Robust Router Congestion Control Using Acceptance and Departure Rate Measures

Ganesh Gopalakrishnan[a], Sneha Kasera[b], Catherine Loader[c], and Xin Wang[b]

[a]{*ganeshg@microsoft.com*},
Microsoft Corporation, Redmond, WA 98052, USA

[b]{*kasera, xwang@research.bell-labs.com*},
Bell Labs Research, Holmdel, NJ 07733, USA

[c]{*catherine@cwru.edu*},
Case Western Reserve University, Cleveland, OH 44106, USA

In packet switched networks, routers need to implement controls to overcome the problems of reduced performance in the event of congestion. In this work, with a view towards robust implementation, we examine a new active queue management scheme, Acceptance and Departure Rate (ADR) that uses a combination of acceptance rate and departure rate measures to control link congestion. Unlike many existing approaches, ADR does not implicitly or explicitly use queue length measures which are not robust to changes in system capacity. The main advantage of our approach being its robustness with varying network parameters including link capacity and network load. We also compare the performance of our approach with other known active queue management schemes.

1. Introduction

A router output queue could suffer congestion conditions when it receives more traffic than its line capacity. Uncontrolled congestion could cause high queue build up resulting in increased queuing delays and packet loss. One potential solution is to allocate more resources, i.e., provide higher line capacities. Unfortunately this is an expensive solution, especially in access networks. A carefully designed congestion control mechanism is more cost-effective. It also provides opportunities for differential treatment of packets under congestion. Active queue management of router queues attempts to provide congestion control by monitoring the congestion state of a router queue and pro-actively dropping or "marking" packets before any impending congestion could cause reduced performance. Several schemes have been proposed for active queue management. These schemes have been modeled and evaluated for stability, throughput and delay performances. Our goal in this paper is not to propose a scheme that outperforms any of these existing schemes. Rather, we would like to examine a scheme that performs reasonably well compared to the existing scheme but we believe is more robust and scalable with varying system and network parameters including buffer size, link capacity and network load. In particular, we examine an congestion control approach that uses measures of acceptance rate in

conjunction with measures of departure rate (ADR) for computing the probability of proactively dropping packets. In our approach, the acceptance rate provides a measure of offered load and the departure rate a measure of processed load on the router queue. A combination of both offered load and processed load is essential in providing fast but stable response under different load conditions. Control based on processed load provides stability and helps clear queue backlog, whereas control based on offered load provides fast response.

In this paper, we first study the applicability of various load measures of a router queue and justify the choice of using acceptance rate and departure rate. Next, using simulations of a simple packet network, we compare the performance of ADR with other well-known schemes including Drop Tail, Random Early Discard, RED [1], and Adaptive Virtual Queue, AVQ [5]. We also examine the performance of ADR, AVQ with UDP traffic. Our simulation results demonstrate the robustness of the ADR scheme under different traffic conditions and system settings.

2. System Load Measure

One of the key requirement of any congestion control scheme is to obtain a robust measure of system load. The offered and processed load constitute the system load. The offered load is a measure of new load on a system whereas the processed load is what the system has already seen and processed. In an congestion control scheme, offered load measurement is necessary to quickly respond to a sudden burst of traffic but not sufficient. It is possible that the offered load in a given time is not very high but due to backlog from load offered at earlier instants the processed load can still be high. Hence a combination of both offered load and processed load should be used for robustly controlling congestion under different traffic arrival patterns. We now discuss the applicability of some of the well known system load measures.

- *Queue Length* - Queue length is one of the most widely used measure of load. It is used naturally in drop tail schemes where incoming packets are dropped when the queue reaches its capacity. It is also used in RED and several of its variants. At any instant, the queue length provides a measure of the load remaining in the system, i.e., the load arriving at the queue minus the processed load. Hence it seems the most natural measure of congestion. The problem with queue length is that it is not robust to changes in the system. If an AQM scheme sets target queue length(s), these targets have to be changed with changes in line capacity. A router operator might be willing to operate with longer queues when capacity increases. This issue also arises in situations where the *available* line capacity for best-effort traffic changes due to any co-existing bandwidth reservation mechanisms, especially in networks supporting multiple service classes. In this scenario, it is not possible to determine the available capacity a priori.

