ANALYZING THE ACCURACY OF CHOKE HITS, CHOKE MISSES AND CHOKE-RED DROPS

Visvasuresh Victor Govindaswamy
Texas A&M-Texarkana,
Texarkana, Texas
victor.govindaswamy@tamut.edu

Gergely Záruba
University of Texas at Arlington,
Arlington, Texas
zaruba@uta.edu

G. Balasekaran
Nanyang Technology University,
Singapore,
gbalas@nie.edu.sg

ABSTRACT

CHOKe, xCHOKe and RECHOKe are preferential dropping schemes that have been proposed for detection, control and punishment of malicious flows at routers in IP networks. They use CHOKe hits, CHOKe misses and/or CHOKe-RED drops to carry out these tasks. In this paper we investigate the accuracy of malicious flow detection by using these hits, misses and drops (using ns-2). We also point out the unreliability of CHOKe hits and misses, when compared to CHOKe-RED drops, as they affect TCP-friendly flows adversely. By doing so, we present two variations of CHOKe called Half1 and Half2 to improve CHOKe and compare them with CHOKe. Half1 and Half2 outperform CHOKe when the combined rates of malicious flows are less or greater than the link capacity respectively.

Index Terms— TCP Congestion Control, Congestion Avoidance, Active Queue Management (AQM), Buffer Management, Random Early Detection (RED)

1. INTRODUCTION

Transmission Control Protocol (TCP) is the connection oriented transport layer protocol of the Internet. One of the services it provides is the adjustment of flow transmission rates whenever congestion is encountered. Such congestion may be caused at the routers by too many sources trying to send an excessive amount of data with a rate too high for the network to handle. This adjustment of rates is based on the feedback from these routers. To provide better feedback to the TCP sending processes (to enable them to detect and react to congestions earlier) various queue management schemes have been proposed. However, most of these schemes do not work well with non-responsive or malicious flows, allowing these flows to dominate the available bandwidth, choking and stealing bandwidth that was being used by the responsive flows. Hence, to protect TCP-friendly flows from non-adaptive sources such as UDP and non-TCP friendly sources at routers in IP networks, several preferential dropping schemes such as CHOKe [1], xCHOKe [2], RECHOKe [3] and RED-PD [4] have been proposed.

CHOKe, xCHOKe and RECHOKe have been proposed as an extension to RED queue management algorithm [5] although they can be used with any queue management schemes. Hence, as in RED, two thresholds are defined Th_{min} and Th_{max} with $Th_{min} < Th_{max} < Buffer\ Size$. If the average queue length (avq) is smaller than Th_{min} then, in all 3 schemes, the arriving segment is accepted. If avq is between Th_{min} and Th_{max}, CHOKe compares a new arriving packet with a randomly selected packet from queue. If they are from the same flow (referred to as a CHOKE hit), they are dropped; otherwise the new packet is allowed to enter the queue (referred to as a CHOKE miss). In xCHOKe, these hits are entered as CHOKe hit history in a table. This history is checked whether the arriving packet's flow label is already in it. If it is (referred to as a table hit), the arriving packet is dropped or marked for dropping with a probability p^*. After this step the packet is compared with a randomly selected packet from the queue. If this results in a hit, then both packets are dropped or marked for dropping and the flow label is added to the CHOKe hit history (associated with a hit counter of one). If the flow is already in the table (referred to as a table hit), the associated hit counter is incremented. The hit counter of flows in the CHOKe hit history is used to compute the probability p^*. The CHOKe hit history is refreshed periodically (e.g., every t ms) if there are no table or CHOKe hits. In RECHOKe, both CHOKe hit and CHOKe-RED drop/mark histories are kept in a table which is checked whether the arriving packet's flow label is already in it. If it is, the arriving packet is dropped or marked for dropping with a probability p^* and boolean value c*, and the associated hit counter is incremented. After this step the packet is compared with a randomly selected packet from the queue. If this results in a hit, the flow label

