WPI-CS-TR-01-20 October 2001

Adaptive Explicit Congestion Notification (AECN) Techniques for Heterogeneous TCP Flows

by

Zici Zheng
Robert E. Kinicki

Computer Science
Technical Report
Series

WORCESTER POLYTECHNIC INSTITUTE

Computer Science Department
100 Institute Road, Worcester, Massachusetts 01609-2280

Adaptive Explicit Congestion Notification (AECN) Techniques for Heterogeneous TCP Flows

Zici Zheng and Robert Kinicki

Worcester Polytechnic Institute
Computer Science Department
Worcester, MA 01609

Abstract. This report investigates techniques that improve the performance of Explicit Congestion Notification for networks with a large number of heterogeneous TCP flows. Four different strategies for adaptive queue management are considered for improving goodput and fairness in a network. Basic AECN divides flows passing through a bottleneck router into flow groups, uses a mark-front strategy per flow group queue, and deploys a different *max_p* for each flow group. A series of AECN simulations were run using ns-2. The simulations show that AECN treats each flow fairer than ECN using two fairness measurements: Jain's fairness index and visual max-min fairness. By using a higher mark probability at higher flow levels, AECN yields fewer packet drops and higher goodput than ECN. Two of the variants on AECN, selectively lowering the minimum marking threshold for robust flows and increasing the maximum mark threshold when there are many flows, are shown to further enhance AECN's ability to provide better fairness and higher goodput.

1 Introduction

TCP is the dominant transport protocol used in the Internet today. While Internet traffic continues to grow, it becomes more challenging to provide good network performance to millions of Web customers. When a packet is dropped, all of the resources consumed by the packet during its traversal through the network will have been wasted. In extreme cases, this situation can lead to congestion collapse [20]. In the past decade, TCP congestion control mechanisms have been used in controlling packet loss and preventing congestion collapse across the Internet. Several variants of TCP (Tahoe, Vegas, Reno and NewReno) [14] have been developed to provide host-centric mechanisms to improve performance during heavy congestion periods. Traditionally, a router simply drops a packet when the queue for an outgoing link overflows. However, this FIFO drop-tail mechanism fails to provide adequate early congestion notification and tends to produce bursts of packet drops that contribute to unfair service. Although TCP Reno includes techniques such as Fast Retransmit and Fast Recovery [7] to react to the congestion signaled by packet drops, these mechanisms are not designed for applications that are sensitive to network delay caused by the loss of one or more packets [32]. The growth of these type of applications on the Web has motivated numerous studies that consider enhancements to TCP that provide better congestion avoidance techniques.

Active queue management (AQM) at the router has been proposed [7] as a new philosophy for reducing packet losses due to buffer overflow and for providing lower delay services. The idea behind active queue management is to detect incipient congestion early and convey congestion notification to the end hosts. The goal is to provide timely notification to sources whereby the effect of sources backing off reaches the congested router before a burst of packet drops occur.

Random Early Detection (RED), an AQM technique proposed by Floyd and Jacobson [12], maintains an exponentially weighted average queue length, *ave_q*, to detect congestion. When *ave_q* exceeds the minimum threshold, *min_th*, arriving packets are probabilistically dropped. When the average queue length exceeds the maximum threshold, *max_th*, all packets are dropped. The original RED proposal also permitted marking via an Explicit Congestion Notification (ECN) bit in a packet instead of dropping the packet when the average queue size is between the minimum and the maximum threshold. The probability that a packet arriving at a RED router is either dropped or marked depends on the average queue length, the time elapsed since the last packet was dropped, and a maximum dropping probability parameter *max_p*.

ECN, an end-to-end congestion avoidance mechanism proposed by Floyd [13], has already been incorporated into RFC2481 [32]. Upon receving congestion notification in a returning ACK packet, an ECN-enabled TCP source halves its congestion window and reduces its slow-start threshold [7], [32] just as if a packet loss had been detected.

Since the introduction of RED, many researchers [1], [9], [10], [11] have proposed a variety of enhancements to router management to improve performance. However, most of the previous research on ECN and RED has focused on networks with a small number of homogeneous flows. Kinicki and Zheng [23] investigate the behavior and performance of RED with ECN congestion control mechanisms with many heterogeneous TCP Reno flows using the network simulation tool, *ns-2*. By comparing the simulated performance of RED and ECN routers, they find that ECN does provide better goodput and fairness than RED for heterogeneous flows. However, when the demand is held constant, the number of flows generating the demand has a negative effect on performance. Meanwhile, the simulations with many flows demonstrate that the bottleneck router's marking probability must be aggressively increased to provide good ECN performance.

This investigation builds on these recent results to experiment with adaptive variations of ECN. We conduct simulation experiments on four Adaptive ECN (AECN) mechanisms. The results show that three of these approaches can be used within an AQM framwork to improve goodput and fairness for ECN routers. Section 2 reviews pertinent previous work and defines performance metrics. The basic AECN mechanism is presented in section 3. This is followed by a discussion of the simulation topology and experimental procedures in section 4. Section 5 presents some preliminary comparisons of RED and ECN that set the stage for the AECN experimental results in section 6. Section 7 shows the results of three further refinements to AECN, and the paper finishes with conclusions in section 8.

2 Background

2.1 Previous Work

Lin and Morris [25] define fragile TCP flows as those eminating from sources with either large round-trip delays or small send window sizes and robust TCP flows as having either short round-trip delays or large send windows. This delineation emphasizes a flow's ability to react to indications of both increased and decreased congestion at the bottleneck router. Their research indicates that RED is not fair when the router traffic includes both robust and fragile flows.

Floyd's original ECN paper [13] shows the advantages of ECN over RED using both LAN and WAN scenarios with a small number of flows. Christiansen et al [8] use a LAN testbed to emulate a large number of web clients accessing a web server through a RED router. They show that RED is difficult to tune for throughput and delay, and they conclude that *min_th* is the most significant RED parameter for performance tuning. Although they do consider flows with a large variation in round-trip time (RTT), they do not consider fairness in their analysis.

Bagal et al [3] compare the behavior of RED, ECN and a TCP rate-based control mechanism using traffic scenarios that include 10 heterogeneous flows. They conclude that RED and ECN provide unfair treatment when faced with either variances due to the RTTs of the heterogeneous flows or variances in flow drop probabilities. Focusing on a window advertising scheme (GWA), Gerla et al [19] compare GWA, RED, and ECN in scenarios with up to 100 concurrent flows. Using the gap between maximum and minimum goodput as a fairness measurement, they show that ECN yields better fairness than RED for homogeneous flows. Ahmed and Salim [1] use Jain's fairness to compare ECN and RED performance for a small number of flows. Their results emphasize that *max_p* can significantly effect performance. Liu and Jain [26] demonstrate that adding a mark front strategy to ECN provides a better congestion control mechanism. Mark front ECN is shown to reduce the buffer size requirement at the router, provide higher link efficiency and better fairness among a small number of heterogenous flows with different RTTs.

2.2 Definitions

The performance metrics used in this investigation include delay, throughput, goodput and two ways to evaluate fairness. The delay is the time in transit from source to destination and includes queuing time at the router. Goodput differs from throughput in that it does not include retransmitted packets in the count of packets successfully arriving at the receiver. Given a set of throughputs

$$(x_1, x_2, \ldots, x_n)$$

Jain's fairness index [21] is defined in terms of the following function:

$$f(x_1, x_2, \ldots, x_n) = \frac{(\sum_{i=1}^{n} x_i)^2}{n \sum_{i=1}^{n} (x_i)^2}$$

A second form of fairness, referred to as visual max-min fairness, focuses on the gap between the maximum and minimum average goodput for groups of heterogeneous flows [19], [30].

3 Adaptive Explicit Congestion Notification (AECN)

The basic idea of providing improved fairness with an adaptive version of ECN comes from the fact that robust flows are quicker to adapt to available resources at a congested router than more fragile flows. AECN divides all flows into three flow group (fragile, average and robust), based on the instantaneous round-trip time of each flow. AECN maintains a flow queue for each flow group. A flow queue is a FIFO queue that stores a pointer to a packet in the router queue for each packet in the flow group. The relationship between the AECN router queue and the three flow queues is shown in Figure 1.