- *Delay* - Queuing delay offers a more robust measure. It is essentially a measure of the queue length normalized by the line capacity. Any queuing delay thresholds need not be changed with changes in line capacity. The problem with using queuing delay is an appropriate choice of an operational value. A typical network consists

of several hops (and at times there are multiple cards a packet has to go through per hop). The choice of an operating delay at each hop is difficult because a smaller value could result in under utilization and a high value could result in large delays. There is no simple way to budget an end-to-end delay requirement across hops or in some cases, even across multiple components in the same hop.

- *Departure Rate* - Departure rate is defined as number of bits transmitted or serviced in a given time interval. This rate provides a very good measure of processed load. Under steady state conditions, processed load could be a good measure of the system load. When there are sudden changes in the system load, just measuring processed load is not sufficient because the offered load might be much higher than the processed load. Congestion control based on departure rate only could result in slow response under a sudden burst of traffic. Line occupancy is departure rate normalized by capacity. Since this is dimensionless, controls based on this measure are robust against capacity changes.

- *Acceptance Rate* - Acceptance rate is the number of bits in packets arriving at the router queue minus the number of bits in dropped packets in a given time interval. It provides an accurate measure of offered load.

3. Related Work

Active Queue Management has been one of the most widely studied areas. Several approaches have been proposed in the literature [1], [2], [3], [5]. One of the most popular AQM approach is RED [1]. In RED, the system load is measured by keeping track of an average queue length which is updated on every packet arrival. This average queue length is compared against two pre-specified limits the minimum queue length and the maximum queue length and drop probabilities are computed based on the relative difference of the measured average queue length and these limits. One of the main problems with RED is how to choose its parameters [7]. As we have argued before, queue lengths are not robust against change of system parameters including line capacity.

The PI controller scheme proposed in [2] uses delays for controlling congestion. We believe that delays are hard to dimension across variable number of hops in a network.

The virtual queue (VQ) proposed by Kelly [3] and the adaptive virtual queue (AVQ) proposed by Kunniyur [5] are much more robust schemes. They use a dimensionless parameter to set targets on line occupancy. This allows for robustness against changing line capacity. Unfortunately, these schemes implicitly depend upon the buffer size because packet drop decisions are still made in a drop tail fashion when the virtual queue is full. How to compute the right buffer size is not defined in [3], [5]. In [5], Kunniyur has also proposed to overcome the problems of drop tail such as TCP synchronization and unfairness by using a RED like scheme in conjunction with AVQ. We believe that this enhancement only reduces the scale of the robustness problem but does not solve it.

4. The ADR algorithm

We now present the acceptance and departure rate (ADR) algorithm that is derived from the Acceptance Rate - Occupancy (ARO) algorithm proposed in [4]. ARO was devised and evaluated for controlling processor overload due to signaling traffic in telecommunication switches. In ADR, "A" represents the acceptance rate of packets i.e. the number of bits arriving at a router queue minus the bits of packets dropped per second and "DR" represents the departure rate (in bits per second) of the packets from the router normalized by the line capacity.

An AQM scheme could be event-based or timer-based. In an event-based scheme, system load is measured and updated on every packet arrival and a probability to drop or mark the packet is computed. In a router, such a scheme needs to be implemented fully in the fast forwarding path. In a timer-based scheme, system load is measured periodically, based on which a probability to mark or drop packets is computed. This probability is then used to mark or drop packets throughout the timer-interval. In a timer-based scheme the algorithm to compute the drop probability need not be executed in the fast path. Only the actual packet drop decision based on the drop probability need be taken in the fast path. This gives the freedom to choose different and even complex policies for determining the drop probability without modifying the packet forwarding hardware.

