

# LRU-RED: An active queue management scheme to contain high bandwidth flows at congested routers

Smitha A. L. Narasimha Reddy

Dept. of Elec. Engg., Texas A & M University, College Station, TX 77843-3128, reddy@ee.tamu.edu

**Abstract:** In this paper, we propose a queue management scheme that is based on partial state. It empowers the routers to contain high bandwidth flows at the time of congestion. The scheme maintains an LRU cache at the routers to record information about the high-bandwidth flows. This can be incorporated in RED, an active queue management scheme. The proposed scheme possesses all the advantages of RED. In addition, it lowers the drop rates of short-lived flows and also of responsive high bandwidth flows. It is shown, by means of simulations, that the method is effective in achieving the objective. The overhead involved is low and the operations incur  $O(1)$  cost per packet.

**Keywords:** High bandwidth flows, LRU, RED, active queue management

## I. BACKGROUND & MOTIVATION

Recently, there has been much interest in developing resource management techniques that can effectively control nonresponsive applications. It has been shown that nonresponsive applications can effectively claim most of the bandwidth at a network element while starving other applications that respond to congestion [1]. This has motivated a number of recent proposals at novel buffer management techniques that allow different drop rates for different flows.

The DropTail buffer management scheme drops packets when the buffer at the router is full. RED (Random Early Detection)[2], is an active queue management scheme that is accommodative to bursty traffic, and it does so by increasing the drop rate of non bursty traffic. LQD (Longest Queue Drop)[3], stores the number of buffers assigned to each individual flow and on congestion drops a packet from the flow that has the longest buffer length/queue. CHOKe [4], picks up a packet randomly from the queue when a packet arrives at the router and compares the two. If both the packets belong to the same flow, it drops both of them. If not, only the arriving packet may be dropped with the same probability as in RED. The complexity of CHOKe grows linearly with the number of unresponsive flows. SRED [5] estimates the number of active flows without collecting information on individual flows. LQD and CHOKe utilize buffer occupancy information in making drop decisions. SRED and LRU-RED employ similar amount of information in a more flexible way for recording longer-term behavior of selective flows than evident with buffer occupancy.

Current Internet traffic is heterogeneous. Most of the bytes, typically, are transferred by a small number of flows (like ftp) while a large number of flows (like HTTP) do not contribute much traffic in bytes [6]. In such an environment, flow based schemes tend to be inefficient as the work done to establish state may not be useful for most short-lived flows. RED and

CHOKe, even though increase drop rates for high bandwidth flows, do not work well as the number of high bandwidth flows increases. With the growing use of multimedia (audio & video) applications, it is expected that traffic due to unresponsive flows will increase in the future. Hence, it is important to find mechanisms that do not employ per-flow state, yet are effective in controlling several high bandwidth (unresponsive) flows at the router.

In this paper, we propose a simple method to identify high bandwidth flows at a network element. The proposed method is totally decoupled from the underlying buffer management scheme in the routers i.e., it can be employed irrespective of the kind of the buffer management scheme used. In addition, we propose a method to couple the above with RED to contain and penalize high bandwidth flows at the time of congestion. Our work focuses on aggregate performance of different kinds of flows (HTTP, TCP and UDP) in contrast with individual flow performance.

## II. OVERVIEW OF THE SCHEME

We consider various types of flows in this scheme viz., long-term high bandwidth flows (referred to as high-BW flows), short-lived flows, and low bandwidth flows. Flows that pump data at a rate that is greater than acceptable to the network (this is typically decided by the ISP) over a period of time are long-term high bandwidth flows. Those that pump bursts of data over a short period and stay idle for some period and continue this process are short-lived flows. The others are classified as low bandwidth flows simply because they do not violate the rate limit. Among the long-term high bandwidth flows we identify two classes, one that reduces its rate and starts sending data at a lower rate when congestion is indicated. The second class of applications are non-responsive to congestion. TCP flows are typical examples of the long-term high bandwidth flows that respond to congestion. UDP sources pumping data at high rates with no congestion control mechanism built into them can be classified as long-term high bandwidth flows that do not respond to congestion. HTTP transfers over the Internet can be classified as short-lived flows. UDP sources that send at a low rate and telnet type interactive applications can be classified as low bandwidth flows.