CCECE/CCGEI May 5-7 2008 Niagara Falls. Canada
978-1-4244-1643-1/08/$25.00

is added to the table (associated with a hit counter of one). If the flow is already in the table, the associated hit counter is incremented. Unlike xCHOKe, the packets are not dropped or marked for dropping as a result of a CHOKe hit. They are allowed to enter the RED FIFO buffer. In RECHOKe, just like in CHOKe and xCHOKe, after CHOKe misses and if the *avq* is between Th_{min} and Th_{max}, then the segment is dropped or marked (in the case of ECN[5]) with probability P_a (a linear function of *avq*). If the calculated *avq* is larger than Th_{max}, the incoming segments are dropped. However, in RECHOKe, these drops and marks are entered in the table and referred to as CHOKe-RED drop/mark history. This table is refreshed periodically (e.g., every t ms) if there are no table or CHOKe hits, or CHOKe-RED drops/marks.

In xCHOKe, CHOKe hits are entered as CHOKe hit history in a table and it uses it, while RECHOKe uses both CHOKe hit and CHOKe-RED drop/mark histories, to detect and thwart malicious flows. Since these schemes are dependent on the accuracy of CHOKe hits, CHOKe misses and CHOKe-RED drops/marks, we analyze all three of them using ns-2 [7], verify their accuracy and provide suggestions. We present two ad hoc variations of CHOKe called Half1 and Half2 CHOKes based on our analysis to show that CHOKe should adapt to the rate of flows. Both, like CHOKe, use a single selection of queued packet for comparison with each new packet arrival.

Due to space constraints, for a detailed description and more simulation results and analysis of RED, CHOKe, xCHOKe and RECHOKe, the reader is advised to refer to [3]. Section II and III provide evaluation of CHOKe and experiments on CHOKe-RED drop respectively. Half1 and Half2 CHOKes are covered in section IV and the paper concludes in section V.

2. EVALUATION OF CHOKE

To analyze the CHOKe buffer, we define the following terms. *CHOKe victim*: the packet that is randomly selected from the FIFO buffer to be compared with each arriving packet. A *CHOKe hit* occurs when the CHOKe victim's flow matches that of the arriving packet; otherwise a CHOKe miss has occurred. A CHOKe hit can be either a good or bad; it is good if the arriving packet belongs to a malicious flow (and hence, the CHOKe victim is also from that malicious flow); it is bad otherwise. A CHOKe miss can also be either good or bad. *Good CHOKe miss* occurs when the arriving packet belongs to a TCP friendly flow and the CHOKe victim is from any other but that particular flow; otherwise we talk about a bad CHOKe miss.

Using ns-2, we simulate a bar-bell topology and analyze the effects of buffer occupancy by a malicious flow (a constant-bit-rate UDP) competing with ten TCP flows over a 1-Mbps link. All TCP flows have the same round trip propagation delay of 20ms with each output link having a latency of 1 ms and capacity of 10 Mbps. The parameters for the RED are: $Th_{min} = 10$, $Th_{max} = 50$, $P_{max} = 0.1$.

We run 11 sets of simulations, starting with a 0.5 Mbps UDP flow rate, increasing it with 1 Mbps increments. Each simulation was long enough for initial transients to settle and be insignificant. If simulation results do not have a time axis, then we have run enough simulation instances to claim a 95% confidence that our results are no more than 5% off.

Fig. 1 shows the percentages of CHOKe hits and misses. We can observe that when the UDP flow is less than three times the link capacity, the percentages of misses is about 5-40% greater than the percentages of hits. However, they drop rapidly from 68% to 53% when the UDP flow rate is increased from 0.5 Mbps to 2 Mbps. Increasing the rate of the UDP flow further only decreases the percentages of misses to about 3-7%. As for hits, its percentages increase rapidly from 31% at 0.5 Mbps to about 47% at 2 Mbps. After 2 Mbps, the increase hovers around 1-8% and improves only about 1-6% over that of the misses. This increase is small compared to that of the UDP rate. The increase in its rate should imply that more UDP packets should occupy the buffer enabling more hits and less misses. However, from Fig. 1, this was not the case.