4.1. Timer-based ADR

We first present the timer-based ADR. This scheme is based on gradual throttling of input traffic to prevent oscillations. We use a time varying variable f - which is the *fraction of the offered traffic to be allowed* into the router queue. When $f = 1$, all the traffic is let in. When there is congestion, f is reduced till the system comes out of congestion. In timer-based ADR, the system load is measured at fixed intervals. In our scheme, the acceptance rate is measured every τ_{ar} time units and normalized by the line capacity, C. This normalized acceptance rate, α, is then compared with a normalized acceptance rate threshold, α_{peak}, and the ratio $\phi_{ar} = \frac{\alpha_{peak}}{\alpha}$ is computed. Similarly, the the departure rate is measured every τ_{dr} time units and normalized with the line capacity. The normalized departure rate, ρ is then compared with a normalized departure rate or line occupancy threshold, ρ_{thresh} and the ratio $\phi_{dr} = \frac{\rho_{thresh}}{\rho}$ is computed. Every time the acceptance rate or departure rate is measured, a fraction of traffic that can be allowed until the next measurement instant is computed by

$$f_{n+1} = [f_n \times min\{\phi_{ar}, \phi_{dr}, \phi_{max}\}]_{f_{min}}^{1} \quad (1)$$

where f_{n+1} represents the fraction of packets allowed in the (n+1)'th probe interval, ϕ_{max} is the upper bound on ϕ_{ar} and ϕ_{dr}, f_{min} is a predefined lower limit on the fraction allowed. An interval is defined as the time between two successive rate measurements, it can either be between two AR measurements or two DR measurements or one AR measurement and one DR measurement. Once f_{n+1} is calculated for the (n+1)'th interval, the same fraction allowed is maintained for all packets arriving in that time interval.

The intuition behind this Eqn 1 is as follows. When $\rho \leq \rho_{thresh}$, and $\alpha \leq \alpha_{peak}$ then there is no congestion and all the packets that arrive at the router queue are allowed in. If $\rho \geq \rho_{thresh}$ or $\alpha \geq \alpha_{peak}$, then the system is in congestion and a fraction of the packets arriving at the queue are dropped to get the system out of congestion. In order to prevent

oscillations in the fraction allowed, we calculate the fraction f_{n+1} as a percentage of the fraction allowed in the previous probe interval. The lower limit on this fraction is set to a predefined minimum value f_{min} and the upper limit is set to 1. ϕ_{max} is set to 20 to ensure that f_{n+1} does not increase drastically and cause oscillations.

Once the fraction to be allowed is determined, a throttling scheme decides which incoming packets should be dropped based on the calculated fraction allowed. We use the deterministic scheme first proposed by Hajek in [8] and later used by [4]. In [4], the authors have found this deterministic scheme to show much less variability from the desired fraction allowed in comparison to uniform drop and RED drop mechanisms proposed in [1]. The Hajek scheme could be described by the following procedure. The variable r is initialized to zero.

$r := r + f.$
If $r \geq 1$
    $r := r - 1$
    accept packet
else reject packet.

The above scheme can be implemented very easily with little overhead. The randomness in the input traffic ensures that no particular connection(s) or flow(s) is able to take undue advantage of the deterministic packet drops.

The choice of τ_{dr} and τ_{ar} influences the response to congestion. A small value of the timers would result in more measurement overhead and potentially over-reaction causing oscillatory behavior. High timer values would result in slower response. In addition to specifying the two timer values, the timer-based ADR also needs to set a limit on the maximum buffer size. This is because a very large burst can fill the buffers in a short time causing large delays and potential timeouts at TCP senders. In our timer-based ADR, we set the buffer size to simply $C * D_{max}$ where C is the line capacity and D_{max} is the specified maximum delay that the router queue could incur in the worst possible scenario. Packets arriving after all the buffers are filled are simply dropped following a drop tail policy. Note that using a maximum delay for setting the buffer size does not contradict our stand against choice of an operational delay for congestion control. *The maximum delay is only a safeguard for extreme scenarios and is not used for congestion control. We use only acceptance rate and departure rate measures for control.*

4.2. Event-based ADR

We now present the event-based ADR scheme. This scheme is very similar to timer-based ADR in the nature of the algorithm except in the event-based scheme, the system load is updated on every packet arrival. Upon each packet arrival, the router calculates ϕ_{ar} and ϕ_{dr}. It then updates the fraction to be allowed into the router queue by using Eqn 1. The subscript n in Eqn 1 now refers to the packet number and f_n is the fraction allowed computed on the arrival of the n'th packet.

