Active Queue Management: A Survey

Richelle Adams, *Member, IEEE*

***Abstract*—Since its formal introduction to IP networks in 1993 as a viable complementary approach for congestion control, there has been a steady stream of research output with respect to Active Queue Management (AQM). This survey attempts to travel the trajectory of AQM research from 1993 with the first algorithm, Random Early Detection (RED), to current work in 2011. In this survey we discuss the general attributes of AQM schemes, and the design approaches taken such as heuristic, control-theoretic and deterministic optimization. Of interest is the role of AQM in QoS provisioning particularly in the DiffServ context, as well as the role of AQM in the wireless domain. For each section, example algorithms from the research literature are presented.**

***Index Terms*—Active Queue Management, AQM, quality-of-service, QoS, congestion control, differentiated services, DiffServ, wireless networks**

I. Introduction

THE TRADITIONAL role of Active Queue Management (AQM) in IP networks was to complement the work of end-system protocols such as the Transmission Control Protocol (TCP) in congestion control so as to increase network utilization, and limit packet loss and delay. During the earlier days of IP networks, the network traffic consisted mainly of bulk data transfers. The volume of web traffic was gradually increasing. The first formal and full proposal of an AQM scheme was Random Early Detection (RED), introduced by[1] in 1993. What followed was a plethora of AQM schemes proposed in the research literature, many of which sort to improve upon the RED algorithm itself in one aspect or another. There were, however, AQM schemes that were completely new. Additionally, there was also work that consisted primarily of a rigorous analysis of RED and which consequently highlighted its drawbacks.

The design of RED and many of its variants, though intuitive, has been, for the most part, heuristic. As a result, parameter-tuning has been one of their main limitations. Some researchers discovered that by applying more formal and rigorous techniques as found in control theory (whether it be classical control, modern control, optimal control or nonlinear control), this limitation may be alleviated if not eliminated. Other researchers have also invented AQM schemes based upon optimization techniques in the context of congestion control.

With the increasingly rapid march to convergence, i.e., data, voice, video and mobility, supported by a common IP platform that is shared by a growing heterogeneous set of communication technologies, the primary focus has shifted from congestion control (though still very important) to the more holistic theme of quality-of-service (QoS) provisioning. The main thrust of the latter is to have the network simultaneously and efficiently service the diverse requirements of the different types of traffic flows. In this new (and broader) context, the role of AQM is to serve as a mechanism for service differentiation. In the DiffServ architecture, in particular, it works in conjunction with other QoS mechanisms such as traffic conditioning and packet scheduling so that their combined effect would be to, in an average sense, distinguish one network service from another in terms of overall end-to-end delay, delay variation or jitter, packet loss and bandwidth according to mutually agreed upon service level agreements (SLAs).

Based on the current specifications for DiffServ, the main candidate AQM scheme is based on RED (specifically RIO-C (RED In/Out and Couple)) having a different set of parameter values for each drop precedence. However, it may be beneficial to capitalize on the vast AQM research that already exists, exploring those feasible alternative schemes and approaches that can be used in the DiffServ context so as to improve network performance and QoS.

The purpose of this literature survey is to revisit AQM research over the past 19 years (1993 - 2011) and outline the different considerations in and approaches to AQM design, taking into account their disadvantages and limitations. We would also look at some of the different schemes proposed for the DiffServ context and wireless context.

In the next section of this survey we recall four previous surveys conducted on Internet congestion control and AQM. There we also outline the value this survey attempts to bring to the research community. We propose an AQM taxonomy in Section III. In Section IV we provide a brief overview of AQM and its motivation for use. In Section V we discuss the RED algorithm and the various analyses conducted with regard to its performance. There we also provide a summary of RED variants and other heuristic schemes. We explore control-theoretic approaches to AQM design in Section VI together with optimization approaches in Section VII. We then examine in Section VIII the role of AQM in multi-service networks (particularly those that employ DiffServ) and look at AQM schemes that have been already devised for that context. With the increasing popularity of wireless and particularly mobile networks, we also look at the role AQM can play in these networks in Section IX. We discuss some challenges and gaps in AQM research in Section X and therein we make some recommendations. We conclude in Section XI.

For ease of reference, Table I provides a list of terms that will be commonly used in this survey. Their meanings will also be repeated within the main body of the text.

II. Other Surveys

Yang and Reddy[2], in 1995 proposed a comprehensive framework or taxonomy that encompassed major aspects of

Manuscript received 31 January 2012; revised 20 May 2012 and 19 August 2012.

R. Adams is with the Department of Electrical and Computer Engineering, The University of the West Indies, St. Augustine Campus, Trinidad and Tobago (e-mail: Richelle.Adams@sta.uwi.edu).

Digital Object Identifier 10.1109/SURV.2012.082212.00018

TABLE I
AQM NOMENCLATURE USED IN THIS SURVEY (UNLESS OTHERWISE STATED)

p	Packet dropping (or marking) probability
p_a	Actual probability that a packet will be dropped/marked
n	The number of unmarked/not-dropped packets since the last marked/dropped packet
$q(t)$	Instantaneous queue length at time t
$\bar{q}(t)$	Average queue length at time t
q_{ref}	Reference or target queue length
$e(t) = \bar{q}(t) - q_{ref}$	Error signal for control-theoretic approaches
B	Buffer limit
T	Sampling time
R_0	Round Trip Time (RTT)
R_0^+	Maximum RTT
N	Number of flows
N^-	Minimum number of flows
$\max_{th}$	Maximum queue length threshold – a RED parameter
$\min_{th}$	Minimum queue length threshold – a RED parameter
$P_{\max}$	Maximum non-congestion probability of dropping at the maximum threshold – a RED parameter
ω_q	Exponential-weighted-moving-average (EWMA) queue weight – a RED parameter
C	Link capacity
Δ	Interarrival time between packets when assumed constant (i.e., $\Delta = \frac{1}{C}$)
K_p	Proportional gain
K_d	Derivative gain
K_i	Integral gain
$[z]^+$	$\max(0, z)$

congestion control in packet-switched networks. They used criteria from control systems and classified congestion control algorithms that existed up to that time. Though there was no explicit mention of the term "Active Queue Management" in that article, one can place AQM under the categorization: closed loop control with global, responsive, explicity feedback, based on their explanations about the different categories. For added perspective, the taxonomy of Yang and Reddy is reproduced in Figure 1 with the position of AQM inserted.

In their 1999 survey[3], Labrador and Banerjee primarily and comprehensively discussed packet dropping policies for asynchronous transfer mode (ATM) networks. They also discussed three AQM schemes for IP networks, namely RED, RED In/Out (RIO) and Flow RED (FRED) that had been proposed in the research literature at that time. They compared RED and RIO in terms of fairness.

Router-based AQM schemes which treated with the unfairness issue between responsive TCP flows and unresponsive, aggressive UDP flows were discussed in the 2004 survey by Chatranon, Labrador and Banerjee[4]. They compared, in sufficient detail, FRED, Balanced RED (BRED), Stabilized RED (SRED), Stochastic Fair Blue (SFB), BLACK, CARE and CHOKe in terms of how these AQM schemes dealt with unresponsive flows. In this survey, an entire section was dedicated to the explanation of TCP flow and congestion control mechanisms and TCP models. The authors also provided a classification system for router-based AQM according to the fairness criterion.

"Advances in Internet Congestion Control" by Ryu, Rump and Qiao[5] was another survey, published in 2003, that treated with AQM. The article discussed both aspects of congestion control, i.e., congestion recovery and congestion avoidance. They explained the TCP mechanism and provided further references to that end. However, the real emphasis in this survey was active queue management. They presented the RED operation; briefly discussed Adaptive RED (ARED), Dynamic RED (DRED), SRED, BLUE, and AVQ; and summarized control-theoretic analysis and design. They then elaborated on three main problems with AQM proposals: parameter tuning, insensitivity to input traffic load variation, mismatch between macroscopic and microscopic queue length behaviour and their implications. They also presented some open issues then and which are still open today, such as interoperability and robustness, stability, convergence and implementation complexity, fairness, assumptions of network dynamics, UDP traffic and link characteristics.

What value does this survey bring? It is intended to expand on what was expounded before and provide an update on the position of router-based AQM work for IP networks up to 2011. Therefore, to provide the reader a more complete perspective on AQM research, especially with regard to the motivation behind the various proposals, we reiterate the RED algorithm, and its control theoretic analysis. We attempt to map the shortcomings of RED with AQM schemes that were consequently developed. We summarize additional AQM schemes proposed in the research literature. We discuss more advanced control techniques that have since been applied to AQM, e.g., Fuzzy control. We also propose a detailed classification system specific to router-based AQM schemes. Additionally, we discuss AQM schemes used and developed in QoS architectures such as DiffServ and also schemes developed for the wireless context. We also present a performance evaluation framework for the comparison of AQM schemes, both present and future.

III. PROPOSED AQM TAXONOMY

There are a number of ways by which different AQM schemes can be classified and compared. They can be compared in terms of their mechanisms of operation. These include the type of congestion indicator used, the manner in which their parameters are tuned, the methods by which they perform flow differentiation if any, their control function, and the nature of their feedback signal to the source algorithms. They can also be classified according to their context of use, whether it be in best-effort networks or networks with differentiated services, wired networks or wireless networks. Another schema for comparison could be by the performance of the AQM according to specified criteria (e.g., fairness, packet loss, throughput, delay, delay variation).

A. *Classification by mechanisms*

Figure 2 shows a classification according to AQM "Mechanisms". AQM proposals, so far have been queue-based, rate-based, load-based, packet-loss-based or a combination of these, and of which the average (usually exponential weighted moving average (EWMA)) or instantaneous samples were

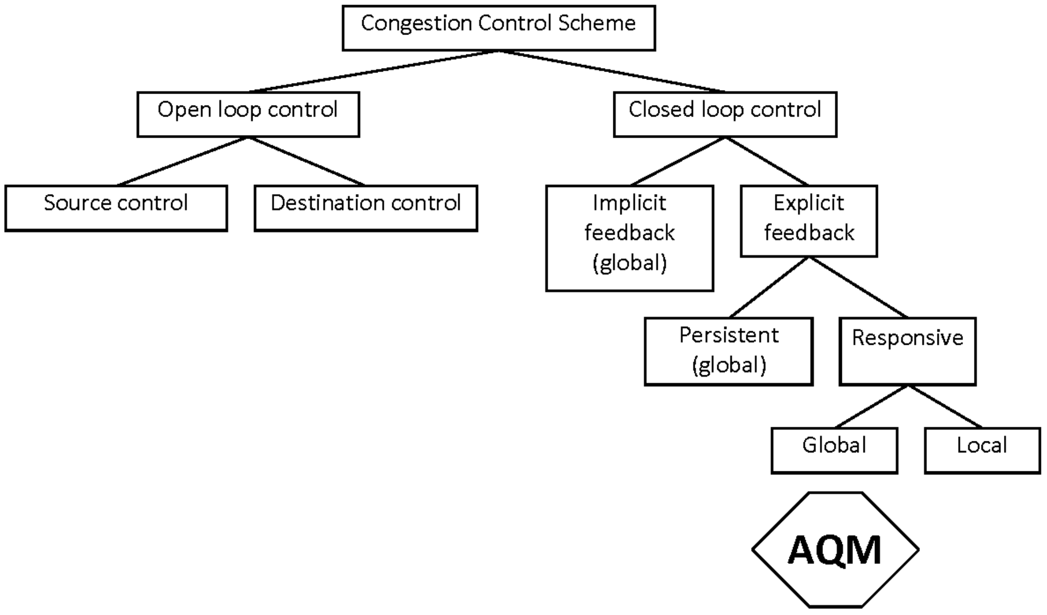


Fig. 1. AQM as part of the congestion control taxonomy

used as the congestion indicator. Some proposals updated the EWMA on each packet arrival while others performed the update at constant, predetermined intervals. This will be discussed in greater detail in Section IV.

Early AQM schemes such as RED had their parameters statically configured. Based on the disadvantages associated with this, many subsequent proposals attempted dynamic parameter configuration based on time-varying network conditions such as changing network load (i.e., number of flows). Additionally, it was realised that non-responsive flows and an increasing volume of short-lived flows posed a risk to TCP flows. Therefore, some AQM schemes had built-in some form of flow differentiation that was simply binary. However, as the years progressed, especially in the context of QoS realization, multi-class flow differentiation was recommended.

The control function is another differentiator among AQM approaches. It could have been derived heuristically, via control theory, or by optimization. As the years progressed, more sophisticated control algorithms have been put forward as plausible AQM schemes. These have been rooted in linear control, nonlinear control, optimal control, robust control, to name a few. And, as will be discussed later, others have sought to tackle the delay compensation issue via these means. What has been found though, is that many of the control-theoretic schemes have been exclusively TCP-centric with the TCP model being only that of the TCP's congestion avoidance phase. It may be necessary to revisit the "Plant" model for the control question so as to include, for example, responsive applications that use UDP as their transport layer. Optimization approaches have also been explored but mostly on the deterministic optimization front. So far, there has been found no proposals based on stochastic optimization.

The final AQM mechanism that may distinguish one AQM scheme from another may be the feedback signal used. Some have been designed exclusively for ECN marking while others can use either the implicit packet dropping or the explicit ECN marking depending on the network capability. Whether ECN or packet drops are used, however, the decision to mark or drop a packet could be random or deterministic. For the latter, typically a virtual queue approach is adopted. In Table II, examples of AQM schemes according to the classification just discussed are presented.

B. Classification by context of use

It may be argued that a proposed AQM scheme should be robust regardless of the context of use. So, for example, an AQM scheme created in the context of wired networks, should work equally well in the wireless domain. However, different contexts may have peculiar requirements and that may even be at odds with those of other contexts. With this in mind, we present in Figure 3 another means of classification among AQM schemes, i.e., according to context. The baseline differentiator would be whether or not they originated from the "best-effort" network philosophy versus the differentiated-services network paradigm that attempt to provide QoS guarantees. Within each of these two broad contexts, the AQM scheme may have been intended primarily for wired networks, while another AQM for wireless networks - which have their own idiosyncrasies. Within the wireless domain, the network under investigation could be adhoc or infrastructure-type. In the latter case, the AQM scheme may be placed at the junction of the air-interface and core, or exclusively in the core. Additionally, some AQM schemes may be built for the uplink, downlink or both. However, the predominant focus of study has been found to be the downlink case. When examining AQM schemes in the wireless context, especially at the air-interface, one must be especially cognizant of the air-interface standards or wireless technologies used,

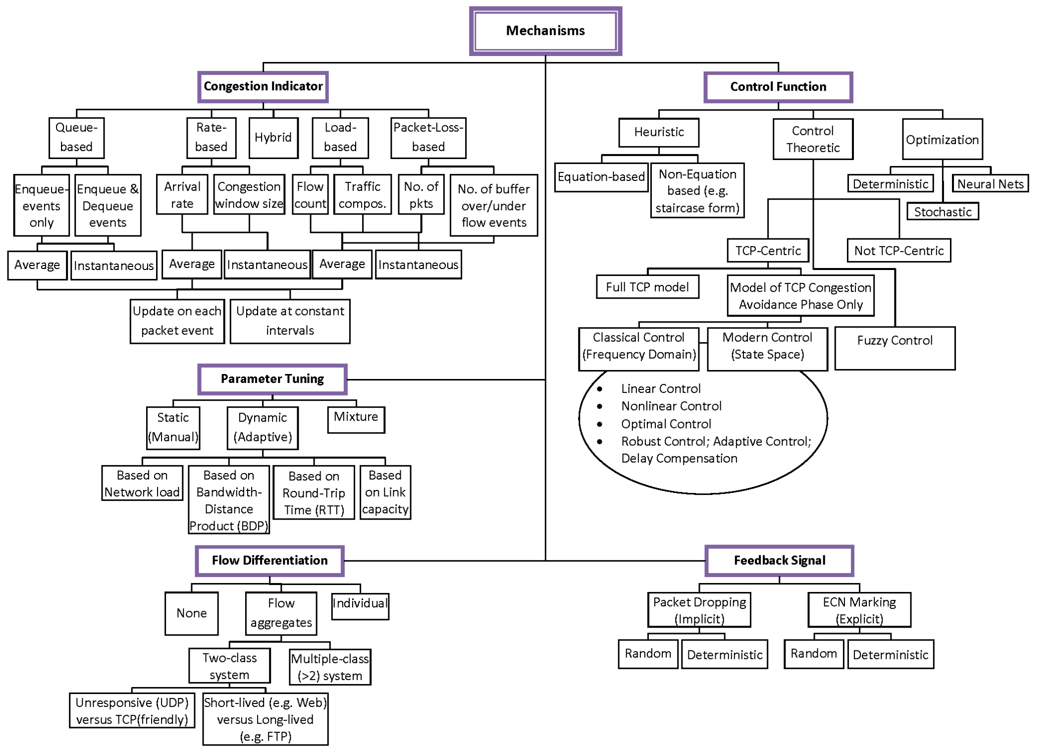


Fig. 2. AQM classification by mechanisms

since these determine the way radio resources are shared among traffic flows. In all cases, wired and wireless, the type of traffic used in the performance evaluation can also be used to refine the classification among AQM schemes.

C. Classification by performance criteria

Performance criteria for the comparison of AQM schmes, according to Figure 3, can be divided up into three main categories: steady-state behaviour, transient behaviour and complexity. The sub-fields for the latter two classes, though often used in the research literature, tend to be too nebulous in definition for the purpose of rigorous and independent comparison. A performance evaluation framework will be proposed in Section IX.

IV. AQM IN GENERAL

A. Congestion

According to [6], congestion occurs when the total demand for a resource, e.g. link bandwidth, exceeds the capacity of the resource. Put another way, congestion occurs when the arrival rate at a link interface exceeds the departure rate out of the link interface[7]. Results of congestion include: high latency in data transfers; wasted resources due to packet losses; and in extreme cases even congestion collapse, for which there is essentially no data transfer through the network: the throughput drops to zero and the response time tends to infinity[8]. The aim would then to be to control congestion or more ideally avoid congestion.

According to [8], congestion control seeks to rapidly bring the network out of an already congested state, whereas congestion avoidance attempts to keep the network at an optimal state (low delay, high utilization) and out of congestion in the first place. Even though there is congestion avoidance in a network, congestion control is still necessary for network recovery in the event congestion avoidance fails. On the other hand, a network with congestion control need not have congestion avoidance, however, optimality may never be realized.

B. Endpoint congestion control

Initial work in the area of congestion control concentrated on those algorithms and protocols employed at end-systems, e.g., the Transmission Control Protocol (TCP), a transport layer protocol. In general, congestion control algorithms could exist at the application layer or transport layer, but more so at the latter.

There are limitations to endpoint congestion control[8], [9], [10]. For the most part, endpoint congestion seeks to cure the network after congestion has already occurred[8]. In other words, it is reactive[11]. Also, there is a time lag between the packet drop event at the router and the source detecting this loss, during which the source will still continue to send at the high transmission rate that the network cannot support, leading to a high number of packet drops[12], [13]. There is the additional issue of bursty traffic. In fact, FTP, Web traffic and video traffic are bursty and selfsimilar in networks of all scales (e.g. Ethernet, WAN)[14]. So to absorb this burstiness and maintain high link utilizations, large buffers at

TABLE II
EXAMPLE AQM SCHEMES CLASSIFIED BY MECHANISMS

AQM Mechanisms			Examples
Congestion Indicator	Queue-based	Enqueue-events only	RED, GRED, CHOKe, DRED
		Enqueue & Dequeue events	FRED
	Rate-based	Arrival rate	LUBA, SFED, RARED
		Congestion window size	SHRED
	Load-based	Flow count	GREEN, BLACK, SFED
		Traffic composition	RED Worcester
	Packet Loss	Loss volume	LRED
		No. of buffer over/under flow events	BLUE
	Mixture		Load/Delay Controllers, Yellow
Parameter Tuning	Static		RED, GRED
	Dynamic	Network load	GREEN
		Bandwidth-Distance Product (BDP)	---
		Round-Trip Time (RTT)	GREEN
		Link Capacity	GREEN
	Mixture		ARED, A-RIO, PSAND
Flow Differentiation	None		RED, GRED
	Flow aggregates	Two-class system : Unresponsive versus TCP	Stochsastic Fair Blue, RIO-C
		Two-class system: Web vs FTP	SHRED
		Multiple-class system	RIO-DC, WRED, Rb-RIO, D-CBT, SFED
	Individual flows		FRED, BRED,BLACK
Control Function	Heuristic	Equation-based	RED, GRED, Hyperbola RED, DSRED
		Non-Equation based	MRED [Koo et. al. 2001]
	Control Theoretic	Classical and Modern Control	PI, PD, PID, Predictive PID, Adaptive GPC, PI-PD, PD-PD
		Fuzzy Control	Fuzzy Control RED, Adaptive Fuzzy RED, FUZREM, Fuzzy Green, Deep Blue, FLCD, Fuzzy PID, Fuzzy Choke
	Optimization	Deterministic	REM, AVQ, SVB
		Stochastic	---
		Neural Networks	Neural-network-based PID controller
Feedback Signal	Packet dropping	Random	RED, GRED, CHOKe, REM, SVB
		Deterministic	AVQ
	ECN Marking	Random	RED, REM, G
		Deterministic	AVQ

the routers would have to be used. But large buffers incur high queuing delays especially during congestion. Smaller buffers will reduce the queuing delays but the number of losses will rise and for the case of drop-tail queues link utilization will decrease dramatically. Essentially, one cannot simultaneously have high link utilization and low queueing delays[9]. Therefore a reasonable tradeoff is required between these two performance measures, i.e., high link utilization and low queueing delay[15]. This tradeoff should really be regarded as a matter of policy[9]. The limitations of endpoint congestion control are magnified by the growing number of multimedia and peer-to-peer applications, and these demand quality-of-service (QoS) guarantees in terms of delay, delay variation, loss volume and bandwidth availability. Current end-point congestion control cannot fully provide such guarantees on its own. See also[16], [17]. In fact, it has been found that TCP has been even detrimental to the performance of these applications due to its Additive-Increase-Multiplicative-Decrease (AIMD) operation. Therefore, although attempts to improve endpoint congestion control algorithms themselves continued, means by which more efficient assistance from the network to counter congestion needed to be explored[10]. This is where algorithms such as Active Queue Management (AQM) and packet (or queue) scheduling come to the rescue. (For a very comprehensive survey on TCP congestion control, see [18].)

The conventional wisdom at the time, was to keep the core network flexible and simple[19], therefore all the complicated tasks such as congestion control should be implemented in the end-points only. However, according to [19], more advanced and powerful hardware technology now exists and this makes it possible for the network (e.g. routers) to perform more complex tasks.

C. Network algorithms

At this point, it is important to distinguish between "scheduling algorithms" and "queue management algorithms"[10], [20], [21], [22]. Scheduling algorithms (e.g., Fair Queueing) guarantee fairness or priority weightings in terms of bandwidth allocations among flows. The buffer at the output of each router is partitioned into separate queues, and each of these queues is assigned to a particular flow or aggregate of flows[23]. Therefore each flow or aggregate of flows is isolated from the other and cannot degrade the performance of the other. The scheduling algorithm then decides the order in which packets from each flow (or aggregate) are transmitted on the link. However, these scheduling algorithms require the maintenance of per-flow state or per-aggregate state. On the other hand, queue management is the process by which a router decides when to drop packets and which packets to drop at its output port when it has become or

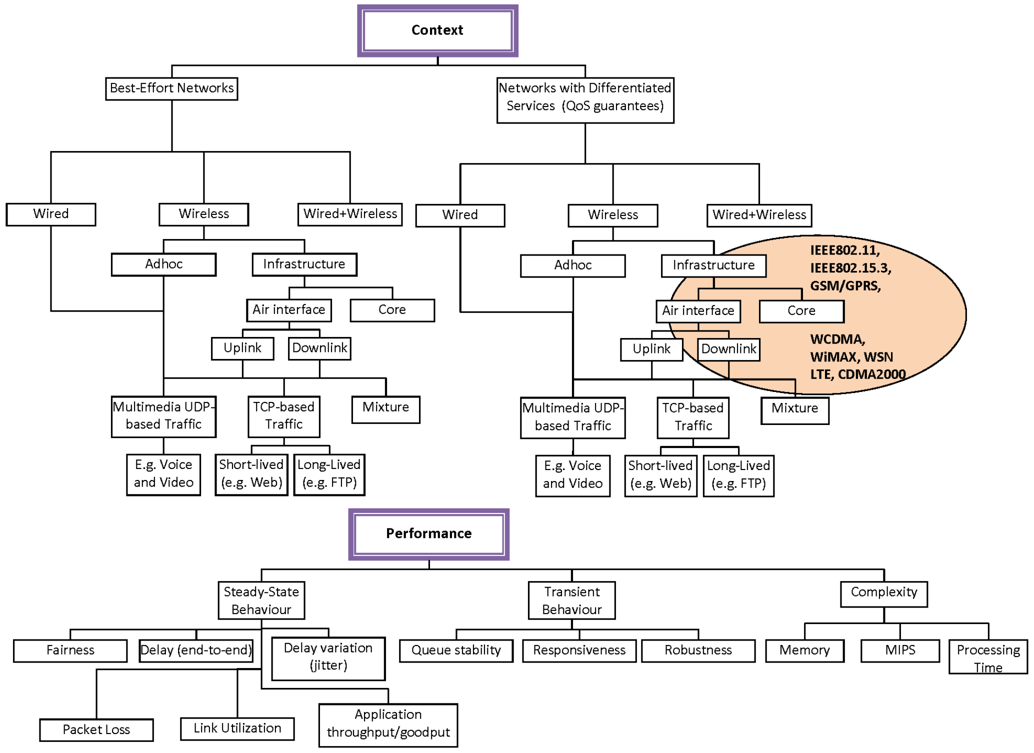


Fig. 3. AQM classification by context and performance

is becoming congested[24]. Queue management algorithms are comparatively more simple to implement since, for the most part, they can be implemented with a single First-In-First-Out (FIFO) queue for all flows and, for the most part, maintain no per-flow state. They attempt to approximate fairness, by appropriately dropping packets so as to limit network congestion and maintain suitable queue lengths[10], [20]. However, on their own, queue management techniques cannot effectively differentiate network flows as their scheduling counterparts[16].

1) Droptail queues: Traditional FIFO droptail queues were the only queue types used in the Internet[11]. They are simple and easily implemented in routers[11], however they exacerbate the limitations of endpoint congestion control schemes such as TCP.

Droptail queues cause 'lock-out'[14], [11], by which a small number of flows can obtain a disproportionate share of the link bandwidth, blocking other competing flows. Additionally, when a droptail queue becomes full, it simply drops all incoming packets thereafter. Thus, it will signal congestion to the sources only when congestion is already in full course (i.e., high packet drops and high queueing delays). The packet drops can also be bursty leading to network instability[25]. There is also the problem of global synchronization of TCP flows. Because the droptail queue drops all incoming packets when the queue becomes full, there is high correlation among these packet drops. It is possible that a large number of flows (to which these dropped packets belong) will be affected, and the TCP mechanisms managing these flows will all simultaneously reduce their congestion windows and hence their sending rates[26]. Therefore, there is huge oscillation between queue overflow and underflow and this contributes to queue-delay variance (also known as jitter). Also a single flow may experience multiple consecutive packet drops and will consequently repeatedly reduce its sending rate. Long queueing delays can be incurred at droptail queues as they fill up, especially for large sizes. This, together with jitter, makes it unacceptable for real-time applications which are delay sensitive. Additionally, jitter interferes with the TCP self-clocking mechanism as the spacing between acknowledgement packets (ACKs) will vary more, causing TCP to become even more bursty[27].

Droptail queues, due to their packet-dropping policy can penalize competing flows with longer round-trip-times more than those with shorter round-trip-times[28]. The growth of TCP window size (and hence sending rate) is inversely proportional to the round-trip-time[29]. Therefore, a flow with a small round-trip-time will have its throughput increase more quickly and will take up much more slots in the queue than a flow with a longer round-trip-time. However, when the queue actually becomes full, all flows will experience packet drops regardless of how much of the queue they presently occupy. In all, flows with larger round-trip-times will suffer more by droptail policy. Droptail queues also penalize bursty connections[28].

2) Active Queue Management (AQM): Active Queue Management (AQM) complements the endpoint congestion control with the aim to prevent congestion[8].

AQM, unlike the droptail queue, is a proactive congestion control scheme by which the network sends information to the sources when it detects incipient congestion. The information can be sent explicitly in the form of Explicit Congestion Notification (ECN) marks or implicitly by packet drops[14]. As the congestion level increases the AQM scheme intensifies its feedback to the TCP endpoints, i.e., by marking or dropping more packets. The sources, in response to the congestion notifications, reduce their transmission rates so as to prevent queue overflow and limit the losses that can result[30], [11]. Therefore, it is imperative that the AQM quickly detect congestion and provide rapid and effective feedback to the sources[11]. Because the AQM algorithm acts within a router, (the place where the congestion is actually taking place,) it will obtain more accurate congestion information faster than the traffic sources[11]. Additionally, even though the source may not be responsive (e.g. constant-bit-rate video, UDP) to the congestion notifications by the AQM, the AQM can still provide some degree of congestion control at the router[11]. However, because AQM was designed to work with purely responsive flows[31], [32], the effect of non-responsive flows on AQM performance should be understood[33].

Another plausible benefit of AQM is that by its random dropping (or marking) of packets before buffer-overflow occurs, it can possibly overcome the global synchronization problem experienced when droptail queues are instead used. It is also believed that random dropping improves fairness among flows[26]. AQM has been recommended by the Internet Engineering Task Force (IETF)[14], [26], [34], [35], [36]. See RFC2309.

In order to detect link congestion, AQM may utilize any combination of the following: queue length, input rate, and events of buffer overflow and emptiness[15]. Therefore an AQM can allow temporary buffering of excess load so that good link utilization be maintained but then it can trigger congestion notifications when this condition persists (e.g., queue length is too high for too long a period) so as to maintain low queueing delay[28], [37]. It should be noted that the actual buffer capacity should be sufficient to absorb data bursts, but overall, a small queue length should be maintained[11].

An AQM scheme, besides providing high link utilization and low queueing delay that is consistent, should provide fairness among flows, by sending congestion notifications fairly to the various sources[28]. Because of its proactiveness an AQM has been an integral QoS mechanism. For example, in DiffServ, a scheduler divides the bandwidth among classes, but within a class, an AQM is used to prioritize its flows in terms of drop precedence.

So besides the major objective of providing congestion control (and more specifically congestion avoidance)[38], [32], [39], two more specific goals of an AQM emerge: (1) to provide predictable queueing delay and (2) to maximize link utilization[40]. These two goals are conflicting[8], [41], [34]. However, AQM can realize a solid compromise between the two[15]. By keeping the queue length around a target that is small, one can realize low and predictable queueing latency, and can make the network performance independent of traffic load (i.e., the number of connections)[40], [30]. Also, by stabilizing the queue length around a low value (with enough extra buffer space to absorb transient bursts) one can realize high link utilization since unnecessary packet drops will not occur[40], [35], [42], [41], [43].

D. AQM components

In general, queue management schemes comprise three components: (1) the congestion indicator, (2) the congestion control function and (3) the feedback mechanism[30], [41], [32], [44]. See Figure 4. The congestion indicator is used by the queue management to decide when there is congestion whereas the congestion control function decides what must be done when congestion is detected. The feedback mechanism is the congestion signal used to alert the source to alter its transmission rates. As an example, the congestion indicator for droptail queues is the instantaneous queue length, the congestion control function is to drop all incoming packets with probability of one (1) when the queue becomes full, and the congestion signal is dropped packets. For Active Queue Management these three components are significantly improved over droptail so as to not be reactive to congestion but rather proactive. Here, we examine more deeply the implications of the choice of congestion indicator, control function and feedback signal in AQM.

1) Congestion indicator: The aim of rate-based AQM is to keep the packet arrival rate at the queue at a target value, say, at some percentage of the link capacity. It indirectly controls queue length[14]. On the other hand, a queue-based AQM focuses on the (instantaneous or average) length of the queue [40] and its control aim is to keep the queue length at a target value[45], [46], [34]. (It does not directly control the arrival rate at the queue.) There are AQM schemes which use a combination of both arrival rate and queue length to measure congestion at the queue. There are relative merits and demerits among the measures.

The early AQM algorithms were based solely on the length of queues[15], but as AQM research progressed, there were questions as to the efficacy of using only queue length as congestion indicators, if at all. This issue is examined extensively in [34].

According to [34], a queue-based AQM algorithm will inflict a large dropping probability on flows when the queue length is large, even though the net-inflow rate at that time might be quite low (i.e., the arrival rate is far less than the drain rate and the queue is draining rapidly). This will result in unnecessary packet losses which in turn can cause low link utilization. Conversely, when the queue is small, the queue-based AQM algorithm will drop a much smaller fraction of packets than it should, if the net-inflow rate itself, is quite high, thus making the AQM inept at detecting incipient congestion. If, however, the net-inflow rate were simultaneously taken into account with the queue length, the AQM response to congestion would be faster and the link utilization will not be lowered[34].

Although for queue-based AQM schemes the queue length is controlled to a target value, it is still non-zero and therefore can still incur a non-zero queueing delay and even some variations in that queueing delay. Additionally, even though

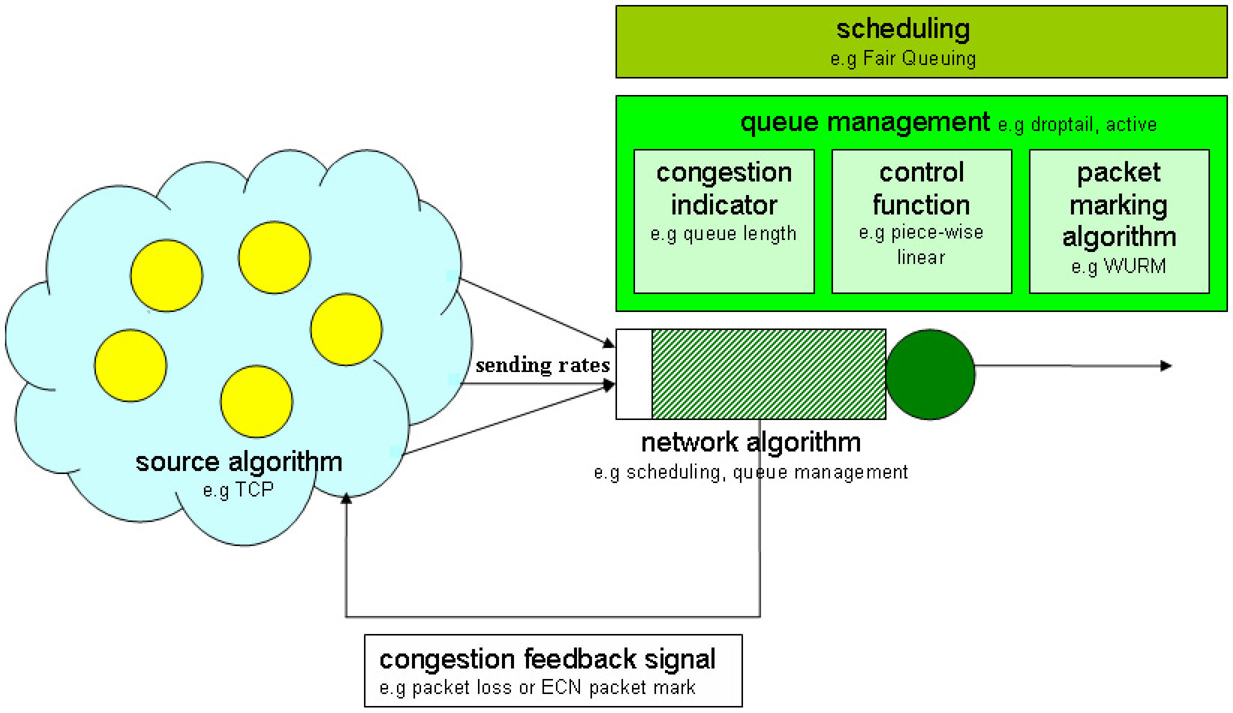


Fig. 4. Congestion control in the Internet

a target queue length is set for queue-based AQM, how close that target is to reality, is highly load dependent[14].

