimum value of congestion window.

Assume a single-flow system with bandwidth B and let W be the the system's capacity in packets. Let $w = yW$ be the decrease window for this flow. After the multiplicative decrease (i.e. when $w + k \geqslant W$), $(y/2)W$ resources will be released [10] and is evident that $(y/2)W$ will be allocated again during additive increase (i.e. $k = (y/2)W$). So, $w + k = yW + (y/2)W = W$ which implies that $y = \frac{2}{3}$. Hence, the decrease window (and the minimum congestion window) value will be $\frac{2}{3}W$ and the additive increase space will be $\frac{1}{3}W$ (see Fig. 3).

Using the same method of AIMD analysis in [10], but based on the window evolution of Fig. 3, the average throughput (per RTT) of this AIMD-FC flow is

[10] From the definition of the algorithm.

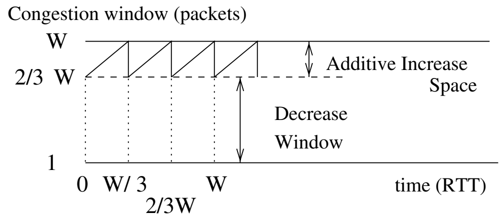


Fig. 3. AIMD-FC window evolution under periodic loss. Each cycle delivers $(\frac{2}{3}W)(\frac{1}{3}W) + \frac{1}{2}(\frac{1}{3}W)^2 = 1/p$ packets and takes $W/3$ round trip times.

$$\frac{\text{MSS}}{\text{RTT}} \frac{5}{6} W \tag{7}$$

or, in terms of a periodic packet loss probability p the average throughput of this flow is

$$\frac{\text{MSS}}{\text{RTT}} \frac{C_2}{\sqrt{p}} \quad \text{where } C_2 = \sqrt{\frac{5}{2}}. \tag{8}$$

The efficiency of AIMD-FC is 8% higher than the efficiency of AIMD.

3.3.3. *Smoothness*

In the above paragraph we saw that the congestion window oscillation margin is $[\frac{2}{3}W, W]$. Obviously, the relative smoothness of AIMD-FC is increased by 33%.

3.3.4. *Responsiveness*

One important observation of the authors in [1] is that the *smoothness* and *responsiveness* of AIMD are in inverse proportion (they depend only on the increase a_I and decrease b_D parameters). This observation still holds for AIMD-FC. Comparatively, responsiveness of AIMD-FC, improves.

Theorem 2 (AIMD-FC responsiveness theorem). *Let* x_1, x_2 ($x_1, x_2 \in N$; $0 < x_1 < x_2$) *denote the initial states of resources/windows of two flows,* n ($n \in N, n > 0$) *the number of cycles in which AIMD-System*$((x_1, x_2), 2, n)$ *converges to fairness, and let* p ($p \geqslant n$) *denote the number of additive increase steps needed by AIMD-System*$((x_1, x_2), 2, n)$ *to converge to fairness.*

If AIMD-System$((x_1, x_2), 2, n)$ *converges to fairness in p steps, then AIMD-FC-System*$((x_1,x_2), 2, n)$ *converges in strictly less than* $p - (n-1)$ *steps.*

Proof by induction. Induction statement: For every cycle j ($j > 1$), AIMD-FC involves at least one additive increase step less than AIMD.

Basis step. After the second cycle ($j = 2$) AIMD-FC gains at least one step towards convergence.

Let two flows have initial windows x_1 and x_2 such that $0 < x_1 < x_2$ and $x_1 + x_2 < W$. Unrolling the series generated by each algorithm we estimate the gain of AIMD-FC after each exponential decrease. The letters k_i and k_i' represent the resources allocated during the additive increase phase. During each additive increase step, the two flows increase their resource consumption by $2a_I$ or 2 units ($a_I = 1$). After k_1 steps $\sum(w+k) \geqslant X_{\text{goal}}$ is satisfied by both algorithms.

$\sum(w+k)$ (AIMD)	$\sum(w+k)$(AIMD-FC)
$x_1 + x_2 + 2k_1 \geqslant W$	$x_1 + x_2 + 2k_1 \geqslant W$

$\sum(w+k)$ state after multiplicative decrease

$\gamma_1 = \frac{x_1}{2} + \frac{x_2}{2} + k_1 \geqslant \frac{W}{2}$ $\qquad \frac{x_1}{2} + \frac{x_2}{2} + 2k_1 > \gamma_1$

There is no gain in the first cycle (i.e. $j = 1$). Both algorithms start with the same initial values and reach the same limit. Second cycle ($j = 2$):

$$\textstyle\sum(w+k)\text{(AIMD)}: \quad \gamma_1 = \frac{x_1}{2} + \frac{x_2}{2} + k_1 \geqslant \frac{W}{2}$$

$$\textstyle\sum(w+k)\text{(AIMD-FC)}: \quad \gamma_1 < \frac{x_1}{2} + \frac{x_2}{2} + 2k_1 < W$$

k_2 and k_2' resources are consumed during additive increase. $\sum(w+k)$ State:

$$\text{AIMD:} \quad \frac{x_1}{2} + \frac{x_2}{2} + k_1 + 2k_2 \geqslant W$$

$$\text{AIMD-FC:} \quad \frac{x_1}{2} + \frac{x_2}{2} + 2k_1 + 2k_2' \geqslant W$$

Since $\gamma_1 < x_1/2 + x_2/2 + 2k_1 < W$, the number of additive increase steps with AIMD-FC at the second cycle is less than the number of increase steps with AIMD. So $k_2' < k_2$. This means that during the second cycle, AIMD-FC requires at least one increase step less than AIMD prior to decreasing. [11]

Inductive step. If AIMD-FC gains one step towards convergence at cycle j ($j > 2$), then it gains a step also at the cycle $j + 1$.

