# A Fast-Converging TCP-Equivalent Window-Averaging Rate Control Scheme

Shih-Chiang Tsao[1*], Yuan-Cheng Lai[2], Ying-Dar Lin[3]
[1,3]Department of Computer Science
National Chiao Tung University (NCTU)
Hsinchu, Taiwan
{[1*]weafon, [3]ydlin}@cs.nctu.edu.tw

[2]Department of Information Management
National Taiwan University of Science and Technology (NTUST)
Taipei, Taiwan
[2]laiyc@cs.ntust.edu.tw

*Abstract*- Smooth rate control schemes are necessary for Internet streaming flows to use available bandwidth. To equally share the Internet bandwidth with existing Transmission Control Protocol (TCP) flows, these new schemes should meet TCP-equivalent criterion, i.e., achieve the same transmission rates as TCP under the same network conditions. However, when the available bandwidth oscillates, many of these schemes fail to meet this criterion due to their slow increasing rate. This study proposes a window-averaging rate control (WARC) scheme to send packets at the same average rate as a TCP flow over a fixed time interval. Considering the TCP rate only over a fixed interval allows WARC to forget the historical loss condition more rapidly than other schemes, thus achieving a faster increasing rate when additional bandwidth becomes available. When the available bandwidth drops dramatically, WARC uses a history-reset procedure to converge its rate to the new steady rate immediately after a specified number of losses. The simulations in this study show that WARC not only achieves the same bandwidth as TCP, but exhibits faster convergent behaviors and has a smoother rate than existing schemes.

***Keywords- TCP-friendly, congestion control, aggressive.***

## I. INTRODUCTION

Transmission Control Protocol (TCP) is a widely used transport protocol in the Internet. Its rate control mechanism, Additive-Increase and Multiple-Decrease (AIMD), frequently and dramatically adjusts the transmission rate to detect the available bandwidth for transmitting packets and to avoid sequential packet losses. However, TCP may not be suitable for streaming because of its high-frequency and large-scale oscillatory rate. The playback of streaming data pauses whenever the TCP sending rate falls below the streaming encoding rate and the receiver has not buffered the streaming data. Although allocating a large buffer can alleviate the oscillatory rate problem, it prolongs the start-up latency for on-demand media traffic [1] and causes an intolerable delay for interactive applications, such as video conferences [2][3].

Researchers have proposed many rate control schemes to address the oscillatory-rate problem [4]-[9] and transmit streaming data at a smooth rate. As indicated in [10], some of these schemes, e.g., GAIMD [4] and TEAR [7], increase their rate slowly to maintain *long-term smoothness* even when additional bandwidth is available. On the other hand, when bandwidth is unavailable, they must decrease the rate as much as TCP. This is because they must be TCP-compatible [11], i.e., responding to network congestion and using no more bandwidth than TCP. Consequently, the streaming data carried by these schemes uses less bandwidth than TCP on average. This creates a potential barrier to promoting these schemes for streaming media even though their smooth rate is suitable for streaming than TCP.

TCP equivalence, i.e., using the same bandwidth as TCP, is important for promoting new rate control schemes, but this objective is hard to achieve in a TCP-compatible flow striving for the long-term smoothness. Fortunately, long-term smoothness may be unnecessary for streaming with dynamic coding rates, like Microsoft Windows Media Codec. This type of streaming can play at a higher coding rate to provide good quality when additional bandwidth is available. Therefore, its rate control scheme should increase the transmission rate as quickly as TCP in the transient state, i.e., when the available bandwidth changes drastically, to grab the available bandwidth. However, since changing the coding rate incurs overhead, the scheme must keep *rate smoothness* when the network conditions stay at steady state. This is a key difference from TCP, and prevents the coding rate from changing too often. SIMD [9] is a scheme for achieving this goal, and adopts exponentially aggressive behavior to converge its rate toward a new steady rate.

This study proposes a rate control scheme called window-average rate control (WARC). WARC is better at meeting TCP-equivalence requirements than existing schemes because it takes a short time to converge its rate toward the TCP whenever the available bandwidth drastically increases or decreases. WARC is the first scheme to provide smoother bandwidth than TCP when the available bandwidth remains stationary, and also equal to TCP *under any distributions* of inter-loss time (i.e., the time between two loss events[1]). Existing schemes [4]-[9] achieve this goal only under some specific assumptions, e.g., the packet losses occur periodically or with a fixed probability [10]. However, these assumptions may not be realistic in the Internet. Previous analysis [14] reveals that the packet loss condition experienced by an Internet flow may consist of multiple *minute-scale* stationary regions, and the time intervals of any two consecutive losses may follow an independently and identically distributed (i.i.d.) exponential distribution within a region.