Let t_d and t_a be the time between two consecutive departures and two consecutive arrivals respectively. At the arrival of n'th packet, normalized departure rate, ρ_n is estimated by

$$\rho_n = \rho_{n-1} \times \beta_d + \frac{(1 - \beta_d) \times PacketSize}{t_d \times C},$$

where $\beta_d = e^{-\frac{t_d}{K}}$. Similarly, the normalized acceptance rate is estimated by

$$\alpha_n = \alpha_{n-1} \times \beta_a + \frac{(1-\beta_a) \times PacketSize}{t_a \times C},$$

where $\beta_a = e^{-\frac{t_a}{K}}$. The justification for using the above equations for normalized rate estimation is described in [9].

On the arrival of the (n+1)'th packet, ϕ_{ar} and ϕ_{dr} are computed by setting, $\alpha = \alpha_{n+1}$ and $\rho = \rho_{n+1}$ respectively and Eqn 1 is used to calculate the fraction allowed. Once the fraction allowed is calculated, the deterministic drop algorithm described above determines whether a packet should be accepted or dropped.

One could choose between event or timer-based ADR depending upon their system requirements. We believe that timer-based scheme is more flexible. It also has less overhead making it specially attractive for systems where per packet measurements could be resource consuming, e.g., systems that use network processors or embedded software in processor cards, for implementing IP forwarding.

5. Performance Evaluation

In this section, we use *ns-2* packet simulator to simulate ADR and other AQM schemes. We only present timer-based ADR results. We first describe the performance metrics, then present tests to evaluate ADR performance under varying network parameters and also compare its performance with other AQM schemes. The main advantage of ADR is its robustness with varying system and network parameters including buffer size, link capacity and network load. We provide different simulation results, ascertaining this fact.

We use a single bottleneck network topology. Although simple, this setup helps us effectively evaluate the AQM schemes. The links can either mark or drop a packet according to the AQM scheme it implements. The propagation delay of the link is 100 msec unless specified otherwise. We have TCP sources on Node 0 and TCP sinks on Node 1. We use TCP New Reno as the transport protocol. In our experiments, packet size is 1000 bytes.

We perform comparisons of ADR with Drop Tail, RED and AVQ. We pick RED and AVQ, because RED has been implemented in commercially available routers and AVQ uses rate based control with the help of dimensionless parameters. AVQ has also been proven to be better than most of the other AQM schemes [5].

5.1. Performance Metrics

The performance metrics that we use are as follows:

- *Link Utilization* is the effective service rate of the link. It is the number of bits transmitted over the link during the simulation divided by the simulation time.

- *Session Goodput* is the effective rate at which information is correctly received by the destination from the source. It is calculated as total number of correctly received bits by the destination, divided by the time taken. This performance metric is useful for understanding the end-to-end behavior that takes into account the effect of lost packets, retransmissions, propagation & queuing delays.

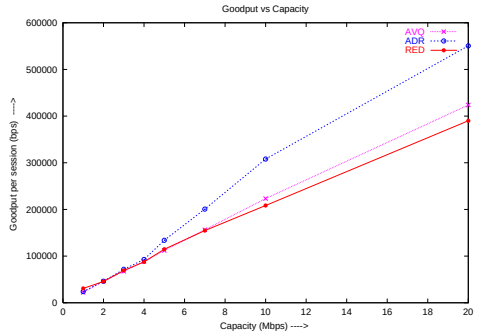

Figure 1. Goodput(y-axis) vs Capacity(x-axis) for ADR, RED and AVQ

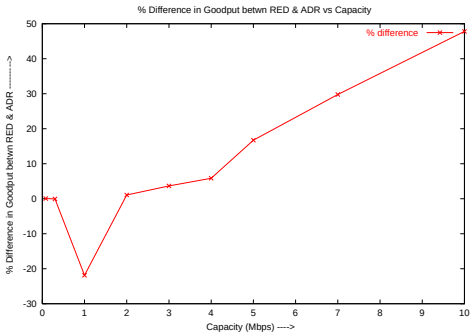

Figure 2. Percent difference in Avg Goodput of ADR and RED

- *Average Queuing Delay* - This is the delay arising due to the queuing and buffering of packets. This is calculated as the difference between the time a packet is served and the time the packet enters the queue.