Based on the aforementioned disadvantages, one may be tempted to use rate information only, however, as cautioned in [34], this can lead to instability. So queue length is still an important measure (although insufficient on its own). It provides a more stable indication of congestion and can provide a direct QoS guarantee of maximum queue latency. It is suggested in [14] that although rate is a better congestion indicator than queue length, the combination of both queue length and rate information can provide a tradeoff between stability and responsiveness for the AQM. Also rate-based AQM is more favoured because it is easier to extend to traffic classes[47]. It was suggested by [34] that queueing delay be used instead of queue length since it provides a more meaningful measure to applications than queue length and of itself is independent of the link capacity. However, what must be done is to convert the target queue-delay into a target queue length. It was suggested in [48] that an even more enriched congestion indicator should be used. It should include the round-trip time, link capacity and number of sources for delay compensation between the time the congestion is measured and the time the source receives the congestion signal. See also[49], [50].

2) Congestion control function: The congestion control function is really a marking/dropping probability function which maps the current level of congestion (as given by the congestion indicator) to the probability of marking or dropping the incoming packets[51]. Closely associated with the congestion control function is the packet marking/dropping algorithm[26]. Given the probability of packet drop calculated by the congestion control function, this algorithm actually determines which packets are actually marked or dropped. This stage is often ignored (as stated in [26]), but what has been realised is that for a given marking/dropping probability, different algorithms can yield different queue variances, loss and delay.

The type of control function also varies in terms of the number of parameters employed. However, although more parameters can mean a more fine-tuned control, it can be an impossible challenge to know exactly how to adjust them. Also, one should note that different control function types may be best suited for different traffic environments. (See[51] for more details.)

3) Feedback mechanism: As mentioned earlier, packets can be dropped (so that, TCP, for example, will infer congestion through triple-duplicate acknowledgements or retransmission timeouts) or packets can be ECN-marked (in which case, the TCP endpoints should be ECN enabled). Although ECN can prevent "low-latency efficiency collapse", it was found that a system using packet dropping instead of ECN marking is more stable[52]. This is because it takes longer for the source to detect packet drops (after triple-duplicate ACK or timeout) than ECN marks.

E. AQM design - heuristic versus theoretic

So far we have looked at the different options that have been used or suggested for the different AQM modules, i.e., congestion indication, congestion control function and congestion signal. We now look briefly at how the control function was acually designed, i.e., whether heuristically or analytically.

1) Heuristic design: Quite a number of AQM schemes have been built on heuristics and have performed respectably well especially when compared to droptail queues, yet many of their parameters must be manually tuned according to differing network conditions[31]. The effects of these parameters are not fully understood and in some cases can quickly drive the system into instability (and hence low utilization) or can cause quite sluggish behaviour (i.e., the system slowly converges to a new operating level when the traffic evironment changes)[43].

2) Theoretic design: The aim of the control theoretic approach, however, is to provide a more stable and faster responding system[7].

Heuristic design depends heavily on intuition[43]. However, not always is intuition reliable. Since there is no cohesive formulation to support the tuning of the parameters[48], discrete-event simulation is crucial to the validation of heuristic designs[53]. By modeling one can gain much deeper insight into the system's behaviour (for a wider range of possible scenarios), and more specifically into the interaction of the endpoint congestion control algorithm with AQM[25], [48].

a) TCP modeling: For the primary purpose of congestion control, the AQM should work with responsive source algorithms, such as TCP. Therefore, the AQM cannot be designed in isolation. The AQM design must incorporate the closed-loop characteristic of the source algorithm[32].

Therefore, the typical theoretical approach thus far in AQM design is to first develop a fluid model for TCP and then use control theory to design the AQM. Fluid models have been derived for the dynamics of long-lived flows using TCP (e.g., FTP connections), modified for the inclusion of unresponsive UDP flows, and also for short-lived flows such as Web traffic[33]. So far the fluid models used for AQM design attempt to capture the TCP AIMD process (i.e., the congestion avoidance phase) while ignoring for example, slow-start, and retransmission timeouts with exponential back-off (which are important to characterising TCP behaviour with short-lived flows (see[54])). According to [17], the congestion avoidance phase causes a quasi-periodic oscillation in window-size for which the duration between loss indications is random and has a statistical mean[49]. From this we can already infer that there are limitations to TCP modeling.

For simplicity and mathematical tractability, the development of the fluid model first begins by linearizing the non-linear TCP system around an operating point and assuming that the system is time-invariant. In other words, the TCP system is converted into a linear-time-invariant (LTI) system. (See [55] for the derivation in detail.) For the first case, i.e., the linearization, problems would arise if the operating point shifted dramatically during actual system operation. For the second case, i.e., assuming stationarity, statistical errors can occur, since TCP is really non-stationary due to the network conditions changing frequently. It is imperative to determine how faithfully the TCP fluid model represent the reality after all these assumptions, and to clearly understand when will these assumptions hold. Discrete-event simulation is still necessary (definitely not as much as for heuristic designs) since it will implicitly take into account the non-linearities, and by it one can see how major a role these non-linearities play and if they can or cannot be ignored. In other words, the simulation validates the model[38], [55]. However the simulation process is more structured and guided than for the heuristic case. For the AQM design itself, classical and modern control theory has been used (see[56], [43], [42], [49], [11], [55], [46], [48]), as well as robust control methods, and optimal control and optimization techniques. The control-theoretic approach to AQM design will be discussed further in Section VI.

b) Optimization approach: The optimization approach developed by Kelly et al.[57], [58] and extended further by Low et al.[59], formulates the congestion control problem as that of the maximization of an aggregate source utility via an approximate gradient algorithm[44] subject to network capacity constraints, hence the optimal controller (i.e., the AQM scheme) from this approach would either optimize source rates or optimize the congestion measure[56]. According to [17], it is a global optimization process with the computation carried out by sources and links in the network (i.e. distributed computation). From the perspective of optimizing source rates, actual AQM realizations have been proposed by Kunniyur et al.[60], [61] using virtual buffers. The main disadvantage of the optimization approach is that the focus is on steady-state equilibria, rather than on the transient performance of the AQM[56]. The optimization approach to AQM design will be discussed further in Section VII.

Control-theoretic and optimization approaches thus far do not deal explicitly with fairness and, let alone, QoS guarantees[62].

Table III summarizes the main advantages, disadvantages and underlying assumptions of the main types of control functions used in AQM schemes. Table IV, on the other hand, provides a list of the AQM schemes mentioned in this survey, sorted according to the AQM control function, as well as according to the differentiated services and wireless contexts which will be discussed in Section VIII and Section IX respectively.

F. AQM performance

Besides achieving its main goals of congestion avoidance, predictable queueing delay (that is low), and high link utilization, an AQM scheme should promote high network stability, robustness, responsiveness and scalability. In terms of robustness, it is necessary that the AQM algorithm perform consistently well under extreme and unfavourable network conditions, i.e., it should not be sensitive to variations in network parameters[13]. Robustness can be enhanced when parameters in the AQM scheme can be dynamically tuned according to changing traffic load[32]. According to [32], robustness can increase when the AQM takes into account the long-range dependence property of the traffic since this can provide sufficient information to predict traffic intensity. According to [33], responsiveness is the speed of convergence to an equilibrium. With regards to stability, the performance of AQM algorithm should not vary dramatically due to a sudden change in network conditions. One test of stability is

TABLE III
COMPARISONS OF DIFFERENT APPROACHES TO AQM DESIGN

	No AQM (i.e., Droptail)	**Heuristic**	**Control-Theoretic**	**Deterministic**
Advantages	• Simplicity in implementation	• Simplicity in design and implementation to a certain extent	• Justification available for choice of parameter values, i.e., stability, responsiveness • Transient analysis	• No TCP model assumed • Can work different round-trip times among users
Disadvantages	• Lock-out • Global synchronization • Network instability • No service differentiation	• Arbitrary and difficult parameter tuning	• Most are TCP-centric • Limited TCP model used ○ Linearization of non-linear system ○ Congestion avoidance phase – long-lived flows • No explicitly treatment for fairness	• No explicit stability analysis • Primarily rate-based approaches therefore difficulty to control queue lengths • Arbitrary tuning of certain parameters • Primarily steady-state solution
Assumptions			Most analyses consider • A single congested link • No retransmission timeouts • Synchronized sources ○ Same RTT ○ Number did not change with time ○ Same window progression ○ Same window sizes ○ Same average packet size	

the extent to which the queue length varies when, for example, the number of connections flowing through the queue suddenly increases. It is imperative to avoid oscillations between queue overflows and underflows since this surely leads to overall link under-utilization and variation in queueing delay. Scalability is essential, since one would want the AQM to continue to operate soundly even when the speed and the number of routers and links have dramatically increased in a network.

1) The ideal AQM: Therefore, an ideal AQM scheme would achieve stability, robustness, responsiveness, high throughput, low latency and fairness all at the same time. It would also be scalable and very simple to implement. To this end such an AQM scheme will ensure that the following be accomplished:

- It will not unnecessarily penalize bursty flows and will prevent to a great extent global synchronization[63]. This in turn will translate into lower packet losses and therefore higher throughputs.
- It should be able to deal with bursts so that packets are not dropped unnecessarily.
- It should minimize jitter which is due to large oscillations in queue length. Also, these oscillations are such that the queue becomes empty quite often. This leads to low link utilization[42].
- It should be able to work in large delay networks[14].
- It should not introduce additional bias to flows with smaller round-trip times, since TCP already does this.
- It should be able to work in an environment with mutliple bottlenecks and should take into account the flows that are already being controlled by other nodes[64]. It is suggested by [64] that the AQM controllers within the network should be cooperative by taking into consideration the congestion status of other nodes in the network.

According to [12] this ideal AQM algorithm will deliver congestion notifications at a rate so as to keep the aggregate TCP sending rates into the queue just below the output rate of the queue. It is suggested by [63] that the congestion indicator and control function should be adaptive to changes in the amount and nature of the traffic. The authors of[24] concede that a practical AQM will not perform optimally in all scenarios. An AQM scheme, for example, may accomplish stability at the expense of responsiveness; low-latency at the expense of ideal throughput; and fairness at the expense of simplicity.

2) Factors that impact AQM performance: Here are some factors that will possibly impact AQM performance, as outlined by [63]:

- Buffer size and link capacity. These factors tend to be time-invariant factors which are pre-determined.
- Traffic load and round-trip time: The traffic load (which is the number of flows) and the RTT change dynamically with time. The AQM must be robust enough to deal with such behaviour. In fact, according to [63] an increase in the round-trip time degrades system stability.
- It was discovered by [26] that the rate-variability within

TABLE IV
AQM SCHEMES MENTIONED IN THIS SURVEY

Heuristic Approaches	Control-Theoretic Approaches	Deterministic Optimization Approaches
Random Early Detection (RED),	Proportional Integral (PI) Controller	Adaptive Virtual Queue (AVQ),
Adaptive RED (ARED),	• PI according to phase margin; control signal: p,	Random Early Marking (REM),
Balanced RED (BRED),	• PI according to gain margin; control signal: p,	Stabilized Virtual Buffer (SVB)
BLACK,	Proportional Derivative (PD) Controller	
BLUE,	• PD with Control signal: $P_{\max}$,	
CHOKe,	• PD based on state-space model with large delay compensation,	
Double Slope RED (DSRED),	• PD with control signal: p,	
DREAM,	Proportional-Integral-Derivative (PID) Controller	
Fair RED (FRED),	• PID with Control signal: p,	
Gentle RED (GRED),	• PID based on IMC principle,	
GREEN,	• Predictive PID based on Generalized Predictive Control (GPC)	
Hyperbola RED (HRED),	Gain Adaptive Smith Predictor with PI,	
Load/Delay Controller,	Robust Smith Predictor based on both Smith Predictor and PI,	
Loss-Ratio-Based RED (LRED),	Adaptive GPC,	
LUBA,	PI-PD,	
Modified RED (MRED)-version 1,	Fuzzy Logic:	
Modified RED (MRED)-version 2,	• Fuzzy Control RED,	
Parabola RED (PRED),	• Adaptive Fuzzy RED,	
PSAND,	• Fuzzy Explicit Marking,	
Random Early Adaptive Detection (READ),	• FUZREM,	
RARED,	• Deep Blue,	
RED-Worcester ,	• FLCD,	
Short-Lived Flow Friendly RED (SHRED),	• Fuzzy PID,	
Stabilized RED (SRED)	• Fuzzy Choke	
Stochastic Fair BLUE (SFB),	• Fuzzy Green,	
Yellow		

Differentiated Services Network Context		Wireless Network Context	
RED In/Out (RIO),	Autonomic AQM (AAQM)	Adaptive QoS and Wireless Bandwidth (Adaptive QWB),	Proxy-RED,
RIO-Coupled (RIO-C),	FEM In/Out	Adhoc Hazard RED	Rate-based Exponential AQM (REAQM),
RIO-Decoupled (RIO-DC),	FUZAQM	Autonomic AQM (AAQM),	Remote AQM
Adaptive RIO (A-RIO),	Hybrid RED	Burst-Sensitive RED (BSRED),	Slope Based Discard (SBD),
Dynamic Class-Based Threshold (D-CBT),	Modified BLUE with In and Out (MBIO)	Channel-Aware AQM (CA-AQM),	TTLRED,
Rate-based RIO (Rb-RIO),	Modified Hybrid RED (HRED)	CLAMP,	Wireless Delay-Based Queue (WDBQ),
Rate-based n-RED,	PID controller with Priority Dropping,	Dual RED,	
Selective Fair Early Detection (SFED)	STAC	EF-AQM,	
Weighted RED (WRED),	Two-level PID using Linear Quadratic Regulators (LQR)	PAQMAN	

and the phase relation among flows impact AQM performance. If window progression of an individual TCP flow is a smooth process so will be its arrival rate into the queue. If all the flows entering a queue are uniformly out-of-phase then AQM performance will not be significantly impacted, however if they are completely in-synch with each other then the burstiness will be more intense and the AQM performance may deteriorate.

- It was shown by [46] that there is a connection between routing and congestion. Using a term called the routing matrix gain to mathematically represent topologies, it was found that the network robustness was inversely proportional to this routing matrix gain. If it were possible to know a priori the topology of the network in which the AQM would exist, then the AQM could be tuned to deal with this potential instability. But, in reality, this could not be done, therefore there was the question of whether or not some range of routing matrix gains exists which can instead be used[46]. Routing affects congestion (and hence AQM performance) because it determines the path of queues through which packets must flow. If there are a number of congested queues along the path, the round-trip time (and hence the TCP sender's ability to respond to congestion) is affected due to varying queue lengths along the path[46].
- The presence of short-lived flows and unresponsive flows among the responsive TCP long-lived flows, and even a mix of TCP variants[29], [14], [13].
- Reverse-path asymmetry. With congestion on the reverse path, ACK packets can be lost or bunched together (ACK compression). This in turn increases the burstiness of TCP[9].

Quite a number of performance metrics have been used and presented in the literature for evaluation and comparison of AQM schemes. It is implied by [36] that the lack of consistency in evaluating AQM may have been the reason that there had been very little deployment in real networks. To this end a benchmarking framework for AQM was proposed by [36]. But it must be understood that even though a standard set of performance criteria were established, the task of comparing AQM algorithms will still be a daunting one, especially for the heuristic schemes. Nevertheless, whatever performance comparison is attempted, it is customary practice to use the simple droptail queue as the baseline[52]. More about the performance evaluation of AQM schemes will be addressed in Section X.

G. AQM fairness, unresponsive flows and QoS guarantees

To deal with the issue of fairness and unresponsive flows, there have been some AQM algorithms that for their own operation do use some degree of per-flow information as do their scheduling counterpart. Although it is preferrable not to use any state information whatsoever, AQM schemes that use no per-flow information, though least complex, give the worst performance in terms of fairness[65]. So according to [65], there are three types of AQM with respect to fair bandwidth allocation: (1) AQM with no per-flow information, (2) AQM with per-flow information and (3) AQM with per-flow scheduling. The latter, of course will achieve the greatest level of fairness but will be the most complex. The second option is a compromise.

1) Fairness: One of the earlier objectives of AQM was to provide fairness in the context of bursty traffic versus non-bursty traffic. Droptail queues unnecessarily penalized bursty traffic flows, therefore the very first AQM at the outset attempted to remove this bias by introducing queue averaging into the congestion indicator. There have been subsequent debates as to the efficacy of this approach.

a) UDP flows: It was found that as new applications were being deployed on the Internet they were opting not to use TCP congestion control. These applications which included realtime multimedia, streaming media and Voice-over-IP (VoIP)[16], [20], [66] generated large amounts of traffic. They tended to have very stringent delay and jitter requirements[1] which TCP could not provide[47] due to its retransmission mechanism and its windowing operation that contributed to traffic burstiness[22]. Hence, some used as their transport layer, UDP, which does not have any congestion control mechanism[2]. These applications might, for the most part, be unresponsive to any congestion indication (e.g., dropped packets) from the AQM, therefore they would not reduce their rates when there is congestion. On the other hand, competing TCP flows, upon detecting congestion, would reduce their rates. This meant that unresponsive flows would obtain a greater share of the available bandwidth[68] and TCP flows would be penalized[69]. This was exacerbated by first-come-first-serve (FCFS) scheduling, in which the output bandwidth consumed by a flow is directly proportional to the space it occupies in the buffer[28], therefore the unresponsive flows with their uninhibited high sending rates would fill up the buffer much more quickly than a responsive TCP flow. Therefore, it would be good if an AQM (that is coupled to FCFS scheduling) be able to accurately identify or isolate nonresponsive flows, protect responsive TCP flows and sufficiently limit these unresponsive flows[12], [69], [29], [70]. This situation had to be addressed soon since the volume of unresponsive traffic was increasing at a fast rate[27], [7], [66], [69], [62], [14], [22].

b) "TCP-friendly" flows: In order to encourage new applications to adopt congestion control a number of source algorithms, compatible with TCP, yet more conducive to the demands of multimedia and other streaming and real-time traffic, were developed. Termed 'TCP-friendly', examples of these protocols included Generalized AIMD (GAIMD), Binomial and TCP Friendly Rate Control (TFRC)[71]. However, even though in the long-term a flow using these protocols should produce, on average, the same throughput as a flow using TCP, there were times when these protocols were more aggressive or more gentle than TCP due to the manner in which they reacted to congestion indications. Therefore, the control action of the AQM would still have maintained some fairness among these different flows within the same network[65]. The authors of[66] claimed to be the first to have provided a comprehensive evaluation of AQM performance in the presence of TCP-friendly flows.

c) Difference in round-trip time (RTT): As was discussed earlier, even when flows use the very same endpoint congestion control implementation, there can be unfairness due to differences in their round-trip times[72], [69]. For example, TCP flows with shorter round-trip times usually obtain more bandwidth than those with longer round-trip times. This is because they get their acknowledgments faster and can therefore ramp up their congestion window faster[72].

d) Web traffic: Traditionally, AQM research focused primarily on long-lived TCP flows (e.g., FTP), however, according to [73] the protection of short-lived flows such as Web transfers was becoming more and more important due to increasing web-traffic volumes and increasing sophistication of web-page content. A typical web-page will initiate a separate connection for each embedded object, and each of these connections can be considered to be a short-lived flow which spend most of its time in slow-start. In slow-start, the transmission rates are sub-optimal due to small window sizes. Also, due to these small window sizes the susceptibility to retransmission timeouts is higher when there are losses. This is because there may not be enough packets in a window to successfully do a triple-duplicate-acknowledgment loss indication (which is faster).

e) TCP burstiness: With respect to TCP specifically, there is the issue of burstiness. Besides the fact that the acknowledgments (ACKs) are cumulative, one significant cause of such burstiness is ACK compression due to path assymmetry. Remember that ACKs are not subject to flow and congestion control and this in turn can cause the flows on the forward path to be very bursty.

f) The role of AQM with respect to fairness: An AQM that does not maintain any state information will have difficulty maintaining fairness since there is no way for it to

[1]According to [67], the delay budget for interactive audio application is typically 100-150ms out of which 50ms is allocated for network delay

[2]In order to be responsive to congestion, the application layer may have to have a built-in congestion control mechanism[33]. But this tended to be rate-based rather than window-based[22].

differentiate between responsive and unresponsive flows, conforming or greedy flows[16]. However, as suggested by [65], an AQM with per-flow information using a single FIFO queue still cannot guarantee fair bandwidth allocation. Even though the AQM may drop packets from individual flows within the buffer to keep the buffer occupancy among the flows equal, the actual buffer output rate (i.e., bandwidth) for each flow may be unequal. Nevertheless, it seems as though one cannot design an AQM that provides superior fairness which at the same time is simple[20], [70]. In fact, several researchers had recommended that scheduling be combined with AQM in order to provide fairness[29], [68], [23], [70]. For Fair-Queueing coupled with per-flow buffer management it was found that the complexity was $O(\log N)$ and the space requirement was $O(N)$ where N is the number of connections[29]. However, it was observed by [29] that the effect of AQM on fairness (with respect to TCP flows) was much stronger than scheduling. This was due to the fact that even though TCP flows might be allocated bandwidth fairly (through scheduling), some dropping mechanism was still required to enforce fair bandwidth usage. It was found by [68] that there was a partial conflict in goals between queue-based AQM and Fair-Queueing, since the former tried to shorten queues, whereas the latter required longer queues for efficiency, and as a result, rate-based AQM was more amenable to fairness and even QoS guarantees.

So according to [65], there are three types of AQM with respect to fair bandwidth allocation: (1) AQM with no per-flow information, (2) AQM with per-flow information and (3) AQM with per-flow scheduling. The latter, of course will achieve the greatest level of fairness but will be the most complex. The second option is a compromise.

2) QoS: The traditional Internet does not make any QoS committments to users but serves them as best it can given the current conditions, in other words, it provides 'best-effort service'. But as the heterogeneity of applications increases in the Internet so does the demand for QoS guarantees[45], [70]. FTP and email, for example, cannot tolerate errors, but they do not have stringent delay constraints. They may require reasonable to high throughputs. Whereas VoIP, video and other real-time and streaming applications have very tight delay and jitter requirements, yet they tend to be more error resilient[72]. If the Internet must simultaneously support these different applications in an acceptable way, it must be re-engineered to meet their different QoS demands. To this end, the issue of fairness and flow-differentiation is taken to another level. Some researchers have been incorporating AQM into architectures that would provide actual QoS guarantees to Internet users. For example, the Differentiated Services (DiffServ) architecture has AQM as a vital player in QoS provisioning/enforcement.

a) DiffServ: At the network edge, flows are tagged according to Service Level Agreements (SLAs) which define expected performance in terms of delay, jitter, loss and throughput. The core (or DS Domain), the tag on the packet (and without regard to what flow it belongs), will forward it accordingly. So instead of the more complex per-flow QoS, there is aggregate QoS. Each tag type is mapped unto a specific Per-Hop Behaviour (PHB) implemented by the router. The Assured Forwarding (AF) is a standardized PHB which consists of four classes and three dropping probabilities within each class[3]. If the actual sending rate of the user is below the minimum assured rate, packets are marked green; whereas if above the minimum assured rate, either yellow or red[62]. A scheduler allocates bandwidth among the classes, whereas AQM enforces priorities within the class, so that during times of long-term congestion, green packets get the lowest drop rates, and red the highest. According to [39] there is intra-class fairness which depends on the AQM and inter-class fairness with depends on the scheduler. It should be noted that QoS based on aggregates only provides statistical QoS assurance[39]. See also RFC 3246, RFC 3260 of the Internet Engineering Task Force (IETF).

b) The role of AQM with respect to QoS: When using an AQM for QoS in networks, one should consider the following:

- It has been recommended that complex AQM algorithms (i.e., those with a large parameter set) should not be used for differentiated forwarding since for each class of flows an independent set of parameters must be tuned[51], [75], [68], [74].
- There have been good proposals which attempted a compromise between AQM complexity and service differentiation. In order to minimize the use of state information, an AQM may just need to identify flows that are misbehaving. One technique for doing just that is the use of 'bloom filters'. Sometimes it may be necessary just to determine the number of active flows. The adaptive bitmap algorithm is one proposal to this end[23]. It provides high accuracy but has low memory requirements.

AQM for DiffServ is dealt with in greater detail in Section VIII.

V. Random Early Detection (RED)

RED[1] was the first developed AQM scheme to be deployed in TCP/IP networks[16], [30] for the replacement of droptail queues[24], [11]. Its conceptual predecessor, Early Random Drop (ERD), dropped packets with a fixed probability when the queue length exceeded a single threshold[38]. This technique had a number of drawbacks among which was its inability to deal with bursty traffic and misbehaving flows.

RED is a queue-based AQM with no per-flow state[28], [27] and designed heuristically. The initial objectives of RED were to detect incipient congestion[12], to achieve fairness among flows with differing levels of burstiness[45], [42], to control queue lengths to low values[45], [76] so as to minimize queueing delay[11], to prevent correlation of packet drops and global synchronization[53], [55], [76], to minimize packet loss and provide high link utilizations[11].

RED has been the most studied AQM in AQM research thus far[39], [40], [35] and has been the basis for the development of new AQM schemes. This is not only by virtue of the fact that it was the first of its kind in the Internet community but that there were quite a number of problems associated with RED. One of the main problems was the difficulty and uncertainty in tuning its parameters[55], [56], [40], [35], [38], [63], [7], [39], [77], [15] for proper performance. This, by

[3]The Type-of-Service (TOS) field in the IP header is used to indicate the PHB classes and drop precedences[74]

far, negatively affected the widespread deployment of RED. On the other hand, even if RED were tuned properly, the parameters of RED (as will be discussed next) would be highly sensitive to network conditions. Thus, one set of parameter values may work well for a certain load level and traffic mix and may be unfit for a next. This is not desirable, since traffic conditions change rapidly in the Internet[49], [8], [31], [74].

Simulation experiments were the main vehicle used for the parameter tuning of RED. It is inevitable that the set of scenarios for testing would be small. However, in [78] multivariate analysis was performed to statistically analyze a large number of simulation experiments at a time so as to provide a better idea as to the behaviour of these various parameters.

A. The anatomy of RED

The description of RED that follows will be in terms of the main components of any queue management scheme: the congestion indicator, the control function, and the packet dropping/marking algorithm. The drawbacks of RED will also be discussed in terms of these main components. The feedback signal for RED can either be ECN marks or packet drops.

1) Congestion indicator: RED uses the exponential-weighted-moving-average (EWMA) of the instantaneous queue length to determine the congestion level at the queue. This average, updated upon every packet arrival at the queue, is calculated as:

$$\bar{q}(t_{i+1}) \quad = \quad (1-w_q)\bar{q}(t_i) + w_q q(t_{i+1}) \tag{1}$$

where t_i is the arrival time of the i-th packet. If there is congestion at the link, then it is usually assumed, for simplicity, that the arrival time between the packets, denoted by Δ, is constant such that $\Delta = \frac{1}{C}$ where C is the capacity of the link[35]. Therefore the update equation is modified to:

$$\bar{q}(t) \quad = \quad (1-w_q)\bar{q}(t-\Delta) + w_q q(t) \tag{2}$$

and in terms of discrete time the format can be:

$$\bar{q}((i+1)\Delta) \quad = \quad (1-w_q)\bar{q}(i\Delta) + w_q q(i\Delta) \tag{3}$$

Note that, in reality, the actual sampling-interval for the algorithm can exceed or be below the link speed[56].

This EWMA, which is an estimate of the actual average queue length[53], really acts as a low-pass-filter which smoothes out the burstiness of the instantaneous queue length[35], [8] so as to provide a more stable measure. The degree of smoothing is determined by the weight w_q. However, many studies have ensued attempting to quantify the actual role that w_q plays when it comes to RED's stability and responsiveness. It was found that the small recommended value of $w_q = \frac{1}{512}$ caused large oscillations in queue lengths, and bias against bursty traffic flows[11].

Upon closer examination of this queue averaging algorithm, it was shown by [49] that the $(i+1)$-th average update could be expressed in terms of the previous samples as follows:

$$\bar{q}((i+1)\Delta) \quad = \quad w_q \sum_{j=0}^{i} (1-w_q)^j q((i-j)\Delta) \tag{4}$$

By the assumption of fixed time intervals of Δ, the contribution to the $(i+1)$-th average update of that sample made m time-slots previous to it is $(1-w_q)^m$, so that if there is a value a below which the sample is considered negligible, then the number of signicant samples m is given by $m = \frac{ln(a)}{ln(1-w_q)}$ so that the time interval of significance, I, is given by $m\Delta$ and therefore the appropriate value for w_q is $w_q = 1 - a^{(\frac{\Delta}{I})}$[49]. See also[79], [9].

Because of the EWMA queue length, RED is unable to detect incipient congestion due to short-term traffic load changes[11]. See also[80]. It is also possible that EWMA queue length could be much greater than the minimum queue length threshold, while the instantaneous queue length is much less, inducing RED to drop packets unnecessarily[37]. It responds to more long-term traffic changes. It was found that $w_q < 10^{-4}$ is too small for the EWMA to track sufficiently closely the instantaneous queue size[79]. Thus for the case of Web traffic, the average queue size would be more or less constant, whereas the instantaneous queue size varies erratically. In terms of fairness, it was also found that the fairness index remains constant with respect to queue weight during high-load scenarios, whereas it peaks at $w_q = 0.01$ for low and medium loads[79]. The fall-off in fairness index is more dramatic for $w_q < 0.01$ than for $w_q > 0.01$. Additionally, it may seem that larger values of w_q (and hence less averaging) will hinder RED's ability to evenly space packet drops so as to prevent burstiness and global synchronization, but this was found to not be the case[79].

It was also said that the sampling frequency for the queue averaging algorithm (i.e., upon every packet arrival) might be unnecessarily high[49], since an update of once per round-trip-time would still lead to accurate results. in [53], the authors even went further by recommending that the sampling frequency be fixed since they observed that a varying sampling interval (e.g., dependent on the packet arrival rate) leads to oscillations and is therefore harmful to RED's performance. See also[9]. From a practical perspective, having a router compute the queue average on every packet arrival will become more infeasible as the linkspeeds increase to terabit values[35]. But it does not only matter how often the sampling is done. What is actually sampled is important. RED does the sampling on every arriving packet without consideration for the dequeue events. It was noticed by [28] that by not taking into account the dequeue events when there were no arrivals, there would be very sharp discrepancies between the true average queue length and that calculated by RED.

Besides the parameter-tuning issue, another major problem with RED is its direct coupling of queue length as a performance measure with queue length as a congestion indicator. This has made RED's performance (throughput and delay) deteriorate with increasing traffic load[56], [40], [35], [78], [81], [76], [64], [34], [11], [44], [7], [77], [9], [14], [13]. Therefore, as claimed by [56], [41], flows doubly suffer during congestion, in that they incur higher loss and higher delay at the same time. This coupling is explained in [44]. As the number of flows increase (and so the congestion at the queue), the marking probability should increase, but in the case of RED this means that the queue length should be made to increase as well, and this would cause the mean queue length to also increase. What is desired is that the queue length remain at a fixed value regardless of the number of flows (i.e. a more predictable queueing delay). In other words, the performance measure (in this case queue length) should be decoupled from the congestion indicator. It was recommended by [41] that RED should take into account the number of flows sharing the queue. They said that if N TCP flows were sharing the bottleneck link (with no unresponsive flows), then upon packet drop, the arrival rate at the RED queue will drop by a factor of $1 - \frac{0.5}{N}$. If N is large then the overall reduction will be minimal. Therefore RED will not effectively control congestion with increasing N[41]. According to [30], the average queue length in RED is proportional to $N^{(\frac{2}{3})}$ until the maximum queue length threshold is reached.

RED needs even larger buffers for high bandwidth-delay products in order to be stable[35]. EWMA (as discussed before) only aggravates the problem and particularly so when the traffic is bursty[12]. Sufficient buffer room is needed to absorb arriving packets between the time the RED sends the congestion notification and the time the sender actually responds by reducing its rate[10], [12]. If not, RED will deteriorate to droptail, as the queue capacity is continuously exceeded. This is why solely depending on queue length as a congestion indicator is not recommended. The queue length is already non-zero and increasing before substantial congestion notifications are given, therefore the likelihood of buffer-overflow is greatly increased. It has been recommended that RED queues should be configured with capacities twice the bandwidth-delay product[12]. Large buffer space, however, means larger end-to-end delay and jitter. The upside to this, however, is that longer round-trip-times (due to the extra buffering) make TCP sources less aggressive[12]. But increasing round-trip-times (even under static load) can cause instability[52].

In short, factors that affect RED's ability to control the system because of its choice of congestion indicator include: the network load (i.e., the number of connections), the link capacity, and the round-trip-time of connections[42], [82].

2) Control function: Besides the EWMA weight, w_q, RED has three more parameters: the minimum threshold, $\min_{th}$, the maximum threshold, $\max_{th}$ and the maximum non-congestion probability of dropping/marking, $P_{\max}$, at the maximum threshold. If the average queue length is below $\min_{th}$ RED drops no packets. However, if the average queue length increases above $\min_{th}$ but is below $\max_{th}$ RED drops incoming packets with a probability proportional to the average queue length, i.e., a linear function[34], [52], [80]. When the average queue length exceeds $\max_{th}$ all arriving packets are dropped, no exceptions. Figure 5 illustrates the RED control function which is the probability of dropping/marking a packet as a function of average queue length. It is expressed mathematically as:

$$p(\bar{q}) = \begin{cases} 0 & 0 \leq \bar{q} \leq \min_{th} \\ \frac{\bar{q}-\min_{th}}{\max_{th}-\min_{th}} P_{\max} & \min_{th} \leq \bar{q} \leq \max_{th} \\ 1 & \bar{q} \geq \max_{th} \end{cases} \quad (5)$$

By having the two parameters, $\min_{th}$ and $\max_{th}$, RED essentially has a range of reference input values instead of single target value for queue length[49], [14]. This complicates its ability to control queue length especially as a performance measure. The difference between $\min_{th}$ and $\max_{th}$ will determine the size of the oscillation around an equilibrium point[24].

