CONGESTION CONTROL IN TCP/IP NETWORKS: A COMBINED ECN AND BECN APPROACH

Frank Akujobi, Nabil Seddigh, Biswajit Nandy, Rupinder Makkar, Ioannis Lambadaris
Department of Systems and Computer Engineering
Carleton University, Ottawa, ON, Canada
fakujobi, nseddigh, bnandy, rup, ioannis@sce.carleton.ca

I. ABSTRACT

In this paper a novel algorithm is proposed which combines the merits of Explicit Congestion Notification (ECN) and Backward Explicit Congestion Notification (BECN) mechanisms for congestion control in TCP/IP networks. A comparative performance evaluation of the combined ECN and BECN mechanism is carried out using both long-lived FTP flows and short-lived web traffic. The simulation results show that the combined ECN+BECN mechanism benefits from BECN's early notification under heavy congestion and ECN's reliable delivery of congestion notification. It is observed that the ECN+BECN mechanism significantly reduces queue fluctuations due to early congestion indication compared to ECN. The loss of BECN Internet Control Message Protocol (ICMP) Source Quenches on the reverse path does not adversely impact performance due to this combined approach. It is also shown that the ECN+BECN scheme can significantly reduce the ICMP Source Quench reverse traffic in a network compared to a BECN only network. Experiments with webtraffic workloads show measurable improvement in both average object transfer delay and fairness over ECN.

Keywords – Backward Explicit Congestion Notification (BECN), Explicit Congestion Notification (ECN), Random Early Detection (RED), goodput, transfer delay.

II. INTRODUCTION

Explicit Congestion Notification (ECN) was proposed for TCP/IP networks as a means of explicitly notifying end-hosts of network congestion by marking, instead of dropping packets [1]. Recent studies have shown that RED with ECN support gives definite improvement over vanilla RED mechanisms in certain situations. For example, ECN has a better transfer delay for short-lived flows than packet drop schemes [2] [3]. It also reduces the number of timeouts for TCP flows. In terms of implementation, the ECN mechanism does not require generation of additional traffic at the router and can be easily implemented in the datapath of routers - it requires marking of a single bit.

This research was funded by grants from Communications and Information Technology Ontario (CITO) and Mathematics of Information Technology and Complex Systems (MITACS).

More recently, studies [4] have reported the use of Backward Explicit Congestion Notification (BECN) for congestion control in IP networks. The scheme relies on sending Internet Control Message Protocol (ICMP) Source Quenches (ISQ) as a means of reverse notification in TCP/IP networks. Specific proposals for generation and handling of ISQs were suggested and comparative performance of the individual ECN and BECN schemes in [4] revealed that BECN reduces the transfer delay for short-lived flows and improves the goodput of long-lived flows under heavy congestion. BECN also showed significant improvement for TCP flows that traverse paths with a high bandwidth delay product. Finally, a key benefit of BECN is that it is independent of the transport protocol and thus, can be used as a congestion notifier for multiple protocols. e.g UDP or SCTP.

A. Issues with BECN and ECN

A number of concerns have been raised over the use of BECN congestion notification in TCP/IP networks. Firstly, there is concern that the performance of Internet routers could be adversely affected due to the overhead required for generating ICMP Source Quenches (ISQ). This is because ISQ generation would be likely performed by the control plane of routers as opposed to the data plane which is much faster. Though no studies have been published reporting the performance overhead caused by ISQ generation, in general it is desirable to reduce the transfer of packets between the control and data plane on high speed routers.

Secondly, the issue of generating additional reverse traffic is of concern. Using ISQs will introduce a certain number of small packets in the reverse direction of the packets experiencing congestion. Although results in [4] showed that the amount of ISQ generated for BECN is bounded due to coupling with RED, it is preferable to minimize such control traffic to the extent possible.

Thirdly, the issue of reliable congestion notification is of concern [5]. ECN notifications can be more reliably delivered compared to ISQ notifications. ACK loss is not as much of a concern since ECN-Echo ACKs are continuously generated until the sender notifies the receiver that congestion notification has been received. This is achieved using a Congestion Window Reduced (CWR) mechanism. In the case

0-7803-8140-8/03/$17.00 (C) 2003 IEEE

of BECN, ISQ loss will not be detected by the sender nor can the congestion notification be reliably delivered. Should congestion persist, there is a stronger probability that a subsequently generated ISQ may serve to replace the lost ISQ. However, while this mitigates the lost ISQ it is not a reliable mechanism. The issue of BECN ISQ loss remains a concern to be addressed.

