Second-Order Rate-Based Flow Control with Decoupled Error Control for High-Throughput 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, which (i) decouples flow control from error control, and (ii) uses a second-order rate control, called α-control, for flow control and a new sliding-window scheme for error control. The α-control minimizes the need for packet retransmissions by adjusting the rate-gain parameter in response to the variations of cross-traffic flows and their round-trip delays (RTDs). Using selective retransmission, the sliding-window scheme guarantees lossless transmission. Separation of flow and error control simplifies both components and enhances the throughput since the source rate control is independent of the dynamics of the error-control window. Applying the α-control, the proposed scheme can drive the flow-controlled system to a retransmission-free equilibrium state. Using the fluid analysis, we model the packet-loss behavior and derive the closed-form expressions for packet losses, loss rate, and the 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 vector-space analysis and simulation results that verify the analytical results, and demonstrate the superiority of the proposed scheme to others in dealing with cross-traffic and RTDs variations, controlling packet losses/retransmissions, and achieving buffer-use fairness and a high throughput.**

Index Terms—**High-throughput transport protocol, decoupled flow and error control, congestion and loss recovery.**

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 reliable transmission. 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 over high-bandwidth channels is often an order-of-magnitude lower than the network capacity. Throughput is often limited by the underlying transport protocol, particularly its flow and error control mechanisms. It is difficult to achieve both high throughput and reliable data transmission across long-delay, large-bandwidth, and unreliable network paths. The network unreliability, delay, and unpredictable network 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 in 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 the data-sending rate in response to network congestion. The window-based scheme is cost-effective as it does not require a fine-grain rate-control timer, and a window automatically limits the damage a source can inflict on the network. However, the window-based scheme also introduces its own problems [2]. First, it only determines the amount of data to be sent, but does not specify the speed of packet transmission within the flow-control window. Consequently, the window-based scheme cannot make per-connection bandwidth guarantees for transmitting continuous media (CM) data such as audio and video [3]. Moreover, unregulated data rates of multiple connections sharing the same bottleneck link can easily generate a large instantaneous aggregate data rate at the bottleneck, congesting the network.

The second problem with the window-based scheme is that its flow-control mechanism is traditionally coupled with the error-control mechanism, since the flow-control window can be used as the error-control window as well. This coupling is often problematic as it may create protocol design conflicts. For instance, while a large window is desired for high throughput, retransmission needs a small window to minimize unnecessary retransmissions for the Go-back-N scheme or to reduce the receiver buffer for re-ordering lost packets in the Selective Retransmission scheme. Moreover, mixing flow and error control in one mechanism makes flow control vulnerable to packet losses and delays since packet losses and retransmissions cause the decrease of source transmission rate.

Third, the performance of the window-based scheme is RTD-dependent. To continue packet transmission while waiting for the receiver's acknowledgment, the window size must be larger for longer RTD paths, so that there are always data packets ready to be transmitted. But how large the window size should be sufficient? Theoretically, there does not exist any upper bound that is absolutely sufficient since it is proportional to RTD $\times$ an unpredictable number of errors [2]. Unfortunately, RTD is not constant, but varies randomly with time, which makes selection of a proper window size even more complicated. In addition, a very large window size for longer RTD paths can in effect eliminate the window's flow-control function, and thus can easily congest the network and overflow bottleneck buffers.

Finally, the window-based scheme works poorly with a retransmission timer. When packets are lost, most reliable transport pro-

The work reported in this paper was supported in part by the U.S. Office of Naval Research under Grant N00014-99-1-0465.

tocols use timers to trigger their retransmissions. A lost packet prevents an acknowledgment from the receiver, stops advancing the flow-control window, and thus tends to shut down the transmission window. So, a longer timer is more likely 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, thus wasting bandwidth. But choosing a proper timer value is a daunting task as pointed out in [4]. Obviously, the timer value should be determined as a function of RTD. But again, RTD varies randomly and measuring RTD is difficult in the presence of packet losses.

To overcome some of the aforementioned problems with the window-based flow-control protocol, the authors of [2] proposed a rate-based flow-control transport protocol NETBLT [2]. Differing from TCP, NETBLT employs the rate-based 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 transmitted 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 network bandwidth 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 overflow or underflow at the bottleneck. More importantly, as analyzed in [5], the AIMD rate control itself cannot upper-bound the maximum queue length at the bottleneck since the queue length is a function of the superposition of the rate-gain parameters (i.e., rate ramp-up speed) of all traffic flowing through the bottleneck and their RTDs. The unbounded bottleneck queue length can cause excessive packet losses, and thus costly retransmissions. Bottleneck queue control is difficult because the number of active cross-traffic flows through the bottleneck and their RTDs are unknown *a priori* to the data source and both vary randomly with time.

In this paper, we propose a second-order rate-control, called α-control, scheme to cope with the variations of cross-traffic flows through the bottleneck and their RTDs. 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 a high throughput is achieved. Unlike TCP using an implicit congestion signal for congestion control, α-control employs a mechanism, similar to Explicit Congestion Notification (ECN) [6] 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 as used in TCP-Tahoe [1], or three duplicate ACKs as used in TCP-Reno [7], to infer network congestion. As a result, the detection of an incipient congestion based on the ECN-like scheme can prevent unnecessary packet losses and retransmissions caused by the TCP flow-control scheme itself [8]. In addition, the proposed scheme uses a new sliding-window scheme for error control, but decouples it from the rate-based flow control. Consequently, 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. We also use periodic exchange of state messages [9] between the transmitter and the receiver to make the flow and error control performance virtually independent of RTD. The proposed scheme employs selective retransmission to conserve bandwidth.

Using the fluid analysis, we model the packet-loss behavior and derive the closed-form expressions for packet losses, loss rate, and link-transmission efficiency. The analytical results are applied to evaluate the loss-control performance of the proposed scheme, and justifies the necessity and feasibility of the α-control. The analysis shows that the α-control can drive the flow-controlled system to an optimal equilibrium state where the *retransmission-free* is guaranteed. We prove that the α-control is feasible and optimal linear control in terms of efficiency and fairness. The dynamic performance of the proposed scheme is evaluated quantitatively by both the analysis and simulations, and the simulation results verify the analytical results. The simulations experiments also demonstrate the superiority of the proposed scheme to the other schemes in dealing with the variations of cross-traffic flows at the bottleneck and their RTDs, controlling losses/retransmissions, achieving fairness in both buffer and bandwidth occupancies, and increasing average throughput.

The paper is organized as follows. Section II describes the proposed scheme, and Section III establishes the flow-control system model. 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 concurrent 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, and the receiver replies to it by returning a feedback control packet to the source. The inter-control packet interval is typically a fraction of RTD. 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 when returning it to the source. When the returned control packet arrives at the source, the control information is split into two parts: the flow-control information contained in ECN is fed back to the rate controller and the error-control information contained in ACK/NACK is forwarded to the error controller. Functionwise, the proposed scheme consists of two decoupled components: flow-control and error-control mechanisms.

A. The Flow-Control Mechanism

The flow control is to dynamically adapt user demand to currently available network resources. The network resources consists of two parts: bandwidth capacity and buffer capacity. As discussed in [10], the traditional AIMD rate control, which only applies direct increasing/decreasing control (first-order rate control) over source rate $R(t)$, is not effective enough to have the maximum queue length Q_{max} upper-bounded by the maximum buffer capacity C_{max}. This is because the first-order rate control can only make $R(t)$ fluctuate around the designated bandwidth, but can-

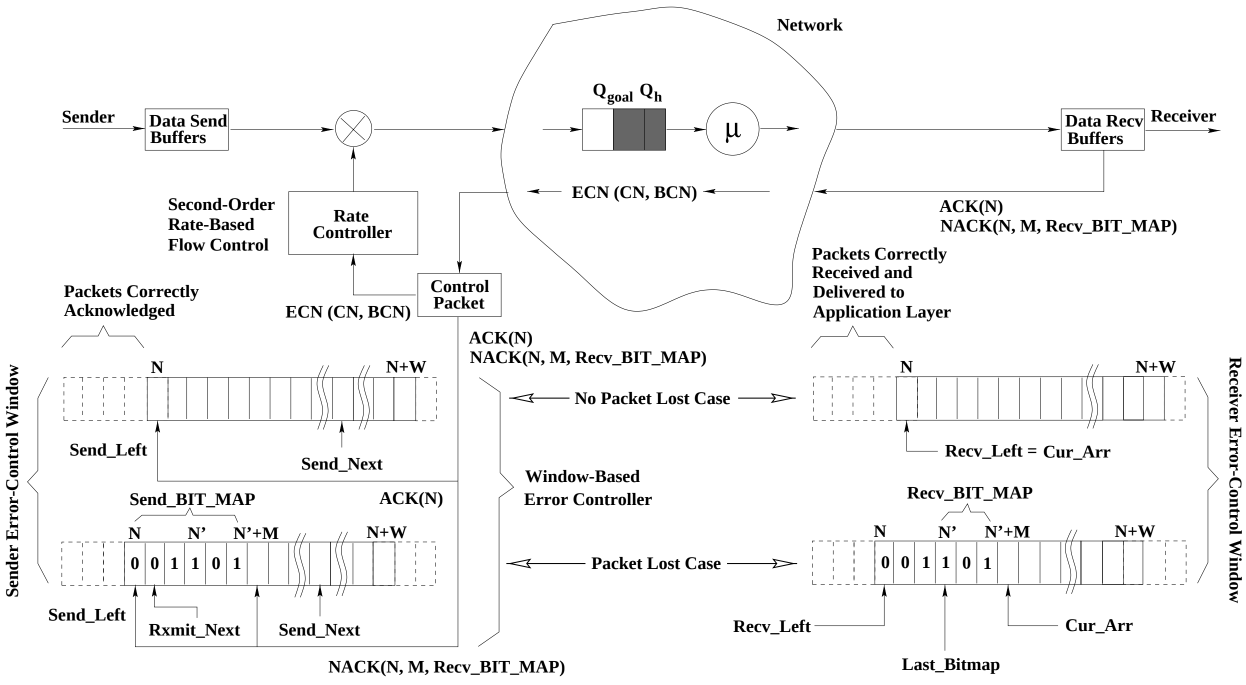


Fig. 1. The proposed flow and error control scheme.

not adjust the rate-fluctuation amplitude that determines Q_{max}. Consequently, the first-order rate control only exercises the control over bandwidth, but leaves bottleneck buffers un-controlled. In [10] and [5], Q_{max} is analytically shown to increase with both the rate-gain parameter and the connection's RTD. In [5], we developed the second-order rate control, called α-control, to deal with the RTD variation due to the bottleneck drift in a multicast-communication tree.