AECN assumes that the TCP header has been modified to include space for an RTT source hint. This source hint can either take the form of the actual instantaneous round-trip time (RTT) of a flow or a few bits that indicate the flow group of an arriving packet. In the former case, a TCP source inserts the RTT of the last ACKed packet of this flow into the outgoing packet. In the latter case, the source converts the RTT into a flow group indicator (e.g, three bits for eight flow groups) that requires less packet overhead. The difference between these two mechanisms involves whether the source or the router makes the flow group determination. Since this choice does not effect our simulation of AECN, for this report we assume that the RTT is included in the packet for flow classification by the router. The implications of this will be briefly discussed in the conclusion section of this report.

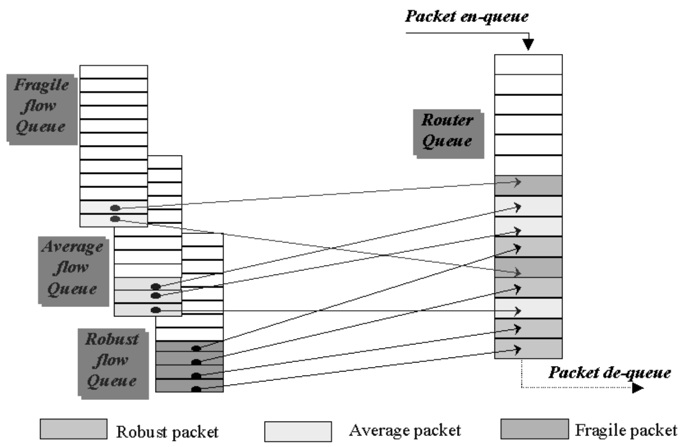

Fig. 1: AECN Router with Three Flow Queues and a Router Queue

Each router will maintain a set of RTT ranges to determine in which flow queue to place an arriving packet. Because the more robust flows have a natural advantage over other flows, AECN attempts to provide fairer treatment by having distinct maximum marking probabilities for each flow group.

AECN uses *base_max_p* for the average flow group which corresponds to *max_p* for RED and ECN. By making the marking probability for the fragile flow group smaller than *base_max_p* and making the marking probability for the robust flow group larger than *base_max_p*, AECN provides a proactive mechanism to improve fairness for heterogeneous flows.

For this preliminary version of AECN, these adjustments in marking probabilities are controlled by two constants α and β. Hence, the fragile marking probability is set to (*base_max_p* / α) and the robust marking probability is set to (*base_max_p* * β). The role of these three probability parameters is the same as *max_p* for ECN except they control the marking process for packets in an individual flow queue.

3.1 AECN Algorithm

The basic AECN router performs the following three operations:

1. Initialization

The router queue and the three flow queues are initialized as empty.

2. Enqueue a packet arriving at the router.

If *avg_q* is greater than or equal to *max_th* drop the incoming packet.
else
  Get the RTT from the incoming packet to determine packet_flow_type.
    If (RTT is in robust RTT range) packet_flow_type = ROBUST;
    else if (RTT is in average RTT range) packet_flow_type = AVERAGE;
    else if (RTT is in fragile RTT range) packet_flow_type = FRAGILE;
  Add the packet with packet_flow_type to the router queue.
  Add an entry for this packet to tail of the appropriate flow queue
  If *avg_q* is between *min_th* and *max_th*
    Determine maximum marking probability (*max_p*),
      if (packet_flow_type == ROBUST) *max_p* = min{ (*base_max_p* * β), 1};
      else if (packet_flow_type == AVERAGE) *max_p* = *base_max_p*;
      else *max_p* = *base_max_p* / α;
    Use RED drop/mark function with *max_p* to determine whether a packet should be marked
    if mark == TRUE
      Using packet_flow_type find the first unmarked entry starting from the front of the corresponding flow queue.
      Mark the flow queue entry and mark the packet this entry points to in the router queue.
    endif
  end if
endif

3. Dequeue the front packet from the router queue

Get the packet_flow_type from the departed packet.
If (packet_flow_type == ROBUST) Check to see that the first node in the robust queue points to the departed packet and remove this node from the robust flow queue.
else if (packet_flow_type == AVERAGE) Check to see that the first node in the average queue points to the departed packet and remove this node from the average flow queue.
else Check to see that the first node in the fragile queue points to the departed packet and remove this node from the fragile flow queue.
Remove the packet from the router queue.

The computation of round-trip time used in the *ns-2* simulations of AECN uses the round-trip time calculations from TCP Vegas [2], [5], [6] which was already available in ns-2.

3.2 AECN with Mark Front Strategy

Liu and Jain [26] make a strong case for using a mark front strategy with ECN instead of the standard mark tail strategy. Given that a major of goal of AECN is improve fairness among heterogeneous flows, mark front is included in the basic AECN technique.

4 Simulation Topology and Experimental Methods

This investigation ran a series of simulation experiments using the *ns-2* simulator to compare the performance of ECN with four AECN variants with heterogeneous TCP Reno sources. The simulated network topolgy (shown in Figure 2) consists of one router, one sink and a number of simulated FTP sources. Each FTP source feeds 1000-byte packets into a single congested link attached to the router. The TCP ACK packets are 40 bytes long and each source has a window size of 64 packets.

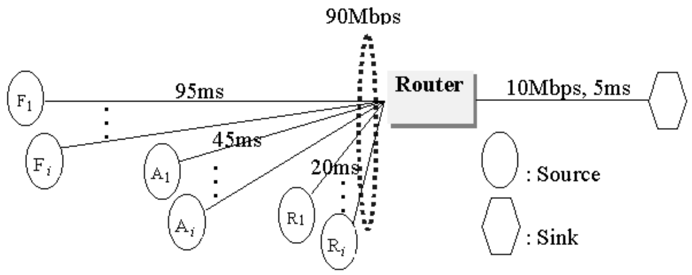

Fig. 2: Simulation Topology

The capacity of the bottleneck link is 10 Mbps with a 5 ms delay to the sink. Using the measured RTT range of 7 ms to 137 ms discussed by Christiansen et al. [8] as a guide, we chose one-way link delays for the fragile, average and robust sources of 95 ms, 45 ms and 20 ms respectively. Thus, the fragile, average and robust flows have round-trip times of 200 ms, 100 ms and 50 ms when there is no queuing delay at the router. The router queue size was fixed at 120 packets based on published rules-of-thumb for accommodating the network bandwidth-delay product.

All simulations for this study ran for 100 simulated seconds and include an equal number of fragile, average and robust TCP flows. Half of the flows in each flow group start at time 0, the second half start at time 2 seconds. For example, for a 60 flow simulation, 10 fragile, 10 average and 10 robust flows start at time 0, and the remaining 30 flows start at 2 seconds. The first 20 seconds of simulated time are not considered to reduce the startup and transient effects.

The router determines the flow group of an arriving packet using ranges of [0.5 ms, 75 ms) for robust flows, [75 ms, 150 ms) for average flows and greater than or equal to 150 ms for fragile flows. These ranges take into account the added delay for RTT caused by waiting in the router queue. For the first packet of any flow, 0 ms is sent for the RTT. AECN handles this case by treating this first packet as an average flow.

The sum of the capacities of all the incoming flows is held constant at 90 Mbps for all simulations in this study regardless of the number of flows. Thus when the number of flows are increased the individual link capacities are proportionally decreased. Unless specifically specified, the values for *min_th* and *max_th* are set at 10 and 30 packets respectively.

5 ECN Simulations

Before running simulations that compare Adaptive-ECN and ECN, preliminary simulations were run to confirm the behavior of ECN and RED using the simulation topology and settings detailed in the previous section. These results differ from previous results [23] in that the sum of the incoming flow capacities (referred to here as demand) has been reduced from 300 Mbps to 90 Mbps. This was done to keep the ratio of the demand to the bottleneck link capacity under 10.

Figure 3 presents average goodput results as the number of flows generating 90 Mbps input demand is varied from 30 to 450 for simulations of RED and ECN routers with three different settings of *max_p*. For 30, 60 and 120 flows, ECN has higher goodput than RED for the same *max_p* setting. The best *max_p* setting for ECN changes with the number of flows: $max\_p = 0.1$ is best for 30 flows; $max\_p = 0.5$ is best for 60, 240 and 450 flows; and $max\_p = 0.8$ is best for 120 flows. At 120 and 240 flows the goodput has declined and the differences in goodput is small.