Fig. 1 presents an interesting phenomenon. To investigate this, we divide the packets within the buffer into 4 quarters, where Qtr1 is at the head of the queue. We call these quarters as Qtr1, Qtr2, Qtr3 and Qtr4. We measure the percentages of average buffer occupancy of the UDP flow at each quarter during CHOKe hits for every 10s of the 30s simulation. These are shown in Figs. 2 and 3 respectively. The third 10s period is similar to that of the second and hence, is left out of this discussion. High UDP buffer occupancy percentages at the tail of the queue are expected since the UDP packets arrive at the tail section of the buffer. However, this results in choking the TCP flows. The UDP flows advance in the queue and are tapered off slowly due to the effects of CHOKe hits. The head of the queue has the least UDP buffer occupancy percentages showing that CHOKe has been successful in reducing the number of packets in the buffer. Although there are some reductions before the first 10 seconds, the average UDP occupancy in Qtr1 is between 22 and 58% as the UDP rate is varied. After the first 12 seconds, increasing the duration of the flows has no big effect on the outcome. The slow reduction in Qtr1 reflects the reduction in the bandwidth of the UDP flow but the unresponsiveness in the other quarters in the reduction is due to bad hits (Fig. 4) and bad misses (Fig. 5).

Although the number of bad hits, compared to those of good hits, is small, it has an adverse effect on the throughput of the TCP flows since it forces the TCP sources into reducing their rates. At 0.5 Mbps, the number of good hits is 14 times greater than those of bad hits. As the UDP rate is increased, the number of UDP packets arriving and occupying the buffer increases. This leads to an increased number of good hits. Meanwhile, the number of bad misses,

in comparison to those of good misses is large. This leads to a greater number of UDP packets occupying and leaving the buffer in comparison to instances when there are successful good hits. Bad misses lead to increases in the throughput of the UDP flow. Hence, to improve the performance of CHOKe we need to *reduce drastically the number of bad hits and misses*. This would lead to more reduction in the throughput of the UDP flow.

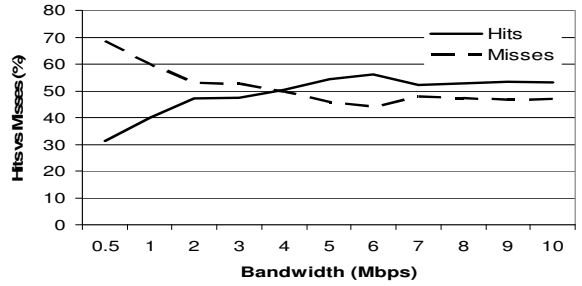

Figure 1. Hits vs Misses.

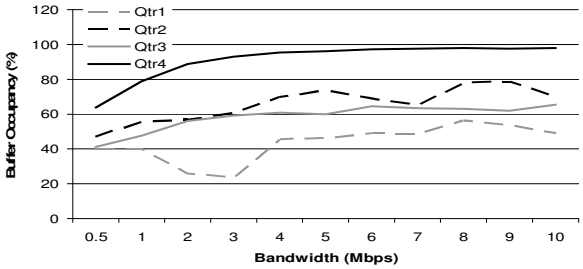

Figure 2. The percentages of average buffer occupancy by the UDP flow during CHOKe hits (0-10s).

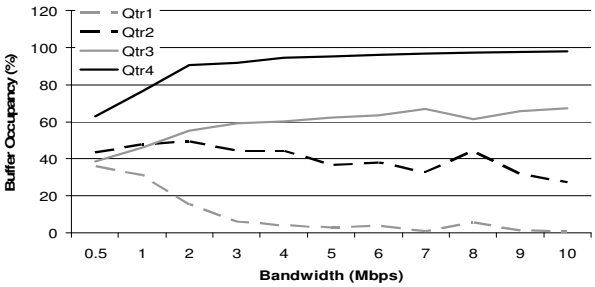

Figure 3. The percentages of average buffer occupancy by the UDP flow during CHOKe hits (10-20s).

Fig. 5 shows CHOKe's ratio of bad misses over good misses. At 0.5 Mbps UDP rate the ratio is at 1.4 implying that the number of bad misses is only slightly higher than those of good misses. However this ratio increases rapidly with a growing UDP transmission rate, implying that arriving UDP packets are increasingly compared with queued TCP packets.

