

Stability and Fairness of Rate Estimation-Based AIAD Congestion Control in TCP

Kai Xu and Nirwan Ansari, *Senior Member, IEEE*

Abstract—We analyze two achievable rate estimators that use different timestamps of consecutive packets. We examine the effect of the choice of rate estimators on the stability and fairness of a class of TCP protocols that use this estimated rate to implement the additive-increase/adaptive-decrease (AIAD) congestion control. Simulation results confirm our analysis that rate estimation based on the inter-arrival times of the ACK packets is not properly bounded and would cause instability of the AIAD algorithm and unfairness among competing TCP flows, particularly in networks with small or moderate buffer space. Whereas, the rate estimation based on the inter-arrival times of the data packet at the receiver maintains its accuracy even when the reverse path is congested, and enables the AIAD algorithm to maintain the stability and fairness as the number of competing flows increases. Our analysis also suggests a straightforward enhancement to TCP Westwood that would improve its stability and fairness. The enhanced algorithm can be easily implemented without any modifications to the TCP receiver-side code by enabling the TCP timestamps option.

Index Terms—AIAD, rate estimation, TCP timestamps.

I. INTRODUCTION

UNLIKE the additive-increase/multiplicative-decrease (AIMD) congestion control algorithm [1] in traditional TCP that reduces the sending rate multiplicatively upon packet loss due to congestion, TCP Westwood [2] adjusts the sending rate adaptively based on the sender's estimations of the achievable throughput or rate. We call this class of TCP schemes the rate estimation-based additive-increase/adaptive-decrease (AIAD) congestion control, or simply AIAD throughout this paper. In the rate estimation-based AIAD, the sender estimates the achievable throughput, or rate, by continuously examining the timestamps and patterns of the returning acknowledgment packets (ACKs). When congestion is detected via packet losses, the sender adjusts the transmission rate according to the estimated achievable rate. The evolution of TCP's congestion window, $w(t)$, of such AIAD can be expressed as follows upon each returned ACK or the detection of loss:

$$w(t+1) = \begin{cases} w(t) + \frac{1}{w(t)}, & \text{no congestion} \\ r(t)d, & \text{congestion loss} \end{cases}, \quad (1)$$

where $r(t)$ is the TCP's estimation of the achievable rate, i.e., the sustainable rate without causing congestion, and d is the round-trip propagation delay. It is clear from (1) that in the absence of congestion, the sender's congestion window

Manuscript received in August 23, 2004. The associate editor coordinating the review of this paper and approving it for publication was Dr. N. Nikolauou. This work has been supported in part by the New Jersey Commission on Science and Technology via NJWINS.

The authors are with the Advanced Networking Laboratory, Department of Electrical and Computer Engineering, NJIT, Newark, NJ.

Digital Object Identifier 10.1109/LCOMM.2005.04027.

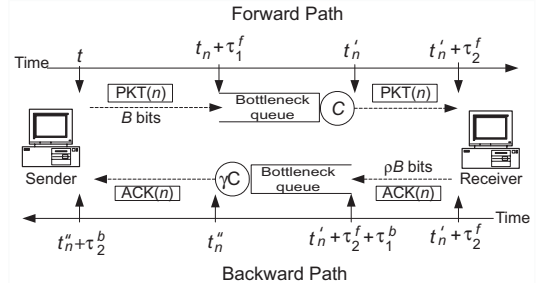


Fig. 1. The passage of a data packet and the corresponding ACK packet, and the definitions of timestamps.

increases additively [1]. On the other hand, if packet loss due to congestion is detected, the sender sets the congestion window according to the estimated achievable rate as $r(t)d$, i.e., the adaptive decrease. It is obvious that over estimation of $r(t)$ above the bottleneck link capacity would have an adverse effect to the congestion condition. In Section II, we analyze two rate estimators that use different timestamps. In Section III, we establish a system model of the rate estimation-based AIAD algorithm and study its conditions of stability when different rate estimators are used. Section IV presents the simulation results that confirm our analysis, and Section V concludes the paper.