Let

$$\underbrace{\frac{x_1}{2^j} + \frac{x_2}{2^j} + \frac{k_1}{2^{j-1}} + \frac{k_2}{2^{j-2}} + \cdots + k_{j-1}}_{\psi_{j-1}} + 2k_j \geqslant W$$

and

$$\underbrace{\frac{x_1}{2^j} + \frac{x_2}{2^j} + \frac{k_1}{2^{j-1}} + \frac{k_2}{2^{j-2}} + \cdots + 2k_{j-1}'}_{\psi_{j-1}'} + 2k_j' \geqslant W$$

be the $\sum(w+k)$ state of the AIMD and AIMD-FC flows, respectively, prior to the jth decrease. From the induction hypothesis the following relations which represent the sum of the windows after the multiplicative decrease are true:

$$\text{AIMD:} \quad \gamma_j = \frac{\psi_{j-1}}{2} + k_j \geqslant \frac{W}{2} \tag{9}$$

$$\text{AIMD-FC:} \quad \gamma_j < \frac{\psi_{j-1}'}{2} + 2k_j' < W \tag{10}$$

During the increase phase, each flow will increase its resource consumption until k_{j+1} and k_{j+1}', at which point the system feedback is changed to 0. $\sum(w+k)$ becomes

$$\begin{array}{ll} \text{AIMD} & \text{AIMD-FC} \\ \frac{\psi_{j-1}}{2} + k_j + 2k_{j+1} \geqslant W & \frac{\psi_{j-1}'}{2} + 2k_j + 2k_{j+1}' \geqslant W \end{array} \tag{11}$$

Eqs. (9)–(11) imply that $k_{j+1}' < k_{j+1}$. So, AIMD-FC gains a step also during the $j + 1$ cycle.

[11] That is, during the time the second cycle starts and up to the last step of the second cycle.

The system needs $n = 1 + \lg(\max(x_1, x_2))$ cycles to converge to fairness. The minimum gain of AIMD-FC algorithm is at least $n - 1$ steps. $\square$

Since smoothness of AIMD-FC improves by $\frac{1}{6}W$ (i.e., '$\frac{2}{3}W - \frac{1}{2}W$'—see Section 3.3.3) it can be proved further that the responsiveness gain of AIMD-FC during each cycle is $\frac{1}{6}W$ steps.

Combining Theorems 1 and 2 we conclude that, with parameters 1 and 1/2, AIMD-FC has higher responsiveness than AIMD.

3.3.5. *Algorithm limitations*

Here we consider those cases where the AIMD-FC algorithm cannot be applied. For example, the AIMD itself can not be applied when the window is equal to one byte/segment/packet.

Consider a single flow system as that of Fig. 3. The decrease window of this flow is w and assume that prior to congestion $k \times a_{\mathrm{I}}$ resources were allocated in additive increase. Therefore, $w + k \times a_{\mathrm{I}} \geqslant W$. The ensuing phase of multiplicative decrease will produce a reduction of resource utilization at

$$w \leftarrow \frac{w}{2} + k \times a_{\mathrm{I}}. \tag{12}$$

AIMD-FC can only be applied if

$$w \geqslant 2a_{\mathrm{I}}. \tag{13}$$

4. AIMD-FC+ convergence algorithm

4.1. *Convergence lemma*

AIMD-FC increases the bandwidth utilization of AIMD from 3/4 to 5/6. However, the efficiency boundaries of AIMD have not yet been exploited. We are interested in this problem from a perspective where fairness plays a pivotal role.

Placing side by side the multiplicative decrease functions of AIMD and AIMD-FC: [12]

[12] Additive increase functions are the same.

$$\text{AIMD: } w \leftarrow 1/2(w + k) \tag{14}$$

$$\text{AIMD-FC: } w \leftarrow 1/2(w) + k \quad \text{which is equivalent to} \quad 1/2(w + k) + (1/2)k \tag{15}$$

We notice that AIMD-FC augments its window by a well-known factor: $\frac{1}{2}k$. This improves its fairness and efficiency and suggests that augmenting the windows after multiplicative decrease, by a well-known increase factor, leads to enhanced efficiency and fairness. Hence, a natural question of practical importance is how far can we adjust the window upwards. We need to determine the appropriate value, which will not violate the constraints of congestion avoidance nor the conditions of equilibrium.

Strict requirement for this scheme to work is the presence of "common knowledge" for all the current flows; the value k in our case. Practical requirement is to avoid an increase which will cause immediate congestion. Recall that in AIMD-FC the window decrease is given by the formula $w \leftarrow (w/2) + k$ (where k is the additive increase value) and in equilibrium, $w/2 = k$. Hence, a hard boundary for the *extra* value in search is half the maximum *decrease window*: $w/2$. Below we formalize the above intuition and present a new algorithm that improves the efficiency and fairness of AIMD-FC.

Lemma 3 (Convergence lemma). *Let* x_1, x_2 ($x_1, x_2 \in N$; $0 < x_1 < x_2$) *denote the initial states of resources/windows of two flows*, n ($n \in N, n > 0$) *the number of cycles in which AIMD-FC-System* $((x_1, x_2), 2, n)$ *converges to fairness*; *let* n' *be an integer* ($1 \leq n' \leq n$); *let* (ϕ_1, ϕ_2) $\leftarrow$ *AIMD-FC-System*$((x_1, x_2), 2, n')$ *be the output of an AIMD-FC system, and let* q *be integer* ($q < \phi_2/2$).