- *Loss Fraction* - This is fraction of packets that were dropped by the link's queuing policy. This value is calculated as the ratio of the number of packets dropped to the number of packets that arrive at the queue.

5.2. Robustness against Capacity Variation

In this section, we study the performance of ADR and other AQM schemes as a function of link capacity. One of ADR's primary design objective is robustness against line capacity variations. The available line capacity for best-effort traffic could change due to any co-existing bandwidth reservation mechanisms. This section analyses the robustness of the AQM schemes under such scenarios.

The link capacity determines the rate at which the queued packets get processed. At higher capacities, since the queues get processed faster, we can allow longer queues and still maintain the same delay guarantees as in the case of lower capacity and smaller queues. The main advantage of allowing longer queues being increased link utilization.

A link's Buffer Size (B), Capacity (C) and maximum queuing delay D_{max} are related as $B = C \times D_{max}$. For e.g., a link with C=10 Mbps, with B=125 packets will have a D_{max}=100 msec. (Packet Size = 1000 bytes). The performance of ADR, RED and AVQ with respect to capacity is discussed in Sec 5.2.1 and Sec 5.2.2

5.2.1. ADR vs RED : Response to changing Link Capacity

We now conduct the following experiment to compare the performance of ADR and RED. We have 50 bursty FTP sources entering the system periodically at different times between $t = 0$ and $t = 20$ secs. ADR parameters: $\alpha_{peak} = 1.7$, $\rho_{thresh} = 1$, $D_{max} = 100$ msec. $\tau_{dr} = 50$ msec, $\tau_{ar} = 28$ msec. RED parameters (default values) : $Q_{min} = 5, Q_{max} = 15, max_p = 0.1, w_q = 0.002$. The buffer size (B) is set according to the relation $B = C \times D_{max}$. C is varied from 1 Mbps to 20 Mbps. Link propagation delay is set at 100 msec. The simulations are run for 100 secs. Inorder to study the robustness, ADR

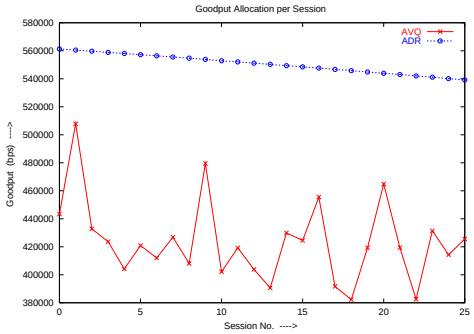

Figure 3. ADR & AVQ : Goodput distribution across Sessions

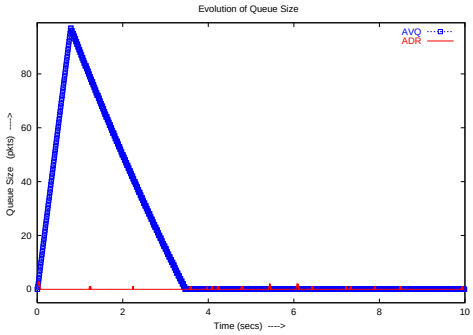

Figure 4. ADR & AVQ : Evolution of Queue Sizes under Steady congestion

and RED parameters are kept constant with link capacity.