II. RATE ESTIMATION

Consider a simple network connected via bi-directional links, where there is only one bottleneck link, on which there are N TCP flows, indexed by $i = 1, 2, \dots, N$, in the same direction contending for its capacity. The bottleneck link is characterized by its forward capacity C , backward capacity γC , forward queue occupancy $q_f(t)$ as seen by the data packet arriving at the queue, and backward queue occupancy $q_b(t)$ as seen by the arriving ACK at the backward bottleneck queue. Packets are indexed by $n = 1, 2, \dots$. A reception of a packet at the destination causes an *immediate* transmission of a corresponding ACK.

As illustrated in Fig. 1, associated with packet n are three timestamps, t_n , t_n' , and t_n'' , representing the time it is sent from the source, the time it departs from the *forward* bottleneck queue, and the time its corresponding ACK departs from the *backward* bottleneck queue, respectively, where τ_1^f , τ_2^f , and τ_1^b , τ_2^b are the propagation delays before and after the bottleneck queues in forward and backward paths, respectively. We assume all data packets have the same size of B bits, and the ACK packets are of size ρB bits, where $\rho < 1$.

Consider two consecutive packets, packet $n-1$ and packet n ; for notational convenience, we define the following inter-packet time intervals $\Delta t_n = t_n - t_{n-1}$, $\Delta t_n' = t_n' - t_{n-1}'$, and

$\Delta t_n'' = t_n'' - t_{n-1}''$. At each instance of either a transmission of a data packet or a reception of a data or an ACK packet, there are three instantaneous rate samples, measurable either at the source or at the destination, that indicate the achievable throughput or rate of the flow. They are,

Sending Rate: $x_{i,n} = B/\Delta t_n''$ is an estimate sample, measurable at the source, based on the inter-transmission time between two consecutive data packets sent by the source;

Receiving Rate: $\varphi_{i,n} = B/\Delta t_n'$ is an estimate sample, measurable at the source by examining the TCP timestamps option field [3] in the ACK packets. It is based on the inter-arrival time of the two consecutive data packets arrived at the destination;

ACK Rate: $\kappa_{i,n} = B/\Delta t_n''$ is an estimate sample based on the inter-arrival time of the returning ACKs measured at the source. It indicates the amount of *data* acknowledged during the interval.

We use $x_i(t)$, $\varphi_i(t)$, and $\kappa_i(t)$ to denote the underlying rate processes which could be constructed by low-pass filtering of the corresponding samples (see [2]). The queuing processes at the forward and backward bottleneck links are $\dot{q}_f(t) = y_f(t) - C$, and $\dot{q}_b(t) = y_b(t) - \gamma C$, respectively, where $y_f(t) = \sum_{i=1}^N x_i(t)$ is the aggregated intensity of the data flows, and $y_b(t) = \rho \sum_{i=1}^N \varphi_i(t)$ is the aggregated intensity of the ACK flows, since the receiver *immediately* acknowledges the data packet upon reception.

Define $\Delta q_f(n) = q_f(t_n + \tau_1^f) - q_f(t_{n-1} + \tau_1^f)$ as the difference of the queue lengths seen by the two consecutive packets when they arrive at the bottleneck. We have $t_n' = (t_n + \tau_1^f) + (q_f(t_n + \tau_1^f) + B)/C$, from which, by approximation $\Delta q_f(n) \approx \dot{q}_f(t) \Delta t_n$, it is easy to derive that

$$\varphi_i(t) = C x_i(t) / y_f(t). \quad (2)$$

Taking into the account that the reception of a B -bit data packet at the destination causes an immediate transmission of a ρB -bit ACK packet, which in turn acknowledges the delivery of a B -bit data packet, we can derive by symmetry that, on the backward path,

$$\kappa_i(t) = \gamma C \varphi_i(t) / y_b(t). \quad (3)$$

In summary, the rate estimator, $\varphi_i(t)$, which is based on the packet receiving timestamps, is always upper bounded by the capacity of the *forward* bottleneck link of the end-to-end path; whereas an estimate based on the inter-arrival time of ACK packets, the $\kappa_i(t)$ estimator, is limited above by the *backward* bottleneck link's capacity multiplied by the ratio of the ACK packet size and the data packet size. $\kappa_i(t)$ could take a value much larger than the capacity of the *forward* bottleneck link, i.e., an over estimation of the bottleneck link capacity.