The basic mechanism of rate control in WARC is called run-time estimation (RTE), which repeatedly estimates the mean rate of TCP and adjusts its sending rate to the estimated rate (packets/RTT). TFRCP [8], TFRC [5], and TEAR [7] also

[1] A loss event means an event causing a TCP flow that halves its congestion window. Such an event may consist of multiple sequent packet losses. For convenience, this study ignores the term "event," as in related works.

use the RTE control, but WARC differs from their mean rate estimates by averaging the *latest fixed-number* congestion windows (CWNDs) which a TCP flow may use to control its rate. Assuming RTT is fixed[2], such an estimated rate represents the mean rate of the TCP flow over a *fixed* time period, instead of over a dynamic time period as in TFRC[3] and TEAR. This difference allows WARC to use the same bandwidth as TCP under all stationary packet loss distributions, as the following discussion proves. Since the number of considered CWNDs is bounded, WARC forgets the early small CWNDs and uses a high increasing rate when additional bandwidth becomes available for a fixed time.

However, the RTE rate control in WARC may not be satisfactory under three special but realistic conditions. First, when the available bandwidth drops abruptly, RTE may encounter many losses before reducing its rate to the available bandwidth. Second, when passing through a link managed by the Drop-Tail queuing algorithm, RTE may exhibit slight rate oscillations. Third, under heavy losses, RTE uses more bandwidth than TCP. Therefore, this study proposes three complementing mechanisms to handle these three conditions: the history-reset procedure, one-RTT reduction mechanism, and the fluid-based timeout mechanism, respectively.

The remainder of this paper is organized as follows. Section 2 discusses related work, while Section 3 introduces the WARC scheme, including RTE and the three complementing mechanisms. Section 4 shows the evaluated results of WARC running in the *ns-2* simulator [15]. Finally, Section 5 provides conclusions.

## II. Related Work

Like the schemes proposed in [4]-[9], WARC aims to use a smooth and TCP-equivalent rate to send packets over the Internet. These schemes, such as TCP NewReno and SACK, consider packet loss as congestion signals to adjust their rates. In addition to packet losses, the RTT variation can indicate congestion, as in TCP Vegas [17] and FAST [18]. Although RTT-based schemes also provide a smooth rate and can carry streaming data [17], their usage of network bandwidth may be unequal to that used by the loss-based versions of TCP [19], which still carry most Internet traffic. C. Zheng and V. Tsaoussidis [20] recently proposed a scheme using both losses and RTTs, which may be a solution to the unfairness problem.

All the schemes mentioned above have low barriers to deployment on the Internet because they detect congestion through packet loss or/and delay and do not require any router feedback. However, adjusting the rate only by loss and delay may not work well in high Bandwidth-Delay Product (BDP) networks [21][22], which the Internet may adopt in the future. Therefore, several studies [22][23] consider a tight cooperation model between congestion control schemes and routers. XCP [22] is such a control scheme, and requires multiple-bits congestion-related feedback from routers. However, since there is no space in the IP header to carry these bits, XCP has a high barrier to deployment on the Internet.

## III. Window-Averaging Rate Control (WARC)

### A. Basic Rate-control Mechanism

The goal of a TCP-equivalent streaming rate-control scheme is to achieve a throughput that is smoother than but equivalent to the average TCP throughput. To achieve this goal, schemes such as TFRC and TEAR send packets at the mean rate of the TCP flow. These schemes repeatedly estimate the mean rate during the whole connection since the mean rate within a period changes according to network conditions. This study calls this rate control approach the run-time estimation (RTE) model[4].

WARC, which inherits the RTE model, considers two key designs. The first is how to estimate the present mean rate (packet/RTT) of a corresponding TCP flow, and the second is how often to adjust the controlled flow rate. For the first point, WARC gets the mean TCP rate by averaging the latest $s$ CWNDs of a corresponding TCP flow, where $s$ is fixed. For the second point, WARC adjusts the rate based on RTT. This study uses CWND to represent the number of packets sent by a TCP flow in one RTT, while packet/RTT is the unit of rate.

The following discussion details the rate-control procedure between the sender and receiver of WARC. The sender sends data packets at the rate periodically assigned by the receiver. The receiver detects the packet losses using the sequence numbers of data packets, and it estimates the CWND used by a TCP flow. If the WARC receiver does not encounter losses, it increases the CWND by one per RTT, as TCP does. However, if it encounters a loss, it reduces the CWND by a half. The receiver then averages the latest $s$ CWNDs to determine the new transmission rate and reports the rate to the sender, repeating the averaging and reporting per RTT.