In addition to the BECN concerns, there are also concerns with ECN as it is currently specified for TCP/IP networks. The primary concern relates to the long delay in notification of congestion (as long as an RTT in some cases). Especially under heavy load, the congestion will persist for a longer time until the sources are notified to slow down. Some of the drawbacks of the delayed congestion are higher queue variance, reduced throughput and longer transfer delays for short-lived flows.

B. Motivation for a combined BECN/ECN

Given that both ECN and BECN have certain unique advantages, the question arises as to whether or not there is any benefit in combining the two mechanisms for improved congestion control in TCP/IP networks. A combined approach would be useful if it could take advantage of BECN's performance enhancements while mitigating the issues associated with ISQ loss and the additional network load generated by ISQ packets.

One approach to combining ECN and BECN would be for RED to always generate ISQ packets and mark the ECN CE bit whenever it seeks to notify the sender of incipient congestion. In this case, BECN would be the primary mechanism for congestion notification with ECN as the backup mechanism. While this would address the issue of ISQ loss, it would not address the issue of additional network load on the network.

A second approach to combining ECN and BECN would be for both mechanisms to co-exist and be used for different purposes. ECN would be used for low to medium level congestion while BECN would be used for the medium to high level congestion. When BECN is generated, ECN mechanisms would be used as a backup in case of ISQ loss. The benefit of this solution is that it reduces ISQ traffic on the network. Such a scheme would need to have a configurable threshold which would trigger BECN ISQ generation by the router in addition to ECN marking of the data packet.

In this paper we present a novel algorithm for congestion control that focuses on the second approach of combining the ECN and BECN schemes as it better addresses the issue of reducing ISQ reverse network traffic and ISQ loss on the reverse path. Simulations are performed to facilitate a comparative evaluation of the ECN+BECN, ECN and BECN mechanisms.

The rest of this paper is organized as follows: Section III gives a brief overview of recent related work on BECN. Section IV describes our proposed ECN+BECN algorithm. In Section V and VI, we describe simulation setup, test scenarios and explain the observed results. Finally, section VII concludes this paper and points to future work.

III. Related Work

Explicit Congestion Notification mechanism for congestion control in IP networks was first suggested and evaluated by Floyd in [2]. RFC 2481, an Experimental RFC described the required behavior for end-to-end hosts and routers for implementing ECN in TCP/IP networks. Hadi Salim et al in RFC 2884 [3] carried out an evaluation of ECN versus random early drop congestion control mechanisms. In [3] it is shown experimentally that ECN is able to give better goodput, web transaction rates and less number of timeouts as compared to random early drop mechanisms in TCP/IP networks. M. Kwon and S. Fahmy in [6] proposed modifications to the current method by which the TCP sender responds to ECN notification - the results showed some improvements over the currently standardized ECN mechanism.

The BECN mechanism has previously been used in non-IP networks, but there had been limited experimental investigation into the application of the BECN scheme as congestion control mechanism in IP networks. In [4], an enhanced algorithm for BECN was considered and a comparative performance evaluation of RED, ECN and the enhanced BECN mechanism was undertaken using both long-lived TCP bulk transfers and short-lived webtraffic workloads. Also in [7] a mechanism similar to BECN using ICMP Source Quenches is studied to regulate unresponsive UDP flows. RFC 896 [8] reported that appropriate handling of ISQs resulted in measurable benefits for the network.

In Frame Relay Networks, a combined mechanism of Forward and Backward Explicit Congestion Notification is used for congestion detection [9]. If congestion happens at one of the Frame relay switches, it sets the FECN bit on the packets being sent to the receiving device and BECN bit on the packets being returned to sending device. A FECN bit notifies the receiving end of the congested path so that upper layer protocols should expect some delay. The BECN bit is used to warn the sender that the frames it is transmitting are encountering congestion and the sending end may then adjust its flow of data accordingly.