Then AIMD-FC-System$((\phi_1 + q, \phi_2 + q), 2, n - n')$ *converges.*

Proof. Expressing ϕ_1 and ϕ_2 as two series we have

$$\phi_1 = \frac{x_1}{2^{n'}} + \frac{k_1}{2^{n'-1}} + \frac{k_2}{2^{n'-2}} + \cdots + k_{n'} = \frac{x_1}{2^{n'}} + \kappa,$$

$$\phi_2 = \frac{x_2}{2^m} + \frac{k_1}{2^{n'-1}} + \frac{k_2}{2^{n'-2}} + \cdots + k_{n'} = \frac{x_2}{2^{n'}} + \kappa.$$

Unrolling the windows returned after $n - n'$ cycles by AIMD-FC-System$((\phi_1 + q, \phi_2 + q), 2, n - n')$ we have the series

$$\psi_1 = \frac{x_1}{2^{n'}2^{n-n'}} + \frac{\kappa}{2^{n-n'}} + \frac{q}{2^{n-n'}} + \frac{k'_1}{2^{n-n'-1}} + \cdots + k'_{n-n'}$$

$$= \frac{x_1}{2^n} + \frac{\kappa}{2^{n-n'}} + \frac{q}{2^{n-n'}} + \frac{k'_1}{2^{n-n'-1}} + \cdots + k'_{n-n'},$$

$$\psi_2 = \frac{x_2}{2^{n'}2^{n-n'}} + \frac{\kappa}{2^{n-n'}} + \frac{q}{2^{n-n'}} + \frac{k'_1}{2^{n-n'-1}} + \cdots + k'_{n-n'}$$

$$= \frac{x_2}{2^n} + \frac{\kappa}{2^{n-n'}} + \frac{q}{2^{n-n'}} + \frac{k'_1}{2^{n-n'-1}} + \cdots + k'_{n-n'}.$$

AIMD-FC-System$((\phi_1 + q, \phi_2 + q), 2, n - n')$ converges if $\psi_1 = \psi_2$. This is true only if $x_1/2^n \simeq x_2/2^n \simeq 0$, which is, in turn, true since AIMD-FC-System$((x_1, x_2), 2, n)$ converges. □

Corollary 4. *After every multiplicative decrease phase all the windows of the flows participating in a system can be adjusted upward by q resource units each. If q is less than half the maximum decrease window, the number of cycles required for AIMD-FC to converge will not be affected.*

If we assume that q units can be added to the window of every AIMD-FC flow after each multiplicative decrease, the minimum value of the window will be '$w + q$'. Furthermore, $q > 0$ implies $(w + q) > \frac{2}{3}W$ (recall that $w \geqslant \frac{2}{3}W$ in AIMD-FC). Based on the above corollary we describe an algorithm (AIMD-FC+) that improves the efficiency, smoothness and fairness of AIMD-FC.

4.2. AIMD-FC+

Our problem is to determine the amount of resources that could be added to the congestion windows after multiplicative decrease. According to the observation above, this needs to be less than half the maximum decrease window in the system; otherwise, an equilibrium will not be reached. The challenging part of this problem is hidden behind the distributed nature of our system: the participating flows do not know what the maximum decrease window is. However, they do have additional resources to utilize for this purpose albeit these will not lead the system utilization to the theoretical boundary. One such common and well-known resource is the bandwidth allocated during additive increase.

Let w^j be the decrease window of an AIMD-FC flow at the beginning of cycle j and assume that after k^j steps congestion occurs in the network. In the next cycle $w^{j+1} \leftarrow w^j/2 + k^j$. In equilibrium $k^j = w^j/2$ and k^j is a common knowledge in the system. Adding resources from k^j into w^{j+1} does not preserve the congestion avoidance property of the algorithm. Imagine a scenario where the system is in equilibrium and flows leave the system. The system will be still in equilibrium but $k^j > w^j/2$. If we add resources from k^j to w^{j+1} the sum of the windows in the system might exceed the system threshold (i.e., $\sum(w^{j+1} + k^j) \geqslant X_{\text{goal}}$). Therefore, a good source of common knowledge is w^j. We know that $w^j \leftarrow w^{j-1}/2 + k^{j-1}$ and $k^{j-1} < w^j$; consequently $k^{j-1}/2 < w^j/2$. So $k^{j-1}/2$ satisfies both our requirements for less than half the maximum decrease window *and* for a well-known amount of resources. We incorporate the new functionality in AIMD-FC+ function presented below. The additional notation used in this section is listed below.

w^j	The window at the beginning of cycle j
dw	Variable that records the decrease window of AIMD-FC+
$f(q)$	$f: N \rightarrow N; 0 \leq f(q) \leq q/2$. From the above discussion, $q = k^{j-2}$

```
AIMD-FC+(w, dw, q)
1 $k \leftarrow 0$
2 while (feedback is 1)
3    process( $w + k$ )
4    $k \leftarrow k + a_I$
5 end
6 $dw \leftarrow \frac{1}{2}dw + (w + k - dw)$
7 $w \leftarrow dw + f(q)$
8 return (w, dw, k)
END
```

```
AIMD-FC+System($(x_1, x_2, \dots, x_m), m, n$)
1  local variables: $dw, k$
2  forall $i = 1$ to $m$ do
3      $k \leftarrow 0$
4      $w_i \leftarrow x_i$
5      $dw \leftarrow x_i$
6  end forall
7  for $j = 1$ to $n$ do
8      forall $i = 1$ to $m$
9      $(w_i, dw, k) \leftarrow$ AIMD-FC+$(w_i, dw, k)$
10         end forall
11   end
12   return $(w_1, w_2, \dots, w_m)$
END
```

4.2.1. Correctness

Since $w \leftarrow dw + f(q)$ and q is well known and the same for every flow and $f(q) \ll dw$, it suffices to prove that

If AIMD-FC-System$((x_1, x_2), 2, n)$ *converges to fairness, then, after n cycles of AIMD-FC+System,* $dw_1 = dw_2$.