The following equation formally expresses the transmission rate used in a WARC sender based on the description above. Assume that RTT separates the real time $t$ into multiple rounds, i.e. $t = \{1, 2, ..., \infty\}$. Suppose that $R(t, s)$ denotes the transmission rate (packets/RTT) of the WARC sender, computed from $s$ CWNDs and used in the $t^{th}$ round. Then, $R(t, s)$ can be written as

$$R(t,s) = \frac{1}{\min(s,t)} \sum_{i=1}^{\min(s,\,t)} W(t-i), \qquad (1)$$

where $W(t)$ is the number of packets transmitted by a TCP sender, i.e. its CWND, in the $t^{th}$ round. Equation (1) contains a minimizing operation because there are at most $t$ CWNDs at the initial time ($t<s$) and thus $R(t, s)$ is the average of the latest $t$ CWNDs. When $t>s$, $R(t, s)$ represents the average of the latest $s$ CWNDs.

When the available bandwidth suddenly increases, the speed at which the scheme ignores the packet loss conditions

[2] Like related works [4]-[9] the analysis in this study assumes the RTT is fixed. However, in the implementation of WARC, the value of RTT refers to the smoothed RTT, which is dynamically updated by the algorithm usually used in TCP.

[3] Actually, TFRC was proposed as a rate control protocol allowing the use of different methods to estimate the mean rate. The method discussed in this work and literatures is based on the descriptions of TFRC in [5].

[4] We do not think that the familiar terms like "rate-based" or "equation-based" can represent such an idea. A RTE-based scheme would be a window-based scheme if using a window to control the release of packets. Besides, TEAR is not an equation-based scheme, but a RTE-based scheme as revealed later.

measured before the increase determines whether it has a fast aggressive behavior, i.e., a high increasing rate. WARC is more aggressive than TFRC and TEAR because it excludes these conditions from the rate computing after a fixed number of RTTs. Conversely, TFRC and TEAR excludes these conditions after a fixed number of packet losses, say 8 losses. Unfortunately, the latter two schemes may have to wait a long time to meet the first loss due to the sudden increase of bandwidth; thus they have a slow aggressive behavior. In fact, both schemes require additional rules to speed up the behavior. However, the following simulation results show that these schemes are still conservative.

### B. Complemental Rate-control Mechanisms

*1) History-reset (HR) procedure for responsiveness under bursty-losses:* The basic control mechanism in WARC has a conservative responsive behavior that allows WARC to achieve a smooth rate, but also results in a slow response to an abrupt increase in the packet loss rate. Thus, this study proposes a history-reset procedure to respond to abrupt changes right after a fixed number of loss events.

The procedure is invoked if the average TCP rate spanning the latest $N$ packet losses, denoted as $\overline{R}_{TCP}(N)$, is smaller than or equal to $1/K$ of the current rate of WARC, which can be expressed as

$$\overline{R}_{TCP}(N) \le \tfrac{1}{K} R(t,s), \qquad (2)$$

where $K$ is a constant larger than 1, and its effect on the HR procedure would be discussed later. $\overline{R}_{TCP}(N)$ is calculated by the formula

$$\overline{R}_{TCP}(N) = \tfrac{3}{2}\tfrac{1}{N}\sum_{j=1}^{N} X(-j), \qquad (3)$$

where $X(-j)$ represents the number of rounds in the *last* $j^{\text{th}}$ epoch, i.e., the period between the $j^{\text{th}}$ and $(j+1)^{\text{th}}$ last losses. According to [16], the $\overline{R}_{TCP}(N)$ produced by Eq. (3) represents the average rate that a TCP flow may have under a loss condition where the number of inter-loss rounds is $\frac{1}{N}\sum_{j=1}^{N} X(-j)$. The following $\sum_{j=1}^{N} X(-j)$ is denoted as $S(N)$.

Figure 1 shows that if Eq. (2) is true at the time $T$, then the packet loss condition has changed abruptly because the fast-responsive TCP has a far smaller rate than WARC. Then, the HR procedure eliminates all $W(t)$'s where $t < T - S(N)$ from the computation of Eq. (1). Contrarily, it retains the latest $S(N)$ CWNDs in the rate computing. By eliminating the old CWNDs, WARC quickly jumps to the mean rate that TCP uses at the time $t$.

Two parameters, $K$ and $N$, in Eq. (2) control the threshold of invoking the history-reset procedure. $K$ decides how many rate differences between WARC and TCP denote an abrupt change in packet loss condition. $N$ decides how long the difference should persist before invoking the HR procedure. With a small $K$ and $N$, the procedure will be invoked for small and short changes, which may damage the smoothness of WARC. Contrarily, with large $K$ and $N$, the procedure may be conservatively invoked and lose track of its objective of quickly responding to a change in packet loss condition. A tradeoff exists here. We analyzes this tradeoff and suggests that with $K$=3 and $N$=12, the procedure has enough fast responsiveness and does not over-invoke to damage the smoothness.