We propose a scheme that can be used by a router to recognise long-term high-BW flows and provide higher drop rates for them when compared to short-lived flows and low-BW flows. Also, our scheme can distinguish between high-BW flows that

respond to congestion and those that do not, in order to give them different drop rates. By doing so, we propose to be able to give short-lived flows and responsive flows, higher throughput. There are two components involved in this process (a) identifying high-BW flows accurately and (b) penalising the identified high-BW non-responsive flows aggressively.

### A. Identifying high bandwidth flows

Packets from high-BW flows will be seen at the router more often than other flows. Short-lived flows, that are characterized by HTTP transfers are typically the ON-OFF type, send data intermittently. Thus, packets from such flows are not seen at a constant rate at the router. When they are seen, the data is much less than that of high bandwidth flows. So, by observing the arrivals of packets for a period of time, the router can distinguish between high-BW and low-BW flows.

In order to identify high-BW flows at the router, we employ an LRU (Least Recently Used) cache. This cache is of a fixed pre-determined size, 'S'. In an LRU cache every new entry is placed at the topmost (front) position in the cache. The entry that was the least recently used is at the bottom. This is chosen to be replaced when a new entry has to be added and there is not enough space in the cache. This mechanism ensures that the recently used entries remain in the cache. The objective is to store state information for only long-term high bandwidth flows in the LRU cache.

With a cache of limited size, a flow has to arrive at the router frequently enough to remain in the cache. Short-term flows or low-BW flows are likely to be replaced by other flows fairly soon. These flows do not pump packets fast enough to keep their cache entries at the top of the LRU list and hence become candidates for replacement. High-BW flows are expected to retain their entries in the LRU cache for long periods of time.

Every time the router sees a packet, it searches the cache to check if that flow's entry exists in the cache. If yes (no), we say that a *hit* (*miss*) for that flow has occurred. On a miss, the flow is added to the cache if there is space in the cache. If there is no space in the cache, it replaces the least recently seen entry (the bottom most in the cache) with a probability 'p'. It adds this entry in the topmost position in the cache. On a hit, the router updates the position of the entry in the cache (brings it to the topmost position). The scheme employed in SRED based on "zombie list" is similar to this approach, but it does not work well in the presence of many short-lived http flows.

When there is no space in the LRU cache, the oldest flow is replaced with a certain probability to make room for the new flow. This reduces the chances of recording short-lived, low-BW flows in the cache and saves space for flows that are actually high-BW in nature. Packet sizes can be taken into account in determining the probability with which a flow is admitted into the cache. In order to keep the discussion simple, in the rest of the paper, we consider packets of same size.

To allow for burstiness of flows, we employ a 'threshold' below which a flow is not considered high bandwidth, even if its entry is in the cache. For each flow in the LRU, we keep track

of its 'packet count' seen at the router. This count is updated on each packet arrival. Only when this count exceeds the 'threshold', a flow is regarded as a high-BW flow. Short-term flows and low-BW flows are likely to be replaced from the cache before they accumulate a count of 'threshold'. RED-PD [7] uses RED packet drop history to identify high-BW flows.

The LRU is implemented as a doubly linked list. Each node contains an entry for the flow id and the packet count. In order to make the search into the linked list easy, it is indexed by a hash table.

### B. Penalizing high bandwidth flows

After having identified the high bandwidth flows at the router, drop probabilities of flows are increased after the flows accumulate a count exceeding the 'threshold' parameter. Once flows accumulate a count greater than this and remain in the cache, they will be dropped at a higher rate until they drop the rate sufficiently enough to be thrown out of the cache. By doing this, we are able to increase the drop probability of high bandwidth flows compared to the other flows. This 'policy enforcement' mechanism can be coupled with any buffer management scheme. Below, we explain how this can be done with RED.

### C. LRU Coupled with RED

The scheme is incorporated in RED, so it preserves all the properties of RED. The scheme modifies RED's drop probabilities using the information in the LRU cache.

When the queue length builds up, and it is in the region between minth and maxth, RED calculates the drop probability of a packet to be dropped. It is here that we bring the distinction of the high-BW and the low-BW flows. In this region, we increase the drop probability of flows that are high bandwidth in nature. This is explained in figure 1.