IV. Proposed ECN+BECN Algorithm

As mentioned previously, it is anticipated that combining BECN and ECN mechanisms can result in a robust performance-enhanced congestion control for IP networks.

There are many different ways in which the two schemes can be combined. The scheme described in this section is intended to apply ECN to low congestion scenarios and BECN to high congestion scenarios. A new threshold between RED's *minth* and *maxth* is required in order to identify the point at which BECN is triggered. The threshold would be configurable. Thus, in one extreme, the new threshold could be set equal to *minth*, thus making the combined scheme operate exactly like BECN alone. In the other extreme, the new threshold could be set to equal *maxth*, thus making the combined scheme operate like ECN alone. In the following subsections, we describe the proposed mechanism in more detail and explain the guidelines for the behavior of all hosts in an ECN+BECN capable network.

A. *Behavior of an ECN+BECN-capable router*

For an ECN+BECN-capable router a new threshold *becnthresh* is defined, henceforth referred to as "BECN threshold". The BECN threshold has a value between the minimum and maximum RED thresholds and is used as a basis for determining when ISQs should be generated for marked and/or dropped packets. When the average queue size is below the BECN threshold the router behaves as a pure ECN router marking packets based on RED probability and forwarding such packets. Under heavier congestion when the average queue size exceeds the BECN threshold ISQs are also generated for every marked and/or dropped packet inaddition to the ECN forwarding. Hence the router essentially behaves as a pure ECN router under low congestion and under heavier congestion both ECN and BECN mechanisms are used for congestion control.

For a new arriving packet, the ECN+BECN-capable router behaves as follows:

If *(arriving packet causes the average queue size to go above maximum RED threshold)*
{
drop the packet just like an ECN packet;
if *(ECT[1] bit was marked in the IP header)*
{
send an ISQ due to a dropped packet back to the source;
}
} else if *(arriving packet causes average queue to go between minimum RED threshold and becnthresh)*{
if *(RED rules chooses this packet for marking and ECT bit is set)* and if *(packet is not already marked)*
{
mark the packet (CE[2] bit) but do not send an ISQ;
}
} else if *(arriving packet causes average queue to go between becnthresh and maximum RED threshold)*{
if *(RED rules chooses this packet for marking and ECT bit is set)* and if *(packet is not already marked)*
{
mark the packet (CE bit) and send an ISQ due to a marked packet back to the source;
}
else if *(RED chooses this packet and ECT bit is not set)*
drop the packet;
}

B. *Behavior of ECN+BECN-capable TCP end hosts*

ECN+BECN TCP end hosts perform initial end-to-end negotiations to establish ECN capability just like pure ECN hosts. At the sametime, the sender is responsive to both ECN and BECN congestion notifications. Under low congestion when the average queue is below *becnthresh* the end hosts respond to ECN signalling as pure ECN hosts. When congestion gets heavier and the average queue size exceeds *becnthresh*, RED's selection of a packet for marking would also cause an ISQ to be sent. As with ECN, when the marked packet arrives at the receiver, an ECE-ACK is generated to the sender. Whichever notification reaches the sender first should cause a window reduction. As allowed by the reduced congestion window, the first packet to be sent after window reduction will have Congestion Window Reduced (CWR) bit set to stop the receiver from sending more ECE-ACKs. Hence, the sender on receipt of an ISQ due to a marked packet before the corresponding ECE-ACK, reduces its congestion window and *ssthresh* to one-half of current congestion window. It sends a CWR packet as allowed by the reduced window and waits a full roundtrip time (RTT) after window reduction before it starts increasing the window. An ISQ due to a dropped packet also causes the sender to set its congestion window and *ssthresh* to one-half of current congestion window and follow the TCP congestion control algorithm thereafter. If for some reason the ECE-ACK reaches the sender before the BECN ISQ, sender responds like a pure ECN sender. The sender does not respond to congestion notification more than once per RTT whether due to ISQs, ECE-ACKs or duplicate ACK's. The advantage of using the BECN scheme under a configurable level of congestion is *early notification* especially for congestion levels that lead to packet drops since the sender does not wait for duplicate acknowledgements or a timeout before responding to congestion.

[1]The ECN-Capable Transport (ECT) bit is set in the IP header of an ECN or BECN IP packet [1] for identifi cation at the router.