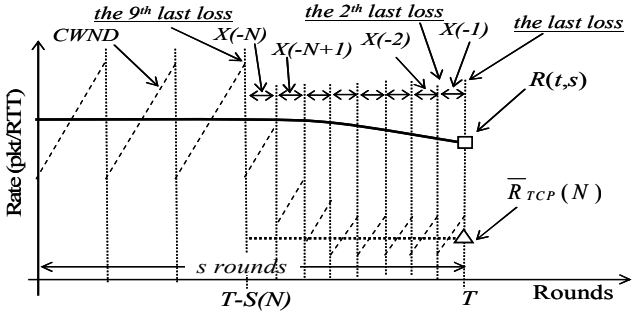


Fig. 1. The HR procedure is invoked when $\overline{R}_{TCP}(N) < 1/K\, R(t,s)$.

*2) One-RTT reduction procedure for smoothness under low-multiplexing networks*: The rate-based flow control scheme may encounter a series of packet losses and exhibit oscillatory rate behavior when passing through a low-multiplexing network [11], e.g. passing alone a link managed by Drop-Tail. This is because the rate-based scheme does not have the same per-packet ACK mechanism as the window-based scheme. A self-clocked mechanism was proposed in [12] to assist the TFRC in such a case. The mechanism bounds the rate increase of TFRC according to the received rate measured at the receiver. However, this mechanism may slow down the aggressive ability of a scheme because it limits the rate increase.

To retain fast aggressiveness, WARC adopts this mechanism only when packet loss occurs. When detecting a loss, the WARC receiver requests the sender to *temporarily* reduce the transmission rate used in the next round to 7/8 of the received rate. The temporality means that the reduction holds only for one RTT. After one RTT, the rate returns to the value computed by Eq. (1).

*3) Fluid-based timeout (FTO) procedure for fairness under heavy-losses*: FTO emulates the CWNDs of TCP running under the timeout mechanism. These CWNDs are involved in the rate computing of Eq. (1) to influence the rate. This complement prevents WARC from grabbing too much bandwidth when competing with TCP under heavy packet loss conditions.

## IV. Simulation Results

This section shows that WARC has better behaviors than eight TCP-friendly schemes in terms of 4 properties: fairness, smoothness, aggressiveness, and responsiveness, by running the *ns*-2 simulator [15]. In this study, a TCP-friendly scheme represents it aims to achieve TCP-equivalence. Table 1 describes the control parameters of each scheme used in the simulation, which were suggested individually according to the corresponding papers.

TABLE 1. The Control Parameters used in Each Scheme

| Scheme | Control Parameters | Ref. |
|---|---|---|
| WARC | s=160, $K$=3, $N$=12 | |
| GAIMD | α=0.2, β=0.125 | [4] |
| IIAD | α=1.0, β=0.67, $k$=1, $l$=0 | [6] |
| SQRT | α=1.0, β=0.67, $k$=0.5, $l$=0.5 | [6] |

| | | |
|---|---|---|
| SIMD | β=0.0625, *k*=-0.5, *l*=1 | [9] |
| AIAD/H | β=0.25, *k*=0, *l*=0 | [9] |
| TFRCP | Interval=5 seconds | [8] |
| TFRC | The number of samples=8 | [5] |
| TEAR | The number of samples=8 | [7] |

### A. Fairness

*1) Simulation topologies:* WARC and other selected schemes are evaluated in two cases. The former uses an artificial packet loss link, as Fig. 2 shows, to drop packets based on mathematical model. This link is usually for testing the fairness between TCP and the TCP-friendly schemes because it ensures the identical loss conditions experienced by any two passing flows, which is a basic assumption for these schemes to perform fairness. The mathematical model used here follows a general exponential distribution, which has two degrees of freedom and thus allows its coefficient-of-variation to be changed while fixing its mean, or vice versa. The general exponential distribution can test the fairness behavior in terms of different means and different CVs. Allocating sufficient bandwidth for this link prevents the packets from being dropped due to overflow.

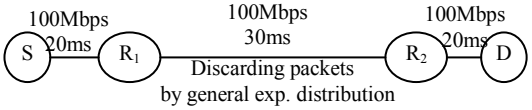


Fig. 2. The artificial loss-link topology is used to provide the same loss condition for any two flows running through the link $R_1$-$R_2$.

The second case makes all flows compete for a single bottleneck, which may be managed by the Drop-Tail or RED algorithms. The test verifies whether a TCP-friendly scheme is safe to deploy on the Internet. Figure 3 shows the dumbbell topology used in this test. The *v* TCP-friendly flows compete with *v* TCP flows for a single bottlenecked link. All flows have backlogged data for the whole testing period. This scenario tests whether *v* TCP-friendly flows can share the same bandwidth with *v* TCP flows under different levels of congestion. The capacity of the congested link is 15Mbps or 60Mbps, while that of other links is 100Mbps. The value *v* changes from 1 to 64. The propagation delay of the links from the sources to $R_1$ or from $R_2$ to the destinations is uniformly distributed between 10 to 30 microseconds. The queue sizes are 1.5 and 2.0 of bandwidth-delay product when $R_1$ is managed by Drop-Tail and RED, respectively. Actually, we tried setting different queue sizes for this experiment, but the results show that the queue size has an insignificant effect on the competition.