Fig. 6 shows the average UDP buffer occupancy percentages during CHOKe misses. Qtr4's average UDP occupancy percentages increase from 46% to 82% (compare to the 63% to 98% during CHOKe hits in Fig. 2). Instead of the average UDP occupancy percentages increasing in Qtr3 from 41% to 70% as is in the case with CHOKe hits, it decreases after it has initially increased. In fact, as expected, all 4 quarters show reduced average UDP buffer occupancy during CHOKe misses when compare to during CHOKe hits. Of all the quarters, Qtr4 reduces the least. This is mainly because most of the random packets are chosen from the first 3 quarters as seen from Figs. 6 and 8, leading to the large number of misses. As a result, UDP packets are increasingly allowed to enter the buffer.

From Figs. 2, 3 and 6, the chances of achieving good hits improves tremendously if victims are chosen from Qtr4. The reason is that Qtr4 has the most number of UDP packets during hits and misses.

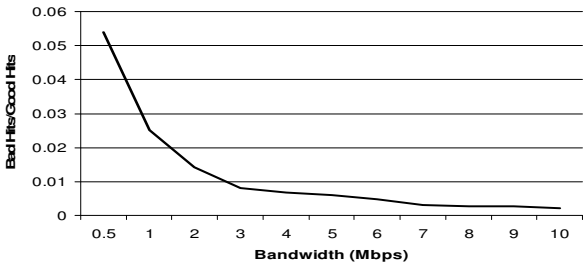

Figure 4. Ratio of Bad and Good Hits against UDP's bandwidth.

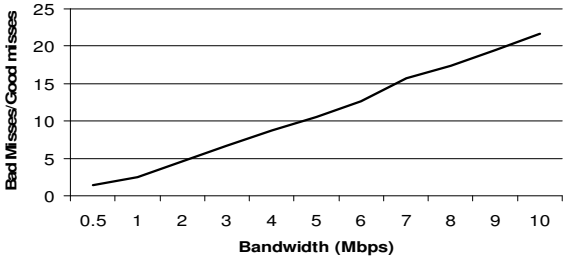

Figure 5. Ratio of Bad Misses over Good Misses.

Figs. 7 and 8 show the percentages of location of victims within each quarter; when the UDP flow's rate is changed, during CHOKe hits and misses respectively. Fig. 7 clearly shows that victims chosen from Qtr3 and Qtr4 increasingly matched the arriving packets as the rate of the UDP flow increased. The rate of increase, however, was more in Qtr4 than Qtr3 since more UDP packets occupied the region (Fig. 3). Next we analyze these CHOKe hits to distinguish between good and bad hits (Figs. 9 and 10 respectively). Fig. 9 shows more good CHOKe hits for Qtr4 and Qtr3 as the rate is increased due to increasing occupancy of Qtr4 and Qtr3 by UDP packets. On the other hand, good CHOKe hits decline for Qtr2 and Qtr1 since there is a decreasing occupancy of Qtr2 and Qtr1 by UDP packets as the UDP rate increases. In Fig. 10, Qtr4 suffers more bad CHOKe hits when the UDP flow is at or around the link capacity. As a result, the TCP flows suffered losses leading to low throughput resulting in a reduction of the number of

bad CHOKe hits at the other quarters. As the UDP rate increased, the number of bad CHOKe hits in Qtr4 has reduced due to the large increase in UDP packets.

In Fig. 7, the victim hits are very small for Qtr1 since this quarter has the lowest number of UDP packets among the 4 regions (Figs. 2 and 3). Conversely, in Fig. 8, the victim misses are very high for Qtr1 since this quarter has the lowest UDP packets among the 4 regions. However, we need to analyze these victim misses again to check whether they are good or bad CHOKe misses (Figs. 11 and 12 respectively).

In Fig. 12, the percentages of bad CHOKe misses were directly proportional to the UDP rate for Qtr1 and Qtr2 while inversely proportional to the UDP rate for Qtr3 and Qtr4. This was mainly because of the concentration of UDP packets in Qtr3 and Qtr4 (Figs. 2 and 3).

