FLoadAutoRED: An Active Queue Management Scheme to Prevent Congestion in a Dynamically Varying Traffic in IP Networks

CHAPTER 2

RELATED WORKS

2.1 ACTIVE QUEUE MANAGEMENT SCHEMES

Active Queue Management (AQM) was introduced in the last decade as a congestion preventive approach. The goal of AQM is to provide efficient resource allocation by reducing the average queue length and packet loss with increased link utilization. AQM drops packets before buffer becomes full. It uses queue length, input rate as congestion indicators to detect incipient congestion. The first AQM algorithm RED detects congestion by observing the queue state. In RED [16] packet drop probability is linearly proportional to queue length. The AQM algorithm RED drops packets before a queue becomes full. This reduces the number of packets dropped. As queue length does not act as a complete congestion indicator, the input rate is also included as a congestion indicator. Some of the AQM methods use both these congestion indicators to detect congestion at the earliest. AQMs try to project a better overall performance compared to the traditional schemes as they detect incipient congestion to prevent congestion occurring in future. Research is being done in AQMs to give the best performance in terms of the QoS requirements of the network.

An active queue management mechanism provides greater capacity to absorb packets in bursts without dropping packets. With an active queue management, the problem of global synchronization is avoided with improved link utilisaion and network throughput. The end source recovers with more difficulty from bursty packet drop than to a single packet drop. As bursty packet drop represents a possible waste of bandwidth on the way to the queue, AQM ensures that buffer is always available for the incoming packets and avoids lock-out behavior.

As in Figure 2.1**,** the AQM observes information such as the packet arrival rate X, the backlog in the queue B, link capacity C and determines the appropriate dropping or marking rate M. An AQM algorithm is to be designed to combine this state information to generate the appropriate feedback signal,

so that the source rates can be controlled quickly to achieve the desired utilisation of the link.

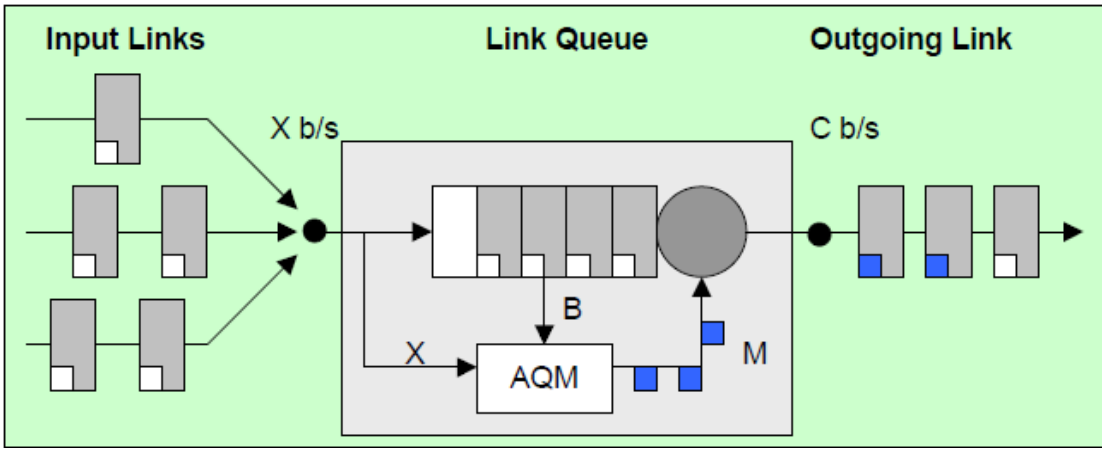

Figure 2.1 AQM and its Links

AQMs drop packets in the gateway to notify sources of congestion at the time of congestion. The feedback mechanism in TCP/IP networks for the RED AQM in gateways is to drop packets. As networks contain sources with a range of burstiness and varying traffic, active queue management schemes must face the problem of global synchronisation while handling this traffic. The more the bursty traffic from a particular source, the more likely it is that the gateway queue will overflow when packets from that source arrive at the gateway [18]. Hence another goal AQM should achieve is to avoid the global synchronization while deciding which sources to notify of congestion. Global synchronisation results from notifying all sources to reduce their windows at the same time. Global synchronization in networks [45] results in loss of throughput in the network. Synchronization is a general network phenomenon that has been explored in [19].

Active Queue Management schemes use distinct algorithms for congestion detection and for deciding which connections to notify of this congestion. The RED AQM uses randomization in choosing which arriving packets to drop. According to [16], the probability of dropping a packet from a particular connection is roughly proportional to that connection's share of the

bandwidth through the gateway. RED AQM method is efficiently implemented without maintaining per-connection state at the gateway. This method is used to control the average queue size even if most connections fail to reduce their throughput in response to marked or dropped packets. Moreover, active queue management schemes use various factors for incipient congestion detection. AQM schemes can be classified as discussed in the following section for easy study and understanding.

2.2 CLASSIFICATION OF AQM SCHEMES

Active Queue Management anticipates or detects congestion to occur in the link and prevents the occurrence of congestion. Since only the routers in IP networks has a unified view of the queuing behavior over time, the most effective detection of congestion can occur only in the router. In addition, a router is shared by many active connections with a wide range of delay, throughput requirements etc. Decisions about the duration and magnitude of transient congestion to be allowed at the router are best made by the router itself. So the AQM schemes implemented at the routers use the information or parameters available at the routers. These parameters are considered as metrics of the AQM schemes to detect congestion at routers.

The existing AQM schemes use various factors or metrics to detect congestion. These factors are used to estimate congestion in the queue and based on this various AQM algorithms are proposed in the past decade. The schemes are based on congestion metrics like Queue-length, Load, both Queue and Load, other parameters like Loss rate. Further some of these schemes also use flow information along with various congestion metrics to analyze and control the congestion in routers more accurately. Considering these factors AQM schemes can be categorized based on congestion metrics without flow information and with flow information as shown in Figure 2.2.

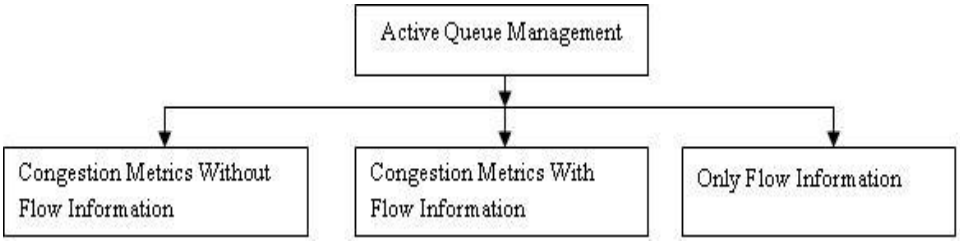

Figure 2.2 Classifications of AQM Schemes

2.2.1 Congestion metric without Flow Information

It is the first category of classification that considers only the congestion metric in the AQM schemes. However, based on the congestion metric, the AQMs can further be classified as queue based, load based, both queue and load based and others.

