THIRD QUARTER 2003, VOLUME 5, NO. 1

IEEE COMMUNICATIONS SURVEYS
The Electronic Magazine of Original Peer-Reviewed Survey Articles

www.comsoc.org/pubs/surveys

ADVANCES IN INTERNET CONGESTION CONTROL

SEUNGWAN RYU, CHRISTOPHER RUMP, AND CHUNMING QIAO,
STATE UNIVERSITY OF NEW YORK AT BUFFALO

ABSTRACT

In this survey, we first review the concept of congestion control with a focus on the Transmission Control Protocol/Internet Protocol (TCP/IP). We describe many recently proposed algorithms to combat congestion and improve performance, particularly active queue management (AQM) algorithms such as random early detection (RED) and its variants. We then survey control-theoretic analysis and design of TCP congestion control with an AQM scheme. In addition, we discuss three problems associated with AQM proposals: parameter setting, the insensitivity to the input traffic load variation, and the mismatch between macroscopic and microscopic behavior of queue length dynamics. As alternatives to AQM algorithms, we also survey architectural approaches such as modification of source or network algorithms, and economic approaches including pricing or optimization of allocated resources. Finally, we list many open issues that persist in the design, operation, and control of the Internet.

Internet congestion occurs when the aggregate demand for a resource (e.g., link bandwidth) exceeds the available capacity of the resource. Resulting effects from such congestion include long delays in data delivery, wasted resources due to lost or dropped packets, and even possible congestion collapse [1], in which all communication in the entire network ceases.

It is therefore clear that in order to maintain good network performance, certain mechanisms must be provided to prevent the network from being congested for any significant period of time. Two approaches to handling congestion are *congestion control* (*or recovery*) and *congestion avoidance*. The former is reactive in that congestion control typically comes into play *after* the network is overloaded, i.e., congestion is detected. The latter is proactive in that congestion avoidance comes into play *before* the network becomes overloaded, i.e., when congestion is expected. In general and throughout this article, the term congestion control is used to denote both approaches.

Congestion control involves the design of mechanisms and algorithms to statistically limit the demand-capacity mismatch, or dynamically control traffic sources when such a mismatch occurs. It has been shown that static solutions such as allocating more buffers, providing faster links or faster processors are not effective for congestion control purposes.

Current usage of the Internet is dominated by transmission control protocol (TCP) traffic such as remote terminal (e.g., Telnet), FTP, Web traffic, and electronic mail (e.g., SMTP). These TCP sources constitute 90 percent of all traffic with 50–70 percent of this TCP traffic being short-lived connections in size and lifetime (so called *mice*) [2–4]. Although these applications are rather *elastic* in nature [5], in that they can tolerate either packet delay or packet losses rather gracefully, congestion remains a major problem that leads to poor performance. If the Internet is to evolve to a high-performance network providing ubiquitous services, including real time voice/video, we must understand how congestion arises and find more efficient ways to keep the network operating within its capacity.

In current TCP/IP networks, TCP packet (or segment) loss, indicated by a timeout [1] or a triple duplicated acknowledgment [6], is used as an indication of network congestion. Once congestion occurs, TCP controls its sending rate by limiting its (congestion) window size (`cwnd`). The data-sending rate of TCP (or the window size) is determined by the rate of incoming Acknowledgments (ACKs) to previous packets. The rate of ACK arrival is in turn determined by the presence or absence of congested link(s) along the path between a source and its destination. In steady state, the source's sending rate will match the arrival rate of the ACKs. Accordingly, TCP automatically detects congestion and regulates its sending rate. This has been referred to as TCP's *self-clocking* behavior. Three major TCP implementations are:

- Slow start and congestion avoidance: *TCP Tahoe* [1].
- Fast Retransmit and Fast Recovery: *TCP Reno* [6, 7].
- *TCP Vegas* [8].

Recent proposals for TCP mechanisms are introduced and discussed further in [9].

However, current Internet congestion control methods are expected to result in unsatisfactory performance, e.g., multiple packet losses and low link utilization, as the number of users and the size of the network increases. Accordingly, many congestion control approaches have been proposed. Network algorithms such as active queue management (AQM), executed by network components such as routers, detect network congestion, packet losses, or incipient congestion, and inform traffic sources (either implicitly or explicitly). In response, source algorithms adjust the source's data-sending rate into the network. The basic design issues are *what to feedback* (network algorithms) and *how to react* (source algorithms).

There are excellent tutorials [10, 11] on current Internet congestion control mechanisms for best-effort services. These articles include an extensive introduction of the concepts of many congestion control mechanisms such as congestion avoidance and control, feedback mechanisms, packet scheduling, buffer management, etc. Lefelhocz *et al*. [11] claim that packet scheduling, buffer management, feedback, and source algorithms (i.e., end-system adjustment) are four necessary and sufficient components for providing better best-effort services. They proposed a general design principle: the network should manage and distribute its resources through packet scheduling and buffer management and give the best possible explicit feedback. In response, the source algorithms should implement the adjustments accordingly.

Gevros *et al.* [10] emphasized the role of cooperation among TCP-friendly sources, describing the concept of fair bandwidth sharing as an incentive for cooperation. In this competition among network users, game-theoretic approaches for Internet congestion control are discussed at length. This discussion includes the relationship between network provisioning and network economics. Active queue management (AQM) was briefly introduced as a solution approach for congestion avoidance with a focus on the random early detection (RED) algorithm.

In this article, we more fully survey the area of active queue management. We describe many recently proposed algorithms to combat congestion and improve performance, particularly AQM algorithms such as RED and its variants. We also survey control-theoretic analysis and design of TCP congestion control with an AQM scheme. In addition, we discuss three problems associated with AQM proposals:

- Parameter setting.
- The insensitivity to the input traffic load variation.
- The mismatch between macroscopic and microscopic behavior of queue length dynamics.

As alternatives to AQM algorithms, we also survey architectural approaches such as modification of source and/or network algorithms, and economic approaches including pricing or optimization of allocated resources.

In this article, we focus on congestion control for *unicast best-effort* traffic/services. Other topics such as multicast, differentiated services (DiffServ), quality of service (QoS), packet scheduling, and multiprotocol label switching (MPLS) are beyond the scope of this article. There are many open issues related to the design, implementation, and evaluation of congestion control that may occur under various network environments. Some of these issues include interoperability, robustness, stability, convergence, implementation complexity, fairness (both in terms of bandwidth sharing and packet marking), TCP-friendliness, link characteristics, and assumptions on network traffic dynamics.

This article is organized as follows. First, we describe existing AQM algorithms including RED, and address several problems associated with their design and implementation. In addition, we introduce control-theoretic design and analysis of Internet congestion control, especially for the TCP/AQM dynamics. We then discuss other emerging Internet congestion control approaches. Next, we then list open issues in current and future Internet congestion control. Finally, we summarize our conclusions.

Active Queue Management

Traditional voice traffic has been believed to be modeled after a governing law such as the Poisson nature of call arrivals in which, as the time scale increases and traffic is heavily aggregated, the traffic smoothes out and becomes quite tame [12, 13]. Conversely, data traffic such as FTP, Web traffic, and video traffic is known to be *bursty* and *self-similar* [12, 14, 15]. This bursty and self-similar traffic pattern has been observed in data networks of all scales, e.g., Ethernet [14], WAN [15], and Internet [12].

In order to cope with the bursty traffic effectively, intelligent congestion control mechanisms for FIFO-based queue management [9, 16], per-flow queue management [10, 11, 17], and scheduling mechanisms [10, 11] are required at routers. In this article, we consider only (FIFO-based) queue management. The traditional queue management technique at a router is *tail drop* (*TD*), which sets a maximum queue length in terms of the number of packets for each queue. This technique has two important drawbacks: lock out and full queues [16]. The *lock out* phenomenon may occur when tail drop allows a few connections to monopolize queue space. Since the router sends back congestion signals to sources by means of packet losses only when the buffer has become full, a *full queue* phenomenon may persist for a long time under tail drop queue management. Therefore, it is necessary for a router to maintain a small queue length with enough buffer capacity to absorb the bursty data traffic. In addition, it is necessary to detect congestion before it becomes a problem (i.e., before overflows) in order to control congestion efficiently and keep the network stable.

One possible solution to overcome the drawbacks of the tail drop scheme is to drop packets before a queue becomes full so that a source can respond to congestion before buffers overflow. This approach is called *active queue management* (*AQM*) [16], and random early detection (RED) [18] is an example of this approach. The Internet Engineering Task Force (IETF) [19] is considering whether to deploy such active queue management algorithms [16].