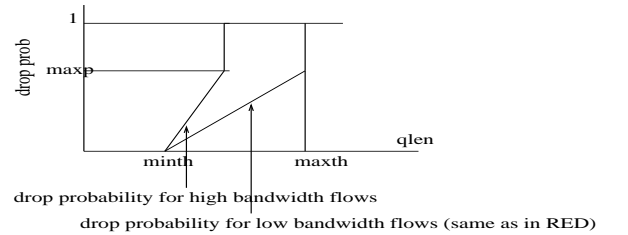


Fig. 1. Drop Probability of flows in the proposed scheme

We define a target rate  $1/\delta$  above which identified flows will be penalized. We record information about the rate at which the flow is sending packets by incrementing 'count' according to the following equation:

$$count = count + 2 - (currenttime - timestamp)/\delta \quad (1)$$

The RHS of the above is derived based on the following:

$$(\delta - (currenttime - timestamp))/\delta \quad (2)$$

Equation 2 gives how much faster the flow is sending its packets compared to our defined rate ' $1/\delta$ '. If the flow is sending at a slower rate, then  $(currenttime - timestamp)$  would be

greater than  $\delta$  and the above term would be negative. This would amount to a decrease of 'count'. If the (currenttime - timestamp) is less than  $\delta$ , the flow is sending at a rate faster than acceptable, so the above term would be positive, resulting in an increase of 'count'. Also, the above takes care of the varying rates of flows. A flow that sends at a higher rate has its 'count' incremented at a higher rate.

The scheme is incorporated in RED. Flows that are not high bandwidth in nature experience a drop probability that is similar to that in ordinary RED. But flows that are classified as high bandwidth flows, have their drop probability scaled by a factor that is proportional to the observed 'count' and 'threshold' as follows:

$$p_{lru} = \text{count}/\text{threshold} \quad (3)$$

$$p_{drop} = p_{red} * p_{lru} \quad (4)$$

When there is no congestion in the network, or when  $p_{red}$  is zero, the factor  $p_{lru}$  does not contribute anything to the drop probability, so the scheme behaves exactly like ordinary RED. Again, it is noted that packet sizes can be taken into account by employing a "byte mode" RED instead of "packet mode" RED.

If a flow is not a high-BW flow, it would not have an entry in the cache and therefore it would observe drop rates similar to what it would in ordinary RED. But if the flow is a high-BW flow, its entry would be found in the cache and it would observe drop rates that are scaled and much higher than those of the low-BW flows. By giving high-BW flows greater drop rates, we propose to be able to keep the drop rates low for the low-BW flows and give them higher throughput.

Also, an increase in the drop probability of the high-BW flow may cause a packet of that flow to be dropped. If this is a responsive high-BW flow, (like some TCP flows), then on a packet drop, which it discerns as an indication of congestion, it would drop the rate at which it is sending. If this happens, the router would see fewer packets from this flow. So, 'count' for that entry in the cache will be updated slowly when compared to what it was earlier. Also, because the router sees packets from this flow less often, it may get to the bottom of the cache and might even get thrown out of the cache. Once this happens, it would no longer be classified as a high-BW flow and would observe drop rates similar to what it would in ordinary RED. If a high-BW flow does not respond to congestion (like many UDP flows) and does not reduce its sending rate on a packet drop, the router would continue to see many more packets from this flow. It would continuously figure in the top positions in the cache and its 'count' would be updated often. This would result in a greater drop probability for these flows.

#### D. Cost Analysis

The LRU when implemented as a doubly linked list, insertion and deletion of a flow takes  $O(1)$  time. Searching for a flow in the linked list would take linear time if it were a simple doubly linked list. A hash table is used to make the search  $O(1)$ . Every time a new flow is added to the LRU, a hash table entry is made corresponding to this so that a search would take  $O(1)$  time. The memory cost is proportional to the size 'S' of the cache.

### III. SIMULATION RESULTS

NS-2 [8] was used to simulate the network conditions in our experiments. The topology used for these is a typical dumbbell one with routers R1 and R2 between a bottleneck link with a BW of 40Mb and a link delay of 2ms. The sources and sinks are connected to R1 and R2 by means of links that have 10Mb and a delay of 32ms (unless otherwise mentioned). HTTP traffic was generated randomly between the link R1 and R2 using TCP flows on both sides. RED parameters of minthresh =  $1/4 * \text{buffer size}$  at R1, maxthresh =  $3/4 * \text{buffer size}$  at R1, maxp = 0.1 and queue weight = 0.002 were used. The flows pumped packets of size 1000 bytes.

#### A. Cache Occupancy

An experiment consisting of 20 TCP, 20 UDP and 300 HTTP flows was conducted to observe the cache occupancies of the flows. It is evident from figure 2 that the LRU cache (of size 30) was able to hold the UDP flows for a longer period (500 seconds -length of the simulation) than the TCP flows (less than a second).