Figures 4 and 5 give two different fairness measures - Jain's fairness and visual max-min fairness. The Jain's fairness behavior in Figure 4 is similar to the goodput results in that ECN does better for the smaller number of flows, but at 120 flows the fairness for all six treatments drops to about 0.5. Figure 5 shows both the aggregate RED and ECN goodput and the share of goodput for the fragile, average and robust flows. The large goodput gap between robust and fragile flows for both the RED and ECN is evidence that fragile and average flows receive a smaller share of the goodput than the robust flows. With *max_th* set at 30, these 120 flows are vying for these buffers at the router queue and the robust flows get better service.

To better understand the performance degradation between 60 and 120 flows in Figures 3 and 4, the next six figures look at the marking and dropping behavior of RED and ECN at 60 and 120 flows. Figures 6, 8 and 10 are results for 60 flows and Figures 7, 9 and 11 provide data for 120 flows. In all of these figures, the lower third of the flow numbers are fragile flows, the middle third are average flows, and the top third are robust flows.

Figures 6 and 7 show the change in ECN marks when the number of flows is doubled. The ECN marks per flow (horizontal pattern) in Figure 7 have a higher density than in Figure 6. Furthermore, one can see the differences in the mark patterns for the fragile, average and robust flows. In Figure 8, there are almost no drops for ECN at 60 flows. However at 120 flows in Figure 9, ECN experiences many more drops with robust flows having the highest drop density. The horizontally white spaces in Figures 7 and 9 represent periods when some flows are "locked out" and experiencing no drops or no marks. This lockout phenonemom significantly lowers the Jain's fairness.

Turning to RED data in Figures 10 and 11, one can see the dramatic increase in RED drops at 120 flows. Comparing ECN to RED at 60 flows, the graphs show that ECN's better performance can be attributed to avoiding drops by timely marking. However, at 120 flows the advantage of the marking scheme has been diminished.

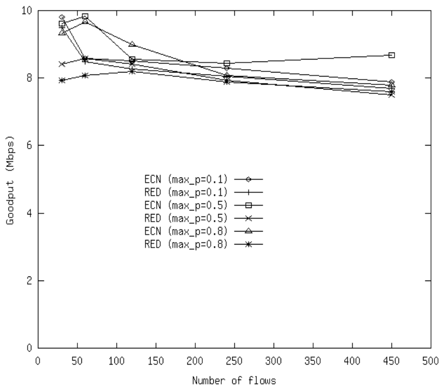

Fig. 3: RED and ECN Goodput

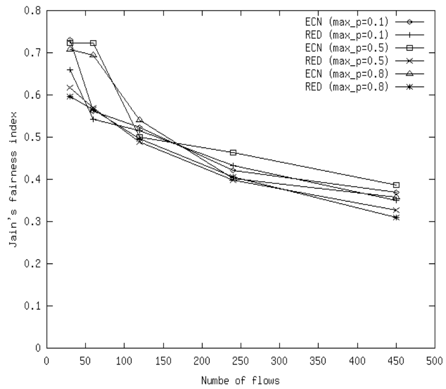

Fig. 4: RED and ECN Jain's Fairness

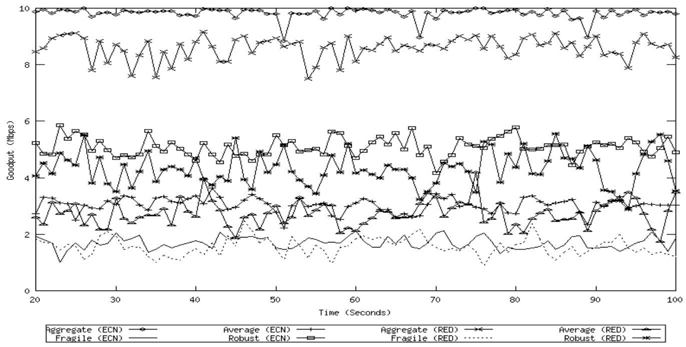

Fig. 5: RED and ECN Goodput, 60 flows, $max\_p = 0.5$

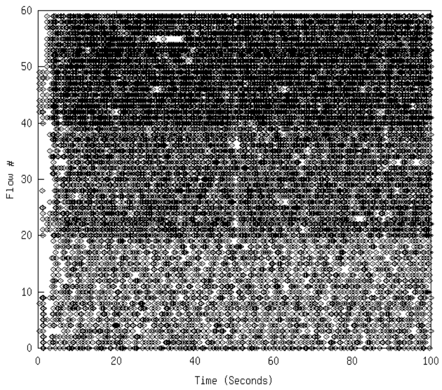

Fig. 6: ECN Marks, 60 flows, $max\_p = 0.5$

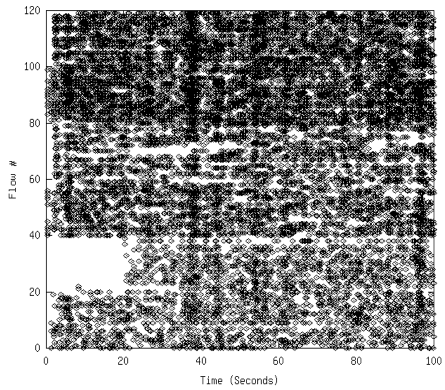

Fig. 7: ECN Marks, 120 flows, $max\_p = 0.5$

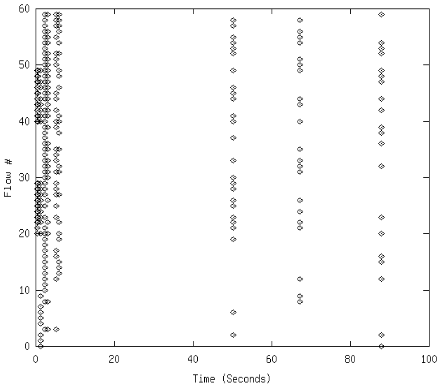

Fig. 8: ECN Drops, 60 flows, $max\_p = 0.5$

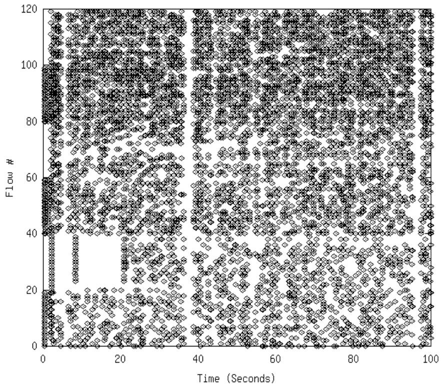

Fig. 9: ECN Drops, 120 flows, $max\_p = 0.5$

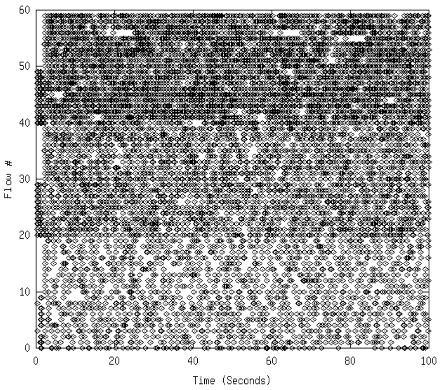

Fig. 10: RED Drops, 60 flows, $max\_p = 0.5$

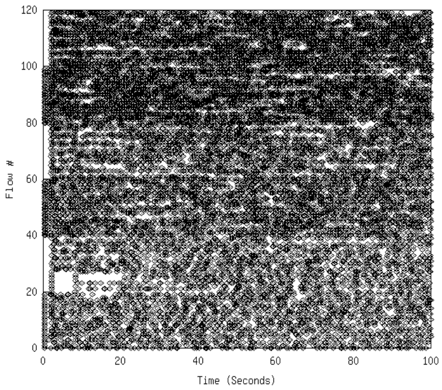

Fig. 11: RED Drops, 120 flows, $max\_p = 0.5$

Our investigation looked at similar data for 240 flows which indicated the lockout phenomenon to be more serious with some flows being locked for almost the entire simulation. This caused us to focus our attention below 240 flows where it appeared that an adaptive version of ECN might prove effective.