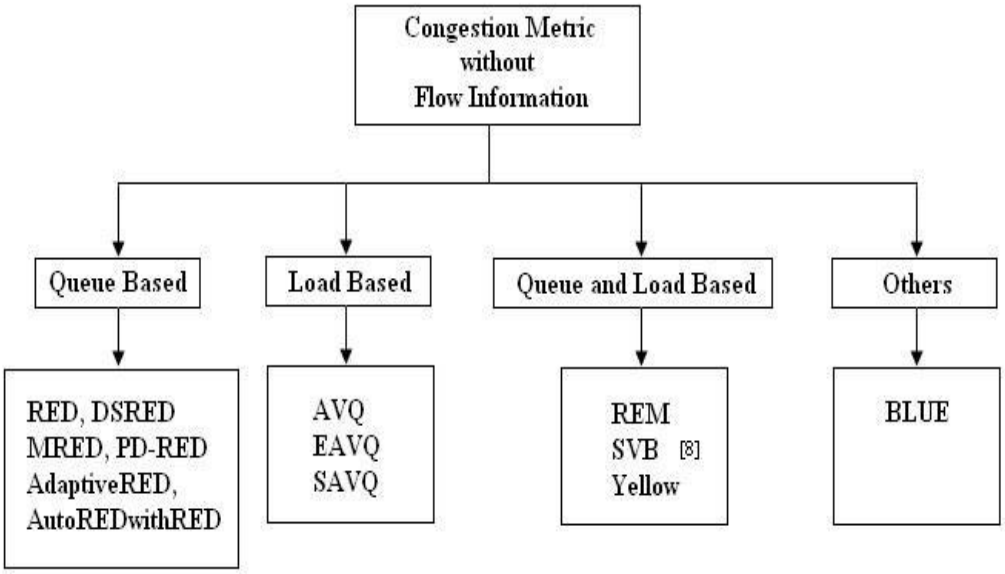

Figure 2.3 Classifications of AQM Schemes based on Congestion Metric without Flow information

2.2.1.1 Queue Based AQMs

As shown in Figure 2.3, some of the queue based algorithms that are significant are RED, AdaptiveRED and AutoREDwithRED.

i) Random Early Detection (RED)

The first most well-known AQM algorithm is the Random Early Detection (RED) algorithm developed by Sally Floyd and Van Jacobson in the year 1999. It is one of the popular algorithms. The design of RED was a response to some of the problems with Drop-Tail queue i.e a traditional queue management. The Internet Engineering Task Force recommends the deployment of RED in [4**].** It tries to avoid problems like global synchronization, lock-out, bursty drops and queuing delay that exist in the Drop-Tail scheme. This algorithm detects congestion by computing the average queue size Q_{avg}. To calculate average queue size, low pass filter is used which is an exponential weighted moving average (EWMA) and uses w_q as a queue weight which is a constant, and Q_t is the instantaneous queue occupancy using the following equation:

$$Q_{avg,t+1} = (1 - w_q) \cdot Q_{avg,t} + w_q \cdot Q_t$$

The average queue is then compared with two thresholds: a minimum threshold min_{th} and a maximum threshold max_{th}. If the average queue size is between minimum and maximum threshold, the packet is dropped with a probability. If it exceeds maximum threshold, then the incoming packets are dropped. Packet drop probability is a linear function of queue length. Global synchronization is a cycle of burst of drops followed by a period of overload resulting in under-utilisation. Global synchronisation occurs because the burst of drops results in a large number of TCP sources reducing their window size / sending rate at the same time. With RED, the early packet drops are randomised and spread over time. Packet marking/dropping is uniformly random for each packet. This reduces global synchronisation of sources.

Lock-Out describes the effect when some TCP sessions receive significantly less bandwidth than the other established connections. This phenomenon is reported in [4], and stems from synchronisation phenomena.

However, it also faces weaknesses such as accurate parameter configuration and tuning. Even though RED avoids global synchronization it fails when load changes dramatically. In case of large queue, RED has continuous packet loss followed by lower load that leads to reduced link utilization.

The instability of RED is discussed in numerous papers. May *et al.*'s critical evaluation of RED in [32] summarizes as follows: "RED with small buffers does not improve significantly the performance of the network", and "Parameter tuning in RED remains an inexact science, but has no big impact on end-to-end performance". In [7], the difficulty of tuning RED parameters to observe performance increase in web traffic is discussed. The oscillatory behaviour and possible instability of RED is again confirmed in [15] by a more analytical approach. In "Reasons not to deploy RED" [33] it is questioned whether RED gives any benefit over Drop-Tail.

AdaptiveRED [17] tries to improve the parameter tuning problem in RED. AdaptiveRED tries to adapt max_p to control the relationship between the average queue size and the packet drop probability to maintain a steady average queue size. While AQMs like PD-RED [38], MRED [27] and DS-RED [46] try to improve their performance compared to RED, the problem of unfairness and Queue oscillation still existed in these AQMs. In MRED, the packet drop probability is computed in a step form using packet loss and link utilization history to solve the problem caused by RED. The changes in the packet drop function of DS-RED attempted to modify the parameters of RED but resulted in limited improvement in throughput. PD-RED is based on the control theory and adapts the maximal drop rate parameter of RED called max_p to stabilize the queue length.

Ziegler *et al.* [47][48][49] explore the stability of RED, and recommend settings for max_p so that the average queue size converges to $(min_{th}+max_{th})/2$. In [49], the goal is to converge to a certain queue size. The authors declare that,

"we define convergence very loosely as achieving a state of bounded oscillation of the queue-size around a value between min_{th} and max_{th} so that the amplitude of the oscillation of the average queue size is significantly smaller than max_{th}- min_{th} and the instantaneous queue-size remains greater than zero and smaller than the total buffer size". In [49], the authors recommend the settings of w_q, max_p, min_{th} and max_{th} to achieve these goals. Ziegler *et al.* set w_q as a function of the link bandwidth.

Christiansen *et al.*[6] evaluated RED experimentally in a laboratory scenario with web traffic with congestion only in the forward path, and concluded that RED offers no clear advantage over tail-drop FIFO in terms of per-connection response times for web users. The paper also reports that performance is quite sensitive to the setting of RED parameters. Misra *et al*. in **[**34**]** also discuss the difficulties in tuning RED parameters. They illustrate the benign oscillations in the instantaneous queue size, and say that they are currently investigating tuning RED parameters. Hollot *et al*. in [20] also focus on oscillations in the queue size, and recommends values for RED parameters.

In [38], the authors propose a new Adaptive RED scheme based on the proportional derivative (PD) control principle, called PD-RED. This scheme is based on control theory and adapts the maximal drop rate max_p parameter of RED to stabilize the queue length. PD-RED adapts every sampling time interval, so that the magnitude of the error signal for sampling interval is kept as small as possible.

SHRED [7] modifies min_{th} and max_p such that the slope of the drop probability line remains the same for a set of RED parameters based on cwnd ratio. Once the min_{th} is adjusted, max_p is modified to maintain the slope of original packet drop probability line. Based on the modified values, SHRED is implemented identical to RED. While RED can achieve an ideal operating point, it can only do so when it has a sufficient amount of buffer space and is correctly parameterized [9] [41].

ii) AdaptiveRED

AdaptiveRED AQM [17] retains RED's basic structure and merely adjusts the parameter max_p to keep the average queue size between min_{th} and max_{th}. However the general design of AdaptiveRED is summarized by setting w_q automatically (based on the link speed) and adapting max_p in response to measured queue lengths using an additive-increase multiplicative-decrease (AIMD) policy.

In particular, the original version of AdaptiveRED proposal of Feng *et al.* [10], increased multiplicatively when the average queue size was below min_{th} and decreased multiplicatively when the average queue size was above max_{th}.

In paper [44], a novel adaptive RED overcomes the drawbacks of original RED gateway, which adjusts the maximum drop ratio to keep the average queue length around the target value using gradient descent method. The gradient descent approach tunes maximum drop probability of the RED controller to near-optimal value. In [44], RED algorithm oscillates for violent heavy traffic loads and does not avoid the oscillation of instantaneous queue length, nor does it stabilize the average queue length even for heavy number of connections as in Figure 2.4. RED responses to the change of connections very slowly and queue length is dependent on network loads as shown in Figure 2.5 as referred in [44] where the bandwidth is shared varyingly by 200 flows. Ziegler *et al.* also show that the original AdaptiveRED [10] [11] does not always give good performance.