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

To distinguish the two classes of network-resource control, we define the following two types of congestions:

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 the target buffer occupancy Q_{goal}, where $Q_h < Q_{goal} < C_{max}$ 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, our congestion-detection employs 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, but the buffer-congestion detection (BCN-bit) differs from ECN since it provides one more dimension to control the dynamics of flow-controlled system.

Fig. 2 shows a pseudocode for 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 a rate-decrease phase to a rate-increase phase (i.e., when LCN (Local CN) is equal to 1, and the CN bit in received control packet is 0), then it is the time to exercise the buffer-congestion control (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 variations: (i) if BCN is 1, RIR is decreased multiplicatively by a factor of GDP (Gain-Decrease Parameter) $(0 < GDP < 1)$; (ii) if both $LBCN$ and BCN are 0, RIR is increased additively by a step of size GIP (Gain-Increase Parameter) > 0; (iii) if $LBCN$ = 1 and BCN = 0, RIR is increased multiplicatively by the same factor of GDP. For all of these three cases, the rate-decrease parameter RDP (Rate-Decrease Parameter) is adjusted according to the estimated bottleneck bandwidth BW_EST. Then, the local NMQ bit is marked and the received BCN bit is saved in $LBCN$ for the next α-control cycle. The source always exercises the (first-order) rate control whenever a control packet is received. Using the same, or updated, rate-gain parameter RIR and RDP, the source regulates its rate R based on AIMD algorithm according to the feedback

```
On receipt of Control Packet:
[1] Flow Control:
  if ($LCN = 1 \wedge CN = 0$) {! Buffer congestion control condition
    if ($BCN = 1$) {$RIR := GDP \times RIR$}; ! Dec RIR multiplicatively
    elseif ($BCN = 0 \wedge LBCN = 0$)
      {$RIR := GIP + RIR$}; ! Increase RIR additively
    elseif ($BCN = 0 \wedge LBCN = 1$)
      {$RIR := RIR/GDP$}; ! BCN toggles around target
    $RDP := e^{-RIR/BW\_BST}$; ! RDP updating
    $LNMQ := 1$ }; ! Start a new measurement cycle
  if ($CN = 0$) {$R := R + RIR$}; ! Increase source rate additively
  else {$R := R \times RDP$}; ! Decrease source rate multiplicatively
  $LCN := CN$;   $LBCN := BCN$; ! Save CN, BCN
[2] Error Control:
  if ACK($N$) received { ! Positive ack received
    $Send\_Left := N$; Discard packets with $pkt\_seqn < N$;};
  if NACK($N$, $M$, $Recv\_BIT\_MAP$) received { ! NACK received
    $Send\_Left := N$; Discard packets with $pkt\_seqn < N$;
    $Send\_M$ := $Send\_M$ + $M$; ! Update sender's bitmap length
    $Send\_BIT\_MAP \overset{cat}{\Longleftarrow} Recv\_BIT\_MAP$}!Concatenate bitmap vectors
```

Fig. 2. Pseudocode for sending end protocol.

```
On receipt of Data Packet $P(k, CN)$:
[1] Flow Control:
  $Local\_CN := CN \vee Local\_CN$ ! Bandwidth congestion notification
[2] Error Control:
  if ($Cur\_Arr = Recv\_Left \wedge k = Cur\_Arr$) {
    $Cur\_Arr := Cur\_Arr + 1$;! Updating next expecting seq. number
    $Recv\_Left := Cur\_Arr$; ! Update left-edge sequence number
    $Last\_Bimap := Cur\_Arr$ }; ! Update starting pointer position
  if ( ($Cur\_Arr > Recv\_Left \wedge k = Cur\_Arr$)
    $\vee$ ($Cur\_Arr = Recv\_Left \wedge k > Cur\_Arr$)
    $\vee$ ($Cur\_Arr > Recv\_Left \wedge k > Cur\_Arr$) ) {
    $Recv\_BIT\_MAP[k - Last\_Bitmap] := 1$; ! Set new bitmap bit
    $Cur\_Arr := k + 1$; }! Update next expecting sequence number
  if ($Cur\_Arr > Recv\_Left \wedge k < Last\_Bitmap$) {
    Received retransmission-packet processing;
    Deliver all packets in sequence to user; ! Sequentially deliver
    Update $Recv\_Left$}; ! Update left-edge pointer of error window
On receipt of Control Packet:
[1] Flow Control:
  $CN := CN \vee Local\_CN$; ! Bandwidth congestion notification
[2] Error Control:
  $N := Recv\_Left$; ! Correctly acknowledged packet sequence number
  if ($Recv\_Left = Cur\_Arr$) { ! No lost packets
    $send\_ACK$:= TRUE}; ! Need to send ACK message
  if ($Recv\_Left < Cur\_Arr$) { ! Lost packets not recovered yet
    $M := Cur\_Arr - Last\_Bitmap$; ! Length of receiver bitmap vector
    $send\_ACK$:= FALSE}; ! Need to send NACK message
  if ($send\_ACK$ = TRUE) {
    send control packet (ECN($CN$, $BCN$), ACK($N$));}! Send ACK
  else{send control packet (ECN($CN$, $BCN$),
                              NACK($N$, $M$, $Recv\_BIT\_MAP$));! Send NACK
    $Recv\_BIT\_MAP := \underline{0}$;} ! Reset the current cycle's receiver bitmap
  $Last\_Bitmap := Cur\_Arr$; ! Update receiver bitmap starting position
```

Fig. 3. Pseudocode for receiving end protocol.

CN bit set by the receiver or IP routers.

B. The Error-Control Mechanism

The proposed scheme realizes both Negative ACKnowledgement (NACK) error detection and Selective Retransmission recovery. Using the NACK error detection, a receiver sends a NACK when it detects a gap in the sequence of packets it received. Combining with selective retransmission, a NACK contains a range of the sequence numbers of packets that have been lost and will be selectively retransmitted. The NACK mechanism and periodic control-packet feedback avoid the usually-difficult timer design and minimize the dependency of the error and flow-control performance on RTDs.

At the sender, all data packets are sequence-numbered, and put in the sender's buffer before being sent into the network, as shown in Fig. 1. A sent 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 the corresponding packet has (not) been correctly acknowledged within the retransmission error-control window at the transmitter.

Decoupling the error control from the flow control, we can set the error-control window size W (see Fig. 1) as large as the buffer resource permits to avoid any decrease of source transmission rate due to packet losses. However, the required error-control window (buffer) size W at both sender and receiver for our proposed scheme is in fact quite limited because we give the lost packet (pointed by $Rxmit\_Next$) a higher priority to be retransmitted than the packet (pointed by $Send\_Next$) to be transmitted for the first time. In addition, our proposed scheme uses the periodic control-packet feedback with a period smaller than RTD, which further reduces the required error-control window (buffer) size W at both sending and receiving ends.

As shown in Fig. 1, the receiver maintains three receiver-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 correctly received in the last arrival; (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 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 by bit 1 (0) during the current error-control cycle. The length of $Recv\_BIT\_MAP$ is determined by $M := Cur\_Arr - Last\_Bitmap$.

A pseudocode for 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 (the $Recv\_Left$ at the receiver). Then, all packets with sequence numbers $< N$ are removed from the sender buffer because they have been correctly acknowledged. 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 for the receiver error-control algorithm consists of two parts: 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 below 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 sequence 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

```
On receipt of a DATA Packet $P(CN)$:
    if (output link $\neq$ busy) { send $P(CN \vee Local\_CN)$ } ! Output packet
    elseif ($size\_of$ ($data\_que$) = $\xi$) { drop $P(CN)$;} ! Packet loss occurs
    else { enque($data\_que, P(CN)$); } ! Buffer this packet
    if ($size\_of(data\_que) > Q_h$) {$Local\_CN$ := 1;}! Bandwidth congest
    elseif ($size\_of(data\_que) < Q_l$) {$Local\_CN$ := 0;} ! No BW congest
    if ($size\_of(data\_que) > Q_{max}$) {$Q_{max} := size\_of(data\_que)$;}
    if ($Q_{max} > Q_{goal}$) {$Local\_BCN$ := 1;} ! Buffer congestion
    else {$Local\_BCN$ := 0;} ! No buffer-congestion
On receipt of a feedback Control Packet $P(CN, BCN)$:
    $CN$:= $Local\_CN \vee CN$; ! CN processing
    $BCN$:= $Local\_BCN \vee BCN$; ! BCN processing
    send Control Packet $P(CN, BCN)$ to up-stream node;
On receipt of a forward Control Packet $P(NMQ)$:
    if ($NMQ$=1) {$Local\_BCN$ := 0; $Q_{max}$ := 0;} ! New cycle starts
    send control packet $P(NMQ)$ to down-stream node;
```

Fig. 4. Pseudocode for IP routers.

lost but not recovered 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 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 sequence 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 the packet being lost multiple times. We developed an efficient false-alarm-free algorithm to deal with multiple losses of a packet, 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 an ACK($Recv\_Left$) to the source. (2) If $Recv\_Left < Cur\_Arr$, there are still unrecovered losses. So, it returns a 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 $\underline{0}$. Whenever receiving a control packet, $Last\_Bitmap$ is updated by Cur_Arr.