III. STABILITY OF RATE ESTIMATION-BASED AIAD

In this section, we model the TCP dynamics, in particular, the evolution of the TCP source's sending rate, according to the AIAD algorithm (1). We discuss the stability and fairness issues of TCP flows when different rate estimators are used in (1), i.e., $r(t) = \kappa(t)$, vs. $r(t) = \varphi(t)$. In the network model described in the previous section, for flow i , define τ_i as the

round-trip-time (RTT), d_i as the round-trip propagation delay ($d_i = \eta_i \tau_i$, $0 < \eta_i \leq 1$), and $x_i(t) \approx w_i(t) / \tau_i$.

Let $p(t) = p(y_f(t))$ be the probability that a packet is lost due to congestion; it is a non-negative, non-decreasing function of the aggregated flow rate on the bottleneck link. Following the modeling method outlined in [4], the AIAD algorithm (1) can be translated into the following system of nonlinear delay differential equations

$$\begin{aligned} \dot{x}_i(t) &= 1/\tau_i^2 - [x_i(t) - \eta_i r_i(t)] x_i(t - \tau_i) p(y_f(t - \tau_i)), \\ \dot{q}_f(t) &= y_f(t) - C. \end{aligned} \quad (4)$$

In what follows, we study the choice of $r_i(t)$ and its effect on the stability of system (4). For clarity, subscripts of the flow index i are dropped.

Let $(x_0, q_f, 0)$ be the equilibrium of (4); it can be derived that $x_0 = \varphi_0 = C/N$, $\kappa_0 = \gamma C / \rho N$, $p_0 \doteq p(t)|_{x_0}$, and $p_0' \doteq \frac{\partial p}{\partial x}|_{x_0}$. Linearizing (4) around the equilibrium yields a linear system with time delay in the form of

$$\dot{\delta x}(t) = K_1 \delta x(t) + K_2 \delta x(t - \tau), \quad (5)$$

where K_1 and K_2 are to be determined by the choice of $r(t)$. Taking the Laplace transform of (5), followed by a substitution of $\lambda \doteq s\tau$, we have

$$\tau K_1 e^\lambda + \tau K_2 - \lambda e^\lambda = 0 \quad (6)$$

as the characteristic equation of (5). For convenience, we restate the following lemma by Johari and Tan (see [5], Lemma 2)

Lemma 1: All the roots of $be^\lambda + c - \lambda e^\lambda = 0$, where b and c are real, have negative real parts if and only if: (i) $b < 1$, (ii) $b < -c$, and (iii) $-c < \sqrt{a_1^2 + b^2}$, where a_1 is the root of $a = b \tan a$ such that $0 < a < \pi$. If $b = 0$, we take $a_1 = \pi/2$.

Case 1: The rate estimation in the AIAD algorithm adopts the receiving rate, i.e., $r_i(t) = \varphi_i(t)$. We call this algorithm AIAD $_\varphi$. This can be implemented using the TCP's timestamps option defined in RFC 1323 [3], by which the receiver stamps the packet arrival time into the header of the corresponding ACK packet. The source, by extracting the timestamps from the ACK packets, can compute $\varphi_{i,n}$ and construct $\varphi_i(t)$ by some low-pass filters as those proposed in [2]. In this case, we have $K_1 = (\frac{N-1}{N^2} \eta - \frac{1}{N}) C p_0$ and $K_2 = \frac{\eta-1}{N} C (p_0 + \frac{C}{N} p_0')$. It is easy to verify that (6) satisfies Conditions (i) and (ii) of Lemma 1, regardless of the delay. It can also be shown that as long as the round-trip queuing delay $d_q = \tau - d$ satisfies $d_q < \frac{\pi N}{2C(p_0 + \frac{C}{N} p_0')}$, due to Condition (iii), the characteristic equation of (5) has all its roots with negative real parts, and hence, AIAD $_\varphi$ is locally stable.