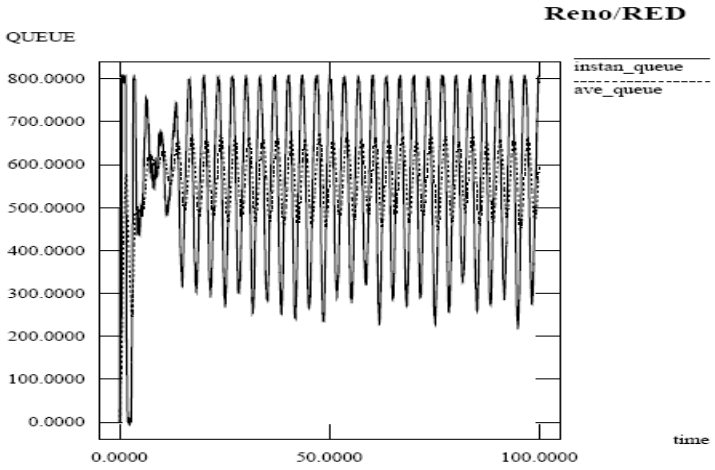

Figure 2.4 Queue Length Vs Time

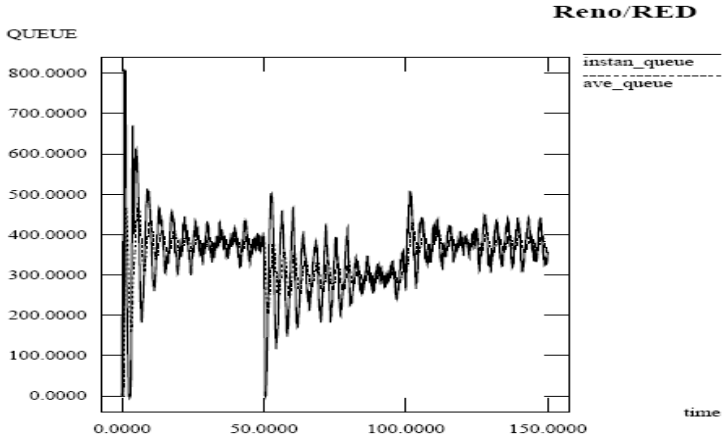

Figure 2.5 Queue Length Vs Time

iii) AutoREDwithRED

RED and its variant are sensitive to congestion and traffic characteristics [39]. The AutoRED feature takes care of the traffic properties, congestion characteristics and the buffer size. It also calculates the average queue size and uses w_q as a dynamic parameter. In this technique, $w_{q,t}$ is a combination of the three main network characteristics such as traffic properties, congestion characteristics and the queue normalization. So $w_{q,t}$ is written as a product of the three network characteristics. The AutoREDwithRED also calculates packet drop probability as RED. However, AutoREDwithRED performs better than the RED scheme. This model reduces the queue oscillations appropriately in the RED-based algorithms. The AutoRED uses the strength and effect of

both the burstiness and the transient congestion. The dynamic varying nature of w_q takes care of the network characteristics but the varying flows like unresponsive flow and misbehaving flows are not taken care in this AQM. The Figure 2.6 shows RED's poor adaptability as in [39] for varying connections.

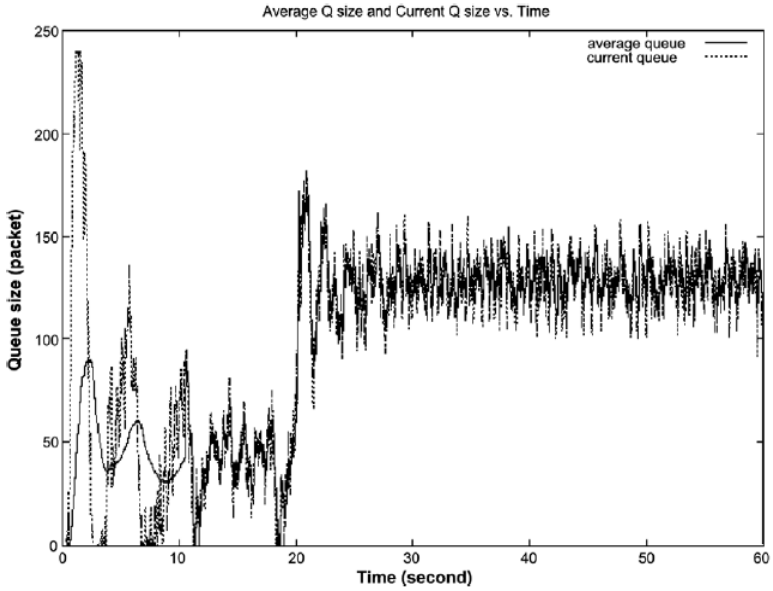

Figure 2.6 Queue Length Vs Time

2.2.1.2 Load-Based AQMs

As in Figure 2.3, among the load based scheme, the Adaptive Virtual Queue is a significant one.

i) Adaptive Virtual Queue

In [28] [36], packet interarrivals from individual sources are driven by TCP dynamics and source interarrivals themselves are heavy-tailed in nature. Therefore AVQ considers packet arrival rate for incipient congestion detection. Virtual Queue-based marking schemes have also been proposed for AQM in Internet routers. Adaptive Virtual Queue (AVQ) [25] scheme is one of those schemes. AVQ is a methodology for finding the fastest rate at which the marking probability is adapted. The marking probability in the AVQ is implicit; no marking probability is explicitly calculated. The marking

probability calculation is replaced with the computation of the capacity of a virtual queue. When a packet arrives in the real queue, the virtual queue is also updated to reflect a new arrival as originally proposed in [26] as a rate-based marking scheme. The virtual queue is updated, when a packet arrives at the real queue to indicate the new arrival of the packet as follows:

$$\tilde{C} = \alpha(\gamma C - \lambda) \qquad 2.1$$

where λ is the arrival rate at the link,

γ is the desired utilization link and

α is the damping factor

According to AVQ, *C* is the capacity of a link at a router. The router maintains a virtual queue whose capacity $\tilde{C} = C$ and whose buffer size is equal to the buffer size of the real queue. Upon each packet arrival, a fictitious packet is enqueued in the virtual queue if there is sufficient space in the buffer. If the new packet overflows the virtual buffer, then the packet is discarded in the virtual buffer and the real packet is dropped at the router. AVQ is primarily a rate-based marking, as opposed to queue length or average queue length based marking. The service rate of the virtual queue is adaptable and a packet is dropped whenever the virtual queue overflows the physical buffer limit. As explored by Hollot et al [20] [21] and advantages mentioned in Kelly *et al* [24], the scheme provides an early feedback. AVQ does not have any explicit control for queue length and it tends to maintain a small queue at the cost of attainable throughput (goodput).

When the virtual queue or buffer overflows, the packets are marked / dropped. The virtual capacity of the link is modified such that total flow entering each link achieves a desired utilization of the link. This is done by

aggressive marking when the link utilization exceeds the desired utilization and less aggressive when the link utilization is below the desired utilization. As a result the method provides early feedback than the RED.

As discussed in [30][40], the robustness and dynamic characteristics of AVQ are rather sensitive to control parameters α and γ. Figure 2.7 shows how the queue length and the transient responsiveness in AVQ are influenced by parameters α and γ largely in [40**]**. The queue size is not controlled explicitly in AVQ which makes queue length oscillate remarkably in a dynamically varying traffic and this scheme does not use the link utilisation entirely.

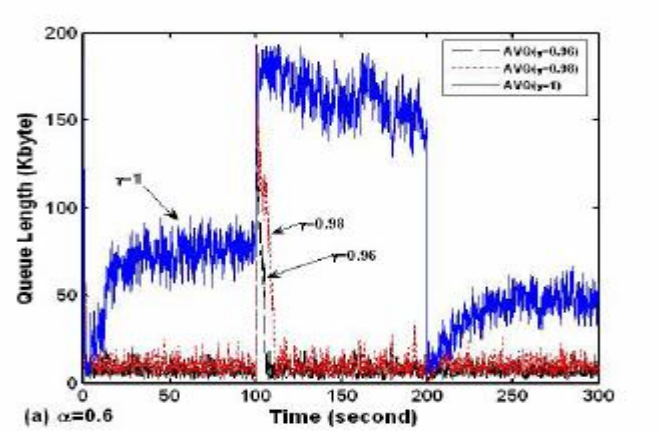

Figure 2.7 Queue Length Vs Time

Even in a rate based AQM, the values of parameter are very important to a very large extent. As discussed in [40], parameter `$\alpha$` is set to a conservative value and parameter `$\gamma$` is a constant value less than 1 to ensure the stability and good performances of the system especially in dynamic scenario. However, the concept of desired link utilization ratio $\tilde{\gamma}$ is introduced into EAVQ scheme to makes much better control flexible by avoidance of the physical limitation of $\gamma < 1$. An adaptive mechanism of the parameter $\tilde{\gamma}$ is followed to solve the problem of sensitivity of parameter γ of the AVQ

scheme. Therefore, the parameter α can be set large value, which ensures the stability and faster responsiveness of the system.