The dw value of a single flow at cycle j ($j < n$) is

$$dw^j = \frac{dw^{j-1}}{2} + (w^{j-1} + k - dw^{j-1}) = \frac{dw^{j-1}}{2} + \{dw^{j-1} + f(q^{j-2}) + k - dw^{j-1}\} = \frac{dw^{j-1}}{2} + k + f(q^{j-2}).$$

Since k and $f(q^{j-2})$ are fairly allocated and well known to every flow, the correctness of the AIMD-FC+ follows from Corollary 4.

4.2.2. Efficiency of AIMD-FC+

The minimum value of congestion window of AIMD-FC+ is '$dw + f(q)$'. By estimating dw and $f(q)$ first, we compute the average throughput per RTT of AIMD-FC+ (Fig. 4).

Assume a single-flow system: after k additive increase steps the congestion window ($w + k$) will reach the maximum value W. If we assume '$w + k - dw = k + f(q)$' to be the increase of AIMD-FC+ (line 6), then $dw = \frac{2}{3}W$ (in Section 3.3.2 we have shown that an algorithm that decreases according to the formula $\frac{1}{2}w + k$ has decrease window $\frac{2}{3}W$ and increase space $\frac{1}{3}W$). The decrease window has the same value as in AIMD-FC.

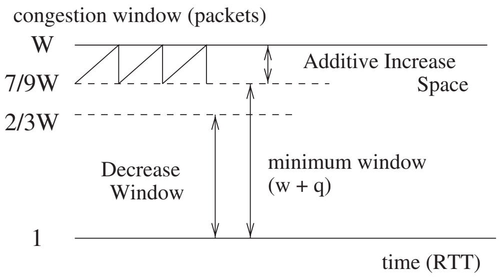


Fig. 4. AIMD-FC+ window evolution under periodic loss. Each cycle delivers $(\frac{7}{9}B)(\frac{2}{9}B) + \frac{1}{2}(\frac{2}{9}B)^2 = 1/p$ packets and takes $\frac{2}{9}W$ round trip times.

Under the same conditions, let q^{j-1} be the number of additive increase steps of the last cycle and assume '$\phi = W - dw = \frac{1}{3}W$' to be the increase space above dw value of AIMD-FC+ (according to the above assumption). The (real) additive increase during the cycle $j + 1$ would be $q^{j+1} = \phi - f(q^{j-1})$ (this is from the definition of the algorithm). If $f(q) = \frac{1}{2}q$ (see Section 4.2) it can be shown that $f(q) = \frac{1}{3}\phi = \frac{1}{9}W$; the proof follows with similar arguments as the proof in Section 3.3.2. This means that '$dw + f(q)$' value (or $w + q$ in Fig. 4) is equal to $\frac{7}{9}W$.

From the window evolution of AIMD-FC+ (Fig. 4), the average throughput (per RTT) of the flow is

$$\frac{\text{MSS}}{\text{RTT}}\frac{8}{9}W \tag{16}$$

which implies 88.9% efficiency.

4.2.3. Responsiveness and smoothness

Theorem 2 in Section 3.3.4 shows that an inherent property of adding a fixed value to all the flows' windows in the system is that both responsiveness and smoothness improve. In essence, any non-zero value of $f(q)$ reduces the number of steps per cycle. In a similar manner that we showed AIMD-FC smoothness and responsiveness we can show that for every value $f(q)$ that we add, smoothness is improved by $f(q)/W$ and responsiveness increases by at least $n - 1$ steps (n is the number of cycles that the protocol converges).

5. Experiments with TCP

We have incorporated the AIMD-FC and AIMD-FC+ algorithms [13] into TCP [3] and have validated its performance on NS-2 [13]. TCP controls the sending rate by a parameter called *congestion window* [2]. When resources are available TCP increases the congestion window by *one* MSS; upon congestion and in the presence of three duplicate acknowledgments, TCP multiplies the congestion window by a factor of 1/2 (this TCP is also known as $\mathrm{TCP}(1, 1/2)$). Recall that in the absence of errors the average long term efficiency of the AIMD mechanism of TCP is 75% [10].

The TCP version of choice in our experiments was TCP-SACK [14]. Due to its Fast Recovery and its capability for multiple retransmissions within one RTT, this version matches better the assumptions of our theoretical work. However, there is an additional component in TCPs congestion control, namely the timeout mechanism; and there is an additional component in the initial window expansion phase, namely, the Slow Start mechanism.

In the experiments we consider a simple network topology (see Fig. 5) with homogeneous flows where all links have the same bandwidth and routers use Droptail and RED [15] queue management (configured both on the bottleneck link and in the access links), devoid of pricing models, fair-queuing disciplines or quality of service mechanisms. Simply we consider an Internet infrastructure based on the end-to-end design argument [16]. In this work we do neither consider the co-existence of AIMD flows with AIMD-FC flows. Stability of the network and fairness issues that come up when protocols have diverse or even greedy (but responsible) congestion avoidance mechanisms is discussed in [17].

In this topology, multiple flows share a high-bandwidth bottleneck link; the fair-share (the Delay $\times$ Bandwidth share per flow) was set relatively high in order to provide the environment for the algorithms to exploit their potential. For example, AIMD is not activated when the fair share is only one packet, or otherwise when contention is too high and bandwidth is limited, efficiency is not really an issue.

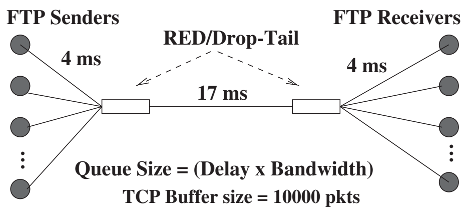

FTP Senders
FTP Receivers
RED/Drop-Tail
4 ms
4 ms
17 ms
Queue Size = (Delay x Bandwidth)
TCP Buffer size = 10000 pkts

Fig. 5. Multiple flows experimental set-up for AIMD evaluation.