Based on the control-theoretic analysis of RED, the slope of RED's control function will influence the stability and responsiveness of the TCP/RED system. A large slope leads to faster convergence but greater instability, whereas a small slope may lead greater stability but slower convergence[49], [35].

There is a huge discontinuity or jump from $P_{\max}$ to one (1), when the queue length exceeds the $\max_{th}$-boundary[53], [35], and this causes oscillations in the queue length. It was suggested that stability would increase if this discontinuity should be replaced by a more gentle slope (as in Gentle-RED (GRED)).

This discontinuity problem is also compounded by RED's queue length being tied to traffic load. It was shown that under consistently high load (during which time the queue length is greater than $\max_{th}$ much of the time), the equilibrium/long-term drop probability p_0 which is proportional to N^2 (where N is the number of TCP connections) would be larger than $P_{\max}$, so that RED acts just like droptail ON/OFF but switching between a drop probability of one (1) and $P_{\max}$ instead of between one (1) and zero (0)[35]. Therefore, the authors of[35] assert that RED would perform even worse than droptail under these circumstances.

The coupling of queue length as a congestion indicator as well as a performance measure together with its consequent dependence on traffic load can also be discussed here with regards to the control-function. It was observed in [30] that there was a value of N which would cause RED to exceed $\max_{th}$, therefore by reducing $\max_{th}$, the traffic load that RED could support with satisfactory performance was also reduced. However, a low max_{th} relative to the actual capacity of the queue will lead to low-queueing delays and enough room to absorb bursts[12]. But if the EWMA queue length continuously exceeds $\max_{th}$, RED essentially acts as droptail[20]. See also[24]. Additionally, if $\min_{th}$ is too close to $\max_{th}$, RED will basically be droptail and will lead to global synchronization[10]. It is therefore important to incorporate load information into these thresholds[83]. Also if N is large and $P_{\max}$ too small, the congestion feedback from RED will be insufficient to stymie incoming traffic, while in the mean time the round-trip-time is increasing due to filling queues (which may exceed $\max_{th}$ very often), adding to the problem[30], [8]. Presented in [30] was an actual mathematical relationship between average queue length, the number of flows, $\max_{th}$ and $P_{\max}$ given by:

$$\bar{q} = 0.91 N^{\frac{2}{3}} \left(\frac{\max_{th}}{P_{\max}} \right)^{\frac{1}{3}} \quad (6)$$

So, one possible way to maintain the average queue length over varying traffic load would be to adapt $P_{\max}$. Therefore for increasing N, an increasing $P_{\max}$ is needed.

According to [28], the proportional packet dropping in RED did not guarantee that all flows through a RED queue would get a fair share of the bandwidth. RED attempts to drop

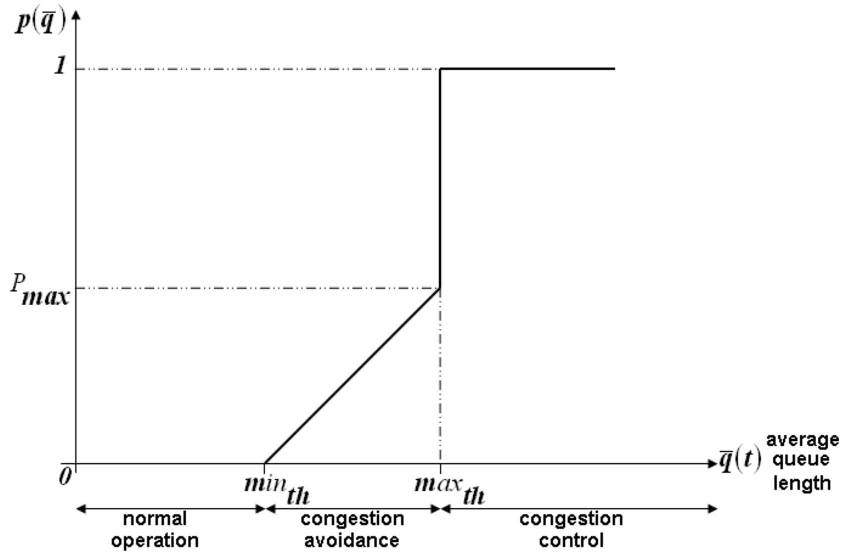


Fig. 5. RED control function

with the same probability all incoming packets (regardless of the number of flows) when the queue is of a certain length. It is therefore assumed that higher bandwidth flows will have a higher proportion of their packets dropped. In fact, the percentage of packets dropped from a given connection in RED is equal to the proportion of bandwidth that that connection would have received using droptail queuing[28]. This is on the long-term. However, on the short-term, all flows, regardless of their arrival rate would see the same drop probability (even flows which use less than their fair share of bandwidth), which would lead to different degrees of flow-rate reductions so that lower-bandwidth flows will suffer most. To exacerbate this situation, unresponsive flows will drive up the drop rates for all flows[28], [70].

3) Packet marking algorithm: Given a marking/dropping probability, RED can actually mark packets according to the Random Marking (RM) scheme, the Uniform Random Marking (URM) scheme or the Wait Uniform Random Marking (WURM) scheme[26]. There is also the Slow Random Marking (SRM)[26]. Now, if p is the given marking probability, the actual probability that a packet will be marked, denoted by p_a, is given by:

1) For RM:

$$p_a = p \tag{7}$$

But, according to [26], the number of unmarked packets between two marked packets is geometrical distributed, i.e., memoryless, therefore one cannot generate a regularly spaced sequence of marked packets using RM. It was already mentioned that a more regular spacing of congestion indications invokes a more gentle TCP reaction. URM seeks to improve on RM and is the algorithm proposed in [1].

2) For URM:

$$p_a = \frac{p}{1-np} \tag{8}$$

where n is the number of unmarked packets since the last packet. The number of unmarked packets between two marked packets is uniformly distributed in $[1, \frac{1}{p}]$, but, according to [26], the queue variance due to URM can be higher than that of RM. WURM is a more mild approach.

3) For WURM:

$$p_a = \begin{cases} 0 & np < 1 \\ \frac{p}{2-np} & 1 \leq np < 2 \\ 1 & \text{otherwise} \end{cases} \tag{9}$$

WURM, however, is a discontinuous function in n and there can be large jumps for certain values of p, which results in the desired marking probability not being realised[26]. For the SRM function, however, there are no discontinuities.

4) For SRM:

$$p_a = \begin{cases} p & np < 1 \\ \frac{p}{2-np} & 1 \leq np < 2 \\ 1 & \text{otherwise} \end{cases} \tag{10}$$

Rapid variations in traffic load can prevent these marking algorithms from uniformly spacing out these congestion indications[12].

Table V outlines some of the shortcomings of RED together with AQM schemes that were subsequently proposed to address the particular shortcoming.

B. Analysis of RED

1) Control theoretic analysis: There has been a substantial number of papers on the analysis of RED using control theory. The main thrust was to decipher its structural problems[49] so as to analytically determine suitable values, or rather, propose more concrete design guidelines for the RED parameters[4]

[4]RED parameters that are typically used in the industry are $w_q = \frac{1}{512}$, $P_{\max} = 1.0$, $\min_{th} = 0.03B$, $\max_{th} = 3\min_{th} \approx 0.1B$, where B is the packet rate

TABLE V
SUMMARY OF SOME OF RED'S SHORTCOMINGS AND RESULTANT AQM SCHEMES THAT ATTEMPT TO OVERCOME SUCH SHORTCOMINGS

RED Shortcomings	Proposed Solution	Resultant AQM scheme
Congestion Indicator		
• EWMA queue length unable to detect incipient congestion due to short-term traffic load changes	Use other congestion indicators apart from or in coordination with EWMA queue length	
• EWMA queue length can induce RED to drop packets unnecessarily	Use instantaneous queue length. However, in conjunction with other congestion indicators	SRED, LRED, MRED (version 2)
• EWMA reduces the responsiveness and stability of RED in high bandwidth-distance product flows and bursty flows		
• Small ω_q can cause instability in queue lengths		
• ω_q affects fairness index for low and medium loads		
• Sampling upon every packet arrival is unnecessary • A variable sampling interval leads to oscillations in queue length	Provide a fixed sampling interval the size of which is dependent on R_0^+, N^- and C	Proportional Integral (PI)
• Sampling that only considers packet arrivals and not dequeue events causes average queue length to deviate far from true length	Include into congestion indicator the dequeue events	FRED
• Direct coupling of queue-length as a congestion indicator and performance parameter	Provide a fixed queue length for predictive queuing delay regardless of the number of flows	PI, PID, DRED
• Reduction in source rates not profound. For N TCP flows (with no unresponsive flows) the reduction is $1-\frac{0.5}{N}$		
Control Function		
• Effectively RED has a range of reference queue lengths between $\mathbf{min}_{th}$ and $\mathbf{max}_{th}$ instead of a single target • The difference between $\mathbf{min}_{th}$ and $\mathbf{max}_{th}$ determine the size of oscillation around an equilibrium point • If $\mathbf{min}_{th}$ is too close $\mathbf{max}_{th}$ RED will lead to global synchronization • If $\mathbf{max}_{th}$ is too low RED essentially acts as droptail	Specify a single reference queue-length	Control-theoretic approaches: PI, PID, etc. DRED, ARED
• Non-adaptive, manual, static parameter tuning • No load information associated with thresholds • If $P_{\max}$ is too small and N is large there would be insufficient congestion feedback.	Make parameters adaptive according to changing network conditions (e.g., load).	ARED, GREEN, BRED
• The larger the slope of RED's control function the faster the convergence but less stability		
• The discontinuity causes oscillations in queue length	Remove discontinuity	GRED
• Proportional packet dropping does not guarantee fair share of bandwidth • Unresponsive flows drive up drop rates for all flows	Additional mechanisms to improve fairness, e.g., incorporating flow information	FR,SHRED, BRED,BLACK, Stochastic Fair Blue.
Feedback Signal		
• For a given dropping/marking probability, the uniformity of packet drops may not be realised	Improve the spacing between packet drops	DREAM

for improved stability and responsiveness[55], [48], [24]. They seem to agree that RED is a delay-independent P-type controller (i.e., Proportional controller) which because of this, has a number of control limitations[48], [14], [13] and is not sufficient to stabilize the system. These modeling efforts have also given clearer understanding of RED's behaviour with changing network conditions such as round-trip-times, flow-count (or load), and link capacities[77].

The following transfer-function model for the RED controller was presented by [55]:

$$C_{red}(s) = \frac{L_{red}}{\frac{s}{K_p}+1} \qquad (11)$$

where $L_{red} = \frac{P_{\max}}{\max_{th} - \min_{th}}$ and $K_p = \frac{log_e(1-w_q)}{\Delta}$. For a desired stability margin, values of L_{red} and K_p are found, from which the RED parameters, w_q, $\min_{th}$, $\max_{th}$ and $P_{\max}$ are determined. So, in effect, RED is a proportional gain in conjunction with a low-pass filter[46]. We see here that RED can never achieve zero steady-state error[53], [64]. Also, for reasonable stability, the TCP/RED closed-loop system is very sluggish. In fact, the responsiveness of the low-pass filter portion of RED depends on the magnitude of K_p[53]. The higher the value of K_p the more quickly will RED respond to changes to the point where RED will be tracking the instantaneous queue length very closely[53]. It is suggested that RED should be modified to lead-compensation which will result in the classical proportional-integral (PI) controller[55]. It compensates explicitly for feedback delay using a knowledge of round-trip-time, link capacity and the number of active flows[48].

Looking more closely at the variable K in the transfer-function model of RED, one can deduce that to increase K (and hence its responsivenes and instability), one can decrease w_q or Δ. But Δ is not only dependent on input packet rate or link speed (as was briefly shown earlier), but also on the average packet size which cannot be controlled by network operators. It is therefore desirable that Δ is made independent of all these factors. Delving further, it can be seen that a changing Δ will change the "forgetting" rate among the queue samples[53], [79], even if w_q were held constant, thus causing the queue averaging algorithm to track very closely at times the instantaneous queue length, while at other times lag significantly behind.

2) Assumptions for analysis: For the RED analysis a number of assumptions were typically made, for example in [76]:

- All the flows are TCP and are in TCP congestion avoidance phase and change their windows synchronously.
- There are no retransmission timeouts, so that all losses are detected only by triple-duplicate ACKs.
- The average queue length for RED is always between $\min_{th}$ and $\max_{th}$, i.e. the linear portion of the control function.

Besides these the following assumptions were also made[49]:

- The number of flows, n remain constant over a long time frame.
- All the flows have the same round-trip time and the same average packet size.
- The receiver advertised window is large enough so as to not inhibit flow rates.

There will be a more in-depth discussion with regard to control-theoretic AQM design in Section VI.

3) Analysis that includes non-responsive and short-lived flows: It was shown that the TCP/RED system can attain equilibrium if there is a unique solution of the following system of equations: (1) $\bar{q} = G(p)$ - the queue law and (2) $p = H(\bar{q})$ - the drop module (or feedback control function)[49]. The system will operate at this equilibrium point on average.

The work of [49] was extended by [25] by developing a queue-law that supports ECN and non-ECN traffic so as to determine the impact of ECN traffic on AQM. The TCP fluid-model was modified by [33] so as to investigate the effects of nonresponsive flows on long-lived flows. They also developed a model for short-lived flows (using a shot-noise process).

4) Nonlinear analysis: A more accurate representation would be to consider that RED is a low-pass-filter followed by a nonlinear gain map[81] (as per the control function). A straight, non-linear analysis of the TCP/RED system was attempted by [82] and in addition to measuring the nominal values of the average queue length, throughput and packet loss rate of the system, they measured the maximal Lyapunov exponent and Hurst parameter for the average queue length. This was to determine how much the performance of the system depended on its nonlinear dynamical behaviour as its parameters varied. The Hurst parameter is typically used to measure the long-range dependence (LRD) of a time-series. The autocorrelation of an LRD time-series dies more slowly with lag time than an ordinary time-series. The Lyapunov exponent is used to indicate whether a system exhibits deterministic chaos, i.e., a system whose behavior is unpredictable after long time-scales. They found that the maximal Lyapunov exponent increased as the value of the EWMA weight w_q increased, although the Hurst parameters had no significant change. They performed tests for one-way FTP traffic as well as for two-way FTP traffic, and found that the latter yielded greater instability in queue lengths as well as higher maximal Lyapunov exponents. The case of two-way Web flows was also investigated. They did not report results for other RED parameters such as $\min_{th}$, $\max_{th}$ and $P_{\max}$.

5) Analysis that include multiple bottlenecks and heterogeneous delays: The issue of multiple-bottlenecks was pursued through theoretical analysis by [64]. They then verified RED stability using Nyquist plots. This problem was also tackled by [46] , using the TCP fluid model proposed in [53] and extending the network model to that of multiple bottleneck links and heterogenous delays, and by this approach, the term: "gain of routing", emerged. It should be noted that the TCP fluid model of[53] was extended by [84] to include UDP traffic.

C. RED variants and new AQM schemes

Although RED is a significant improvement over simple droptail, there were still problems of low throughput, high delay and jitter, instability and parameter configuration[80]. It is stated that even though RED may reduce average delay, the jitter actually increases and the number of consecutive packet drops gets higher with RED than with droptail[62]. Additionally, as stated by [62] RED is not a clear winner over droptail with respect to Web traffic.

Therefore, after RED (which has essentially become the de facto standard for AQM[15]), there have been two main directions in AQM design[43]: (1) to improve on some aspect of RED (heuristically or by control-theory)[55], [56], [43], [42], [7] or (2) to build an entirely new algorithm (heuristically or by control-theory). According to [36], up to that time of writing, there were over fifty AQM schemes since the original RED proposed in the literature. However, the problem with any subsequent design that is heuristic, whether it be RED-based or completely novel, would be the difficulty to understand the effects of its parameters on performance (stability etc.) and consequently, the challenge to tune its parameters, just like RED. Eventually some measure of theoretical analysis would have to be performed[43], [48], [34], [7].

Several RED variants have made marginal changes to some component(s) of RED while others substantial changes[9]. Their aim was to improve on the weaknesses of RED with respect to fairness, throughput, delay, drop rates[7], stability. However, one of the key weaknesses addressed by quite a number of them was to specify automatically tuned parameters[15], [7], [14] so that the AQM could be adaptable to changing network conditions, thus making them more robust. Many, though, still use queue length as the sole congestion indicator[15]. Table XIV to Table XXVII in the Appendix provide summaries of a number of RED variants and heuristic AQM schemes.

D. RED deployment

According to [7], of all the new AQM schemes proposed (RED variants included), the only two that had been implemented in actual routers (at the time) had been Gentle-RED and Parabola-RED. This was because many of these AQMs have high computational complexity and can require significant changes to the router's software architecture.

Upon examining recent datasheets of routers produced by some of the major equipment manufacturers, it was found that WRED (Weighted RED) (used in the DiffServ context for QoS provisioning (to be discussed later)) is the predominant

AQM scheme implemented. Examples of routers with WRED include:

- Alcatel-Lucent OmniAccess 5510 and 5740 unified services gateway (Release 3.0)
- Brocade MLX Series - Multiservice IP/MPLS Routers
- Cisco ASR 9000 System Aggregation Services Routers
- HP MSR50 Series
- Juniper Networks M7i and M10i Multiservice Edge Routers

VI. Control Theoretic Approaches

In this section we introduce the control-theoretic approach to AQM design. We also discuss a few algorithms that have emerged from the realm of control theory. In Section V we had briefly stated some results from the control-theoretic analysis of RED.

A. The feedback control system

Using control theory terminology, the network can be seen as a "feedback control system" in which the endpoints and routers (in aggregate) constitute the "plant" which has to be controlled. The AQM acts as the controller or "compensator"and signals to the plant the "control signal" which, in this case, is the drop-probability, so as to control the source rates. The reference input to the system could be, for example, the target queue length to which the control system aims to keep the actual queue length (the "controlled variable"). The difference between the reference value and the actual value of the controlled variable is called the "error signal". It is the error signal (congestion indicator) that drives the AQM compensator to control the plant[42], [55], [49], [11], [43]. Figure 6 illustrates these concepts.

In control theory the main performance measures for the AQM would be transient response, stability and steady-state error. Control theory is a very mature field of study which over the years has produced so many well established tools (e.g root locus, Bode plots, etc.) to not only analyse control systems but to design such systems within tolerable margins of these performance measures. Many papers generated so far on theoretic AQM design have utilized several of these tools. The transient response measures how quickly the system output reaches target (input signal). The stability measures the extent of oscillation (in magnitude and growth) of the output around the target, and the steady-state error measures to what degree (in the long-term) the output has reached the target. Terms such as phase-margins and gain-margins provide a measure of how easy it is to send the system into instability, i.e. its robustness. As mentioned before, a system should be robust to variations in system parameters (e.g. the number of flows) as well as model errors[55], [14], [13]. The wider the stability margins the better, but typically wider margins mean slower responses. If the transient response is sluggish, then it will be difficult maintain the controlled variable around the target when the network conditions change frequently[43]. However for faster response, one will have to tolerate larger degrees of oscillations (e.g. in queue lengths) and for networks this means more delay jitter.

B. Classical control theory

In classical control theory, the transfer function approach (i.e., analysis in the frequency domain) is used, whereby the product of the Laplace Transform of the compensator and the plant (represented by the TCP fluid model mentioned before) is called the open-loop transfer function, and the ratio of the Laplace Transform of the system output (e.g. actual queue length) to the system input (i.e. reference queue length) is the closed-loop transfer function. By control theory, it is the poles of the closed-loop transfer function that determine the stability and the transient response of the system[76]. The Laplace Transform of the compensator itself is really the transfer function from the congestion indicator (or error between the target and the actual queue length or arrival rate) to the dropping probability. The Laplace Transform of TCP endpoints is the transfer function from the dropping probability and the window size (or sending rate). The Laplace Transform of the queue is the transfer function from the arrival rate to the queue length. There is typically a delay of one round trip time, R_0 between the compensator and the plant, and its Laplace Transform is e^{-sR_0}.

Early work on AQM design using classical control theory, considered only a single congested link and synchronised sources (i.e. all flows had the same round-trip-time). In addition, it was assumed that all the sources, the number of which did not change with time, had the same window progression and that the window sizes were the same at all times, thus all the sources could be treated as one huge source scaled by N, the number of sources. Even though these assumptions may seem far-fetched, the resulting stability analysis did yield some plausible explanations to the various system behaviours observed in previous heuristic designs. For example, it was shown in [79] that using queue averaging reduces the responsiveness of the controller and reduces system robustness. It was shown by [55] that the high frequency plant gain was inversely proportional to TCP load (i.e. the number of connections N), so that for very low loads, the stability margin decreased (more oscillatory action) and the responsiveness increased. Also the delay between the AQM compensator and the TCP plant, puts a hard-limit to the speed of response for a stable system to $\frac{R_0}{2}$[55].

There has been further work, using classical control theory, for networks with arbitrary topologies and heterogenous delays[46]. The approach in [46] in dealing with arbitrary topologies was to use a single value called the gain of the routing matrix that represented the topology. It was found that as the gain of the routing matrix decreases, the stability margin of the system also decreases. So far, no work has been found that deals with the number of TCP sources varying with time.

Modern control theory uses a more rich characterization of the system called the state-space model. As an example, this state-feedback control approach is used in [48] to analyse AQM in TCP networks.

We now discuss a few AQM schemes that have been designed according to control theory.

C. Proportional-Integral (PI)

The authors of [55] provided a transfer-function model for RED together with design rules for tuning the RED

$$p(kT) = a\left(q(kT) - q_{ref}\right) - b\left(q((k-1)T) - q_{ref}\right) + p((k-1)T) \quad (12)$$

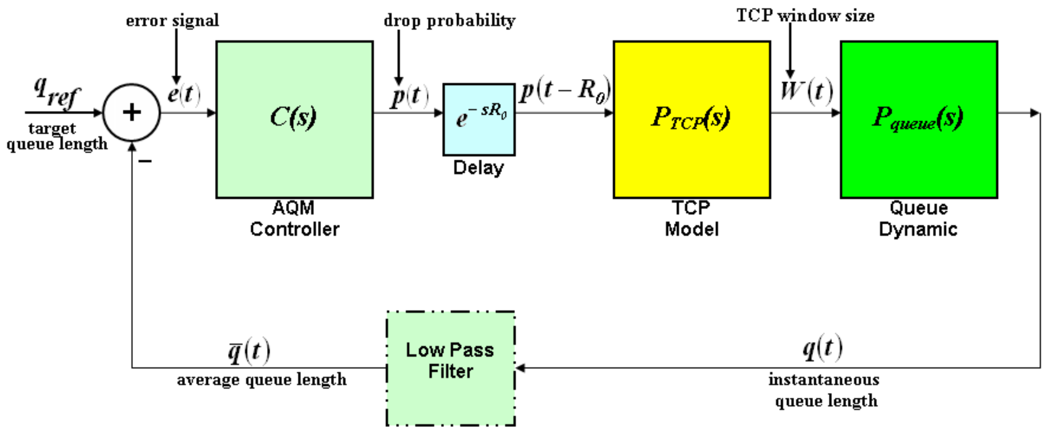


Fig. 6. AQM design using classical control theory

parameters. They also performed analyses which led them to suggest lead compensation into RED so as to improve its response in terms of stabilizing the feedback control system. The AQM that resulted was the Proportional-Integral (PI) algortihm, outlined in [56], and the digital form of which was as in Eq. (12) above, where $p(kT)$ is the dropping probability at sample time kT, a and b are PI coefficients based on the Bilinear Transform (a.k.a. Tustin's method), and q_{ref} is the reference queue length. Now, $T = \frac{1}{f_s}$ where f_s is the sampling frequency which should be be 10 to 20 times $\frac{\omega_g}{2\pi}$ where $\omega_g = \frac{2N^-}{\left(R_0^+\right)^2 C}$ where N^- is the minimum number of TCP sessions, i.e., the load factor, C is the link capacity and R_0^+ is the maximum round-trip time chosen.

The authors of [85] claimed that the PI controller designed in [56] might have a large phase margin which led to sluggish behaviour. Their approach instead was to design the PI controller according to phase margin specifications. As a follow-up, a self-tuning PI TCP flow controller was designed in [86] to deal with varying traffic load in networks. In [87] a self-tuning PI controller was designed according to gain margin specifications.

D. Proportional-Derivative (PD)

In [40], a Proportional-Derivative (PD) controller was added to the original RED AQM. Hence the name PD-RED. The PD controller adapted the RED parameter $P_{\max}$ which the authors believed would more systematically improve queue length stabilization attempted by Adaptive RED (ARED). The error signal during sampling interval i is given by

$$e(i) = \bar{q}(i) - q_{ref} \quad (13)$$

where $\bar{q}(i)$ is the average queue length during sampling interval i and q_{ref} is the target queue length. The PD controller algorithm follows:

$$P_{\max}(i) = P_{\max}(i-1) + K_p\frac{e(i)}{B} + K_d\frac{(e(i)-e(i-1))}{B} \quad (14)$$

where K_p, K_d and B are the proportional gain, derivative gain and buffer limit, respectively. Simulations were run comparing ARED with PD-RED.

Based on state-space model, a PD-type state-feedback AQM in terms of queue-length was derived by [48] to completely replace RED. The model also compensated for large delay. A more thorough treatment on this approach can be found in [88]. Another PD-Controller AQM scheme was proposed by [89] similar to that of [40]. However, instead of adapting the RED $P_{\max}$ parameter and working in conjunction with RED, it itself calculates the dropping probability according to the following algorithm

$$p(i) = p(i-1) + K_p\frac{e(i)}{B} + K_d\frac{(e(i)-e(i-1))}{B} \quad (15)$$

where $e(i)$ is as defined before, and K_p, K_d and B are the proportional gain, derivative gain and buffer limit, respectively. In order to maintain high-link utilization, no packet is dropped if the instantaneous queue length, $q(i) < L$ where L is the No-drop threshold.

E. Proportional-Integral-Derivative (PID)

To provide a faster response than that which PI controller can afford, the Proportional-Integral-Derivative (PID) controller was proposed by [43] as another viable AQM. The dropping probability is adjusted according to (16) on the following page, where $T_i = \frac{K_p}{K_i}$, $T_d = \frac{K_d}{K_p}$, T is the sampling time, and K_p, K_d, K_i are the proportional gain, derivative gain and integral gain respectively. In [43], guidelines based on gain margin and phase margin are provided to tune the parameters K_p, K_d, and K_i. Another attempt to design a PID-controller for AQM can be found in [90].

F. Robust control

To rigorously address the issue of large delays in networks, the DC-AQM algorithm based on internal mode compensation

$$p(i) = K_p \left\{ \left(1 + \frac{T}{T_i} + \frac{T_d}{T}\right) e(i) - \left(1 + \frac{2T_d}{T}\right) e(i-1) + \frac{T_d}{T} e(i-2) \right\} \tag{16}$$

(IMC) principle in control theory was proposed by [42]. From the IMC controller derived, the corresponding feedback controller was solved, the structure of which is that of the classical PID controller. This time, the parameters K_p, K_d, and K_i are tuned according to the following:

$$K_p = \frac{2T + L}{2.6KL} \tag{17}$$

$$K_i = \frac{1}{1.3KL} \tag{18}$$

$$K_d = \frac{T}{2.6K} \tag{19}$$

where $T = \sqrt{T_1^2 + T_2^2}$ and $L = T_1 + T_2 + R_0 - \sqrt{T_1^2 + T_2^2}$, $K = \frac{(R_0 C)^3}{4N^2}$, $T_1 = R$ and $T_2 = \frac{R_0^2 C}{2N}$ where R_0 is round-trip time, C is link capacity and N the number of active sessions. (See Table I for nomenclature.) To also contend with large delay, a Gain Adaptive Smith Predictor with PI controller (GAS-PI) was devised to improve robustness in [91]. The authors claim that the GAS-PI algorithm for AQM outperforms other AQM schemes (i.e., RED and PI) even with dynamically changing traffic loads. In [92], IMC was combined with a Smith estimate controller for AQM to also tackle the large delay issue in networks. See also [93] for an AQM called Robust Smith Predictor based on both the Smith predictor and IMC. In [94] a robust H_∞ optimal AQM controller was designed to deal with the delay. For a μ-optimal controller design for AQM, see [95].

In [38] a predictive PID controller is proposed. They used the generalized predictive control (GPC) method to determine suitable values for K_p, K_d, and K_i so as to make the system more robust to changes in model parameters such as load, round-trip time etc. In [96] an adaptive GPC (AGPC) was devised for AQM. A PI-PD controller was proposed in [63] which the authors claimed would provide robust and predictive control. The design of this controller consists of two parts: the proportional-integral (PI) for stability and the proportional-derivative (PD) for responsiveness. An alternative PI-PD controller was developed by [97]. In this case, the PD controller is used in the inner feedback loop to directly determine the packet dropping/marking probability.

G. Fuzzy Logic

In [98], an AQM scheme based on a variable-length virtual-output-queue fuzzy-congestion-control mechanism was proposed. According to [98], fuzzy logic control provide better control of the nonlinear, time-varying systems since it can dynamically adapt its parameters which for other control types would have to be held constant in spite of time-varying traffic load, roundtrip times, etc. One can remember that one had to first linearize the non-linear time-varying TCP/IP plant before applying classical control techniques. Fuzzy control obviates this need. According to [99], fuzzy-logic control does not need a precise model. An overview of the use of fuzzy logic in telecommunications networks was presented in [100].

Fuzzy Control RED (FCRED) was proposed in [101]. It consists of a fuzzy controller adjusting the $P_{\max}$ parameter of the RED algorithm. This paper also gave a brief summary of fuzzy control theory. The fuzzy controller comprise three parts: the fuzzification unit followed by the fuzzy-inference engine with fuzzy-rule base and then a defuzzification unit. The fuzzification module maps the input values to be controlled to a fuzzy set (i.e., membership functions[5]). The fuzzy-rule base provides the connection between the input signals and the appropriate output variable. It is constructed based on a combination of trial and error (with the aid of a domain-expert's knowledge and experience) and theoretical methods (to fine-tune the parameters)[102]. It consists of a set of IF-THEN rules. The defuzzification unit maps this fuzzy output variable (which has an associated membership function) to a crisp controller output. According to [103], defuzzification methods include: center of area (CoA), center of maximum (CoM) and mean of maximum (MoM) that the plant understands. See Figure 7 for the general structure of a fuzzy-logic controller.

A fuzzy controller based on RED was also developed in [102], the input variables of which were the average queue length and the packet loss rate, and the output the packet dropping probability. Another RED-based fuzzy controller was developed in [104] which dynamically tuned the maximum drop probability parameter ($P_{\max}$) of RED. The inputs to this controller are $P_{\max}$ and the error signal, $e = \bar{q} - \frac{\max_{th} + \min_{th}}{2}$. The output of the controller is $\Delta_{P_{\max}}$, i.e., the change in the maximum drop probability parameter, determined after every 0.5 seconds. AFRED[99], an adaptive Fuzzy-based AQM which also has Gentle RED at its core, also adapts the fuzzy rule and parameters in membership functions so as to improve stability. Therefore, it not only consists of the fuzzy-control module but also an "adaptive adjust module". AFRED uses as its input variable the instantaneous queue length. Its output is packet drop probability. It measures the real packet drop ratio so as to determine when the fuzzy rules should be changed. Based on the Random Early Marking (REM) algorithm, a Fuzzy Logic Controller called FUZREM was designed in [105]. And just like AFRED, there is AFREM (Adaptive fuzzy REM) which adapts the fuzzy rules for fuzzy REM[106]. There is also fuzzy GREEN[103]. DEEP BLUE[107] consists of combining the BLUE AQM algorithm with the fuzzy extention of Q-learning, a reinforcement learning technique, that provides online model-free optimization. Then there is the Fuzzy Logic Congestion Detection (FLCD) algorithm which has embedded within it, fuzzy CHOKe[108].

In [109], a fuzzy-based proportional-integral-derivative (PID) controller AQM was presented. It consisted of a fuzzy PID controller working in conjunction with a conventional PID controller via switching mechanism. The fuzzy PID controller

[5]For computational simplicity triangular and trapezoidal membership functions

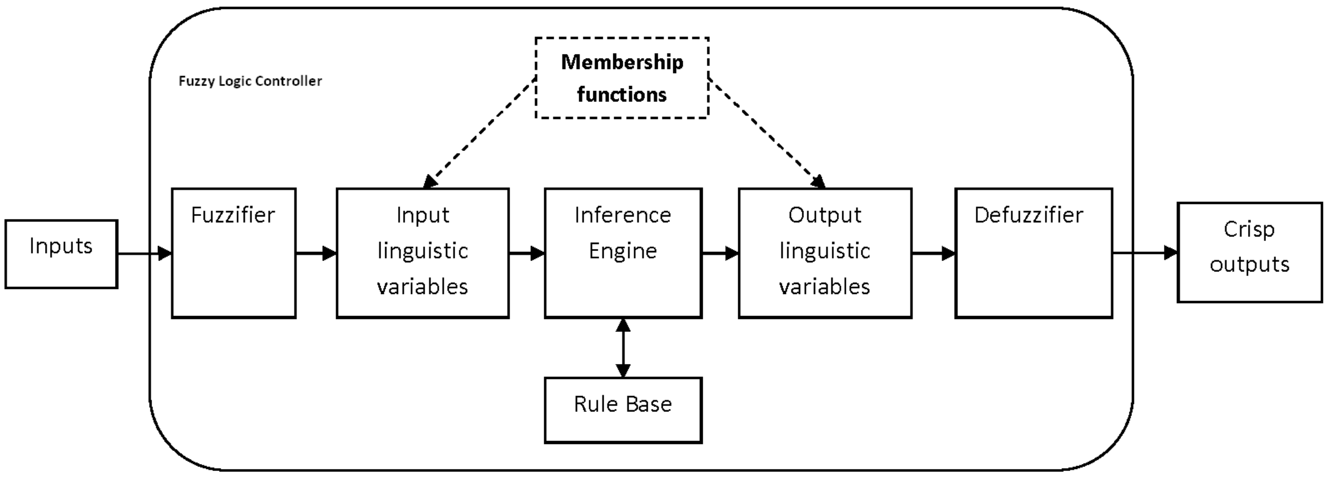


Fig. 7. Fuzzy-logic controller structure

consists of two intputs (conventional error and the rate of change of error between queue length and the target queue length), and a single rule base composed of 49 rules. A fuzzy controller which not only had an input variable queue length error but also the error between link input rate and link capacity, was presented in [110]. Its output variable was the dropping probability. Returning to [111], a fuzzy-logic controller was designed to adjust the maximum queue limit of virtual output queues so as to induce packet drops in the real queue. In [112] was proposed a self-adaptive fuzzy controller that calculated the learning rate for a neural-network-based PID controller that was previously presented in [113]. For further work on fuzzy control for active queue management see [114], [115], [116], [117], [118], [119]. In [120], a wavelet neural network controller for AQM was presented.