2.2.1.3 Summary – Congestion metric without Flow information

One of the fundamental problems with RED based AQM techniques is that they rely on queue length as an estimator of congestion. While the presence of a persistent queue indicates congestion, its length gives very little information about the severity of congestion, that is, the number of competing connections sharing the link. In a busy period, a single source transmitting at a rate greater than the bottleneck link capacity can cause a queue to build up just as easily as a large number of sources can. From well-known results in queuing theory, it is only when interarrival time of packets have a Poisson distribution, the queue lengths directly relate to the number of active sources indicating the true level of congestion. Unfortunately, the interarrival time packets across network links are decidedly non-Poisson. Packet interarrivals from individual sources are driven by TCP dynamics and source interarrivals themselves are heavy-tailed in nature. This makes placing queue length at the heart of an active queue management scheme dubious. Since the RED based algorithms only rely on queue lengths, it has an inherent problem in determining the severity of congestion. As a result, Queue based AQMs requires additional parameters to operate correctly under different congestion scenarios.

So some of the AQM schemes detect congestion based on the arrival rate of the packets at the link (e.g., virtual queue-based schemes). These AQM schemes involve adapting the marking probability well to changing network conditions. This feature does not exist in Queue-based AQM schemes. Though the adaptability of these AQMs is better but does not achieve a high utilisation in case of a dynamic traffic consisting of different types of flows.

Some AQM schemes based on concurrent queue and load merits determine the packet dropping probability and try to minimise queue size rather

than keeping it around a target resulting in a poor utilisation. These AQM schemes achieve a higher queuing delay incase of varying traffic types in a network. Therefore, the queue dynamics is uncontrolled, which results in a subtle problem of large queuing delay jitter. Some of these AQMs consider the input traffic rate and queue size as peer metrics whereas some of them consider the load factor as the main merit and queue size as the secondary merit. Another difficulty is the parameter setting problem under a wide range of traffic environments.

Firstly although these above AQMs perform better they still have a bias against fragile connections or adaptive flows. The fact that in these AQMs all connections achieve the same loss rate and even a connection using much less than its fair share will experience packet loss. This can prevent a low-bandwidth TCP from ever reaching its fair share, since each loss may cause TCP to reduce its window size by one half. Secondly accepting a packet from one connection causes higher drop probability for future packets from other connections, even if the other connections consume less bandwidth. This causes temporary undesirable non-proportional dropping even among identical flows. Thirdly a non-adaptive connection forces these AQMs to drop packets at a high rate from all connections. This contributes to these AQM's inability to provide a fair share to adaptive connections in the presence of aggressive users even if the congestion is not severe.

A non-adaptive traffic shares resources with TCP data traffic in an IP network. However, non-adaptive connections do not stop sending packets under congestion. Unless they are taken care of, they compete unfairly with adaptive sources for buffer space and bandwidth. An AQM must be able to isolate non-adaptive greedy traffic more effectively and improve fairness when different traffic types and different speed flows share a gateway. This is possible by considering additionally the flow information by these AQMs.

Therefore the following section discusses about the AQMs that consider the flow information for incipient congestion detection.

2.2.2 Congestion Metric with Flow information

AQMs also belong to this category that uses both congestion metric and the flow information to detect congestion in routers. AQMs that used only congestion metric and not flow information faced the problem of unfairness in handling the different types of traffic. While considering the congestion metric they can be further classified as queue-based or load based and others as shown in Figure 2.10.

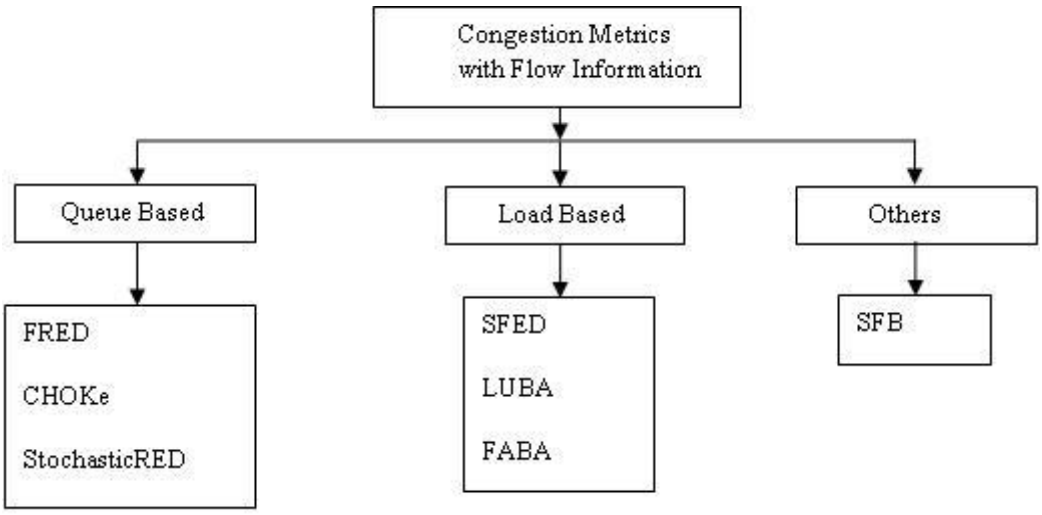

Figure 2.10 Classifications of AQM Schemes based on Congestion Metric with Flow information

2.2.2.1 Queue based

Some of the significant algorithms are FRED, CHOKe, StochasticRED.

i) FlowRED

In this paper [29] it is pointed out that RED allows unfair bandwidth sharing when a mixture of the three traffic types shares a link. This unfairness is caused by the fact that at any given time RED imposes the same loss rate on all flows, regardless of their bandwidths. FRED uses per-active-flow

accounting to impose on each flow a loss rate that depends on the flow's buffer use. It is shown that FRED provides better protection than RED for adaptive (fragile and robust) flows. In addition, FRED is able to isolate non-adaptive greedy traffic more effectively. A "two-packet-buffer" gateway mechanism is used to support a large number of flows without incurring additional queuing delays inside the network. FRED's per active-flow accounting uses memory in proportion to the total number of buffers used. A FRED gateway maintains state only for flows for which it has packets buffered, not for all flows that traverse the gateway.

To improve RED'S ability for distinguishing unresponsive users, Flow Random Early Drop (FRED) has been proposed in [29]. However, this incurs extra implementation overhead since they need to collect certain types of state information and FRED needs information about active connections. Figure 2.11 shows a poor performance of FRED as in **[**23**].**

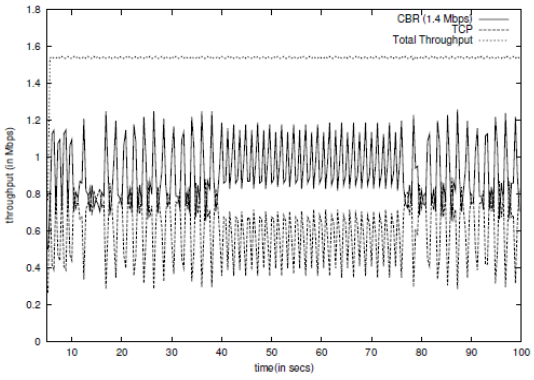

Fig. 7. Throughput vs time for FRED algorithm

Figure 2.11 Throughput Vs Time

ii) CHOKe

CHOKe (CHOose and Keep for responsive flows, and CHOose and Kill for unresponsive flows) [35] algorithm penalizes misbehaving flows by dropping more of their packets. So CHOKe tries to bring fairness for the flows that pass through a congested router. CHOKe calculates the average occupancy of the buffer using EWMA. If average queue size is greater than min_{th}, the

flowid of each arriving packet and a randomly selected packet called drop candidate packet is compared. If the packets are of the same flow then both the packets are dropped. Otherwise if average queue is greater than max_{th}, then the new packet is dropped else the packet is placed in the buffer and the new packet is admitted with a probability p.

