

Second-Order Rate-Control Based Transport Protocols

Xi Zhang and Kang G. Shin

Real-Time Computing Laboratory
 Department of Electrical Engineering and Computer Science
 The University of Michigan
 Ann Arbor, MI 48109-2122, USA
Email: {xizhang,kgshin}@eecs.umich.edu

Abstract—We propose an efficient flow and error control scheme for high-throughput transport protocols by using a second-order rate control, called the α -control, and a new sliding-window scheme for error control. The α -control minimizes the packet retransmissions by adjusting the rate-gain parameter to the variations in the number and round-trip times (RTTs) of cross-traffic flows that share the bottleneck. Using selective retransmission, the sliding-window scheme guarantees lossless transmission. By applying the α -control, the proposed scheme can drive the flow-controlled system to a retransmission-less equilibrium state. Using the fluid analysis, we establish the flow-control system model, obtain the greatest lower bound for the target buffer occupancy, and derive closed-form expressions for packet losses, loss rate, and link-transmission efficiency. We prove that the α -control is feasible and optimal linear control in terms of efficiency and fairness. Also presented are the extensive simulation results that confirm the analytical results, and demonstrate the superiority of the proposed scheme to others in dealing with the variations of cross-traffic flows sharing the same bottleneck and their RTTs, controlling packet losses/retransmissions, and achieving buffer-usage fairness as well as high throughput.

Index Terms—High-throughput transport protocol, second-order rate control, decoupled flow and error control, Internet, TCP/IP, TCP-Friendly.

I. INTRODUCTION

There has been a growing number of applications of bulk data transmission over wide-area networks. The two key requirements of any bulk data-transfer protocol are high throughput and transmission reliability. In theory, a packet-switched network allows a best-effort user to have as much a network-capacity share as is available. In reality, however, an achievable end-to-end throughput is often an order-of-magnitude lower than the network capacity. Throughput is often limited by the underlying transport protocol, particularly by its flow and error control mechanisms. It is difficult to achieve both high throughput and transmission reliability along long-delay, high-bandwidth, and unreliable network paths. The network unreliability, delay, and unpredictable cross-traffic are the major culprits for the low end-to-end performance of transport protocols.

There are mainly two types of flow-control schemes for transport protocols: window-based (e.g., TCP [1]) and rate-based (e.g., NETBLT [2]). The window-based scheme dynamically adjusts the upper-bound of the number of packets that the transmitter may send without receiving an acknowledgment from the receiver. In the rate-based scheme, the transmitter regulates its sending rate based on network-congestion feedback. The window-based scheme is cost-effective as it does not require any fine-grain rate-control timer, and the window size automatically limits the load a source can impose on the network. However, the window-based scheme

also introduces its own problems [2]. First, since the window scheme does not specify the speed of packet transmission within the flow-control window, it cannot make per-connection bandwidth guarantees for continuous media (CM) (e.g., audio and video) data [3]. Moreover, unregulated data rates of multiple users can easily generate a large instantaneous aggregate data rate at the bottleneck router, thus causing network congestion.

Second, the window scheme traditionally couples error and flow controls. This coupling is often problematic as it may create protocol design conflicts. For instance, while a large window is desired for high throughput, a small window is preferred to minimize the retransmission cost. In addition, mixing flow and error controls in one mechanism makes flow control vulnerable to packet losses and delays since packet loss and retransmission decrease transmission rate significantly. Third, the performance of the window scheme is RTT-dependent. Clearly, the window size must be larger for longer-RTT paths, but how large should it be? Theoretically, there does not exist any upper bound that is absolutely sufficient since it is proportional to $RTT \times$ [an unpredictable number of errors] in the worst case [2]. Unfortunately, RTT varies randomly with time, which further complicates selection of the proper window size. Moreover, a very large window for longer-RTT paths can in effect eliminate the window's flow-control function.

Finally, the window scheme works poorly with a retransmission timer due to the complicated timer design [4]. On one hand, a longer timer tends to close the flow-control window, and hence, reduces the transmission rate and link utilization. On the other hand, a shorter timer may easily cause false alarms which, in turn, trigger superfluous retransmissions. Moreover, the timer value is also a function of RTT, which varies randomly and is difficult to measure in the presence of packet losses.

To overcome some of the aforementioned problems with the two types of transport protocols, the authors of [2] proposed a rate-based flow-control transport protocol, NETBLT [2]. Differing from TCP, NETBLT employs the rate scheme and separates flow control from error control. Consequently, packet losses and retransmissions, which modify the error-control window, do not directly affect the rate at which data is injected into the network. This decoupling of error and flow control simplifies both components considerably. The original NETBLT targeted at matching the sender and receiver rates, but ignored the network-congestion problem. The revised NETBLT protocol applies the Additive-Increase and Multiplicative-Decrease (AIMD) algorithm to adapt the source rate to network congestion. However, this adaptation is effective only for the case of slowly-changing available network bandwidth

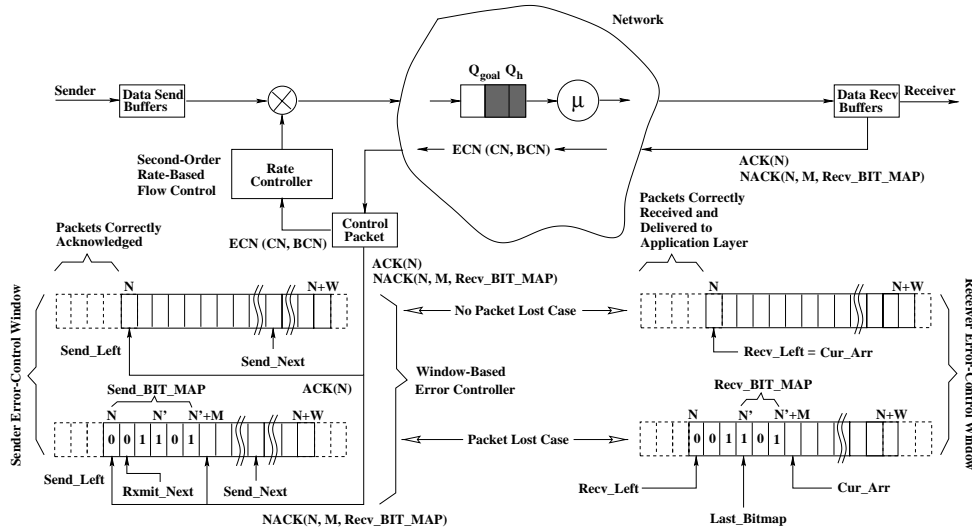


Fig. 1. The proposed flow and error control scheme.

since the source takes a rate-control action only once each time when an entire block of data packets have been transmitted and positively or negatively acknowledged. Consequently, the slow adaptive algorithm tends to cause either buffer overflow or underflow at the bottleneck. More importantly, as analyzed in [5], the AIMD algorithm itself cannot upper-bound the maximum queue length at the bottleneck since the queue length is a function of the superposition of the rate-gains parameters (i.e., rate ramp-up speed) of all traffic flows that share the same bottleneck and their RTTs. The unbounded bottleneck queue length can cause excessive packet losses, and thus costly retransmissions. It is difficult to control the queue at a bottleneck router because the number and RTTs of active cross-traffic flows sharing the same bottleneck are unknown *a priori* to the source and also vary randomly with time.

In this paper, we propose a second-order rate-control (called the α -control) scheme to cope with the variations of RTTs and the number of cross-traffic flows that share the same bottleneck. In particular, besides adapting the transmission rate based on congestion feedback, the source also adjusts the rate-gain parameter such that the number of retransmissions can be minimized while achieving a high throughput. Unlike the TCP that uses an implicit congestion signal, the α -control employs a mechanism, similar to Explicit Congestion Notification (ECN) [6, 7] set by an IP router, to detect an incipient congestion. The ECN-like mechanism can inform sources of congestion quickly and unambiguously, instead of making the source wait for either a retransmission timeout (TCP-Tahoe [1]), or three duplicate ACKs (TCP-Reno [8]), to infer network congestion. As a result, the early detection of congestion by using the ECN-like scheme can minimize packet losses and retransmissions caused by the TCP flow-control scheme itself [9, 10].