Random Early Detection (RED) and its Variants

TCP Tahoe, TCP Reno, and TCP Vegas detect congestion only after a buffer at an intermediate router has already overloaded and packets have been lost. *Random early detection* (*RED*) [18] attempts to avoid congestion by providing a warning to sources when congestion starts to develop. In other words, a major difference between RED and the TCP congestion control schemes is that RED employs a network algorithm to detect *incipient* congestion. RED's design objectives are to minimize packet loss and queuing delay, maintain high link utilization, and remove biases against bursty sources [18]. It is designed to also avoid global synchronization, which occurs when all sources detect congestion and reduce their sending rates at the same time, resulting in a fluctuation of link utilization. RED achieves these goals by introducing an enhanced control mechanism involving randomized packet dropping and queue length averaging.

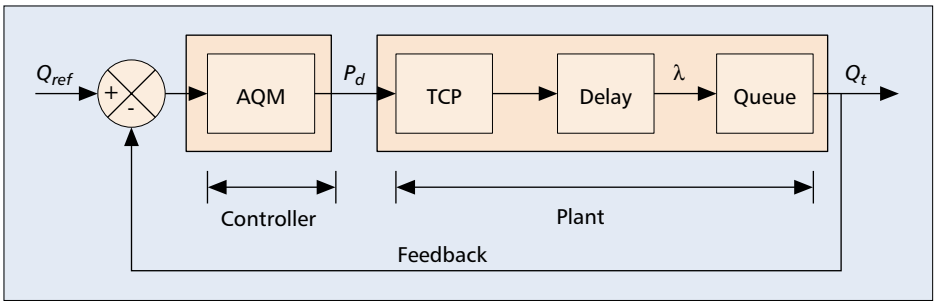

FIGURE 1. *Feedback control modeling of TCP congestion control with an AQM algorithm.*

With RED, a link maintains an exponentially weighted moving average (EWMA) queue length,

$$avg = (1 - w_Q) \times avg + w_Q \times Q, \quad (1)$$

where Q is the current queue length and w_Q is a weight parameter, $0 \le w_Q \le 1$. When avg is less than the minimum threshold (min_{th}), no packets are dropped (or marked). When it exceeds the maximum threshold (max_{th}), all incoming packets are dropped. When it is in between, a packet is dropped with a probability p_a that is an increasing function of avg. More specifically, if $min_{th} \le avg \le max_{th}$, then the temporary dropping (or marking) probability, p_b, is calculated as:

$$p_b = max_p \times \frac{avg - min_{th}}{max_{th} - min_{th}}, \quad (2)$$

where max_p is the maximum value of p_b. Then p_a is calculated as:

$$p_a = p_b \times \frac{1}{1 - count \times p_b}, \quad (3)$$

where *count* is the number of undropped packets since the last dropped packet. In this way RED drops (or marks) packets in proportion to the input rates of the connections. Connections with higher input rates receive proportionally more drops (or marks) of packets than connections with lower input rates. By doing so, RED tries to maintain equal rate allocation and remove biases against bursty connections. By using probabilistic packet dropping RED also eliminates global synchronization.

Since RED was first proposed in 1993, many AQM-based approaches such as Adaptive-RED (ARED) [20], Dynamic-RED (DRED) [21], Stabilized-RED (SRED) [22], BLUE [23], and adaptive virtual queue (AVQ) [24] have been proposed, and their performance has been evaluated [2, 25–27].

ARED [20] attempts to maintain suitable operating parameters in RED by dynamically adjusting max_p in Eq. 2 based on observed queue length dynamics. ARED increases max_p when avg exceeds max_{th} and decreases max_p when avg goes below min_{th}.

DRED [21] attempts to maintain the EWMA queue length close to a desired queue length, Q_{ref}, to stabilize the utilization around a pre-defined level. DRED adjusts the packet drop probability based on the deviation of the queue length from Q_{ref}.

SRED [22] drops packets with a load-dependent probability based on an estimated number of flows and the instantaneous queue length. SRED estimates the number of flows without maintaining a per-flow account. SRED stabilizes the buffer utilization at a level independent of the load level.

BLUE [23] uses packet loss and link-idle events rather than the queue length to control congestion. BLUE increases the packet drop probability in response to a buffer overflow (i.e., a packet drop) and decreases the packet drop probability when the link becomes idle.

AVQ [24] uses a modified token bucket model as a virtual queue (VQ) to regulate buffer utilization rather than the queue length. AVQ adjusts the size and link capacity of the VQ proportional to the measured input rate and drops packets when the VQ overflows.

In addition, since RED is an AQM algorithm focused on TCP/IP-based best-effort services, some variants also have been proposed and evaluated for differentiated services [28], multimedia (UDP-like) [29], and ATM traffic control [30].

CONTROL-THEORETIC ANALYSIS AND DESIGN

Recently, some AQM algorithms have been proposed based on control-theoretic analysis and design. In these approaches, the TCP/AQM flow dynamics are modeled and analyzed in terms of feedback control theory. Then, using control theory, AQM algorithms are designed to increase the speed of response (the short-term performance) and improve the stability and robustness (the long-term performance) of TCP/AQM congestion control. These goals can be achieved by regulating the queue length to agree with a desired value (or by eliminating the difference between the queue length and the desired value).

TCP congestion control dynamics with an AQM scheme can be modeled as a feedback control system (Fig. 1). In this model, the feedback control system consists of:

- A *plant,* which represents a combination of subsystems such as TCP sources, routers, and TCP receivers that send, process, and receive TCP packets respectively.
- An *AQM controller*, which controls the packet arrival rate to the router queue by generating a packet drop probability (P_d) as a control signal.
- The queue length at a router as a plant variable (i.e., a *controlled variable*) denoted by Q_t.
- A desired queue length at a router (i.e., the reference input) denoted by Q_{ref}.
- A *feedback signal*, which is a sampled system output (i.e., queue length) used to obtain the error term, $e(t) = Q_{ref} - Q_t$.

Mascolo [31] used classical control theory and *Smith's principle* to design an effective and simple congestion control law for high-speed data networks. They proposed a rate control equation that guarantees stability of the network queues and full utilization of network links in a general network topology. Then, a digitized control law was developed and applied to ATM networks with ATM available bit rate (ABR) flow control. In addition, the control law was transformed to a window form, and revealed that the current TCP/IP implements the Smith predictor for congestion control. Finally, they identified drawbacks of the current window-based TCP congestion control such as queue length oscillation and link under-utilization.

Firoiu *et al.* [25] modeled the RED algorithm as a feedback control system and derived fundamental control laws governing the traffic dynamics in TCP/IP networks. Then, they recommended rules for the configuration of the RED control law such as a drop-conservative policy and a delay-conservative policy. In addition, they derived a set of recommendations for configuration of RED queue length averaging mechanisms such as the sampling frequency and the averaging weight (w_Q).

Misra *et al.* [27] developed a system of nonlinear differen-

tial equations for TCP/AQM dynamics using fluid-flow analysis that ignores the TCP time-out mechanism and slow-start phase. A linearized TCP/AQM dynamic model was developed and analyzed especially for TCP/RED dynamics in terms of (feedback) control [32]. The forward-path transfer function of the plant, $P(s) = P_{tcp}(s) \cdot P_{queue}(s) \cdot e^{-sR0}$, was given by

$$P(s) = \left(\frac{\frac{R_0 C^2}{2N^2}}{S + \frac{2N}{R_0^2 C}} \right) \cdot \left(\frac{\frac{N}{R_0}}{S + \frac{1}{R_0}} \right) \cdot e^{-sR_0}, \quad (4)$$

where N is a load factor (number of TCP connections), R_0 is a round trip time, C is the link capacity, and e^{-sR0} is the time delay.

Since the forward-path transfer function of TCP flows, i.e., $P(s)$, has two non-zero poles (where $s \neq 0$), it is a type 0 system [33]. Thus there always exists constant steady-state error, K_P, for the step-function input. Since $\lim_{s \to 0} P(s) = K_P = (R_0 C)^3/(4N^2)$, the steady-state error of the forward-path transfer function is $e_{ss} = \lim_{s \to 0} R/(1 + P(s)) = R/(1 + K_P)$, where R is the magnitude of a reference step-function input.

RED attempts to eliminate the steady-state error by introducing the EWMA queue length (or equivalently EWMA error terms [21]) as an integral (I)-control to the forward-path transfer function. However, since RED introduces a range of reference input values, i.e., any queue length between two thresholds, min_{th} and max_{th}, rather than a constant reference input for I-control, the TCP/RED model shows oscillatory system dynamics. Moreover, the very small exponential smoothing weight factor ($w_Q = 1/512 \cong 0.002$) recommended in [18] for application to the current queue length in the EWMA calculation of Eq. 1 creates a large integral time in I-control, and may be accompanied by a large overshoot [33]. As a result, RED shows oscillatory queue length dynamics and gives poor performance under a wide range of traffic environments. Furthermore, RED shows sluggish response to the traffic dynamics.