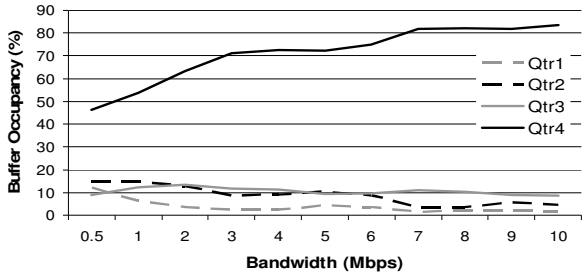

Figure 6. Percentages of average buffer occupancy by the UDP at each quarter during CHOKe Misses for the entire simulation.

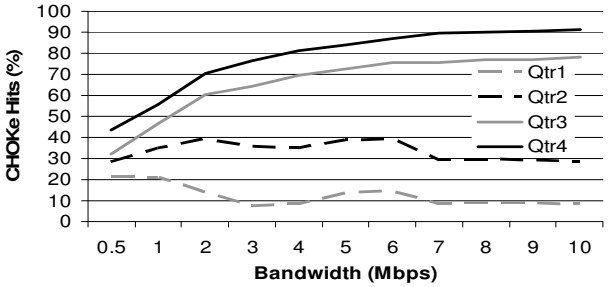

Figure 7. Percentages of CHOKe Hit locations.

Thus, the number of bad CHOKe hits increases i) when the number of flows using the buffer decreases, ii) when the rates of the TCP-friendly flows using the buffer increases, and iii) when the rates of the malicious flows using the buffer decreases. On the other hand, the number of good CHOKe hits increases i) when the number of flows using the buffer decreases, ii) when the rates of the TCP friendly flows using the buffer decreases and iii) the rates of the malicious flows using the buffer increases.

In Fig. 11, the percentages of good CHOKe misses are inversely proportional to the UDP rate. As the UDP rate is increased, the percentages of good CHOKe misses decrease. The percentages of good CHOKe misses for all the quarters are large at and around the link capacity since the number of packets from different flows using the buffer are greater at low UDP rates.

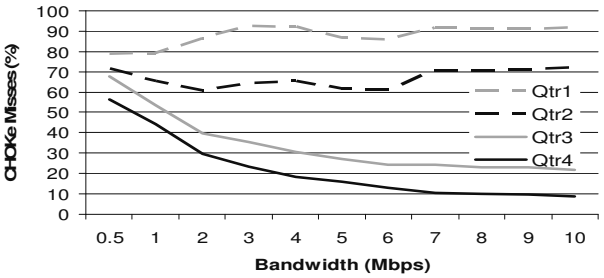

Figure 8. Percentages of CHOKe Miss locations.

The number of bad CHOKe misses increases i) when the number of flows using the buffer increases, ii) when the rates of the TCP friendly flows using the buffer increases, and iii) when the rates of the malicious flows using the buffer decreases. On the other hand, the number of good CHOKe misses increases i) when the number of flows using the buffer increases, ii) when the rates of the TCP friendly flows using the buffer decreases, and iii) when the rates of the malicious flows using the buffer increases.

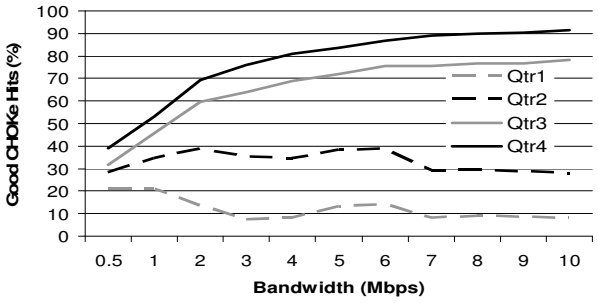

Figure 9. Percentages of Good CHOKe Hit locations.