C. Flow and Error Control Algorithms at IP Routers

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

When a data packet received: 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 data packet header), if the queue size exceeds Q_h. Set $Local\_BCN := 1$ (buffer congestion), if $Q_{max} > Q_{goal}$; otherwise $Local\_BCN := 0$.

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, it resets Q_{max} = 0 and $Local\_BCN$ = 0 for the next buffer-congestion control.

III. The System Model

A transport-layer connection under the proposed flow-control scheme is a dynamic feedback control system. We model this system by using the first-order fluid analysis, where two real-valued functions $R(t)$ and $Q(t)$ represent the source-rate and the bottleneck queue-length functions, respectively. We also assume the existence of only a single bottleneck with queue length $Q(t)$ and a "persistent" source, which always has data packets to send at a rate subject to $R(t)$, for each connection. Such a data source model does represent many bulk data-transfer applications such as large file transfer and image retrieval.

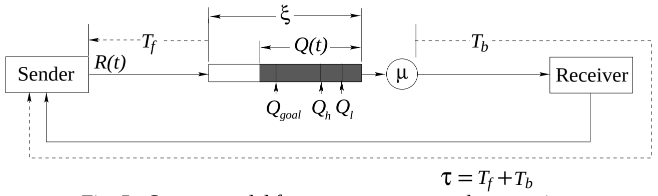


Fig. 5. System model for a transport protocol connection.

A. System Description and State Equations

Fig. 5 depicts the system model for a transport protocol 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 RTD. The source data rate $R(t)$ is dictated by the bottleneck's currently-available bandwidth capacity μ (BW). When $R(t) > \mu$, the bottleneck queue $Q(t)$ builds up, and the bottleneck drops newly-arriving packets if $Q(t)$ 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:

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

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

where the rates "additive increase" and "multiplicative decrease" are modeled by "linear increase" and "exponential decrease", respectively, in a continuous-time domain [11]; and $\alpha = \frac{1}{\Delta}(RIR)$ and $\beta = 1 + log\ (RDP)$ for a rate-adjustment interval Δ (control packet interval).

The second-order rate control described in Section II is a discrete-time control process since it is only exercised 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:

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

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

B. Rate-Control Performance Analysis

Using Eqs. (1)–(2) for the case of $Q_{max} < \xi$, we derive a set of rate-control performance expressions. We only list some of them, which will be used in the following sections, but others and more detailed derivations can be found in [10]. Fig. 6 illustrates the dynamic behavior of $R(t)$ and $Q(t)$. The maximum rate R_{max} is given by:

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

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 to R_{max} by:

$$T_{max} \overset{\triangle}{=} T_f + T_q + T_b = T_f + \sqrt{\frac{2Q_h}{\alpha}} + T_b. \tag{5}$$

Then, the maximum queue length is expressed as

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

where T_d is the time for $R(t)$ to drop from R_{max} back to μ, and is obtained, by letting $R(T_d) = \mu$, as:

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

Then, the maximum queue length is obtained as:

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

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

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

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

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

The minimum rate is then given as

$$R_{min} = \mu e^{-(1-\beta)\frac{(T_l + T_f + T_b)}{\Delta}}. \tag{11}$$

We define the rate-control cycle as

$$T \overset{\triangle}{=} T_q + T_d + T_l + 2\tau + T_r, \tag{12}$$

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

$$\begin{aligned} \overline{R} \overset{\triangle}{=} & \frac{1}{T} \int_{t_0}^{t_0+T} R(t) dt = \frac{1}{T} \left[\int_0^{T_{max}} (\mu + \alpha t) dt \right. \\ & \left. + \int_0^{T_e} R_{max} e^{-(1-\beta)\frac{t}{\Delta}} dt + \int_0^{T_r} (R_{min} + \alpha^* t) dt \right] \end{aligned} \tag{13}$$

where $T_e = T_d + T_l + \tau$. Simplifying Eq. (13), we obtain

$$\begin{aligned} \overline{R} = & \frac{1}{T} \left[\mu T_{max} + \frac{\alpha}{2} T_{max}^2 + R_{max} \frac{\Delta}{1-\beta} \right. \\ & \left. \cdot \left(1 - e^{-(1-\beta)\frac{T_e}{\Delta}} \right) + T_r R_{min} + \frac{\alpha^*}{2} T_r^2 \right]. \end{aligned} \tag{14}$$

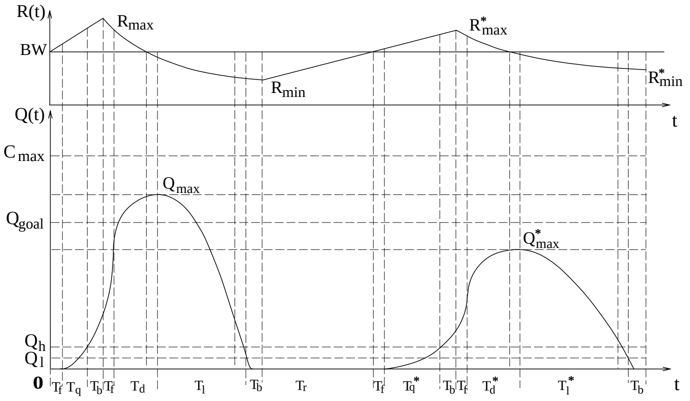


Fig. 6. Dynamics of $R(t)$ and $Q(t)$ for $Q_{max} < \xi \ (= C_{max})$.

IV. PACKET-LOSS ANALYSIS

In reality, the buffer capacity ξ at a bottleneck router is always finite. In this section, we consider the case where $Q_{max} > \xi$ and packets are lost due to buffer overflow.

A. Packet-Loss Calculation

In order to quantitatively evaluate the loss-control performance of the proposed error and flow control 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; and then γ and η of one rate-control cycle are expressed as:

$$\gamma \overset{\triangle}{=} \frac{\rho}{T\overline{R}} \quad \text{and} \quad \eta \overset{\triangle}{=} 1 - \gamma = 1 - \frac{\rho}{T\overline{R}} \tag{15}$$

where T is the rate-control cycle specified by Eq. (12), ρ is the number of lost packets during T, and $\overline{R}$ is the average throughput determined by Eq. (14). □

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 without any retransmission. 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 1: If a protocol connection with the buffer capacity $Q_h < \xi < \infty$ is flow-controlled under the rate-control scheme described by the state equations: Eqs. (1)–(2) and α-control law defined in Eq. (3), then the number ρ of lost packets during one rate-control cycle T is determined by:

$$\rho = \begin{cases} \frac{1}{2}\alpha \left(T_{max}^2 - t_\xi^2 \right) - \mu T_d + R_{max} \frac{\Delta}{1-\beta} & \\ \quad \cdot \left[1 - e^{-\frac{1-\beta}{\Delta} T_d} \right]; & \text{if } t_\xi \le T_{max} \\ \mu \left(t_\xi - T_{max} - T_d \right) + R_{max} \frac{\Delta}{1-\beta} & \\ \quad \cdot \left[e^{-\frac{1-\beta}{\Delta}(t_\xi - T_{max})} - e^{-\frac{1-\beta}{\Delta} T_d} \right]; & \text{if } t_\xi > T_{max} \end{cases} \tag{16}$$

where all variables are the same as defined in Section III, except that $t_\xi = \sqrt{\frac{2\xi}{\alpha}}$ if $\xi \le \frac{1}{2}\alpha T_{max}^2$; else t_ξ is the non-negative real

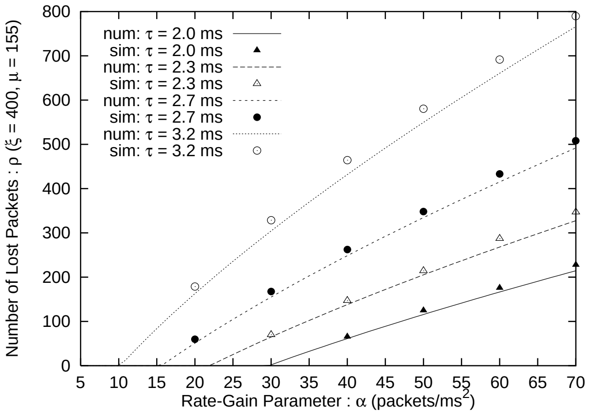


Fig. 7. Number of lost packets (ρ) vs. α.