The reason for this approach is that the buffer is more likely to have packets belonging to a malicious flow and so these packets are more likely to be chosen for comparison. Further, packets belonging to a malicious or misbehaving flow arrive more numerously and are more likely to trigger comparisons. The intersection of these two high probability events is precisely the event when the packets belonging to misbehaving flows are dropped. Therefore, packets of misbehaving flows are dropped more often than packets of well-behaved flows.

The CHOKe algorithm is interesting because of its performance as well as its simple and elegant implementation. The algorithm is local to a link and requires no per-flow information. In [42], Wang *et al.* show analytically that this simple algorithm controls high-bandwidth unresponsive flows such that as their bandwidth increases, their share of the link capacity decreases to 0. This can be intuitively understood, since the probability of dropping a packet increases for a high-bandwidth flow since (a) more packets arrive and trigger comparison and drop operation (b) the randomly picked packet is likely to be from a high-bandwidth flow. Results shown in [35] indicate that CHOKe is able to control high-bandwidth unresponsive UDP flows, so that TCP connections can share the link more equitably. Figure 2.12 shows a poor performance of CHOKe as in [23].

.

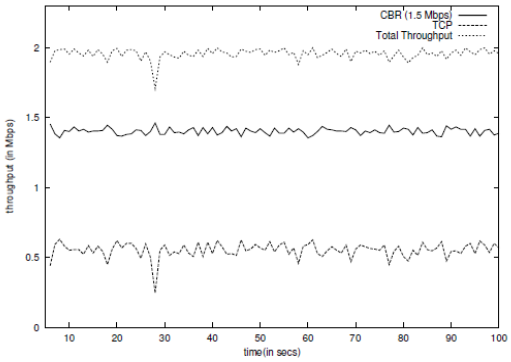

Fig. 6. Throughput vs time for CHOKe algorithm

Figure 2.12 Throughput Vs Time

2.2.2.2 Summary – Congestion Metric with Flow information

The above AQM schemes use the flow information to control the fair share bandwidth of the link. These AQMs tune the packet dropping probability for all the flows by considering the link utilization or bandwidth share obtained by the flows. The dropping probability is adjusted such that the packets of the flow with higher transmission rate will more likely be dropped than flows with lower rate. They also consider the input arrival rate that helps in identifying the malicious flows. The above AQMs bring in fairness with or without per flow active flow state. These AQMs use hash function and requires table operation and multiple parameters to be tuned to take care of the flow information of the malicious flow. Some AQMs do not maintain per-flow state information but instead employs hash flows into different bins. Most of these AQMs isolate the varying flows to adjust the drop probability to control the fairness in the routers.

Many of the AQMs use the congestion metric with flow information to detect congestion. However, there exist very few AQMs that use only the flow information to detect congestion.

This section classified and discussed about the various existing AQM schemes. The various AQMs are compared with respect to characteristics and performance in the next section.

2.3 COMPARISON OF AQM SCHEMES

In the recent years many AQM mechanisms have been developed which try to solve the Internet congestion at router level. The various problems like lock-out, global synchronization and fairness are the issues that are considered in these AQMs. These problems are addressed using various factors. The existing AQM schemes are classified and their performance is studied. According to the classification, basically most of the AQMs employed only congestion metric to detect the congestion. However, some of the AQMs required additional flow information other than the congestion metric to know the accurate status of the queue. Very few of the AQMs required only the flow information to spot out the congestion.

Considering these AQMs relevant to classification, the first category AQMs based on congestion metric without flow information are more simple and easy to design compared to the second category AQMs based on congestion metric with flow information. Additionally, the second category AQMs required extra overhead to achieve desired performance compared to the first category AQMs. The third category AQMs has a still greater complexity in identifying the flow information for calculating the marking probability.

Table 2.1 projects the AQMs queue occupation status. Most of the AQMs tried to keep the queue size around a target rather than maximizing or minimizing the queue. AQMs that tried to have the queue size around a target performed better than the other AQMs. RED is the first widely employed AQM which detects congestion using only the congestion metric and without flow information.

Table 2.1 Comparison of AQMs based on Classification

Classification	Congestion Metrics	AQM	Approach	Queue Occupation	Traffic
Congestion Metric without Flow Information	Queue	RED	Heuristic	Keeps Queue around a Target	Adaptive
		AutoRED	Heuristic	Keeps Queue around a Target	Adaptive
	Load	AVQ	Heuristic	Keeps Queue around a Target	Adaptive
	Queue and Load	REM	Optimisation	Minimises Queue	Adaptive
	Others	BLUE	Heuristic	Maximises Queue	Adaptive
Congestion Metric with Flow Information	Queue	CHOKe	Heuristic	Keeps Queue around a Target	Adaptive, Non-Adaptive
	Load	SFB	Heuristic	Maximises Queue around a Target	Adaptive, Non-Adaptive
Flow Information		GREEN [14]	Heuristic	Keeps Queue around a Target	Adaptive

Table 2.1 indicates that irrespective of the congestion indicator additional flow information gives better awareness of congestion in routers. Based on RED AQM, many variant AQMs are developed. RED AQM uses multiple parameters that are to be fine tuned. So RED faced this problem of parameter tuning. As a result, packet loss and utilization at the link varied with regard to the network load variation. Network load variation also leads to the existence of global synchronization. RED based AQMs like DSRED, MRED, AdaptiveRED and AutoREDwithRED techniques tried to solve the limitations of RED. DSRED and MRED show better performance compared to RED. AdaptiveRED and AutoREDwithRED techniques tried to eliminate the problem of parameter tuning by adapting to the parameters. Though RED and its variant are simple to handle, the difficulty with them is the parameter tuning problem.

RED based AQMs are vulnerable to unresponsive flows dominating router queues. To overcome this problem, FRED is proposed that improved uniformity by constraining all flows to occupy loosely equal shares of the queue's capacity. It removed the problem of unresponsive flows dominating a queue. Though the congestion metric is used in the design, it also has to keep track of the additional flow information to control congestion which became the major weakness of FRED. Based on this, certain AQMs are developed to get rid of the overhead. Combination of Flow and congestion metric based AQMs like CHOKe, SFB [13], SFED [23], FABA [22] and StoRED [5] are proposed to allocate fair buffer between flows considering the effects of misbehaving or non-responsive flows. CHOKe provides much better fairness than FRED but penalizes higher bandwidth flows and does not handle unresponsive flows in case of few packets.

Flow based AQMs with congestion metric are able to discriminate between responsive and non-responsive flows. The malicious flows are identified which might cause congestion at the router. Stochastic RED is based on the concept of flow-based AQM and it is a simple powerful RED algorithm. To avoid maintaining per flow state as in other flow-based AQM, StoRED uses the idea of the time varying hash function to map flows to different counting buns. StoRED is outstanding in disciplining misbehaving flow, making unresponsive flows TCP-friendly, and improving the responsive users of web transfers.

Further these AQMs are classified based on the congestion metric. Most of the AQMs used congestion metric to detect congestion. A variety of congestion indicators like queue length, input rate, packet loss and link utilization are used for congestion detection. RED based AQMs used queue length as congestion indicator. Some of the AQMs tried to prove that Queue status does not give a clear status of the congestion.

Table 2.2 Comparison of AQMs based on Performance

Classification	Congestion Metrics	AQMs	Link Utilisation	Loss Rate	Queue Stability	Fairness	Queuing Delay
Congestion Metric without Flow Information	Queue	RED	High	Moderate	Moderate	Low	High
		AutoRED	High	Moderate	Moderate	Low	High
	Load	AVQ	Moderate	High	High	Low	Low
	Queue and Load	REM	Low	High	Low	Low	Very High
	Others	BLUE	Low	Moderate	Moderate	Low	High
Congestion Metric with Flow Information	Queue	CHOKe	High	High	High	High	Moderate
	Load	SFB	Moderate	Moderate	Moderate	Low	Low
Flow Information		GREEN	Low	High	Low	Low	Low

REM [2] used both input rate and queue length that illustrated very low throughput compared to Queue based RED. YELLOW **[**31**]** used input rate as the primary congestion indicator and queue as the secondary indicator to demonstrate that it performed well in terms of link utilisation and packet loss. BLUE [12] **[**43] [3]. used packet loss and link utilization as congestion indicators to give a moderate throughput and utilisation with low queue stability. Table 2.2 indicates that the Load-based AQM AVQ performs better with queuing delay, adaptability compared to the Queue-based AQMs. Table 2.2 indicates that irrespective of the additional flow information, Load-based AQMs give better strength in adapting to packet arrivals in routers.