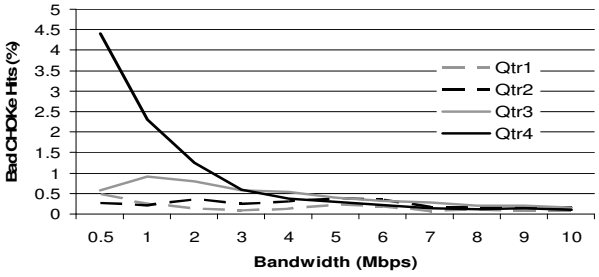

Figure 10. Percentages of Bad CHOKe Hit locations.

Until now, our simulations made use of a single UDP flow. However, we have carried out many simulations with more UDP flows as well. We have found that when the number of UDP flows is increased, the amount of buffer space occupied by each one of them is dependent on that flow's rate. The greater the rate of a UDP flow, the greater the buffer space it will occupy at Qtr4. For example, if there

are two UDP flows (at 2 and 10 Mbps) the packets from the UDP flow of 10 Mbps will occupy more buffer space in Qtr4 than those from the 2Mbps UDP flow.

3. EXPERIMENTS ON CHOKE-RED DROPS

CHOKe victims that escape being dropped (Fig. 1 - those involved in CHOKe misses) are received by the RED queue which either accepts or drops (marks in the case of ECN) them. To analyze CHOKe with RED, we define the following terms for RED drops. A CHOKe miss victim is a packet which after experiencing a CHOKe miss enters the RED queue. *CHOKe miss RED hit* occurs when the flow id of the CHOKe miss victim matches the flow whose packets dominate the buffer at the time of selection of the CHOKe miss victim; otherwise a *CHOKe miss RED miss* has occurred. A CHOKe miss RED hit could be either good or bad. *Good CHOKE miss RED hit*: during a CHOKe miss RED hit, the CHOKe miss victim belongs to a malicious flow; it is bad otherwise. *CHOKe miss RED miss* again could be either good or bad; it is good if during a CHOKe miss RED miss, the CHOKe miss victim belongs to a non-malicious flow; it is bad otherwise.

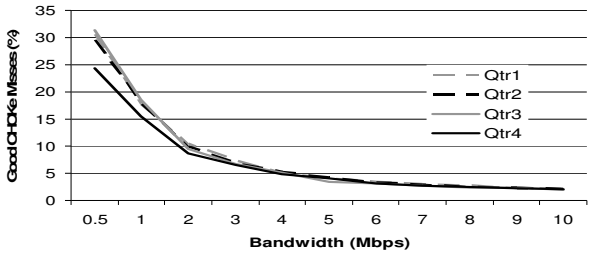

Figure 11. Percentages of Good CHOKe Miss locations.

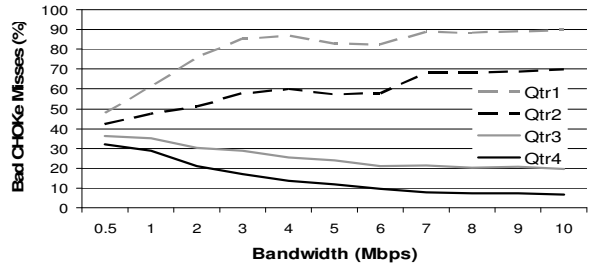

Figure 12. Percentage of Bad CHOKe Miss locations.

Fig. 13 shows that of the packets that are dropped by the RED buffer, a large majority experience RED hits. Almost all RED hits are good CHOKE miss RED hits, thus *RED was accurate in identifying malicious flows from CHOKe misses.* It seems that at lower rates, there is a large number of CHOKe misses. However, these are very few since at these rates, the queue drops or marks few packets. We also find that all CHOKe miss RED misses, although very few in numbers were all bad. Hence, RED also acted as another filter, after CHOKe, to weed out the packets belonging to these malicious flows. Although CHOKe-RED drops/marks are more accurate than CHOKe hits and misses, it occurred less often since in RED, a packet is either accepted or dropped whenever *avq* is between Th_{min} and Th_{max}. Moreover, only one packet is dropped while in CHOKe, two packets are dropped for each hit. Hence, in the next section, we aim to improve on the accuracy of CHOKe hits and misses.