If a constant reference input is introduced to an AQM controller, the steady-state error of the TCP dynamics can be eliminated. To this end, the proportional-integral (PI)-controller [34] introduces a desired queue length, Q_{ref}. Using the linearized TCP/AQM dynamics, the PI-controller has been proposed not only to improve responsiveness of the TCP/AQM dynamics but also to stabilize the router queue length around Q_{ref}. The latter can be achieved by means of integral (I)-control, while the former can be achieved by means of proportional (P)-control using the instantaneous queue length rather than using the EWMA queue length. The resulting PI-controller is capable of eliminating the steady-state error regardless of the load level.

Since currently proposed AQM algorithms only use the (average) queue length to measure the severity of congestion, the congestion detection and control of these algorithms are *reactive* to current or past congestion, not *proactive* to incipient congestion. To address this problem, we have proposed *Pro-Active Queue Management* (*PAQM*) [35], which attempts to predict incipient congestion using recent input traffic history. A classical Proportional-Integral-Derivative (PID) feedback control is used in designing PAQM not only to have an anticipatory congestion detection and control capability but also to achieve long-term control performance such as acceptable queue length behavior (or equivalently delay), acceptable packet loss rates, or high link utilization. As a result, the PAQM controller can predict and reduce upcoming error (i.e., the incipient congestion) using Derivative (D)-control as well as eliminate steady state error by means of PI-control.

A classification of representative AQM algorithms and their control characteristics in terms of the control-theory is shown in Table 1.

Problems with Existing AQM Proposals

In order to control bursty Internet traffic effectively and efficiently, most AQM proposals recommend using the (average) queue length as a congestion indicator. However, there remain several critical problems:

- Mismatch between macroscopic and microscopic behavior of queue length.

AQMs	Control types	Control characteristics
Tail drop	On-off	Simple to implement. Plant output (i.e., the queue length) can fluctuate.
BLUE [23]	On-off	A modified on-off control. Dynamically adjusts the packet drop probability by a small amount: •Increase when a packet loss occurs. •Decrease when the link becomes idle.
RED [18]	I	Through EWMA queue length (I-control): •Can eliminate the steady-state error. •Could cause sluggish response. Reference input range [min_{th}, max_{th}] can cause fluctuated plant output.
DRED [21]	I	Uses EWMA as an I-control. Uses a constant reference input (a target queue length) (Q_{ref}).
AVQ [24]	P	A modified token bucket model: •Maintains a virtual queue (VQ). •Adjusts proportional gain (link. capacity) of VQ based on the measured input rate.
PI-controller [34]	PI	Uses a constant reference input (a desired queue length) (Q_{ref}). P-control for faster response: uses the instantaneous queue length (Q_t). I-control for elimination of the steady-state error, $e(t) = Q_{ref} - Q_t$.
PAQM [35]	PID	PI-control for faster response and steady-state error elimination. D-control for anticipatory congestion avoidance and control. Control law: based on current and predicted congestion measure.

Table 1. *A classification of AQM algorithms in terms of control-theoretic design and analysis.*

- Insensitivity to the input traffic load variation.
- Configuration or parameters-setting problem.

Mismatch Between Macroscopic and Microscopic Behavior of Queue Length: AQM algorithms try to avoid congestion, stabilize queue dynamics, and maintain low end-to-end delay by controlling the (exponentially weighted moving) average queue length at a router. We call the (stable) dynamics of the *average* queue length the *macroscopic* behavior of a router because it focuses on the overall and long-term behavior of a router. Conversely, we call the short-term dynamics of the *actual* queue length the *microscopic* behavior of a router.

Many studies have shown different queue length dynamics between the actual queue and the average queue [2, 22, 26]. For example, if many bursts of data arrive at a RED router, the actual queue length will increase rapidly and eventually overflow the buffer. But because of the small weight ($w_Q \simeq 0.002$) assigned to the actual queue length (Q) in Eq. 1, the average queue length (avg) will increase only slightly during the buildup of congestion. After a while, sources will detect congestion by the timeout caused by packet drops at the router and reduce their sending rate. Consequently, the actual queue length at the congested router will be decreased back to normal or even below min_{th}. However, if the router experiences peak traffic for some time, the average queue length will stay high. In that case, the router will continue to drop packets after the congestion, unfairly penalizing new, innocent packets.

This phenomenon is caused by the mismatch between *microscopic* behavior of the actual queue length and the *macroscopic* design goals of using the average queue length with a small weight for smoothing the total traffic. This phenomenon is also caused by the mismatch between the *router centric* goal of controlling *aggregate* input traffic and the *user centric* goals of maximizing individual goodput or minimizing round-trip time (RTT). As a result, a router tends to send congestion indication back to sources after congestion occurs [36], or even after multiple packet losses occur.

Insensitivity to the Input Traffic Load Variation: Another related drawback of currently proposed AQM methods is that their congestion avoidance algorithms depend only on the current queue status. In other words, those algorithms only take the buffer utilization into account as a measure of the severity of congestion. For example, in RED [18], since the router uses only the average queue length, avg, as a congestion indicator and the average queue length is insensitive to the input traffic load variation, the router cannot detect incipient congestion effectively. Moreover, this *insensitivity* of the congestion indicator to the input traffic load variation causes a fairness problem of packet dropping among connections.

May *et al.* [26] show that Gentle RED (GRED) [37] with the instantaneous (or actual) queue length, called GRED-I, outperforms other queue management algorithms (i.e., TD, RED, GRED) in a realistic experimental setting. However, since this algorithm tries to avoid and control congestion using only the current (instantaneous) queue length, it is also insensitive to the input traffic load variation, so that it cannot detect incipient congestion effectively.

The main reason for this insensitivity of proposed AQM algorithms to the input traffic load variation is the fact that most AQM proposals use the (average) queue length as a congestion indicator even though the window size and the packet drop probability are a function of the input traffic load. The idea behind using only the queue length as a congestion indicator in AQM proposals is that the queue length can fully represent the input traffic load, which has been supported and verified by simulation studies. However, most of these simulation studies assumed idealized traffic, which differs significantly from real IP traffic [2, 3]. For example, most simulations have been performed assuming persistent (long-lasting) connections, constant RTTs, and a limited number of connections. This traffic environment provides only quite tame traffic conditions, and is very different from the bursty traffic nature and resulting traffic load fluctuations in real IP networks. Therefore, under realistic traffic environments, proposed AQMs often do not outperform TD [2, 26, 38].

Configuration Problem: As a result of the above two related problems, AQM (particularly RED) parameter configuration is a difficult design task. Many modified AQMs [20, 22, 23, 39, 40] have been proposed, in which they have shown better performance by means of analytic modeling and simulation study. However, each modified AQM proposal is good only for some particular traffic conditions, neither for realistic IP traffic [2] nor a heterogeneous traffic environment [3]. We conjecture that the parameter configuration problem mainly stems from the gap between the design goals of AQM and the characteristics of real IP traffic. When RED was proposed in 1993, the (exponentially weighted) average queue length was proposed for smoothing out excessive input traffic. Most subsequent AQM proposals have adopted this queue length averaging method. However, 90 percent of Internet traffic today is TCP traffic, and 50–70 percent of TCP traffic is short-lived Web-like *mice* traffic [2], which did not exist in 1993 [41]. Also, instead of the Mb/s T1 links, prevalent at the time of RED development, high speed WANs use links operating at hundreds of Mb/s or Gb/s.

To examine the effect of the mice traffic on the performance of RED, we evaluate the queue length dynamics of RED with two different traffic flows. One consists of FTP traffic only, and the other consists of traffic where only 1/3 of the flows are FTP flows and the remaining 2/3 of the flows are Web-like mice flows. Default RED parameters, $min_{th} = 70$, $max_{th} = 200$, $max_p = 0.1$, and $w_Q = 0.002$, are used. Figure 2 shows the queue length dynamics over time in each case under a total of 135 TCP flows. Figure 3 shows the mean and the standard deviation of the queue length as a function of the number of TCP flows with and without mice flows. As shown in Figures 2 and 3, although RED maintains the same mean queue length in both cases, it shows severe fluctuating queue length dynamics with the presence of mice flows.

Therefore, we need a new congestion indicator and a control function for AQM that provides adaptive control to the traffic characteristics such as the amount of traffic, fluctuation of traffic load, and the traffic nature (whether long-lived or short-lived).