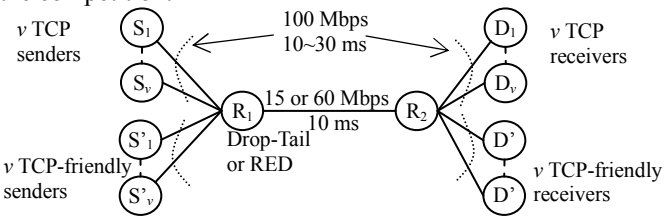


Fig. 3. The dumbbell topology used to test the fair sharing between TCP and TCP-friendly flows.

*2) Case I - Under artificial-loss links with different CVs*: The following results reveal the fairness behaviors of schemes under the link with different coefficient-of-variations in inter-loss time. The used link drops packets per *T* seconds where *T* follows a general exponential distribution with $E[T]$=5 and $CV[T]$ uniformly increases from 0 to 1. In this case, the link drops packets based on the escaping time from the last loss, instead of the number of received packets. This is because the dropping-by-time is a more realistic emulation of the loss conditions in highly multiplexing networks like the Internet [10]. This study also examines the fairness behaviors of schemes under the link with different coefficient-of-variations of inter-loss packets. However, we do not present these results because they are similar to those in Fig. 4.

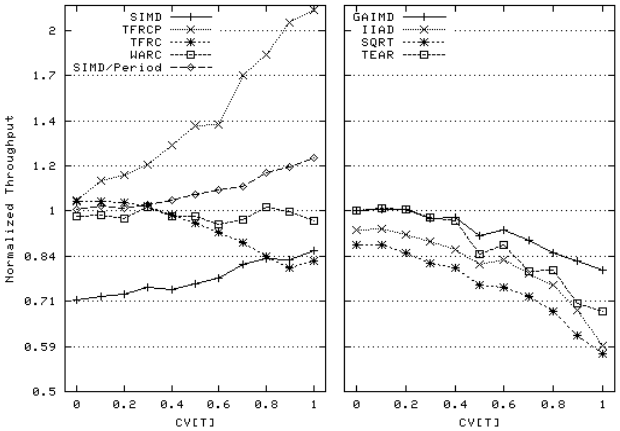


Fig. 4. The throughputs of the TCP-friendly schemes normalized with that of TCP under the artificial-loss links with different $CV[T]$. Results are separately plotted in two parts for clarity.

Figure 4 shows that WARC uses the same throughput as TCP under all $CV[T]$'s because it is supposed to achieve fairness under stationary losses, as Section 3 shows. Contrarily, most schemes only achieve fairness when $CV[T]$=0, i.e., as the loss occurs periodically, because these schemes assume periodic losses. Actually, this assumption is not applicable to loss patterns on the Internet. The inter-loss time on the Internet may approximate an i.i.d. exponential distribution [14], which is the link with $CV[T]$=1. Under $CV[T]$=1, WARC achieves fairness, but GAIMD and TFRC only have about 80% throughput of TCP while TEAR, IIAD, SQRT, and AIAD/H have 60% on average.

*3) Case II - Under Drop-Tail or RED*: Figures 5 and 6 show that TCP individually competes with five TCP-friendly schemes under the four congested link configurations: (Drop-Tail, 15Mbps), (Drop-Tail, 60Mbps), (RED, 15Mbps), and (RED, 60Mbps). For each configuration, these two figures show five schemes for clearness. Figure 5(a)(c) and Fig. 6(a)(c) show that WARC has almost the same behavior as TFRC, equally sharing the bottleneck bandwidth with TCP under the four configurations.

Figure 5(c) shows that WARC, as well as TFRC and TEAR, has a slightly lower throughput than TCP because the rate-based control mechanism adopted in these three schemes may experience a bit higher loss ratio than the window-based control mechanism in GAIMD and SIMD. These additional losses result are because the former does not really control the data packet transmission by the received acknowledgement

packet, and thus cannot respond to the overflow of the Drop-Tail queue within a RTT.