4. HALF1 AND HALF2 CHOKES

From the experiments above, we see that if UDP flow rates are high, and if a victim packet is chosen from Qtr3 and Qtr4 (or Half-2), it will improve the chances of having i) a good CHOKe hit if the arriving packet belongs to a malicious flow, or ii) a good CHOKe miss if the arriving packet belongs to a TCP-friendly flow, while decreasing the chances of iii) bad CHOKe hits if the arriving packet belongs to a malicious flow, or iv) bad CHOKe misses if the arriving packet belonged to a TCP friendly flow. However, for UDP rates at or below the link capacity, the victim packet should be chosen from Qtr1 and Qtr2 (or Half-1). Although at low rates, Half-2 has slightly greater percentages in good CHOKe hits than at Half-1, the latter gives better protection to non-malicious flows (Fig. 10). Here we show by experiments that at lower rates, Good CHOKe hits are more effective in Half-1 than at Half-2. Hence, we propose two versions of CHOKe. The first, Half2 CHOKe, chooses victim packets from Half-2 whereas the second, Half1 CHOKe, chooses from Half-1. In this paper, we keep these two variations apart to demonstrate their effectiveness. They can be combined together by keeping a table of CHOKe hit history and sampling at certain intervals, say at time t=250ms, the first quarter of the buffer with the entries in the table. If the occupancy percentage at the first quarter of the occupied queue by flows with entries in the table during CHOKe hits is greater than 30%, then use Half1 CHOKe, otherwise use Half2 CHOKe. From Figs. 2 and 3, UDP packets occupancy of Qtr1 is consistently greater than 30% when the UDP rate is at and less than the link capacity. This held true whenever the summation of the UDP rates was at and less than the link capacity.

Using the same network topology as before, simulations using ns-2 are carried out to compare Half1, Half2 and CHOKe. Figs. 14 and 15 illustrated two examples when the UDP rate is set to a rate greater than the link capacity. Here, Half2 CHOKe performs better than both Half1 CHOKe and the normal CHOKe algorithm. Even when the number of UDP flows is increased, as long as the sum of their bandwidth is greater than to the link capacity, the Half2 CHOKe version outperforms the other two. Many other experiments (that we have performed but are omitted here) confirm this result.

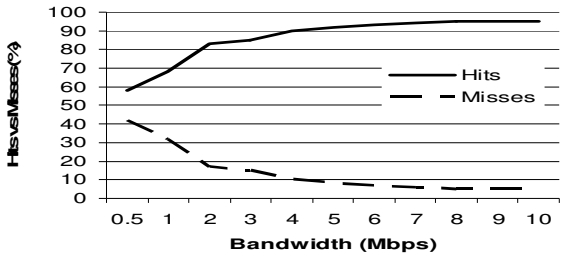

Figure 13. Percentages of CHOKe-RED Hits and Misses after CHOKe Misses.

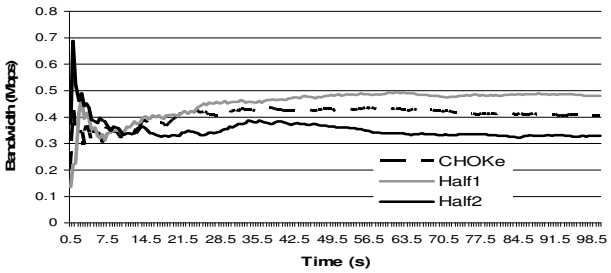

Figure 14. Link Utilization of UDP flow at 2 Mbps.

Fig. 16 illustrates an example when the sum of UDP rates is greater than the link capacity. Half2 CHOKe performs better than both Half1 CHOKe and the normal CHOKe algorithm. To illustrate an example when the sum of UDP rates is less than the link capacity we use three UDP flows with a rate of 0.3 Mbps each. Fig. 17 shows that Half1 CHOKe performs better than both Half2 CHOKe and the normal CHOKe algorithm in this case. When the number of UDP flows increases, as long as the sum of their rates is less than or equal to the link capacity, the Half1 CHOKe outperforms the other two. The results show the effects of CHOKe hits are far greater at Half1 than at Half2.