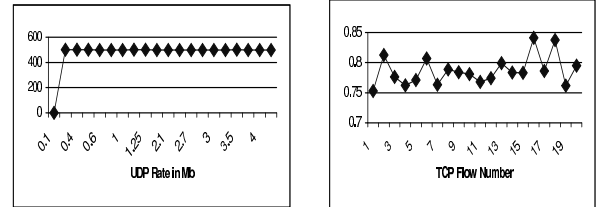


Fig. 2. Cache Occupancy

#### B. Effect of varying cache size

The following experiments study the effectiveness of the proposed scheme with different LRU cache sizes. The number of TCP and UDP flows were 20 each. UDP sources were pumping data at full link capacity(40Mb). The probability was set to  $1/40$ , threshold to 135 and interval to 4ms, i.e., flows pumping data at a rate greater than 0.5Mb were branded high-BW flows, and were penalised.

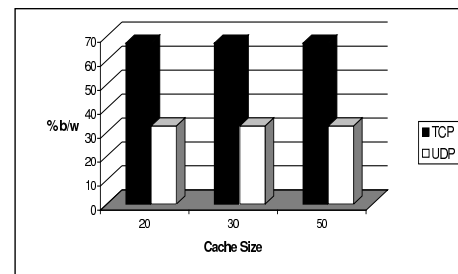


Fig. 3. Effect of varying cache size

#### C. Effect of varying threshold

The parameter 'threshold' decides the limit beyond which we start penalising a cached flow. A flow can send data and accumulate count to 'threshold -  $\theta$ ', where  $\theta$  is a very small number, and still get away unpunished if it is able to get out of the cache. The larger the value of 'threshold', the burstier the flows can be

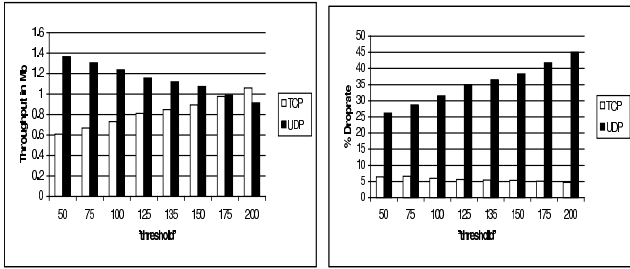


Fig. 4. Effect of varying threshold

without getting penalized. We expect larger values of 'threshold' to benefit TCP flows and HTTP flows. The setup for the experiment had 20 UDP and TCP flows each, with a probability of 1/40, LRU cache size of 30 and an interval of 8ms. The rest of the parameters were unchanged. Figure 4 shows the results of the experiments when the 'threshold' was varied from 50 to 175. With a smaller 'threshold', there is a greater chance of the TCP flows being penalised before they get replaced. Also, it is worthy to mention that despite the fact that TCP flows get penalised, their drop rate is much less compared to those of the UDP flows. HTTP flows obtain zero drop rates in all the cases.

#### D. Effect of varying interval

The following set of experiments clearly show that we are able to control the drop rates of flows that are not high bandwidth by using 'interval'. The experiment had 20 UDP and TCP flows, probability 1/40, LRU cache size of 30 and 'threshold' 135. Figure 5 shows that with intervals of 4ms and 8ms, flows above our target rates of 1Mb and 0.5Mb respectively, experienced higher droprates.

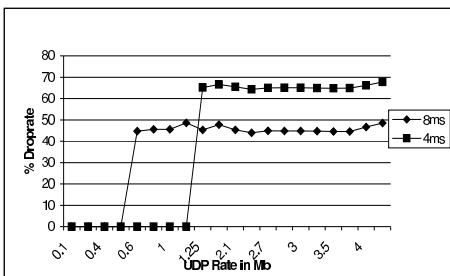


Fig. 5. Effect of Varying Interval - UDP Droprate

#### E. Impact of multiple congested links

This experiment was conducted to observe the effect of multiple congested links. The topology for this is similar to the previous one, in addition another bottleneck link is added between routers R2 and R3. The buffer sizes at the routers was 160 and 120 at R1 and R2 respectively. The bottleneck bandwidths and the delays were 40Mb, 2ms and 30Mb, 2ms. The LRU cache size was 30, threshold was 135, probability was 1/40 and the interval was 8ms at both the routers. This meant that flows that were pumping data at or lower than 0.5Mb are considered low-bandwidth flows. There were 40 UDP flows, pumping at the

link capacity of 40Mb, 20 TCP flows, and about 300 HTTP flows. When the number of flows exceeds the amount of state we have in the cache, (in this case, the total number of long-lived flows was 60 and the LRU size was 30) the cache is unable to capture all the long-lived flows. As a result, some of them escape punishment. These flows could eventually be captured and punished at routers further downstream. If a high bandwidth flow escapes one router, it is recognised at another router and is penalised there.