Other Emerging Congestion Control Approaches

The prevalent paradigm used in the design and operation of the current Internet is "keep it simple." The design principle of TCP is "do not ask the network to do what you can do yourself" [42]. However, as the number of users and the size of the Internet increases, more packet losses and other performance degradations are experienced. Moreover, the Internet is evolving from providing single best-effort service to a variety of services including real-time service. Therefore, many algorithms have been proposed recently to con-

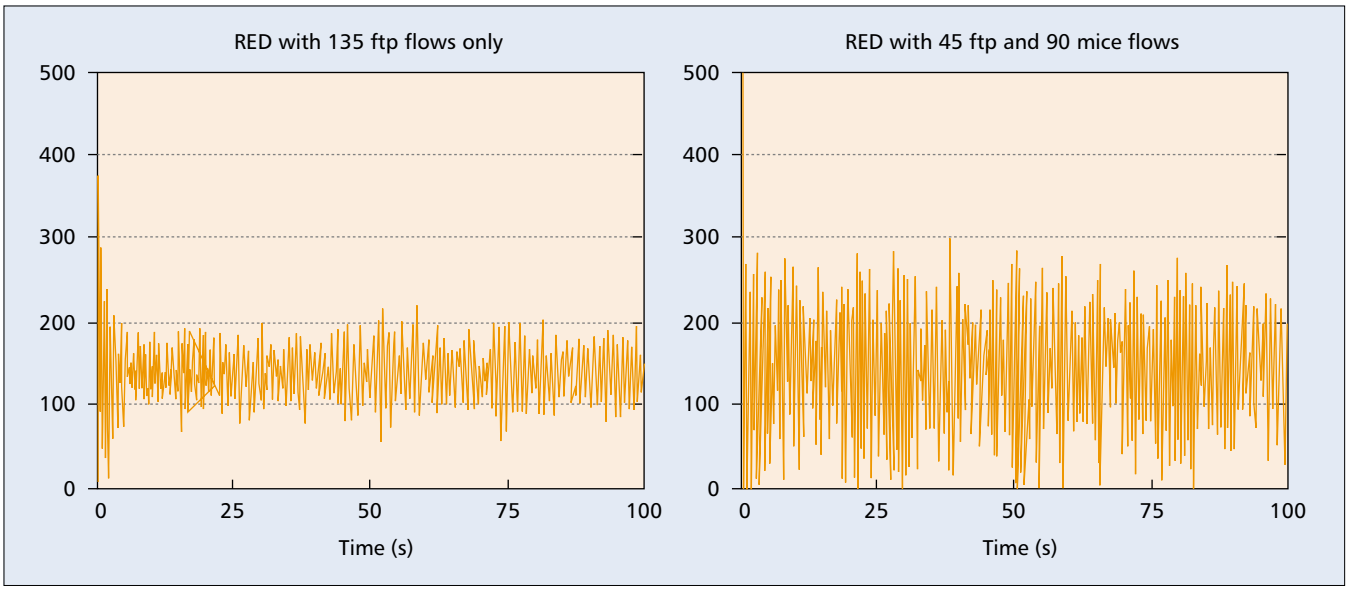

FIGURE 2. *Queue length dynamics of RED: (a) FTP flows only (left), (b) 33 percent FTP and 67 percent mice flows.*

trol congestion efficiently and to cope with the evolution of Internet services.

Architectural approaches refer to approaches that modify the source or/and network algorithms to provide better congestion control while keeping the paradigm of the current Internet and the design principles of TCP. Source algorithm modification uses only implicit congestion information (i.e., delay) at the source without any assistance from the network. On the other hand, network algorithm modification uses explicit congestion information to deliver better congestion indication to the source. Each of these modifications can be implemented alone. However, cooperation among source algorithms and with network components is very important to achieve efficient network resource allocation as well as effective congestion control [10].

Economic approaches have been proposed that introduce concepts of network economics using mathematical modeling and analysis. Pricing-based approaches use price (per packet or per bandwidth) as a device for congestion control and even for service differentiation. Optimization-based approaches use either mathematical programming or game-theoretic techniques to analyze and optimize the behavior of the network or the users. However, these economic approaches may require breaking the paradigm of the current Internet and/or the paradigm of TCP.

ARCHITECTURAL APPROACHES

The main function of congestion control in the current Internet is performed at the transport layer. Here, only the end systems (i.e., sources) that want to implement different, complex, or efficient functions need to be upgraded. Thus, source algorithm modification can be implemented easily with current end-to-end Internet congestion control without any modification of network mechanisms.

Mo *et al.* [43] demonstrated the existence of fair end-to-end window-based congestion control protocols based on a multiclass fluid model with first-come-first-serve (FCFS) routers and links with infinite buffer space. In this model, sources receive implicit feedback, such as estimated RTT or throughput, but no explicit congestion indications from the network. They generalized proportional fairness[1] [44] and

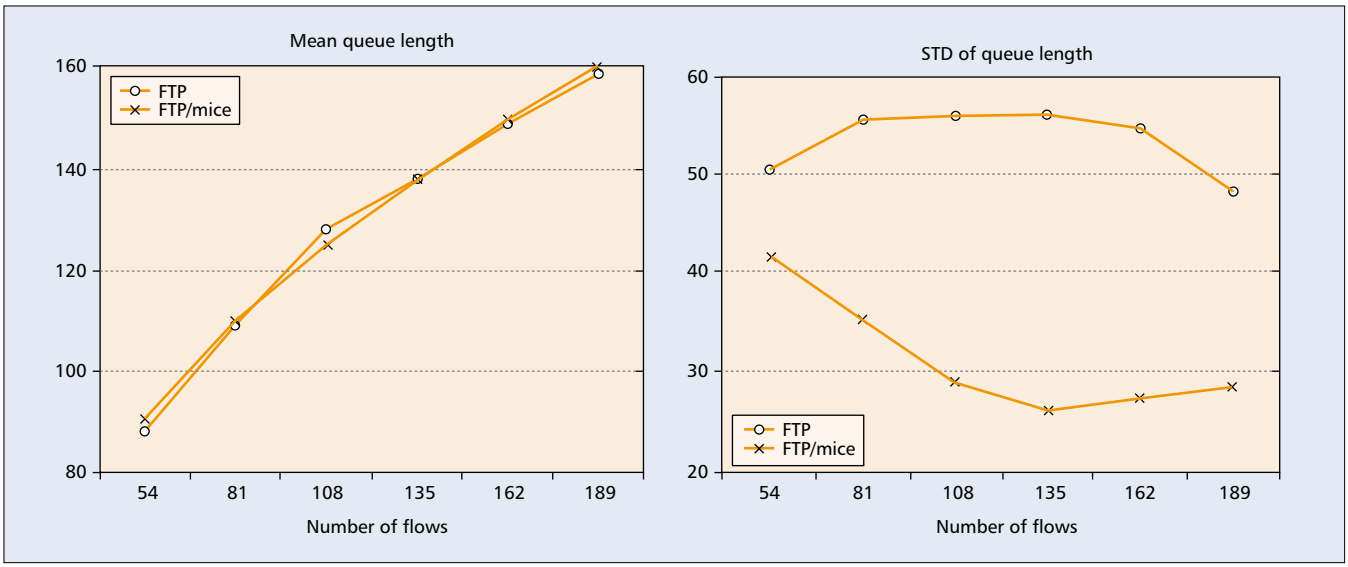

FIGURE 3. *Mean and standard deviation of the queue length of RED with and without mice flows.*

Directions	Where the source gets information and how to react? Consistent to the paradigm of the current Internet. Compatible with the current-TCP based control method.	
Approaches	**Design concepts**	**Solution methods**
Source algorithm	Do not ask the network to do. Keep it simple. Support best-effort services.	Modify source algorithm. Use implicit congestion feedback.
Network algorithm	Ask the network to do. Comparatively simple. May support variety of services.	Modify network algorithm. Use explicit congestion feedback. Cooperation with source algorithm is recommended.

Table 2. *A summary of directions, design concepts, and solution methods of architectural approaches.*

max-min fairness[2] [45], and propose a window adjustment scheme that converges to the solution.

Mitra *et al.* [46] proposed a distributed algorithm for end-to-end calculation of window sizes called *dynamic adaptive windows* (*DAW*) using an analysis of closed queuing networks. Unlike TCP congestion control, the DAW algorithm updates the window sizes based on packet-delay measurements, rather than loss observations. The end-to-end packet delays could be used to detect congestion if the propagation component of the delay was known and if some idea regarding the acceptable range of queuing delays existed.

Massoulié *et al.* [47] propose a *potential delay maximization* criteria, which is interpreted in terms of the delay experienced by ongoing transfers. They proved that fixed-size window control can achieve fair bandwidth sharing according to any fairness criteria, provided that the scheduling at each link is performed in an appropriate manner.