H. Summary

Tables VI to IX provide a summary of the design and performance considerations of a number of control-theoretic AQM schemes sorted according to their theoretical basis and then according to the year they were proposed (so as to provide some form of chronology). It can be seen that as the years progress, the main direction has been to more and more sophisticated robust-control techniques, with some classical techniques interspersed. For the most part, the linearized TCP model by Hollot et al.[55] continues to be at the core of such algorithms. In these tables one can also find AQM schemes proposed by [121], [122] and [123].

Tables X to XII also provide a summary of fuzzy-based active queue management schemes proposed over the years. There was found no definite trend so to speak. The number and type of inputs were quite variable. The same could be said about the output variables, the rule-base and the defuzzification methods chosen. There was found no study to show which defuzzification method or inference engine may be most suitable for the AQM context. From the tables one can also see that some fuzzy-logic controllers were AQM schemes in themselves while others worked along with existing AQM schemes such as RED by fine-tuning the latter's parameters.

VII. Deterministic Optimization Approaches

Besides heuristic and control-theoretic approaches to AQM design, there is the deterministic optimization approach in which the AQM scheme, in conjunction with the source algorithm, explicitly attempts to drive the network to a globally optimal operating point[59]. Early contributors to line of AQM research have been Kelly and Low, among others.

A. The optimization problem

The optimization problem, which is a nonlinear programming problem that assumes perfect fluid flows[124], is usually formulated as follows[59], [125], [126], [57], [58]:

Consider an arbitrary network consisting of a set $\mathcal{L}$ of links: $\mathcal{L} = \{1, ...l, ...L\}$ each with finite capacities $c_l, l \in \mathcal{L}$ and a set $\mathcal{S}$ of sources: $\mathcal{S} = \{1, ...s, ...S\}$ each with a transmission rate of $x_s(t)$ at time t in packets per second. Let $x(t) = [x_1(t), ..., x_S(t)]^T$ be the vector of all source rates. Now each source uses a set $\mathcal{L}_s \subseteq \mathcal{L}$ of the links. The set of sources that use link l is denoted by $\mathcal{S}_l = \{s \in \mathcal{S} | l \in \mathcal{L}_s\}$. Each link has a scalar congestion measure called the "price" per unit bandwidth, denoted by $p_l(t)$. (Let $p = [p_1(t), ..., p_L(t)]^T$ be the vector of all link prices.) The $L \times S$ routing matrix $\mathcal{R}$ can more conveniently depict these source-link interdependencies where

$$\mathcal{R}_{ls} = \begin{cases} 1 & \text{if } l \in \mathcal{L}_s \text{ or } s \in \mathcal{S}_l \\ 0 & \text{otherwise} \end{cases} \quad (20)$$

Hence, the aggregate source rate at link l is $y_l(t) = \sum_s \mathcal{R}_{ls} x_s(t) = \mathcal{R}x(t)$ and the end-to-end congestion measure for source s is $q_s(t) = \sum_l \mathcal{R}_{ls} p_l(t) = \mathcal{R}^T p(t)$.

When source s transmits at a rate $x_s(t)$ it attains a utility $\mathcal{U}_s(x_s)$. It is assumed that this utility function is increasing, strictly concave and twice continuously differentiable with respect to $x_s(t)$ and that the utilities of the sources are additive, i.e., $\sum_s \mathcal{U}_s(x_s)$.

TABLE VI
SUMMARY OF CONTROL-THEORETIC AQM ALGORITHMS

AQM	Year	Compared with	Control Type	Controller Parameters	Plant Model	Control signal	Controlled signal	Feedback signal	Performance tests	Traffic types	Simulator
PI [56]	2001	RED	Classical	Static values C=3750 pkts/s; N^{-}=60; R^{+}=0.246 s	Linearized TCP/AQM model by Hollot et al.	$p(t)$	$q(t)$	$q(t)$	$q(t)$and $p(t)$ versus time, t; Varying N in a given run; Not varying N in a given run; Utilization versus q_{ref}; Queuing delay versus q_{ref};	FTP with HTTP	ns
PID [43]	2003	PI	Classical	Static values C=10^5 pkts/s; N =30; R =0.03 s	Linearized TCP/AQM model by Hollot et al.	$p(t)$	$q(t)$	$q(t)$	$q(t)$and $p(t)$ versus time, t; Varying N in a given run;	FTP	ns2
Predictive PID [90]	2003	RED, PI	Classical	Static C=3750 pkts/s; N =60; R =246 ms	Linearized TCP/AQM model by Hollot et al.	$p(t)$	$q(t)$	$q(t)$	$q(t)$ versus time, t; Distribution of queue lengths Quadratic average of control deviation (QACD) ; Packet loss rate versus time.	FTP with short-lived flows	ns-2
PI based on Phase Margin [85]	2004	RED, P, PI	Classical	Static values C=10433 pkts/s; N =100; R =0.25 s	Linearized TCP/AQM model by Hollot et al.	$p(t)$	$q(t)$	$q(t)$	$q(t)$ versus time, t;	FTP with HTTP	OPNET
Self-tuning PI [86]	2004	RED, PI	Classical	Parameters based on estimate of N, R and C	Linearized TCP/AQM model by Hollot et al.	$p(t)$	$q(t)$	$\bar{q}(t)$	$q(t)$ versus time, t; Varying N in a given run;	FTP with HTTP	OPNET
Self-tuning AQM [87]	2007	RED, PI	Classical	Parameters based on estimate of N, R and C	Linearized TCP/AQM model by Hollot et al.	$p(t)$	$q(t)$	$q(t)$	$q(t)$ versus time, t; Varying N in a given run; Varying RTT in a given run	FTP with HTTP	OPNET

The primal problem then becomes

$$\text{maximize } \sum_{s} \mathcal{U}_s(x_s) \tag{21}$$

$$\text{subject to } \sum_{s} \mathcal{R}_{ls} x_s(t) \leq c_l \quad l = 1, ..., L \tag{22}$$

The solution to this optimization problem would be a vector of source rates, x, that will maximize the objective function ($\sum_s \mathcal{U}_s(x_s)$), i.e., the sum of all the utilities in the network, subject to the constraint that the aggregate source rate at each link does not exceed the link capacity. Because of the assumptions made for the properties of the utility functions, a unique optimal solution does exist for this primal problem. There are, however, two issues. Firstly, the utility functions may not be known by the network and they may differ among the sources[57]. Secondly, solving the primal problem directly, according to [59] will require coordination among potentially all the sources (a complex and impractical situation). This is because the source rates are coupled by the constraint, although they are separable in the objective function.

The more appropriate line of attack would be a decentralized, distributed solution, for which each source observes its own rate and end-to-end congestion measure with no other knowledge about other sources and links in the network, and for which each link observes its own aggregate rate and independently sets its own price, without explicit knowledge of other links and sources in the network[125].

It should be noted that from the formulation it is only the rates that are directly modeled and not queue lengths[57], therefore any AQM algorithms derived directly from the framework outlined will be primarily rate-based.

According to [59], the conceptual difference between Low's approach and Kelly's approach is that in the case of Kelly, the source decides how much it is willing to pay over each unit of time, and the network allocates its rate accordingly, whereas in Low's case, the source tells the network the rate it wants, and the network charges a price accordingly.

Out of this work on deterministic optimization for active queue management, the Random Early Marking (REM), the Adaptive Virtual Queue (AVQ) and the Stabilized Virtual Buffer (SVB) algorithms were born.

B. Adaptive Virtual Queue (AVQ)

Adaptive Virtual Queue (AVQ) is a rate-based AQM based ([14], [13], [32], [127], [34], [52], [45]) that attempts to maintain input (arrival) rate at a desired utilization. According to [48], AVQ like RED is a delay-independent control since it does not compensate for delay in the congestion feedback.

As mentioned before, AVQ is based on the penalty function approach to optimization as proposed by Kelly et al.[128]. The form of the penalty function is derived from loss probability for a $M/M/1/B$ which is $p_l(c_l, y_l) = \frac{(1-\rho)\rho^B}{1-\rho^{B+1}}$ where B is the buffer limit, and $\rho = \frac{y_l}{c_l}$ is the link utilization. The arrival

TABLE VII
SUMMARY OF CONTROL-THEORETIC AQM ALGORITHMS CONT'D

AQM	Year	Compared with	Control Type	Controller Parameters	Plant Model	Control signal	Controlled signal	Feedback signal	Performance tests	Traffic types	Simulator
PI-PD [97]	2010	PI	Classical	Static C=3750 pkts/s; N =60; R =0.246s	Linearized TCP/AQM model by Hollot et al.	$p(t)$	$q(t)$	$q(t)$	$q(t)$ versus time, t; Varying N in a given run; Addition and removal of UDP flows	FTP UDP	ns-2
PD-RED [40]	2003	ARED	No details	Static values set apriori	TCP – no details	P_{max}	$q(t)$	$\bar{q}(t)$	$q(t)$and $p(t)$ versus time, t; Varying N in a given run	FTP	ns
PI-PD [63]	2004	RED, PI	Did not say how parameters were tuned	Static C=3750 pkts/s; N =9; R =120 ms	---	$p(t)$	$q(t)$	$q(t)$	$q(t)$ versus time, t; Quadratic average of control deviation (QACD) versus number of flows Packet loss rate versus time.	FTP with short-lived flows	ns-2
PD [48]	2003	RED-like AQM (linear model)	Modern (state-space model) with delay compensation and LQ stabilized gains	Static values C=4000 pkts/s; N =100; prop. delay = 150 ms; 350 ms	Linearized TCP/AQM model by Hollot et al.	$p(t)$	$\bar{q}(t)$	$\bar{q}(t)$	$q(t)$ versus time, t; Varying propagation delay for different runs	TCP Reno	ns
SFC [121]	2003	RED, SRED, PI, AVQ	Modern control (state feedback)	Static C=1250 pkts/s; N =300; R =0.6s	Enhanced TCP model	$p(t)$	$q(t)$ $\dot{q}(t)$	$q(t)$ $\dot{q}(t)$	$q(t)$ versus time, t; Packet loss rate and throughput for each flow; Varying N in a given run;	TCP (bulk data transfer)	ns-2
Predictive PID [38]	2004	PID	CARIMA Generalized Predictive Control (GPC)	Static values C=3750 pkts/s; N =30; R =0.14 s	Linearized TCP/AQM model by Hollot et al.	$p(t)$	$q(t)$	$q(t)$	$q(t)$and $p(t)$ versus time, t; Varying N in a given run;	TCP persistent flows	ns2
DC-AQM [42]	2004	RED, PI, REM	Robust Control Internal Mode	Static C=3750	Linearized TCP/AQM	$p(t)$	$q(t)$	$q(t)$	$q(t)$and $p(t)$ versus time, t;	---	ns-2

rate, y_l, link capacity, c_l, and the buffer limit, B, are then scaled by a factor K in this loss probability expression and the limit as $K \to \infty$ is taken, so that $p_l(c_l, y_l) = \frac{(y_l - c_l)^+}{y_l}$, where $[z]^+ = \max(0, z)$.

Therefore AVQ employs a virtual queue, the link capacity of which is less than that of the real queue[37]. And packets are marked when the arrival rate to the queue is greater than the virtual capacity[128]. In order to theoretically compute an appropriate virtual capacity so that the penalty function above is satisfied, the queue must know the number of flows passing through it as well as the utility function of each of the flows. The number of flows is time-varying, so that this computation must occur when network load changes. To make this computation practically independent of the number of users, the link capacity of the virtual queue is adjusted based on the differential equation:

$$\dot{\tilde{C}}(t) \quad = \quad \alpha(\gamma C - y(t)) \qquad (23)$$

where $\tilde{C}$ is the link capacity of the virtual queue, C is the link capacity of the real queue, $y(t)$ is the input rate to the system, and α and γ are AVQ control parameters. Namely, γ is the desired utilization and α is the damping factor or smoothing parameter[45] or step-size[128]. From Eq. (23), it can be seen that AVQ attempts to match the input rate to the virtual capacity so as to achieve the desired utilization[13]. It is said that a key AVQ design issue is the choice of the damping factor α since it determines the speed of adaptation of the virtual link capacity[127]. However, according to[45] both α and γ determine the stability of the system, and γ determines how robust the system will be in the presence of short flows.

The size of the virtual queue is the same as that of the real queue which is fixed. Because the service rate of the virtual queue is smaller than that of the real queue (i.e., by a factor of $\alpha\gamma$), and both queues experience the same arrival rate, then the virtual queue will build up faster than the real queue, and will tend to overflow more quickly. Anytime the virtual queue overflows, those packets that are dropped from the virtual queue are also marked/dropped from the real queue[11]. This is deterministic marking as opposed to probabilistic marking used in RED[15], i.e., there is no explicit calculation of a marking probability[45], however the effect is similar. It can be seen from Eq. (23) that the marking will be very aggressive when the link utilization is exceeded and it will become much less intense when the link utilization is below the target[45]. According to [13], the effective dropping probability of AVQ is

$$p(t) \quad = \quad \left[1 - \frac{\tilde{C}(t)}{y(t)}\right]^+ \qquad (24)$$

where $[z]^+ = \max(0, z)$.

TABLE VIII
SUMMARY OF CONTROL-THEORETIC AQM ALGORITHMS CONT'D

AQM	Year	Compared with	Control Type	Controller Parameters	Plant Model	Control signal	Controlled signal	Feedback signal	Performance tests	Traffic types	Simulator
			Control (IMC) for delay compensation	pkts/s; N =60; R =0.4s	model by Hollot et al.				Utilization versus q_{ref}; Utilization versus RTT		
AGPC [96]	2005	RED, REM, AVQ, PI	Adaptive GPC	Model parameters estimated using standard gradient algorithm	Controlled Autoregressive Moving Average (CARMA) model	$p(t)$	$q(t)$	$q(t)$	$q(t)$ versus time, t; Varying N in a given run;	TCP Reno with ECN marking	ns-2
Gain Adaptive Smith Predictor and PI Controller (GAS-PI) [91]	2006	RED, PI	Robust Control Gain Adaptive Smith Predictor	Static C=3750 pkts/s; N =60; R =246 ms	Linearized TCP/AQM model by Hollot et al.	$p(t)$	$q(t)$	$q(t)$	$q(t)$ versus time, t; Varying N in a given run;	FTP with web-like flows	ns-2
IMC-Smith [92]	2006	Smith Controller	Robust Control Internal Mode Control (IMC) with Smith Predictor for delay compensation	Static C=3750 pkts/s; N =60; R =0.4s	Linearized TCP/AQM model by Hollot et al.	$p(t)$	$q(t)$	$q(t)$	$q(t)$ versus time, t; Different delay times and flows for different runs	---	Matlab
μ- optimal Controller [95]	2007	RED, PI	Robust control H_∞ control	Static C=9708 pkts/s; N =100; R =0.25s	Linearized TCP/AQM model by Hollot et al.	$p(t)$	$q(t)$	$q(t)$	$q(t)$ versus time, t; RTT versus time, t;	FTP with HTTP	OPNET
Robust Discrete-Time Controller [122]	2009	RED, PI, PID	Robust Control	Static C=3750 pkts/s; N =60; R =240 ms	Linearized TCP/AQM model by Hollot et al.	$p(t)$	$q(t)$ $W(t)$- TCP window size	$q(t)$ $W(t)$	$q(t)$ and utilization versus time, t; Addition and removal of Constant Bit Rate (CBR) traffic disturbance	TCP; Constant Bit Rate (CBR) traffic disturbance	ns-2
RSP [93]	2010	REM, PI, DC-AQM	Robust Smith Predictor using	Static	Linearized TCP/AQM	$p(t)$	$q(t)$	$q(t)$	$q(t)$ versus time, t; Varying N in a given	TCP	ns-2

By setting γ to one (1) and solving Eq. (23) with initial conditions $\tilde{C}(0) = C$ and $q(0) = 0$ where $q(t)$ is the instantaneous queue length, it was shown by [13] that:

$$\tilde{C}(t) = C - \alpha q(t) \tag{25}$$

which indicates that AVQ adjusts the virtual capacity according to the queue-length. As queue length increases, the virtual capacity decreases.

The AVQ algorithm can also be considered as a token bucket[45], [14], [9], [11]. Tokens are generated at a rate of $\alpha\gamma C$. αs tokens are removed from the bucket upon each packet arrival where s is the packet size.

The designers of AVQ performed stability analysis of AVQ[45] in the presence of feedback delays using the TCP fluid model and linearizing the system dynamics. Based on this analysis they recommended design rules for the control parameters γ and α in terms of the number of flows, and maximum delay. They did assume that all the users had the same round-trip-time. Also, slowstart and retransmission timeouts were not included in the analysis. They did mention that in the absence of feedback delays, AVQ is fair in that it maximizes the sum of all the sources' utilities in the network. Even when the utility function was changed to the 'potential delay model' so as to reflect the different round-trip-times of the users, it was found that AVQ converged to the optimal point where the sum of these utilities was maximized.

AVQ is solely a rate-based AQM so that it does not explicitly control the queue length to a target value[14], [52], therefore queue-length effects (i.e., delay and jitter) may vary widely especially when there are high levels of congestion[14]. Through simulation experiments,it was found by [11] that AVQ experienced lower packet loss rates and higher link utilization that RED and PI controller in a traffic environment consisting mainly of long-lived flows. Disadvantages: AVQ was not designed to deal with unresponsive flows (e.g. UDP); AVQ, like RED, is computationally intensive as the virtual buffer update and control function are executed on every packet arrival[14], [11].

C. Stabilized Virtual Buffer (SVB)

Developed by [52], Stabilized Virtual Buffer (SVB) uses both packet arrival rate and queue-length as part of its congestion indicator and attempts to keep the the packet arrival rate and the queue-length around their individual target values.

SVB is similar to AVQ in that it uses a virtual queue in order to determine packet marking/dropping, but it differs in the following ways:

TABLE IX
SUMMARY OF CONTROL-THEORETIC AQM ALGORITHMS CONT'D

AQM	Year	Compared with	Control Type	Controller Parameters	Plant Model	Control signal	Controlled signal	Feedback signal	Performance tests	Traffic types	Simulator
			Internal Mode Control (IMC) for time-delay networks		model by Hollot et al.				run; Varying propagation delays across runs.		
H_∞ optimal AQM Controller [94]	2010	PI	Robust control H_∞ control	Static C=300 pkts/s; N =50; R =0.533s	Linearized TCP/AQM model by Hollot et al.	$p(t)$	$q(t)$	$q(t)$	Queue length error, $e(t)$ versus time, t; Addition of UDP flows	TCP	Matlab
SPIDNN [113]	2010	PID	Three-layer feed-forward neural network	Adjusted by neural network algorithm	Linearized TCP/AQM model by Hollot et al.	$p(t)$	$q(t)$	$q(t)$	$q(t)$ versus time, t; Varying N in a given run;	FTP	ns-2
Hybrid GA-BF Based Intelligent PID [123]	2011	RED, Droptail	Genetic Algorithm (GA) and Bacterial Foraging (BF) to derive controller gains of a PID controller for time-delayed system	Static But based on GA-BF algorithm	Linearized TCP/AQM model by Hollot et al.	$p(t)$	$q(t)$	$q(t)$	$q(t)$ versus time, t;	TCP	---

- The virtual capacity is kept fixed at the real capacity C (whereas in AVQ is dynamically adjusted with respect to the incoming packet rate), and the virtual buffer limit is adaptable to the incoming packet rate (whereas in AVQ the virtual buffer limit is fixed and equal to the real buffer limit B).
- The packets in the real queue are not deterministically marked/dropped when the virtual queue overflows as in AVQ. Instead, the packets are marked with a probability based on the current virtual buffer limit and the virtual capacity.

The virtual buffer limit is updated as

$$b_v(t) \quad = \quad \gamma(C - y(t)) \qquad (26)$$

where $b_v(t)$ is the virtual buffer limit, C is the virtual capacity (= real capacity), $y(t)$ is the packet arrival rate and γ is the SVB parameter.

It is claimed by [52] that SVB is robust to and responds quickly to changes in network load. The queue-length remains stable for different traffic conditions

D. Random Early Marking (REM)

The full algorithm was presented for REM by [44].

REM differs significantly from RED by its congestion measure and by its probability dropping/marking function. The aim of REM is to decouple the congestion measure from the performance measure[37], [34], [11], [17], [44], [14], so as to achieve its target queue length (for low delay) and its target rate (for high utilization) independent of congestion levels in the network (i.e., the changing number of flows)[52], [85] while at the same time reach a global optimal operating point[34].

The REM congestion measure, also known as a price, is the weighted sum of the rate mismatch (i.e., deviation of the actual aggregate input rate from the target) and the queue mismatch (deviation of the actual queue length from the target queue length). To be more precise:

$$\mu_l(t+1) \quad = \quad [\mu_l(t) + \gamma(a_l(q_l(t) - q_{ref}) + (x_l(t) - c_l))]^+ \qquad (27)$$

where $[z]^+ = \max(0, z)$. μ_l is the price of link l, q_l, the actual queue length, q_{ref}, the target queue length, c_l, the link speed and x_l, the actual input rate. Therefore, if the weighted sum of the mismatch is positive (when there is congestion), the price increases, and if negative (meaning that congestion is decreasing) the price decreases[17], [41]. At equilibrium (when the price stabilizes), the weighted sum should be zero (i.e.,$q_l(t) = q_{ref}$ and $x_l(t) = c_l$)[44]. It should be noted that with REM all the queues/links independently set their price[44] and there is no need for coordination among them to bring about this optimum[59].

Now, because it is easier to sample queue length than input rate, the rate mismatch can be determined by the rate of growth in queue length, i.e., $q_l(t+1) - q_l(t)$, so that the price is modified as in (28) and (29) on the following page [44], [15]. In terms of the probability dropping/marking function, REM employs an exponential form so that the source sees the sum of the prices of the individual links along the path its flow takes, since along the path can be a number of congested links. This will be the true measure of congestion along the path. In particular, the probability of marking a packet at link l is

$$p_l(t) \quad = \quad 1 - \phi^{-\mu_l(t)} \qquad (30)$$

where ϕ is a constant and $\phi > 1$.

TABLE X
SUMMARY OF FUZZY-BASED AQM ALGORITHMS

AQM	Year	Compared With	Adaptive Module	Associated AQM	Input variables	Output Variables	Membership Functions	Inference Engine	No. of Rules in Rule Base	Defuzzifier
AFRED [99]	2003	RED, PI	Yes; Periodically measure the real packet drop ratio to readjust fuzzy rules	nil	Instantaneous queue length, $q(t)$	Packet drop probability	Input: Trapezoidal, variable number; Output: Single-line, variable number	Product	Variable	Center-average
FPI [117]	2003	Droptail, PI, RED	Genetic Algorithm (GA)	nil	Errror $e(t) = q(t) - q_{ref}$ and $\Delta e(t)$	Increment in probability of dropping/marking , $\Delta p(t)$	Input: Triangular, 5 fuzzy sets; Output: Triangular, 7 fuzzy sets	Tabulated	15 rules	Not shown
FIPD [114]	2003	RED	nil	nil	Queue length; change in queue length	Congestion index	Input: triangular, 7 fuzzy sets; Output: Triangular; 9 fuzzy sets	Tabulated	49 rules	Center-average
Fuzzy Controller with RED [118]	2004	RED	nil	RED	$\bar{q}(t) - \frac{max_{th} + min_{th}}{2}$ Difference between arrival rate and target link capacity	The change in the maximum drop probability of RED, i.e., ΔP_{max}	Input: Trapezoidal and triangular, 9 fuzzy sets and 3 fuzzy sets; Output: Trapezoidal and triangular, 9 fuzzy sets	Not shown	Not shown	Not shown
FAFC [115]	2004	RED, PID	Classical PID with EWMA	nil	Error $e(t) = q(t) - q_{ref}$ and derivative of the error	Proportional Gain	Not shown	Control surface shown	3 rules	Not shown
ARAD [119]	2004	RED, Droptail	nil	nil	Radio Link Control (RLC) queue length; aggregate packet arrival rate; queue length change rate;	No. of retransmissions times; Packet accept/drop status	Input: Trapezoidal and triangular, 4, 3 and 2 fuzzy sets; Trapezoidal and triangular, 6 and 4 fuzzy sets	Max-min	24 rules	Center of Area

$$\mu_l(t+1) \quad = \quad [\mu_l(t) + \gamma(a_l(q_l(t) - q_{ref}) + (q_l(t+1) - q_l(t)))]^+ \tag{28}$$

$$\Rightarrow \quad \mu_l(t+1) = [\mu_l(t) + \gamma(q_l(t+1) - (1-a_l)q_l(t) - a_l q_{ref})]^+ \tag{29}$$

Therefore, the end-to-end marking probability will be, according to [44],

$$1 - \prod_{i=1}^{L}(1 - p_l(t)) \quad = \quad 1 - \phi^{-\sum_l \mu_l(t)} \tag{31}$$

so that as the individual prices increase so will the end-to-end marking probability. For low drop probabilities, the end-to-end marking probability can be approximated to $(\ln\phi)\sum_l p_l(t)$.

It was shown by [14], [13] that REM is essentially the same as a Proportional Integral (PI) Controller (designed from classical control theory). (Their equivalency was mentioned by [44] as well.) The congestion measure for REM can be rewritten as (32) on the following page, and since for low probabilities $p_l(t) \approx (\ln\phi)\mu(t)$ then

$$p_l(t) \quad = \quad K_P e(t) + K_I \int_0^t e(\tau)d\tau \tag{33}$$

where $K_P = (1-a_l)\gamma\ln\phi$ and $K_I = a_l\gamma\ln\phi$.

Now, although REM was designed to be optimal during steady-state, it is not so during the transient[44], [27]. Additionally, based on stability analysis and experiments, it was found that if the network parameters (e.g., number of flows and round-trip-times) were known before hand then stability could be guaranteed for satisfactory responsiveness. But in reality, these network parameters change so often that designs are more cautious with regard to stability at the expense of responsiveness[15].

VIII. AQM FOR DIFFSERV

For traditional best-effort networks, the AQM designers were attempting to integrate with congestion control (the original objective), the goals of fairness, and to a growing extent QoS differentiation into active queue management itself. However, within the DiffServ framework, one aspect of service differentiation, i.e., (aggregate) flow isolation, has, for the most part, been achieved by queue scheduling algorithms

TABLE XI
SUMMARY OF FUZZY-BASED AQM ALGORITHMS CONT'D

AQM	Year	Compared With	Adaptive Module	Associated AQM	Input variables	Output Variables	Membership Functions	Inference Engine	No. of Rules in Rule Base	Defuzzifier
Fuzzy GREEN [103]	2006	GREEN	nil	GREEN	Queue drops and current utilization	Adaption parameter, $\gamma(t)$, of GREEN	Input: Trapezoidal and triangular, 3 fuzzy sets; Output: Trapezoidal and triangular, 3 fuzzy sets	Product t-norm	9 rules	Center of Maximum
Fuzzy Logic Congestion Detection (FLCD) [108]	2006	REM	Adaptive Multi-Objective Particle Swarm Optimization (AMOPSO)	CHOKe	Backlog factor (ratio of queue length to buffer size) Weighted average packet arrival rate	Change in packet marking/dropping probability	Input: Trapezoidal, 3 fuzzy sets; Output: Trapezoidal and triangular, 5 fuzzy sets	Not shown	Not shown	Not shown
Fuzzy Control RED (FCRED) [101]	2007	ARED, RED, PI	nil	RED	Queue length error $e(t)$; change in $e(t)$	Maximum drop probability of RED, i.e., P_{max}	Input: Triangular and trapezoidal, 7 fuzzy sets and 5 fuzzy sets Output: Triangular, 9 fuzzy sets	Tabulated	35 rules	Weighted average
Fuzzy REM (FUZREM) [105]	2008	FEM, REM, PI, ARED	nil	REM	Current price Error of price	Probability of packet dropping	Input: Triangular and trapezoidal, 7 fuzzy sets; Output: 7 fuzzy sets	Tabulated	49 rules	Center of Area
FLC of RED [102]	2008	RED	nil	nil	Average queue length Packet loss rate	Probability of packet dropping	Input: Trapezoidal, 3 fuzzy sets; Output: Trapezoidal and triangular, 4 fuzzy sets	Rules provided	9 rules	Center of Gravity
Fuzzy AQM [110]	2009	REM, RED, LRED	nil	nil	Error between instant queue length and reference queue length; Error between link rate and link capacity	Dropping probability	Input: Triangular and trapezoidal, 9 fuzzy sets; Output: Tabulated, 9 fuzzy sets	Tabulated	54 rules	Look-up table

$$\mu_l(t) \quad = \quad \left[(1-a_l)\gamma(q(t)-q_{ref}) + a_l\gamma\int_0^t (q(\tau)-q_{ref})d\tau\right]^+ \qquad (32)$$

(e.g., Weighted Fair Queueing (WFQ)). Another aspect, i.e., traffic conformance, has been managed by traffic conditioners at the edge. This in turn has allowed the AQM to concentrate more closely on the mechanism of dropping/marking packets appropriately. Therefore less complex AQM algorithms can be employed within the DiffServ network core. In this section we first describe the DiffServ architecture, then we consider research work that examined the effect of AQM on traffic types such as voice and video. Then we look at the performance of AQM schemes (new and old) within the DiffServ context.

A. The DiffServ architecture

Figure 8 provides a general overview of the DiffServ architecture. All customer traffic (voice, video, data), via their respective service level agreements (SLAs), are mapped to either of six possible traffic classes (also known as per-hop behaviors (PHBs): Expedited Forwarding (EF) (given the highest priority and is typically for low-latency, low-loss traffic), Assured Forwarding (AF1, AF2, AF3, AF4), and Best-Effort(BE). To differentiate each aggregate flow, a differentiated services code point (DSCP) is assigned.

Associated with each class is a meter specification and a conformance specification. The meter specification outlines the committed information rate (and in some cases the peak information rate) as well as the maximum burst size associated for the traffic flow. The conformance specification outlines what must be done if a packet belonging to a flow conforms or violates the meter specification. In some cases, the packet is dropped at once, or the packet's drop precedence is increased. For the latter case, if the network has extra capacity the non-conformant packet may be allowed through. However, if there is congestion, the non-conformant packet with the higher drop

TABLE XII
SUMMARY OF FUZZY-BASED AQM ALGORITHMS CONT'D

AQM	Year	Compared With	Adaptive Module	Associated AQM	Input variables	Output Variables	Membership Functions	Inference Engine	No. of Rules in Rule Base	Defuzzifier
DEEP BLUE [107]	2009	BLUE	Q-learning	BLUE	Queue length, packet drop probability,now-last update	Optimal Q-value	Input: Triangular, 3 fuzzy sets	Not shwon	Not shown	Equation provided
Fuzzy AQM [104]	2010	ARED	nil	RED	$\bar{q}(t) - \frac{max_{th} + min_{th}}{2}$ Maximum drop probability of RED, i.e., ΔP_{max}	The change in the maximum drop probability of RED, i.e., ΔP_{max}	Not shown	Not shown	Not shown	Equation provided
Variable-length virtual output queue (VOQ) based fuzzy congestion control [98]	2011	Droptail, RED	nil	Droptail	Instantaneous queue length. For an NxN switch, each input port has N virtual output queues (VOQs)	Maximum queue length limit for each VOQ at port	Not shown	Tabulated	Variable	A normalized summation

precedence will be dropped or marked more readily. It is the traffic conditioner at the edge node which determines which packets are conformant and which are not.

Each class of traffic is granted a provisioned proportion of the network bandwidth, and this allocation is realised by queue scheduling. Therefore, there is a single queue associated with each class of traffic at the output of the router. These individual per-class queues may become congested. It is within these individual queues that AQM algorithms are executed so as to mark or drop the packet according to its drop precedence.

B. The effect of active queue management on different traffic types

Voice over IP (VoIP) is addressed in [129]. Here the authors investigate voice quality in networks that employ active queue management, and more specifically, the RED and ARED algorithms. Voice quality is affected to a great extent by delay, jitter, and, bursty loss. Depending on the degree of compression, the voice traffic can be more or less tolerant to the latter. VoIP traffic uses UDP as its transport. In their analysis, the authors looked at a mixture of short-lived traffic (i.e., Web) and long-lived traffic (i.e., FTP). They modeled the G.711 codec as a constant bit rate source consisting of UDP packets that are 92 bytes long, 12 bytes of which constituted the RTP header. There was no Voice Activity Detection (VAD), but there was some pseudo adaptive jitter buffer. They used the E-model with MOS to provide a measure of voice quality. The authors concluded that the use of AQM did indeed improve voice quality. Based on previous work performed by the authors of [130], they claimed in [130] that the RED algorithm can work efficaciously with VoIP flows.

Scalable H.264 Coding (SVC) consists of multiple layers (basic, enhancement) and frames (I, P and B), each of which has a different level of precedence depending on its importance to the overall quality of the video. For networks that carry this type of traffic, the authors of [131] proposed PID-PD, i.e., a PID controller with Priority Dropping based on an $m \times n$ priority matrix, where m is the number of SVC layers and n is the number of priority ranks within each layer. The dropping probability of a packet is calculated according to the PID control. If it is determined that the packet is to be dropped, the priority queues are searched out to determine if there are any lower priority packets. If there is, that one would be dropped instead.

For modeling efforts on scalable video streaming with active queue management, see [132]. Also in [132] there is a proposed video streaming framework called Partitioned Enhancement Layer Streaming (PELS). In this framework, a router requires two queues - the PELS queue for multimedia traffic and the Internet queue for non-multimedia traffic - with weighted round-robin (WRR) scheduling executed between them.

Video transmission in autonomic networks is discussed in [133]. Here the authors propose a new AQM scheme called Autonomic AQM (AAQM) howbeit in the context of Mobile Ad Hoc Networks (MANETs). Again, the concern is that different video frame types have different levels of impact on the video quality and should be prioritized appropriately. For AAQM, service context information (such as frame type, frame number and frame size and priority index,) is embedded in the IP header. This information is extracted by nodes in the network to determine what treatment should be meted out to the packets constituting the frame(s). RED is at the core of AAQM, but based on the service context information, the maximum probability parameter, $P_{\max}$ is adjusted.