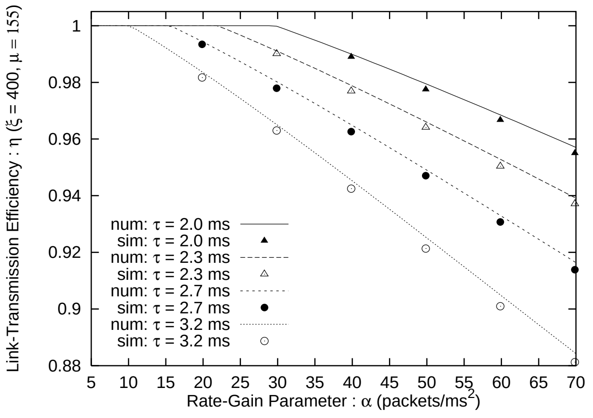


Fig. 8. Link-trans. efficiency (η) under retransmissions vs. α.

root of the following non-linear equation

$$\frac{1}{2}\alpha T_{max}^2 + R_{max}\,\frac{\Delta}{1-\beta}\left(1-e^{-(1-\beta)\frac{t_\xi - T_{max}}{\Delta}}\right) - \mu(t_\xi - T_{max}) - \xi = 0, \qquad (17)$$

if $\xi > \frac{1}{2}\alpha T_{max}^2$.

Proof: The detailed proof is provided in Appendix A. □

B. Performance Evaluation of Loss Control

Consider the bottleneck with μ = 367 packets/ms (155 Mbps), ξ = 400 packets; Q_h = 50 packets, and $q = 0.6$. Fig. 7 plots the number of lost packets, ρ, obtained from Eq. (16), against α for different RTD τ's. We observe that ρ increases with α, and if α is given, ρ gets larger as τ increases. It is therefore necessary to apply α-control to reduce the packet losses due to the increases of the number of cross-traffic flows and their RTD's. Packet losses cause retransmissions, and thus affect link-transmission efficiency. In Fig. 8, the link-transmission efficiency η is plotted versus α for the same parameters. As illustrated in Fig. 8, η = 1 at the beginning, implying that there is no retransmission (losses) if α is small enough. When 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/ms^2 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. Using the NetSim, we also simulated packet-losses ρ and link-transmission efficiency η, 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 function between Q_{max} and α as shown in Eq. (8), buffer-allocation control can be treated 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 $\boldsymbol{\alpha}(k) = (\alpha_1(k), \cdots, \alpha_n(k))$ represent the rate-gain parameter 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 defined by the closeness between the superposed α-allocation, $\alpha_t(k) \overset{\Delta}{=} \sum_{i=1}^n \alpha_i(k)$, 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 $\boldsymbol{\alpha}(k)$ to α_{goal} as close as possible and as fast as possible from any initial state.

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

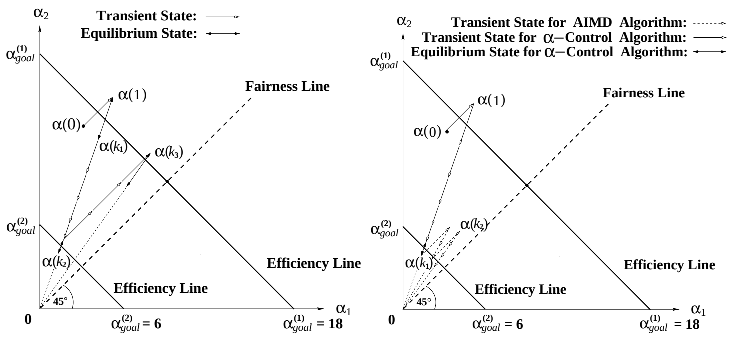
Notice that $\frac{1}{n} \leq \phi(\boldsymbol{\alpha}(k)) \leq 1$. $\phi(\boldsymbol{\alpha}(k)) = 1$ if $\alpha_i(k) = \alpha_j(k)$, $\forall i \neq j$. This corresponds to the "best" fairness. $\phi(\boldsymbol{\alpha}(k)) = \frac{1}{n}$ if the entire α is allocated to only one of n active connections.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

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 $\alpha(k{+}1) = g(\alpha(k), BCN(k{-}1,k)) \overset{\Delta}{=} p{+}q\alpha(k)$, where coefficients p and q are determined by feedback information $BCN(k{-}1,k)$. The theorem given below describes the feasibility and optimalness of the linear α-control, which ensures the convergence of α-control to the efficiency and fairness of buffer allocation.

Theorem 2: 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 detailed proof is provided in Appendix B. □

Remark. *Theorem 2* is an extension from bandwidth control [12] to buffer control, but differs from [12] 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 to 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



(a) $\alpha(k) \longrightarrow$ efficiency/fairness. (b) α-control vs. AIMD.

Fig. 9. α-allocation convergence to efficiency and fairness.

total allocation $\alpha_t(k)$, or the number of connections, keeps on going up and down due to the cross-traffic variation in real 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 analysis of Section III, we compute two examples in a 2-Dimension space (for two connections) to show the α-allocation convergence under the α-control in terms of efficiency and fairness. As shown in Fig. 9, any α-allocation of two connections at the k-th α-control is represented as a point $\boldsymbol{\alpha}(k) = (\alpha_1(k), \alpha_2(k))$ in a 2-Dimension space. All allocation points (α_1, α_2) for which $\alpha_1+\alpha_2 = \alpha_{goal}$ form the *efficiency line*, and all points for which $\alpha_1 = \alpha_2$ form the *fairness line* which is a 45° line. It is easy to verify that an additive increase, $(\alpha_1, \alpha_2)+p \triangleq (\alpha_1+p, \alpha_2+p)$, corresponds to moving up ($p>0$) along a 45° line, and a multiplicative decrease or increase, $q(\alpha_1, \alpha_2) \triangleq (q\alpha_1, q\alpha_2)$ ($0<q<1$ or $q>1$), corresponds to moving along the line that connects the origin to (α_1, α_2).

Example 1. Let two connections sharing a bottleneck be flow-controlled by the α-control law. The connection bottleneck is characterized by: μ = 184 packets/ms, Q_{goal} = 200 packets, Q_h = 18 packets, and τ = 2 ms (so, α_{goal} = 18 packets/ms^2). Consider a scenario (see Fig. 9(a)) where α_{goal} is equal to $\alpha_{goal}^{(1)} = 18$ initially, but reduces to $\alpha_{goal}^{(2)} = 6$ at the k_1-th α-control, and then returns to $\alpha_{goal}^{(1)}$ after the k_2-th α-control. The variation of α_{goal} is due to the variation in the number of connections between n = 2 and n = 6, or due to the variations in τ between $\tau^{(1)}$ = 2 ms and $\tau^{(2)}$ = 3.34 ms. We take q = 0.8 and p = 4 for the two connections with Q_{goal} = 200 and τ = 2 ms. Thus, $\frac{1}{2}p = 2$ for each of the two connections. Suppose $\boldsymbol{\alpha}(0) = (3.035, 12.76)$ initially. Then, by α-control, $\boldsymbol{\alpha}(1) = \boldsymbol{\alpha}(0) + 2 = (5.035, 14.76)$ and $\boldsymbol{\alpha}(2) = 0.8\boldsymbol{\alpha}(1) = (4.028, 11.81)$ since $\alpha_1(0) + \alpha_2(0) = 15.795 < \alpha_{goal}^{(1)}$ and $\alpha_1(1) + \alpha_2(1) = 19.795 > \alpha_{goal}^{(1)}$. Thus, α-control enters equilibrium state around $\alpha_{goal}^{(1)}$ during which $\boldsymbol{\alpha}(k)$ fluctuates between $(4.028, 11.81)$ and $(5.035, 14.76)$. When α_{goal} reduces to $\alpha_{goal}^{(2)}$, equilibrium is broken and $\boldsymbol{\alpha}(k)$ converges to a new equilibrium state multiplicatively by 5 α-control cycles, and fluctuates between $(1.32, 3.87)$ and $(1.65, 4.838)$. Finally, α_{goal} returns back to $\alpha_{goal}^{(1)}$, $\boldsymbol{\alpha}(k)$ converges to the new equilibrium state additively through 3 α-control cycles and fluctuates between $(6.12, 8.671)$ and $(7.65, 10.838)$. We observe that in transient state, α-control not only guarantees the monotonic convergence to the neighborhood of efficiency-line in both increase and decrease phases, but also improves the fairness index from $\phi(\boldsymbol{\alpha}(0)) = 0.725$ to $\phi(\boldsymbol{\alpha}(k_3)) = 0.971$ as shown in Fig. 9(a), where $\boldsymbol{\alpha}(k_3) = (7.65, 10.838)$ is closer to the fairness line than $\boldsymbol{\alpha}(0) = (3.035, 12.76)$.