The end-to-end congestion control of the current Internet is appropriate for pure best-effort data carried by TCP that has little or no sensitivity to delay or loss of individual packets. However, current Internet mechanisms become inadequate with the growth of interactive traffic such as Telnet, Web browsing, and the transfer of audio and video data that is sensitive to packet loss or latency. Thus, network algorithms such as AQM at a router need to be modified not only to provide better congestion information to sources, but also to detect and/or control congestion quickly. However, in the current Internet environment, AQM is restricted to using packet drops as a mechanism for congestion indication.

Ramakrishnan *et al.* [48] extended the explicit congestion notification (ECN) algorithm [49] to IP. When a router detects congestion before the queue overflows, it signals congestion via *marking* packets rather than dropping them. This algorithm would require an ECN field in the IP header with two bits: one bit for the indication of ECN capability and the other bit for congestion indication. By combining explicit notification with AQM, the performance of both delay-sensitive (telnet-like) and delay-insensitive (FTP-like) traffic can be improved.

Kalampoukas *et al.* [50] proposed an explicit feedback scheme, called *explicit window adaptation* (*EWA*), in an internetwork consisting of both rate-controlled and non-rate-controlled subnetworks. EWA is based on modifying the receiver's advertised window in TCP ACKs returning to the source. The window size indicated to TCP is a function of the available space in the buffer at the edge routers. A mismatch between the TCP window and the bandwidth-delay product of the backbone network will result in accumulation of large queues at the edge of the rate-controlled backbone network.

A summary of directions, design concepts, and solution methods of architectural approaches are shown in Table 2.

Economic Approaches

Pricing-Based Approaches: Economists have developed ways to model the problem of individuals competing for limited resources (e.g., bandwidth) by treating *prices* as a mechanism for directing consumption [51]. The difference from standard economic theory is that the technological infrastructure of the Internet may allow a breakthrough in the area of online distributed accounting. The breakthrough is the ability to charge users in a way that precisely reflects their actions using only a simple pricing mechanism on each packet.

MacKie-Mason *et al.* [52] have described a *smart market* approach to allocating resources in a network, where a price (or a bid) is set for each packet depending on the level of the demand for bandwidth. The network admits packets with bid prices that exceed the current cut-off amount. This cut-off is determined by the marginal congestion costs imposed by the next additional packet. Rejected packets are bounced back to the users or may be routed to a slower link.

Shenker *et al.* [53] argued against pricing structures solely based on marginal congestion costs. They argued that marginal costs might not be sufficient to recover revenue and that congestion costs are difficult to compute. Their proposal was to focus more on structural and architectural issues. In this context, they proposed an *edge-pricing scheme*, which describes the place at which charges are assessed. These charges could be either per-packet or per-byte for purchasing a certain amount of capacity from the network.

Odlyzko [54] has suggested the Paris Metro Pricing (PMP) scheme for providing differentiated services in the Internet. In this approach, the network is partitioned into several logically separate channels, differing only in the prices paid for using them. The idea is that with less traffic in channels with higher price, better quality of service would be provided.

Optimization-Based Approaches: Optimization-based approaches employ *mathematical programming* or *game theory* to solve the congestion control problem. In these approaches, we can formulate congestion control as an analytic model, and thereby obtain steady-state operating points. These approaches can also be used to analyze the dynamics of the network in a decentralized control environment. We can also find guidelines for the modification (or improvement) of current congestion control methods as well as the design and operation of future congestion control algorithms.

The network resource allocation problem (e.g., congestion control) has two components: users and the network. Users can be distinguished by their traffic characteristics represented

[1] *A set of rates* $\{x_s, s \in S\}$ *is* proportional fair *if it is feasible, that is the rate is nonnegative and the aggregate rate is not greater than a link capacity, and if for any other feasible set of rates,* $\{x_{s^*} \in R\}$*, the aggregate of proportional change is zero or negative, where* S *is a set of possible user paths from sources to destinations and* R *is a set of possible user rates.*

[2] *A set of rates is* max-min fair *if no rate may be increased without simultaneously decreasing another rate which is already smaller [45]. More specifically, a vector* $x = \{x_s, s \in S\}$ *is max-min fair if it is feasible, and if for each* $s \in S$, x_s *cannot be increased (while maintaining feasibility) without decreasing* x_{s^*} *for some* s^* *for which* $x_{s^*} \le x_s$.

by a *utility function* [5]. A user wants to maximize their benefit, utility minus bandwidth cost, through congestion control. A network consists of a set of links with capacity constraints. The network resources are shared by a set of users. The network resource allocation problem then becomes a nonlinear programming (NLP) problem with linear constraints [44, 55, 56]. The objective is to choose source rates so as to maximize the aggregate utility subject to the constraint that the total source rate at any link does not exceed the link capacity. Through this formulation, we can achieve a *Pareto-efficient*[3] or global optimal set of rates for users.

Directions	Which method can be used for better control? May introduce new paradigm for design and operation. Cooperation between sources and network is essential. Can support variety of services.	
Approaches	**Design concepts**	**Solution methods**
Pricing	Use price per packet (or bandwidth) as an incentive.	Needs a breakthrough in the area of online distributed accounting.
Math programming/ game theory	Formulate as an analytic model: •Network resource allocation model. •Multi-player game. Find global optimal or Pareto efficiency.	Use utility function. Rate-based bandwidth allocation.

Table 3. *A summary of directions, design concepts, and solution methods of economic approaches.*

A Nonlinear Programming Formulation: Consider a network that consists of a set $\mathcal{J} = \{1, \ldots, J\}$ of links with capacities $\mathbf{C} = \{C_1, \ldots, C_J\}$. The network is shared by a set $\mathcal{S} = \{1, \ldots, S\}$ of sources. A source s is characterized by two parameters $(J(s), U_s)$ where the path $J(s) \subseteq \mathcal{J}$ is a set of links that source s uses, and $U_s(x_s) \in \mathcal{U}$ is a utility function of source s when it transmits data at rate $x_s \geq 0$ [5]. Let A be the routing matrix where $A_{js} = 1$ if $j \in J(s)$ and zero otherwise. Then the nonlinear programming formulation is:

$$SYSTEM(\mathcal{U}, \mathbf{A}, \mathbf{C}) \quad \max_{x_s \geq 0} \sum_{s \in S} U_s(x_s) \quad \text{subject to } \mathbf{A}x \leq \mathbf{C} \tag{5}$$

The key premise of the NLP formulation is that sources with different valuation of bandwidth should react differently to network congestion. Using this concept, we can convert congestion control to a rate-based optimization problem. The resulting rate control algorithm can be implemented easily by means of the relation of the window size, the sending rate, and the round trip delay [55]:

$$W_s = x_s \times T_s = x_s \times (D_s + d_s), \tag{6}$$

where for user $s \in S$, W_s is the window size, x_s is the sending rate, T_s is the round trip delay, D_s is total processing delay (queuing delay and data processing time at a router) at links on its path, and d_s is a total propagation delay.

Golestani *et al.* [55] formulated end-to-end congestion control as a global optimization problem and proposed a class of *minimum cost flow control* (*MCFC*) algorithms for adjusting sending rates or window sizes. They reformulated Eq. 5 by absorbing the link capacity constraint into the objective function as a cost of violating the constraint. Then, MCFC adjusts the source rate to equalize marginal utility with the marginal cost of the path.

Kelly [44] proposed a congestion control algorithm that decomposed Eq. 5 into a user subproblem and a network subproblem. Given the price for a unit of bandwidth, the user subproblem chooses a willingness-to-pay in order to maximize its benefit. Then given a set of willingness-to-pays, the network subproblem involves choosing source rates in order to maximize revenue. The network subproblem determines the users' rates according to a *proportional fairness* criterion applied to the rate per unit charge.

Low [56] proposed a different solution procedure for solving Eq. 5, in which network links and sources are considered as processors of a distributed computation system, and solved the dual problem of Eq. 5 using a gradient projection method. In this approach, sources select data-sending rates to maximize their own benefits, and the network links adjust bandwidth prices to coordinate the sources' decisions. Additionally, an implementation algorithm, *random early marking* (*REM*), was proposed via probabilistic packet marking using the proposed ECN bit in an IP header [48, 49].

Since the data-sending rate is adapted by each user individually based on congestion feedback, the congestion control problem can also be formulated as a *multi-player game* [57]. Most often this is viewed as a non-cooperative game, where each player (or source) selfishly maximizes their own utility, $U_s(x_s)$. Many game-theoretic approaches [57–59] have been proposed to examine rate-based decentralized flow control[4] in which congestion feedback is localized to a single link. In this setting, a user (or a source) regulates its sending rate synchronously (in sequence) or asynchronously in order to maximize network power (the ratio of the throughput to the delay) based on feedback information about the other users' aggregate sending rate as measured by the delay. In game-theoretic models, the resulting *Nash equilibrium*[5] is less efficient than the more desirable *Pareto-efficient* set of flows. In the non-cooperative setting, one approach to drive the users toward a Pareto-efficient solution is to use pricing as an incentive.