In [134], there is a survey on active discarding packet mechanism for video transmission. Like AAQM, these go beyond the many conventional AQM schemes discussed, in that more intelligence is added into the packet dropping/marking mechanism to decipher not only frame types but relationships among the frames. This type of information is included in pseudoheaders. The premise is that if certain high priority frames (e.g., I frames) are dropped or become lost, then it makes no sense to transport the associated P and B frames. It

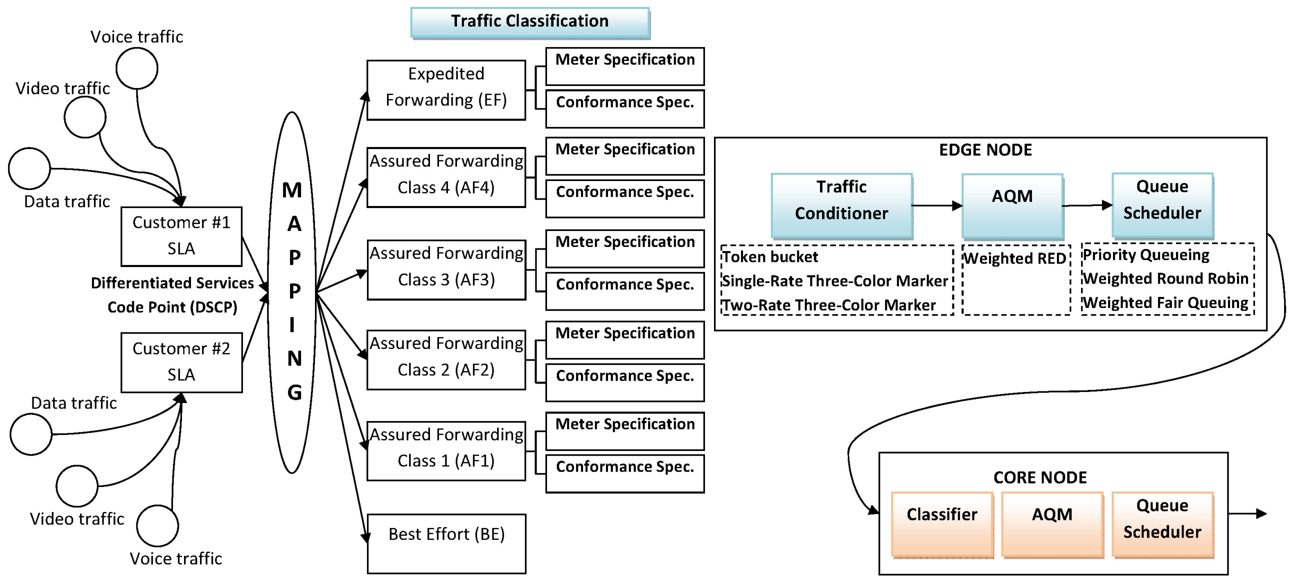


Fig. 8. The DiffServ architecture

was found that this approach alleviated congestion. It should be noted that the DiffServ framework was not explicitly mentioned in the discussion of these active discarding packet mechanisms.

AQM schemes were designed to address fairness issues between responsive TCP flows and unresponsive UDP flows in a common IP network. The traditional approach was to protect the TCP flows and punish (even severely) the UDP ones which would in turn incur large loss rates. In [135], a strategy is proposed to provide greater support to higher priority real-time multimedia flows that rely on UDP. The aim is not only to control congestion but, to a satisfactory extent, preserve the quality of the multimedia flows which are becoming a greater factor in contemporary networks.

C. The performance of AQM schemes implemented within the DiffServ framework

In Table XVII of the Appendix, a number of AQM crafted for differentiated services are outlined. We continue the discussion here by identifying a few more AQM proposals specifically for the DiffServ framework.

In [136], a Fuzzy Explicit Marking (FEM) algorithm is discussed. See Section VI for more on AQM schemes based on fuzzy logic and control. Written by the same authors of [136],[137] outlines FEM In/Out which consists of two identical FEM controllers; one for each class of service (i.e. assured and best-effort), and each having a different target queue length. In [138], Modified BLUE with IN and OUT is proposed. It seeks to address queue length stability. Its performance is compared to that of RIO. In [139], there is Fuzzy Active Queue Management (FuzAQM). It works in conjunction with RED and DiffServ. FuzAQM uses fuzzy logic to monitor network conditions and adapts the two thresholds of RED ($\min_{th}$ and $\max_{th}$) (which normally remains static) according to the traffic type of the arriving packet so as to more ably deal with network dynamics. This paper considers two traffic types distinguished by different DiffServ Codepoints (DSCPs). The FuzAQM approach is different from the RIO mechanism in that the latter has different sets of thresholds that are statically pre-configured for each traffic class.

Instead of having AQM in all routers throughout the network, the authors of [140] have suggested Edge-based AQM, i.e., AQM deployed only at the network edge and drop-tail in the core. This was to minimize congestion control complexity. It would be interesting to see how this coincides with the DiffServ framework in which traffic classification and conditioning also occur at the edge.

Hybrid RED (HRED) is modified in [141] for realization of the drop precedence levels in Assured Forwarding (AF) Per Hop Behaviours (PHB) in DiffServ. The authors of [141] outline three properties that an AQM scheme should have in the DiffServ context: sheltering (i.e., loss rates of lower drop precedence levels should not be impacted greatly by higher drop precedence levels), load tolerance (i.e., high drop precedence traffic should not be starved and the hierarchy among the levels should be preserved), and assured drop probability for each level. In [142], RIO and WFQ are used as the basic QoS control mechanism in the DiffServ network together with a mechanism proposed by the authors to control the threshold values of RIO based on flow count. The control mechanism considers both input and output queues. However, it also determines the contract speed of each flow so as to decide how to adjust the threshold values. This seems redundant given that traffic conditioning already penalizes traffic that is not conforming.

In [143], an empirical study was conducted as to the interplay between existing AQM schemes such as RIO, PI, AVQ, REM and LRED and DiffServ mechanisms such as traffic-conditioning. How the steady-state and transient behaviours of these AQM schemes affect the performance of AF services (in terms of average goodput of each aggregate, and average link throughput) was investigated.

In [144], Adaptive Rate Management (ARM) was used at the edge to adjust the token-bucket parameters of the traffic conditioning mechanism according to network conditions. For the core, a two-level PID controller designed using Linear Quadratic Regulators (LQR) was proposed. Here we see the use of control-theoretic approaches within the DiffServ framework. The ARM concept was introduced in [145] and expanded upon in [146] for the AF PHB in DiffServ. The authors suggested that throughput performance of flows is not solely affected by the traffic conditioning at the network's edge but also, in a more complex way, by TCP's congestion-control mechanism operating at the source nodes, hence the need for the ARM so as to provide the minimum throughputs to the traffic classes. They model not only the TCP flows entering the core, but also the color-marking process at the edge and then determine the equilibria for the system. Using the open-loop DiffServ network model thus derived, they designed an ARM controller to regulate token-bucket rate. This ARM was a Proportional-Integral (PI) controller with low-pass filtering. The AQM adopted for the core in [145] was the two-level PI controller, although the authors have suggested that other AQM algorithms can be used there as well, such as RED, and AVQ. It should be noted that their main focus was on TCP flows dominating the network's traffic profile. What if this were not the case? In fact, challenges in this regard were alluded to in [147], which investigated video multicast (particularly receiver-driven layered multicast (RLM)) in DiffServ networks that employed extended versions of RED/RIO AQM. This issue of non-TCP flow impact on network performance was addressed in [148], which derived an analytical model for a priority system with RED in the presence of self-similar traffic. Though not specifically for the DiffServ context, the work of [149] may be considered related since in it a rate-adaptive video streaming system with AQM is modeled as a feedback control system using control-theoretic approaches. In [150] a self-tuning adaptive controller (STAC) was proposed for the DiffServ architecture. The controller, based on a finite set of past values of queue length and packet marking probability that were plugged into a difference equation, continuously estimated control parameters so as to determine the packet marking probability for the TCP/AQM DiffServ plant.

Other work with regard to AQM in DiffServ networks include [151], [152], [153], [154], [155], [156], [157].

IX. AQM in Wireless Systems

Traditional AQM research has primarily targeted wired networks. As mentioned before, the main thrust initially was congestion control with system stability and robustness amidst changing network conditions. Issues of fairness were also tackled especially in the wake of increasing volumes of unresponsive or not-as responsive multimedia flows that use UDP. The research trajectory for AQM moved toward its role in more formal QoS architectures such as DiffServ. Of late, AQM research has turned more squarely to the realm of wireless networks. In addition to discussing factors specific to wireless that can affect or be affected by AQM performance, in this section we attempt to organise examples of AQM work by wireless technologies, i.e., Wireless LANs, Mobile Ad Hoc Networks (MANETs), WiMAX (IEEE 802.16), GPRS, UMTS and LTE mobile networks, Wireless Personal Area Networks (WPANs), Wireless Sensor Networks (WSNs).

A. AQM considerations specific to Wireless

When considering AQM-in-Wireless research, one may have to look at strategies for the downlink versus the uplink, and whether the wireless network is infrastructure-type or infrastructureless. For the infrastructure-type network there is a base-station that acts as the intermediary between the wired world and the wireless access network, and will become a potential bottleneck due to the speed mismatch between the two regimes. There can be a large drop in available bandwidth from the wired network to the wireless network since interference, collisions, multipath-fading, propagation distance and shadowing effects, reduce the available bandwidth in the wireless medium. For the latter case (i.e., infrastructureless) the mobile nodes themselves act as routers which can become congested.

To help TCP contend with "rate diversity" in the wireless channel, the authors of [158] propose Channel-Aware AQM (CA-AQM). Rate diversity comes about when different users, who compete for the same wireless channel, independently adjust their rates based on their own perception of channel conditions. Based on some experiments they ran, the authors found that one connection can negatively impact the performance of all other TCP transmission in such a multi-rate wireless environment. To develop CA-AQM they used a utility-based optimization framework that takes into account wireless networks' characteristics. When the queue length is below a pre-set threshold, the price is updated according to the cumulative ratio of queue length to bit rate of all the flows. However, when the queue length exceeds this threshold, the price adjustment also takes into account estimated channel capacity as seen by each flow. This estimate is based on measurements from other packet transmissions sensed over the wireless. Measurements include the length of the contention period and the size of the packets.

Rate-based exponential AQM (REAQM) was proposed in [159] as an AQM scheme to improve TCP performance over wireless links. It uses input rate mismatch as the primary congestion metric and queue length as the secondary. It also uses ECN marking as opposed to packet drops so as to minimize the ambiguity between packet losses due to congestion and packet losses due to channel errors. REAQM is similar to REM. However, the price update formula is more complex, so as to enable a more suitable tradeoff between systems stability and utilization.

In [160] was presented an algorithm called Adaptive QoS and Wireless Bandwidth (adaptive-QWB). To address the time-varying available wireless link bandwidth, the algorithm attempts to find the best target queue length which minimizes the variation in end-to-end delay and ensures that the end-to-end delay remains below a maximum. The target queue length calculated is then used to determine an appropriate packet drop probability using a PID algorithm.

In [161] another short-coming of RED with regards to mobile networks was highlighted. Because of the exponential

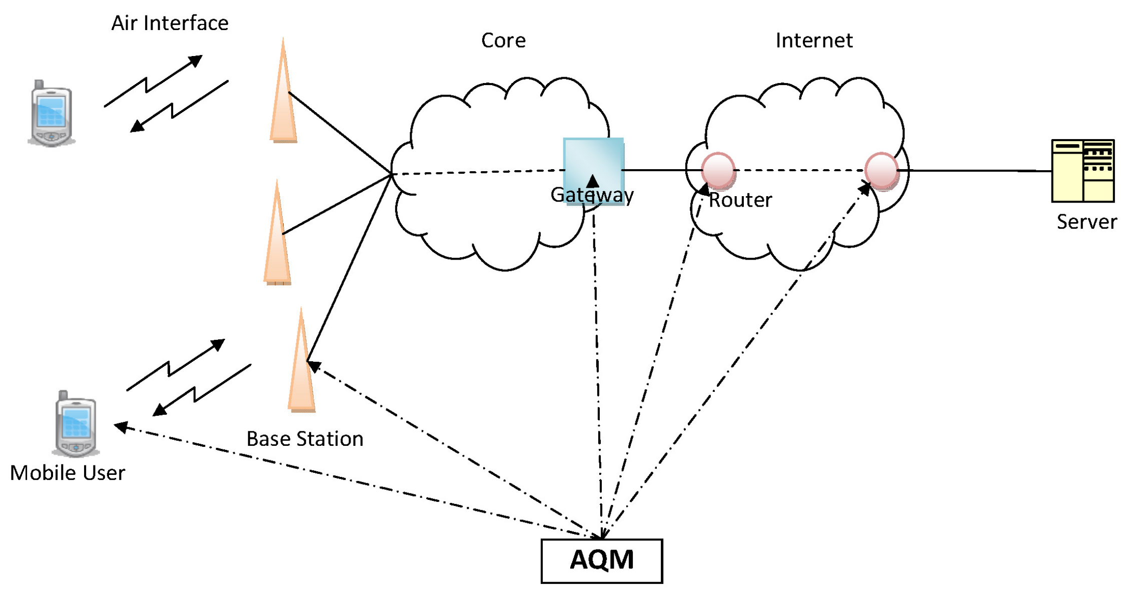


Fig. 9. Location of AQM in infrastructure-type wireless access networks

weighted averaging of the instantaneous queue length, RED cannot respond quickly to sudden changes in network load that result from handovers. To address this, Dual RED is proposed. Dual RED consists of the original RED algorithm and an "add-on" RED (aRED) which comes into play only during a handover so as to differentiate forwarded traffic from ordinary traffic. The structure of Dual RED is similar to RIO.

In terms of analytical work, the stability of TCP/AQM wireless networks with feedback delays was investigated by [162] with the aid of Delay Markov Jump Linear System (DMJLS). To deal with capacity variations in wireless links due to fading, an H_∞ robust controller design for bottleneck node (i.e., the interface between wired and wireless network) in the downlink direction was performed by [163].

B. WLAN

In [164], [165], the CLAMP algorithm for AQM is presented. It adjusts TCP receiver's advertised window limit for flow control. The topology under consideration consists of an access point (AP) which maintains per-user queues. However, any given user may have multiple TCP connections competing for its assigned queue at the AP. For it to work, CLAMP must be implemented at both the AP and the mobile client. At the AP, an average queue length that is based on the length of each user's queue at the AP, is determined and incorporated into a measure of congestion. This information is sent to the mobile clients via the IP header in packets. The client, upon receiving the measure of congestion adjusts the advertised window according to an equation derived by the authors that is based on TCP flow dynamics.

The fusion point between an IEEE 802.11 wireless LAN network and a wired network is examined in [166]. It is at this point, where there is a great likelihood of congestion, that AQM is applied, particularly RED and one devised by the authors called EF-AQM. They highlight two challenges for TCP with IEEE 802.11 wireless LANs: the single collision domain in which a high number of nodes contending for channel resources can exist at any given point in time, and the fact that nodes can dynamically change their MAC-layer data rates to 1, 2, 5.5 or 11 Mbps. The EF-AQM is based on feedback control theory, but details as to the tuning of its parameters are not provided.

The authors of [167] examined the performance of RED at the wired-wireless (IEEE 802.11) interface, i.e., the access point. They argued that RED negatively impacts WLAN performance in terms of packet-loss rate and goodput, although it could be improved with ECN. They then proposed Proxy-RED, but in a network infrastructure which consists of light-weight access points and a gateway which performs most of the networking functionality such as QoS, mobility, security, etc. For Proxy-RED, the RED algorithm is implemented at the gateway, but the gateway must monitor the queues of the APs it manages by periodically sampling their instantaneous queue length to estimate the average queue length value to use at the gateway. Proxy-RED therefore reduces the AQM overhead at the APs.

The authors of [168] sought to improve fairness among uplink and downlink flows in an infrastructure WLAN using the Distributed Coordination Function (DCF) mode. To do this, they used a PI controller to dynamically change the minimum congestion window in conjunction with A^{2}RED developed in another publication.

C. 2G/3G mobile networks

The authors of [169] investigated the use of AQM in the downlink Radio Link Control (RLC) buffers of the Radio Network Controller (RNC)in 3G radio access networks. They compared the performance of RED and a new AQM scheme they proposed called Slope Based Discard (SBD) with the case of simply having Droptail. This work only considered TCP flows and not multimedia-over-UDP flows. AQM is recommended for the RLC since it stores data from the upper layers until fully acknowledged across the wireless link which in turn can incur large frame losses. As a result the RLC buffers can become highly congested impacting on TCP flows. The SBD algorithm computes a critical rate based on buffer occupancy and the reaction time which is the sum of the time the congestion signal gets to the TCP source and the time rate reduction is actually detected at the buffer employing the AQM. The actual incoming rate is compared to the critical rate and consequent drop decisions are made.

In [170], AQM is applied to the Radio Network Controller (RNC) in WCDMA networks that employ High Speed Downlink Packet Access (HSDPA), since it acts as the interface between the core network (CN) and the Radio Access Network (RAN), and so can be a bottleneck due to the rate mismatch between the two systems. It is the RNC which performs admission control and resource management. The authors' focus here was again on TCP flows. AQM schemes considered in their analysis included RED, ARED, Packet Discard Prevention Counter (PDPC) and Time-to-Live-based RED (TTLRED) - an AQM scheme they proposed in that paper. In TTLRED, time-stamps are applied to IP packets entering the PDCP buffer. These packet life times are used as the congestion indicator instead of queue length as in standard RED. Two versions of TTLRED are presented, one for incoming packets, the other for buffered packets. For incoming packets, the shape of the packet drop probability characteristic follows that of Gentle RED. For the latter, packets are checked every T seconds and if the current time is greater than the timestamp value, the packet is dropped/marked.

The same authors of [170] investigated the use of AQM in 2G-EGPRS (Enhanced GPRS) in [171]. For this study, they consider as a greater potential for bottleneck the 2G Serving GPRS Support Node (SGSN) rather than Base Station Subsystem (BSS), and so apply AQM on the downlink there. The AQM schemes under test were RED and TTLRED mentioned before.

GPRS links are also addressed in [172]. However, the AQM analysis performed was in the context of vertical handoffs between the GSM/GPRS system and the IEEE 802.11 WLAN, i.e., within a heterogeneous environment. The authors of [172] provide guidelines for parameter values for RED in the GPRS link queues. For example, the upper bound of the queue weight, w_q, they believe depends on the number of packets in a burst (L) when the RED queue is in its initial state (i.e., average and instantaneous queue lengths are zero). They also propose a new AQM called Burst-Sensitive RED (BSRED) to counteract the effect of packet bursts in these links. BSRED adds a new parameter to the standard RED called the Burst Threshold (B_t). If the number of consecutive packets enqueued without a packet being dequeued exceeds this threshold, a packet dropping event must occur regardless of the queue length.

According to [173] channel impairments that distinguish wireless links from wired links include lower capacity, higher bit-error rates, longer delays and channel variability due to multipath propagation, shadowing, power limitations, and handoffs. Therefore techniques devised for wired systems cannot be immediately or directly applied to wireless sytems. There should be careful investigation as to the performance of such methods in the wireless domain. This is true for AQM. In [173] a CDMA2000 1xRTT network with commercial-grade equipment such as the Radio Base Station (RBS), Base Station Controller (BSC) and Packet Data Serving Node (PDSN) was the testbed for their analysis. A wireless channel emulator and noise generater were used to emulate the wireless link. The authors proposed and implemented a new AQM scheme called Remote-AQM (R-AQM) for such a network. The algorithm is located in the client node (i.e., the mobile handset (MH)) and regulates the queue length at the BS and hence the TCP traffic by using the advertised window. To perform its AQM function, the MH needs an estimate of the round-trip delay. This client-side approach they claim is feasible since each MH has a dedicated buffer in the BS.

Fuzzy logic is used in [119] to design a queue-based congestion mechanism called Adaptive Retransmission and Active Drop (ARAD) for the 3G downlink. Inputs to the fuzzy system include RLC queue length, packet arrival rate and the queue length change rate. The outputs relate to the number of retransmissions and the decision as to whether to drop or accept a packet.

D. WiMAX

In the study of [174], the backhaul link in the WIMAX network was treated as the main potential bottleneck instead of the air-interface as is usually the case. The authors examined the interaction between DiffServ mechanisms (which included the RED AQM), connection admission control and WiMAX. They mapped the WiMAX scheduling and data delivery services to three DiffServ classes: Expedited Forwarding (EF), a single Assured Forwarding (AF) class, and Best Effort (BE). Actual traffic types used in the simulation included VoIP, web browsing, video streaming and file downloads. Beside the role that AQM plays in the context of DiffServ itself, AQM can act a part in congestion control at the WiMAX Base Station (BS) downlink queues per connection. So, in this regard, the authors looked at the combined effect of having Time-to-Live(TTL)-based RED (TTLRED) operating at the BS and RED at the last router, and compared that to having no AQM at all; only having TTLRED at the BS, and only having RED in the core.

A new AQM algorithm named Wireless Delay-Based Queue (WDBQ) was proposed for IEEE 802.16 (WiMAX) networks in [175]. Because the cause of packet loss could be attributed to not only congestion but to channel impairments in wireless systems, congestion detection based on packet loss could be misleading. The authors proposed queueing delay instead. With the WDBQ algorithm, if the packet queueing delay exceeded a threshold (which was based on the packet round-trip times and the number of flows) the packet would be marked so that the TCP sending station would adjust its sending rate accordingly. The threshold was updated every T seconds to reflect changes in the network.

AQM was applied in the WiMAX Base Station downlink queues by [176] and the end-to-end performance of VoIP connections, video streaming connections, web-browsing connections and file download connections between mobile subscriber stations and an Internet server was examined. The AQM schemes under examination were RED, Packet Discard

Prevention Counter (PDPC) and TTLRED. Their conclusion was that AQM did improve performance in terms of queueing delays, which led to better user experience.

In [177], the performance of RED, REM, BLUE and drop-tail as possible queue management schemes for congestion control in WiMAX for the transport of a single video stream over multiple connections using MULTFRC over UDP, was compared. The variable-bit rate (VBR) video was mapped to the Real-time Polling Service (rtPS) service class buffer in the WiMAX bases station (BS). The video quality was judged in terms of Peak Signal-to-Noise Ratio (PSNR) and it was discovered that, for the most part, up to three connections could maintain acceptable PSNR and, that BLUE provided the best results of the three given this context.

E. MANETs

MANETS are infrastructureless wireless networks, with mobile nodes communicating peer-to-peer. The topology changes are quite frequent and much resources must be devoted to route change and its consequent issues. So besides the typical channel impairments of wireless networks, MANETs have to contend with more stringent constraints in memory, processing power and battery-life. Therefore, an appropriate AQM scheme in such a case would be one that is lightweight, and efficient. PAQMAN was proposed by [178], [179] to be such a scheme. Its congestion indicator is average queue length and at its core it uses the Recursive Least Squares (RLS) algorithm to predict average queue length in the next prediction interval. The difference between the predicted queue length and the target queue length is what determines the packet drop probability which remains constant over the prediction interval, at the end of which the prediction algorithm is re-executed. The performance of PAQMAN has been compared to that of Droptail. Though it might have been also useful to also compare its performance with that of RED or other well-known AQM algorithms.

In a MANET a mobile node may simultaneously connect with other mobile nodes and other networks (i.e., acting as a gateway). The authors of [180] model such a node as a finite-capacity, multi-server queuing system hosting multiple classes of Poisson traffic, eaching having its own priority.

Ad hoc Hazard RED (AHRED) has been proposed by[181] as another AQM algorithm suited to MANET. It has been compared to RED, REM and SRED. The packet dropping probability characteristic follows a Weibull model of hazard rate function, with queue length as the congestion indicator, i.e., $p = \beta q^{\beta - 1}$. But β itself changes with queue length so as to accelerate the AQM's response to increasing congestion.

F. WPAN

In [182], the performance of MPEG-4 transmission over an IEEE 802.15.3 WPAN that utilize AQM schemes such as RED and WRED is analyzed and compared with Droptail. Two of the performance metrics were: mean propagation time (MPT) and job failure rate (JFR). In terms of MPT, WRED was found to be the winner for larger number of devices in the piconet. In terms of JFR, the performance of RED and WRED was found to be the same.

G. 4G mobile networks

In [183], AQM is one of three mechanisms used to improve TCP performance during an in intra-LTE handover, i.e., when a mobile unit moves from one eNodeB to another eNodeB. During a handover the target eNodeB can suffer increased congestion. There are packets destined for this additional mobile unit that can now be found in the territory of the target eNodeB . These packets come directly from the serving gateway. Also, there are packets for this same mobile unit that are sent by the previous eNodeB. These packets were held there until the handover process was completed. In [183] three mechanisms are presented to address this challenge. Two of the mechanisms addressed the increase in roundtrip time that would result and which could negatively impact TCP's performance. The third mechanism, AQM, addressed the problem tunneling presents for ECN marking. The tunneling between the eNodeB and the serving gateway made the network ECN-unaware. Therefore some ECN translation between the tunnel and actual user IP headers had to be done, if any AQM algorithm were to be implemented in the core.

H. WSN

To alleviate congestion in resource-strapped WSN, the authors of [184] explored the possibility of adopting AQM into this regime. Simulations were run for three algorithms: RED, REM and PI. The issue they raised was that it was at the sink node that congestion occured in that many-to-one schema. They used AQM on the neighbour nodes of the sink, so as to prevent congestion at the sink node. They found that with RED the queues saturated frequently, causing greater variation in throughput. REM provided better queue-length stability around a smaller queue length than RED. They also found that PI was superior to both RED and REM in that it not only maintained shorter queue lengths but provided higher throughput.

Table XIII summarizes the wireless issues addressed by the various AQM schemes devised for that context.

X. Discussion

Upon examination of the AQM research literature, we discuss here in this section a number of challenges and research gaps in AQM research.

A. Research gaps

Very much has been accomplished in terms of control-theoretic approaches to AQM design and analysis for the express purpose of congestion control in wired networks. We have seen the use of more and more sophisticated robust control techniques. The main thrust of control theory is to ensure good stability and responsiveness of the system even in the presence of disturbances and large delays. However, a number of issues come to light. These are now discussed but not in any order of priority.

TABLE XIII
WIRELESS ISSUES ADDRESSED BY AQM APPROACHES

Issues Addressed	AQM scheme	Context	Traffic Focus
Rate diversity in wireless channel	Channel-Aware AQM (CA-AQM)	---	TCP
Rate mismatch	Rate-based Exponential AQM (REAQM)	---	TCP
Time-varying available wireless link bandwidth	Adaptive QoS and Wireless Bandwidth (AQWB)	---	---
Sudden changes in network load due to handovers	Dual RED	---	---
Competing TCP flows per end user (mobile client)	CLAMP	WLAN	TCP
Single collision domain and dynamic MAC-layer rate changes among nodes	EF-AQM	WLAN	TCP
Improve fairness among uplink and downlink flows	PI with A^2RED	WLAN (DCF mode)	---
RLC stores data from upper layers until full acknowledged across wireless link	Slope-Based Discard (SBD)	3G Cellular	TCP
2G SGSN node acts as the bottleneck rather than the BSS	Time-to Live Based RED (TTLRED)	WCDMA with HSDPA 2G-EGPRS	TCP
Vertical handoffs between GSM/GPRS and IEEE 802.11 WLAN	Burst-Sensitive RED (BSRED)	GSM/GPRS/IEEE 802.11	---
To distinguish between packet loss due to congestion and channel impairments in wireless systems	Wireless Delay-Based Queue (WDBQ)	WiMAX	TCP

1) The need for revised "plant" models: The TCP-centric model around which many control-theoretic AQM schemes have been built may not be sufficient due to the growing volume of non-responsive and not-as-responsive UDP-based traffic. A revised "plant" model that successfully incorporates not only TCP-based traffic (both long-lived and short-lived flows) but also UDP-based traffic with application-layer congestion control, TCP-friendly traffic (e.g., TFRC and GAIMD), as well as UDP-based non-responsive traffic.

The main emphasis of this survey has been router-based AQM schemes, much of which have been devised after and around the TCP endpoint congestion control. However, to overcome the limitations of TCP, some researchers have revisited the entire congestion control problem by performing a joint-design that comprise both the end system and routers as one. To this end there is eXplicit Control Protocol (XCP)[185] - a window-based congestion control protocol for best effort traffic. XCP was extended in [186]to work in the wireless domain wherein is variable capacity. This modified XCP was called XCP-b. Another protocol that follows this new paradigm is the Rate Control Protocl (RCP) and its extended version RCP with Acceleration Control (RCP-AC)[187].

2) Integration of control theory into AQM of the DiffServ and the wireless domain: Nevertheless, it would be good to explore how the control-theoretic and optimization-based AQM schemes can be systematically integrated within the DiffServ domain so that the latter can accrue benefits of assured stability even in a multiservice scenario. At this point, the research is predominantly separate with only a few examples of control-theoretic schemes being used in DiffServ with limited stability analysis to support their usage. This can also be said for the role of control-theoretic schemes in the wireless domain.

3) Stochastic optimization: Stochastic optimization is yet to make significant inroads into AQM. One stochastic optimization technique that may be suitable for AQM realization could be Infinitesimal Perturbation Analysis (IPA). With IPA, derivative estimators of a suitable cost function can be gleaned directly from the live traffic (hence they are called "online" estimators) and simultaneously fed into a gradient-descent algorithm to perform the optimization. Typically the queue dynamics are modelled as stochastic fluid queues for high volume traffic. One major benefit of this approach is that there is no need to know beforehand the underlying traffic distribution.

4) Complexity: Typically, the more sophisticated an algorithm, the greater the complexity in terms of processing power required and memory usage. Additionally, adaptive forms of AQM would have to measure variable network parameters such round-trip times, flow counts. For these techniques to be feasible (especially in terms of actual deployment), effective methods with much lower complexity would have to be sought.

5) AQM performance in the amalgamation of DiffServ and heterogeneous wireless networks: Though DiffServ and wireless issues were treated separately in this survey, there is the need to investigate the role of AQM when these two domains converge, i.e., DiffServ and wireless, so as to efficiently support mobile multimedia. One may want to determine whether or not it would be cost-effective for AQM to reside either at the air-interface or within the core or at both locations. Empirical studies may have to be conducted so as to determine what AQM characteristics would be crucial to such a scenario and which would be secondary.

B. Performance evaluation framework

One of the challenges with regard to AQM research is the lack of a common yet flexible performance evaluation framework by which new AQM schemes can be fairly compared with past ones. As an example, consider the control theoretic schemes in Tables VI to IX. One can see that from one AQM proposal to another, the network parameters, test procedures and performance criteria can differ, depending on what the author wishes to showcase. Additionally, the choice of simulator and the AQM schemes used for comparison may vary, together with the choice of topology (e.g., single or multiple bottleneck) and the traffic types used in the simulations. To illustrate: if one author proposed AQM X and compares its transient behaviour to RED, while another author proposed AQM Y and compared its steady-state behaviour to REM, it will be difficult to make a valid comparison between AQM X and AQM Y in terms of, say fairness, unless one implemented the two schemes oneself. However, the feasibility of this latter option depends heavily on how much of and how clearly the authors of the AQM schemes presented their algorithms in the research literature. What is proposed later in this section is a performance evaluation framework in terms of its components and their associated considerations. This performance evaluation framework is shown in Figure 10.

1) Previous work: The authors of [36] also stated the need for standardised performance evaluation of AQM schemes. They even suggested that AQM deployment has not progressed mainly due to a lack of detailed, objective and consistent evaluation.

They had proposed an integrated framework for benchmarking AQM, and in such they chose five metrics - utilisation, delay, jitter, drop rate and fairness - to capture and compare end-user experience. They also identified the choice of topology and traffic mix as critical to the framework. Additionally, they emphasized statistical techniques to calculate the length of the initial transient in simulations, the length of the entire simulation run and the number of independent replications according to pre-specified confidence intervals so as to overcome issues of biasedness of results.

We reiterate some of the main aspects suggested by the authors in [36], and we also recommend a number of additional features for consideration for such a framework. For example, we recommend the use of additional standardised criteria for stability, responsiveness and robustness - performance criteria not mentioned by [36]. Also, unlike [36], the framework we suggest does not revolve around a single simulator (in their case, ns-2). In fact, the authors of [36] implemented a high-level interface around ns-2 for the specification of experiments and for the production of reports.

2) Performance criteria: As mentioned earlier in this survey (see Figure 3), different authors use different performance metrics to demonstrate the effectiveness of their proposed AQM scheme. These include fairness, delay jitter, packet loss, application throughput and goodput, link utilization, queue stability, responsiveness, robustness and complexity. It may be useful for AQM research to develop standardized definitions for all these metrics together with standardized application dependent thresholds where appropriate. A common measurement methodology and set of statistics for reporting may also be necessary.

The IETF has already specified IP Performance Metrics (IPPM) Framework by which performance measures such as delay, end-to-end delay variation, round-trip-time, packet loss, and throughput are well defined. Although there are defacto measures for fairness, there is still room for variation. In classical control there are performance measures that attempt to capture responsiveness, stability and robustness such as maximum overshoot and rise time. These have not been consistently used in AQM research, particularly when illustrating how well the proposed AQM scheme responds to changing network conditions. A more thorough formulation of these measures for the purpose of AQM research may be required. And again, this formulation would require standardized definitions, measurement methodology, statistics for reporting and application dependent thresholds.

3) Performance scenarios: Figure 10 also outlines some of those network parameters that impact AQM performance. Hence, the choice of scenarios for performance evaluation should seek to address these issues comprehensively. To therefore make a comparison across AQM schemes concerning their response to any (changes) of these network parameters, the scenario should be the same across AQM schemes. As a result, an agreed upon format for such scenarios should be articulated.

As part of the scenarios, there is the choice of traffic types (e.g. FTP, HTTP, voice (UDP), video (UDP)). The models used for such should be outlined or explicitly declared (e.g. constant bit-rate at 64 kbps) for reproducibility. Additionally, the composition of the traffic mixes should also be clearly stated.