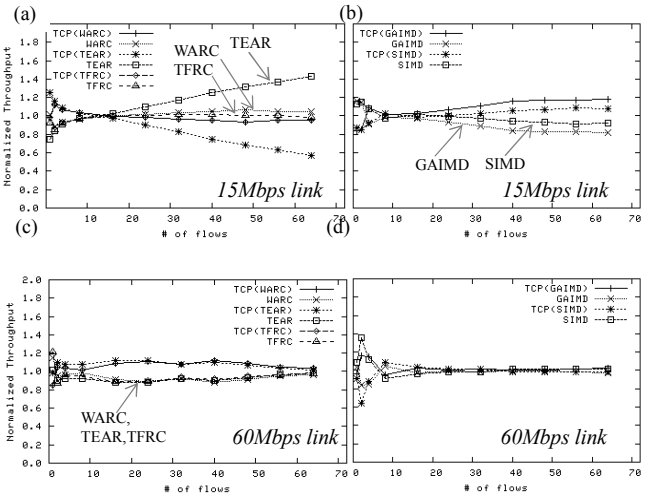


Fig. 5. The competing results between TCP and five TCP-friendly schemes under the links managed by Drop-Tail are shown. TCP(*X*) plots the average normalized throughput of TCP flows which compete with the flows

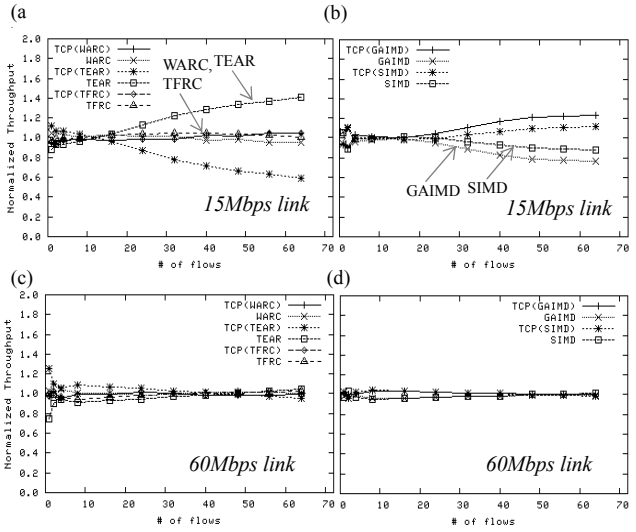


Fig. 6. The competing results between TCP and five schemes under the links managed by RED are shown.

## *B. Smoothness*

We adopt the test presented in [2] to demonstrate that the smoothness of WARC is favorable for streaming. This test uses TCP to carry the traffic pushed from a CBR streaming source and examines how much pre-buffering time is enough to alleviate the oscillatory rate of TCP. One streaming flow competes with 15 TCP flows for a bottleneck link with 16 Mbps capacity managed by RED. Each flow should get 1Mbps bandwidth, which is 1.2 times of the data rate pushed from the streaming source.

The left side of Fig. 7 shows the testing results when different schemes carry streaming traffic. Each curve is averaged from 10 tests of 5 minutes. When TCP carries the streaming, it requires a pre-buffering time of 4s to ensure that less than 1% of packets arrive later than their expected time for playback. This time is difficult to tolerate in a video conference. Unlike TCP, WARC needs only 0.1s of pre-buffer time to keep the ratio of late packets lower than 1%. The right side of Fig. 17 plots each scheme according to the number of packets sent per 0.1 second. Obviously, the excellent short-term smoothness of WARC and TFRC is the reason for their lower ratio of late packets than SIMD and SQRT.

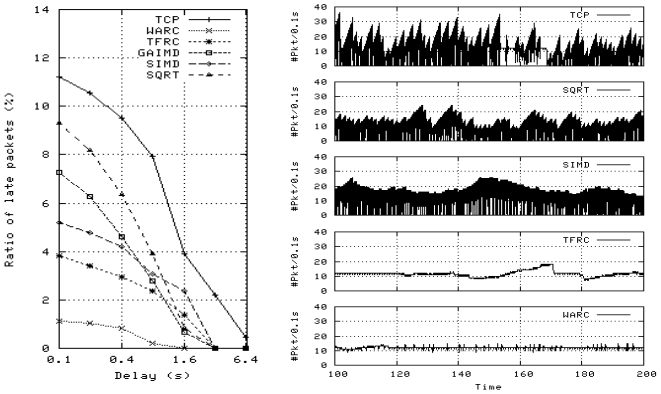


Fig. 7. Comparison on the ratio of late packets and the rate variations of TCP and TCP-friendly schemes

## *C. Aggressiveness and Responsiveness*

To demonstrate the fast-convergent behavior in WARC, Fig. 8 uses a bursty arrival and departure of TCP traffic to bring dramatic changes on the packet loss condition and thus provide required transient-state scenarios. In each test, TCP-friendly scheme or TCP controls the flow S-D to compete with 16 TCP flows active for the entire 350-second testing time and with 8 TCP flows, which are only active for 0~100s and 250~350s. That is, the available bandwidth for the tested flow will increase at the 100$^{th}$ second and decrease at the 250$^{th}$ second.