Fig. 1 shows the average goodput as a function of capacity. We can see that the performance of RED deteriorates after C=2 Mbps, whereas the performance of ADR is consistently good. This is because, due to its dimensionless thresholds, ADR is able to better utilize the fact that the capacity has increased and allows for more queuing and increased throughput while not increasing the delay. This is better illustrated by Fig. 2. This graph plots the % difference in goodput values of ADR from RED [calculated as $(ADR_{util} - RED_{util})/RED_{util}$]. The positive values correspond to ADR performing better than RED and the negative values to RED performing better. We can see from the graph that RED performs well only for capacities around C=1 Mbps. This is because the RED parameters that were chosen for this experiment appear to be suitable for this value of capacity. For all other capacities, these parameters need to be changed in order to obtain better performance. This is not the case with ADR. Even with the same parameter setting, ADR achieves a higher average goodput for the different capacity values.

5.2.2. ADR vs AVQ : Response to changing Link Capacity

We now compare the performance of AVQ and ADR with varying link capacity. The experimental setup is same as in Sec 5.2.1. AVQ parameters are set to default values (with γ=1). We see from Fig. 1 that ADR performs better than AVQ. ADR has a slightly higher average queuing delay than AVQ, but we see in Fig. 1 that the slight increase in delay does not affect avg. session goodput much and ADR's goodput is still higher than AVQ's. We also notice that ADR has a higher link utilization and lower loss than AVQ.

5.3. Fairness

It is very important that any AQM scheme is fair in its bandwidth allocation across several sessions using the common link. A well known drawback of Drop Tail is the problem of *global synchronization & unfairness*. Detailed experimental results showing that ADR is very fair when compared to Drop Tail are available in [9]. We now compare ADR and AVQ on fairness. Since both AVQ and Drop Tail are queue overflow based, there might be unfairness in bandwidth allocation even in AVQ. A TCP session that enters and sees an empty queue may end up using a lot more than its fair share of bandwidth, just

because it came first. The following simulation clearly illustrates this observation.

We have 50 FTP sources entering periodically into the system between time t = 0 and time t = 20 secs. The link has a capacity C = 20 Mbps, B = 250 packets and propagation delay of 100 msec. We choose a higher value of capacity, to clearly show the behaviour of the schemes under extreme cases. ADR parameters: $\alpha_{peak} = 1.7$, $\rho_{thresh} = 1$, $\tau_{dr} = 50$ msec, $\tau_{ar} = 28$ msec. For AVQ, $\gamma = 1$, and the other parameters are set to their default values. $D_{max} = 100$ msec. The simulations are run for 100 secs.

The bandwidth allocation in case of AVQ and ADR is shown in Fig. 3. The graph plots the Goodput received by each session vs session number. We can see that ADR has a very uniform as well as higher allocation amongst the sessions when compared to AVQ. The peaks in allocated goodput seen in the case of AVQ are due to AVQ's drop tail approach.

5.4. Performance under Steady Congestion - UDP Sources

In todays internet, even though majority of the traffic is from TCP, there is still some *non-conforming* traffic from UDP or other sources. These non-elastic sources do not adjust their transmission rate to the network congestion level and hence might lead to a steady congestion in the network. If this happens, it becomes very important that the AQM scheme detects this and reacts accordingly to bring the system out of congestion. This section focuses on the performance of ADR and AVQ under steady congestion.

We conduct two experiments. In one experiment, we study the queue in the event of steady congestion and in the other experiment we analyze the performance with respect to the incoming traffic from multiple sources. In both experiments, C=1Mbps, B = 125 packets and propagation delay = 100 msec. ADR parameters: $\alpha_{peak} = 0.85$, $\rho_{thresh} = 0.9$, $\tau_{dr} = 50$ msec, $\tau_{ar} = 28$ msec. For AVQ, $\gamma = 0.9$, and the other parameters are set to their default values. $D_{max} = 100$ msec. The simulations are run for 100 secs. In the first experiment, we have one UDP Constant Bit Rate (CBR) source accessing the link, with constant rate = 2 Mbps (twice the link capacity, such that there is steady congestion). In the second experiment, we use more than one CBR source, and the sources enter the system at different times. Each CBR source has a constant rate of twice the link capacity.