Moreover, the proposed scheme uses a new sliding-window scheme for error control, and also decouples it from the rate flow control. The error-control window can be chosen as large as resources permit for high throughput since the transmission rate is independent of the error-control window. Since the idea of decoupling error and flow control was first proposed in [2], it continues to draw considerable interest. A new error-control scheme called SMART (Simple Method to Aid ReTransmission) [11], also differentiates error control from flow control. Our scheme differs from SMART in that the SMART rate control is based on the packet-pair scheme while ours is based on the α -control, which is more cost-

effective than the packet-pair scheme. Realizing the inappropriateness of TCP for real-time applications due to the coupled error and flow control of TCP, the authors of [3] present a TCP-Friendly Rate Control Protocol (TFRC) that also separates error-recovery from congestion control. However, TFRC uses a formula-based feedback-loop approach for flow control which is different from α -control. We also use periodic exchange of state messages [12] between the sender and receiver to make the flow and error control performance virtually independent of RTTs. The proposed scheme uses selective retransmission to save bandwidth.

The paper is organized as follows. Section II describes the proposed scheme, and using fluid analysis, Section III establishes the flow-control system model, and derives performance measures and the greatest lower bound for the target buffer occupancy. Section IV models the packet-loss behavior and derives loss-control performance metrics. Section V analyzes efficiency and fairness of the α -control for multiple connections. Section VI evaluates and compares the proposed scheme with the other schemes via simulations. The paper concludes with Section VII.

II. THE PROPOSED SCHEME

Our proposed flow and error control scheme is illustrated in Fig. 1. Control packets are used to periodically convey both flow and error control information. The source sends a forward control packet periodically for every Δ time unit, and the receiver replies with a feedback control packet. The inter-control packet interval is typically a fraction of RTT. Control packet's flow-control information (ECN) is set by the receiver or IP routers when the control packet passes through in either direction, and error-control information (ACK/NACK) is updated by the receiver before returning a feedback control packet to the source. Upon arrival of a feedback control packet at the source, the control information is split into: (i) the flow-control information contained in ECN for the rate controller and (ii) the error-control information contained in ACK/NACK for the error controller. That is, the proposed scheme consists of separate flow-control and error-control mechanisms.

A. The Flow-Control Mechanism

The purpose of flow control is to dynamically adapt user demands to the available bandwidth and buffer capacities. As dis-

```

00. On receipt of Control Packet:
01. [1] Flow Control:
02. if ( $LCN = 1 \wedge CN = 0$ ) { ! Buffer congestion control condition
03.   if ( $BCN = 1$ ) {  $RIR := GDP \times RIR$ ; ! Dec RIR multiplicatively
04.     elseif ( $BCN = 0 \wedge LBCN = 0$ )
05.       {  $RIR := GIP + RIR$ ; ! Increase RIR additively
06.     elseif ( $BCN = 0 \wedge LBCN = 1$ )
07.       {  $RIR := RIR/GDP$ ; ! BCN toggles around target
08.      $RDP := e^{-RIR/BW_{EST}}$ ; ! RDP updating
09.      $LNMQ := 1$ ; ! Start a new measurement cycle
10.   if ( $CN = 0$ ) {  $R := R + RIR$ ; ! Increase source rate additively
11.   else {  $R := R \times RDP$ ; ! Decrease source rate multiplicatively
12.    $LCN := CN$ ;  $LBCN := BCN$ ; ! Save CN and BCN
13. [2] Error Control:
14.   if ACK( $N$ ) received { ! Positive Acknowledgment received
15.      $Send\_Left := N$ ; Discard packets with  $pkt\_seqn < N$ ;
16.   if NACK( $N, M, Recv\_BIT\_MAP$ ) received { ! NACK received
17.      $Send\_Left := N$ ; Discard packets with  $pkt\_seqn < N$ ;
18.      $Send\_M := Send\_M + M$ ; ! Update sender's bitmap length
19.      $Send\_BIT\_MAP \stackrel{\alpha}{\leftarrow} Recv\_BIT\_MAP$ ; ! Concatenate bitmap vectors.

```

Fig. 2. Pseudocode for sending end protocol.

cussed in [5], the traditional AIMD rate control, which only applies direct increase/decrease (thus the first-order) control of source rate $R(t)$, is not effective enough to upper-bound the maximum queue length Q_{max} with the buffer capacity C_{max} . This is because the first-order rate control can only make $R(t)$ fluctuate around the designated value, but cannot adjust the rate-fluctuation amplitude that determines Q_{max} . Consequently, the first-order control only exercises the control over bandwidth, but leaves bottleneck buffers un-controlled. In [5] Q_{max} is analytically shown to increase with both the rate-gain parameter and the connection's RTT. We proposed the second-order rate control, i.e., the α -control [5], to deal with RTT variations in an ATM ABR multicast tree.

In this paper, we propose the use of α -control to handle the variations in the superposition of rate-gain parameters of the traffic flows that share the same bottleneck, and their RTTs as well. Basically, α -control is a queue control mechanism at the bottleneck buffer, making Q_{max} converge to the target buffer occupancy Q_{goal} (setpoint) in response to the variations of both the number of traffic flows sharing the bottleneck and their RTTs. If the number of flows sharing the same bottleneck or RTT increases, Q_{max} will get larger. When Q_{max} eventually grows beyond Q_{goal} , the buffer will likely overflow, indicating that the current value of the superposed rate-gain parameter is too large. The sources of all the connections sharing the bottleneck must then reduce their rate-gain parameters to prevent packet losses and the subsequent costly retransmissions. On the other hand, when the measured $Q_{max} < Q_{goal}$, only a small portion of buffer is utilized, indicating that the current value of rate-gain parameter is too small for the reduced number of cross-traffic flows or their RTTs. The sources should increase their rate-gain parameters to avoid buffer under-utilization while improving the responsiveness by grabbing available bandwidth quickly.

Considering the need for controlling both network bandwidth and buffer, we define the following two types of congestion:

Bandwidth Congestion: If the queue length $Q(t)$ at a router becomes larger than a predetermined threshold Q_h , then the router sets the local CN (Congestion Notification) bit to 1.

Buffer Congestion: If the maximum queue length Q_{max} at a router exceeds Q_{goal} , where $2Q_h \ll Q_{goal} < C_{max}$ (see Theorem 1) and C_{max} is the buffer capacity, then the router sets the local BCN (Buffer Congestion Notification) bit to 1.

Unlike TCP that uses packet losses as an implicit congestion signal, we use an ECN-like scheme to detect incipient congestion and avoid unnecessary packet losses. While our bandwidth-congestion detection (CN -bit) is similar to the ECN mechanism, the buffer-

congestion detection (BCN -bit) differs from ECN since it provides one more dimension to control the dynamics of a flow-controlled system.

Fig. 2 shows a pseudocode of the source rate control algorithm. The flow-control information carried by the feedback control packet includes CN and BCN . The forward control packet carries a New Maximum Queue (NMQ) bit which is used by the source to notify the routers along the connection path to recalculate their maximum queue lengths. Upon receiving a feedback control packet, if the source detects a transition from rate-decrease to rate-increase — that is, when LCN (Local CN) is equal to 1, and the CN bit in the received control packet is 0 — then it is the time to exercise the buffer-congestion control, or α -control. The rate-gain parameter RIR (Rate-Increase Rate) is adjusted according to the one-step-old BCN value saved in the local BCN ($LBCN$) and the current BCN bit in the control packet just received. There are three cases to consider: (i) if $BCN = 1$ then RIR is decreased multiplicatively by a factor of GDP (Gain-Decrease Parameter) ($0 < GDP < 1$); (ii) if $LBCN = BCN = 0$ then RIR is increased additively by a step of size GIP (Gain-Increase Parameter) > 0 ; (iii) if $LBCN = 1$ and $BCN = 0$ then RIR is increased multiplicatively by a factor of GDP . For all of these three cases, the rate-decrease parameter RDP is adjusted according to the estimated bottleneck bandwidth BW_{EST} . Then, the local NMQ bit (saved in $LNMQ$) is marked and the received BCN bit is saved in $LBCN$ for the next α -control cycle. The source exercises the (first-order) rate control whenever a control packet is received. Using the same, or updated RIR and RDP , the source regulates its rate R using the AIMD algorithm, depending on the feedback CN bit ($= 0$ or 1) set by the receiver or IP routers.