Figure 6 shows the effects of having an increasing number of UDP applications in the system. RED, LQD, CHOKe, Drop-tail and the LRU scheme are analyzed here. With 20, 40, 60 and 80 UDP flows pumping at the bottleneck link capacity of 40Mb, the LRU scheme does consistently better than the other schemes considered here. As the number of UDP flows increased from 20 to 80, the aggregate rate at which they were pumping was kept the same, viz., 40Mb. This resulted in an increasing number of UDP flows that pumped data at smaller rates per UDP flow as we move from 20 to 80. The interval being 8ms, the goal was to protect all the flows that were pumping at the rate of 0.5Mb. When the number of UDP flows was 20, most UDP flows were sending at a rate greater than 1Mb. So most of the UDP flows were penalised which resulted in TCP obtaining high throughput. As the number of UDP flows increased, the number of possible candidates for high bandwidth flows decreased and fewer of them were punished. This resulted in a lower throughput for TCP flows. Figure 6 shows the effect on short-lived HTTP flows. We only consider drop rates here, because talking about throughput for these flows does not exactly describe the effectiveness of the scheme. As is evident from the figure, HTTP flows observe low drop rates when compared to the other schemes discussed here.

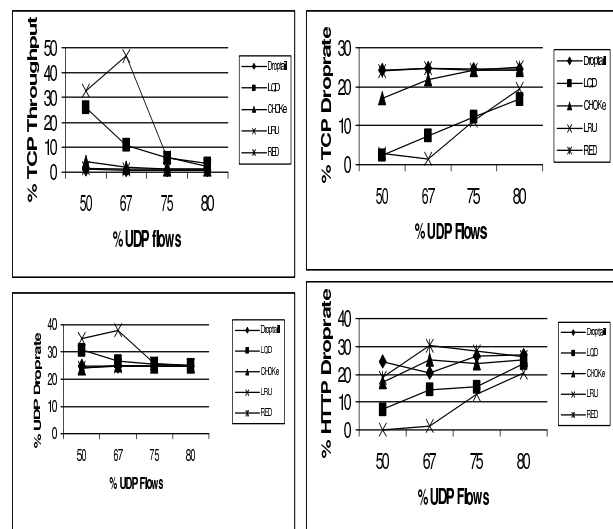


Fig. 6. Unresponsive high bandwidth flows and Multiple Congested Links

### F. Effect of Varying Load on the bottleneck link

In these set of experiments, we address the cases of lightly loaded and heavily loaded bottleneck link. The experiments consisted of varying the number of UDP flows and the amount of data they pumped into the network. For the cases where the load was less than 100% of the bottleneck link bandwidth, viz., for the 25%, 50% and the 75% cases, there were 10 UDP sources each pumping data at the rate of 10Mb, 20Mb and 30Mb together. There were 20 UDP flows pumping data at 40Mb for the 100% case, and 60 UDP flows pumping data at 60Mb for the 150% case. The number of TCP sources was 20, and HTTP flows was 300. All other parameters remained the same. Figure 7 shows the results of the simulations for TCP flows. It is evident that the proposed scheme does better than the rest in all the cases. We are able to give smaller drop rates to TCP flows and penalise the high bandwidth unresponsive flows. Figure 7 also shows the results for UDP flows. The drop rates for the UDP flows in cases when the link is underutilized is zero, showing that the scheme does not affect the utilization adversely. Also this shows that the dropping functionality is employed only when there is congestion and not otherwise. Once the load goes above 100%, the drop rates increase. As is evident from this, we perform better than the schemes considered.