[2]The Congestion Experienced (CE) bit is set in the IP header of an ECN or BECN IP packet [1] as an indication of congestion.

The ECN+BECN algorithm also requires the use of a single bit to differentiate between an ISQ due to a marked packet and an ISQ due to a dropped packet. An ISQ due to a marked packet would have this bit set so that the sender detects it should wait a full RTT before increasing its window according to the TCP algorithm. When the bit is unset, the sender assumes the ISQ was generated due to a dropped packet. It halves its congestion window and starts increasing the window according to the TCP algorithm similar to ECN.

In [4] we proposed the use of a single bit in the 32 bit unused field in the ICMP source quench packet header for the ISQ differentiation.

V. Simulation Setup

Study and evaluation of the ECN+BECN mechanism is done using the ISI's Network Simulator (NS ver 2.1b8a). In this section, we describe the network scenarios, simulation parameters and performance metrics used in evaluating the algorithm.

A. Simulation Topology

Here we describe the two network topologies (Network 1 and Network 2) used in our experiments.

1) Network 1: Fig. 1 illustrates Network Topology 1. The bottleneck bandwidth is 10Mbps with a propagation delay of 40ms. All other links have 100Mbps capacity with 2ms propagation delay. Nodes Fc(1)..Fc(n) serve as FTP clients with nodes Fs(1)..Fs(n) as corresponding FTP servers. Hosts Wc(1)..Wc(m) serve as web clients with corresponding Ws(1)..Ws(m) hosts serving as web servers. Traffic flow is from servers to clients. The FTP traffic model in the Network Simulator (NS) is used with an infinite amount of data to send. TCP type is NewReno with a data packet size of 1000 bytes and ACK packet size of 40 bytes. The TCP clock granularity is set to 100ms. The maximum TCP congestion window was set to 100KB to ensure that in all experiments the TCP transfers were in the order of the bottleneck bandwidth delay product of the link. Delayed ACKs are not used in the experiments. We follow guidelines by Floyd [10] in setting queue RED parameters as follows: *minth* = 15KB, *maxth* = 3**minth*, *buffer size* = 2**maxth*, *becnthresh* = 20KB and 40KB, *maxp* = 0.1, *wq* = 0.002. Byte-based dropping for RED is used.

2) Network 2: Fig. 2 illustrates Network Topology 2 used in our experiments with a lossy reverse path. The bottleneck bandwidth is 0.5Mbps with a propagation delay of 10ms while the outer links have 10Mbps capacity with 2ms propagation delay. A single sender (Host A) sends traffic to a single receiver (Host B). Data traffic flow is in one direction and this single flow is used as our test flow. The reverse link

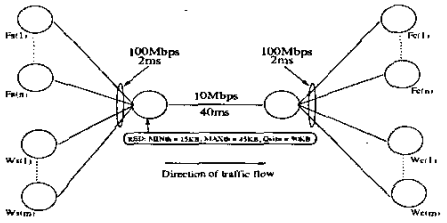

Fig. 1. Network Topology 1

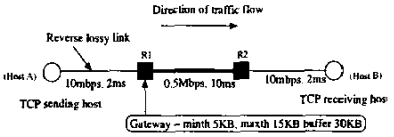

Fig. 2. Network Topology 2

between router R1 and Host A is designed to be lossy, hence causing packet drops even in the absence of congestion. The same FTP traffic model used with Network 1 described above is used. Also following guidelines in [10], RED parameters are set as follows: *minth* = 5KB, *maxth* = 3**minth*, *buffer size* = 2**maxth*, *becnthresh* = 6KB, *maxp* = 0.1, *wq* =0.002. Byte-based dropping for RED is also used here.

B. Performance Metrics

We use the following performance metrics for evaluation:

1) Goodput for a TCP flow: computed based on the number of data packets received by the receiver. The number of ACKs received by the sender within simulation time is used for the computation.

2) Average goodput: For a number of TCP flows this is the average of their individual goodputs.

3) Bottleneck utilization (U): Utilization is computed using the following formula:

$$U = \frac{(8 * 1000 * n) + (8 * 40 * m) + (8 * 40 * p)}{c} \quad (1)$$

where n, m and p are the number of data, ACKs, and ICMP packets respectively that successfully traverse the bottleneck link and are received at the receiver. c is the capacity of the bottleneck link.