Tables 2.1 and 2.2 make clear that Queue based AQMs are simple to design with certain metrics to be improved. From the literature study, it is well understood that AutoREDwithRED a Queue based AQM considers the congestion and traffic factors to perform better than the other Queue based AQMs. Hence the Queue based AQM AutoREDwithRED is considered for an

improvement in the next chapters of the thesis. The next section discusses about AutoREDwithRED elaborately.

2.3.1 AutoREDwithRED - QUEUE BASED AQMs

The AutoREDwithRED [39] AQM shown in Table 2.3 calculates the queue weight w_q and then calculates average queue size for every packet arrival. The average queue size is compared to two thresholds: a *minimum* and a *maximum* threshold. When the average queue size is less than the minimum threshold, no packets are dropped. When the average queue size is greater than the maximum threshold, every arriving packet is dropped. If packets are, in fact, dropped or if all source nodes are cooperative, this ensures that the average queue size does not significantly exceed the maximum threshold.

When the average queue size is between the minimum and maximum thresholds, each arriving packet is dropped with probability p_b, where p_b is a function of the average queue size. Thus, the AutoRED AQM has two separate algorithms. The algorithm for computing the average queue size determines the degree of burstiness that will be allowed in the gateway queue.

Table 2.3 Pseudocode of AutoREDwithRED

```
For every packet arrival {
Calculate w_q
Calculate Q_avg
if (Q_avg < min_th)
        Forward the new packet
Else
        If Q_avg > = max_th
                Drop the new packet
        Else
                Calculate p_b = max_p(Q_avg - min_th) / (max_th - min_th)
                Calculate p_a = p_b/(1 - count*p_b)

                Drop arriving packet with a probability p_a

}
```

The algorithm for calculating the packet-dropping probability determines how frequently the gateway drops packets, given the current level of congestion. The goal is for the AQM to drop packets at fairly evenly spaced intervals, in order to avoid bias and avoid global synchronization, and to drop packets sufficiently frequently to control the average queue size.

The RED AQM uses a lowpass filter to calculate the average queue size as follows:

$$Q_{avg=} (1- w_q) \cdot Q_{avg} + w_q \cdot Q_t \qquad 2.2$$

In this case, if w_q is too large, then the averaging procedure will not filter out transient congestion at the gateway. If w_q is set too low, then Q_{avg} responds too slowly to changes in the actual queue size. In such a case, the gateway is unable to detect the initial stages of congestion.

Therefore the average queue size is slowly varied due to small and inaccurate value of w_q and it moves the queue dynamics from a stable point to an oscillatory behaviour. Hence the AutoRED technique modifies the fundamental way of calculating the average queue size. It interprets the effect of the weight parameter w_q and the network characteristics on the average queue size as follows:

$$Q_{avg,t} - Q_{avg,t-1} = w_{q,t} \cdot (Q_t - Q_{avg,t-1}) \qquad 2.3$$

The term $Q_{avg,t} - Q_{avg,t-1}$ defines the changes in the average queue size over a time. The term $Q_t - Q_{avg,t-1}$ describes the changes in the instantaneous queue size with respect to the average queue size over a time. These two changes reflect two types of network characteristics namely congestion characteristics and traffic characteristics. The purpose of the weight w_q on the right hand size is to filter the short term changes in the queue size resulted from

the bursty traffic or transient congestion. These characteristics are incorporated in the calculation of the average queue size using the weight parameter w_q. Therefore, this way of calculating the weight parameter and the average queue size eliminates the chaotic queue oscillation. Thus, the short-term increase in the queue size resulting from bursty traffic or transient congestion does not result in a significant increase in the average queue size.

AutoREDwithRED tries to achieve the goal of stable average queue size by removing the parameter sensitivity problem in terms of dynamic w_q. According to AutoRED, the slowly varying nature of the average queue size in RED is due to computation using an exponentially weighted moving average with constant w_q that causes the chaotic oscillation. AutoRED uses a mathematical function to model the weighting parameter w_q used in the EWMA. This weighting function uses the knowledge of the dynamic changes in the congestion characteristics, traffic characteristics and queue normalization. This reduces the chaotic queue oscillation significantly but it also tries to provide low delay with high throughput and removes the parameter tuning in terms of w_q.

The definition of the weighting parameter w_q is modelled as a product of three functions as follows:

$$w_{q,t} = p_t \cdot (1 - p_t) \times \frac{2(5.923 + |Q_t - Q_{avg,t-1}|)}{\ln(5.923 + |Q_t - Q_{avg,t-1}|)} \times \frac{1}{bs} \qquad 2.4$$

where

$w_{q,t}$ = Time dependent weighing function

Q_t = Instantaneous queue size at time t

$Q_{avg,\ t-1}$ = Average queue size at time t-1

p_t = Probability that the network can lead to congestion at time t

bs = Buffer size

The first, second and third functions are denoted by T_t, J_t and K_{bs} that represents congestion characteristics, traffic characteristics and queue normalisation respectively as follows:

$$T_t = p_t\ (1 - p_t) \quad 2.5$$

$$J_t = \frac{2(5.923 + |Q_t - Q_{avg,t-1}|)}{\ln(5.923 + |Q_t - Q_{avg,t-1}|)} \quad 2.6$$

$$K_t = \frac{1}{bs} \quad 2.7$$

The weight parameter w_q is chosen as a small constant to ignore the short lived bursty traffic when the average queue size is calculated to make the packet drop decision. The choice of small value for the weight leads to slowly varying effect on the average queue size resulting in slow response to transient congestion as well as high continuous bursty traffic. This response characteristic causes instantaneous queue oscillation. This results in fluctuations between the buffer overflow and under-utilisation. Hence, the strength and effect of the burstiness or transient congestion is used in the calculation of w_q in addition to incorporating bursty traffic. This reduces the effect of slowly varying nature of the average queue size resulting in a reduced queue oscillation. Therefore, to remove the instantaneous queue oscillation, AutoRED considers both the strength and effect of the burstiness and transient congestion in calculating the average queue size. The strength quantifies the amount of the burstiness and the transient congestion, while the effect quantifies the amount of threat of the burstiness and the transient congestion.

The strength of the burstiness and the transient congestion are determined by the congestion characteristics parameter T_t and the buffer normalization. The effect of the burstiness and the transient congestion are

determined by the parameter traffic characteristics J_t. If the gateway is less like to have transition between congestion and no congestion or vice versa, then small value of w_q can be used to ignore the burstiness. However, in case of transition between congestion and no congestion or vice versa, then burstiness or transient congestion is to be incorporated in the calculation of average queue size. Therefore, AutoRED tries to incorporate the traffic characteristics, congestion characteristics and queue normalization to the weighting parameter w_q.

AutoRED AQM shows a very high performance due to the dynamic weighting function w_q. It also tries to remove the parameter tuning problem by incorporating the dynamic weighting parameter w_q and considers the changes in the average queue size and changes in the instantaneous queue size. As a result, it reflects two types of network characteristics namely the congestion characteristics (which include transient characteristics) and the traffic characteristics (which include bursty traffic). The purpose of the weight w_q with the changes in the instantaneous queue size is to filter the short-term changes in the queue size resulted from the burst of traffic or transient congestion. As a result, the short term change in the queue size does not significantly affect the average queue size increase, because the weight w_q is set to a small constant value recommended by the RED scheme.

The probability p_t that the network can lead to congestion at t+1 based on the queue status at time 't' is calculated using the following equation:

$$p_t = \frac{n_t}{n_t + n_t'} \qquad 2.8$$

where n_t and n_t' stand for the number of times $q_i \geq q_{avg,i-1}$ and $q_i < q_{avg,i-1}$ respectively within the time duration t.