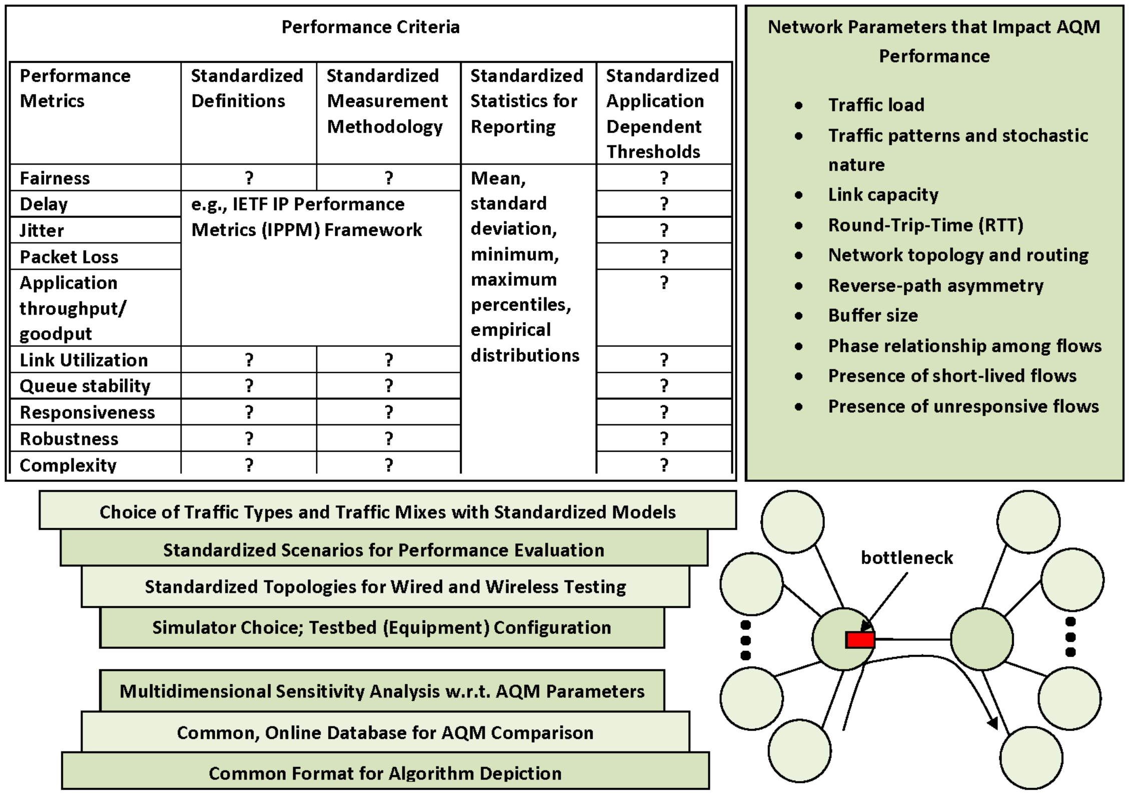
An additional component of the performance scenario is the topology choice (e.g. single bottleneck, or multiple bottleneck). For the most part, the topology is completely specified. What may differ from one AQM study to another may be the number of flows, packet sizes, the link speeds, link delays and the buffer sizes.

Though the dumbell topology is the most common choice when reporting AQM performance in wired networks, the topology choice tends to be more variable in the wireless context. This is an area worthy of further exploration.

4) Common format for algorithm depiction: It has been found that not many authors completely specify the AQM algorithm that they propose. So that, outside of a common, agreed upon framework of testing and measurement, it may be a challenge for another author to faithfully and fairly reproduce their results. It may be necessary to encourage AQM researchers to follow an agreed upon format for outlining their algorithms.

5) Additional considerations: It may be useful to have a common, online and open repositiary of AQM performance results to which authors can upload such for their algorithm. This enables an ongoing comparison. This may also provide a more definite means to judge the progression (in terms of performance) of the entire AQM research arena. By this equipment manufacturers may even be more encouraged to deploy AQM schemes they deem suitable.

It is inevitable that there may be some AQM parameters that would have to be tuned manually. It may be valuable for authors of AQM schemes to conduct and report multidimensional sensitivity analyses with respect to such parameters.



Performance Criteria

Performance Metrics	Standardized Definitions	Standardized Measurement Methodology	Standardized Statistics for Reporting	Standardized Application Dependent Thresholds
Fairness	?	?	Mean, standard deviation, minimum, maximum percentiles, empirical distributions	?
Delay	e.g., IETF IP Performance Metrics (IPPM) Framework			?
Jitter				?
Packet Loss				?
Application throughput/ goodput				?
Link Utilization	?	?		?
Queue stability	?	?		?
Responsiveness	?	?		?
Robustness	?	?		?
Complexity	?	?		?

Network Parameters that Impact AQM Performance

- Traffic load
- Traffic patterns and stochastic nature
- Link capacity
- Round-Trip-Time (RTT)
- Network topology and routing
- Reverse-path asymmetry
- Buffer size
- Phase relationship among flows
- Presence of short-lived flows
- Presence of unresponsive flows

Choice of Traffic Types and Traffic Mixes with Standardized Models
Standardized Scenarios for Performance Evaluation
Standardized Topologies for Wired and Wireless Testing
Simulator Choice; Testbed (Equipment) Configuration
Multidimensional Sensitivity Analysis w.r.t. AQM Parameters
Common, Online Database for AQM Comparison
Common Format for Algorithm Depiction
bottleneck

Fig. 10. AQM performance evaluation framework

XI. Conclusion

In this survey we journeyed through AQM research from its formal beginnings in 1993 to 2011. We looked at the RED algorithm and the analyses that ensued. We saw the transition to control theoretic approaches starting with the Proportional Integral controller. This particular branch of AQM research has mushroomed as the various techniques developed in the realm of control theory and practice are also finding application in the context of network congestion control. Optimization approaches to AQM were also discussed. With the growing importance of Quality-of-Service (QoS) provisioning in multimedia networks, and the widening popularity of the DiffServ framework for carrying out such, AQM has found its niche as a mechanism for service differentiation. To this end, we have looked at some AQM approaches proposed in the research literature. We have also looked at the benefits wireless networks can accrue with the assistance of AQM for congestion control, and saw examples of AQM techniques to realize these benefits.

Sure enough, many more AQM schemes would be proposed in the future. However, what might be required would be a comprehensive framework within which AQM schemes could be rigorously and fairly compared. We discussed such a framework and suggested that it should include a system of performance criteria, performance measurements, and test procedures. This framework development could be followed by a full analysis campaign which would include many of the existing AQM schemes.

Appendix A
RED variants and heuristic AQM schemes

Tables XIV to XXVII summarize and categorize a number of AQM schemes according to improvement in stability or responsiveness, auto-tuning, improved fairness, and differentiated services.

References

[1] S. Floyd and V. Jacobson, "Random early detection gateways for congestion avoidance," *IEEE/ACM Transactions on Networking*, vol. 1, no. 4, pp. 397–413, 1993.
[2] C. Yang and A. Reddy, "A taxonomy for congestion control algorithms in packet switching networks," *Network, IEEE*, vol. 9, no. 4, pp. 34–45, 1995.
[3] M. Labrador and S. Banerjee, "Packet dropping policies for ATM and IP networks," *Communications Surveys & Tutorials, IEEE*, vol. 2, no. 3, pp. 2–14, 1999.
[4] G. Chatranon, M. A. Labrador, and S. Banerjee, "A survey of TCP-friendly router-based AQM schemes," *Computer Communications*, vol. 27, no. 15, pp. 1424–1440, 2004. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0140366404001689
[5] S. Ryu, C. Rump, and C. Qiao, "Advances in Internet congestion control," *Communications Surveys & Tutorials, IEEE*, vol. 5, no. 1, pp. 28–39, 2003.
[6] R. Jain, "Congestion control in computer networks: issues and trends," *Network, IEEE*, vol. 4, no. 3, pp. 24–30, 1990.
[7] L. Hu and A. Kshemkalyani, "HRED: a simple and efficient active queue management algorithm," in *Proceedings of the 13th International Conference on Computer Communications and Networks*, Chicago, IL, USA, 2004, pp. 387–93.
[8] M. Arpaci and J. Copeland, "An adaptive queue management method for congestion avoidance in TCP/IP networks," in *Proceedings of the IEEE Global Telecommunications Conference (Globecom '00)*, vol. 1, San Francisco, CA, USA, 2000, pp. 309–15.
[9] S. Floyd, R. Gummadi, and S. Shenker, "Adaptive RED: An algorithm for increasing the robustness of RED," 2001. [Online]. Available: citeseer.ist.psu.edu/floyd01adaptive.html

TABLE XIV
SUMMARY OF HEURISTIC AQM SCHEMES THAT IMPROVE STABILITY OR RESPONSIVENESS

AQM	Congestion Indicator	Control Function	Special Characteristics
Gentle RED (GRED) [25], [78], [24]	Average (EWMA) queue length	$p(\bar{q}) = \begin{cases} 0 & 0 \le \bar{q} \le \min_{th} \\ \frac{\bar{q}-\min_{th}}{\max_{th}-\min_{th}} P_{\max} & \min_{th} \le \bar{q} \le \max_{th} \\ \frac{1-P_{\max}}{\max_{th}} \bar{q} + 2\max_{th} - 1 & \max_{th} \le \bar{q} \le 2\max_{th} \\ 1 & \bar{q} \ge 2\max_{th} \end{cases}$	Has the same congestion indicator and marking algorithm as RED. Discontinuity removed by a finite linear slope
Parabola RED [7] and Hyperbola RED	Average (EWMA) queue length	$p(\bar{q}) = \begin{cases} 0 & 0 \le \bar{q} \le \min_{th} \\ \left(\frac{\bar{q}-\min_{th}}{\max_{th}-\min_{th}}\right)^2 & \min_{th} \le \bar{q} \le \max_{th} \\ 1 & \bar{q} \ge \max_{th} \end{cases}$	
Stabilized RED (SRED) [188]	Instantaneous queue length	$pzap = p_{sred}(q) \min\left(1, \frac{1}{(256P(n))^2}\right)\left(1 + \frac{H(n)}{P(n)}\right)$ where $p_{sred}(q) = \begin{cases} p_{\max} & \frac{1}{3}B \le q < B \\ \frac{1}{4}p_{\max} & \frac{1}{6}B \le q < \frac{1}{3}B \\ 0 & 0 \le q < \frac{1}{6}B \end{cases}$ where B is the buffer size, q is the current queue length and $p_{\max}$ is an SRED parameter to limit the maximum drop probability, and $P(n) = (1-\alpha)P(n-1) + \alpha H(n)$ where α is another SRED control parameter and $P(n)$ is the estimate of the probability that the zombie list has a "hit". $H(n)$ is a binary variable where $H(n) = 1$ if there is a "hit" or $H(n) = 0$ if there is a "no hit". Now, $0 < \alpha < 1$ and $\alpha \approx \frac{p}{M}$	Maintains a "zombie" list of flow identifiers from which it randomly chooses to compare with an incoming packet . If there is a match (or "hit") on which the dropping probability is calculated
Dynamic RED (DRED) [13]	Average (EWMA) queue length	At $t = n\Delta T$, the error signal is calculated as $e(n) = q(n) - q_{ref}$ where $q(n)$ is the instantaneous queue length and q_{ref} is the target queue length. The EWMA estimate of the error signal is then computed as: $\hat{e}(n) = (1-\beta)\hat{e}(n-1) + \beta e(n)$ where β is a DRED control parameter that smooths out the error signal. The drop probability is then calculated as: $p_d(n) = min\left(\max\left(p_d(n-1) + \alpha\frac{\hat{e}(n)}{B}, 0\right), \theta\right)$ where B is the buffer limit, α, another DRED control parameter for feedback gain, and θ, the maximum packet drop probability. DRED will not drop the packet if the queue length $q(n) < L$ where L is the no-drop threshold	Attempts to maintain the EWMA queue length around a given target so as to be load dependent

[10] J. Koo, B. Song, K. Chung, H. Lee, and H. Kahng, "MRED: a new approach to random early detection," in *Proceedings of the 15th International Conference on Information Networking*, Beppu City, Oita, Japan, 2001, pp. 347–52.

[11] S. Ryu, C. Rump, and C. Qiao, "Advances in active queue management (AQM) based TCP congestion control," *Telecommunication Systems - Modeling, Analysis, Design and Management*, vol. 25, no. 3-4, pp. 317–51, 2004.

[12] W. Feng, K. Shin, D. Kandlur, and D. Saha, "The BLUE active queue management algorithms," *IEEE/ACM Transactions on Networking*, vol. 10, no. 4, pp. 513–28, 2002.

[13] E. Park, H. Lim, K. Park, and C. Choi, "Analysis and design of the virtual rate control algorithm for stabilizing queues in TCP networks," *Computer Networks*, vol. 44, no. 1, pp. 17–41, 2004.

[14] C. Long, X. Guan, B. Zhao, and J. Yang, "The Yellow active queue management algorithm," *Computer Networks*, vol. 47, no. 4, pp. 525–50, 2005.

[15] C. Wang, B. Li, Y. Hou, K. Sohraby, and Y. Lin, "LRED: a robust active queue management scheme based on packet loss ratio," in *Proceedings of the Twenty-third Conference of the IEEE Communications Society (IEEE INFOCOM 2004)*, Hong Kong, China, 2004, pp. 1–12.

[16] H. Xu, T. Wu, X. Zhou, and Y. Zhu, "IP network control and AQM," in *Proceedings of the 2004 International Conference on Machine Learning and Cybernetics*, vol. 1, Shanghai, China, 2004, pp. 500–4.

[17] T. Wu, H. Xu, and S. Tian, "End-to-end congestion control and active queue management," in *Proceedings of the 2003 International Conference on Machine Learning and Cybernetics*, vol. 2, Xi'an, China, 2003, pp. 946–50.

[18] A. Afanasyev, N. Tilley, P. Reiher, and L. Kleinrock, "Host-to-host congestion control for TCP," *Communications Surveys & Tutorials, IEEE*, vol. 12, no. 3, pp. 304–342, 2010.

[19] C. Ku, S. Chen, J. Ho, and R. Chang, "Improving end-to-end performance by active queue management," in *Proceedings of the 19th International Conference on Advanced Information Networking and Applications*, vol. 2, Taipei, Taiwan, 2005, pp. 337–40.

[20] R. Pan, B. Prabhakar, and K. Psounis, "CHOKe - a stateless active queue management scheme for approximating fair bandwidth allocation," in *Proceedings of the Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies on Computer Communications (IEEE INFOCOM 2000)*, vol. 2, Tel Aviv, Israel, 2000, pp. 942–51.

TABLE XV

SUMMARY OF HEURISTIC AQM SCHEMES THAT IMPROVED STABILITY OR RESPONSIVENESS (CONT'D)

AQM	**Congestion Indicator**	**Control Function**	**Special Characteristics**
Double Slope RED (DSRED) [80]	Average (EWMA) queue length	$$p(\bar{q}) = \begin{cases} 0 & 0 \le \bar{q} \le K_l \\ \alpha(\bar{q} - K_l) & K_l \le \bar{q} \le K_m \\ 1 - \gamma + \beta(\bar{q} - K_m) & K_m \le \bar{q} \le K_h \\ 1 & \bar{q} \ge K_h \end{cases}$$ where $\alpha = \frac{2(1-\gamma)}{K_h - K_l}$, $\beta = \frac{2\gamma}{K_h - K_l}$ and $\bar{q} = (1 - w_q)\bar{q} + w_q q$. γ is a DSRED parameter called the mode-selector. K_l and K_h are equivalent (conceptually) to RED's $\min_{th}$ and $\max_{th}$ respectively. K_m is set to $\frac{K_l + K_h}{2}$.	Instead of having a single slope value between the two thresholds $\min_{th}$ and $\max_{th}$ in RED, DSRED has two.
Load/Delay Controllers [34]	EWMA load and queueing delay	$$p(\bar{q}, \bar{r}) = \alpha \min\left(1, \left(\frac{\bar{q}}{\bar{q}_{target}}\right)^n\right) + (1 - \alpha) \min\left(1, \left(\frac{\bar{r}}{\bar{r}_{target}}\right)^n\right)$$ where r, the instantaneous load factor, is the ratio of the arrival rate to the drain rate of the queue and $$\bar{q}(k\Delta) = (1 - w_q)\bar{q}((k-1)\Delta) + w_q q(k\Delta)$$ $$\bar{r}(k\Delta) = (1 - w_r)\bar{r}((k-1)\Delta) + w_r r(k\Delta)$$	A generic class of AQM algorithms which aims to provide better response to load changes. The EWMA queue length and load are calculated at fixed time intervals, Δ, instead of upon every packet arrival.
Loss Ratio based RED (LRED) [15]	instantaneous queue length and EWMA packet loss ratio	To actually measure the packet loss ratio, LRED first measures the number of packet drops and the number of packet arrivals in the previous M measurement intervals. The packet loss ratio for the k-th interval is calculated as: $$l(k) = \frac{\sum_{i=0}^{M-1} N_d(k - i - 1)}{\sum_{i=0}^{M-1} N_a(k - i - 1)}$$ where $N_d(j)$ is the number of packet drops in the j-th interval, and $N_a(j)$ is the number of packet arrivals in the j-th interval. The EWMA packet loss ratio is then calculated as $$\hat{l}(k) = w_l \hat{l}(k-1) + (1 - w_l) l(k)$$ where w_l is the EWMA weight which is set to a small value to more closely track the current packet loss ratio. The actual packet drop probability is calculated as $$p(q, \hat{l}) = \hat{l}(k) + \beta\sqrt{\hat{l}(k)}(q - q_0)$$ where β is another LRED parameter ($\beta > 0$).	According to [15], packet loss ratio is an important indicator of heavy congestion and its inclusion can make an AQM more robust and adaptive.
Modified RED (MRED) [10]	Average (EWMA) queue length, packet loss and link utilization	If the average queue size is between $\min_{th}$ and $\max_{th}$, the packet is dropped with probability p. This probability does not follow the linear function in RED, but is rather a step function, the decrements and increments of which (i.e. the actual step sizes) change with traffic. If there is an increase in $\bar{q}$, and the time lapse since the last update is greater than "uptime", the current drop probability p is incremented by d_1 (i.e. $p \leftarrow p + d_1$). When $\min_{th} < \bar{q} < \max_{th}$, and there is a decrease in $\bar{q}$, and the time lapse since the last update is greater than "uptime", the current drop probability p is decremented by d_2 (i.e. $p \leftarrow p - d_2$). Initially, d_1 is set to a large value and then reduced after repeated increases in the drop probability so that p converges to the maximum drop probability. On the other hand d_2 is increased with decreasing $\bar{q}$. d_1 is set to be larger than d_2.	RED structure at core

[21] Y. Jiang, M. Hamdi, and J. Liu, "Self adjustable CHOKe: an active queue management algorithm for congestion control and fair bandwidth allocation," in *Proceedings of the Eighth IEEE Symposium on Computers and Communications (ISCC 2003)*, vol. 2, Kemer-Antalya, Turkey, 2003, pp. 1018–25.

[22] J. Chung and M. Claypool, "Dynamic-CBT - Better performing active queue management for multimedia networking," in *Proceedings of the 10th ACM International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV)*, Chapel Hill, NC, USA, 2000.

[23] W. Sun and K. Shin, "TCP performance under aggregate fair queueing," in *Proceedings of the IEEE Global Telecommunications Conference (GLOBECOM '04)*, vol. 3, Dallas, TX, USA, 2004, pp. 1308–13.

[24] C. Brandauer, G. Iannaccone, C. Diot, T. Ziegler, S. Fdida, and M. May, "Comparison of tail drop and active queue management performance for bulk-data and Web-like Internet traffic," in *Proceedings of the Sixth IEEE Symposium on Computers and Communications*, Hammamet, Tunisia, 2001, pp. 122–9.

[25] J. Chung and M. Claypool, "Analysis of active queue management," in *Proceedings of the Second IEEE International Symposium on Network Computing and Applications (NCA 2003)*, Cambridge, MA, USA, 2003, pp. 359–66.

[26] N. Li, G. de Veciana, S. Park, M. Borrego, and S. Li, "Minimizing queue variance using randomized deterministic marking," in *Proceedings of the IEEE Global Telecommunications Conference (GLOBECOM'01)*, vol. 4, San Antonio, TX, USA, 2001, pp. 2368–72.

[27] M. Agarwal, R. Gupta, and V. Kargaonkar, "Link utilization based AQM and its performance," in *Proceedings of the IEEE Global Telecommunications Conference (GLOBECOM '04)*, vol. 2, Dallas, TX, USA, 2004, pp. 713–18.

[28] D. Lin and R. Morris, "Dynamics of random early detection," *Computer Communication Review*, vol. 27, no. 4, pp. 127–37, 1997.

TABLE XVI
SUMMARY OF HEURISTIC AQM SCHEMES THAT IMPROVED STABILITY OR RESPONSIVENESS (CONT'D)

AQM	**Congestion Indicator**	**Control Function**	**Special Characteristics**
Modified RED (MRED) [37]	Average (EWMA) queue length and instantaneous queue length	Upon each packet arrival, this MRED algorithm updates the EWMA queue length average, $\bar{q}$. MRED is essentially RED except for when $\bar{q} > \max_{th}$ and the instantaneous queue length $q > \max_{th}$ also. Then, the packet is dropped/marked with probability 1. This prevents packet dropping when the $\bar{q} > \max_{th}$ and the instantaneous $q < \max_{th}$.	RED structure at core. Developed independently from the aforementioned MRED algorithm.
BLUE [12]	occurrence of buffer overflow and buffer underflow	BLUE uses a single value for packet drop probability, p, which is periodically adjusted if necessary after a fixed time interval called the *freeze-time*. p is increased by a small fixed increment, δ_1, when there is a packet loss due to buffer overflow and decreased by a small fixed value, δ_2 when the queue becomes empty (i.e. additive increase additive decrease). Therefore all during the period that the queue is non-empty (but not full) the drop probability is constant.	Performance degradations due to multiple packet losses and periods of emptiness is unavoidable with BLUE [11]
Yellow [14]	link utilization	The QCF is denoted by the function $f(q)$ where q is the queue length, and q_{ref} is the reference queue length. $$f(q) = \begin{cases} \max\left(\mathit{QDLF}, \frac{\gamma\alpha q_{ref}}{(\alpha-1)q+q_{ref}}\right) & q > q_{ref} \\ \frac{\gamma\beta q_{ref}}{(\beta-1)q+q_{ref}} & 0 \leq q \leq q_{ref} \end{cases}$$ where *QDLF* is the queue drain limit factor, γ is the link utilization, α and β are two parameters that provide the tradeoff between responsiveness and stability of the Yellow algorithm. The load factor (which is the congestion measure for Yellow), is $z = \frac{\textit{link input rate}}{\tilde{c}}$ where $\tilde{c}$ is the virtual available capcity computed as $\tilde{c} = f(q)c$ where c is the link capacity and $f(q)$ is as given above. z can be considered a mismatch between the virtual capacity and the input rate. The dropping probability function is then given as the recursive form: $$p = \begin{cases} p + \frac{z\Delta}{c} & z \geq 1+\delta \\ p - \frac{\Delta}{zc} & z < 1 \\ p & \text{otherwise} \end{cases}$$ Now, $[1, 1+\delta]$ is the range of desired link utilization. If the input rate exceeds the virtual capacity by δ, the dropping probability increases and if the input rate is less than the virtual capacity, the dropping probability decreases.	To improve the transient performance when there are sudden changes in network load, a secondary queue control function (QCF) is used. This is unlike other AQM schemes which use both load and queue length as peer congestion measures.
DREAM [26]	—	modifies only the marking algorithm, i.e. the function that decides which packet is actually going to be dropped/marked given a dropping/marking probability, p_b. Instead of Random Marking (RM), Uniform Random Marking (URM), Wait Uniform Random Marking (WURM) and Slow Random Marking (SRM), it uses the following function $$p_a(n) = \begin{cases} 0 & \text{if } n \times p_b < 1 \\ 1 & \text{otherwise} \end{cases}$$	

[29] B. Suter, T. Lakshman, D. Stiliadis, and A. Choudhury, "Design considerations for supporting TCP with per-flow queueing," in *Proceedings of the Seventeenth Annual Joint Conference of the IEEE Computer and Communications Societies on Computer Communications (IEEE INFOCOM '98)*, vol. 1, San Francisco, CA, USA, 1998, pp. 299–306.

[30] C. Zhu, O. Yang, J. Aweya, M. Ouellette, and D. Montuno, "A comparison of active queue management algorithms using the OPNET modeler," *IEEE Communications Magazine*, vol. 40, no. 6, pp. 158–67, 2002.

[31] R. Pletka, M. Waldvogel, and S. Mannal, "PURPLE: predictive active queue management utilizing congestion information," in *Proceedings of the 28th Annual IEEE International Conference on Local Computer Networks (LCN 2003) held in conjunction with the Workshop on High-Speed Local Networks (HSLN) and the Workshop on Wireless Local Networks (WLN 2003)*, Bonn/Konigswinter, Germany, 2003, pp. 21–30.

[32] S. Oruganti and M. Devetsikiotis, "Analyzing robust active queue management schemes: a comparative study of predictors and controllers," in *Proceedings of the 2003 IEEE International Conference on Communications*, vol. 3, Anchorage, AK, USA, 2003, pp. 1531–6.

[33] C. Hollot, Y. Liu, V. Misra, and D. Towsley, "Unresponsive flows and AQM performance," in *Proceedings of the Twenty-second Annual Joint Conference of the IEEE Computer and Communications Societies (IEEE INFOCOM 2003)*, vol. 1, San Francisco, CA, USA, 2003, pp. 85–95.

[34] M. Kwon and S. Fahmy, "Comparison of load-based and queue-based active queue management algorithms," in *Proceedings of the SPIE - The International Society for Optical Engineering*, vol. 4866, Boston, MA, USA, 2002, pp. 35–46.

[35] W. Wu, Y. Ren, and X. Shan, "Stability analysis on active queue management algorithms in routers," in *Proceedings of the Ninth International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS 2001)*, Cincinnati, OH, USA, 2001, pp. 125–32.

[36] A. Bitorika, M. Robin, and M. Huggard, "An evaluation framework for active queue management schemes," in *Proceedings of the 11th IEEE/ACM International Symposium on Modeling Analysis and Simulation of Computer Telecommunications Systems (MASCOTS 2003)*, Orlando, FL, USA, 2003, pp. 200–6.

[37] G. Feng, A. Agarwal, A. Jayaraman, and C. Siew, "Modified RED gateways under bursty traffic," *IEEE Communications Letters*, vol. 8, no. 5, pp. 323–5, 2004.

[38] R. Zhu, H. Teng, and J. Fu, "A predictive PID controller for AQM router supporting TCP with ECN," in *Proceedings of the 2004 Joint Conference of the 10th Asia-Pacific Conference on Communications and the 5th International Symposium on Multi-Dimensional Mobile Communications Proceeding (APCC/MDMC '04)*, vol. 1, Beijing, China, 2004, pp. 356–60.

[39] R. Cartas, J. Orozco, J. Incera, and D. Ros, "A fairness study of the

TABLE XVII
SUMMARY OF HEURISTIC AQM SCHEMES WITH AUTO-TUNING

AQM	Congestion Indicator	Control Function	Special Characteristics
Adaptive RED (ARED) [9], [189]	Average (EWMA) queue length	1) $P_{\max}$ is adjusted so as to keep the average queue length at the half-way point between $\min_{th}$ and $\max_{th}$, not just anywhere between $\min_{th}$ and $\max_{th}$. 2) Instead of using Multiplicative Increase Multiplicative Decrease (MIMD) for adjustment of $P_{\max}$, Additive Increase Multiplicative Decrease (AIMD) is used. 3) $P_{\max}$ is adjusted slowly in small steps (in the order of round-trip times) 4) $P_{\max}$ is not allowed to exceed 0.5 or to fall below 0.01. 5) $\max_{th}$ is set to $3 \times \min_{th}$ 6) w_q to $1 - e^{-\frac{1}{C}}$ where C is the link capacity in packets/second. 7) $\min_{th} = \max\left[5, \frac{delay \times C}{2}\right]$ assuming a default round-trip time of 100 ms. 8) Recommended values for the increase constant, α, and the decrease factor, β are $\alpha < 0.25$ and $\beta > 0.83$	According to [9], ARED can achieve a target queue length set at around $\frac{\min_{th} + \max_{th}}{2}$ providing predictable average queueing delay and preventing overshoot of $\max_{th}$ with the associated higher packet loss. A nonlinear analysis was performed on RED and ARED by [82], so as to determine how their performance is affected by their nonlinear dynamical behaviour. It was found that ARED performed much better than RED on most occasions, having smaller maximal Lyapunov exponents and lower Hurst parameters. It was also found by [19] that ARED converged more quickly than RED.
RARED [19]	Average (EWMA) queue length and average (EWMA) input rate	RARED uses the EWMA queue length at the current and two previous packet arrivals to estimate the input rate. The EWMA rate $\bar{r}(t_i)$ can be computed as $$\bar{r}(t_i) = \frac{\bar{q}(t_i) - \bar{q}(t_{i-1})}{t_i - t_{i-1}}$$ where t_i is the arrival time of the i-th packet and $\bar{q}(t_i)$ is the EWMA queue length computed at t_i. However, the computation was simplified by [19] considering the climb in input rate so that if $$[\bar{q}(t_i) - \bar{q}(t_{i-1})] - [\bar{q}(t_{i-1}) - \bar{q}(t_{i-2})] > \delta$$ (where δ is a parameter to be configured), then $P_{\max}$ is changed to a higher value $P^r_{\max}$, otherwise it remains the same as in the original RED scheme.	The basic structure of RED is preserved. The choice of δ determines how aggressive the algorithm will be to increasing traffic loads. A large δ will make it too sluggish, whereas a small value may make it too aggressive. A $\delta = 1$ was suggested by [19] as the largest value they found to not degrade performance. They also suggested a $P^r_{\max} = 0.8$. Anything greater does not cause much improvement.
RED-Worcester [190]	Average queue length	RED Worcester extends ARED directly so as to make it more sensitive to the average QoS demands of the incoming traffic. ARED maintains the queue-length at a fixed target so as to keep the queueing delay more or less constant for all the flows passing through. However, different applications require different delay bounds, for example real-time traffic is highly delay sensitive. RED Worcester adapts this target queue length to more closely reflect the average nature of the traffic, so that if the traffic is predominantly delay-sensitive the queue-length target will be lowered, and if it is predominantly throughput-sensitive the queue-length target will be increased. Sources use delay hints to indicate a bound on queueing delay for the flow. RED Worcester does not guarantee that this delay bound will be met, it just uses it to get an average sense of the aggregate tradeoff between throughput and delay. The queue target is set as the EWMA of the delay hints of the incoming packets.	RED-Worcester is not a differential service and therefore does not require any components associated with such (e.g. policing, charging, etc.). However, it is claimed by [190] that RED Worcester improves the QoS of delay-sensitive flows as their proportion in the traffic mix increases.

adaptive RIO active queue management algorithm," in *Proceedings of the Fifth Mexican International Conference in Computer Science*, Colima, Mexico, 2004, pp. 57–63.

[40] J. Sun, K.-T. Ko, G. Chen, S. Chan, and M. Zukerman, "PD-RED: to improve the performance of RED," *IEEE Communications Letters*, vol. 7, no. 8, pp. 406–8, 2003.

[41] L. Le, J. Aikat, K. Jeffay, and F. Smith, "The effects of active queue management on Web performance," *Computer Communication Review*, vol. 33, no. 4, pp. 265–76, 2003.

[42] F. Ren, C. Lin, and B. Wei, "Design a robust controller for active queue management in large delay networks," in *Proceedings of the Ninth International Symposium on Computers And Communications (ISCC 2004)*, vol. 2, Alexandria, Egypt, 2004, pp. 748–54.

[43] F. Yanfie, R. Fengyuan, and L. Chuang, "Design a PID controller for active queue management," in *Proceedings of the Eighth IEEE Symposium on Computers and Communications (ISCC 2003)*, vol. 2, Kemer-Antalya, Turkey, 2003, pp. 985–90.

[44] S. Athuraliya, S. Low, V. Li, and Q. Yin, "REM: active queue management," *IEEE Network*, vol. 15, no. 3, pp. 48–53, 2001.

[45] S. Kunniyur and R. Srikant, "An adaptive virtual queue (AVQ) algorithm for active queue management," *IEEE/ACM Transactions on Networking*, vol. 12, no. 2, pp. 286–99, 2004.

[46] H. Han, C. Hollot, Y. Chait, and V. Misra, "TCP networks stabilized by buffer-based AQMs," in *Proceedings of the Twenty-third Conference of the IEEE Communications Society (IEEE INFOCOM 2004)*, vol. 2, Hong Kong, China, 2004, pp. 964–74.

[47] J. Chung and M. Claypool, "Rate-based active queue management with priority classes for better video transmission," in *Proceedings of the Seventh International Symposium on Computers and Communications (ISCC 2002)*, Taormina-Giardini Naxos, Italy, 2002, pp. 99–105.

[48] K. Kim, A. Tang, and S. Low, "Design of AQM in supporting TCP based on the well-known AIMD model," in *Proceedings of the IEEE Global Telecommunications Conference (GLOBECOM '03)*, vol. 6, San Francisco, CA, USA, 2003, pp. 3226–30.

[49] V. Firoiu and M. Borden, "A study of active queue management for congestion control," in *Proceedings of the Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies (IEEE INFOCOM 2000)*, vol. 3, Tel Aviv, Israel, 2000, pp. 1435–44.

[50] B. Wydrowski and M. Zukennan, "On the transition to a low latency TCP/IP Internet," in *Proceedings of the 2002 IEEE International Conference on Communications (ICC 2002)*, vol. 4, New York, NY, USA, 2002, pp. 2631–5.

[51] X. Wang and M. Azizoglu, "On the dropping probability function in active queue management schemes," in *Proceedings of the IEEE Global Telecommunications Conference (GLOBECOM'01)*, vol. 4, San Antonio, TX, USA, 2001, pp. 2511–16.