4) Web object transfer delay: For a transfered web object, this is the interval between the time a web client makes an initial request (GET message) and the time the server receives the ACK to the last data packet for the object requested by client.

5) Fairness Index for web object transfer delay: For N transfered objects the Fairness Index (FI) [11] for transfer

delay is computed using the following formula:

$$FI = \frac{(\sum_{i=1}^{N} T_i)^2}{N \sum_{i=1}^{N} T_i^2} \quad (2)$$

where T_i is the transfer delay incurred by object i.

6) Percentage loss: Measures the ratio of the number of packets dropped at the bottleneck link to the total number of packets injected into the bottleneck link for a particular flow or set of flows.

7) Percentage ISQ reverse traffic: computed as the number BECN ISQs generated in the reverse direction of data flow as a ratio of the total number of packets in the reverse direction.

VI. Experiments and Results

In this section, we describe four sets of experiments and present explanations for the observed behavior of the ECN+BECN algorithm. The same set of seeds used for the BECN and ECN experiments are used in all random number generators to ensure the ECN+BECN algorithm is tested based on same input variables. In experiment A, B and D we show two results for ECN+BECN each corresponding to simulations using two different values for *becnthresh* (20KB and 40KB) to enable observation of the impact of varying *becnthresh* on TCP performance.

A. Competing long-lived ECN+BECN flows with plain TCP flows - [Fig. 3 - Fig. 4]

In this experiment, a number of either BECN, ECN or ECN+BECN FTP flows compete with an equal number of plain TCP FTP flows. We describe flows as "plain TCP" if they are treated based on pure RED algorithm at the routers while ECN, BECN and ECN+BECN flows are treated based on the ECN, BECN and ECN+BECN algorithms respectively. The total number of competing flows is varied using 8, 12, 16, 18, 20, 26, 30, 36, 40 and 46 flows, to observe performance under different levels of congestion. The start time for all flows is a random variable uniformly distributed between 0s and 5s. Fig. 3 shows the measured gain in average goodput for ECN, BECN and ECN+BECN flows over plain TCP flows while Fig. 4 shows the measured losses.

First, we observe in Fig. 3 that due to early notification BECN and ECN+BECN with 20KB *becnthresh* attain 45.97% and 45.43% gain in average goodput over plain TCP respectively under high congestion (46 flows). ECN and ECN+BECN with 40KB *becnthresh* attain only upto 34.58% gain and 40.48% gain respectively. Second, we observe in Fig. 4 that BECN and ECN+BECN with 20KB *becnthresh* due to early congestion notification experience a loss of only 0.41% and 0.42% respectively under high congestion while ECN and ECN+BECN with 40KB *becnthresh* experience higher losses of 0.57% and 0.46% respectively.