B. The Error-Control Mechanism

The proposed scheme uses both NACK error detection and selective-retransmission recovery. Combining with selective retransmission, a NACK contains a range of the sequence numbers of packets that were lost and will be selectively retransmitted. This combination of NACK and periodic control-packet feedback eliminates the need for the usually-difficult timer design and minimizes the dependency of error and flow-control performance on RTT.

At the sender, all data packets are sequence-numbered, and put in the sender's buffer before their transmission as shown in Fig. 1. A transmitted packet is not removed from the buffer until it is correctly acknowledged. The transmitter maintains three sender-buffer pointer variables: (i) $Send_Left$ — the maximum packet sequence number below which all packets have been correctly acknowledged; (ii) $Send_Next$ — the sequence number of the packet to be sent next; (iii) $Rxmit_Next$ — the sequence number of the packet to be retransmitted. Associated with the error-control window at the transmitter is a sender-bitmap vector, $Send_BIT_MAP$ where bit 1 (0) indicates that the corresponding packet has (not) been acknowledged within the retransmission error-control window at the transmitter.

As shown in Fig. 1, the receiver maintains three buffer pointer variables: (i) $Recv_Left$ — the maximum packet sequence number below which all packets have been correctly received; (ii) Cur_Arr — the immediate-next packet sequence number that follows the packet received most recently; (iii) $Last_Bitmap$ — the value of Cur_Arr when sending the last feedback control packet in the last error-control cycle. If all packets are received

```

00. On receipt of Data Packet  $P(k, CN)$ :
01. [1] Flow Control:
02.  $Local\_CN := CN \vee Local\_CN$  ! Bandwidth congestion notification
03. [2] Error Control:
04. if  $(Cur\_Arr = Recv\_Left \wedge k = Cur\_Arr)$  {
05.    $Cur\_Arr := Cur\_Arr + 1$ ; ! Updating next expecting sequence number
06.    $Recv\_Left := Cur\_Arr$ ; ! Update left-edge sequence number
07.    $Last\_Bitmap := Cur\_Arr$ ; ! Update starting pointer position
08. }
09. if  $((Cur\_Arr > Recv\_Left \wedge k = Cur\_Arr) \vee (Cur\_Arr = Recv\_Left \wedge k > Cur\_Arr) \vee (Cur\_Arr > Recv\_Left \wedge k < Cur\_Arr))$  {
10.    $Recv\_BIT\_MAP[k - Last\_Bitmap] := 1$ ; ! Set new bitmap bit
11.    $Cur\_Arr := k + 1$ ; ! Update next expecting sequence number
12. }
13. if  $(Cur\_Arr > Recv\_Left \wedge k < Last\_Bitmap)$  {
14.   Received retransmission-packet processing;
15.   Deliver all packets in sequence to user; ! Sequentially deliver;
16.   Update  $Recv\_Left$ ; ! Update left-edge pointer of error window;
17. On receipt of Control Packet:
18. [1] Flow Control:
19.  $CN := CN \vee Local\_CN$ ; ! Bandwidth congestion notification
20. [2] Error Control:
21.  $N := Recv\_Left$ ; ! Correctly acknowledged packet sequence number
22. if  $(Recv\_Left = Cur\_Arr)$  { ! No lost packets;
23.    $send\_ACK := TRUE$ ; ! Need to send ACK message;
24. }
25. if  $(Recv\_Left < Cur\_Arr)$  { ! Lost packets not recovered yet;
26.    $M := Cur\_Arr - Last\_Bitmap$ ; ! Length of receiver bitmap vector
27.    $send\_ACK := FALSE$ ; ! Need to send NACK message
28. }
29. if  $(send\_ACK = TRUE)$  {
30.   send control packet  $(ECN(CN, BCN), ACK(N))$ ; ! Send ACK
31. }
32. else { send control packet  $(ECN(CN, BCN), NACK(N, M, Recv\_BIT\_MAP))$ ; ! Send NACK
33.    $Recv\_BIT\_MAP := 0$ ; ! Reset the current cycle's receiver bitmap
34.    $Last\_Bitmap := Cur\_Arr$ ; ! Update receiver bitmap starting position.

```

Fig. 3. Pseudocode for receiving end protocol.

correctly, then $Recv_Left = Cur_Arr$. When some packets are lost or received in error before Cur_Arr , a receiver-bitmap vector $Recv_BIT_MAP$ (see Fig. 1) for the current error-control cycle is used at the receiver to record which packet has (not) been received correctly during the current error-control cycle. The length of $Recv_BIT_MAP$ is determined by $M := Cur_Arr - Last_Bitmap$.

A pseudocode of the source error-control algorithm is given in Fig. 2. After receiving a feedback control packet, if the error-control message is $ACK(N)$, the transmitter first updates its $Send_Left$ by N ($Recv_Left$ at the receiver). Then, all packets with sequence numbers $< N$ are removed from the sender buffer. If the error-control message is $NACK(N, M, Recv_BIT_MAP)$, in addition to updating $Send_Left$ by N and removing all correctly acknowledged packets from the sender buffer, the transmitter increases $Send_BIT_MAP$'s length $Send_M$ by M , and concatenates $Send_BIT_MAP$ with $Recv_BIT_MAP$.

A pseudocode of the receiver error-control algorithm consists of data and control packet processing, as shown in Fig. 3. When a data packet $P(k, CN)$ is received, where k is the packet sequence number and CN is the ECN-bit marked by IP routers and carried in each data packet header, the receiver needs to deal with the following three cases:

- Condition $(Cur_Arr = Recv_Left) \wedge (k = Cur_Arr)$ indicates that no packets were lost (or all losses have been recovered) and the current arrival is also in correct order. So the receiver just needs to increase its three receiver-buffer control pointers by 1.
- Condition $((Cur_Arr > Recv_Left) \wedge (k = Cur_Arr)) \vee ((Cur_Arr = Recv_Left) \wedge (k > Cur_Arr)) \vee ((Cur_Arr > Recv_Left) \wedge (k < Cur_Arr))$ implies that there were lost but unrecovered packets, or there are new losses immediately before the current arrival, or both. In this case, the receiver needs to record the newly-lost packets and mark the just

```

00. On receipt of a DATA Packet  $P(CN)$ :
01. if (output link  $\neq$  busy) { send  $P(CN \vee Local\_CN)$  } ! Output packet;
02. elseif  $(size\_of(data\_queue) = \xi)$  { drop  $P(CN)$ ; } ! Packet loss occurs;
03. else { enqueue  $(data\_queue, P(CN))$ ; } ! Buffer this packet;
04. if  $(size\_of(data\_queue) > Q_h)$  {  $Local\_CN := 1$ ; } ! Bandwidth congestion
05. elseif  $(size\_of(data\_queue) < Q_l)$  {  $Local\_CN := 0$ ; } ! No bandwidth congestion;
06. if  $(size\_of(data\_queue) > Q_{max})$  {  $Q_{max} := size\_of(data\_queue)$ ; }
07. if  $(Q_{max} > Q_{goal})$  {  $Local\_BCN := 1$ ; } ! Buffer congestion;
08. else {  $Local\_BCN := 0$ ; } ! No buffer congestion;
09. On receipt of a feedback Control Packet  $P(CN, BCN)$ :
10.  $CN := Local\_CN \vee CN$ ; ! CN processing;
11.  $BCN := Local\_BCN \vee BCN$ ; ! BCN processing;
12. send Control Packet  $P(CN, BCN)$  to upstream node;
13. On receipt of a forward Control Packet  $P(NMQ)$ :
14. if  $(NMQ = 1)$  {  $Local\_BCN := 0$ ;  $Q_{max} := 0$ ; } ! New measurement cycle starts;
15. send control packet  $P(NMQ)$  to downstream node.

```

Fig. 4. Pseudocode for IP routers.

received packet in $Recv_BIT_MAP$ at the corresponding bit position specified by $(k - Last_Bitmap)$. Then, Cur_Arr is updated by its new value $k + 1$.

- Condition $(Cur_Arr > Recv_Left) \wedge (k < Last_Bitmap)$ means that the current arrival is a retransmission and there are still unrecovered losses. If $k = Recv_Left$, then the transport protocol can deliver this packet and all subsequent packets, if they are all in correct order, to the application layer. As correctly-acknowledged packets are removed from the receiver buffer, $Recv_Left$ is updated to its new position. However, if $k > Recv_Left$, then there must be a packet lost multiple times. We have developed an efficient false-alarmless algorithm to deal with loss of a packet multiple times, but omitted it here due to space limit.