We evaluate three distinct scenarios: Our first scenario is characterized by stationarity in terms of the number of participating flows. The scenario matches well the theoretical assumptions. We study comparatively the behavior of the algorithms and we present experiments with both default and RED gateways. Our second scenario involves progressive contention due to periodic increase of the number of flows. The subject matter we investigate with this experiment is the mechanism's potential for efficient congestion avoidance and control, i.e., not only its convergence behavior. In our third experiment we evaluate the system's responsiveness: bandwidth becomes available and protocols ought to demonstrate capabilities to consume the available resource fast. Both our second and third experiments aim at alleviating reasonable concerns regarding the algorithm's behavior in dynamic (and hence more realistic) environments. We note that our experiments do not cover the whole spectrum of experimental evaluation, which can be a subject of study in its own right; we provide here substantial evidence on the algorithm's practical impact, along with our theoretical perspective. Further experimental studies may be driven by specific network and flow characteristics, or protocol, application and device properties.

A TCP flow runs at each end node and an FTP application generates the traffic for each source. The TCP buffer size was set large enough so that it can exceed the delay bandwidth product of the network. The rest of TCP parameters are the default ns2 [13] parameters. The task of the application is to send data for 60 s. For both

[13] A detailed technical report can be found in [12].

RED and Droptail policies the queue buffers were set equal to the Delay × Bandwidth product. The RED minimum drop threshold was set to (1/3) and the maximum drop threshold was set to (2/3) of the RED buffer size. The rest of the RED parameters are the default ns2 [13] parameters.

We measured the number of packets that arrive at the receivers; since the time of the experiments is fixed we report this number as *Goodput* in the figures (average of 30 experiments with minimal statistical deviation). Goodput is a metric for the system efficiency. In line with our theoretical findings and in order to measure the convergence behavior of the participating flows, we use the Fairness Index used in [18]

$$F(g) = \frac{(\sum g_i)^2}{n(\sum g_i^2)} \tag{17}$$

where g_i is the goodput achieved by each flow.

5.1. Stationary environment

The experiments with NS-2 simulator countersign the potential of AIMD-FC/FC+ in TCP. Fig. 6 shows that Goodput is improved up to 5% for a small number of flows (high fair-share; e.g. 2, 4, 8 flows) and Fig. 7 shows the same improvement when the fair-share is high. When the number of flows increases, there is still improvement but there is also a tendency to approach the goodput of AIMD-based TCP. When the fair-share is getting

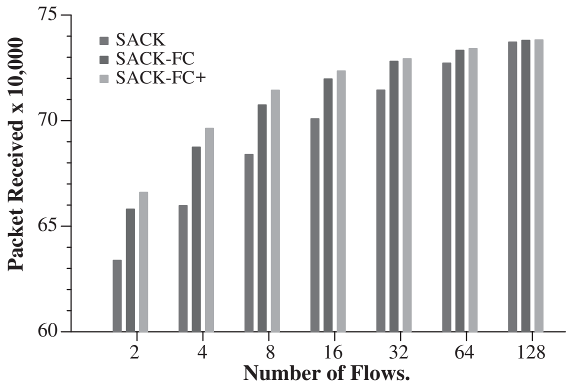


Fig. 6. Goodput performance of TCP-SACK with AIMD, AIMD-FC and AIMD-FC+ on a 100 Mbps link and Droptail gateway.

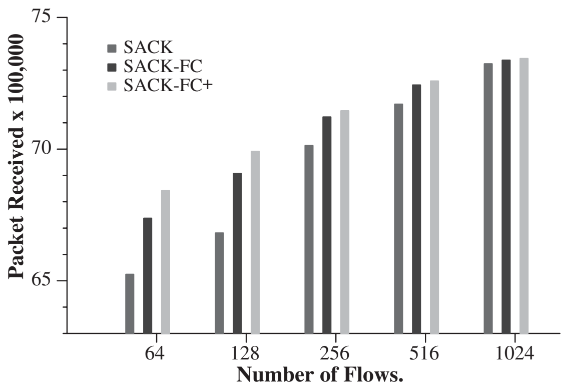


Fig. 7. Goodput performance of TCP-SACK with AIMD, AIMD-FC and AIMD-FC+ on a 1 Gbps link and Droptail gateway.

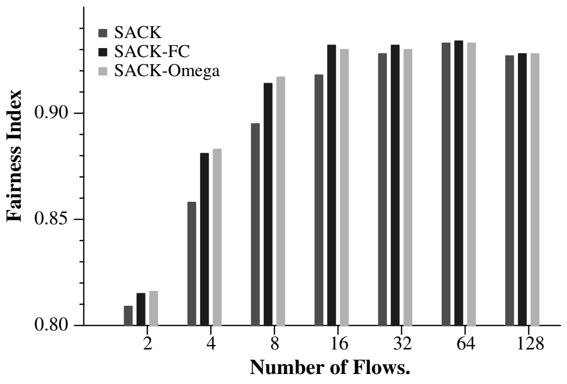


Fig. 8. Fairness with the 100 Mbps link and Droptail gateway.

rather small, the goodput performance of the two algorithms is balanced. Note that this is an expected result. When the number of competing flows increases, efficiency is not really an issue.

Fig. 8 shows the fairness of the protocols. In accordance with our expectations that rise from the second theorem, the fast convergence algorithms achieve better results on fairness.