TABLE XVIII
SUMMARY OF HEURISTIC AQM SCHEMES WITH AUTO-TUNING (CONT'D)

AQM	**Congestion Indicator**	**Control Function**	**Special Characteristics**
Random Early Adaptive Detection [8]	Average (EWMA) queue length and EWMA queue slope	Besides calculating the EWMA of the queue length as in RED, READ also calculates its EWMA slope so as to track the sign changes in the average queue length and so track its local maximum and minimum points. $$\bar{q}_{t+1} = (1 - w_q)\bar{q}_t + w_q q_t$$ $$\bar{sl}_{t+1} = (1 - w_{sl})\bar{sl}_t + w_{sl}(\bar{q}_{t+1} - \bar{q}_t)$$ where $w_q = 0.002$ and $w_l \approx 0.001$ At each sign change in the slope sl, the value of the local maxmimum ($\max$) and the local minimum ($\min$) are recorded. (Note: these are not $\min_{th}$ nor $\max_{th}$.) From this the level around which the queue length oscillates is estimated. This queue level (*level*) is updated on every change in the local minimum as $level = \frac{\max + \min}{2}$. The dropping probability is then increased or decreased according to the value of *level*: $$p = \begin{cases} p + INC & level > buffer \times 0.52 \\ p - DEC & level < buffer \times 0.48 \\ p & \text{otherwise} \end{cases}$$ We see here that there are no maximum or minimum thresholds and the attempt is to keep the queue length halfway through the queue. The authors chose the parameters $INC = 0.02$ and $DEC = 0.002$. They set $INC > DEC$ so that the algorithm more aggressively acts when there is the onset of congestion.	READ, according to [8], achieves higher power levels than RED (where power is defined as $\frac{throughput^{\alpha}}{response\ time}$) and it adapts automatically to traffic loads.
PSAND [77]	average queue length	Like ARED it adjusts $P_{\max}$ based on the average queue length. However, unlike ARED, the multiplicative factor that it uses is not fixed but dynamically adjusts to changing queue size and the deviation from the target queue size. This multiplicative factor is calculated as: $$r = \left(\frac{\hat{K}_{cur}}{\hat{K}_{prev}}\right)\left(\frac{\hat{K}_{cur}}{\hat{K}_{T}}\right)$$ where $\hat{K}_{cur}$ is the weighted average queue size in the current interval, $\hat{K}_{prev}$ is the weighted average queue size in the previous interval and $\hat{K}_T$ is the target weighted average queue size. These are calculated at a regular interval not on every packet arrival. It was found that the multiplicative factor r was not aggressive enough, so instead they, the authors of [77] modified the multiplicative factor to $$\beta = coef \times r^{\gamma}$$ where r is the same as before and γ and *coef* are two new parameters that must also be tuned. They used values of $coef = 1.75$ and $\gamma = 1.5$ based on extensive simulations. $P_{\max}$ is therefore adjusted $$P_{\max} = P_{\max} \times \beta$$ It was also recommended by [77] that the RED parameters $\min_{th}$ and $\max_{th}$ should be set as follows: $$\min_{th} = \begin{cases} 2\hat{K}_T - B & \hat{K}_T > \frac{B}{2} \\ 0 & \hat{K}_T \leq \frac{B}{2} \end{cases}$$ $$\max_{th} = \begin{cases} B & \hat{K}_T > \frac{B}{2} \\ 2\hat{K}_T & \hat{K}_T \leq \frac{B}{2} \end{cases}$$	The aim of PSAND is to improve the queueing delay and delay jitter performance of ARED without jeopardizing its loss rate performance. RED is still the core of PSAND

[52] X. Deng, S. Yi, G. Kesidis, and C. Das, "Stabilized virtual buffer (SVB) - an active queue management scheme for Internet quality-of-service," in *Proceedings of the IEEE Global Telecommunications Conference (GLOBECOM '02)*, vol. 2, 2002, pp. 1628–32.

[53] V. Misra, G. W., and D. Towsley, "Fluid-based analysis of a network of AQM routers supporting TCP flows with an application to RED," *Computer Communication Review*, vol. 30, no. 4, pp. 151–60, 2000.

[54] N. Cardwell, S. Savage, and T. Anderson, "Modeling the performance of short TCP connections," Computer Science Department, Washington University, Tech. Rep.

[55] C. Hollot, V. Misra, D. Towsley, and W. Gong, "A control theoretic analysis of RED," in *Proceedings of the Twentieth Annual Joint Conference of the IEEE Computer and Communications Society (IEEE INFOCOM 2001).*, vol. 3, Anchorage, AK, USA, 2001, pp. 1510–19.

[56] ——, "On designing improved controllers for AQM routers supporting TCP flows," in *Proceedings of the Twentieth Annual Joint Conference of the IEEE Computer and Communications Society (IEEE INFOCOM 2001).*, vol. 3, Anchorage, AK, USA, 2001, pp. 1726–34.

[57] F. Kelly, A. Maulloo, and D. Tan, "Rate control for communication networks: shadow prices, proportional fairness and stability," *Journal of the Operational Research Society*, vol. 49, no. 3, pp. 237–52, 1998.

[58] F. Kelly, "Mathematical modelling of the Internet," in *Mathematics Unlimited – 2001 and Beyond*, B. Engquist and W. Schmid, Eds. Berlin: Springer-Verlag, 2001, pp. 685–702. [Online]. Available: citeseer.ist.psu.edu/kelly99mathematical.html

[59] S. Low and D. Lapsley, "Optimization flow control — I: basic algorithm and convergence," *IEEE/ACM Transactions on Networking*, vol. 7, no. 6, pp. 861–874, 1999. [Online]. Available:

TABLE XIX
SUMMARY OF HEURISTIC AQM SCHEMES FOR IMPROVED FAIRNESS

AQM	**Congestion Indicator**	**Control Function**	**Special Characteristics**
Fair RED (FRED) [28]	average queue length	FRED maintains state for each active flow[6] based on how much of the buffer they currently occupy [27], [34], [28], [70]. The memory requirements for this state information is proportional to the buffer size and is independent of the total number of flows [28] in that the maximum number of flows that can be supported at any one time is equal to the size of the buffer (in packets). A flow will not experience loss in a FRED queue if its buffer occupancy (denoted by $qlen_i$)is less than $\min_q$ and the average queue length is less than $\max_{th}$. If the buffer occupancy of a flow is greater than $\max_q$ or the average queue length is greater than $\max_{th}$, then its packets will be dropped with a probability of 1. Otherwise the flow will be subject to RED's dropping policy. $\min_q$ is dynamically adjusted to the global variable *avgcq* when the number of active flows is small. *avgcq* is the average per-connection queue length (i.e. the average queue length divided by the number of active connections). FRED keeps a count of how many times a flow has exceeded $\max_q$ in the variable called *strike*. Flows with high *strike* values are not allowed to have more than *avgcq* packets at a time in the queue. It should be noted that the average queue length calculated in FRED is slightly different to that in RED. In RED it is calculated only on packet arrivals, however, FRED calculates it on both packet arrival and departures.	FRED attempts to punish "mis-behaving" flows so as to protect "well-behaved" flows. See also [31], [70], [22]. According to R-1 FRED is for the most part fairer than RED. However, according to [70], [69], in many cases FRED is unfair. Also the algorithm needs large buffer space to sufficiently decipher non-responsiveness [12]. FRED is memoryless in that a flow is immediately reclassified as responsive once all its packets clear the buffer [12].
Balanced RED (BRED) [69]		For each flow i that is currently in the buffer, BRED keeps two state variables: 1) $qlen_i$ — the number of packets of flow i in the buffer. 2) gap_i — the number of packets accepted from flow i since a packet from flow i was dropped. BRED has five parameters: l_1, p_1, l_2, p_2 and W_m. W_m is the maximum number of packets that any flow can have in the buffer. Therefore packets are dropped with probability 1 when $qlen_i$ exceeds W_m. Packets are dropped with a probability p_2 once the number of packets belonging to a flow exceeds l_2 but is less than W_m and $gap_i > l_2$. Packets are dropped with an even smaller probability p_1 if the number of packets belonging to the flow exceeds l_1 but is less than l_2 and $gap_i > l_1$. In all other cases, the packet is accepted into the buffer. The main motivation for gap_i is to prevent multiple consecutive drops to which the performance of TCP Reno flows are highly vulnerable. BRED also maintains another variable, N_{active} which is a measure of the number of flows currently in the buffer. The actual values of the parameters are set as follows: $$l_2 = \frac{B}{2\lceil \hat{N} \rceil}$$ $$p_2 = \frac{\sqrt{\lceil \hat{N} \rceil}}{\sqrt{\lceil \hat{N} \rceil} + 10}$$ $$p_1 = \frac{p_2}{10}$$ where $\hat{N}$ is EWMA of the number of flows currently in the buffer. It is given by: $$\hat{N} = (1 - w_n)\hat{N} + w_n N_{active}$$ where $0 < w_n < 1$. The authors of BRED have chosen $w_n = 0.02$.	BRED attempts to protect adaptive flows from non-adaptive flows using minimal flow state information. The performance of BRED is more sensitive to p_2 than to p_1. Also, of the three thresholds, BRED is most sensitive to W_m. In [65] BRED was augmented by a virtual buffer (henceforth referred to as BRED/VBO) to provide fairer bandwidth allocation. The virtual buffer occupancy represents that of a queue which has a round-robin service discipline so that the average buffer utilization of each flow is equal. Instead of dropping packets based on the real queue occupancy (as in BRED), BRED/VBO drops packets based on the virtual buffer occupancy.

citeseer.ist.psu.edu/article/low99optimization.html
[60] S. Kunniyur and R. Srikant, "Analysis and design of an adaptive virtual queue (AVQ) algorithm for active queue management," *Computer Communication Review*, vol. 31, no. 4, pp. 123–34, 2001.
[61] ——, "An adaptive virtual queue (AVQ) algorithm for active queue management," *IEEE/ACM Transactions on Networking*, vol. 12, no. 2, pp. 286–99, 2004.
[62] E. Bowen, C. Jeffries, L. Kencl, A. Kind, and R. Pletka, "Bandwidth allocation for non-responsive flows with active queue management," in *Proceedings of the 2002 International Zurich Seminar on Broadband Communications Access - Transmission - Networking*, Zurich, Switzerland, 2002, pp. 13–1.
[63] S. Ryu and C. Cho, "PI-PD-controller for robust and adaptive queue management for supporting TCP congestion control," in *Proceedings of the 37th Annual Simulation Symposium*, Arlington, VA, USA, 2004, pp. 132–9.
[64] D. Bauso, L. Giarre, and G. Neglia, "About the stability of active queue management mechanisms," in *Proceedings of the 2004 American Control Conference*, vol. 4, Boston, MA, USA, 2004, pp. 2954–9.
[65] M. Nabeshima, "Improving the performance of active buffer management with per-flow information," *IEEE Communications Letters*, vol. 6, no. 7, pp. 306–8, 2002.
[66] G. Chatranon, M. Labrador, and S. Banerjee, "Fairness of AQM schemes for TCP-friendly traffic," in *Proceedings of the IEEE Global Telecommunications Conference (GLOBECOM '04)*, vol. 2, Dallas, TX, USA, 2004, pp. 725–31.
[67] P. Hurley, J.-Y. Le Boudec, P. Thiran, and M. Kara, "ABE: providing a low-delay service within best effort," *IEEE Network*, vol. 15, no. 3, pp. 60–9, 2001.
[68] R. Pletka, A. Kind, M. Waldvogel, and S. Mannal, "Closed-loop congestion control for mixed responsive and non-responsive traffic," in *Proceedings of the IEEE Global Telecommunications Conference (GLOBECOM '03)*, vol. 7, San Francisco, CA, USA, 2003, pp. 4180–5.

TABLE XX
SUMMARY OF HEURISTIC AQM SCHEMES FOR IMPROVED FAIRNESS (CONT'D)

AQM	**Congestion Indicator**	**Control Function**	**Special Characteristics**
Short-lived Flow Friendly RED (SHRED) [73]		For each arriving packet, the router will then compute the EWMA congestion window (cwnd) as: $$cwnd_{avg} \leftarrow (1-w_c)cwnd_{avg} + w_c cwnd_{sample}$$ where $cwnd_{sample}$ is the cwnd extracted from the arriving packet. The RED parameters $\min_{th}$ and $P_{\max}$ are then modified using the congestion window information, but the slope of the RED characteristic is maintained . The modifications are as follows: $$\min_{th-mod} = \min_{th} + (\max_{th} - \min_{th}) \left(1 - \frac{cwnd_{sample}}{cwnd_{avg}}\right)$$ $$P_{\max -mod} = P_{\max} \left(\frac{\max_{th} - \min_{th\text{-}mod}}{\max_{th} - \min_{th}}\right)$$ $$p(\bar{q}) = \begin{cases} 0 & 0 \le \bar{q} \le \min_{th-mod} \\ \frac{\bar{q} - \min_{th-mod}}{\max_{th} - \min_{th-mod}} P_{\max -mod} & \min_{th-mod} \le \bar{q} \le \max_{th} \\ 1 & \bar{q} \ge \max_{th} \end{cases}$$ Therefore a packet with a congestion window greater than the average will have a modified $\min_{th}$ lower than the original $\min_{th}$ and will therefore experience a higher drop probability. A packet with a congestion window less than the average will experience a lower drop probability.	SHRED attempts to improve fairness for short-lived TCP flows such as HTTP traffic. With SHRED, flows with smaller congestion windows (the typical situation for short-lived flows) experience lower dropping probabilities than those with larger windows. For SHRED to work, however, the router must know what the current congestion window is for a flow. This can be facilitated by placing this information in the IP header and have the router extract it.
CHOKe [20]	Average (EWMA) queue length	CHOKe both identifies and penalizes unresponsive flows. CHOKe does not store any information regarding active flows whether it be their flow ID or their number. It derives all its information from the queue occupancy itself. Just like RED it calculates the average queue length by EWMA. It also has the two thresholds $\min_{th}$ and $\max_{th}$. No packet from any flow is dropped if the average queue length is less than $\min_{th}$. Every packet is dropped if the average queue length is greater than $\max_{th}$. However, if $\min_{th} < \bar{q} < \max_{th}$, the flow ID of an arriving packet is compared to that of a randomly chosen packet from the queue. If the flow ID matches, both the arriving packet and the packet that was chosen for comparison are dropped. If not, the arriving packet is dropped with a probability according to the RED control function. Note this double dropping comparison is done even when the average queue length is greater than $\max_{th}$. CHOKe fairness enforcement is based on the premise that unresponsive flows will have a high buffer occupancy and a high incoming rate, so that the likelihood of choosing a packet at random from the queue that belongs to an unresponsive flow is higher. CHOKe can be generalized to instead choose (and drop) m packets from the queue instead of 1. Having $m > 1$ increases CHOKe performance when the number of unresponsive flows increases, since the chances of catching an unresponsive flow becomes larger. However, to choose a suitable value for m especially in light of varying traffic characteristics, may require some state information. To work around this and to maintain CHOKe as completely stateless, the authors of CHOKe decided to divide the region between $\min_{th}$ and $\max_{th}$ into k separate intervals, and to associate with each of these intervals (starting from $\min_{th}$ to $\max_{th}$ an increasing value of m). In other words, m increases monotonically with average queue size.	The CHOKe (CHOose and Keep for responsive flows, CHOose and Kill for unresponsive flows) algorithm (proposed by [20]) is a simple extension of REDthat seeks to achieve max-min fairness without any state information whatsoever [70], [21], [20]. Another version of CHOKe called ECHOKe in which the RED mechanism is replaced by another AQM scheme called Random Early Marking (REM) was proposed by [16]. Yet another version called Self Adjustable CHOKe (or SAC) was proposed by [21].

[69] F. Anjum and L. Tassiulas, "Fair bandwidth sharing among adaptive and non-adaptive flows in the Internet," in *Proceedings of the Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies on Computer Communications (IEEE INFOCOM '99)*, vol. 3, New York, NY, USA, 1999, pp. 1412–20.

[70] A. Kamra, S. Kapila, V. Khurana, V. Yadav, H. Saran, S. Juneja, and R. Shorey, "SFED: A rate control based active queue management discipline," IBM India Research Laboratory, Tech. Rep. 00A018, 2000.

[71] M. Handley, J. Padhye, and S. Floyd, "TCP friendly rate control (TFRC): Protocol specification," *IETF RFC 3448*, 2003. [Online]. Available: citeseer.ist.psu.edu/article/handley01tcp.html

[72] G. Chatranon, M. Labrador, and S. Banerjee, "BLACK: detection and preferential dropping of high bandwidth unresponsive flows," in *Proceedings of the 2003 IEEE International Conference on Communications*, vol. 1, Anchorage, AK, USA, 2003, pp. 664–8.

[73] M. Claypool, R. Kinicki, and M. Hartling, "Active queue management for Web traffic," in *Proceedings of the 2004 IEEE International Performance, Computing, and Communications Conference*, Phoenix, AZ, USA, 2004, pp. 531–8.

[74] S. De Cnodder, K. Pauwels, and O. Elloumi, "A rate based RED mechanism," in *Proceedings of the 10th ACM International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV)*, Chapel Hill, NC, USA, 2000.

[75] J. Orozco and D. Ros, "An adaptive RIO (A-RIO) queue management algorithm," in *Quality for All*, ser. Lecture Notes in Computer Science, G. Karlsson and M. Smirnov, Eds. Springer-Verlag Berlin/Heidelberg, 2003, vol. 2811, pp. 11–20.

[76] M. Kisimoto, H. Ohsaki, and M. Murata, "On transient behavior analysis of random early detection gateway using a control theoretic approach," in *Proceedings of the 2002 IEEE International Conference on Control Applications*, vol. 2, Glasgow, UK, 2002, pp. 1144–6.

[77] T. Alemu and A. Jean-Marie, "Dynamic configuration of RED parameters," in *Proceedings of the IEEE Global Telecommunications Conference (GLOBECOM '04)*, vol. 3, Dallas, TX, USA, 2004, pp. 1600–4.

[78] T. Eguchi, H. Ohsaki, and M. Murata, "On control parameters tuning

TABLE XXI
SUMMARY OF HEURISTIC AQM SCHEMES FOR IMPROVED FAIRNESS (CONT'D)

AQM	Congestion Indicator	Control Function	Special Characteristics
GREEN [191]	Rate-based	Based on an analytic model developed by Mathis et al. [192] for the steady-state behaviour of TCP, it determines the packet dropping probability. This TCP model for the throughput (T) of a long-lived flow is as follows: $$T = \frac{c \times s}{rtt\sqrt{p}}$$ where c is a constant that depends on the acknowledgment strategy (i.e. acknowledgment of every packet of delayed acknowledgment), s is the maximum segment size, *rtt* is the round-trip time of the connection and p is the packet loss probability. If L is the capacity of the outgoing link, and N is the number of active flows at the queue (where an active flow has had at least one packet in the queue during a given time-interval), then the fair-share throughput of each flow is $\frac{L}{N}$ to which T, the throughput of a given flow should be roughly equal. Substituting this into the TCP model and making p the subject of the formula: $$\frac{L}{N} = \frac{c \times s}{rtt\sqrt{p}} \Rightarrow p = \left(\frac{c \times s \times N}{L \times rtt}\right)^2$$ It can be deduced that as N increases or *rtt* decreases, p, the dropping probability will increase. Having p dependent on N and *rtt* makes the overall AQM performance independent of load and more fair to connections with larger *rtt* without maintaining per-flow state. GREEN slows down flows with low *rtt*	To work, GREEN needs to determine s, *rtt*, and N. According to [191], the router estimates s by looking at the size of each packet. N is estimated by counting the number of flows that have at least one packet pass through within a certain time-frame. If the duration this time interval is too long, then N will be high and the overall link utilization will drop. If too short, the number of flows will be underestimated and the allowed throughput per flow will be made higher than it should be. To estimate the *rtt* at the router, the TCP sender transmits this *rtt* information in the IP header which the router will retrieve. As an alternative, it was suggested by [191] that the router employ IDMaps tracer and maintain a database of *rtt* estimates for each flow identified by source and destination IP addresses.
Stochastic Fair BLUE [193]		An extension to the BLUE algorithm ([68], [62], [12]) it attempts to protect responsive TCP flows from non-responsive flows by rate-limiting the latter to their fair share. To perform this rate-limiting, it must first identify the non-responsive flows. It does this by employing a bloom filter with multiple levels of independent hash functions [68], [66], [12]. This requires a small amount of state (and hence some additional memory resources) but not on a per-flow basis. SFB has L different levels with N accounting bins within each level. Associated with each bin is a dropping probability which is based on the queue occupancy for the flows that have been hashed into that bin. There are L independent hash functions, one per level, which maps a flow based on its flow ID (source address, destination address, source port, destination port, protocol) into one of the N accounting bins in that level. A flow is therefore mapped into L-bins, one per level. The dropping probability is increased as the bin occupancy increases and is decreased as the occupancy decreases according to the BLUE algorithm. A non-responsive flow will typically push the dropping probability to one for the bins it occupies. It is true that a responsive flow may occupy some of the bins that a non-responsive flow occupies, but provided that the number of non-responsive flows is much smaller than the number of available bins it is likely that the responsive flow will occupy unpolluted bins which will have lower dropping probability values. The actual dropping probability a flow will experience will be the minimum across all the bins it occupies. If this value is equal to one, SFB identifies this flow as non-responsive and rate-limits it.	To address misclassification problem, it was suggested by [12] the use of "moving" hash functions. The hash functions are changed and the bins reset at random or periodic times effectively creating virtual bins across time. A non-responsive flow will always behave badly with every reconfiguration, whereas a responsive flow will lose the risk of being misclassified. However, at the time point of reconfiguration, non-responsive flows will be temporarily treated as responsive flows. To remedy this, it was then suggested by [12] that "double-buffered moving" hash functions be used for which there are two sets of bins instead of one.

for active queue management mechanisms using multivariate analysis," in *Proceedings of the 2003 Symposium on Applications and the Internet*, Orlando, FL, USA, 2003, pp. 120–7.

[79] T. Ziegler, "On averaging for active queue management congestion avoidance," in *Proceedings of the Seventh International Symposium on Computers and Communications (ISCC 2002)*, Taormina-Giardini Naxos, Italy, 2002, pp. 867–73.

[80] B. Zheng and M. Atiquzzaman, "DSRED: improving performance of active queue management over heterogeneous networks," in *Proceedings of the IEEE International Conference on Communications (ICC 2001)*, vol. 8, Helsinki, Finland, 2001, pp. 2375–9.

[81] Y. Chait, C. V. Hollot, and V. Misra, "Fixed and adaptive model-based controllers for active queue management," in *Proceedings of the American Control Conference*, Arlington, VA, USA, 2001, pp. 2981–6.

[82] K. Jiang, X. Wang, and Y. Xi, "Nonlinear analysis of RED - a comparative study," in *Proceedings of the 2004 American Control Conference*, vol. 4, 2004, pp. 2960–5.

[83] J. Aweya, M. Ouellette, D. Montuno, and A. Chapman, "An adaptive buffer management mechanism for improving TCP behavior under heavy load," in *Proceedings of the IEEE International Conference on Communications*, vol. 10, Helsinki, 2001, pp. 3217–23.

[84] W. Li, L. Zeng-zhi, and C. Yan-ping, "A control theoretic analysis of mixed TCP and UDP traffic under RED based on nonlinear dynamic model," in *Proceedings of the Third International Conference on Information Technology and Applications (ICITA 2005)*, vol. 2, July 2005, pp. 747–750.

[85] Y. Hong and O. Yang, "Design of TCP traffic controllers for AQM routers based on phase margin specification," in *Proceedings of the 2004 Workshop on High Performance Switching and Routing*, Phoenix, AZ, USA, 2004, pp. 314–18.

[86] Y. Hong, O. Yang, and C. Huang, "Self-tuning PI TCP flow controller for AQM routers with interval gain and phase margin assignment," in *Proceedings of the IEEE Global Telecommunications Conference (GLOBECOM '04)*, vol. 3, November 2004, pp. 1324–1328.

[87] Y. Hong and O. Yang, "Self-tuning TCP traffic controller using gain margin specification," *Communications, IET*, vol. 1, no. 1, pp. 27–33,

TABLE XXII
SUMMARY OF HEURISTIC AQM SCHEMES FOR IMPROVED FAIRNESS (CONT'D)

AQM	**Congestion Indicator**	**Control Function**	**Special Characteristics**
BLACK [72]		Based on the premise that a FIFO queue approximately assigns to a flow bandwidth proportional to its buffer occupancy, BLACK uses a buffer occupancy fraction as an indicator of bandwidth share and attempts to equalize this bandwidth share among the flows so as to improve fairness among them. BLACK uses limited state information to estimate the buffer occupancy fraction. It uses cache memory (called High Bandwidth Flows (HBF) cache memory) to store the flow id of those flows that have occupied the queue. Upon each packet arrival, and if the queue is above a certain threshold, the router randomly picks a packet from the queue. If the flow ids are the same, then if the flow was not recorded in the HBF cache, then it is stored with "Hit" value of one (1). To store this new flow in the cache, the last item in the cache (which is the least recently seen flow) is replaced by this new flow with a probability of 0.05, and moved to the top of the cache. This replacement will occur only when the estimated buffer occupancy is greater than the fair share. If the flow was already recorded then "Hit" value is incremented. The flow id moves to the top of the cache. After m samplings, the "Hit Fraction" for flow i is calculated as $$H_i = (1-\alpha)H_i + \alpha\hat{H}_i$$ where H_i (on the RHS) $= \frac{Hit_i}{m}$, $\hat{H}_i$ is the Hit Fraction over the previous sampling interval and $\alpha < 1$ is a constant. It is this "Hit Fraction" that gives an estimate of the buffer occupancy of a flow. The dropping probability for flow i for the next sampling period is updated on every sample of that sampling period as $$\hat{p}_i = \frac{\bar{H}_i - \frac{1}{N_{act}}}{\left(\frac{1}{N_{act}}\right)}$$ where $$\bar{H}_i = \frac{Hit_i + H_i m}{m' + m}$$ where Hit_i is the number of Hits at the sampling time, H_i, the current Hit Fraction as calculated from before, m', the number of samples taken so far in the new sampling interval, and N_{act}, the estimate of the number of active flows. The inverse of N_{act} is the approximate fair share of the bandwidth. This dropping probability is further scaled according to RED's congestion avoidance state as $$p_{final,i} = \hat{p}_i \times \frac{\bar{q} - \min_{th}}{\max_{th} - \min_{th}}$$ Those flows that do not make the HBF cache memory are governed entirely by RED. High bandwidth flows will more likely remain in the HBF cache and will be penalized in proportion to how much greater than the fair share of bandwidth they use.	To estimate N_{act}, the original BLACK algorithm assumed that the traffic intensity of all the flows were roughly the same so that N_{act} is equal to m divided by the number of match events, however, this assumption is far from reality and in an enhanced version proposed in [66], N_{act} is estimated by the Direct Bitmap method. (The authors suggest that for higher accuracy and less memory requirements, the Multiresolution Bitmap, Triggered Bitmap, or Adaptive Bitmap be used.) N_{act} is determined by $b\ln(b/z)$ where z is the number of zero bits in the hash table during an interval. They used a $b = 100$bits.

February 2007.

[88] K. Kim, "Design of feedback controls supporting TCP based on the state-space approach," *Automatic Control, IEEE Transactions on*, vol. 51, no. 7, pp. 1086–1099, July 2006.

[89] J. Sun, G. Chen, K. Ko, S. Chan, and M. Zukerman, "PD-controller: a new active queue management scheme," in *Proceedings of the IEEE Global Telecommunications Conference (GLOBECOM '03)*, vol. 6, December 2003, pp. 3103–3107.

[90] S. Ryu, C. Rump, and C. Qiao, "A predictive and robust active queue management for Internet congestion control," in *Proceedings of the Eighth IEEE International Symposium on Computers and Communication (ISCC 2003)*, vol. 2, June 2003, pp. 991–998.

[91] S. Xiang, B. Xu, S. Wu, and D. Peng, "Gain adaptive smith predictor for congestion control in robust active queue management," in *Proceedings of the Sixth World Congress on Intelligent Control and Automation (WCICA 2006)*, vol. 1, 2006, pp. 4489–4493.

[92] L. He, H. Zhu, Y. Jing, and F. Gao, "Application of IMC-Smith controller in the large-delay network congestion control," in *Proceedings of the Sixth World Congress on Intelligent Control and Automation (WCICA 2006)*, vol. 1, 2006, pp. 4595–4599.

[93] H. Wang, Z. Tian, and C. Qiu, "RSP: Robust Smith predictor for queue management in time-delay networks," in *Proceedings of the 2010 IEEE International Conference on Wireless Communications, Networking and Information Security (WCNIS)*, June 2010, pp. 623–627.

[94] B. Meng, "Robust controller design for active queue management system," in *Proceedings of the 2010 IEEE Fifth International Conference on Bio-Inspired Computing: Theories and Applications (BIC-TA)*, September 2010, pp. 1037–1040.

[95] Q. Chen and O. W. W. Yang, "Robust controller design for AQM router," *Automatic Control, IEEE Transactions on*, vol. 52, no. 5, pp. 938–943, May 2007.

[96] M. Firuzi and M. Haeri, "Adaptive generalized predictive control of active queue management in TCP networks," in *Proceedings of the International Conference on Computer as a Tool (EUROCON 2005)*, vol. 1, November 2005, pp. 676–679.

[97] P. Padhy and R. Sundaram, "Analysis and design of improved PI-PD controller for TCP AQM routers," in *Proceedings of the 2010 International Conference on Power, Control and Embedded Systems (ICPCES)*, December 2010, pp. 1–5.

[98] P. Singh and S. Gupta, "Variable length virtual output queue based fuzzy congestion control at routers," in *Proceedings of the 2011 IEEE 3rd International Conference on Communication Software and Networks (ICCSN)*, May 2011, pp. 29–33.

[99] C. Wang, B. Li, K. Sohraby, and Y. Peng, "AFRED: an adaptive fuzzy-based control algorithm for active queue management," in *Proceedings of the 28th Annual IEEE International Conference on Local Computer Networks (LCN '03)*, October 2003, pp. 12–20.

[100] S. Ghosh, Q. Razouqi, H. Schumacher, and A. Celmins, "A survey of recent advances in fuzzy logic in telecommunications networks and new challenges," *Fuzzy Systems, IEEE Transactions on*, vol. 6, no. 3,

TABLE XXIII
SUMMARY OF HEURISTIC AQM SCHEMES FOR IMPROVED FAIRNESS (CONT'D)

AQM	Congestion Indicator	Control Function	Special Characteristics
LUBA [27]		It uses link utilization and arrival rates to penalize misbehaving flows. The overload factor, U, is calculated as $U = \frac{\lambda}{\mu}$ where λ is the aggregate arrival rate at the router and μ is the outgoing link capacity. λ is calculated as $\frac{B}{\tau}$ where B is the number of bytes of packets arriving at the queue and τ is the measurement interval called the *lubaInterval*. There is a target utilization, $\hat{U} < 1$ from which the fair share is determined as $FS = \frac{\tau\mu\hat{U}}{n}$, where n is the number of active flows in the *lubaInterval*. Now, if $U < \hat{U}$, no packets are dropped at all. LUBA maintains a history table fo flow IDs. Associated with each entry are the variables: *count* (which is the number of consecutive *lubaInterval*s the flow exceeds its fair share), *mar* (which is the mean arrival rate or the number of bytes over the *count lubaInterval*s), *last bytes* (which is the number of bytes in the current *lubaInterval*) and *dropRatio* (which is the packet drop probability in the current *lubaInterval*). There is a lower bound on this *dropRatio*, p_a, assigned to all flows that are determined to be well-behaved. The *countThreshold*, T — a design parameter — is used to decide when a flow is well-behaved and when it is not. A flow is purged from the history table if its *mar* $<$ *FS*. However if its *mar* $\geq$ *FS* and *count* $> T$, its *dropRatio* is updated to $1 - \frac{FS}{mar}$. This update only occurs when $U > \hat{U}$. However, if the opposite is true (i.e. $U < \hat{U}$) then the history list is completely purged and re-initialized. To avoid punishing bursty flows, a $U < \frac{3}{2}$ is tolerated and the length of the *lubaInterval* is adjusted by $\tau = \frac{cL}{\lambda - \mu}$ where c is a constant.	

pp. 443–447, August 1998.

[101] J. Sun, M. Zukerman, and M. Palaniswami, "Stabilizing red using a fuzzy controller," in *Proceedings of the IEEE International Conference on Communications (ICC '07)*, June 2007, pp. 266–271.

[102] H. Abdel-jaber, M. Mahafzah, F. Thabtah, and M. Woodward, "Fuzzy logic controller of Random Early Detection based on average queue length and packet loss rate," in *Proceedings of the International Symposium on Performance Evaluation of Computer and Telecommunication Systems (SPECTS 2008)*, June 2008, pp. 428–432.

[103] S. Zargar, M. Yaghmaee, and A. Fard, "Fuzzy proactive queue management technique," in *Proceedings of the 2006 Annual IEEE India Conference*, September 2006, pp. 1–6.

[104] M. Moghaddam, "A fuzzy active queue management mechanism for Internet congestion control," in *Proceedings of the 2010 Third International Workshop on Advanced Computational Intelligence (IWACI)*, August 2010, pp. 203–208.

[105] X. Changbiao and L. Fengfeng, "A congestion control algorithm of fuzzy control in routers," in *Proceedings of the Fourth International Conference on Wireless Communications, Networking and Mobile Computing (WiCOM '08)*, October 2008, pp. 1–4.

[106] Y. Xian, L. Wang, and Y. Wen, "An adaptive target queue length FREM algorithm," in *Proceedings of the IEEE International Conference on Communications Technology and Applications (ICCTA '09)*, October 2009, pp. 845–850.

[107] S. Masoumzadeh, G. Taghizadeh, K. Meshgi, and S. Shiry, "Deep blue: A fuzzy Q-learning enhanced active queue management scheme," in *Proceedings of the International Conference on Adaptive and Intelligent Systems (ICAIS '09)*, September 2009, pp. 43–48.

[108] C. Nyirenda and D. Dawoud, "Multi-objective particle swarm optimization for fuzzy logic based active queue management," in *Proceedings of the 2006 IEEE International Conference on Fuzzy Systems*, 2006, pp. 2231–2238.

[109] S. Mohammadi, H. Pour, M. Jafari, and A. Javadi, "Fuzzy-based PID active queue manager for TCP/IP networks," in *Proceedings of the 2010 10th International Conference on Information Sciences Signal Processing and their Applications (ISSPA)*, May 2010, pp. 434–439.

[110] Z. Chuan and L. Xuejiao, "A robust AQM algorithm based on fuzzy-inference," in *Proceedings of the International Conference on Measuring Technology and Mechatronics Automation (ICMTMA '09)*, vol. 2, April 2009, pp. 534–537.

[111] P. Singh and S. Gupta, "Variable length virtual output queue based fuzzy congestion control at routers," in *Proceedings of the Third IEEE International Conference on Communication Software and Networks (ICCSN)*, May 2011, pp. 29–33.

[112] Y. Qiao and L. Qiongyu, "A new active queue management algorithm based on self-adaptive fuzzy neural-network PID controller," in *Proceedings of the 2011 International Conference on Internet Technology and Applications (iTAP)*, August 2011, pp. 1–4.

[113] Y. Qiao and H. Xiaojuan, "A new PID controller for AQM based on neural network," in *Proceedings of the 2010 IEEE International Conference on Intelligent Computing and Intelligent Systems (ICIS)*, vol. 1, October 2010, pp. 804–808.

[114] F. Yanfei, R. Fengyuan, and L. Chuang, "Design of an active queue management algorithm based fuzzy logic decision," in *Proceedings of the International Conference on Communication Technology (ICCT 2003)*, vol. 1, April 2003, pp. 286–289.

[115] Y. Aoul, A. Nafaa, D. Negru, and A. Mehaoua, "FAFC: fast adaptive fuzzy AQM controller for TCP/IP networks," in *Proceedings of the IEEE Global Telecommunications Conference (GLOBECOM '04)*, vol. 3, November 2004, pp. 1319–1323.

[116] Y. Jing, Z. Chen, and G. Dimirovski, "Robust fuzzy observer-based control for TCP/AQM network systems with state delay," in *Proceedings of the American Control Conference (ACC)*, July 2010, pp. 1350–1355.