When a control packet is received, the receiver needs to handle two cases: (1) if $Recv_Left = Cur_Arr$, indicating that no loss or all losses have been recovered. So, it returns $ACK(Recv_Left)$ to the source. (2) If $Recv_Left < Cur_Arr$, there are still unrecovered losses. So, it returns $NACK(N, M, Recv_BIT_MAP)$ to the source, where $N = Recv_Left$ and $M = Cur_Arr - Last_Bitmap$ (see Fig 3). Then, reset $Recv_BIT_MAP$ to 0. Whenever receiving a control packet, $Last_Bitmap$ is updated by Cur_Arr .

C. Flow and Error Control Algorithms at Routers

Fig. 4 shows a pseudocode of the IP router algorithm which handles three different events as follows.

Upon receipt of a data packet: forward it if the output link is idle. If the link is busy and its buffer is full, then drop this packet; else buffer the packet. Mark the $Local_CN$ bit (to set ECN-bit in the data packet header) if $Q(t)$ exceeds Q_h . $Local_BCN := 1$ (buffer congestion) if $Q_{max} > Q_{goal}$; $Local_BCN := 0$ otherwise.

When a feedback control packet received: mark both CN and BCN in the control packet by $Local_CN$ and $Local_BCN$, using an OR operation.

When a forward control packet received: if NMQ is set, starting a new rate-control cycle, then $Q_{max} := 0$ and also $Local_BCN := 0$ for the next buffer-congestion control.

III. THE SYSTEM MODEL AND ANALYSIS

A transport-layer connection under the proposed flow-control scheme is a dynamic feedback control system, which we model by applying the first-order fluid analysis [13–16]. We assume the existence of only a single bottleneck with queue length $Q(t)$ and

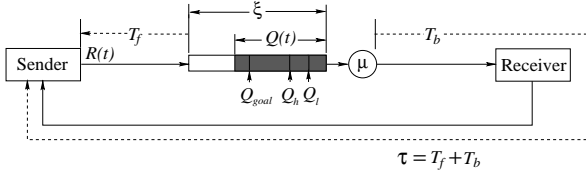


Fig. 5. System model for a transport-layer connection.

a “persistent” source,¹ which always has data packets to send at a rate $R(t)$ for each connection.

A. System Description and State Equations

Fig. 5 depicts the system model for a transport-layer connection under the proposed flow-control scheme. The connection model is characterized by a set of flow-control parameters. T_f represents the “forward” delay from the source to the bottleneck, and T_b the “backward” delay from the bottleneck to the source via the receiver. Clearly, $T_b = \tau - T_f$, where τ is the connection’s RTT. $R(t)$ is dictated by the bottleneck’s currently-available bandwidth μ . When $R(t) > \mu$, the bottleneck queue builds up, newly-arriving packets are dropped after $Q(t)$ reaches buffer capacity ξ . The bandwidth congestion (set $CN = 1$) or buffer congestion (set $BCN = 1$) is detected if $Q(t) > Q_h$ or $Q(t) > Q_{goal}$.

According to the rate-control algorithms described in Section II, the first-order (AIMD) rate control can be modeled by the following state equations:

$$R(t) = \begin{cases} R(t_0) + \alpha(t - t_0); & \text{If } Q(t - T_b) < Q_l \\ R(t_0)e^{-(1-\beta)\frac{(t-t_0)}{\Delta}}; & \text{If } Q(t - T_b) \geq Q_h \end{cases} \quad (1)$$

$$Q(t) = \int_{t_0}^t [R(v - T_f) - \mu] dv + Q(t_0). \quad (2)$$

where “additive increase” and “multiplicative decrease” are modeled by “linear increase” and “exponential decrease”, respectively, in a continuous-time domain [14]; $\alpha = \frac{1}{\Delta}(RIR)$ and $\beta = 1 + \log RDP$ for a rate-adjustment interval Δ , i.e., the control packet interval; t_0 is the last rate update time instant; and Q_h and Q_l are upper and lower queue-length thresholds, respectively, used to indicate the beginning and termination of the bandwidth congestion.²

The second-order rate control described in Section II is exercised only when the source rate control is in a “decrease-to-increase” transition based on the feedback BCN . According to our proposed flow-control scheme in Section II, and using Eq. (1), the second-order rate control can be modeled by the following equations in the continuous-time domain:

$$\alpha_{n+1} = \begin{cases} \alpha_n + p; & \text{if } BCN(n-1, n) = (0, 0), \\ q\alpha_n; & \text{if } BCN(n) = 1, \\ \frac{1}{q}\alpha_n; & \text{if } BCN(n-1, n) = (1, 0), \end{cases} \quad (3)$$

where $p = \frac{1}{\Delta}(GIP)$ ($p > 0$) and $q = GDP$ ($1 > q > 0$) for rate-adjustment interval Δ , and $\alpha_n > 0, \forall n = 1, 2, \dots, \infty$. Since the second-order rate control is applied to $\alpha = \frac{dR(t)}{dt}$, we also call it α -control.

To balance $R(t)$ ’s increase and decrease rates and to ensure the average of the offered traffic load not to exceed the bottleneck

¹ The single bottleneck and persistent-source assumption is only needed for the fluid modeling analysis [13–15], but is not necessary for the simulations.

² $Q_l > 0$ (but typically smaller than Q_h) is used to increase average throughput and bandwidth utilization by allowing the source to start increasing its rate earlier in the next rate-control cycle before the queue drains out and source rate is decreased to too low. Also, we choose $Q_{goal} \gg 2Q_h$ for the reasons described in Theorem 1.

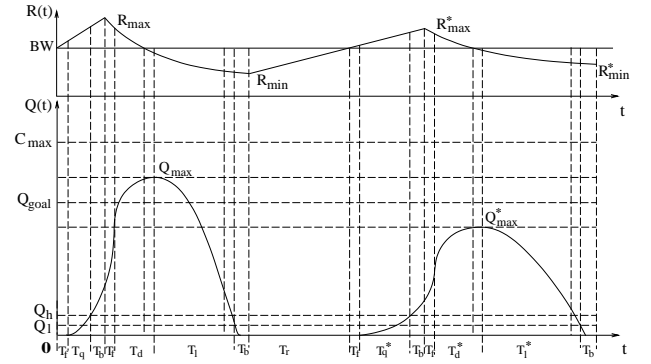


Fig. 6. Dynamics of $R(t)$ and $Q(t)$ for $Q_{max} < \xi (= C_{max})$.

bandwidth, each time when α_n is updated by the α -control law specified by Eq. (3), the proposed algorithm also updates the rate decrease factor by $\beta_n = 1 - \frac{\alpha_n}{\mu} \Delta$ accordingly.

B. Rate-Control Performance Analysis

Using Eqs. (1)–(2) for the case of $Q_{max} < \xi$, we derive a set of rate-control performance measures. We only list some of them, which will be used in this paper. Fig. 6 illustrates the dynamic behavior of $R(t)$ and $Q(t)$. The maximum rate is given by

$$R_{max} = \mu + \alpha(T_q + T_f + T_b) = \mu + \alpha(T_q + \tau) \quad (4)$$

where $T_q = \sqrt{\frac{2Q_h}{\alpha}}$ is the time for $Q(t)$ to reach Q_h from zero. We define the time for $R(t)$ to increase from μ (also denoted by BW, see Fig. 6) to R_{max} as

$$T_{max} \triangleq T_f + T_q + T_b = T_f + \sqrt{\frac{2Q_h}{\alpha}} + T_b = \tau + \sqrt{\frac{2Q_h}{\alpha}}. \quad (5)$$

Then, the maximum queue length is expressed as

$$Q_{max} = \int_0^{T_{max}} \alpha t dt + \int_0^{T_d} (R_{max} e^{-(1-\beta)\frac{t}{\Delta}} - \mu) dt \quad (6)$$

where T_d is the time for $R(t)$ to drop from R_{max} back to μ (i.e., BW, see Fig. 6), and is obtained, by letting $R(T_d) = \mu$, as

$$T_d = -\frac{\Delta}{(1-\beta)} \log \frac{\mu}{R_{max}}. \quad (7)$$

Then, the maximum queue length is obtained as

$$Q_{max} = \frac{\alpha}{2} T_{max}^2 + \alpha \frac{\Delta}{1-\beta} \left(T_{max} + \frac{\mu}{\alpha} \log \frac{\mu}{R_{max}} \right). \quad (8)$$