Case 2: The AIAD algorithm adopts the ACK rate as its rate estimation, i.e., $r_i(t) = \kappa_i(t)$, referred to as algorithm AIAD $_\kappa$. An example implementation of this algorithm is TCP Westwood [2], where the source examines the inter-arrival time of the ACK packets to compute the rate estimate samples $\kappa_{i,n}$, which are then fed to a low-pass filter to construct $\kappa_i(t)$. In this case, we have $K_1 = (\frac{N-1}{N^2} \alpha \eta - \frac{1}{N}) C p_0$, $K_2 = \frac{\alpha \eta - 1}{N} C (p_0 + \frac{C}{N} p_0')$, where $\alpha = \gamma / \rho$. Subjecting (6) to the stability conditions of Lemma 1, we can show that

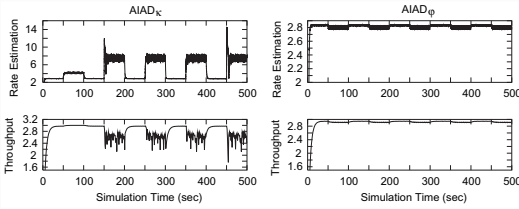


Fig. 2. Rate estimation and throughput (Mb/s).

Condition (ii) requires

$$\alpha\eta \doteq \frac{\gamma d}{\rho(d + d_q)} < \frac{2N + Cp'_0/p_0}{2N + Cp'_0/p_0 - 1}, \quad (7)$$

and Condition (iii) requires $d_q < \frac{\pi N}{2C(p_0 + \frac{C}{N}p'_0)} + (\alpha - 1)d$. Since γ , ρ , and d are fixed by the network configurations and the implementation of the TCP protocol, (7) implies that for system (5) with AIAD $_{\kappa}$ to remain stable as the number of flows increases, the network has to increase its buffer size, or, in other words, with AIAD $_{\kappa}$, fixed buffer size would lead to instability as the number of flows grows.

IV. SIMULATION RESULTS

We use the packet-level network simulator, ns-2, to verify our analysis presented in the previous sections. We also discuss the impact on fairness among competing TCP flows when different rate estimators are used in the AIAD congestion control. We use TCP Westwood, that adopts ACK rate, i.e., $r_i(t) = \kappa_i(t)$, for its rate estimator, as the representative of the AIAD $_{\kappa}$ algorithm. To facilitate our study, we have also modified several lines of the Westwood code so that it uses the receiving rate, i.e., $r_i(t) = \varphi_i(t)$, as the rate estimator, and named our modified version as the AIAD $_{\varphi}$ algorithm.

Our simulations are conducted on a simple network consisting of one symmetric bottleneck link with capacity of 3 Mb/s and buffer size of 20 packets in both forward and backward directions. A varying number of TCP flows traverse from the sources to the destinations in the forward direction via their respective 100 Mb/s access links to the bottleneck. To simulate the congestion at the backward bottleneck link, an ON/OFF constant-bit-rate (CBR) flow is devised to run in the backward direction. All data packets are 1000 bytes in size, and since ACK packets are 40 bytes in size, our settings yield $\gamma = 1$ and $\rho = 0.04$. All links adopt the FIFO service discipline. All simulations are run for 500 seconds, during which, starting from time 50s, the 2 Mb/s CBR flow is set to be ON and OFF for 50 seconds alternately.

In the first simulation, there is only one TCP flow in the forward direction. We run the simulation for AIAD $_{\kappa}$ and AIAD $_{\varphi}$ separately and plot the TCP's throughput as well as its rate estimation in the left and right side of Fig. 2, respectively. The results agree with our analysis in Section II that rate estimator $\kappa_i(t)$ exhibits significant over estimation during the periods when the backward bottleneck is congested by the CBR flow, thus leading to fluctuations of the throughput. This is because the over estimation of the achievable rate causes the sender to overflow the bottleneck buffer, and consequently, induces more packet losses.