A summary of directions, design concepts, and solution methods of economic approaches are shown in Table 3.

Open Issues

There still persist many open issues related to the design, operation, and evaluation of the Internet congestion control mechanisms. The major open issues are interoperability, robustness, stability, convergence, implementation complexity, and fairness. Since congestion occurs when the network is overloaded and is closely related to the network traffic environments, there are related open issues that should be taken

[3] *An allocation of goods and prices said to be* Pareto efficient *if there is no charge that would simultaneously benefit someone and harm no one, as measured by their utility functions [51].*

[4] *They use* flow control *for the same meaning of congestion control in this article, not for the hop-by-hop flow control provided by the data link layer entities.*

[5] *A* Nash equilibrium *is a point at which no player (e.g., sources) has incentive to deviate unilaterally.*

into account for solving congestion control. These related issues are link characteristics, TCP-friendliness vs. user datagram protocol (UDP) traffics, and assumptions on network traffic dynamics.

Interoperability and Robustness

Since congestion control methods each have their own mechanism, all the sources and network entities do not follow the same congestion control. For example, UDP applications are usually unresponsive to the dominating TCP congestion control. Thus, the congestion control given by a router or a gateway may not affect all sources. Floyd *et al.* [37] note that an increasing deployment of traffic lacking end-to-end congestion control may cause congested links to occupy themselves sending packets that will only be dropped later in the network. They describe how this leads to *congestion collapse* [1] in the network. Therefore, the heterogeneity of congestion control mechanisms and the lack of end-to-end congestion control may result in an *interoperability* problem. This problem may also cause inefficient performance when all the different algorithms have conflicting optimization goals such as power and (net) benefit, etc.

It is also necessary to maintain TCP's *robust* nature in the presence of packet losses and a wide range of offered load, link speeds, packet sizes, and congestion levels [60]. A robust algorithm will be insensitive to the various traffic environments. Developing a new congestion control mechanism or improving an existing congestion control mechanism has to be done without affecting the robustness of the congestion control algorithms for the network as a whole.

Stability, Convergence and Implementation Complexity

Two concepts of stability exist in Internet congestion control. First, Internet congestion control may have asymptotic convergence to a fixed operating point. Second, it may have controlled oscillation points, i.e., limited cycles with bounded variations. In both cases, the objective of a congestion control algorithm is to achieve steady-state performance in terms of link utilization, throughput, and RTT.

The variance and speed of *convergence* of a congestion control algorithm to a stable operating point (either fixed or oscillatory) depends on the delays between resources (e.g., congested links) and sources [37]. In this case the action delay[6] [61] has to be considered if it is large and uncertain.

Since one of the goals of congestion control is increasing the speed and nature of convergence to the stable operating point, an algorithm must also have a reasonable *implementation complexity*, not only in terms of the number of operations required to implement, but also the amount of information to be maintained.

Fairness

Two fairness issues may occur in Internet congestion control: the *fair bandwidth sharing* [62] between competing connections and the *fair marking* (or dropping) [36] of packets when congestion occurs. The fair sharing of bandwidth among connections depends on the fact that all connections are running basically the same congestion avoidance algorithms, conforming with the current TCP specification [7]. However, fair marking has not been studied actively.

The issue of fair bandwidth sharing among competing TCP connections has become increasingly important for several reasons. One reason is due to the growth of individual TCPs that can use high bandwidth even over high-propagation-delay paths. Second, with the growth of the Web, Internet users want high-bandwidth and low-delay communications. In addition, the growth of best-effort traffic that does not use TCP, i.e., non-TCP-friendly connections [37], also raises fair bandwidth sharing problems between competing best-effort traffic in times of congestion.

A bandwidth allocation scheme is *fair* if it does not offer different treatment to connections, either based on the time order in which they request a share of the available bandwidth, or on the particular location of their source and destination points [63]. A set of rates is *max-min fair* if no rate may be increased without simultaneously decreasing another rate which is already smaller [45]. In a network with a single congested link, max-min fairness implies an equal share of the bandwidth for each connection through it. To achieve a max-min fairness, the global optimality of the network, often called *Pareto efficiency*, needs to be sacrificed. In order to handle this inefficiency and balance the tradeoff between fairness and efficiency, Kelly [44] proposed another fairness concept called *proportional fairness*.

In the future, network entities such as routers may also have the ability to indicate congestion by *marking* packets using an ECN mechanism [48, 49]. Several questions arise in the marking mechanism: How packets might be marked fairly [36]? How TCP might be adapted to react to marked packets? How to handle the mismatch problem between the macroscopic and the microscopic dynamics of queue length that may result in unfair marking (or dropping). In RED [18], for example, a parameter *count* is used to give uniformly distributed packet marking to eliminate bias against bursty connections. However, since RED imposes packet marking based on EWMA queue length, *unfair penalization* (or marking) may occur.

TCP-friendliness vs. UDP Traffic

One drawback of FIFO-based queue management (e.g., TD or AQM) is that there is no way to regulate misbehaving connections that send more than their bandwidth share and are non-responsive or less responsive [37] to congestion indication. On the one hand, in order to provide a fair share of bandwidth to all *TCP-friendly* connections that are responsive to the congestion indication, a queue management algorithm must regulate misbehaving connections effectively. One possible approach is to use per-flow queuing in order to discriminate against those *non-TCP-friendly* connections and to provide fair bandwidth share to connections. It is also possible to use *pricing* to give an incentive to TCP-friendly connection in terms of financial benefit. Another possible approach is to add new concepts of service, e.g., differentiated services, to connections. This is being studied by the Differentiated Services Working Group in the IETF [19].

On the other hand, since there is delay-sensitive traffic such as UDP-like real-time multimedia traffic in TCP/IP networks, and the demand for this type of traffic is increasing, queue management algorithms at a router should be efficient in providing some degree of QoS to such delay-sensitive traffic without degrading TCP traffic performance. Several

[6] *Action delay is defined in terms of two components. The first component is the* backward delay*, i.e., the delay between the time that the congested link issues congestion signals (e.g., packet losses or ECN marking) and the time that the source receives this signals. The other component is the* forward delay*, i.e., the time that it takes for packets generated by the source to reach the bottleneck link.*

approaches [37, 64] have been studied to provide better service to both TCP-friendly traffic and UDP-like delay-sensitive traffic.

Link Characteristics

With the growth of the size of the Internet and the evolution of communication techniques, high-speed links (e.g., optical fibers), long-delay and variable-delay paths (e.g., satellite links), lossy links (e.g., wireless networks), and others are becoming widely embedded in the Internet. Such different link characteristics on a path may result in a large bandwidth-delay product (BDP), long RTTs, bandwidth asymmetry, and non-congestion (or random) losses including white noise, frequency interference, signal fading, and attenuation on wireless links. Barakat *et al.* [65] describes the impact of each of the above characteristics on TCP, and present proposed solutions as well as their drawbacks and limitations. For example, a high-speed link (e.g., optical fibers at gigabits per second (Gb/s)) may have paths of large BDP. In this case, a TCP window must be able to reach a large value in order to maintain high utilization of bandwidth. However, at large windows congestion may lead to multiple packet losses. Queue management at the router becomes more important in maintaining TCP with good performance. In addition, long RTTs (e.g., satellite links) may cause other problems, such as long duration of the slow-start phase and unfair bandwidth allocation for the connection with long RTT. If packets are lost for reasons other than congestion, it may cause severe degradation of throughput. This problem can be solved by adding some mechanisms to deal with different types of losses separately.

Assumptions on Network Traffic Dynamics

The statistical properties of network traffic are highly dependent on the assumptions about the individual traffic sources being superposed. The bursty nature of packet-switched (or data) network traffic was observed in the 1970s and has been studied since [15, 66]. The bursty characteristic comes from the nature of packet switching, in which a large amount of data such as FTP data is chopped into a number of packets and sent consecutively in a short time. Based on observation and studies of network traffic, including LAN and WAN traffics with packet counts and/or byte counts, the long-range dependence and the heavy-tailed distribution had been observed and assumed as fundamental characteristics of network traffic [14, 15]. As a result, the packet arrival process was assumed not to be a Poisson process because the packet inter-arrival times were not independent and their marginal distributions were not exponential. Also, packet sizes were shown to have long-range dependence. Therefore, many studies have reported that the data traffic pattern is well modeled by a *self-similar* process in a wide range of network situations, rather than by a Poisson process [14, 15, 66, 67].