6 Performance Comparison Between AECN and ECN

The results from the previous section reconfirm that generally ECN performs better than RED and indicate that an adaptive version of ECN could be used to combat the unfair treatment of fragile and average flows seen in Figure 5. This section focuses on the basic adaptive ECN (AECN) strategy explained in section 3.

AECN attempts to provide fairer treatment by using a different value for max_p for packets in each of the three flow groups. The key to this concept is making good choices for α and β.

6.1 The Selection of α and β

Initially, this research considered developing a closed form relationship between a flow's round-trip time (RTT) and a *max_p* setting that would be optimally fair. However, the number of tuneable RED/ECN parameters that effect router performance precluded any closed form relationship. Thus a number of ns-2 simulations were run to gauge the influence of different α, and β settings on AECN performance.

For simplicity, we assume α is equal to β. This symmetry assumption only holds when the effect due to relative difference in RTTs between robust and average flows matches the effect due to the relative difference in RTTs between average and fragile flows. Figures 12 through 15 show results designed to capture the effect of varying α on the performance of AECN with 60 flows and 120 flows.

Figures 12 and 13 show the effect of varying α on AECN performance with $base\_max\_p = 0.5$ for 60 flows while Figures 14 and 15 are the equivalent results for 120 flows. Remembering that an $\alpha = 1$ is close to a normal ECN router, Figure 12 shows that increasing α from 1.0 to 3.0 has a significant effect on the visual max-min fairness with the smallest goodput gap occurring when α is 2.8. Notice that while reducing the goodput gap between the three flow groups the aggregate goodput is decreased only slightly. So the price for improved fairness is small. $\alpha = 2.5$ is clearly the optimal setting in Figure 13 which graphs the effect of α on Jain's fairness. Since the difference between the visual fairness gap for α between 2.5 and 2.8 in Figure 12 is small, $\alpha = 2.5$ is selected as the best choice that takes into account both of these fairness measures.

Figures 14 and 15 demonstrate comparable results when there are 120 flows with one significant diffference. Note that in Figure 14 there is no value of α that significantly reduces the robust flows' share of the goodput. The upper bound of 1 for *max_p* is limiting the effect of AECN. $\alpha = 2.5$ remains the best choice at 120 flows when one considers both fairness indicators. Thus α set to 2.5 for AECN will be used in the next section to compare AECN with ECN.

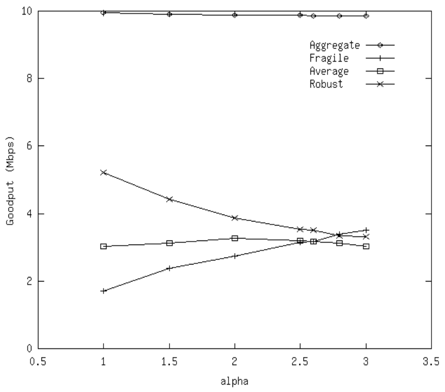

Fig. 12: AECN Goodput, 60 flows, $base\_max\_p = 0.5$

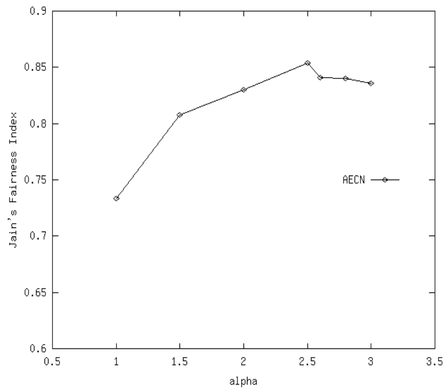

Fig. 13: AECN Jain's Fairness, 60 flows, $base\_max\_p = 0.5$

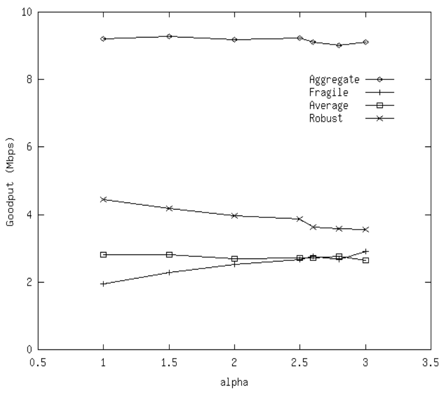

Fig. 14: AECN Goodput, 120 flows, $base\_max\_p = 0.5$

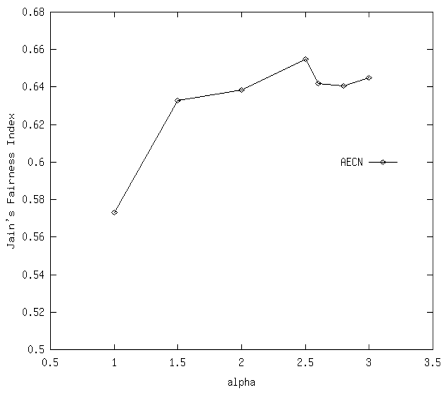

Fig. 15: AECN Jain's Fairness, 120 flows, $base\_\ max\_p = 0.5$

6.2 Performance of AECN with $\alpha = 2.5$

This section presents and analyzes simulation results to compare the performance of AECN with $\alpha = 2.5$ and $\beta = 2.5$ with the performance of ECN using the simulation topology and ECN router parameter settings discussed in section 3.

Goodput and Throughput Figures 16 compares ECN and AECN goodput with $base\_max\_p = 0.5$ and Figure 17 presents goodput when $base\_max\_p = 0.8$. AECN yields higher aggregate goodput than ECN in all cases. In Figure 16 AECN has a smaller visual max-min fairness gap for 30 and 60 flows; but the aggregate goodput decreases and the visual fairness gap increases for 120 flows. AECN with $base\_max\_p = 0.8$ (in Figure 17) has higher aggregate goodput at 120, 240 and 450 flows and it also reduces the fairness gap at 120 flows, but at 240 flows the fairness gap has widened compared to Figure 16. These results suggest that the value of $base\_max\_p$ could be adaptively changed based on some indication of the number of flows at the router.

Figures 18 and 19 show that AECN also yields higher throughput than ECN. Note, the importance of goodput at the higher flow levels. Figures 18 and 19 indicate little drop in throughput at 240 and 450 flows, but the drops in goodput shown in Figures 16 and 17 imply that a significant number of retransmissions are contributing to the throughput.

Packet Marks, Packet Drops and Queue Length Figures 20-23 are presented to further explain the advantage of AECN over ECN by contrasting the AECN mark and drop patterns at 60 and 120 flows versus the equivalent results presented in Figures 6-9. At 60 flows AECN has more marks than ECN for all flow groups and AECN has drops after the startup period compared to a few drops for ECN. At 120 flows AECN has more marks and less drops than ECN. This is the main cause for better goodput and faireness for AECN.

Figure 24 shows both the instanteous queue length and the average queue length behavior for the simulation used in Figures 22 and 23. One curious pattern in Figure 23 are occasions

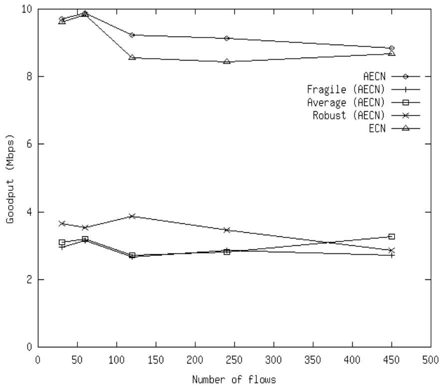

Fig. 16: AECN Goodput, *base_max_p* = 0.5

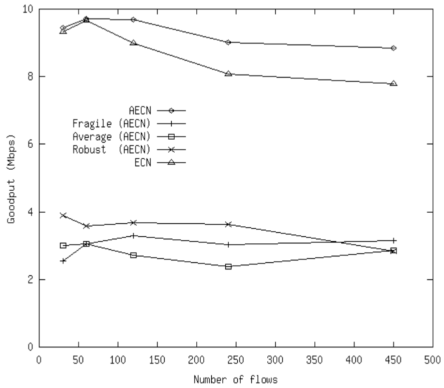

Fig. 17: AECN Goodput, *base_max_p* = 0.8

when AECN results in no drops (i.e., vertical white space). Close inspection of Figure 24 reveals that at 120 flows there are short periods when the average queue size stays below *max_th*. During these intervals, the AECN marking is timely enough to keep the router temporarily in a semi-stable state where there are no AECN drops.