Let T_l be the duration for $Q(t)$ to decrease from Q_{max} to Q_l , then T_l can be determined by

$$Q_{max} - Q_l = \int_0^{T_l} \mu(1 - e^{-(1-\beta)\frac{t}{\Delta}}) dt \quad (9)$$

So, T_l is the non-negative real root of the non-linear equation

$$e^{-(1-\beta)\frac{T_l}{\Delta}} + \frac{1-\beta}{\Delta} T_l - \left(\frac{Q_{max} - Q_l}{\mu} \right) \left(\frac{1-\beta}{\Delta} \right) - 1 = 0. \quad (10)$$

The minimum rate is then given as $R_{min} = \mu e^{-(1-\beta)\frac{(T_l + T_f + T_b)}{\Delta}}$. We define the rate-control cycle as

$$T \triangleq T_q + T_d + T_l + 2\tau + T_r, \quad (11)$$

where $T_r = (\mu - R_{min})/\alpha^*$ is the time for $R(t)$ to grow from R_{min} to μ with the new α^* specified by Eq. (3). The average throughput, denoted by \bar{R} , can be obtained by

$$\bar{R} \triangleq \frac{1}{T} \int_{t_0}^{t_0+T} R(t) dt = \frac{1}{T} \left[\int_0^{T_{max}} (\mu + \alpha t) dt + \int_0^{T_e} R_{max} e^{-(1-\beta)\frac{t}{\Delta}} dt + \int_0^{T_r} (R_{min} + \alpha^* t) dt \right] \quad (12)$$

where $T_e = T_d + T_l + \tau$. Simplifying Eq. (12), we obtain

$$\bar{R} = \frac{1}{T} \left[\mu T_{max} + \frac{\alpha}{2} T_{max}^2 + R_{max} \frac{\Delta}{1-\beta} \cdot \left(1 - e^{-(1-\beta)\frac{T_e}{\Delta}} \right) + T_r R_{min} + \frac{\alpha^*}{2} T_r^2 \right]. \quad (13)$$

C. The Greatest Lower Bound for the Target Buffer Occupancy

How to choose the target buffer occupancy Q_{goal} is a practically important design problem associated with the α -control. Usually, as long as Q_{goal} can ensure the full bandwidth utilization, a small Q_{goal} is desired, because a large Q_{goal} may increase queuing delay and delay variations, affecting the network dynamics and stability. Using the analytical results derived in Section III-B, the theorem given below finds the greatest lower bound for Q_{goal} and its relationships with α , τ , and Q_h .

Theorem 1: Consider a connection flow-controlled by the proposed rate-control scheme described by Eqs. (1) and (2). If (i) the upper queue-length threshold $Q_h < \frac{1}{2}\xi < \infty$, (ii) its RTT $\tau > 0$, and (iii) the rate-gain parameter α is controlled by the α -control law defined in Eq. (3), then the following claims hold:

Claim 1: The greatest lower bound of $Q_{goal}(\alpha_n, \tau)$ under the α -control defined in Eq. (3) exists and is determined by:

$$\inf_{\tau > 0, \alpha_n > 0, n=1,2,\dots,\infty} \{Q_{goal}(\alpha_n, \tau)\} = 2Q_h; \quad (14)$$

Claim 2: The right-hand limit of $Q_{goal}(\alpha, \tau)$ at $\alpha = 0$ in the continuous-domain of α exists and is determined by:

$$\lim_{\alpha \downarrow 0} Q_{goal}(\alpha, \tau) = 2Q_h; \quad (15)$$

where all variables are the same as defined in Section III-A and Section III-B.

Proof: The proof is omitted, but available on-line in [17]. ■

Remarks on Theorem 1: Claim 1 derives the greatest lower bound of $Q_{goal}(\alpha, \tau)$ under the proposed α -control law, showing that Q_{goal} must be at least larger than $2Q_h$ for $\alpha > 0$ and $\tau > 0$. Claim 2 shows that α must approach 0 for $Q_{goal}(\alpha, \tau)$ to converge to its greatest lower bound $2Q_h$. Combining Claim 1 and Claim 2, we thus choose $Q_{goal} \gg 2Q_h$ as specified in Section II-A. In addition, Theorem 1 also provides the network designer with an explicit guidance on how to select the upper queue-length threshold Q_h for any desired target buffer occupancy Q_{goal} and the given buffer capacity C_{max} at routers. As shown in [17], the maximum queue length $Q_{max}(\alpha, \tau)$ increases as Q_h increases, and so does $Q_{goal}(\alpha, \tau)$. On the other hand, a too small Q_h is also undesirable because a too small Q_h may decrease the bandwidth utilization.

IV. PACKET-LOSS ANALYSIS

Since the buffer size at routers is always finite, in this section we focus on the case where packets are lost due to buffer overflow.

A. Packet-Loss Calculation

To quantitatively evaluate the loss-control performance of the proposed scheme, we introduce the following definition:

Definition 1: The **packet-loss rate**, denoted by γ , is the percentage of the lost packets among all the transmitted packets and the **link-transmission efficiency**, denoted by η , is the fraction of packets successfully transmitted (without retransmitting them) among all packets transmitted. Then γ and η in one rate-control cycle are expressed as:

$$\gamma \triangleq \frac{\rho}{T\bar{R}} \quad \text{and} \quad \eta \triangleq 1 - \gamma = 1 - \frac{\rho}{T\bar{R}} \quad (16)$$

where T is the rate-control cycle specified by Eq. (11), ρ is the number of lost packets during T , and \bar{R} is the average throughput determined by Eq. (13). ■

The link-transmission efficiency η is an important metric for flow and error control since it measures the percentage of link bandwidth used by successfully-transmitted packets. The following theorem gives an explicit formula to calculate the number ρ of packet losses from which both η and γ can be derived.

Theorem 2: If a connection with buffer capacity $Q_h < \xi < \infty$ is under the rate-control scheme described by the state equations (1)–(2) and the α -control law by Eq. (3), then the number, ρ , of lost packets during one rate-control cycle T is determined by:

$$\rho = \begin{cases} \frac{1}{2} \alpha \left(T_{max}^2 - t_\xi^2 \right) - \mu T_d + R_{max} \frac{\Delta}{1-\beta} \cdot \left[1 - e^{-\frac{1-\beta}{\Delta} T_d} \right]; & \text{if } t_\xi \leq T_{max} \\ \mu \left(t_\xi - T_{max} - T_d \right) + R_{max} \frac{\Delta}{1-\beta} \cdot \left[e^{-\frac{1-\beta}{\Delta} (t_\xi - T_{max})} - e^{-\frac{1-\beta}{\Delta} T_d} \right]; & \text{if } t_\xi > T_{max} \end{cases} \quad (17)$$

where all variables are the same as defined in Section III, except that $t_\xi = \sqrt{\frac{2\xi}{\alpha}}$ if $\xi \leq \frac{1}{2}\alpha T_{max}^2$ (i.e., $t_\xi = \sqrt{\frac{2\xi}{\alpha}} \leq T_{max}$ which is for the first part of Eq. (17)); else t_ξ is the non-negative real root of the following non-linear equation for the second part of Eq. (17):

$$\frac{1}{2} \alpha T_{max}^2 + R_{max} \frac{\Delta}{1-\beta} \left(1 - e^{-(1-\beta)\frac{t_\xi - T_{max}}{\Delta}} \right) - \mu(t_\xi - T_{max}) - \xi = 0, \quad \text{if } \xi > \frac{1}{2} \alpha T_{max}^2. \quad (18)$$