In AVQ, packets are dropped only after the virtual buffer overflows. Also, virtual capacity ($\tilde{C}$) needs to be adapted according to the incoming traffic. Both these result in a sluggish response to congestion and hence increased delays. This alongwith the unfair nature of AVQ (Sec. 5.3), causes reduction of goodput and rejection of service to some sessions. ADR on the other hand, uses a fraction based drop mechanism and hence has a quicker response and well regulated queues.

In the first experiment, a CBR source that has a rate twice the link capacity, enters the system at time t = 0. We can observe from Fig. 4, that AVQ lets the queues build to a huge value before starting to react to congestion. Since arrival rate is greater than link capacity, ADR immediately recognises the congestion due to its AR control and starts throttling the traffic. This can be observed from smaller ADR queues of Fig. 4.

The next experiment demonstrates ADR's robustness with respect to the load on the system and how the traffic is distributed. Performance is studied for one CBR source, 2 CBR sources entering at t=0 and 15 sec, 3 CBR sources entering at t = 0, 5, 10 sec and for 3 CBR sources entering at t = 0, 15, 30 sec. Detailed results are available in a longer version of this paper. In the case of AVQ with two sources, only one of the sessions gets

served, where as both the sessions get served equally in the case of ADR. We also observe a similar behaviour in AVQ with 3 sessions spaced at 5 sec. Here only one session gets served. With 3 sessions separated by 15 sec, only two sessions get served. This unfairness in AVQ is noted for the case of C=10 Mbps also. Whereas with ADR, all the sessions get served equally for both C=1Mbps and C=10 Mbps. We also evaluated the above experiment for ρ_{thresh}, $\gamma = 0.7$ and 0.8 with different values of α_{peak} and observed that ADR is still very fair in its allocation whereas AVQ still shows the unfairness.

In [9] we discuss how to tune ADR and choose the parameter values. We also show that ADR's performance is independent of buffer size and how AVQ and RED's performance get affected by buffer size. A typical network is likely to experience sudden bursts of traffic. In [9], we present results to show that ADR reacts quickly to such bursts and regulates traffic on the link such that the link utilization and average goodput remain high without increasing the delay.

6. Conclusions and Future Work

In this paper, we proposed a robust AQM scheme, ADR, that uses a combination of acceptance and departure rate measures to control link congestion. Using simulations, we demonstrated the robustness of ADR with varying network parameters including link capacity and network load. Potential future extensions to our work include performance evaluation for complex network topologies and multi-class traffic scenarios. We also plan to study functions other than *min* (e.g. weight function) to reduce ADR's aggressiveness.

REFERENCES

1. S. Floyd and V. Jacobson, *Random Early Detection Gateways for Congestion Avoidance*. In IEEE/ACM Transactions on Networking, August 1993.
2. C. Hollot, V. Misra, D. Towsley and W. Gong, *On Designing Improved Controllers for AQM Routers Supporting TCP Flows*. In IEEE Infocom, April 2001.
3. R. Gibbens and F. Kelly, *Distributed Connection Acceptance Control for a Connectionless Network*. In Proceedings of the 16th International Teletraffic Congress, Edinburgh, Scotland, June 1999.
4. S.K. Kasera, J. Pinheiro, C. Loader, M. Karaul, A. Hari and T. LaPorta, *Fast and Robust Signaling Overload Control*. In Proceedings of IEEE ICNP, November 2001.
5. S. Kunniyur and R. Srikant, *Analysis and Design of an Adaptive Virtual Queue (AVQ) Algorithm for Active Queue Management*. In Proceedings of ACM Sigcomm, August 2001.
6. V. Misra, W. Gong and D. Towsley, *A Fluid-based Analysis of a Network of AQM Routers Supporting TCP Flows*. In Proceedings of ACM Sigcomm, August 2000.
7. M. May, J. Bolot, C. Diot and B. Lyles *Reasons not to deploy RED*.
8. B. Hajek *Extremal Splitting of point processes*. In Mathematics of Operations Research, 1985
9. G. Gopalakrishnan, S. Kasera *Robust Router Congestion Control* . www.comm.csl.uiuc.edu/ggopalak/ADR, 2003