Hence, the network with no congestion is expressed as

$$1 - p_t = \frac{n_t'}{n_t + n_t'} \quad 2.9$$

Therefore, the probability of congestion is defined as follows:

$$p_t = \begin{cases} \dfrac{n_t}{(n_t + n_t')}, & n_t = n_t + 1, \ \ if\ Q_i \geq Q_{avg,i-1} \\ & n_t' = n_t' + 1, \ \ if\ Q_i < Q_{avg,i-1} \end{cases}$$

AutoRED algorithm calculates the number of times n_t the instantaneous queue size is greater or equal to the average queue size and n_t' that is lesser than the average queue size. Based on the number of times n_t and n_t', calculate the probability of congestion p_t. This probability defines the system that alternates between congestion and no congestion. This probability is used to alter the change in the instantaneous queue size with respect to average queue size. This is incorporated as the congestion characteristics in the calculation of the average queue size. The dynamic value of w_q is calculated using the congestion characteristics and traffic characteristics and queue normalisation. Using the dynamic characteristics of w_q, the average queue size of the buffer is calculated. It also indicates two thresholds on the buffer, a minimum threshold min_{th} and a maximum threshold max_{th}.

AQM drops each packet that arrives at the gateway when the average queue size Q_{avg} exceeds max_{th}. When the Q_{avg} is between min_{th} and max_{th}, the probability of packet drop is as follows:

$$p_b = \frac{Q_{avg} - min_{th}}{max_{th} - min_{th}} \cdot max_p \quad 2.10$$

$$p_a = \frac{p_b}{1 - count \cdot p_b} \quad 2.11$$

As average queue size Q_{avg} varies from min_{th} to max_{th}, the packet-dropping probability p_b varies linearly from 0 to max_p, final packet-dropping probability p_a increases slowly as the count increases since the last dropped packet.

The optimal values for min_{th} and max_{th} depend on the desired average queue size. If the typical traffic is fairly bursty, then min_{th} must be correspondingly large to allow the link utilization to be maintained at an acceptably high level. AutoRED AQM functions most effectively when $max_{th} - min_{th}$ is larger than the typical increase in the calculated average queue.

Probability of packet drop is expressed as follows

$$p_b = \begin{cases} \max\left(0, \max_p \times \dfrac{Q_{avg} - \min_{th}}{Cur_{\max_{th}} - \min_{th}}\right), \\ \qquad if\ Q_{avg} < Cur_{\max_{th}} \\ 1, \qquad Otherwise \end{cases}$$

2.4 OBSERVATIONS MADE DUE TO LITERATURE REVIEW

Table 2.4 shows the performance values of the existing AQMs. The performance value proves the discussion in the previous section of this chapter. Comparing the existing AQMs, AutoREDwithRED outperforms other AQMs in terms of link utilization. Hence, the performance efficient AQM AutoREDwithRED can be improved in terms of performance metrics like

- High Fairness
- Reduced Packet Drop Rate
- Moderate and Stable Average Queue Size
- Increased Utilisation with minimum Queuing Delay

- Providing better QoS by preventing congestion in a dynamically varying traffic in IP networks.

Table 2.4 Performance Metrics of Existing AQMs

AQMs	Link Utilisation (In %)					In %	# Packets	
	CBR Utilisation	TCP Utilisation	Overall Utilisation	Fairness	Variation	Packet Drop Rate	Average Queue Size	Std. Dev. Avg. Queue Size
AVQ	97.05	0.64	97.70	0.5	0.22	9.72	50.2	2.8
REM	88.57	2.39	90.97	0.5	0.18	9.73	70.7	4.3
AutoREDwithRED	**97.29**	**1.70**	**99.00**	**0.5**	**0.22**	**9.74**	**190.4**	**3.5**
RED	97.166	1.83	98.99	0.5	0.22	9.72	170.7	5.2
CHOKe	40.54	54.50	95.05	0.54	0.034	9.85	100.5	6.7

The existing active queue management schemes faces the problem of bias against flows in an IP network of varying traffic flows. AutoRED AQM does not attempt to ensure that each connection receives the same fraction of the total throughput, and do not explicitly control non-adaptive flows. AutoRED AQM provides a mechanism to identify the level of congestion, but could not identify connections using a large share of the total bandwidth. Non-adaptive connection forces AutoRED to drop packets at a high rate from other connections. This contributes to AutoRED's inability to provide a fair share to adaptive connections in the presence of aggressive users even if the congestion is not severe. Therefore, additional mechanisms are required to control the throughput of such connections during periods of congestion.

The average queue length depends on the traffic loads and parameter configuration, when the link is lightly congested or max_p is high, the average

queue length is near min_{th}, and when the link experiences serious congestion or max_p is very low, the average queue size is near max_{th}. Because of coherent relation among queue length, queuing delay, drop probability and utilization etc., the algorithm may lead to link underutilization with light traffic loads or decrease in utilization and increase in packet drop with heavy network traffic. Delay being a major component of the quality of service delivered to the customers, to achieve such predictable average delays, constant tuning of the parameters is to be implemented to adjust to current traffic conditions.

Therefore AutoREDwithRED algorithm requires an improvement as a simple, stateless algorithm that can achieve flow isolation and/or approximate fair bandwidth allocation. A solution is to be identified to the above problem in the context of the Internet. Thus, it is important to improve AutoREDwithRED that differentially penalize "unfriendly" or "unresponsive" flows, which implies bad implementations of TCP, and UDP-based flows. Further, some key features are to be preserved that AutoREDwithRED possesses; such as its ability to remove parameter tuning problem by having a dynamic w_q.

2.5 SUMMARY

The comparison of the various congestion indicators indicates that Queue based AQMs are simple to design except for the parameter tuning problem compared with the other AQMs. Irrespective of the AQMs that depend on flow information, Load based AQMs performed better than Queue-based in terms of high adaptability. Flow based AQM shows better fairness compared to others. This study indicates that most of the AQMs used queue length or input rate as their congestion indicators. While using the flow information, the AQMs used either queue length or input rate and not both. Further AQMs can be designed that can use both queue length and input rate as congestion metric with flow information. Therefore, an Active Queue Management is to be designed that inherits the advantages of Queue-based AQMs, Load-based AQMs and AQMs with flow information. Hence, the performance efficient

existing Queue based AQM AutoREDwithRED is improvised to prevent congestion in a dynamically varying traffic in IP networks. It should provide a solution to networks that vary in terms of fairness, packet drop rate, moderate queue size with stability and link utilization with queuing delay. The next section discusses the proposed methodology followed in the research work to enhance the performance metrics of AutoREDwithRED.

Chapter References

1. M. K. Agarwal, R. Gupta, V. Kargaonkar, "Link Utilisation Based AQM and its Performance", IEEE Communications Society, Globecom 2004, December 2004.

2. S. Athuraliya, V. H. Li, S. H. Low, and Q. Yin, "REM: Active Queue Management" IEEE Network, Vol. 15, pp. 48-53, 2001.

3. Y.H. Aoul, A. Mehaoua, C. Skianis, "A fuzzy logic-based AQM for real-time traffic over Internet", ScienceDirect, Computer Networks 51, 4617 – 4633, June 2007.

4. B. Braden, D. Clark, J. Crowcroft et al., "Recommendations on Queue Management and Congestion Avoidance in the Internet", RFC 2309, April 1998.

5. S. Chen, Z. Zhou, B. Bensaou, "Stochastic RED and its applications", International Conference on Communication ICC'07, pp. 6362 – 6367, June 2007.

6. M. Christiansen, K. Jeffay, D. Ott et al. "Tuning RED for Web Traffic", Proceedings of the conference on Applications, Technologies, Architectures, and Protocols for Computer Communication SIGCOMM 2000, pp. 139-150, 2000.

7. M. Claypool, R. Kinicki, M. Hartling, "Active Queue Management for Web Traffic", IEEE International Conference on Performance, Computing and Communication, 2004.

8. X. Deng, S. Yi, G. Kesidis, Chita R. Das, "Stabilised Virtual Buffer (SVB) - An Active Queue Management Scheme for Internet Quality of Service", IEEE Globe Telecommunication Conference GLOBECOM 2002, November 2002.