Proof: The proof is omitted, but available on-line in [17]. ■

B. Performance Evaluation of Loss Control

Consider the bottleneck with $\mu = 367$ packets/ms (155 Mbps), $\xi = 400$ packets; $Q_h = 50$ packets, and $q = 0.6$. Fig. 7 plots the number of lost packets, ρ , obtained from Eq. (17), against α for different RTTs τ 's. Note that ρ increases with α , and for a given α , ρ gets larger as τ increases. It is therefore necessary to apply α -control to reduce the packet losses due to the increase in the number and RTT of cross-traffic flows. Packet losses cause retransmissions, and thus affect link-transmission efficiency η . In Fig. 8, η is plotted against α for the same parameters. As illustrated in Fig. 8, $\eta = 1$ at the beginning, implying that there is no retransmission (loss) if α is controlled to be small enough under the α -control

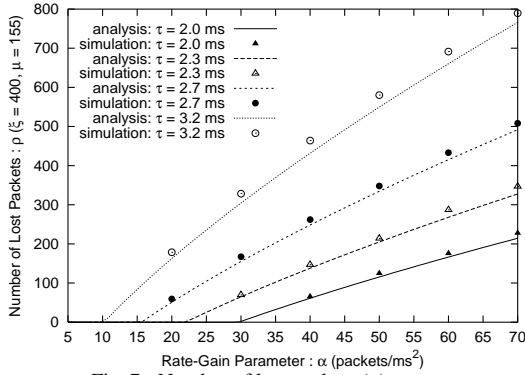


Fig. 7. Number of lost packets (ρ) vs. α .

for any given τ . As α increases, Fig. 8 shows that η is a decreasing function of α , and drops faster for larger τ 's. For instance, $\gamma = 1 - \eta \leq 2\%$ of packets need to be retransmitted if α is controlled to be smaller than 50 packets/ms² for $\tau = 2$ ms, but to keep $\eta \geq 98\%$ for $\tau = 3.2$ ms, α needs to be limited to no larger than 22 packets/ms². Using the NetSim [18], we also simulated packet losses and link-transmission efficiency, which agree well with the numerical results (see Figs. 7–8).

V. EFFICIENCY AND FAIRNESS OF α -CONTROL

Since $Q_{max}(\alpha)$ is a one-to-one correspondence function between Q_{max} and α as shown in Eq. (8), buffer-allocation control can be handled equivalently by α -allocation control. We introduce the following criteria to evaluate the α -control law for buffer management in terms of α -allocation.

Definition 2: Let vector $\alpha(k) = (\alpha_1(k), \dots, \alpha_n(k))$ be the rate gain parameters at time k for n connections sharing a common bottleneck characterized by $\alpha_{goal} = Q_{max}^{-1}(Q_{goal})$. The *efficiency* of α -allocation is measured by the distance between the superposed α -allocation, $\alpha_t(k) \triangleq \sum_{i=1}^n \alpha_i(k)$, and its target value α_{goal} . ■

Neither over-allocation $\alpha_t(k) > \alpha_{goal}$, nor under-allocation $\alpha_t(k) < \alpha_{goal}$ is desirable and efficient, as over-allocation may result in packet losses and under-allocation yields poor transient response, buffer utilization, and transmission throughput. The goal of α -control is to drive the total or aggregate α -allocation $\alpha_t(k)$ of $\alpha(k)$ to α_{goal} as close and as fast as possible from any initial state.

Definition 3: The *fairness* of α -allocation $\alpha(k) = (\alpha_1(k), \dots, \alpha_n(k))$ for n connections of the same priority sharing the common bottleneck at time k is measured by the *fairness index* $\phi(\alpha(k)) \triangleq \frac{[\sum_{i=1}^n \alpha_i(k)]^2}{n [\sum_{i=1}^n \alpha_i^2(k)]}$. ■

Notice that $\frac{1}{n} \leq \phi(\alpha(k)) \leq 1$. $\phi(\alpha(k)) = 1$ if $\alpha_i(k) = \alpha_j(k)$, $\forall i \neq j$, corresponding to the “best” fairness. $\phi(\alpha(k)) = \frac{1}{n}$ if α is allocated to only one of n active connections. This corresponds to the “worst” fairness and $\phi(\alpha(k)) \rightarrow 0$ as $n \rightarrow \infty$. So, the fairness index $\phi(\alpha(k))$ should converge as close to 1 as possible as $k \rightarrow \infty$.

The α -control is a negative feedback control over the rate-gain parameter, and computes $\alpha(k+1)$ based upon the current value $\alpha(k)$ and the feedback $BCN(k-1, k)$. Thus, $\alpha(k+1)$ can be expressed by the control function as $\alpha(k+1) = g(\alpha(k), BCN(k-1, k))$. For implementation simplicity, we only focus on a linear control function $g(\cdot, \cdot)$ by which we mean that $\alpha(k+1) = p + q\alpha(k)$, where coefficients p and q are determined by feedback information $BCN(k-1, k)$. The theorem given below describes the feasibility and optimality of the linear α -control,

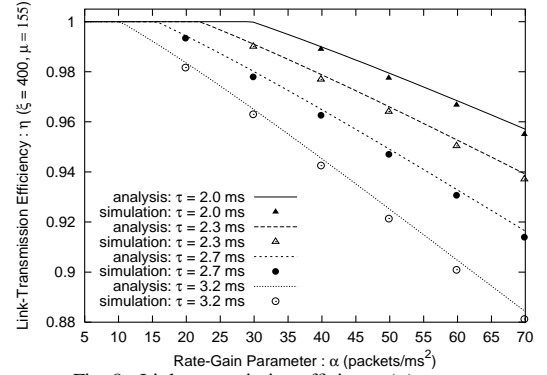


Fig. 8. Link-transmission efficiency (η) vs. α .

which ensures the convergence of α -control to the efficiency and fairness of buffer allocation as defined by Definitions 2 and 3.

Theorem 3: Suppose n connections sharing a common bottleneck are synchronously flow-controlled by the proposed α -control. Then, (1) in *transient* state, the α -control law is feasible and optimal linear control in terms of convergence to the efficiency and fairness of buffer allocation; (2) in *equilibrium* state, the α -control law is feasible and optimal linear control in terms of maintaining the efficiency and fairness of buffer allocation.

Proof: The proof is omitted, but available on-line in [17]. ■

Remarks on Theorem 3: Theorem 3 is an extension from bandwidth control [19] to buffer control, but differs from [19] as follows. Unlike the bandwidth control exerted at the control-packet transmission rate, the α -control is exercised once every rate-control cycle. As a result, the α -control distinguishes transient state from equilibrium state, and applies different control algorithms in these two states, which makes $\alpha_t(k)$ not only monotonically converge to, but also lock within, a small neighborhood of its target α_{goal} . Since the total allocation $\alpha_t(k)$, or the number of connections, keeps on going up and down due to cross-traffic variations in real-world networks (or equivalently, the target α -allocation for each connection is “moving” up and down), it suffices to ensure convergence to fairness/efficiency in transient state and maintain the achieved fairness/efficiency in equilibrium state. Using the analytical results of Section III, we conducted the vector-space analysis through two examples in a 2-dimensional vector space to show the convergence of α -allocation under the α -control in terms of α -allocation efficiency and fairness. The vector-space analysis and the two examples are omitted for lack of space, but are available on-line in [17].

VI. PERFORMANCE EVALUATION

Using the NetSim [18], we have built a simulator to implement the proposed flow and error control scheme. As shown in Fig. 9, the simulated network carries the traffic of three connections C_1 , C_2 , and C_3 which share a bottleneck link between Router-1 and Router-2. C_i 's data packets are sent from sender S_i to its receiver R_i . The simulation parameters are bottleneck bandwidth $\mu = 367$ packets/ms, RTTs $\tau = 2$ ms, and Router-1's buffer size $\xi = 800$ packets (for $Q_{max} < \xi$), or $\xi = 400$ (for $Q_{max} > \xi$). We set $Q_h = 50$ packets, $Q_{goal} = 300$ packets, $R_0 = 30$ packets/ms, $\Delta =$