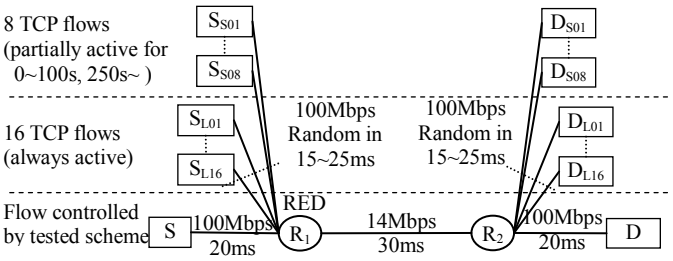


Fig. 8. The topology with oscillating CBR background traffic, used to test TCP-friendly schemes in terms of aggressiveness and responsiveness

Figure 9 first shows that WARC has fast aggressiveness when additional bandwidth becomes available at the 100$^{th}$ second. Like SIMD, WARC converges toward the new rate within about 20 seconds, which is shorter than GAIMD, TFRC, and TEAR. These fast aggressiveness results are because WARC forgets the measured packet loss conditions earlier than a fixed number of RTTs. However, Section 2.1 shows that TEAR and TFRC cannot do this.

On the other hand, using the HR procedure, WARC, like TFRC and TEAR, displays fast responsiveness at the 255$^{th}$ second. A closer look at Fig. 9 around the 250$^{th}$ second reveals that the HR procedure in WARC is invoked at the 255$^{th}$ second. Once the HR procedure is invoked, the rate of WARC

immediately reduces to the expected TCP rate.

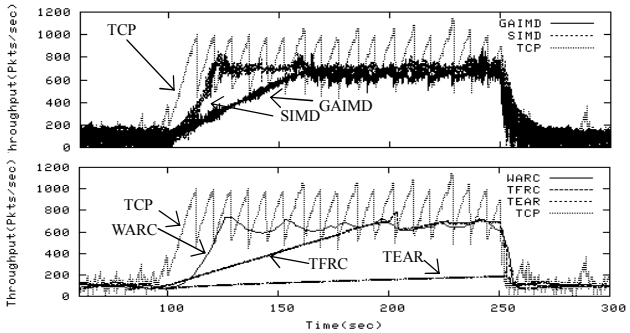


Fig. 9. The comparison between five TCP-friendly schemes on aggressiveness and responsiveness under the on/off CBR background traffic

## V. CONCLUSIONS

This study proposes a window-averaging rate control (WARC) scheme to simultaneously meet the four requirements of a TCP-friendly streaming-carrying scheme: fairness, smoothness, aggressiveness, and responsiveness.

WARC uses the run-time estimation (RTE) model to adjust the rate. This model averages the rate over a constant number of CWNDs, leading to its two inherent advantages: *fairness under stationary* conditions and *fast aggressiveness*, neither of which appear in most existing well-known schemes, such as GAIMD, TFRC, TEAR, and IIAD. WARC also employs three complementing mechanisms including the history-reset procedure, the one-RTT reduction procedure, and the fluid-based timeout mechanism. These mechanisms handle three special but realistic loss conditions: bursty-losses, low-multiplexing network, and heavy-losses, respectively.

The results of this study show that WARC, like SIMD, exhibits fast aggressive behavior and achieves better tradeoffs between smoothness and aggressiveness than other schemes. That is, WARC takes a shorter time to converge its rate, while providing a smoother rate most of the time. The HR procedure further ensures that WARC can respond to abrupt decreases in available bandwidth immediately after a fixed number of packet losses. Simulation results confirm that WARC uses the same bandwidth as TCP under stationary *non-periodic* losses, where most schemes fail. Results also show that WARC uses the bandwidth close to TCP's even under the oscillating background traffic.

While WARC uses the same bandwidth as TCP under cases in which other well-known TCP-friendly schemes cannot, WARC still provides a smoother rate than all of them. While WARC provides far faster aggressiveness behavior than other schemes, it also quickly decreases its rate after an abrupt decrease in available bandwidth immediately after a fixed number of loss events. In the future, we plan to study the influence of a slow-start mechanism on bandwidth sharing between WARC and TCP to ensure that WARC is suitable for more general network conditions.

## REFERENCES


[1] L. Guo, E. Tan, S. Chen, Z. Xiao, O. Spatscheck and X. Zhang, "Delving into Internet Streaming Media Delivery: A Quality and Resource Utilization Perspective," in *Proc*. of ACM IMC'06, Oct. 2006.

[2] B. Wang, J. Kurose, P. Shenoy, and D. Towsley, "Multimedia Streaming via TCP: An Analytic Performance Study", in *Proc.* of ACM MM'04, Oct. 2004.