On the other hand, algorithm AIAD $_{\varphi}$, using $\varphi_i(t)$ as the rate estimator, results in a more accurate estimate of the achievable

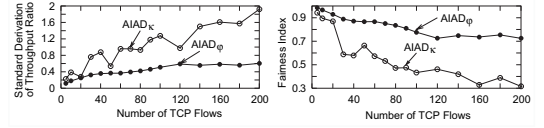


Fig. 3. Standard deviation of \hat{x}_i/x_0 , and fairness index $\phi = \frac{(\sum \hat{x}_i)^2}{N \sum \hat{x}_i^2}$.

rate, and hence, a more stable throughput. $\varphi_i(t)$ is more robust when ACKs are subject to congestion and loss.

The instability of system (4) manifests itself in the fluctuations and uneven distribution of the throughputs among the flows. In the second simulation, we increase the number of TCP flows from 5 to 200 and observe the throughput distributions. Let \hat{x}_i be flow i 's measured throughput; we define the *throughput ratio* of flow i as \hat{x}_i/x_0 . Fig. 3 plots the standard deviation of the throughput ratio as N increases. It shows that as N increases, algorithm AIAD $_{\kappa}$ leads to a much more diverse throughput distribution than does algorithm AIAD $_{\varphi}$. From the same simulation, we also plot, in the same figure, Jain's fairness index ϕ as defined in [6]. $\phi = 1$ indicates a perfectly even distribution of resource, and $\phi = \frac{1}{N}$ implies that all but one flow gets the entire resource. A stable congestion control algorithm should result in a fair and stable ϕ as N increases. Fig. 3 confirms our analysis in Section III that AIAD $_{\kappa}$ becomes unstable and unfair when N increases.

Also, recall (7), in the limiting case, we have $\alpha\eta < 1$. In our simulation settings, this is equivalent to $Q > 24Cd$, where Q is the total combined buffer size of the forward and backward bottleneck links. We have also verified by simulations (not shown here) that the performance of AIAD $_{\kappa}$ with buffer size of 200 packets becomes close to that of AIAD $_{\varphi}$ with buffer sizes of 20 packets.

V. CONCLUSIONS

In this letter, we have analyzed and verified via packet-level simulations that, in the TCP protocol using the rate estimation based AIAD algorithm, the ACK rate estimator $\kappa_i(t)$ is not upper bounded by the forward bottleneck capacity and will result in significant over estimation when the backward path experiences congestion; AIAD $_{\kappa}$ tends to become unstable and unfair as the number of competing flows increases. Whereas, the receiving rate estimator $\varphi_i(t)$ is robust to reverse congestion; when used in the AIAD algorithm, the system maintains its stability and fairness as the number of flows increases.

REFERENCES

- [1] V. Jacobson, "Congestion avoidance and control," *Comp. Commun. Rev.*, vol. 18, pp. 314-329, Aug. 1988.
- [2] C. Casetti, M. Gerla, S. Mascolo, M. Y. Sanadidi, and R. Wang, "TCP Westwood: bandwidth estimation for enhanced transport over wireless links," in *Proc. ACM/IEEE MobiCom 2001*, 2001, pp. 287-297.
- [3] V. Jacobson, R. Braden, and D. Borman, "TCP Extensions for High Performance," *RFC 1323*, 1992.
- [4] C. Hollot, V. Misra, D. Towsley, and W.-B. Gong, "A control theoretic analysis of RED," in *Proc. IEEE INFOCOM 2001*, 2001, pp. 1510-1519.
- [5] R. Johari and D. K. H. Tan, "End-to-end congestion control for the Internet: delays and stability," *IEEE/ACM Trans. Networking*, vol. 9, pp. 818-832, Dec. 2001.
- [6] R. Jain, D. Chiu, and W. Hawe, "A quantitative measure of fairness and discrimination for resource allocation in shared computer systems," *DEC Research, Tech. Rep. TR-301*, 1984.