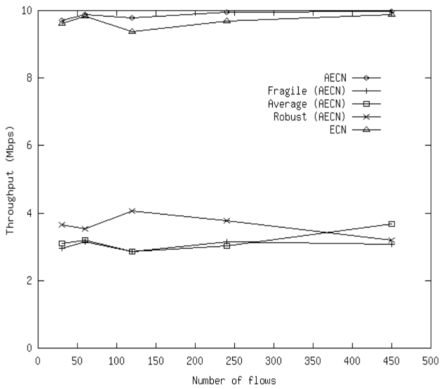

Fig. 18: AECN Throughput, *base_max_p* = 0.5

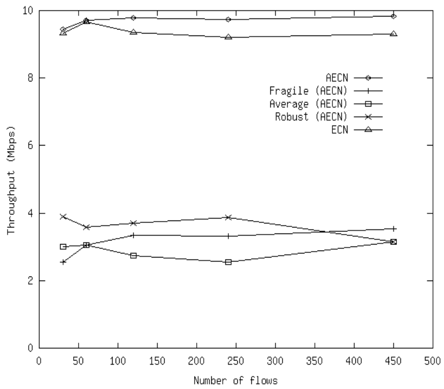

Fig. 19: AECN Throughput, *base_max_p* = 0.8

Fairness Figures 25 and 26 compare the fairness of AECN and ECN using Jain's fairness index. AECN is fairer in all cases. Below 240 flows, AECN improves Jain's fairness over ECN by more than 10% for both *base_max_p* settings. At 120 and 240 flows, AECN is fairer with *base_max_p* at 0.8. As with the previous discussion of visual max-min fairness, this suggests dynamically adjusting *base_max_p* based on the number of flows. However, the 450 flow data points imply there is a limit to the improvement that can be made by adjusting *base_max_p*.

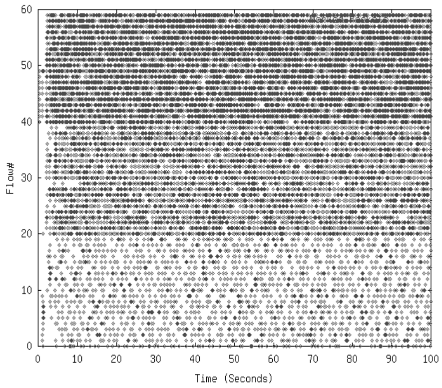

Fig. 20: AECN Marks, 60 flows, $base\_max\_p = 0.5$

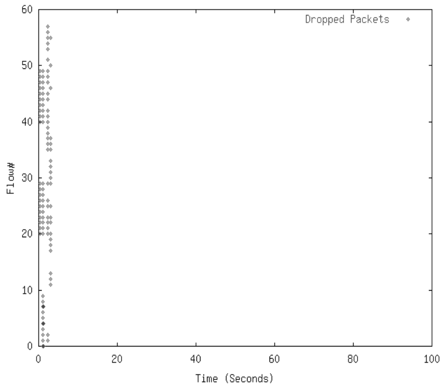

Fig. 21: AECN Drops, 60 flows, $base\_max\_p = 0.5$

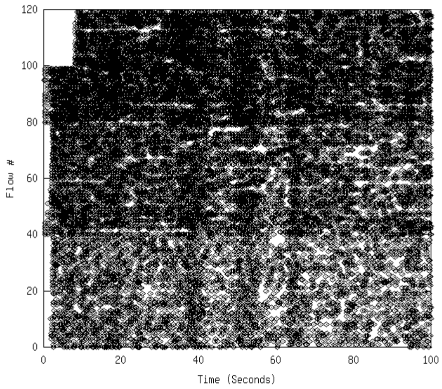

Fig. 22: AECN Marks, 120 flows, $base\_max\_p = 0.5$

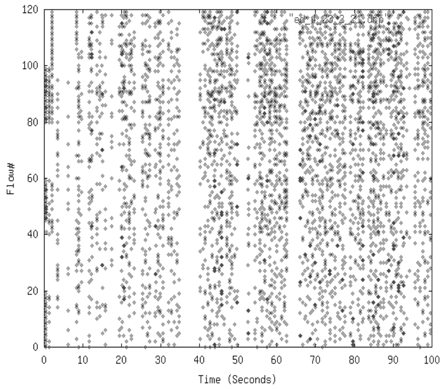

Fig. 23: AECN Drops, 120 flows, $base\_max\_p = 0.5$

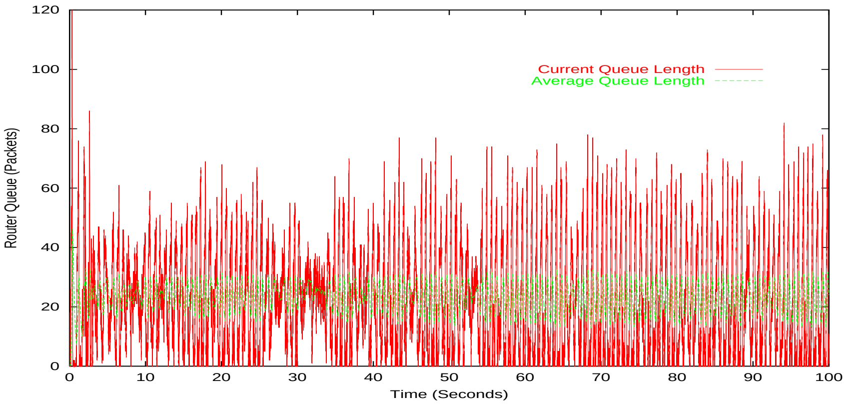

Fig. 24: AECN Queue Length, 120 flows, $base\_max\_p = 0.5$

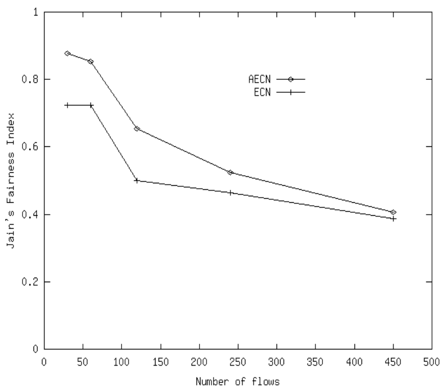

Fig. 25: ECN and AECN Jain's Fairness, $base\_max\_p = 0.5$

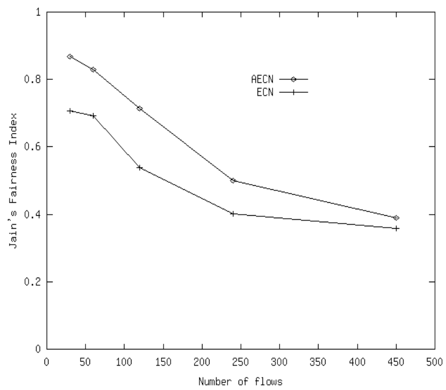

Fig. 26: ECN and AECN Jain's Fairness, $base\_max\_p = 0.8$

Delay Figure 27 presents the average one-way delay distribution for the three flow groups as the number of flows increase. Increasing the number of flows generating a fixed demand causes the delay to go up in two ways. First, since the individual link capacities decrease as the number of flows increase, there will be an increase in the packet transmission time per flow. Second, the queueing delay at the router will increase as average queue length goes up.

The important point from the delay results is that AECN provides better goodput and fairness than ECN with only a slightly higher delay. Although it is difficult to see in Figure 27, at 30 and 60 flows the AECN delay is lower than the ECN delay. This can be attributed to the mark front policy built into the AECN algorithm.