Figs. 9 and 10 show the results from experiments over a 100 Mbps link where each node is connected to the back-bone through a RED gateway [15]. The Goodput of the system is shown in Fig. 9. The experimental results match the theoretical performance albeit the conditions of the experiment do not exactly match the assumptions of the control system in Section 2. Although,

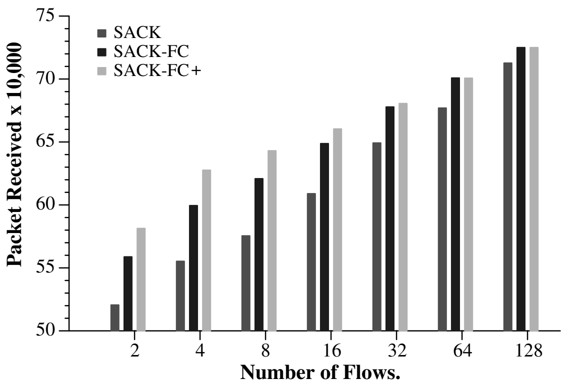


Fig. 9. Goodput performance of TCP-SACK with AIMD, AIMD-FC and AIMD-FC+ on a 100 Mbps link and RED gateway.

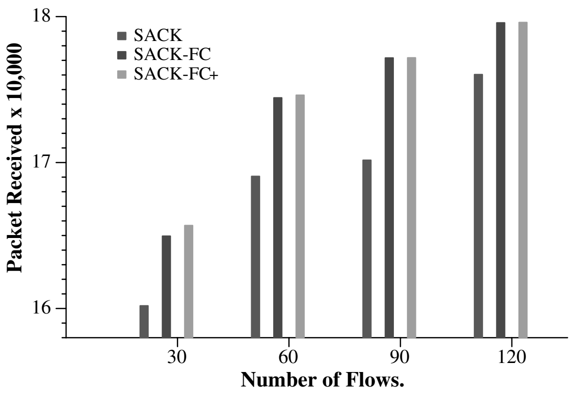


Fig. 11. Goodput performance of TCP-SACK with AIMD, AIMD-FC and AIMD-FC+ on a 100 Mbps link and a RED gateway. 30 flows join the system every 15 s.

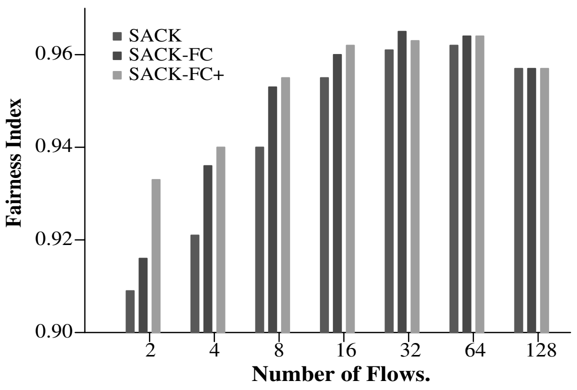


Fig. 10. Fairness with the 100 Mbps link and RED gateway.

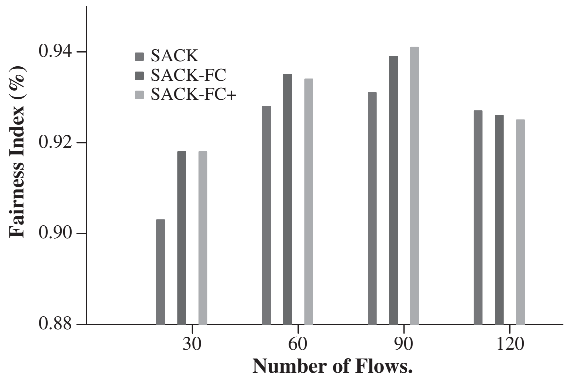


Fig. 12. Fairness with a 100 Mbps link and a RED gateway. 30 flows join the system every 15 s.

comparatively, RED does not outperform Drop-tail in Goodput, the relative performance gains of AIMD-FC and AIMD-FC+ are apparent. Fairness of the system (Fig. 10) appears to be improved with RED gateways. The relative fairness performance of the three algorithms with TCP gains further significance since it confirms practically our assertion that the improvement on efficiency does not come at the cost of fairness.

5.2. Graduated contention increase

In addition to the above simulations where the number of flows is constant during the experiment, we evaluate the performance of the algorithms with graduated contention increase; there, new flows enter the system periodically. Figs. 11 and 12 outline the performance of the protocols under this scenario. At time 0.0, 30 nodes start to send data to their peers; 30 more nodes join the network every 15 s (up to a total of 120 nodes). The goodput of the system (total goodput of the flows) over a 15 s time interval is presented in Fig. 11. Fig. 12 presents the corresponding results of fairness.

The major contribution of this experiment is not to highlight the gain of AIMD-FC but rather to confirm experimentally that the new approach does not exhibit any conflicting behavior with the congestion control mechanisms of TCP. The combined results of efficiency and fairness suggest that it's potential for congestion avoidance remains high.

5.3. Graduated bandwidth increase

We observe the system and flow behavior under network conditions of graduated availability of bandwidth (Figs. 13 and 14). We note that increase is not a consequence of bandwidth provisioning but instead, of flow duration. That is, some flows complete their task and, periodically, leave the system, making available space to existing flows to consume more bandwidth. Clearly, the dominant factor for that target is the number of steps required for the flows to reach their fairshare. In other terms, protocol responsiveness to conditions of rapid bandwidth availability is the real issue with this experiment.

Both charts below reveal a clear-cut advantage for AIMD-FC and AIMD-FC+. What is not exactly clear from the experiments is the specific conditions that favor AIMD-FC over AIMD-FC+. The Goodput performance with minimal contention (i.e., small number of competing flows) indicate the extent at which *extra* amount of resources may be exploited with AIMD schemes; when the number of participating flows increases, the available extra resource space per flow does not translate into extra packets (MSS) and hence, the difference is not reflected on protocol Goodput. Driven by the same argument, an interesting conclusion can be drawn from the results of fairness. Occasionally, as in the case of 60 flows in Fig. 14, some flows may be able to exploit that extra amount of available resource due to a timeout of some TCP senders. Although that behavior is not captured by the protocol Goodput, it is captured indeed by the Protocol Fairness: AIMD-FC performs better than AIMD-FC+ in this occasion.