Example 2. The second example compares α-control with the AIMD (Additive-Increase and Multiplicative-Decrease) algorithm applied to α (see Fig. 9(b)). The parameters and $\boldsymbol{\alpha}(0)$ are the same as in Example 1 except that α_{goal} reduces to, and stays with, $\alpha_{goal}^{(2)}$ after $\boldsymbol{\alpha}(k)$ reaches $\boldsymbol{\alpha}(1)$. We observe that both schemes share the control trajectory from $\boldsymbol{\alpha}(0)$ up to $\boldsymbol{\alpha}(k_1)$. However, after $\boldsymbol{\alpha}(k)$ is driven to $\boldsymbol{\alpha}(k_1)$, the two trajectories split. Under α-control, $\boldsymbol{\alpha}(k)$ converges to an equilibrium state and locks itself within a small neighborhood of $\alpha_{goal}^{(2)}$: $\{(1.32, 3.87), (1.65, 4.838)\}$. In contrast, under the AIMD algorithm, $\boldsymbol{\alpha}(k)$ does not confine itself within a small neighborhood of $\alpha_{goal}^{(2)}$ and, in fact, $\boldsymbol{\alpha}(k)$ cannot even reach any equilibrium state. The resultant maximum buffer-allocation "overshoot" for the AIMD at $\boldsymbol{\alpha}(k_2)$ is as high as $Q_{max}^{(k_2)} - Q_{goal} = 261 - 200 = 61$ packets, which is about 9 times as large as that for α-control (with the maximum overshoot equal to $Q_{max}^{(k_1+1)} - Q_{goal} = 207 - 200 = 7$). So, even though the AIMD algorithm is better than α-control in term of speed of convergence to fairness, the AIMD's maximum buffer requirement and potential loss rate are much higher than α-control, especially when the variation of n, the number of connections, or RTD: $(\tau^{(2)} - \tau^{(1)})$, is large.

VI. Performance Evaluations

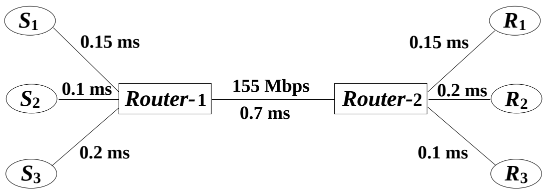
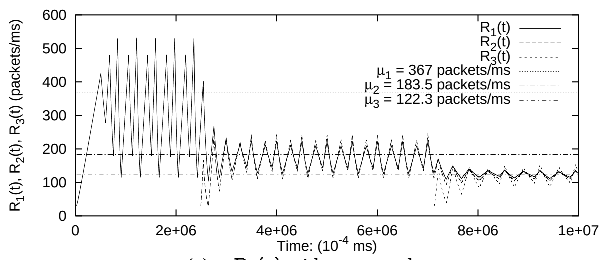
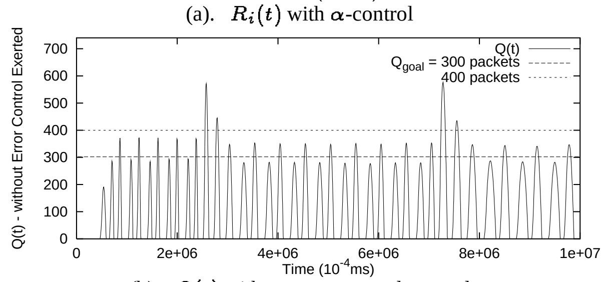


Fig. 10. The simulation model.

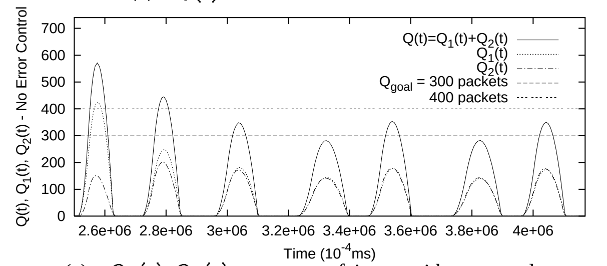
Using the NetSim, we built up a simulator which implemented our proposed flow and error control scheme. As shown in Fig. 10, the simulated network consists of three connections C_1, C_2, and C_3 which share a common network bottleneck link between $Router$-1 and $Router$-2. The connection C_i's data packets are sent from sender S_i to its corresponding receiver R_i. The simulation parameters for the network are bottleneck bandwidth $\mu = 367$ packets/ms (155 Mbps), RTDs $\tau = 2$ ms, and $Router$-1's buffer size $\xi = 800$ packets (for $Q_{max} < \xi$), or $\xi = 400$ (for $Q_{max} > \xi$). For rate-control parameters: Q_h = 50 packets, $Q_{goal} = 300$ packets, $R_0 = 30$ packets/ms, Δ = 0.4 ms, q = 0.6, p = 2.9, α_0 = 8.7, 14.7, and 17.7 packets/m^2 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 equipped with the α-controlled and non-α-controlled schemes. The simulated source rate $R_i(t)$ $(i = 1, 2, 3)$ and the bottleneck queue length $Q(t)$ are plotted in Figs. 11(a)–(d) for the α-controlled scheme, and in Figs. 12(a)–(d) for the non-α-controlled scheme. We compare the two schemes in the following two cases.



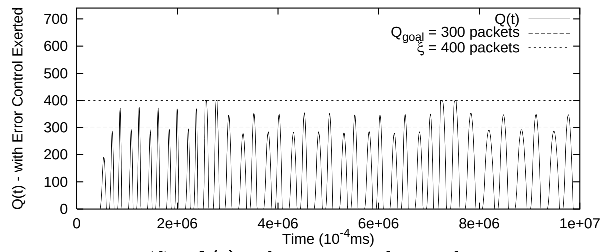
(a). $R_i(t)$ with α-control



(b). $Q(t)$ without error control exerted

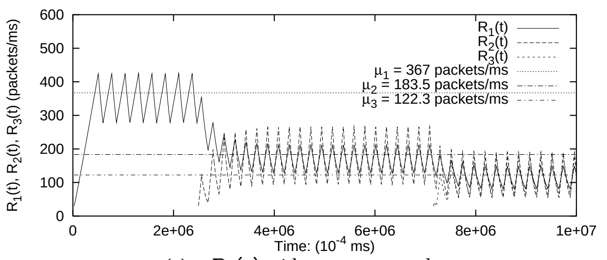


(c). $Q_1(t), Q_2(t)$ converge to fairness with α-control

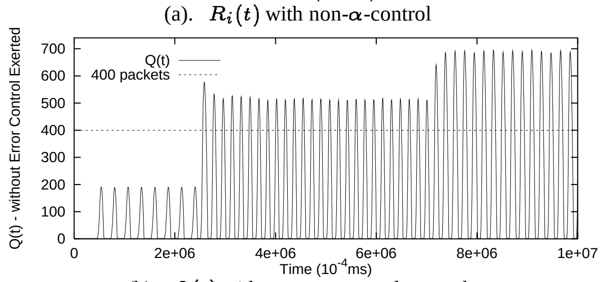


(d). $Q(t)$ with error control exerted

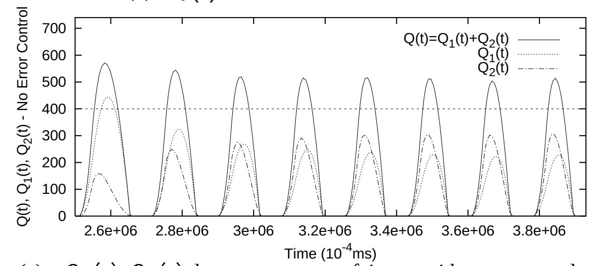
Fig. 11. Dynamics of $R_i(t)$ and $Q(t)$ with α-control.



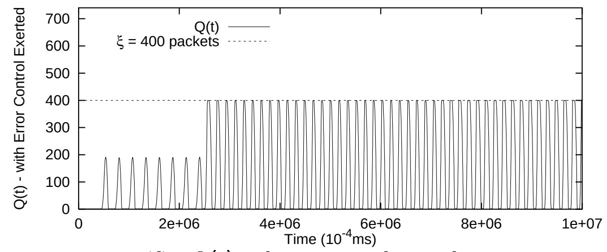
(a). $R_i(t)$ with non-α-control



(b). $Q(t)$ without error control exerted



(c). $Q_1(t), Q_2(t)$ do not converge to fairness without α-control



(d). $Q(t)$ with error control exerted

Fig. 12. Dynamics of $R_i(t)$ and $Q(t)$ with non-α-control.

CASE I. $Q_{max} < \xi = 800$: error control not exerted.

(1) During T_1 (n = 1). For the α-controlled scheme, Fig. 11(a) shows that $R_1(t)$ converges to μ_1=367 packets/ms since only C_1 is active and $R_1(t)$ grabs all the available bandwidth. From Figs. 11(a)–(b), we observe that experiencing one transient cycle due to $Q(t)$'s maximum $Q_{max} = 190 < Q_{goal}$ at 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. With sufficient available buffer space, the increased α enhances the system responsiveness to grab newly created available bandwidth if any. In contrast, for non-α-control, Fig. 12(a) shows $R_1(t)$ also converges to μ_1 =367, but Q_{max} (see Fig. 12(b)) is always 190 during T_1, utilizing less than 25% of buffer capacity without enhancing the system responsiveness.

(2) During T_2 (n = 2). For the α-controlled scheme, Fig. 11(a) shows $R_1(t)$ and $R_2(t)$ experience two transient cycles during which $R_1(t)$ gives up $\frac{1}{2}\mu_1 = \mu_2$ bandwidth to $R_2(t)$. Fig. 11(b) shows that a big queue build-up $Q_{max} = 590$ starting at $t_1 = 245$. This is expected because the the number of active connections increases from n = 1 to n = 2, and thus the new superposed rate-gain parameter is in effect equal to the sum of each connection's rate-gain parameter. Driven by the 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. Additionally, convergence to the buffer-occupancy fairness under the α-control is also verified by C_1, C_2's per-connection queues $Q_1(t)$ and $Q_2(t)$ (see Fig. 11(c), the zoom-in picture of Fig. 11(b)), which converge to each other during the two-cycle transient states (notice $Q(t) = Q_1(t) + Q_2(t)$). By contrast, for the non-α-controlled scheme, Fig. 12(b) illustrates that Q_{max} shoots up to 590 and remains above 520 even after entering the equilibrium. Moreover, Fig. 12(c), the zoom-in picture of Fig. 12(b), shows that buffer occupancy is not fair because Q_{max} of $Q_1(t)$, which is larger than Q_{max} of $Q_2(t)$ during transient state, becomes smaller than that of $Q_2(t)$ after entering the equilibrium as $R_1(t)$'s rate gain-parameter $\alpha_1 = 8.7$ is smaller than $R_2(t)$'s rate gain-parameter $\alpha_2 = 14.7$.