[3] S. Boyden, A. Mahanti, and C. Williamson, "TCP Vegas Performance with Streaming Media," in *Proc.* of IEEE IPCCC 2007, Apr. 2007.

[4] Y. Yang and S. Lam, "General AIMD Congestion Control," in *Proc.* of *IEEE ICNP 2000*, Nov 2000, pp. 187-198.

[5] S. Floyd, M. Handley, J. Padhye, and J. Widmer, "Equation-based Congestion Control for Unicast Applications," in *Proc.* of *ACM SIGCOMM'00*, Aug 2000, pp. 43-56.

[6] D. Bansal and H. Balakrishnan., "Binomial Congestion Control Algorithms," In *Proc. of IEEE INFOCOM*, April 2001, pp. 631-640.

[7] I. Rhee, V. Ozdemir, and Y. Yi, "TEAR: TCP Emulation at Receivers-Flow Control for Multimedia Streaming," Tech. Rep., Department of Computer Science, NCSU, Apr. 2000.

[8] J. Padhye, J. Kurose, D. Towsley, and R. Koodli., "A Model Based TCP-friendly Rate Control Protocol," In *Proc. of NOSSDAV*, June 1999.

[9] Shudong Jin, Liang Guo, Ibrahim Matta, and Azer Bestavros, "A Spectrum of TCP-friendly Window-based Congestion Control Algorithms," *IEEE/ACM Transactions on Networking*, vol. 11, no. 3, pp. 341-355, June 2003.

[10] Shih-Chiang Tsao, Yuan-Cheng Lai, and Ying-Dar Lin, "Taxonomy and Evaluation of TCP-Friendly Congestion-Control Schemes on Fairness, Aggressiveness, and Responsiveness," IEEE Network, Vol. 21, No. 6, pp. 6-15, November/December 2007.

[11] B. Braden, D. Clark and J. Crowcroft, "Recommendations on Queue Management and Congestion Avoidance in the Internet," RFC 2309, Apr 1998, Available at http://www.ietf.org.

[12] D. Bansal, H. Balakrishnan, S. Floyd, S. Shenker, "Dynamic Behavior of Slowly-responsive Congestion Control Algorithms," in *Proc.* of *ACM SIGCOMM'01*, Aug. 2001, pp. 263-274.

[13] Milan Vojnovic and J.-Y. Le Boudec, "On the Long-Run Behavior of Equation-Based Rate Control," *IEEE/ACM Transactions on Networking*, vol. 13, no 3, pp. 568-581, June 2005.

[14] Y. Zhang, N. Duffield, V. Paxson, and S. Shenker, "On the Constancy of Internet Path Properties, " in *Proc*. of *ACM SIGCOMM* Internet Measurement Workshop, Nov. 2001, pp 197-211.

[15] The Network Simulator - ns-2, ver. 2.28, http://www.isi.edu/nsnam/ns/.

[16] E. Altman, K. Avrachenkov, and C. Barakat, "A Stochastic Model of TCP/IP with Stationary Random Losses," IEEE/ACM Transactions on Networking, vol. 13, no. 2, pp. 356-369, April 2005.

[17] L. Brakmo and L. Peterson, "TCP Vegas: End to End Congestion Avoidance on a Global Internet," *IEEE Journal on Selected Areas in Comm.*, vol 13, no. 8, Oct. 1995, pp. 1465-1480.

[18] D. Wei, C. Jin, S. H. Low, and S. Hegde, "FAST TCP: Motivation, Architecture, Algorithms, Performance, " IEEE/ACM Transactions on Networking, vol. 14, no. 6, pp. 1246-1259.

[19] K. A. Tang, J. Wang, S. Hegde, and S. H. Low, "Equilibrium and Fairness of Networks Shared by TCP Reno and FAST", Telecommunications Systems special issue on High Speed Transport Protocols, vol. 30, no. 4, pp. 417-439, Dec. 2005.

[20] C. Zhang and V. Tsaoussidis, "TCP Smoothness and Window Adjustment Strategy", *IEEE Transactions on Multimedia*, vol 8, no. 3, pp. 600-609, June 2006.

[21] S. H. Low, F. Paganini, J. Wang, and J. Doyle, "Linear Stability of TCP/RED and a Scalable Control," Computer Networks Journal, vol. 43, no. 5, pp. 633-647, Dec. 2003.

[22] D. Katabi, M. Handley, and C. Rohrs, "Congestion Control for High Bandwidth-Delay Product Networks," in *Proc.* of *ACM SIGCOMM*'02, Aug. 2002, pp. 89-102.

[23] Y. Xia, L. Subramanian, I. Stoica, and S. Kalyanaraman, "One More Bit Is Enough", in *Proc.* of *ACM SIGCOMM*'05, Aug. 2005