[117] G. Di Fatta, F. Hoffmann, G. Lo Re, and A. Urso, "A genetic algorithm for the design of a fuzzy controller for active queue management," *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, vol. 33, no. 3, pp. 313–324, August 2003.

[118] M. Yaghmaee and H. Farmad, "Improving the loss performance of random early detection gateway using fuzzy logic control," in *Proceedings of the Ninth International Symposium on Computers And Communications (ISCC 2004)*, vol. 2, Alexandria, Egypt, 2004, pp. 927–32.

[119] C. Luo and C. Ran, "An adaptive retransmission and active drop mechanism based on fuzzy logic," in *Proceedings of the 2004 Asia-Pacific Radio Science Conference*, August 2004, pp. 162–165.

[120] J. Kim, J. Park, and Y. Choi, "Adaptive wavelet neural network controller for AQM router in TCP network," in *Proceedings of the Fifth Mexican International Conference on Artificial Intelligence (MICAI '06)*, November 2006, pp. 388–397.

[121] Y. Gao and J. Hou, "A state feedback control approach to stabilizing queues for ECN-enabled TCP connections," in *Proceedings of the Twenty-second Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2003)*, vol. 3, 2003, pp. 2301–2311.

[122] M. Ardestani and M. Beheshti, "A robust discrete-time controller for delay sensitive applications," in *Proceedings of the Seventh International Conference on Information, Communications and Signal Processing (ICICS 2009)*, December 2009, pp. 1–6.

[123] A. Abharian and M. Alireza, "Hybrid GA-BF based intelligent PID active queue management control design for TCP network," in *Pro-*

TABLE XXIV
SUMMARY OF HEURISTIC AQM SCHEMES FOR DIFFERENTIATED SERVICES

AQM	Congestion Indicator	Control Function	Special Characteristics
RIO [194]	Average (EWMA) queue length	It can be used within the DiffServ framework and more specifically for the AF PHB (Assured Forwarding PHB). At the network's edge, packets are tagged as "in" or "out" (according to their service level agreement (SLA)) and therefore will be treated differently in the network especially when it approaches congestion [194], [75]. RIO is based on RED and essentially employs two parallel instances of RED, one for the "in" packets and one for the "out" packets. However, this implies two sets of parameters that must be tuned, exacerbating the parameter-tuning problem of RED [39]. For the "in" packets the parameters are $\bar{q}_{in}$, $\max_{th\_in}$, $\min_{th\_in}$ and $P_{\max\_in}$. For the "out" packets the parameters are $\max_{th\_out}$, $\min_{th\_out}$ and $P_{\max\_out}$. Upon the arrival of an "in" packet, the average queue occupancy for "in" packets ($\bar{q}_{in}$) is updated. The average queue occupancy for all packets, $\bar{q}_{total}$, is updated when either "in" or "out" packets arrive.	To ensure proper differentiation between the higher priority "in" packets and the "out" packets, $\min_{th\_out} < \max_{th\_in}$, $P_{\max\_out} > P_{\max\_in}$ and $\max_{th\_out} < \max_{th\_in}$. This pushes the "out" packets into congestion avoidance/control phases faster than the "in" packets. Thus RIO drops "out" packets more quickly. Also $\bar{q}_{total}$ is used as the congestion indicator for "out" packets whereas $\bar{q}_{in}$ for the "in" packets. This is because it is for the "in" packets the network was provisioned (and not for the "out"), and it was desired that RIO perform well regardless of the traffic mix (of "in" and "out" packets).
RIO-Coupled (RIO-C)	Average (EWMA) queue length	The original RIO algorithm had been extended to support n priority classes instead of two (2). When $n = 3$ packets are labelled red (lowest priority, highest drop precedence), yellow, green (highest priority, lowest drop precedence). Again, there will be n sets of RED parameters for differentiation among the classes. The average queue length for a given class j will depend on the occupancy of all the packets belonging to its class and all the classes of higher priority, so that the lowest priority class will have its average queue length based on the total queue length (all packet clases). What has just been described is more specifically RIO-C (i.e., RIO-Coupled).	
RIO-Decoupled (RIO-DC) & WRED	Average (EWMA) queue length	For RIO-Decoupled (RIO-DC), average queue length for a given class j depends only on the queue occupancy of packets belonging to class j [75]. Another variant, Weighted RED (WRED), the average queue length is based on the total queue length which is used for all classes [75], [62]. In all cases, the EWMA queue averaging weight, w_q, is the same for all classes.	Again, RED's parameter tuning problem is multiplied some n-times over.

ceedings of the 2011 3rd International Conference onElectronics Computer Technology (ICECT), vol. 4, April 2011, pp. 227–232.

[124] L. Massoulie and J. Roberts, "Bandwidth sharing: objectives and algorithms," *IEEE/ACM Transactions on Networking*, vol. 10, no. 3, pp. 320–8, 2002.

[125] S. Low, "A duality model of TCP and queue management algorithms," *IEEE/ACM Transactions on Networking*, vol. 11, no. 4, pp. 525–36, 2003.

[126] F. Kelly, "Charging and rate control for elastic traffic," *European Transactions on Telecommunications*, vol. 8, pp. 33–37, 1997. [Online]. Available: citeseer.ist.psu.edu/kelly97charging.html

[127] S. Kunniyur and R. Srikant, "Stable, scalable, fair congestion control and AQM schemes that achieve high utilization in the Internet," *IEEE Transactions on Automatic Control*, vol. 48, no. 11, pp. 2024–8, 2003.

[128] ——, "End-to-end congestion control schemes: utility functions, random losses and ECN marks," *IEEE/ACM Transactions on Networking*, vol. 11, no. 5, pp. 689–702, 2003.

[129] V. Reguera, F. Alvarez Paliza, E. Garcia Fernandez, and W. Godoy, "Voice over IP quality of service using active queue management," in *Proceedings of the 2006 International Telecommunications Symposium*, September 2006, pp. 835–840.

[130] Q. Yang and J. Pitts, "Scalable voice over Internet protocol service-level agreement guarantees in converged Transmission Control Protocol/ Internet Protocol networks," *Communications, IET*, vol. 4, no. 8, pp. 1026–1035, 2010.

[131] Y. Xiaogang, L. Jiqiang, and L. Ning, "Congestion control based on priority drop for H.264/SVC," in *Proceedings of the International Conference on Multimedia and Ubiquitous Engineering (MUE '07)*, April 2007, pp. 585–589.

[132] S.-R. Kang, Y. Zhang, M. Dai, and D. Loguinov, "Multi-layer active queue management and congestion control for scalable video streaming," in *Proceedings of the 24th International Conference on Distributed Computing Systems*, 2004, pp. 768–777.

[133] Y. Li, X. Gong, W. Wang, X. Que, and J. Ma, "An autonomic active queue management mechanism to improve multimedia flow delivery quality," in *Proceedings of the 2010 International Conference on Communications and Mobile Computing (CMC)*, vol. 1, April 2010, pp. 493–497.

[134] V. Robles, M. Siller, and J. Woods, "Active discarding packet mechanisms for video transmission," in *Proceedings of the IEEE International Conference on System of Systems Engineering (SoSE '07)*, April 2007, pp. 1–5.

[135] M. Usha and R. S. D. Wahida Banu, "Pushout policy in active queue management to support quality of service guarantees in IP routers," in *Proceedings of the 10th IEEE Singapore International Conference on Communication systems (ICCS 2006)*, October 2006, pp. 1–5.

[136] C. Chrysostomou, A. Pitsillides, G. Hadjipollas, A. Sekercioglu, and M. Polycarpou, "Fuzzy explicit marking for congestion control in differentiated services networks," in *Proceedings of the Eighth IEEE International Symposium on Computers and Communication (ISCC 2003)*, vol. 1, June 2003, pp. 312–319.

[137] C. Chrysostomou, A. Pitsillides, G. Hadjipollas, M. Polycarpou, and A. Sekercioglu, "Congestion control in differentiated services networks using fuzzy logic," in *Proceedings of the 43rd IEEE Conference on Decision and Control (CDC)*, vol. 1, December 2004, pp. 549–556.

[138] M. Jiang, J. Wu, and C. Wu, "MBIO: An new active queue management algorithm for DiffServ network," in *Proceedings of the 2006 IEEE International Conference on Networking, Sensing and Control (ICNSC '06)*, 2006, pp. 803–806.

[139] E. Ng, K. Phang, T. Ling, and L. Por, "Fuzzy active queue management for assured forwarding traffic in differentiated services network," in *Proceedings of the International Conference on Computing Informatics (ICOCI '06)*, June 2006, pp. 1–5.

[140] L. Zhu, N. Ansari, G. Cheng, and K. Xu, "Edge-based active queue management," *Communications, IEE Proceedings-*, vol. 153, no. 1, pp. 55–60, February 2006.

[141] C. Joo, J. Hong, and S. Bahk, "Assuring drop probability for delay-insensitive traffic in a differentiated service network," in *Proceedings*

TABLE XXV
SUMMARY OF HEURISTIC AQM SCHEMES FOR DIFFERENTIATED SERVICES (CONT'D)

AQM	Congestion Indicator	Control Function	Special Characteristics
Adaptive RIO [39], [75]	Average (EWMA) queue length	Adaptive RIO (ARIO) extends RIO by using Adaptive RED (ARED) instead of RED. ARIO therefore is a priority-class based AQM like RIO which provide mechanisms for differentiation among packet classes flowing through the network. One of the banes of RIO was the problematic parameter tuning inherited from RED, which was magnified by the multiple number of classes to be handled in RIO. ARED was developed to significantly alleviate this issue in RED by having automatic adaptation of all RED's parameters save one. ARIO uses a full ARED for each class. This ARED is itself based on Gentle RED (GRED). ARIO, like ARED, attempts to maintain the queue occupancy near a target level. In fact, it attempts to keep the average queue size within the interval $(\bar{q}_{low}, \bar{q}_{high})$ where $\bar{q}_{low} = \min_{th} + 0.4(\max_{th} - \min_{th})$ and $\bar{q}_{high} = \min_{th} + 0.6(\max_{th} - \min_{th})$. It automatically translates the QoS parameter of delay into the algorithm's parameters. Let $\bar{q}_j$ and $P_{\max\_j}$ be the average queue occupancy and maximum drop probability of priority class j respectively. With the required delay, d_t, as the in input, the automatic parameter tuning occurs as follows: $$\min_{th} = \max\left(5, d_t \frac{C}{2}\right)$$ $$\max_{th} = 3 \min_{th}$$ $$w_q = 1 - e^{-\frac{1}{C}}$$ where $\max_{th}$ and $\min_{th}$ are the same for all classes. If $\bar{q}_j > \bar{q}_{high}$ and $P_{\max\_j} < 0.5$, $P_{\max\_j}$ is increased additively as $P_{\max\_j} = P_{\max\_j} + \alpha$ where $\alpha = \min\left(0.01, \frac{P_{\max\_j}}{4}\right)$. But if the packet belongs to the highest drop precedence (i.e. lowest priority), then $P_{\max\_j}$ is taken as the minimum of the two lower drop precedences. Now, if $\bar{q}_j < \bar{q}_{low}$ and $P_{\max\_j} > 0.01$, $P_{\max\_j}$ is decreased multiplicatively as $P_{\max\_j} = P_{\max\_j} \times \beta$ where $\beta = 0.9$ is the decrease constant. It can be seen that $P_{\max\_j}$ is bounded between 0.01 and 0.5 for all classes.	The average queue occupancy for each class is calculated individually according to the EWMA method as in RED. It is updated on each packet arrival for that class. However, $P_{\max\_j}$ is updated after a fixed time interval (0.5 second). With all the parameters updated, the decision process for packet discard for each class is as in RED with their average queue occupancies compared to the queue thresholds. The version of RIO here, as one can deduce, is RIO-DC instead of RIO-C. The reason for his design choice according to [75] is to maintain an average queue target for all traffic under all conditions. If RIO-C was used there would be different average queue sizes for different traffic mix (of classes) which would be more difficult to control.
Rate-based RIO [47]	current traffic load which is the ratio of the estimated arrival rate to the service rate	Rb-RIO supports three priority classes. A weighted average arrival rate is maintained for each class (*EAR*$_j$ where $j = 1, 2, 4$ in ascending order of priority). If the sum of the estimated arrival rate for all three classes (*TEAR*) is less than the service rate then all packets are accepted into the queue. But if not, and the sum of the estimated arrival rates for the two higher priority classes is less than the service rate (*SR*), accept all packets belonging to those classes, but drop packets belonging to the lowest class with probability = $\frac{TEAR-SR}{EAR_1}$. If however, the sum of the estimated arrival rates for the two higher priority classes is greater than the service rate but the estimated arrival rate of the highest priority class is less than the service rate, all packets for the highest priority class are accepted, the packets of the second-highest priority class are dropped with probability = $\frac{EAR_2+EAR_3-SR}{EAR_2}$, but all packets of the lowest priority are dropped. Finally, if the estimated arrival rate of the highest priority class is greater than the service rate, all packets of the lower priority classes are dropped with probability one, but the packets of the highest priority are dropped with probability = $\frac{EAR_2-SR}{EAR_3}$.	

of the 2005 Second IEEE Consumer Communications and Networking Conference (CCNC)*, January 2005, pp. 515–520.

[142] T. Minagawa and T. Ikegami, "Controlling user flows with RIO and WFQ," in *Proceedings of the 2010 International Symposium on Communications and Information Technologies (ISCIT)*, October 2010, pp. 87–92.

[143] X. Chang and J. Muppala, "The effects of AQM on the performance of assured forwarding services," in *Proceedings of the Performance,24th IEEE International Computing, and Communications Conference (IPCCC 2005)*, April 2005, pp. 321–328.

[144] D. Agrawal, N. da Fonseca, and F. Granelli, "Integrated ARM/AQM mechanisms based on PID controllers," in *Proceedings of the 2005 IEEE International Conference on Communications (ICC 2005)*, vol. 1, May 2005, pp. 6–10.

[145] Y. Chait, C. Hollot, V. Misra, D. Towsley, H. Zhang, and J. Lui, "Providing throughput differentiation for TCP flows using adaptive two-color marking and two-level AQM," in *Proceedings of the Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2002)*, vol. 2, 2002, pp. 837–844.

[146] Y. Chait, C. Hollot, V. Misra, D. Towsley, H. Zhang, and Y. Cui, "Throughput differentiation using coloring at the network edge and preferential marking at the core," *Networking, IEEE/ACM Transactions on*, vol. 13, no. 4, pp. 743–754, August 2005.

[147] K. Nahm, Q. Li, and C.-C. Kuo, "Layered video multicast with ECN over differentiated service networks," in *Proceedings of the 2002 IEEE International Conference on Multimedia and Expo (ICME '02)*, vol. 1, 2002, pp. 381–384.

[148] G. Min and X. Jin, "Performance modelling of Random Early Detection based congestion control for multi-class self-similar network traffic," in *Proceedings of the IEEE International Conference on Communications (ICC '08)*, May 2008, pp. 5564–5568.

[149] Y. Huang, s. Mao, and S. Midkiff, "A control-theoretic approach to rate control for streaming videos," *Multimedia, IEEE Transactions on*, vol. 11, no. 6, pp. 1072–1081, October 2009.

[150] M. Farrokhian and M. Haeri, "AQM for dynamic QoS adaptation in DiffServ networks based on STAC," in *Proceedings of the International*

TABLE XXVI
SUMMARY OF HEURISTIC AQM SCHEMES FOR DIFFERENTIATED SERVICES (CONT'D)

AQM	**Congestion Indicator**	**Control Function**	**Special Characteristics**
Rate-based n-RED [74]	arrival-rate and average queue length	Let the highest priority class be denoted by $j = 3$, the second-highest, $j = 2$ and the lowest, $j = 1$. Let EAR_j be the estimated arrival rate for class j and *TEAR*, the total estimated arrival rate where $\mathit{TEAR} = \sum_{j=1}^{3} \mathit{EAR}_j$. Let *AT* be the accepted traffic. If $\mathit{TEAR} < \mathit{AT}$ all packets from all classes are accepted. If $\mathit{TEAR} > \mathit{AT}$ but $\sum_{j=2}^{3} \mathit{EAR}_j < \mathit{AT}$, all class 2 and 3 packets are accepted but class 1 packets are dropped with probability $\frac{\mathit{CP} \times \mathit{TEAR}}{\mathit{EAR}_1}$. If, however, $\sum_{j=2}^{3} \mathit{EAR}_j > \mathit{AT}$ but $\mathit{EAR}_3 < \mathit{AT}$, then all class 3 packets are accepted, class 1 packets are dropped with probability one and class 2 packets with probability $\frac{(\mathit{CP} \times \mathit{TEAR}) - \mathit{EAR}_1}{\mathit{EAR}_2}$. But if $\mathit{EAR}_3 > \mathit{AT}$, then all class 1 and 2 packets are dropped with probability 1 and class 3 packets are dropped with probability $\frac{(\mathit{CP} \times \mathit{TEAR}) - \mathit{EAR}_1 - \mathit{EAR}_2}{\mathit{EAR}_3}$. Now $\mathit{CP} = \mathit{CF} \times P_{drop}$ where $P_{drop} = \max\left(0, \frac{\mathit{TEAR} - R}{\mathit{TEAR}}\right)$ where R is the service rate and *CF* is a correction factor. $\mathit{CF} = \frac{\bar{q}}{B}\mathit{AP}$, where $\bar{q}$ is the average queue occupancy, B is the queue size and *AP* is the aggressiveness factor. In all, $\mathit{CP} = \frac{\bar{q}}{B}\mathit{AP} \max\left(0, \frac{\mathit{TEAR} - R}{\mathit{TEAR}}\right)$. $\mathit{EAR}_j = \min\left(\mathit{EAR}_j^{arr}, \mathit{EAR}^{upper}\right)$, where $\mathit{EAR}_j^{arr} = (1 - w)\frac{L}{T} + w\mathit{EAR}_j^{arr}$, $\mathit{EAR}^{upper} = (1 - w)\frac{L_{\max}}{T} + w\mathit{EAR}_j^{arr}$ and $R = (1 - w)\frac{L}{T} + wR$, where $w = e^{-\frac{T}{K}}$, and where T is the difference between the current time and the time of the last update. L is the packet size ($L_{\max}$ is the maximum packet size) and K is a constant. A too small value for K makes w small, hence it tracks the instantaneous arrival rate more closely. If, on the other hand, K is too large, the *EAR* will not be able to catch bursts. EAR^{upper} is calculated across all packets regardless of class and is employed so as to prevent a constant value of EAR_j when no packets for class j arrives for a good while.	
Dynamic Class-Based Threshold (D-CBT) [22]		There are three classes of traffic in CBT: tagged (flow-controlled multimedia) UDP2, untagged (other) UDP and TCP. CBT attempts to protect TCP flows from UDP flows and UDP multimedia flows from other UDP flows by applying threshold tests for the two UDP classes. Associated with each of these UDP classes is a static threshold. If the weighted-average queue occupancy of either of these two classes exceed their respective thresholds, their incoming packets will be dropped before being sent to the RED algortihm of the queue. TCP packets do not go through a threshold test but are directly controlled by the RED algorithm. This is more specifically called "CBT with RED for all". There is a different version called "CBT with RED for TCP" for which only the TCP packets go througth the RED algorithm and the average queue size is calculated only on the TCP packet occupancy. The reasoning is that subjecting UDP flows, which are mostly unresponsive, to early congestion notifications is useless. D-CBT builds on "CBT with RED for all". Instead of having static thresholds, D-CBT adapts the thresholds as the traffic mix (UDP, TCP) changes, so that there can be a more fair bandwidth allocation. The threshold tests are activated only when the average queue length (calculated by RED) is greater than the RED $\min_{th}$ parameter value. All queue averages are computed using the same weight upon each packet arrival. More specifically	D-CBT will therefore need to keep some state information, i.e. the number of active flows for each class. D-CBT defines "active" differently from FRED in that a flow is active if is has a packet in the outbound queue within a predefined interval since last checked. D-CBT authors say that an exact flow count is not necessary for D-CBT to work although higher accuracy is preferred.

SICE-ICASE Joint Conference, October 2006, pp. 3223–3227.

[151] Y. Xiao, H. Du, Z. Cao, and M. Lee, "Active queue management for differentiated service network," in *Proceedings of the 2006 IET International Conference on Wireless, Mobile and Multimedia Networks*, November 2006, pp. 1–5.

[152] B. Ng, D. Chieng, and A. Malik, "POWer Adaptive Random Early Detection for Diff-Serv Assured Forwarding service classes," in *Proceedings of the 2006 2nd IEEE/IFIP International Conference in Central Asia on Internet*, September 2006, pp. 1–5.

[153] S. Tartarelli and A. Banchs, "Random Early Marking: improving TCP performance in DiffServ Assured Forwarding," in *Proceedings of the IEEE International Conference on Communications (ICC 2002)*, vol. 2, 2002, pp. 970–975.

[154] H. Ma, W. Yuan, and S. Qin, "Realizing a loss proportional differentiation Assured Forwarding service through active queue management," in *Proceedings of the 2010 2nd International Conference on Future Computer and Communication (ICFCC)*, vol. 1, May 2010, pp. 62–66.

[155] Z. Cao and Y. Xiao, "A new DiffServ supported AQM algorithm," in *Proceedings of the 2006 8th International Conference on Signal Processing*, vol. 3, November 2006.

[156] Y. Xiaoping, C. Hong, and Z. Zhenyu, "A queue management algorithm for differentiated services," in *Proceedings of the 2011 International Conference on Intelligent Computation Technology and Automation (ICICTA)*, vol. 2, March 2011, pp. 941–944.

[157] S. Wen, Y. Fang, and H. Sun, "Differentiated bandwidth allocation with TCP protection in core routers," *Parallel and Distributed Systems, IEEE Transactions on*, vol. 20, no. 1, pp. 34–47, January 2009.

[158] Y. Xue, H. Nguyen, and K. Nahrstedt, "CA-AQM: Channel-aware active queue management for wireless networks," in *Proceedings of the IEEE International Conference on Communications (ICC '07)*, June 2007, pp. 4773–4778.

[159] J. Wang, M. Song, and H. Yang, "Rate-based active queue management for congestion control over wired and wireless links," in *Proceedings of the First International Conference on Communications and Networking in China (ChinaCom '06)*, October 2006, pp. 1–6.

[160] H. Wang, Y. Zhang, and C. Wang, "A wireless network congestion control algorithm based on adaptive QoS and wireless bandwidth,"

TABLE XXVII
SUMMARY OF HEURISTIC AQM SCHEMES FOR DIFFERENTIATED SERVICES (CONT'D)

AQM	Congestion Indicator	Control Function	Special Characteristics
D-CBT (cont'd)		Tagged UDP threshold $= \left(\frac{\text{number of tagged active flows}}{\text{total number of active flows}}\right) \times \bar{q}_{red}$ Untagged UDP threshold $= \left(\frac{\text{number of tagged active flows}}{\text{total number of active flows}}\right) \times (\min_{th} + 0.1(\max_{th} - \min_{th}))$	
Selective Fair Early Detection [70]	rate-based	This algorithm attempts to provide fair bandwidth allocation and can be easily adjusted to assign priorities to flow aggregates as in a differentiated services framework. For each active flow or flow aggregate SFED maintains a token bucket. The rate at which tokens fill the token bucket is proportional to the allocated bandwidths. The rate at which tokens are removed from the token bucket is equal to the incoming rate of the flow. The occupancy of the bucket (which quantifies this rate mismatch) is then used to determine the probability of dropping/marking a packet. The height of the each token bucket represents the burst length it can accommodate, therefore bursty flows are not unduly punished. A flow is active if its bucket is not full. The bucket for the inactive flows is removed and the tokens are redistributed fairly (e.g. round robin) across all the other token buckets (i.e. the number of tokens is conserved). Also, the token addition rate is increased fairly across all the token buckets. In a priority-less system, all the heights of the buckets are the same. The sum of the heights is proportional to the buffer size and is conserved when a flow is added or deleted. When there is a new flow, all the bucket heights are reduced, to accommodate the new token bucket and when the flow becomes inactive, the height of all the other token buckets are increased. The probability dropping function is similar to Gentle RED. For the ith token bucket, its bucket occupancy is denoted be x_i. Also N is the number of flows (hence the number of buckets) in the system, and L_N is the maximum height of each bucket. $p(x_i) == \begin{cases} 0 & \lambda_1 < \frac{x_i}{L_N} < 1 \\ P_{\max}\left(\frac{\lambda_1 - \frac{x_i}{L_N}}{\lambda_1 - \lambda_2}\right) & \lambda_2 < \frac{x_i}{L_N} < \lambda_1 \\ P_{\max} + (1 - P_{\max})\left(\frac{\lambda_1 - \frac{x_i}{L_N}}{\lambda_2}\right) & 0 < \frac{x_i}{L_N} < \lambda_2 \end{cases}$ where λ_1 and λ_2 are occupancy ratio thresholds. Also associated with SFED is the parameter σ which accounts for the tokens during deletion and creation of token buckets. In terms of implementing priorities among flows so that bandwidths are allocated proportional to their weight, the heights and token addition rate into each bucket is scaled in proportion to the flow's weight.	SFED's complexity in terms of token distribution and bucket creation is $O(N)$. SFED's authors then proposed Randomized SFED (R-SFED) to reduce this complexity to $O(1)$. They claim that R-SFED performs almost as good as SFED.

in *Proceedings of the 2nd International Conference on Biomedical Engineering and Informatics (BMEI '09)*, October 2009, pp. 1–4.

[161] D. Li, J. Theunis, K. Sleurs, J. Potemans, E. Van Lil, and A. Van de Capelle, "Improving RED performance during handovers in wireless IP networks," in *Proceedings of the 4th International Symposium on Wireless Communication Systems (ISWCS 2007)*, October 2007, pp. 441–445.

[162] H. Mukaidani, L. Cai, and X. Shen, "Stable queue management for supporting TCP flows over wireless networks," in *Proceedings of the 2011 IEEE International Conference on Communications (ICC)*, June 2011, pp. 1–6.

[163] K. Chavan, R. Kumar, M. Belur, and A. Karandikar, "Robust active queue management for wireless networks," *Control Systems Technology, IEEE Transactions on*, vol. 19, no. 6, pp. 1630–1638, November 2011.

[164] L. Andrew, S. Hanly, and R. Mukhtar, "CLAMP: Active queue management at wireless access points," *Proceedings of the 11th European Wireless Conference 2005 - Next Generation Wireless and Mobile Communications and Services (European Wireless)*, pp. 1–7, April 2005.

[165] ——, "Active queue management for fair resource allocation in wireless networks," *Mobile Computing, IEEE Transactions on*, vol. 7, no. 2, pp. 231–246, February 2008.

[166] R. Alasem and H. Abu-Mansour, "EF-AQM: Efficient and fair bandwidth allocation AQM scheme for wireless networks," in *2010 Second International Conference on Computational Intelligence, Communication Systems and Networks (CICSyN)*, July 2010, pp. 169–172.

[167] S. Yi, M. Kappes, S. Garg, X. Deng, G. Kesidis, and C. Das, "Proxy-RED: an AQM scheme for wireless local area networks," in *Proceedings of the 13th International Conference on Computer Communications and Networks (ICCCN 2004)*, October 2004, pp. 460–465.

[168] X. Chang, X. Lin, and J. Muppala, "A control-theoretic approach to improving fairness in DCF based WLANs," in *Proceedings of the 25th IEEE International Performance, Computing, and Communications Conference (IPCCC 2006)*, April 2006, p. 86.

[169] J. Alcaraz and F. Cerdan, "Using buffer management in 3G radio bearers to enhance end-to-end TCP performance," in *Proceedings of the 20th International Conference on Advanced Information Networking and Applications (AINA 2006)*, vol. 2, April 2006.

[170] J. Lakkakorpi and R. Cuny, "Comparison of different active queue management mechanisms for 3G radio network controllers," in *Proceedings of the IEEE Wireless Communications and Networking Conference (WCNC 2006)*, vol. 1, April 2006, pp. 80–85.

[171] R. Cuny and J. Lakkakorpi, "Active queue management in EGPRS," in *Proceedings of the IEEE 63rd Vehicular Technology Conference (VTC*

2006), vol. 1, May 2006, pp. 373–377.

[172] J. Zhang and D. Pearce, "On designing a burst-sensitive RED queue at GPRS links in a heterogeneous mobile environment," in *Proceedings of the IEEE 16th International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC 2005)*, vol. 3, September 2005, pp. 1719–1723.

[173] O. Akin, S. Ergut, and R. Rao, "Client side active queue management for 3G cellular networks," in *Proceedings of the 3rd IEEE Consumer Communications and Networking Conference (CCNC 2006)*, vol. 2, January 2006, pp. 907–912.

[174] J. Lakkakorpi and A. Sayenko, "Backhaul as a bottleneck in IEEE 802.16e networks," in *Proceedings of the IEEE Global Telecommunications Conference (GLOBECOM 2008)*, December 2008, pp. 1–6.

[175] L. Jun, Y. Wu, F. Suili, G. Hua, and Z. Hongcheng, "A cross-layer queue management algorithm in 802.16 wireless networks," in *Proceedings of the International Conference on Communication Software and Networks (ICCSN '09)*, February 2009, pp. 25–29.

[176] J. Lakkakorpi, A. Sayenko, J. Karhula, O. Alanen, and J. Moilanen, "Active queue management for reducing downlink delays in WiMAX," in *Proceedings of the 2007 IEEE 66th Vehicular Technology Conference (VTC-2007)*, October 2007, pp. 326–330.

[177] S. Saleh and M. Fleury, "Video streaming with multi-TFRC and uplink queue management," in *2010 Digest of Technical Papers of the International Conference on Consumer Electronics (ICCE)*, January 2010, pp. 75–76.

[178] P. Kulkarni, M. Nazeeruddin, S. McClean, G. Parr, M. Black, B. Scotney, and P. Dini, "Deploying lightweight queue management for improving performance of Mobile Ad-hoc Networks (MANETs)," in *Proceedings of the International conference on Networking and Services (ICNS '06)*, July 2006, p. 102.

[179] K. Kumar, I. Ramya, and M. Masillamani, "Queue management in mobile adhoc networks (manets)," in *Proceedings of the 2010 IEEE/ACM International Conference on Green Computing and Communications (GreenCom), International Conference on Cyber, Physical and Social Computing (CPSCom)*, December 2010, pp. 943–946.

[180] L. Suo-ping and H. Zhi-peng, "A AQM model with multi-priority and multi-receiver for Mobile Ad Hoc Networks," in *Proceedings of the International Conference on Wireless Communications, Networking and Mobile Computing (WiCom 2007)*, September 2007, pp. 1457–1460.

[181] B. Abbasov, "AHRED: A robust AQM algorithm for wireless ad hoc networks," in *Proceedings of the International Conference on Application of Information and Communication Technologies (AICT 2009)*, October 2009, pp. 1–4.

[182] W. Lee, F. Liu, and H. Lo, "Improving the performance of MPEG-4 transmission in IEEE 802.15.3 WPAN," in *Proceedings of the 8th IEEE International Conference on Computer and Information Technology (CIT 2008)*, July 2008, pp. 676–681.

[183] D. Pacifico, M. Pacifico, C. Fischione, H. Hjalrmasson, and K. Johansson, "Improving TCP performance during the intra LTE handover," in *Proceedings of the IEEE Global Telecommunications Conference (GLOBECOM 2009)*, December 2009, pp. 1–8.

[184] S. Zhao, P. Wang, and J. He, "Simulation analysis of congestion control in WSN based on AQM," in *Proceedings of the 2011 International Conference on Mechatronic Science, Electric Engineering and Computer (MEC)*, August 2011, pp. 197–200.

[185] D. Katabi, M. Handley, and C. Rohrs, "Congestion control for high bandwidth-delay product networks," in *Proceedings of the 2002 conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*. New York, NY, USA: ACM, 2002, pp. 89–102. [Online]. Available: http://doi.acm.org/10.1145/633025.633035

[186] F. Abrantes and M. Ricardo, "XCP for shared-access multi-rate media," *SIGCOMM Comput. Commun. Rev.*, vol. 36, no. 3, pp. 27–38, July 2006. [Online]. Available: http://doi.acm.org/10.1145/1140086.1140091

[187] N. Dukkipati, N. McKeown, and A. Fraser, "RCP-AC: Congestion control to make flows complete quickly in any environment," in *Proceedings of the 25th IEEE International Conference on Computer Communications (INFOCOM 2006)*, April 2006, pp. 1–5.

[188] T. Ott, T. Lakshman, and L. Wong, "SRED: stabilized RED," in *Proceedings of the Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies on Computer Communications (IEEE INFOCOM '99)*, vol. 3, New York, NY, USA, 1999, pp. 1346–55.

[189] W.-C. Feng, D. Kandlur, D. Saha, and K. Shin, "A self-configuring RED gateway," in *Proceedings of the Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM '99)*, vol. 3, March 1999, pp. 1320–1328.

[190] V. Phirke, M. Claypool, and R. Kinicki, "RED-worcester - traffic sensitive active queue management," in *Proceedings of the 10th IEEE International Conference on Network Protocols*, Paris, France, 2002, pp. 194–5.

[191] W. Feng, A. Kapadia, and S. Thulasidasan, "GREEN: proactive queue management over a best-effort network," in *Proceedings of the IEEE Global Telecommunications Conference (GLOBECOM'02)*, vol. 2, Taipei, Taiwan, 2002, pp. 1774–8.

[192] M. Mathis, J. Semke, J. Mahdavi, and T. Ott, "The macroscopic behaviour of the TCP congestion avoidance algorithm," *Computer Communication Review*, vol. 27, no. 3, pp. 67–82, 1997.

[193] W. Feng, D. Kandlur, D. Saha, and K. Shin, "Stochastic fair blue: a queue management algorithm for enforcing fairness," in *Proceedings of the Twentieth Annual Joint Conference of the IEEE Computer and Communications Society on Computer Communications (IEEE INFOCOM 2001)*, vol. 3, Anchorage, AK, USA, 2001, pp. 1520–9.

[194] D. Clark and W. Fang, "Explicit allocation of best-effort packet delivery service," *IEEE/ACM Transactions on Networking*, vol. 6, no. 4, pp. 362–73, 1998.

Richelle Adams received the B.Sc. degree in Electrical and Computer Engineering in 1998 and the M.Sc. degree in Communication Systems in 2001 from the University of the West Indies, St. Augustine Campus, Trinidad and Tobago. In 2007, she received the Ph.D. degree in Electrical and Computer Engineering from the Georgia Institute of Technology, Atlanta, in 2007. Currently, she is a Lecturer in the Department of Electrical and Computer Engineering, The University of the West Indies. Her research interests include infinitesimal perturbation analysis for active queue management in communication networks, wireless networks and disaster communications. She also holds the IEEE Wireless Communication Engineering Technologies (WCET) certification.