(3) During T_3 (n = 3). At $t_2 = 710$ ms, C_3 joins in, thus n increased to 3. For the α-controlled scheme, Fig. 11(a) shows that after 2 transient cycles, $R_1(t)$ and $R_2(t)$ both yield some bandwidth to $R_3(t)$ such that they take one third bandwidth μ_3 each. Again, Fig. 11(b) shows that Q_{max} increases dramatically up to 585 at t_2 as a result of one more connection joining in and n = 3. With the α-control, Q_{max} quickly returns to Q_{goal}'s neighborhood within 2 transient cycles. In contrast, for the non-α-controlled, after Q_{max} jump up to 700 (see Fig. 12(b)), it never drops from 700 all time in T_3.

		***α*-Control-Based Protocols**								**Non-*α*-Control Based Protocols**							
C_i	T_{trans}	N_{trans}	N_{recv}	$N_{retrans}$	γ	η	$\overline{R}_{trans}$	$\overline{R}_{recv}$	N_{trans}	N_{recv}	$N_{retrans}$	γ	η	$\overline{R}_{trans}$	$\overline{R}_{recv}$
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 α-CONTROLLED AND NON-α-CONTROLLED SCHEMES.

CASE II. $Q_{max} > \xi = 400$**: error control exerted.**
The other parameters remains the same. For the α-controlled scheme, Fig. 11(d) shows that the packet droppings only occur during the short transient (only 2 cycles) state starting at t_1 and t_2, where $Q(t) = \xi$. However, as soon as the flow-controlled system, driven by the α-control, settles down to an equilibrium state, the bottleneck stops dropping packets, because Q_{max} already converges to the neighborhood of Q_{goal} upper-bounded by ξ. Since there is no packet dropping during the designated equilibrium, and thus no need for retransmissions, we call the optimal equilibrium state specified by the α-control the *retransmission-free equilibrium state*. The retransmission-free equilibrium of the α-control ensures that the need for the retransmission due to congestion is minimized. In contrast, Fig. 12(d) shows that, for the non-α-control, packet-droppings occur not only during the transient state when the number of active connections increases at t_1 and t_2, but also after the system enters an equilibrium state. In fact, as shown in Fig. 12(d), the non-α-controlled scheme may never reach a retransmission-free equilibrium state.

During the packet-loss periods, our proposed error-control mechanism is kicked in and each lost packet is retransmitted (can be more than once if it is lost again) until it is successfully received. TABLE I collects the error and flow control data from the three connections C_1, C_2, C_3, under both α-controlled and non-α-controlled schemes. As shown in TABLE I, the number of retransmissions, denoted by $N_{retrans}$, i.e., the number of lost packets, is verified by difference of N_{trans} (the number of both transmitted and retransmitted packets) minus N_{recv} (the number of correctly received packets) during the transmission time period T_{trans}. The corresponding packet-loss rate γ and link-transmission efficiency η are calculated by *Definition 1*.

For the α-controlled scheme, we observe that the number of retransmissions $N_{retrans}$ and loss rate γ are very small and the corresponding link-transmission efficiency η is as high as 99.8% for all three connections. This is because α-control always drives the flow-controlled system to settle down to an equilibrium state where there is no loss, and hence the retransmission-free is guaranteed. By contrast, for the non-α-controlled scheme, the number of retransmissions $N_{retrans}$ and loss rate γ are 10 to 35 times as large as those in α-controlled scheme for the three connections. Consequently, the link-transmission efficiency 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. These observations are expected since the retransmission-free (no losses) equilibrium of α-control minimizes the retransmission due to congestions. TABLE I also shows that the α-controlled scheme outperforms the non-α-controlled scheme on the average throughputs $\overline{R}_{trans}$ (sending end) and $\overline{R}_{recv}$ (receiving end — *goodput*). The difference ($\overline{R}_{trans} - \overline{R}_{recv}$) is also found much smaller for α-controlled scheme than that for non-α-controlled scheme due to much less packet dropping (and hence much fewer retransmissions) of α-control.

Based on the simulation data for a single connection, TABLE II compares the α-controlled protocol with the non-α-controlled protocol in terms of transmission time and packet losses for transferring bulk data files of different sizes. The network and flow-control parameters are as follows. For both the α-controlled and non-α-controlled schemes, $\xi = 300$ packets, $Q_{goal} = 200$ packets, $\mu = 155$Mbps, $\tau = 2$ ms $\Delta = 1$ ms, $\alpha_0 = 91.6$ packets/ms^2, but q = 0.6 for α-control.

From TABLE-II, we observe that for any given file size (K-byte), the non-α-control scheme's transmission time, denoted by $T_{\overline{\alpha}}$, is larger than α-control scheme's transmission time, denoted by T_α. As shown in TABLE-II, $T_{\overline{\alpha}}$ is about 20% larger than T_α for all scenarios simulated. The difference, $T_{\overline{\alpha}} - T_\alpha$, of the two schemes measures the performance improvement of α-controlled protocol over the non-α-controlled protocol, in terms of file transmission time. TABLE II also shows $T_{\overline{\alpha}} - T_\alpha$ monotonically increases as the file size increases. This is expected since, as shown in TABLE II, the number of lost packets (***Loss***) is fixed for α-controlled scheme (where loss only occurs during the transient state, and no-loss/retransmission-free — due to α-control — is achieved during the equilibrium state) while the number of lost packets (***Loss***) increases with the file size for the non-α-controlled scheme (where loss occurs during both transient and equilibrium states). Note that the total loss number during a file transmission, denoted by ***Loss*** (with its unit converted into K-byte) in TABLE II, includes both the single-loss and multiple-losses which means a packet is lost and retransmitted multiple times before it is correctly received at the receiver. If a packet is retransmitted m ($\geq$1) times before it is received correctly, it will contribute m losses to ***Loss***.

	Non-*α*-Controlled		***α*-Controlled**		
File Size (Kb)	$T_{\overline{\alpha}}$ (ms)	***Loss*** (Kb)	T_α (ms)	***Loss*** (Kb)	$T_{\overline{\alpha}} - T_\alpha$ (ms)
1700	124.474	220	102.971	21	21.503
3400	246.200	444	202.827	21	43.373
5100	363.515	654	302.876	21	60.639
6800	483.100	885	402.276	21	80.824
8500	602.314	1098	502.666	21	99.648
10200	721.870	1308	602.476	21	119.394
11900	837.735	1542	702.111	21	135.624
13600	956.639	1752	802.340	21	154.299
15300	1077.165	1964	902.000	21	175.165
17000	1197.559	2197	1002.133	21	195.426
18700	1318.305	2407	1102.000	21	216.305
20400	1439.800	2631	1202.770	21	237.030
22100	1565.014	2850	1302.870	21	262.144
23800	1680.730	3061	1402.770	21	277.960
25500	1797.434	3294	1502.818	21	294.616
27200	1919.119	3505	1602.541	21	316.578
28900	2047.026	3715	1702.787	21	344.239
30600	2164.099	3949	1802.227	21	361.872
32300	2277.014	4159	1902.887	21	374.127
34000	2410.499	4378	2002.664	21	407.835

TABLE II

FILE-TRANSMISSION TIME COMPARISON BETWEEN α-CONTROLLED AND NON-α-CONTROLLED PROTOCOLS.

The second reason for the monotonic increase of $T_{\overline{\alpha}} - T_\alpha > 0$ is that the average throughput of the α-controlled scheme is higher than that of the non-α-controlled scheme, as shown in the previous simulation results (see TABLE I). As a result, under the same network conditions, the α-controlled protocol needs relatively much less time than the non-α-controlled protocol to finish transferring a file of the same size, particularly when the file size gets large.

VII. Conclusion

We proposed and analyzed an efficient flow and error control scheme for high-throughput transport protocols. We employ a second-order rate-control scheme for flow control and decouples it from error control. The second-order rate control minimizes the packet losses and retransmissions by adjusting the rate-gain parameter in response to the variations of cross-traffic flows and their RTDs. Using NACK and selective retransmission, the error control scheme recovers the lost packets, only if needed. The separation of flow and error control enhances the throughput since the source rate control is independent of the dynamics of error-control window. By fluid analysis, we modeled the packet-loss behavior and derived the closed-form expressions for packet-loss rate and link-transmission efficiency. The analytical results show that the α-control can drive the system state to an optimal equilibrium state, where the retransmission-free is guaranteed. The simulation experiments verify the derived analytical results, and demonstrate the superior of the α-controlled scheme to other non-α-controlled schemes in terms of loss/retransmission control, link-transmission efficiency, file-transmission time, throughput, and buffer-occupancy fairness.