However, it has been shown recently that network traffic has *nonstationarity* properties for the long-range dependence and the heavy-tailed distribution [68, 69]. The nonstationarity of network traffic means that as the network traffic load and the types of the traffic sources increase, the long-range dependence of network traffic decreases to independence. Moreover, the superposition, i.e., statistical multiplexing, of the traffic sources changes the statistical properties of network traffic. As a result, it has been observed that packet queuing (i.e., the packet inter-arrival process) is expected to behave as a Poisson process arrival with independent inter-arrival times at higher traffic rates, while it behaves with long-range dependence at lower rates [68, 69].

Conclusion

The current Internet provides only *best-effort* service to users, and mainly depends on end-to-end congestion control, which is implemented in TCP/IP. Another way to deal with congestion is queue management (or network-assisted congestion control). Because of the bursty nature of Internet traffic and the deficiencies of traditional queue management (e.g., tail-drop) in handling such traffic, active queue management (AQM) techniques are being considered for deployment by the IETF. However, the problems associated with AQM, such as optimal parameter setting, insensitivity to the input traffic load fluctuation, and the mismatch between microscopic and macroscopic behavior of queue length dynamics, still need further attention.

TCP/AQM dynamics can be designed and analyzed by means of the feedback control modeling. The design goals of control theoretic modeling and design are increasing the speed of response and improving stability and robustness of congestion control. Other possible congestion control methods under consideration include architectural approaches such as source algorithm and/or network algorithm modification, and economic approaches such as pricing algorithms and optimization algorithms. However, many open issues still remain in the design and operation of current and future Internet congestion control. These issues include interoperability, robustness, stability, convergence, implementation complexity, fairness, TCP-friendliness, assumptions on network traffic dynamics and link characteristics.

References

[1] V. Jacobson, "Congestion Avoidance and Control," *Proc. ACM SIGCOMM '88*, 1988, pp. 314–29.

[2] M. Christiansen *et al.*, "Tuning RED for Web Traffic," *IEEE/ACM Trans. Net.*, vol. 9, no. 3, June 2001, pp. 249–64.

[3] Y. Zhang and L. Qiu, "Understanding the End-to-End Performance Impact of RED in a Heterogeneous Environment," Technical Report 2000-1802, Cornell University, Jan. 2000.

[4] L. Guo and I. Matta, The War Between Mice and Elephants," *Proc. IEEE ICNP '01*, Nov. 2001.

[5] S. Shenker, "Fundamental Design Issues for the Future Internet," *IEEE JSAC*, vol. 13, 1995, pp. 1176–88.

[6] W. Stevens, "TCP Slow-Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery Algorithms," IETF RFC2001, Jan. 1997.

[7] M. Allman, V. Paxson, and W. Stevens, "TCP Congestion Control," IETF RFC2581, Apr. 1999.

[8] L. S. Brakmo and L. L. Peterson, "TCP Vegas: End-to-End Congestion Avoidance on a Global Internet," *IEEE JSAC*, vol. 13, no. 8, Oct. 1995, pp. 1465–80.

[9] S. Floyd, "A Report on Recent Developments in TCP Congestion Control," *IEEE Commun. Mag.*, vol. 39, no. 4, Apr. 2001, pp. 84–90.

[10] D. Gevros *et al.*, "Congestion Control Mechanisms and the Best-Effort Service Models," *IEEE Network*, vol. 15, no. 3, May/June 2001, pp. 16–26.

[11] C. Lefelhocz *et al.*, "Congestion Control for Best-Effort Service: Why We Need a New Paradigm," *IEEE Network*, vol. 10, no. 1, Jan./Feb. 1996, pp. 10–19.

[12] W. Willinger and V. Paxon, Where Mathematics Meets the Internet," *Notices of the American Mathematical Society*, vol. 45, no. 8, Sept. 1998, pp. 961–70.

[13] W. Stallings, *High-Speed Networks: TCP/IP and ATM Design Principles*, Prentice-Hall, First Edition, 1998.

[14] W. E. Leland *et al.*, "On the Self-Similarity Nature of Ethernet Traffic (extended version)," *IEEE/ACM Trans. Net.*, vol. 2, no. 1, Feb. 1994, pp. 1–15.

[15] V. Paxson and S. Floyd, "Wide Area Traffic: The Failure of Poisson Modeling," *IEEE/ACM Trans. Net.*, vol. 3, no. 3, June 1995, pp. 226–44.