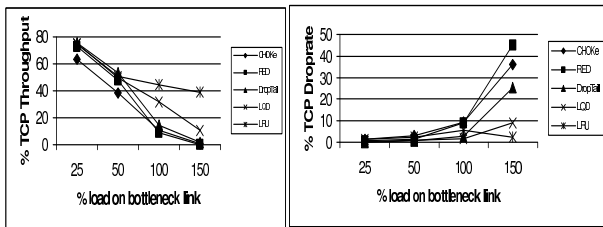


Fig. 7. Effect of varying load on the bottleneck link

### G. Reducing RTT bias

In this experiment, the RTTs of various TCP flows was varied to study the new scheme when flows have different RTTs. A setup consisting of mainly TCP flows was used. There were 18 TCP sources with 6 different RTTs. The bottleneck link capacity was reduced to 20Mb, the rest of the parameters remained the same. There were no UDP flows. The cache size was 30. Figure 8 shows the results of the experiment. The LRU scheme was able to give TCP flows with short RTTs, greater drop rates compared to TCP flows with longer RTTs. RED and CHOKe, give similar drop rates to all the flows. LQD, though it is supposed to reduce the RTT bias, the difference in the drop rates is not significant.

### H. Sampling

The present scheme does  $O(1)$  work for every packet that arrives at the router. To make the scheme more efficient, sampling could be employed. This scheme samples arriving packets with a probability  $q$ . Since the LRU-RED scheme now looks at fewer packets, 'threshold', 'probability' and ' $\delta$ ' are adjusted by a factor  $1/q$ . The work done per packet is  $q$  times that of the previous value. We show the results when  $q = 0.5$ , i.e., when half the packets are sampled, in figure 9. The results obtained are

almost similar to when all the packets are passed through the LRU-RED scheme.

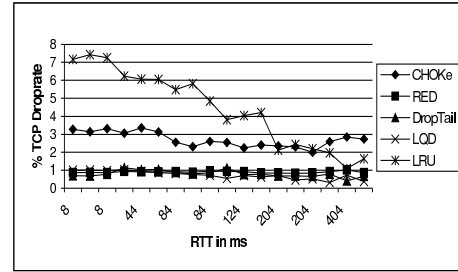


Fig. 8. Effect of TCP flows with varying RTTs

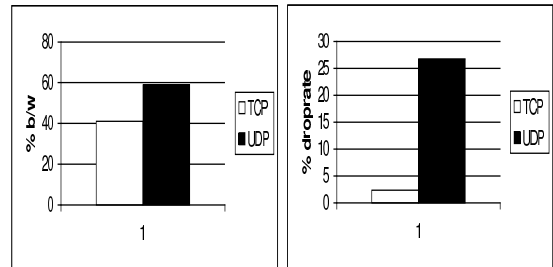


Fig. 9. Effect of Sampling

## IV. CONCLUSION

We have proposed a mechanism that could be incorporated in routers to (a) identify high bandwidth flows, (b) penalize the high bandwidth flows at the time of congestion and (c) protect responsive, short-lived and low bandwidth flows. The identification of high bandwidth flows uses a simple LRU cache and this is independent of the underlying queue management scheme. We proposed LRU-RED, merging the identification approach with RED to show the effectiveness of the scheme. The proposed scheme is accommodative of bursty TCP traffic in contrast to regular RED. Simulation results show that the scheme is successful in achieving all of the design goals. In addition, it is able to give higher drop rates for TCP flows with smaller RTTs compared to those with larger RTTs. The LRU-RED packet handling cost remains  $O(1)$  and that the memory cost is proportional to the size of the LRU cache.

## REFERENCES

- [1] S.Floyd and K.Fall, *Promoting the use of end-to-end congestion control in the internet*, IEEE-ACM Transactions on Networking, pp. 458-472, August 1999.
- [2] S.Floyd and V.Jacobson, *Random early detection gateways for congestion avoidance*, IEEE-ACM Transactions on Networking, pp. 397-413, August 1993.
- [3] B. Suter, T.V. Lakshman, D. Stiliadis and A K. Choudhary, *Design Considerations for supporting TCP with per-flow queuing*, INFOCOMM'98.
- [4] Rong Pan, Balaji Prabhakar and Konstantinos Psounis. *CHOKe, A Stateless Active Queue Management Scheme for Approximating Fair Bandwidth Allocation* in IEEE INFOCOMM, March 2000.
- [5] T.J. Ott, T.V.Lakshman, and L.H. Wong, *SRED: Stabilized RED* in Proceedings of IEEE INFOCOM, pp. 1346-1355, March 1999.
- [6] Inko Kim and A.L.N Reddy, *Analyzing Network Traces to Identify Long-Term High-Rate Flows* in Technical Report, TAMU.
- [7] Ratul Mahajan and Sally Floyd *Controlling High-Bandwidth Flows at the Congested Router* in ACIRI, November 20, 2000.
- [8] S. Floyd, *NS network simulator*, www.isi.edu/nsnam.