9. S. Doran, "RED Experience and Differentiated Queueing", NANOG Meeting, June 1998.

10. W. Feng, D. Kandlur, D. Saha et al., "A Self-Configuring RED Gateway", Annual Joint Conference of the IEEE Computer and Communication Societies INFOCOM `99, March 1999.

11. W. Feng, D. Kandlur D. Saha et al., "Techniques for Eliminating Packet Loss in Congested TCP/IP Network", UMCSE-TR-349-97, November 1997.

12. W. Feng, D. Kandlur, D. Saha et al. "Blue: A New Class of Active Queue Management Algorithms", UMichigan CSE-TR-387-99, 1999.

13. W. Feng, D. Kandlur, D. Saha et al., "Stochastic Fair Blue: A Queue Management Algorithm for Enforcing Fairness", IEEE INFOCOM, 2001.

14. W. Feng, A. Kapadia, S.Thulasidasan, "GREEN: Proactive Queue Management over a Best-effort Network", IEEE GLOBECOM 2002, November 2002.

15. V. Firoiu and M. Borden, "A Study of Active Queue Management for Congestion Control", IEEE INFOCOM 2000, pp. 1435-1444, 2000.

16. S. Floyd and V. Jacobson, "Random early detection gateways for congestion avoidance", IEEE/ACM Transaction Networking, Vol. 1, pp. 397–413, Aug. 1993.

17. S. Floyd, S. Gummadi, S. Shenkar and ICSI, "Adaptive RED: An algorithm for increasing the robustness of RED's active Queue Management", http:www.icir.org/floyd/red.html.

18. S. Floyd and V. Jacobson, "On traffic phase effects in packet-switched gateways, Internetworking: Research and Experience", Vol. 3, No. 3, pp. 115-156, September 1992.

19. S. Floyd and V. Jacobson, "The synchronization of periodic routing messages", IEEE/ACM Transactions on Networking, Vol. 2 No.2, pp. 122-136, April 1994.

20. C. Hollot, V. Misra, D. Towlsey et al., "A control theoretic analysis of RED", UMass CMPSCI Technical Report 00-41, 2000.

21. C. Hollot, V. Misra, D. Towlsey et al., "On designing improved controllers for AQM routers supporting TCP flows", UMass CMPSCI Technical Report 00-42, 2000.

22. A. Kamra, H. Saran, S. Sen et al., "Fair Adaptive Bandwidth allocation: a rate control based active queue management discipline", Computer Networks 44, pp.135–152, 2004.

23. A. Kamra, S. Kapila, V. Khurana, et al, "SFED: a rate control based based active queue management discipline", IBM India Research Laboratory Research report #00A018, November 2000.

24. F. Kelly, P .Key and S. Zachary, "Distributed admission control", IEEE Journal on Selected Areas in Communications, 18, pp. 2617-2628, 2000.

25. S. Kunniyur, R. Srikant, "Analysis and design of an adaptive virtual queue (AVQ) algorithm for active queue management", Proceedings of ACM SIGCOMM'01, 27-31, August 2001.

26. S. Kunniyur and R. Srikant, "End-to-end congestion control: utility functions, random losses and ECN marks", Proceedings of Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies INFOCOM `00, Vol. 3, pp. 1323 – 1332, March 2000.

27. J. Koo, B. Song, K. Chung et al., "MRED: A New Approach to Random Early Detection", 15th International Conference on Information Networking, pp. 347 – 352, February 2001.

28. W. Leland, M. Taqqu, W. Willinger, and D. Wilson, "On the Self-Similar Nature of Ethernet Traffic (Extended Version)", IEEE/ACM Transactions on Networking, 2(1), February 1994.

29. D. Lin, and R. Morris, "Dynamics of random early detection", Proceedings of the Conference on Applications, technologies, architectures, and protocols for computer communication ACM SIGCOMM '97, pp. 127-137, October 1997.

30. C. Long, Bin Zhao, Xin-Ping Guan, "SAVQ: Stabilized Adaptive Virtual Queue Management Algorithm", IEEE Communications Letters, Vol. 9, pp. 78 – 80, January 2005.

31. C. Long., Bin Zhao., Xinping Guan., Jun Yang., ”The Yellow active queue management algorithm”, Computer Networks 47, pp. 525-550, November 2004.

32. M. May, T. Bonald, and J.C. Bolot, "Analytic Evaluation of RED Performance" Proceedings of INFOCOM 2000, 19th Annual Joint Conference of the IEEE Computer and Communication Societies, Vol. 3, pp. 1415-1424, March 2000.

33. M. May, J. Bolot, C. Diot, and B. Lyles, "Reasons Not to Deploy RED" Proceedings of 7th. International Workshop on Quality of Service IWQoS'99, pp. 260-262, 1999.

34. V. Misra, W. Gong, D. F. Towsley, “Fluid-based Analysis of a Network of AQM Routers Supporting TCP Flows with an Application to RED”, Proceedings of the conference on Applications, Technologies, Architectures, and Protocols for Computer Communication ACM SIGCOMM ’00, pages 151–160, 2000.

35. R. Pan, B. Prabhakar, and K. Psounis, "CHOKe, A Stateless Active Queue Management Scheme for Approximating Fair Bandwidth Allocation." Proceedings of INFOCOM 2000, 19th Annual Joint Conference of the IEEE Computer and Communication societies, Vol. 2, pp. 942-951, 2000.

36. V. Paxson and S. Floyd, "Wide-Area Traffic: The Failure of Poisson Modeling", Proceedings of ACM SIGCOMM, Conference on Communications architectures, protocols and applications SIGCOMM '94, pp. 257–268, August 1994.

37. W. Stevens et al., "TCP Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery Algorithms", RFC 2001, January 1997.

38. J. Sun, K. Ko, G. Chen et al., "PD – RED: To Improve Performance of RED", IEEE Communications Letter, Vol.7, pp. 406 – 408, August 2003.

39. S. Suthaharan "Reduction of queue oscillation in the next generation Internet routers", Computer Communications 30, pp. 3881-3891, October 2007.

40. Q. Yanping, L. Qi, L. Xiangze et al., "A Stable Enhanced Adaptive Virtual Queue Management Algorithm for TCP Networks", IEEE International Conference on Control and Automation ICCA 2007, pp. 360-365, 2007

41. C. Villamizar and C. Song, "High Performance TCP in ANSNET", ACM SIGCOMM Computer Communication Review, vol. 24, pp. 45–60, October 1994.

42. J. Wang, A. Tang, and S. H. Low, "Maximum and asymptotic UDP throughput under CHOKe", Proceedings of the ACM SIGMETRICS International conference on Measurement and modeling of computer systems SIGMETRICS '03, 2003.

43. B. Wydrowski and M. Zukerman, "GREEN: An Active Queue Management Algorithm for a Self Managed Internet", Proceedings of IEEE International Conference on Communications ICC 2002, pp. 2631-2635, 2002.

44. Y. Xu, Z. Wang, H. Wang, "ARED: A Novel Adaptive Congestion Controller", Proceedings of the Fourth International Conference on Machine Learning and Cybernetics, Vol. 2, pp. 18-21, August 2005.

45. L. Zhang and D. Clark, "Oscillating behavior of network traffic: A case study simulation", Internetworking: Research and Experience, Vol. 1, pp. 101- 112, 1990.

46. B. Zheng and M. Atiquzzaman, "DSRED: An Active Queue Management Scheme for Next Generation Networks", 25th Annual IEEE International Conference on Local Computer Networks LCN 2000, November 2000

47. T. Ziegler, S. Fdida, and C. Brandauer, "Stability Criteria for RED with Bulk-data TCP Traffic", 2001 Technical Report, August 1999.

48. T. Ziegler, S. Fdida, and C. Brandauer, "Stability Criteria for RED with TCP Traffic", Technical Report, May 2000.

49. T. Ziegler, S. Fdida et al., "Stability of RED with Two-way TCP Traffic", IEEE ICCCN, October 2000.

Publication

- **International Journal**

i. "Classification and Performance of AQM-Based Schemes for Congestion Avoidance", International Journal of Computer Science and Information Security, ISSN 1947-5500, Vol. 8 No. 1, pp. 331 – 340, April 2010. Impact Factor 0.423.