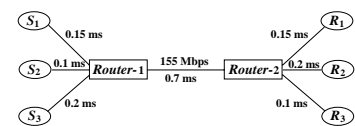
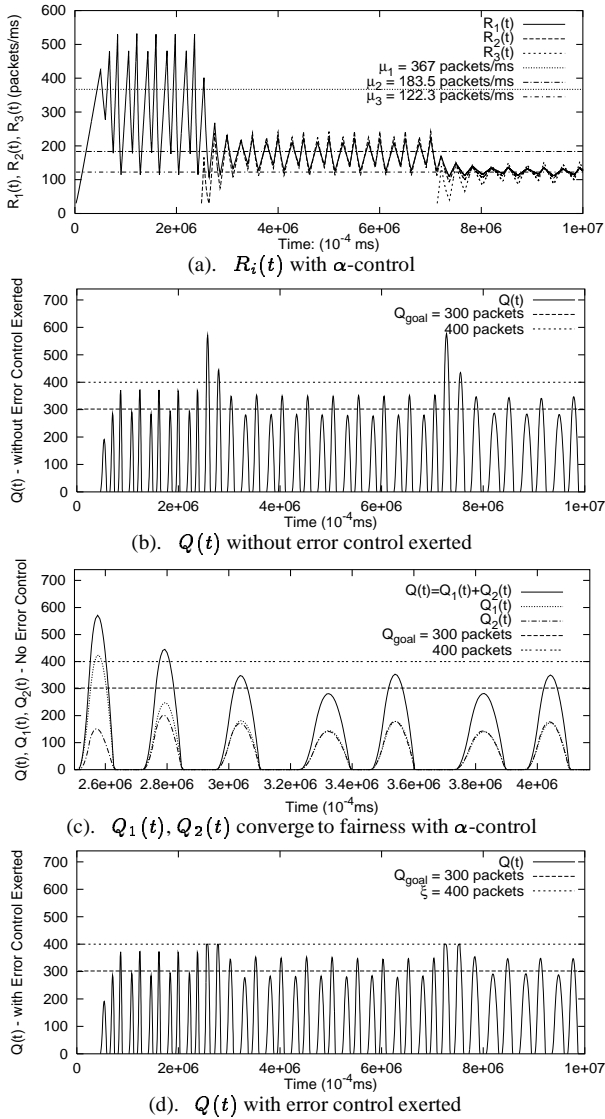


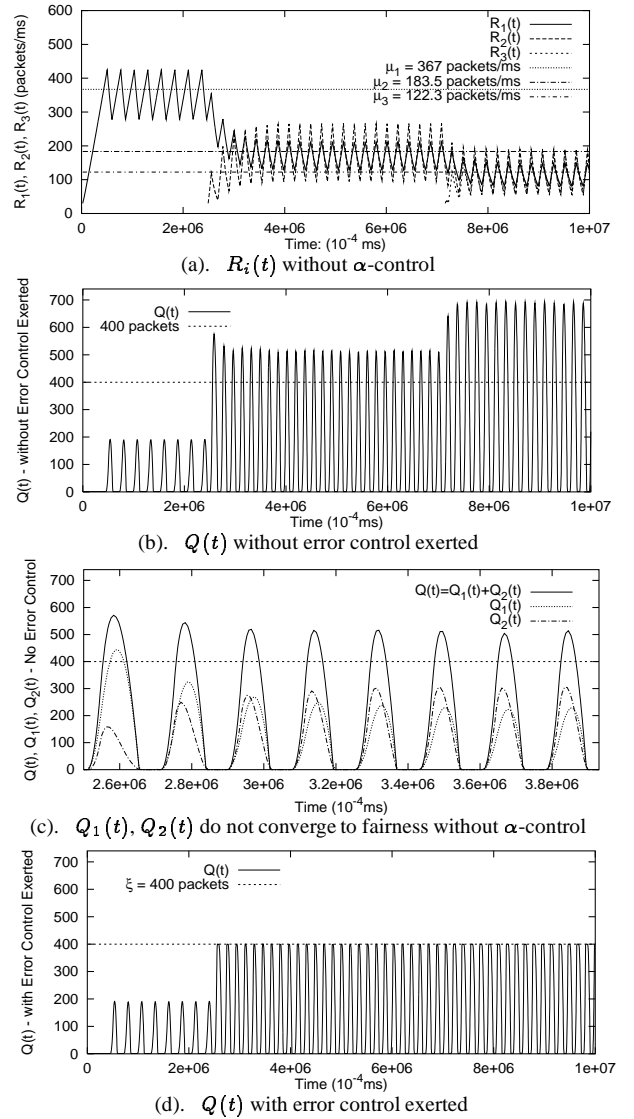
Fig. 9. The simulation model.

Fig. 10. Dynamics of $R_i(t)$ and $Q(t)$ with α -control.

0.4 ms, $q = 0.6$, $p = 2.9$, $\alpha_0 = 8.7, 14.7$, and 17.7 packets/ms² for C_1, C_2 , and C_3 , respectively. C_1 starts transmitting at $t_0=0$, C_2 at $t_1=245$ ms, and C_3 at $t_2=710$ ms such that the number of active connections, denoted by n , increases from 1 to 3. Consequently, t_1 and t_2 partition the entire simulation time 1000 ms into 3 periods: $T_1 = [0, 245]$ with $n = 1$, $T_2 = [245, 710]$ with $n = 2$, and $T_3 = [710, 1000]$ with $n = 3$. We simulated the network with and without the α -control. The simulated source rate $R_i(t)$ ($i = 1, 2, 3$) and the bottleneck queue length $Q(t)$ are plotted in Figs. 10(a)–(d) for the case with α -control, and in Figs. 11(a)–(d) for the case without α -control. We compare the two schemes with and without α -control in the following two cases.

CASE I. $Q_{max} < \xi = 800$ without error control.

(1) During T_1 ($n = 1$). With α -control, Fig. 10(a) shows that $R_1(t)$ converges to $\mu_1=367$ packets/ms since only C_1 is active and it grabs all available bandwidth. Figs. 10(a)–(b) show that experiencing one transient cycle due to $Q(t)$'s maximum $Q_{max} = 190 < Q_{goal}$ at the beginning, the rate-gain parameter (α_1) of $R_1(t)$ is linearly increased by α -control such that Q_{max} converges to and stays within Q_{goal} 's neighborhood (verifying Theorem 3). With sufficient buffer size, the increased α_1 enhances responsiveness in grabbing newly-available bandwidth, if any. In contrast, without α -control, Fig. 11(a) shows that $R_1(t)$ also converges to μ_1

Fig. 11. Dynamics of $R_i(t)$ and $Q(t)$ without α -control.

$=367$, but Q_{max} (see Fig. 11(b)) is always 190 during T_1 , utilizing less than 25% of buffer size without enhancing the responsiveness. **(2) During T_2 ($n = 2$).** With the α -control, Fig. 10(a) shows $R_1(t)$ and $R_2(t)$ experience two transient cycles during which $R_1(t)$ yields bandwidth $\frac{1}{2}\mu_1 = \mu_2$ to $R_2(t)$. Fig. 10(b) shows that a large queue build-up $Q_{max} = 590$ starting at $t_1 = 245$. This is expected because n increases from 1 to 2, and thus, the new superposed rate-gain is in effect equal to the sum of C_1 and C_2 's rate-gains. Driven by α -control, both $R_1(t)$ and $R_2(t)$ reduce their rate-gain parameters such that Q_{max} converges to Q_{goal} 's neighborhood within 2 transient cycles. Moreover, convergence to the buffer-occupancy fairness (verifying Theorem 3) under the α -control is verified by the convergence of $Q_1(t)$ and $Q_2(t)$ to each other during two transient cycles (note $Q(t) = Q_1(t) + Q_2(t)$). (See Fig. 10(c), a zoom-in of Fig. 10(b).) By contrast, without α -control, Fig. 11(b) shows that Q_{max} shoots up to 590 packets and remains above 520 packets even after entering the equilibrium. Moreover, Fig. 11(c), a zoom-in of Fig. 11(b), shows that buffer occupancy is not fair because $Q_1(t)$'s maximum, which is larger than $Q_2(t)$'s maximum during transient state, becomes smaller than $Q_2(t)$'s maximum after entering the equilibrium as $\alpha_1 (= 8.7)$ is smaller than $\alpha_2 (= 14.7)$. **(3) During T_3 ($n = 3$).** At $t_2 = 710$ ms, C_3 joins in, and thus n increases from 2 to 3. With α -control, Fig. 10(a) shows that after

C_i	T_{trans}	α -Control-Based Protocols								Non- α -Control-Based Protocols							
		N_{trans}	N_{recv}	$N_{retrans}$	γ	η	\bar{R}_{trans}	\bar{R}_{recv}	N_{trans}	N_{recv}	$N_{retrans}$	γ	η	\bar{R}_{trans}	\bar{R}_{recv}		
C_1	1000	175559	175333	226	1.289e-3	99.871 %	175.559	175.333	163171	160877	2294	1.405e-2	98.595 %	163.171	160.877		
C_2	755	102642	102489	153	1.491e-3	99.851 %	135.950	135.747	96142	92373	3769	3.920e-2	96.080 %	127.340	122.348		
C_3	290	27097	27048	49	1.808e-3	99.819 %	93.438	93.269	25485	23748	1737	6.816e-2	93.184 %	87.879	81.890		

TABLE I
ERROR AND FLOW CONTROL PERFORMANCE COMPARISON BETWEEN WITH AND WITHOUT α -CONTROL.