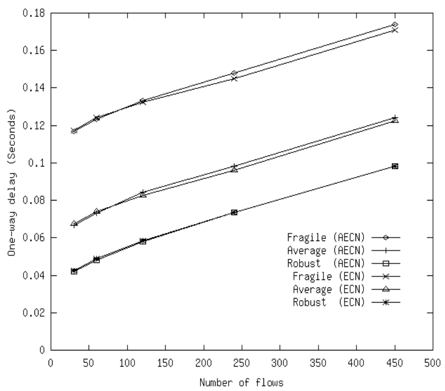

Fig. 27: Delay with $base\_max\_p = 0.5$

In summary, the basic AECN mechanism performed better than ECN with respect to goodput and fairness in the simulations discussed in this session. The analysis suggests that AECN can be improved by dynamically adjusting $base\_max\_p$ based on some estimate of the number of flows active at the bottlenecked router. However, with a large number of flows other adjustments need to be considered because the basic AECN mechanism is unable to throttle back robust flows that get more than their fair share of the goodput.

7 AECN Refinements

The section discussed three possible refinements that were simulated to investigate dynamically tuning ECN parameters to improve performance in some circumstances. A major objective of these possible enhancements is to provide fairness when the number of flows is large. These three separate schemes are referred to as AECN2, AECN3 and AECN4 respectively.

7.1 AECN2: Population-Balanced AECN

This method is based on the concept that fairness between the three flow groups can be maintained by dynamically adjusting the AECN mark probability such that fairness will occur when each of the flow queues has the same population. Thus AECN2 linearly increases the drop probability when a flow queue is larger than the average queue and linearly decreases the drop probability when the flow queue size is smaller than the average queue size. The hope is that this scheme will adequetely throttle the robust flows.

The detailed algorithm for AECN2 used to adjust *max_p* when the average queue size is between *min_th* and *max_th* is given below.

avg_fq_len = the length of the average flow queue;
if (avg_fq_len==0)
    avg_fq_len = the length of the router queue;
if (packet_flow_type == FRAGILE)
    α = avg_fq_len / the length of the fragile flow queue;
    max_p = base_*max_p* / α;
else if (packet_flow_type == AVERAGE)
    max_p = base_*max_p*;
else if (packet_flow_type == ROBUST)
    β = the length of the robust flow queue / avg_fq_len;
    max_p = base_*max_p* * β;

Figures 28 and 29 show goodput and fairness results for a series of ns-2 simulations of AECN2 with the same topology and settings given in section 3 with 60 flows. The Jain's fairness index for AECN2 in Figure 28 is lower than for AECN 60 flows in Figures 25 (*base_max_p* = 0.5) and 26 *base_max_p* = 0.8). Figure 29 further demonstrates that the population-balanced method of AECN2 does not reduce the aggressiveness of the robust flows that always get more than 1/3 of goodput regardless of the *max_p* setting. The adjustment by population ratio in AECN2 is linear with respect to *max_p* and the implication of these simulations is the impact of RTTs on fair share of goodput is nonlinear. Keeping the population of the flow queues balanced does not appear useful when good fairness is the goal.

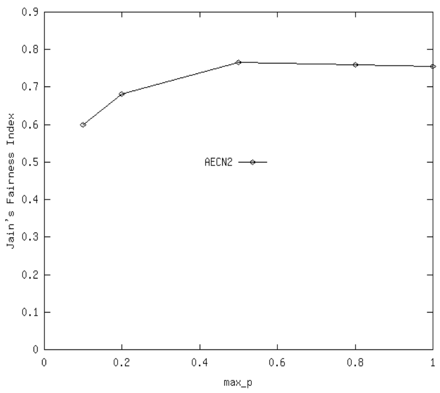

Fig. 28: AECN2 Jain's Fairness with 60 flows

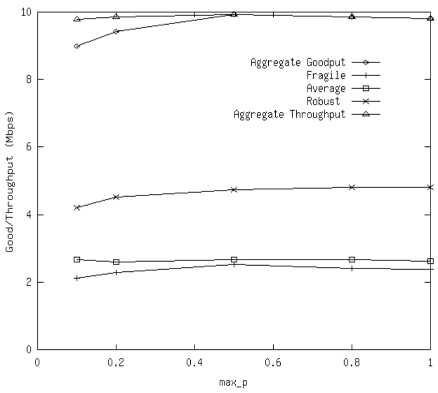

Fig. 29: AECN2 Goodput and Throughput with 60 flows

7.2 AECN3: Adjusting min_th

Figure 14 showed that when the demand is generated by a large number of flows increasing the robust flow drop probability has little effect on reducing the robust flows' share of goodput. Figure 30 shows that a smaller value for *min_th* with a fixed *max_th* should yield both more marks and earlier marks for robust flows. This section considers AECN3 which adds a lower *min_th* value for robust flows to the basic AECN algorithm. Figures 31 through 36 present simulation results designed to study the potential of AECN3.

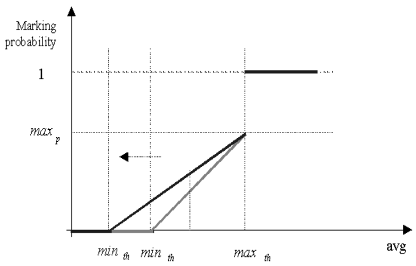

Fig. 30: *min_th* - Marking Probability Relationship

Looking at a series of 60 flow AECN3 simulations in Figure 31, one can see that lowering *min_th* for robust flows from 10 towards 5 causes a slight drop in Jain's fairness. However, Figure 32 shows that $min\_th = 9$ is the optimal point with respect to the visual fairness gap. Figure 33 shows that the throughput and goodput are almost the same for 60 flows because there are very few drops.

However in Figures 34-36 with 120 flows generating the demand, the performance behavior is different when *min_th* is varied. As *min_th* is lowered in Figure 34 from 10 (the initial AECN setting), the Jain's fairness improves. Furthemore Figure 35 shows that the aggregate goodput goes up slightly and the visual max-min fairness goes down as *min_th* is lowered from 10 to

1 for robust flows. The visual max-min fairness is the best when *min_th* is set to 3 packets for robust flows. Figure 36 shows throughput higher than the goodput in Figure 35. With 120 flows and *max_th* at 30, AECN3 experiences dropped packets that cause goodput to be smaller than throughput.

These results show that adjusting *min_th* for robust flows does improve fairness, but the best *min_th* setting depends on the number of flows.

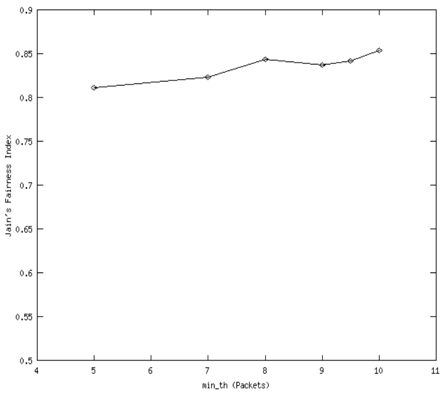

Fig. 31: AECN3 Jain's Fairness, 60 flows, $base\_max\_p = 0.5$

.

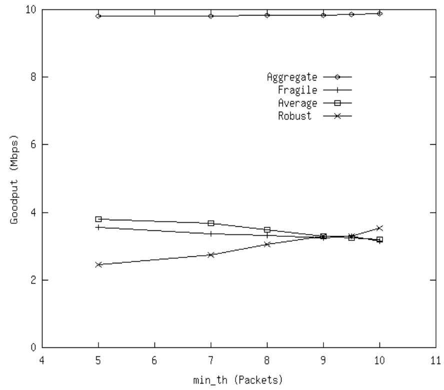

Fig. 32: AECN3 Goodput, 60 flows, $base\_max\_p = 0.5$

.

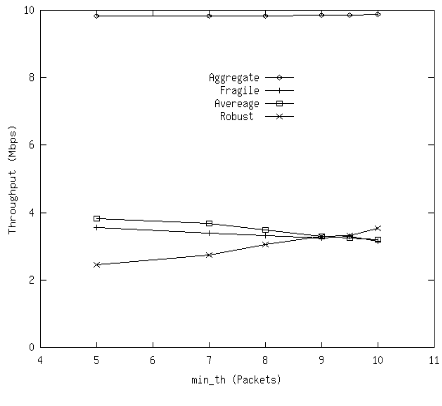

Fig. 33: AECN3 Throughput, 60 flows, $base\_max\_p = 0.5$

.