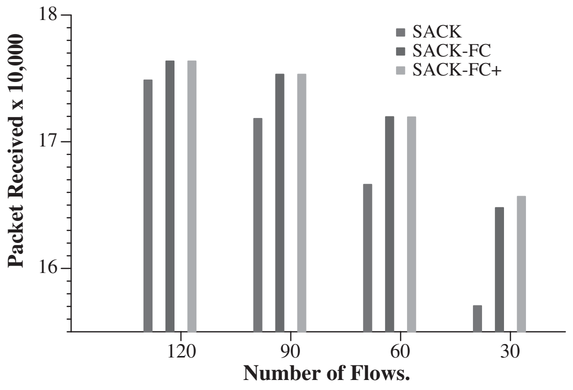


Fig. 13. Goodput performance of TCP-SACK with AIMD, AIMD-FC and AIMD-FC+ on a 100 Mbps link and a RED gateway. 30 flows leave the system every 15 s.

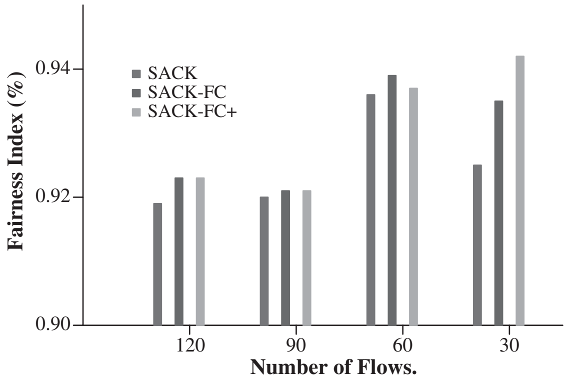


Fig. 14. Fairness with the 100 Mbps link and a RED gateway. 30 flows leave the system every 15 s.

6. Discussion

We have emphasized our packet-network perspective of AIMD. That is, practically, our potential solutions need to be applicable in this context. It is notable, however, that, from the algorithmic perspective, further changes can be made to approach more effectively the target of efficiency and fairness. For example, at the second cycle towards convergence our system has m flows with a common property: they have all increased their windows by k_1 packets. It is hence not unreasonable to expect that all sources could adjust their windows to k_1, reaching fairness in a single cycle. Furthermore, the sources could maintain their additive behavior until the next cycle. Again, all sources would have been growing similarly, at a level k_2, where they reach their maximum efficiency (i.e. all flows have windows $w = k_1 + k_2$). That is, our scenario involves a one-cycle convergence to fairness and a two-cycle convergence to efficiency with a success rate 100% and without a need for continuous adjustments. However, the assumptions made here do not hold in packet networks, or more precisely, further modifications are needed in order for them to form a functional system. A first concern is the system's inability to accept new flows when the system reaches a equilibrium. Of course, an algorithm can be made to allow for small oscillations so that new flows will find some space to grow. Then, upon a system's feedback, all flows will be needed to re-initiate the convergence procedure. A second concern is that a system in equilibrium will not be able to exploit the extra bandwidth that may become available when some flows complete their task (i.e. leave the system). Finally, when new flows enter the system at a point where other flows have converged to fairness [14] but have not yet reached efficiency they will not share common information. That is, while the old group of flows attempts to converge to efficiency, the new group will attempt to converge to fairness. However, the system's capacity will be exhausted; at their second cycle, the new flows will not have much opportunity to exploit new bandwidth, nor will the old flows release some of their resources. It is therefore necessary the participating flows to release resources uniformly *every* time there is a drop; a two-cycle procedure that is associated with two distinct *tactics* (i.e. adjust to k_1 or adjust to $k_1 + k_2$) cannot work. Although this discussion requires further analysis, and an immediate solution is not apparent, we note that we do not consider this direction of research unreasonable.

We also note that in the context of AIMD-FC+ a new type of tradeoff is developed: the more we approach the bandwidth line, the less space we leave for new incoming flows. This tradeoff is not associated with the dynamics of the algorithm but rather with the dynamics of the network. The impact is expected to be rather small since a packet drop due to increasing contention will signal the end of the equilibrium and re-initiate the convergence procedure.

By the same token, AIMD-FC properties apply not only for better utilization. For example, the multiplicative decrease factor can be selected so that it will approximate the bandwidth of standard TCP; obviously, this will lead to a TCP protocol with enhanced capability to reach an equilibrium faster, i.e. a responsive TCP. Similarly, TCP-friendly protocols can be made to reach equilibrium faster. Note that TCP-friendly protocols are designed to share bandwidth fairly with standard TCP; however, they degrade a system's ability to reach an equilibrium fast [19]. In addition, TCP-friendly protocols grab more bandwidth when bandwidth is in demand (i.e. during congestion—see also [20]) unlike TCP with AIMD-FC which adjusts backwards more rapidly during congestion. Indeed, the properties of AIMD-FC can be used to indicate the latitudes within which the tradeoff of responsiveness and smoothness is amenable to further refinement and deployment. Beyond that, TCP with AIMD-FC can be deployed in the exact form that is presented here. Recently, several versions of TCP co-exist in the Internet since they are integrated in different operating systems. It is worthnoting the example of TCP-SACK which co-exists with TCP-Tahoe; although SACK is far more aggressive than Tahoe, it has seen a rapid deployment due to its enhanced sophistication and the low-cost of deployment (e.g. no router modification is needed).

Finally, a requirement for our system is to provide feedback upon congestion. Modern networks have implemented more sophisticated mechanisms to provide feedback to the transport layer. Such examples are RED gateways [15] and ECN-capable routers [21]. Although a Droptail Router seems to satisfy better the system assumption of synchronous feedback, our experimental results with RED were encouraging. The collaborative potential of our algorithm with the functionality of devices such as RED and its variants can be further investigated.