5. CONCLUSION

In this paper, we analyzed the accuracy of CHOKe hits and misses, and CHOKe-RED drops/marks. We found that the presence of 1) CHOKE hits, in the form of bad CHOKe Hits, were unreliable in that they affected non-malicious flows adversely by dropping their packets, 2) CHOKE misses, in the form of bad CHOKe misses, were unreliable in that they allowed UDP packets to steal more bandwidth from TCP flows. To reduce these problems, we presented two variations of CHOKe called Half1 and Half2 to improve CHOKe and compared their performance with that of the normal CHOKe algorithm.

We are currently working on research to implement Half1 and Half2 CHOKes in Linux based packet routers and by using various flavors of TCP with multi-bottleneck topologies. We are also working on a mathematical model for both these schemes.

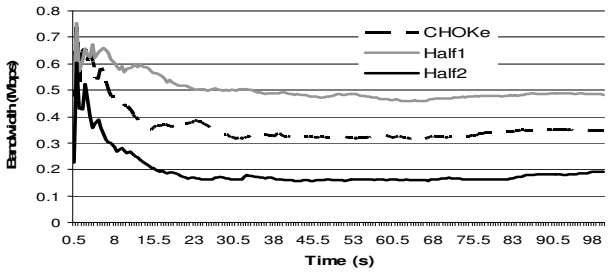

Figure 15. Link utilization of UDP flow at 10 Mbps.

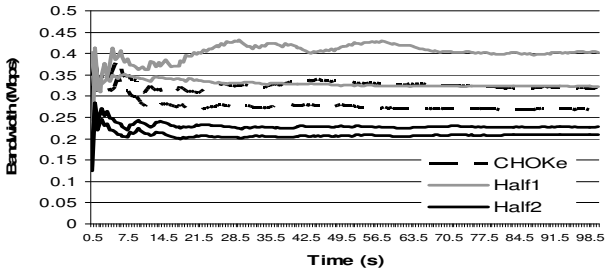

Figure 16. Link utilization of 2 UDP flows at 1 Mbps.

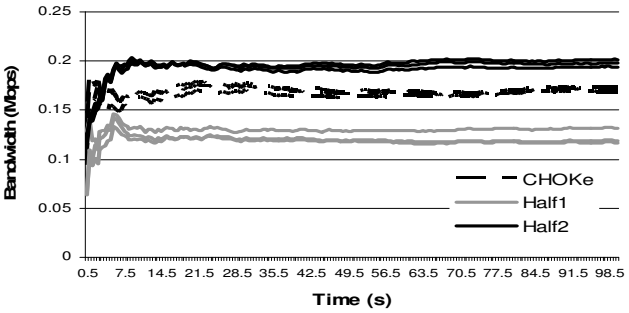

Figure 17. Link utilization of 3 UDP flows at 0.3 Mbps.

6. REFERENCES

[1] R. Pan, B. Prabhakar, and K. Psounis, "CHOKe: a stateless active queue management scheme for approximating fair bandwidth allocation," *Proc. of IEEE INFOCOM*, Tel Aviv, Israel, March 2000, vol. 2, pp. 942–951.

[2] P. Chhabra, A. John, H. Saran, and R. Shorey. "Controlling Malicious Sources at Internet Gateways," *IEEE Int. Conf. Communication*, Anchorage, Alaska, May 2003, vol. 3, pp. 1636-1640.

[3] V. V. Govindaswamy, G Záruba and G. Balasekaran, "RECHOKe and RCUBE," *UTA Technical Report 2006-7*, September, 2006.

[4] R. Mahajan, S. Floyd and D. Wetherall, "Controlling high-bandwidth flows at the congested router," *9th Int. Conf. Network Protocols*, Nov. 2001, pp. 192-201.

[5] S. Floyd and V. Jacobson, "Random early detection gateways for congestion avoidance," *IEEE/ACM Trans. Networking*, August, 1993, vol. 1, no. 4, pp. 397-413.

[6] K. Ramakrishnan, S. Floyd and D. Black, "The Addition of Explicit Congestion Notification (ECN) to IP," *RFC 3168*, September, 2001.

[7] NS2 simulator. Available: http://www.isi.edu/nsnam/ns/.