2 transient cycles, C_1 and C_2 both yield some bandwidth to C_3 such that they each take one third of the bandwidth μ_3 . Again, Fig. 10(b) shows that Q_{max} increases dramatically up to 585 at t_2 as a result of C_3 's joining in. With the α -control, Q_{max} quickly returns to Q_{goal} 's neighborhood within 2 transient cycles. In contrast, without α -control, after Q_{max} jumped up to 700 packets (see Fig. 11(b)), it never drops from 700 packets throughout T_3 .

CASE II. $Q_{max} > \xi = 400$: error control exerted.

The other parameters remain the same. With α -control, Fig. 10(d) shows that packets are dropped only during the short transient (only two cycles) state starting at t_1 and t_2 , where $Q(t) = \xi$. However, as soon as the α -flow-controlled system settles down to an equilibrium state, the bottleneck stops dropping packets, because Q_{max} already converged to the neighborhood of Q_{goal} upper-bounded by ξ . Since no packets are dropped during the designated equilibrium (thus no retransmission), we call the optimal equilibrium state specified by the α -control the *retransmission-less equilibrium state*. In contrast, Fig. 11(d) shows that, without α -control, packets are dropped not only during the transient state (as n increases at t_1 and t_2), but also after the system enters the equilibrium state.

In case of packet loss, our proposed error-control mechanism kicks in and each lost packet is retransmitted (more than once if it is lost again) until it is successfully received. TABLE I shows the simulated error and flow control data from C_1 , C_2 , C_3 , with and without α -control. TABLE I shows that the number of retransmissions, denoted by $N_{retrans}$ is verified by the difference of N_{trans} (the number of both transmitted and retransmitted packets) minus N_{recv} (the number of correctly received packets) during the transmission period T_{trans} . The corresponding packet-loss rate γ and link-transmission efficiency η are calculated by Definition 1.

With α -control, the number of retransmissions $N_{retrans}$ and loss rate γ are very small and the corresponding η is as high as 99.8% for all C_1 , C_3 , and C_3 . This is because α -control always drives the flow-controlled system to settle down to a lossless equilibrium state, hence guaranteeing retransmission-less transfer. By contrast, without α -control, $N_{retrans}$ and γ are 10 to 35 times as large as those of the α -control case for C_1 , C_3 , and C_3 . Consequently, the η is much lower than that under α -control. For instance, C_3 's $\eta = 93.184\%$, i.e., about 7% bandwidth is wasted for retransmissions. TABLE I also shows that the system with α -control outperforms that without α -control on average throughput \bar{R}_{trans} (sending end) and \bar{R}_{recv} (receiving end — *goodput*). The difference ($\bar{R}_{trans} - \bar{R}_{recv}$) is also found much smaller with α -control than that without α -control due to much fewer packet drops (and hence much fewer retransmissions) under α -control.³

VII. CONCLUSION

We proposed and analyzed an efficient flow and error control scheme for high-throughput transport protocols. It is built on the α -control, a second-order rate control, as well as on a separate, new sliding-window scheme for error control. The α -control minimizes

³As a result, to transfer a file of the same size, the transmission time with α -control is much shorter than that without α -control, as shown by simulations in [17].

packet losses and retransmissions by adjusting the rate-gain parameter to the variations in the number and RTTs of cross-traffic flows that share the bottleneck. Using NACKs and selective retransmissions, our error-control scheme recovers packet losses, if any. Applying the fluid analysis, we modeled the proposed scheme, and derived the greatest lower bound for the target buffer occupancy and the other various performance measures. The α -control is analytically shown to be able to drive the flow-control system from any initial state to an optimal equilibrium state in which retransmission-less transfer is guaranteed. Our extensive simulation experiments confirmed the analytical findings, and demonstrated the superiority of the α -control to other non- α -control schemes in terms of loss/retransmission control, link-transmission efficiency, data-transmission time, throughput, and buffer-occupancy fairness.

ACKNOWLEDGMENT

We would like to thank the anonymous reviewers for their helpful suggestions and comments on this paper.

REFERENCES

- [1] V. Jacobson, "Congestion avoidance and control," in *ACM SIGCOMM*, 1988.
- [2] D. D. Clark, M. Lambert, and L. Zhang, "NETBLT: A high throughput transport protocol," in *ACM SIGCOMM*, pp. 353–359, 1987.
- [3] J. Padhye, J. Kurose, D. Towsley, and R. Koodli, "A model based TCP-friendly rate control protocol," in *NOSSDAV*, 1999.
- [4] L. Zhang, "Why TCP timers don't work well," in *ACM SIGCOMM*, 1986.
- [5] X. Zhang, K. G. Shin, D. Saha, and D. Kandlur, "Scalable flow control for multicast ABR services in ATM networks." To appear in *IEEE INFOCOM '99*, and available via URL http://www.eecs.umich.edu/~xizhang/papers/ToN_mcast.pdf.
- [6] S. Floyd, "TCP and explicit congestion notification," *ACM SIGCOMM Computer Communication Review*, vol. 24, no. 5, pp. 10–23, October 1994.
- [7] F. Kelly, "Models for a self-managed Internet," Available *on-line via*: URL <http://www.statslab.cam.ac.uk/~frank/smi.html>, August 2000.
- [8] W. R. Stevens, *TCP/IP Illustrated, Vol. 1*, Addison-Wesley, 1994.
- [9] L. S. Brakmo and L. L. Peterson, "TCP Vegas: End to end congestion avoidance on a global Internet," *IEEE Journal on Selected Areas in Communications*, vol. 13, no. 8, pp. 1465–1480, October 1995.
- [10] S. H. Low, L. Peterson, and L. Wang, "Understanding TCP Vegas: A duality model," in *Proc. of ACM SIGMETRICS*, 2001.
- [11] S. Keshav and S. Morgan, "SMART retransmission: performance with overload and random losses," in *Proc. of IEEE INFOCOM*, April 1997.
- [12] B. T. Doshi, P. K. Johri, A. N. Neeravali, and K. K. Sabnani, "Error and flow control performance of a high speed protocol," *IEEE Trans. on Communications*, vol. 41, no. 5, pp. 707–720, May 1993.
- [13] J. Bolot and A. Shankar, "Dynamical behavior of rate-based flow control mechanism," *ACM SIGCOMM Computer Communication Review*, vol. 20, no. 4, pp. 35–49, April 1990.
- [14] N. Yin and M. G. Hluchyj, "On closed-loop rate control for ATM cell relay networks," in *IEEE INFOCOM*, pp. 99–109, 1994.
- [15] S. Mascolo, "Smith's principle for congestion control in high-speed data networks," *IEEE Trans. on Automatic Control*, vol. 45, pp. 358–364, Feb. 2000.
- [16] S. Bohacek, J. P. Hespanha, J. Lee, and K. Obraczka, "A hybrid systems framework for TCP congestion control: A theoretical model and its simulation-based validation," *Hybrid Systems: Computation and Control*, pp. 291–304, Mar. 2001.
- [17] X. Zhang and K. G. Shin, "Second-order rate-based flow control with decoupled error control for high-throughput transport protocols," *Full paper version*: URL <http://www.eecs.umich.edu/~xizhang/papers/rw.pdf>, July 2001.
- [18] A. Heybey, *The Network Simulator*, Lab. for Compt. Scie., MIT, Oct. 1990.
- [19] D. Chiu and R. Jain, "Analysis of the increase and decrease algorithms for congestion avoidance in computer networks," *Computer Networks and ISDN Systems*, pp. 1–14, 1989.