References

[1] V. Jacobson, “Congestion avoidance and control,” in *Proc. of ACM SIGCOMM*, pp. 1–12, 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, and D. Towsley, “A TCP-friendly rate adjustment protocol for continuous media flows over best effort networks,” in *Proc. of ACM SIGMETRICS*, pp. 220–221, 1999.

[4] L. Zhang, “Why TCP timers don't work well,” in *Proc. of ACM SIGCOMM*, 1986.

[5] X. Zhang, K. G. Shin, D. Saha, and D. Kandlur, “Scalable flow control for multicast ABR services,” in *Proc. of IEEE INFOCOM*, pp. 837–846, March 1999.

[6] S. Floyd, “TCP and explicit congestion notification,” *ACM SIGCOMM Computer Communication Review*, pp. 60–72, April 1995.

[7] W. R. Stevens, *TCP/IP Illustrated, Vol. 1*, Addison-Wesley, 1994.

[8] L. S. Brakmo and L. L. Peterson, “TCP vegas: end to end congestion avoidence on a global internet,” *IEEE Journal on Selected Areas in Communications*, vol. 13, no. 8, pp. 1465–1480, October 1995.

[9] B. T. Doshi, P. K. Johri, A. N. Netravali, 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 1995.

[10] X. Zhang, K. G. Shin, and Q. Zheng, “Integrated rate and credit feedback control for ABR services in ATM networks,” in *Proc. of IEEE INFOCOM*, pp. 1297–1305, April 1997. *Full paper version: Tech. Report, CSE-TR-354-97, The University of Michigan.*

[11] N. Yin and M. G. Hluchyj, “On closed-loop rate control for ATM cell relay networks,” in *IEEE INFOCOM*, pp. 99–109, 1994.

[12] 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.

Appendix

A. Proof of Theorem 1

We first need to determine the upper and lower bounds of “loss period”, as shown in Figs. 13–14, defined by $[t_1, t_2]$ within a rate-control cycle where t_1 (t_2) is the time when the router starts (stops) dropping packets. So, t_1 can be obtained by solving the bottleneck queue state equation Eq. (2) as follows:

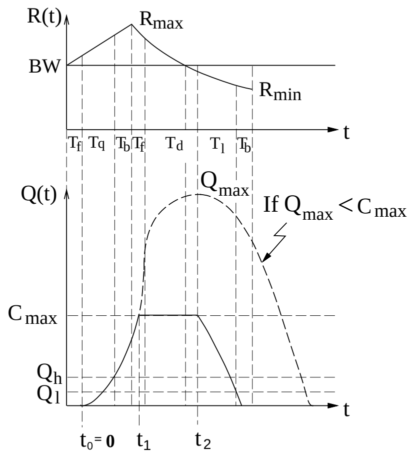


Fig. 13. Number of lost packets derivation — **Case 1.**

$$Q(t_1) = \int_0^{t_1} \left[R(v - T_f) - \mu \right] dv = \xi = C_{max} \qquad (18)$$

where, to simplify the calculations, we shift the time-zero point to $t_0 = 0$ when $R(t - T_f)$ reaches bandwidth capacity μ = BW. Depending upon the the rate-control parameters, t_1 can be either smaller (see Fig. 13), or larger (see Fig. 14), than $T_{max} \triangleq T_q + T_b + T_f$. Since T_{max} is the last moment of $R(t)$ applying linear-control, and t_1 is the time when $Q(t)$ hits $\xi = C_{max}$ for the first time, the conditions $t_1 \le T_{max}$ and $t_1 > T_{max}$ can be equivalently expressed as $\xi \le \frac{1}{2}\alpha T_{max}^2$ and $\xi > \frac{1}{2}\alpha T_{max}^2$, respectively, and also we use t_ξ to represent the lower bound of $[t_1, t_2]$ from now on. These conditions generate the following two different cases in calculating $t_\xi \triangleq t_1$:

Case 1. If $\xi \le \frac{1}{2}\alpha T_{max}^2$: $Q(t)$ is generated only by $R(t)$'s linear-control period, thus (see Fig. 13)

$$Q(t_\xi) = \int_0^{t_\xi} \alpha\, t\, dt = \xi \quad \Longrightarrow \quad t_\xi = \sqrt{\frac{2\xi}{\alpha}} \qquad (19)$$

which gives the first case of computing t_ξ in *Theorem 1.*

Case 2. If $\xi > \frac{1}{2}\alpha T_{max}^2$: $Q(t_\xi)$ is contributed by both $R(t)$'s linear-control and exponential-control periods, thus (see Fig. 14)

$$\begin{aligned} Q(t_\xi) &= \int_0^{T_{max}} \alpha\, t\, dt \\ &+ \int_0^{t_\xi - T_{max}} \left(R_{max} e^{-(1-\beta)\frac{t}{\Delta}} - \mu \right) dt = \xi. \end{aligned} \qquad (20)$$

Eq. (17) follows by reducing Eq. (20).

By definition of $[t_1, t_2]$, $R(t) \ge \mu$ must hold during $[t_1, t_2]$, which is the condition for $Q(t) = \xi$. During $[t_1, t_2]$, when $R(t) > \mu$, $Q(t)$ tends to increase, but upper-bounded by ξ, thus $Q(t) = \xi$. When $R(t) = \mu$, $Q(t) = \xi$ retains because the bottleneck queue length remains unchanged when the bottleneck's arrival rate equals to its departuring rate. Therefore, the lower bound t_1 is the time when $R(t)$ drops back exactly to μ and $R(t)$'s any further

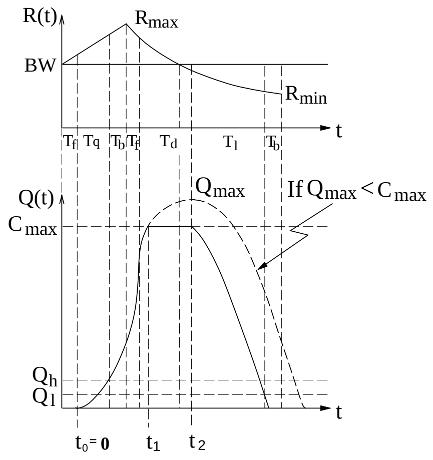


Fig. 14. Number of lost packets derivation — **Case 2.**

decrease will lead to $Q(t) < \xi$. By the definition of T_d given in Eq. (7), we obtain $t_2 = T_{max} + T_d$.

During $[t_1, t_2]$, $R(t) \geq \mu$, and thus we can rewrite $R(t)$ into two terms: $R(t) = \mu + (R(t) - \mu)$. The first term μ maintains $Q(t) = \xi$, and the second term $(R(t) - \mu)$ generates packet drops. Therefore, the *packet-dropping rate* is $(R(t) - \mu)$. Then, the number of lost packets ρ, within one rate-control cycle, can be obtained by

$$\rho = \int_{t_1}^{t_2} [R(t) - \mu]\, dt = \int_{t_\xi}^{T_{max}+T_d} [R(t) - \mu]\, dt. \qquad (21)$$

which is also divided into two cases as follows, because $R(t)$ has different expressions, depending on $\xi \leq \frac{1}{2}\alpha T_{max}^2$ or $\xi > \frac{1}{2}\alpha T_{max}^2$.

Case 1. If $\xi \leq \frac{1}{2}\alpha T_{max}^2$: $R(t)$ consists of two parts, and thus (see Fig. 13)

$$\begin{aligned} \rho &= \int_{t_\xi}^{T_{max}+T_d} [R(t) - \mu]\, dt \\ &= \int_{t_\xi}^{T_{max}} \alpha t dt + \int_0^{T_d} \left(R_{max} e^{-(1-\beta)\frac{t}{\Delta}} - \mu\right) dt. \end{aligned} \qquad (22)$$

Reducing Eq. (22) yields the first part of Eq. (16).

Case 2. If $\xi > \frac{1}{2}\alpha T_{max}^2$: $R(t)$ has only one part, and thus (see Fig. 14)

$$\begin{aligned} \rho &= \int_{t_\xi}^{T_{max}+T_d} [R(t) - \mu]\, dt \\ &= \int_0^{T_d} \left(R_{max} e^{-(1-\beta)\frac{t}{\Delta}} - \mu\right) dt \\ &\quad - \int_0^{t_\xi - T_{max}} \left(R_{max} e^{-(1-\beta)\frac{t}{\Delta}} - \mu\right) dt. \end{aligned} \qquad (23)$$

Simplifying Eq. (23) leads to the second part of Eq. (16). This completes the poof. □

B. *Proof of Theorem 2*

We prove this theorem by considering the transient state and equilibrium state, respectively.

(I) In Transient State. The linear α-control function can be expressed by

$$\alpha_i(k+1) = \begin{cases} p_I + q_I \alpha_i(k); & \text{if } BCN(k-1,k) = (0,0), \\ p_D + q_D \alpha_i(k); & \text{if } BCN(k) = 1, \end{cases} \qquad (24)$$

We now derive the constraints to determine the control-function coefficients p_I, q_I, p_D, and q_D to guarantee convergence to both efficiency and fairness.