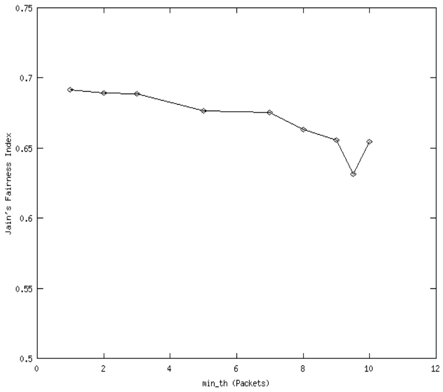

Fig. 34: AECN3 Jain's Fairness, 120 flows, $base\_max\_p = 0.5$

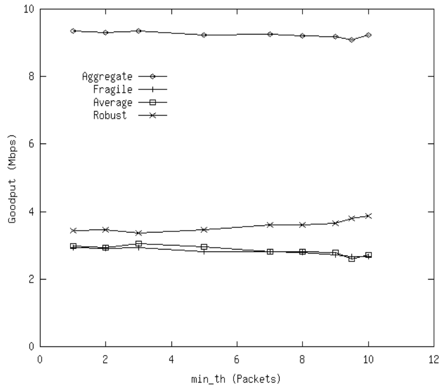

Fig. 35: AECN3 Goodput, 120 flows, $base\_max\_p = 0.5$

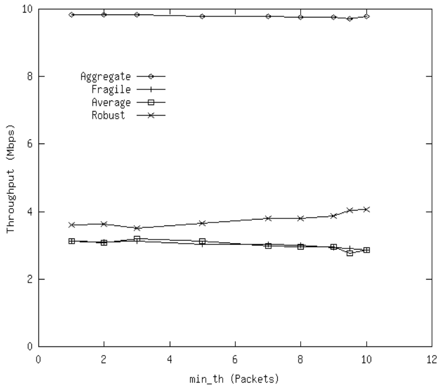

Fig. 36: AECN3 Throughput, 120 flows, $base\_max\_p = 0.5$

7.3 AECN4: Adjusting max_th

Although AECN3 was able to provide better fairness by adjusting *min_th*, the goodput for 120 flows was less than the goodput for 60 flows due to ECN drops that occur when *ave_q* exceeds *max_th*. This section considers another variant of AECN, AECN4 that dynamically adjusts *max_th* to avoid ECN drops. Two possible approaches are to adjust *max_th* based on some measure of the router queue drops or to adjust *max_th* based on some estimate of the number of flows currently passing through the bottlenecked link. We examine the second concept and assume there exists a mechanism at the router to estimate the current number of active flows.

To investimate the plausibility of this strategy, a series of simulations were run where *max_th* was varied while the remainder of the tuning parameters were fixed at the best settings from the previous AECN3 discussion. These AECN4 settings include:

1. *base_max_th* equal to 0.5,
2. *min_th* = 9 for robust flows when there are 60 flows
3. *min_th* = 3 for robust flows when there are 120 flows
4. *min_th* = 10 for average and fragile flows,

5) $\alpha = \beta = 2.5$

Figures 37 through 40 present simulations of 60 and 120 flow runs of AECN4 using the above parameters with *max_th* varied from 30 to 60. Figures 37 and 38 show that at 60 flows increasing *max_th* has little effect on fairness and goodput because there were very few drops when *max_th* was at 30. However, Figures 37 and 39 indicate that at 120 flows Jain's fairness, visual max-min fairness and goodput are best with *max_th* = 60.

Figures 39 and 40 demonstrate the tradeoff of higher goodput and higher delay for 120 flows. For example, when *max_th* is increased from 30 to 50 packets, the goodput and Jain's fairness both increase about 7%, while the one-way delays of robust, average and fragile flows increase about 17%, 8% and 7% respectively. These experiments indicate that if a network application can tolerate delay increases then both goodput and fairness can be improved by having the router dynamically adjust *max_th* based on a good estimate of the number of flows. This adjustment when there are many flows lowers the likelihood that ECN will drop packets.

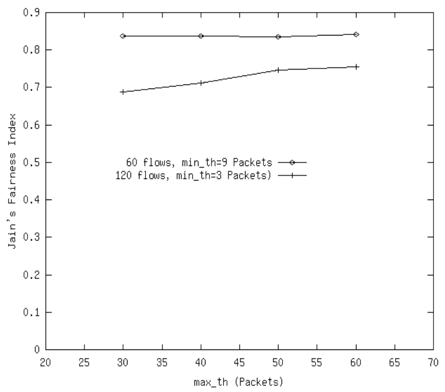

Fig. 37: AECN4 Jain's Fairness

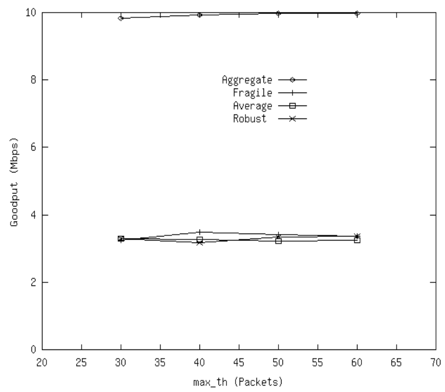

Fig. 38: AECN4 Goodput with 60 flows

8 Conclusions and Future Work

This paper presents the major results from Zici Zheng's masters thesis [**?**]HE01. The main research contribution is the basic Adaptive ECN (AECN) mechanism. The significant change

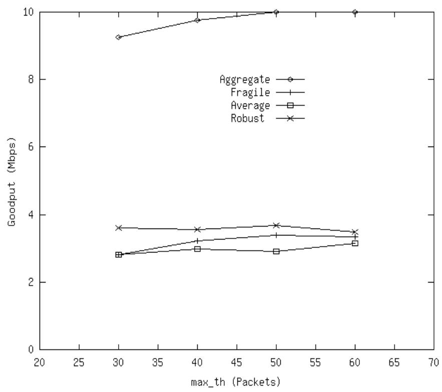

Fig. 39: AECN4 Goodput with 120 flows

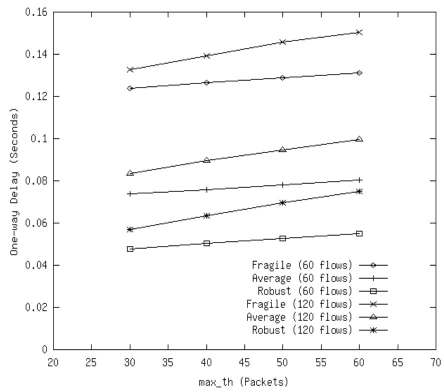

Fig. 40: AECN4 Delay

required to deploy AECN is that TCP packets must contain an indication of current round-trip time for a given flow. This indicator may be a specific RTT added to the packet or possibly a few bits that stratify flow RTTs into levels. The former approach when implemented would probably imply coming up with a function to vary *max_p* with the incoming RTT and eliminate flow queues. The latter scheme results in less packet overhead but requires careful designation of the specific flow queue ranges.

The basic AECN mechanism divides TCP flows into distinct flow groups at the router based on the RTT indicator in the arriving packet. AECN uses mark front per flow queue and distinct *max_p* values for each flow group. The ns-2 simulations show that AECN provides better Jain's fairness and visual max-min fairness than ECN for heterogeneous flows. AECN is better than ECN in regions where the relationship between the number of flows, the input demand and the *max_th* setting are such that AECN yields few drops. Because the mark front strategy causes the ECN marks to reach the TCP sources quicker, AECN provides for more responsive adjustments to congestion that limits the packet drops at the router. Higher drop probabilities for robust flows and lower drop probabilities for fragile flows produce the improved AECN fairness. However, this AECN improvement is reduced when too many flows pass through the bottlenecked link.

Three AECN variants (AECN2, AECN3 and AECN4) were considered as possible adaptive measures to deal with a large number of flows. The simulations of AECN2, population-balanced AECN, indicate that dynamically balancing the population of the flow groups did not improve AECN performance.

AECN3 lowers *min_th* for robust flows. The simulations show adding this to AECN further improves AECN fairness when there are 60 and 120 flows. However, to be most effective AECN3 needs to dynamically change the *min_th* setting depending on the number of flows.