[16] B. Braden *et al.*, "Recommendations on Queue Management and Congestion Avoidance in the Internet," IETF RFC2309, Apr. 1998.
[17] A. Demers, S. Keshav, and S. Shenker, "Analysis and Simulation of a Fair Queuing Algorithm," *Internetworking: Research and Experience*, vol. 1, 1990, pp. 3–26.
[18] S. Floyd and V. Jacobson, "Random Early Detection Gateways for Congestion Avoidance," *IEEE/ACM Trans. Net.*, vol. 1, no. 4, Aug. 1993, pp. 397–413.
[19] The Internet Engineering Task Force, IETF home page, http://www.ietf.org/
[20] W. Feng *et al.*, "A Self-Configuring RED gateway," *Proc. INFOCOM '99,* 1999.
[21] J. Aweya, M. Ouellette, and D. Y. Montuno, "A Control Theoretic Approach to Active Queue Management," *Computer Networks*, vol. 36, no. 2–3, 2001, pp. 203–35.
[22] T. J. Ott, T. V. Lakshman, and L. Wong, "SRED: Stabilized RED," *Proc. IEEE INFOCOM '99*, 1999, ftp://ftp.telcordia.com/pub/tjo/SRED.ps
[23] W. Feng *et al.*, "Blue: A New Class of Active Queue Management Algorithms," Technical Report CSE-TR-387-99, University of Michigan, 1999.
[24] S. Kunniyur and R. Srikant, "Analysis and Design of an Adaptive Virtual Queue (AVQ) Algorithm for Active Queue Management," Technical Report, UIUC, Feb. 2001.
[25] V. Firoiu and M. Borden, "A Study of Active Queue Management for Congestion Control," *Proc. INFOCOM 2000*, 2000, pp. 1435–44.
[26] M. May *et al.*, "Influence of Active Queue Parameters on Aggregate Traffic Performance," Technical Report no. 3995, INRIA, Sophia Antipolis, France, 2000, http://www.inria.fr/RRRT/RR-3995.html
[27] V. Misra, W. Gong, and D. Towsley, "Fluid-Based Analysis of a Network of AQM Routers Supporting TCP Flows with an Application to RED," *Proc. ACM SIGCOMM 2000*, Sept. 2000. ftp://gaia.cs.umass.edu/pub/
[28] D. D. Clark and W. Fang, "Explicit Allocation of Best-Effort Packet Delivery Service," *IEEE/ACM Trans. Net.*, vol. 6, no. 4, Aug. 1998, pp. 362–73.
[29] M. Parris, K. Jeffay, and F. D. Smith, "Lightweight Active Router-Queue Management for Multimedia Networking," *Proc. Multimedia Computing and Net.*, vol. 3654, Jan. 1999, pp. 162–74.
[30] R. Rosolen, O. Bonaventure, and G. Leduc, "A RED Discard Strategy for ATM Networks and its Performance Evaluation with TCP/IP Traffic," *ACM Comp. Commun. Review*, vol. 29, no. 3, July 1999.
[31] S. Mascolo, "Congestion Control in High-Speed Communication Networks Using the Smith Principle," *Automatica*, vol. 35, 1999, pp. 1921–35.
[32] C. V. Hollot *et al.*, "A Control Theoretic Analysis of RED," *Proc. INFOCOM 2001*, 2001.
[33] B. C. Kuo, *Automatic Control Systems*, John Wiley & Sons, Inc., 7th Ed., 1995.
[34] C. V. Hollot *et al.*, "On Designing Improved Controllers for AQM Routers Supporting TCP Flows," *Proc. INFOCOM 2001*, 2001.
[35] S. Ryu and C. Rump. "PAQM: Pro-Active Queue Management for Internet Congestion Control," *Telecommun. Network Design*, edited vol. from *INFORMS Telecom 2002*, 2002, pp. 245–71.
[36] D. Wischik, "How to Mark Fairly," http://www.statslab.cam.ac.uk/~djw1005/, Sept. 1999.
[37] S. Floyd and K. Fall, "Promoting the Use of the End-to-End Congestion Control in the Internet," *IEEE/ACM Trans. Net.*, vol. 7, no. 4, Aug. 1999, pp. 458–72.
[38] M. May *et al.*, "Reasons Not to Deploy RED," *Proc. IEEE/IFIP IWQoS '99*, June 1999.
[39] D. Lin and R. Morris, "Dynamics of Random Early Detection," *Proc. ACM SIGCOMM*, Canne, France, Sept. 1997, pp. 127–37.
[40] S. Floyd, "Notes on Testing RED Implementation," http://www.icir.org/floyd/papers/redtesting, Oct. 1996.
[41] V. Jacobson, K. Nichols, and K. Poduri, "RED in a Different Light," http://www.cnaf.infn.it/~ferrari/papers/ispn/red-light-9-30.pdf, Sept. 1999.
[42] J. Walrand, *Communication Networks: A First Course*, McGraw-Hill, 2nd Ed., 1998.
[43] J. Mo and J. Walrand, "Fair End-to-End Window-Based Congestion Control," Technical Report M98/46, UCB/ERL, University of California at Berkeley, July 1998, http://www.path.berkeley.edu/~jhmo
[44] F. P. Kelly, A. K. Maulloo, and D. K. H. Tan, "Rate Control in Communication Networks: Shadow Prices, Proportional Fairness, and Stability," *J. Operational Research Society*, vol. 49, 1998, http://www.statslab.cam.ac.uk/~frank/rate.html, pp. 237–52.
[45] D. P. Bertsekas and R. G. Gallager, *Data Networks*, Prentice-Hall, 2nd Ed., 1992.
[46] D. Mitra and J. B. Seery, "Dynamic Adaptive Windows for High-Speed Data Networks with Multiple Paths and Propagation Delays," *Computer Networks and ISDN Systems*, vol. 25, 1993, pp. 663–79.
[47] L. Massoulié and J. Roberts, "Bandwidth Sharing: Objectives and Algorithms," *Proc. IEEE INFOCOM '99*, vol. 3, 1999, http://research.microsoft.com/users/lmassoul/infocom9.ps, pp. 1395–403.
[48] K. Ramakrishnan and S. Floyd, "A Proposal to Add Explicit Congestion Notification (ECN) to IP," IETF RFC2481, Jan. 1999.
[49] S. Floyd, "TCP and Explicit Congestion Notification," *ACM Comp. Commun. Review*, vol. 24, no. 5, Oct. 1994, pp. 10–23.
[50] L. Kalampoukas and A. Varma, "Explicit Window Adaptation: A Method to Enhance TCP Performance," *Proc. IEEE INFO-COM '98*, pp. 242–51, 1998.
[51] H. R. Varian, "Economic Issues Facing the Internet," http://www.sims.berkeley.edu/~hal/Paper/econ issues internet.ps, Sept. 1996.
[52] J. K. MacKie-Mason and H. R. Varian, "Pricing Congestible Network Resources," *IEEE JSAC*, vol. 13, no. 7, Sept. 1995, pp. 1141–49.
[53] S. Shenker *et al.*, "Pricing in Computer Networks: Reshaping the Research Agenda," *ACM Comp. Commun. Review*, vol. 26, no. 2, Apr. 1996, pp. 19–43.
[54] A. Odlyzko, Paris Metro Pricing for the Internet, http://www.research.att.com/~amo/
[55] S. J. Golestani and S. Bhattacharyya, "A Class of End-to-End Congestion Control Algorithms for the Internet," *Proc. 6th Int'l. Conf Network Protocols (ICNP)*, Oct. 1998, http://www.bell-labs.com/user/golestani/
[56] S. H. Low and D. E. Lapsley, "Optimization Flow Control, I: Basic Algorithm and Convergence," *IEEE/ACM Trans. Net.*, vol. 7, no. 6, pp. 861-874, Dec. 1999.
[57] E. Altman and T. Basar, "Multi-User Rate-Based Flow Control," *IEEE Trans. Commun.*, vol. 46, no. 7, pp. 940-949, July 1998.
[58] A. A. Lazar and Y. A. Korilis, "On the Existence of Equilibria in Noncooperative Optimal Flow Control," *J. Association for Comp. Machinery*, vol. 42, no. 3, May 1995, pp. 584–613.
[59] S. Shenker, Making Greedy Work in Networks: A Game-Theoretic Analysis of Switch Service Discipline," *IEEE/ACM Trans. Net.*, vol. 3, no. 6, Dec. 1995, pp. 819–31.
[60] S. Floyd, "Internet Research: Comments on Formulating the Problem," unpublished manuscript, http://www.aciri.org/floyd/, Jan. 1998.
[61] R. Srikant, "Control of Communication Networks," http://comm.csl.uiuc.edu:80/~srikant/Papers/, May 1999.
[62] S. Floyd, "Congestion Control Principles," IETF draft, draft-floyd-cong-01.txt, Jan. 2000.
[63] F. Bonomi and K. W. Fendick, "The Rate-Based Flow Control Framework for the Available Bit Rate ATM Service," *IEEE Network*, vol. 9, no. 2, Mar./Apr. 1995, pp. 25–39.
[64] Pittsburgh Supercomputing Center, the TCP-friendly Web site, http://www.psc.edu/networking/tcp friendly.html
[65] C. Barakat, E. Altman, and W. Dabbous, "On TCP Performance in a Heterogeneous Network: A Survey," *IEEE Commun. Mag.*, vol. 38, no. 1, Jan. 2000, pp. 40–46.
[66] K. Park and W. Willinger, *Self-Similar Network Traffic and Performance Evaluation*, John Wiley and Sons, 2000.
[67] W. Willinger, M. S. Taqqu, and A. Erramilli, "A Bibliographical Guide for Self-Similar Traffic and Performance Modeling for Modern High-Speed Networks," F. P. Kelly, S. Zachary, and I.

Ziedins, Eds., *Stochastic Networks: Theory and Applications*, vol. 4 of Royal Statists Lecture Note Series, Oxford, UK, 1996, Charendon Press, pp. 339–66.
[68] J. Cao *et al.*, "Internet Traffic Tends to Poisson and Independence as the Load Increases," Bell Labs Tech. Report, Bell Labs, 2001.
[69] J. Cao *et al.*, "On the Nonstationarity of Internet Traffic," *Proc. ACM SIGMETRICS 2001*, 2001, pp. 102–12.

BIOGRAPHIES

SEUNGWAN RYU (sryu@eng.buffalo.edu, rush@etri.re.kr) received the B.S. and M.S. degrees from Korea University, Seoul, Korea, in 1988 and 1991, respectively, and a Ph.D. from the State University of New York at Buffalo (SUNY), Buffalo, NY, USA, in February 2003, all in operations research. He joined the Electronics and Telecommunications Research Institute (ETRI), Teajon, Korea, in 1993, and is currently a senior member of engineering staff at the Mobile Telecommunications Research Lab., ETRI, Teajon, Korea. His research interests include modeling and analysis of communication network performance, modeling and control of Internet traffic, beyond the third (B3G) and the fourth generation (4G) wireless communication system design and analysis, and design and analysis of wireless sensor networks (WSN) and wireless personal area networks (WPAN).

CHRISTOPHER M. RUMP (crump@eng.buffalo.edu) is an assistant professor of industrial engineering at the State University of New York at Buffalo (SUNY). He earned a B.S. in applied mathematical sciences at Texas A&M University and a Ph.D. in operations research at the University of North Carolina at Chapel Hill. His research interests are in stochastic modeling, particularly in the design and control of queuing and inventory systems with application to telecommunication, transportation, and other service networks. This work has been partially funded by United Airlines, the Buffalo Police Department, the National Institute of Justice, and the Air Force Office of Scientific Research. Besides his work as co-director of the Applied Operations Research Lab at UB, he is also an active researcher in the National Center for Geographic Information and Analysis (NCGIA) and the Center for Multisource Information Fusion (CMIF).

CHUNMING QIAO (qiao@computer.org) is currently an associate professor at the State University of New York at Buffalo (SUNY), where he directs the Lab for Advanced Network Design, Evaluation and Research (LANDER). He has published more than 100 papers in leading technical journals and conference proceedings, and is recognized for his pioneering research on optical Internet and in particular, the optical burst switching (OBS) paradigm. His work on integrated cellular and ad hoc networking systems (iCAR) is also internationally acclaimed. He is an editor of several journals and magazines, including *IEEE/ACM Transactions on Networking (ToN)*, and *IEEE Optical Communications*, as well as a guest editor for several issues of *IEEE JSAC* and other publications. He has chaired and co-chaired many conferences and workshops, including the Symp. on Optical Networks at ICC'03 and Opticomm'02. He is also the founder and chair of the Technical Group on Optical Networks (TGON) sponsored by SPIE, and a vice chair of the IEEE Technical Committee on Gigabit Networking (TCGN).