(1) Convergence to Efficiency. To ensure $\alpha_t(k)$ converges to its target α_{goal}, the α-control must be a negative feedback at each α-control cycle, i.e., it is requested that

$$\begin{cases} \alpha_t(k+1) > \alpha_t(k); & \text{if } BCN(k-1,k) = (0,0) \\ \alpha_t(k+1) < \alpha_t(k); & \text{if } BCN(k) = 1 \end{cases} \qquad (25)$$

where $\alpha_t(k+1) = \sum_{i=1}^n \alpha_i(k+1)$ and $\alpha_t(k) = \sum_{i=1}^n \alpha_i(k)$. By Eq. (24), Eq. (25) reduces to

$$\begin{cases} q_I > 1 - \frac{np_I}{\sum_{i=1}^n \alpha_i(k)}, & \forall\, n \text{ and } \forall\ \sum_{i=1}^n \alpha_i(k); \\ & \text{if } BCN(k-1,k) = (0,0), \\ q_D < 1 - \frac{np_D}{\sum_{i=1}^n \alpha_i(k)}, & \forall\, n \text{ and } \forall\ \sum_{i=1}^n \alpha_i(k); \\ & \text{if } BCN(k) = 1, \end{cases} \qquad (26)$$

(2) Convergence to Fairness. Convergence of $\boldsymbol{\alpha}(k)$ to fairness can be expressed by

$$\lim_{k\to\infty} \phi(\boldsymbol{\alpha}(k)) = \lim_{k\to\infty} \frac{\left[\sum_{i=1}^n \alpha_i(k)\right]^2}{n\ \sum_{i=1}^n \alpha_i^2(k)} = 1. \qquad (27)$$

Plugging linear-control function of $g(\cdot,\cdot)$ into the fairness index and defining $\theta \triangleq \frac{p}{q}$, we get

$$\begin{aligned} \phi(\boldsymbol{\alpha}(k+1)) &\triangleq \frac{\left[\sum_{i=1}^n \alpha_i(k+1)\right]^2}{n \sum_{i=1}^n \alpha_i^2(k+1)} \\ &= \frac{\left(\sum_{i=1}^n [p + q\alpha_i(k)]\right)^2}{n \sum_{i=1}^n [p + q\alpha_i(k)]^2} \\ &= \frac{\left(\sum_{i=1}^n [\theta + \alpha_i(k)]\right)^2}{n \sum_{i=1}^n [\theta + \alpha_i(k)]^2} \\ &= \phi(\boldsymbol{\alpha}(k)) + [1 - \phi(\boldsymbol{\alpha}(k))] \\ &\quad \cdot \left[1 - \frac{\sum_{i=1}^n \alpha_i^2(k)}{\sum_{i=1}^n [\theta + \alpha_i(k)]^2}\right], \end{aligned}$$

and further,

$$\begin{aligned} \phi(\boldsymbol{\alpha}(k+1)) - \phi(\boldsymbol{\alpha}(k)) &= [1 - \phi(\boldsymbol{\alpha}(k))] \\ &\quad \cdot \left[1 - \frac{\sum_{i=1}^n \alpha_i^2(k)}{\sum_{i=1}^n [\theta + \alpha_i(k)]^2}\right] \end{aligned} \qquad (28)$$

Note that $\phi(\boldsymbol{\alpha}(k+1)) - \phi(\boldsymbol{\alpha}(k))$ in Eq. (28) is a monotonic-increasing function of $\theta \triangleq \frac{p}{q}$, and $\phi(\boldsymbol{\alpha}(k+1)) \geq \phi(\boldsymbol{\alpha}(k))$ *iff* $\theta \geq 0$. Thus, if $\theta > 0$, fairness increases: $\phi(\boldsymbol{\alpha}(k+1)) > \phi(\boldsymbol{\alpha}(k))$; if $\theta = 0$, the fairness maintains: $\phi(\boldsymbol{\alpha}(k+1)) = \phi(\boldsymbol{\alpha}(k))$. Since

$\theta \triangleq \frac{p_I}{q_I}$ in α-increase phase and $\theta \triangleq \frac{p_D}{q_D}$ in α-decrease phase, we get four possible cases as follows:

$$\begin{cases} \textbf{1.}\ \text{if } \frac{p_D}{q_D} > 0 \wedge \frac{p_I}{q_I} > 0, & \text{then } \phi(\boldsymbol{\alpha}(k+1)) > \phi(\boldsymbol{\alpha}(k)) \text{ in both } \alpha\text{-decrease and } \alpha\text{-increase;} \\ \textbf{2.}\ \text{if } \frac{p_D}{q_D} > 0 \wedge \frac{p_I}{q_I} = 0, & \text{then } \phi(\boldsymbol{\alpha}(k+1)) > \phi(\boldsymbol{\alpha}(k)) \text{ in } \alpha\text{-decrease and } \phi(\boldsymbol{\alpha}(k+1)) = \phi(\boldsymbol{\alpha}(k)) \text{ in } \alpha\text{-increase;} \\ \textbf{3.}\ \text{if } \frac{p_D}{q_D} = 0 \wedge \frac{p_I}{q_I} > 0, & \text{then } \phi(\boldsymbol{\alpha}(k+1)) = \phi(\boldsymbol{\alpha}(k)) \text{ in } \alpha\text{-decrease and } \phi(\boldsymbol{\alpha}(k+1)) > \phi(\boldsymbol{\alpha}(k)) \text{ in } \alpha\text{-increase;} \\ \textbf{4.}\ \text{if } \frac{p_D}{q_D} = 0 \wedge \frac{p_I}{q_I} = 0, & \text{then } \phi(\boldsymbol{\alpha}(k+1)) = \phi(\boldsymbol{\alpha}(k)) \text{ in both } \alpha\text{-decrease and } \alpha\text{-increase;} \end{cases} \tag{29}$$

Eq. (29) implies that control-function coefficients p_D, p_I, q_D, and q_I must all have the same signs if not zeroes. Combining Eq. (29) with Eq. (24), we conclude that these four control-function coefficients must be all positive if not zeroes, and q_I and q_D must be positive since $\alpha_i(k)\ \forall\ i$ are always positive numbers. The convergence condition given in Eq. (26) adds further constraints on q_D such that $0 < q_D < 1$. Thus, the constraints on the control-function coefficients, in terms of convergence to fairness and efficiency, can be summarized as

$$\textbf{constraint:}\{0 < q_D < 1, 0 < q_I, (p_D \geq 0 \wedge p_I > 0) \vee (p_D > 0 \wedge p_I \geq 0)\} \tag{30}$$

which include cases **1, 2,** and **3** as described in Eq. (29).

Since α-control is exercised on a per-connection basis, and i-th source does not have any information on $\alpha_j(k)$, $\forall\ j \neq i$ and value of n (α-control is a distributed algorithm), the convergence condition given in Eq. (26) cannot be explicitly used to further specify the control-function coefficients. In the absence of such information, each connection must satisfy the negative feedback condition as follows, which represents a stronger condition for convergence to the efficiency:

$$\begin{cases} \alpha_i(k+1) > \alpha_i(k) \Longrightarrow p_I + (q_I - 1)\alpha_i(k) > 0, \forall\ i, \\ \qquad\qquad \text{if } BCN(k-1,k) = (0,0); \\ \alpha_i(k+1) < \alpha_i(k) \Longrightarrow p_D + (q_D - 1)\alpha_i(k) < 0, \forall\ i, \\ \qquad\qquad \text{if } BCN(k) = 1; \end{cases} \tag{31}$$

Eq. (31) yields further constraints in determining control-function coefficients. Since $(q_D - 1)\alpha_i(k) < 0$ (due to Eq. (30)) may have an arbitrarily small absolute value, the second inequality in Eq. (31) requires $p_D = 0$, which implies $p_I > 0$ by Eq. (30) for convergence to fairness in α-increase. The first inequality in Eq. (31) requires $q_I \geq 1$ to ensure $p_I + (q_I - 1)\alpha_i(k) > 0$ $\forall\ \alpha_i(k) > 0$. Since $\theta = \frac{p_I}{q_I}$ (fairness increases only in α-increase phase) and $\phi(\boldsymbol{\alpha}(k+1)) - \phi(\boldsymbol{\alpha}(k))$ is an increasing function of θ, we let q_I take its minimum $q_I = 1$ which is the *optimal* value for the convergence to the fairness. Thus, we obtain the feasible and optimal linear control function defined by the following constraints:

$$\textbf{constraint:}\{0 < q_D < 1, q_I = 1, p_D = 0, p_I > 0\} \tag{32}$$

which is the exactly what we proposed for the α-control in the transient state, i.e.,

$$\alpha_i(k+1) = \begin{cases} p + \alpha_i(k); & \text{if } BCN(k-1,k) = (0,0) \\ q\alpha(k); & \text{if } BCN(k) = 1 \end{cases} \tag{33}$$

where $p_I = p > 0$, $q_I = 1$, $p_D = 0$, and $0 < q_D = q < 1$.

(II) In Equilibrium State. The linear α-control function is expressed by

$$\alpha_i(k+1) = \begin{cases} \frac{1}{q}\alpha_i(k); & \text{if } BCN(k-1,k) = (1,0), \\ q\alpha_i(k); & \text{if } BCN(k) = 1, \end{cases} \tag{34}$$

Since $p_D = 0, p_I = 0, q_D = q$ $(0 < q < 1)$, and $q_I = \frac{1}{q} > 1$, this control function belongs to case **4** in Eq. (29) where $\theta = 0$. Thus, the fairness is maintained as the α-control enters the equilibrium state. On the other hand, when $p_D = 0$ and $p_I = 0$, the constraints for convergence to efficiency become:

$$\textbf{constraint:}\{0 < q_D < 1, q_I > 1\} \tag{35}$$

which also satisfies Eq. (31) and Eq. (26). Thus, the convergence to efficiency is also maintained for that connection. This completes the proof. □