Since *max_th* artificially controls the average queue size at the router, this tuning parameter also affects the contention for these queue buffers. This contention becomes acute when the number of flows increases. AECN4 simulations show that dynamically raising *max_th* when the number of flows gets too high, alleviates the resource contention and can dramatically reduce ECN packet drops. This packet drop avoidance scheme permits the other AECN mechanisms to continue to provide better goodput and fairness than ECN. However, these performance improvements are offset by higher delay which many network applications cannot tolerate.

This investigation reconfirms that generally ECN performs better than RED. The ECN simulations indicate however than moving from a RED mechanism to an ECN mechanism require different parameter settings. Specifically, ECN appears to need a more aggressive *max_p* to outperform RED.

This study provides three adaptive ECN mechanisms that can improve goodput and fairness for heterogeneous TCP flows. Although this investigation did not produce a version of AECN that dynamically adjusts to changes experienced at a congested router, it did provide evidence that with hints from the TCP sources (i.e, RTTs) and local flow count indicators, an AECN router can outperform an ECN router.

Many questions remain about the viability of this approach. We have not addressed the issue of whether ECN is easier to tune than RED. This investigation has not considered topologies of cascaded ECN routers or mixed ECN, RED and drop-tail routers. Little is known about the fairness of ECN when some of the flows are not ECN-compliant. Thus future work includes more complicated network topologies and more diverse traffic flows. The basic AECN scheme presented in this study used three flow groups with fixed RTT ranges to define flow groups. Further investigation is needed to generalize this scheme. The flow groups could be eliminated if a function can be determined that automatically adjusts *max_p* based on RTT. It remains to be seen whether a scheme for dynamically adjusting *max_p*, *min_th*, and *max_th* can be developed that is robust across the domain space of possible flow situations. Furthermore, mark front ECN needs to be studied with web server traffic to see if the performance improvements can be retained with short-lived flows.

References

1. Ahmed, U., and Salim, J., "Performance Evaluation of Explicit Congestion Notification (ECN) in IP Networks", IETF Internet Draft, RFC2884, December 1999.
2. Ahn, J., Danzig, P., Liu, Z. and Yan, L., "Evaluation of TCP Vegas: Emulation and Experiment", SIGCOMM 1995, pp. 185-195.
3. Bagal, P., Kalyanaraman, S., Packer, B., "Comparative Study of RED, ECN and TCP Rate Control", Technical Report, March 1999.
4. Balakrishnan, H., Padmanabhan,V., Seshan, S., Stemm, M., and Katz, R., "TCP Behavior of Busy Internet Server: Analysis and Improvement", INFOCOM'98, March 1998.
5. Brakmo, L., Malley, S., and Peterson, L., " TCP Vegas: New Techniques for Congestion Detection and Avoidance", SIGCOMM 1994, August 1994, pp. 24-35.
6. Brakmo, L., and Peterson, L., "TCP Vegas: End to End Congestion Avoidance on a Global Internet", IEEE Journal on Selected Areas in Communications, vol. 13, no. 8, October 1995, pp. 1465-1480.
7. Braden, R., Clark, D., Crowcroft, J., Davie, B., Deering, S., Estrin, D., Floyd, S., Jacobson, V.. Minshall, G., Patridge, C., Peterson, L., Ramakrishnan, K.,Shenker, S., Wroclawski, J., and Zhang, L., "Recommendations on Queue Management and Congestion Avoidance in the Internet", RFC2309, April 1998.
8. Christiansen, M., Jeffay, K., Ott, D. and Smith F., "Tuning RED for Web Traffic", SIGCOMM'00, August 2000, pp. 139-150.
9. Feng, W., Kandlur, D., Saha, D., and Shin, K., "A Self-Configuring RED Gateway", INFOCOM'99, March 1999, pp. 1320-1328.

10. Feng, W., Kandlur, D., Saha, D., and Shin, K., "Blue: A New Class of Active Queue Management Algorithms", U. Michigan CSE-TR-387-99, April 1999.
11. Floyd, S., "Connections with Multiple Congested Gateways in Packet-Switched Networks Part1: One-way Traffic". ACM Computer Communication Review, vol. 21, no. 5, October 1991, pp. 30-47.
12. Floyd, S., and Jacobson, V., "Random Early Detection Gateways for Congestion Avoidance", IEEE/ACM Transactions on Networking, vol.1, no. 4, August 1993, pp. 397-413.
13. Floyd, S., "TCP and Explicit Congestion Notification", ACM Computer Communication Review, vol. 24, no. 5, October 1994, pp. 10-23.
14. Floyd, S., and Fall, K., "Simulation-based Comparisons of Tahoe, Reno and SACK TCP", Computer Communication Review, vol. 26, no. 3, July 1996. pp. 5-21.
15. Floyd, S., "RED: Discussions of Setting Parameters", http://www.aciri.org/floyd/REDparameters.txt, November 1997.
16. Floyd, S., "Implementing ECN in TCP", http://www.aciri.org/floyd/ECN-TCP.txt, January 1998.
17. Floyd, S., "RED with drop from front", email discussion on the end2end mailing list, ftp://ftp.ee.lbl.gov/email/sf.98mar11.txt, March 1998.
18. Floyd, S., "Congestion Control Principles", IETF Internet Draft, http://www.aciri.org/floyd/papers/draft-floyd-cong-01.txt, January 2000.
19. Gerla, M., Cigno, R., Mascolo, S., and Weng, W., "Generalized Window Advertising for TCP Congestion Control", Technical Report No. 990012, Computer Science Department, UCLA, January 1999.
20. Jacobson, V., "Congestion Avoidance and Control".SIGCOMM '88, Palo Alto, CA, August 1988.
21. Jain, R., "The Art of Computer Systems Performance Analysis", John Wiley and Sons, 1991.
22. Karn, P., and Partridge, C., "Improving Round-Trip-Time Estimates in Reliable Transport Protocol," ACM Transaction on Computer Systems (TOCS), vol. 9, no. 4, pp.364-373, November 1991.
23. Kinicki, R., and Zheng, Z., "Performance Research of Explicit Congestion Notification (ECN) with Heterogeneous TCP Flows", International Conference on Networking (ICN), Colmar, France, July 2001, pp. 98-106.
24. Kunniyur, S., and Srikant, R., "End-to-End Congestion Control Schemes: Utility Functions, Random Losses and ECN Marks", Infocom 2000, pp.1323-1332.
25. Lin, D. and Morris, R. "Dynamics of Random Early Detection", ACM Computer Communication Review, vol. 27, no. 4, October 1997, pp. 127-138.
26. Liu, C., and Jain, R., "Improving Explicit Congestion Notification with the Mark-Front Strategy", Computer Networks, vol. 35, no. 2-3, January 2001, pp. 185-201.
27. Mathis, M., Semke, J., and Mahdavi, J., "The Macroscopic Behavior of the TCP Congestion Avoidance Algorithm", ACM Computer Communication Review, vol. 27, no. 3, July 1997.
28. Morris, R., "TCP Behavior with Many Flows", IEEE ICNP'97, October 1997.
29. http://www.isi.edu/nsnam/ns/doc/index.html.
30. Peterson, L.L., and Davie, B.S., "Computer Networks, A Systems Approach", 2nd Ed., Morgan Kaufmann, San Francisco, 2000.
31. Qiu, L., Zhang, Y., and Keshav, S., "On Individual and Aggregate TCP Performance", IEEE ICNP'99, 1999.
32. Ramakrishnan, K., and Floyd, S., "A Proposal to Add Explicit Congestion Notification (ECN) to IP", ftp://ftp.isi.edu/in-notes/rfc2481.txt, January 1999,
33. Rammakrishnan, K., Floyd, S., and Black, D., "The Addition of Explicit Congestion Notification (ECN) to IP", IETF Draft, draft-ietf-tsvwg-ecn-02.txt, February 2001.
34. Yin, N., and Hluchyj, M., "Implication of Dropping Packets from the Front of a Queue", 7th ITC, Copenhagen, Denmark, October 1999.
35. Zheng, Z. "Adaptive Explicit Congestion Notification (AECN) for Heterogeneous Flows", WPI Master's Thesis, May 2001.