7. Conclusion

We have shown that the potential of AIMD has not been fully exploited. We presented a modification and demonstrated the corresponding performance improvement, in the context of packet networks. In comparison with recent proposals,

[14] Notably, this pattern of arrivals can happen with the original algorithm and the algorithms presented above. However, in that case, the multiplicative decrease plays also the role of a corrective procedure. The number of steps that has been wrongly "recorded" as a common fair-share, is being refreshed at every cycle.

AIMD-FC has two distinctive properties (i) it does not favor one performance characteristic at the expense of another, and (ii) it does not damage the capability of the algorithm to deal with congestion avoidance.

We have tested the performance of AIMD-FC/FC+ with TCP. The results reveal a great potential of AIMD-FC/FC+ for packet networks. Further analysis will be done for more complex systems, more metrics may be defined, and more experimental results will follow the present work. We note that AIMD-FC/FC+ is compatible, in principle, with the original AIMD algorithm. The required modifications are minor, yet the service improvement for applications that use TCP could be significant. However, since AIMD is not restricted to any specific protocol, the modifications proposed here can be practically useful to other protocols that apply congestion avoidance and control but require increased smoothness and responsiveness.

References

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

[2] V. Jacobson, Congestion avoidance and control, in: Proceedings of ACM SIGCOMM'88, 1988, pp. 314–329.

[3] J. Postel, Transmission Control Protocol, RFC 793.

[4] J. Halpern, Y. Moses, Knowledge and common knowledge in a distributed environment, Journal of the ACM 37 (3) (1990) 549–587.

[5] Y. Yang, S. Lam, General AIMD congestion control, in: Proceedings of the IEEE International Conference on Network Protocols, 2000.

[6] D. Bansal, H. Balakrishnan, Binomial congestion control algorithms, in: Proceedings of IEEE INFOCOM'01, 2001, pp. 631–640.

[7] S. Jin, L. Guo, I. Matta, A. Bestavros, TCP-friendly SIMD congestion control and its convergence behavior, in: Proceedings of ICNP'2001, 2001.

[8] J. Padhye, V. Firoiu, D. Towsley, J. Kurose, Modeling TCP throughput: A simple model and its empirical validation, in: Proceedings of ACM SIGCOMM, 1998.

[9] S. Floyd, K. Fall, Promoting the use of end-to-end congestion control in the internet, IEEE/ACM Transactions on Networking 7 (4) (1999) 458–472.

[10] M. Mathis, J. Semke, J. Mahdavi, T. Ott, The macroscopic behavior of the TCP congestion avoidance algorithm, ACM Computer Communication Review 27 (1997) 20–26.

[11] R. Karp, E. Koutsoupias, C. Papadimitriou, S. Shenker, Optimization problems in congestion control, in: IEEE Symposium on Foundations of Computer Science, 2000, pp. 66–74.

[12] A. Lahanas, V. Tsaoussidis, Exploiting the efficiency and fairness potential of AIMD-based congestion avoidance and control, Tech. Rep. NU-CCS-02-04, Web Page: http://www.ccs.neu.edu/~ladrian, April 2002.

[13] The Network Simulator—NS-2, Tech. rep., Web Page: http://www.isi.edu/nsnam/ns/. Version 2.1b7a, October 2000.

[14] M. Mathis, J. Mahdavi, S. Floyd, A. Romanow, TCP Selective Acknowledgment Options, RFC 2018.

[15] S. Floyd, V. Jacobson, Random early detection gateways for congestion avoidance, IEEE/ACM Transactions on Networking 1 (4) (1993) 397–413.

[16] J.H. Saltzer, D. Reed, D. Clark, End-to-end arguments in system design, ACM Transactions on Computer Systems 2 (4) (1984) 277–288.

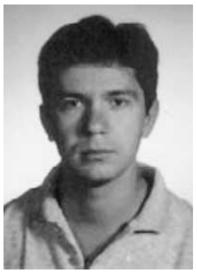
[17] A. Akella, S. Seshan, R. Karp, S. Shenker, C. Papadimitriou, Selfish behavior and stability of the Internet: a game-theoretic analysis of TCP, in: ACM SIGCOMM, 2002.

[18] R. Jain, D.M. Chiu, H. Hawe, A quantitative measure of fairness and discrimination for resource allocation in shared systems, Tech. Rep. DEC-TR-301, Digital Equipment Corporation, 1984.

[19] C. Zhang, V. Tsaoussidis, The interrelation of TCP responsiveness and smoothness in heterogeneous networks, in: Proceedings of the 7th IEEE Symposium on Computers and Communications, ISCC, 2002.

[20] M. Vojnovic, J. Boudec, On the long-run behavior of equation-based rate control, in: ACM SIGCOMM, 2002.

[21] K. Ramakrishnan, S. Floyd, A Proposal to add Explicit Congestion Notification (ECN) to IP, RFC 2481.



Adrian Lahanas received his B.Sc. degree in Computer science from University of Cyprus (1993–1997). Currently he is Ph.D. student at Northeastern University, and is expected to graduate in June 2003. His interests are transport protocols over wired/wireless networks.

Vassilis Tsaoussidis specializes in Network Protocols, QoS and Mobile Computing. He is currently with the Engineering Department of Democritos University, Greece. Vassilis was a faculty member of the Computer Science Department of SUNY Stony Brook (1998–2000) and Northeastern University (2000–2003). He is an editor for Wiley's journal of Wireless Communication and Mobile Computing, and a committee member of INFOCOM, ISCC, ICCCN, GlobeCom

etc. He has chaired the conference on Internet Computing 2002 and the Workshop on Wired/Wireless Internet Communications; he has was also the guest editor of three journal special issues on Internetworking protocols, performance evaluation and wireless computing. Vassilis has published over 50 papers and supervised 2 doctoral theses and several master projects.