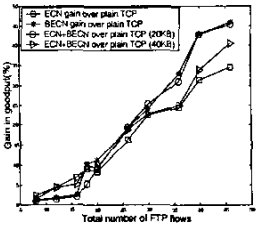

Fig. 3. Average goodput gain over TCP

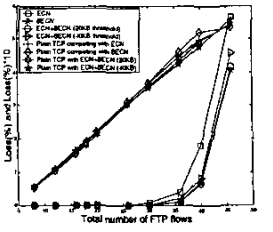

Fig. 4. Percentage loss

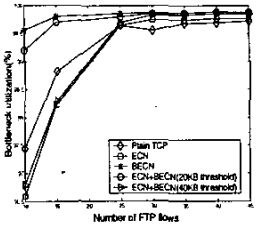

Fig. 5. Bottlebeck utilization

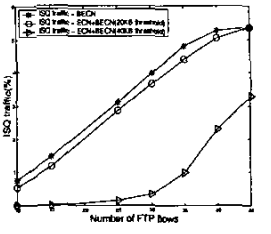

Fig. 6. ISQ reverse traffi c

B. Homogeneous long-lived flows - [Fig. 5 - Fig. 8]

In this experiment, we call the flows "homogeneous" as all flows in a particular experiment are ECN, BECN or ECN+BECN capable. The FTP flows start randomly within the initial 5s of simulation. The number of FTP connections is varied using 10, 15, 25, 30, 35, 40, and 45 flows. We measure the bottleneck utilization, percentage ISQ reverse traffic and queue size. Fig. 5 - Fig. 8 summarizes the observed results. The results show that with a lower *becnthresh* (20KB) the ECN+BECN mechanism performs much like BECN as expected achieving a bottleneck utilization of between 99.2% and 99.9% (Fig. 5). With a higher *becnthresh* (40KB) ECN+BECN behaves more like ECN achieving a lower utilization of between 96.8% and 99.85% (Fig. 5). First, we observe that the percentage ISQ reverse traffic is significantly reduced compared to BECN (Fig. 6) since ISQs are generated only when average queue exceeds *becnthresh*. With 40KB *becnthresh* we measure upto 38.8% reduction (Fig. 6) in ISQ reverse traffic under high congestion - 45 FTP flows however the average goodput and loss percent for ECN+BECN flows show no significant improvement over ECN and BECN for this test case. Second, compared to ECN we observe that due to early ISQ notification ECN+BECN with 20KB *becnthresh* experienced lower variation in queue size (Fig. 7 and Fig. 8).

C. Lossy reverse path - [Fig. 9 - Fig. 12]

This test case investigates the impact of loss of ISQ packets on the behavior of BECN by controlling the loss rate on the reverse traffic path. We use a simple topology - Network Topology 2 (Fig. 2) with a single TCP connection and a lossy

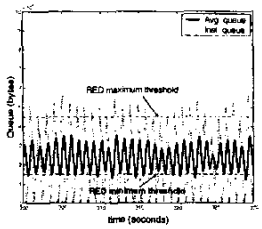

Fig. 7. Queue - 15 ECN FTP

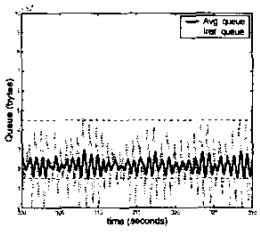

Fig. 8. Queue - 15 ECN+BECN (20KB) FTP

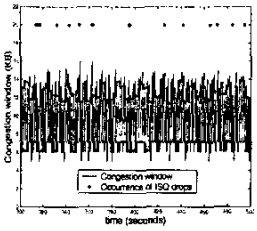

Fig. 11. Effect of ISQ loss on ECN+BECN window - 10% lossrate

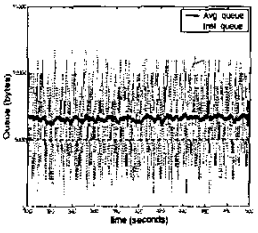

Fig. 12. Effect of ISQ loss on ECN+BECN queue - 10% lossrate

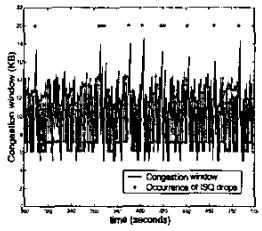

Fig. 9. Effect of ISQ loss on BECN window - 10% lossrate

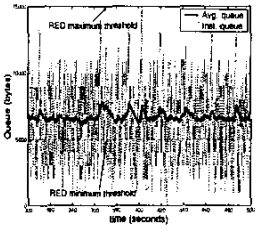

Fig. 10. Effect of ISQ loss on BECN queue - 10% lossrate

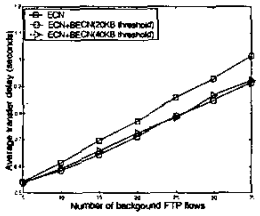

Fig. 13. Average object transfer delay

reverse path to enable detailed observation of the impact of losing reverse flowing packets on TCP performance. Four sets of simulations are done in the experiment. In each simulation, the FTP test flow is either plain TCP, BECN-capable, ECN-capable or ECN+BECN-capable. The loss rate on the lossy link is varied using loss rates of 0%, 2%, 4%, 6%, 8%, 10% to observe performance with different loss rates on the reverse path. The start times for the test flow varies uniformly between 0s and 5s. We measure the average goodput for the test flow, the average queue size for the gateway and monitor the TCP congestion window. Fig. 9 - Fig. 12 summarizes our observed results.

First, Fig. 9 shows that due to loss of ISQs the BECN congestion window experiences temporal surges since congestion notification is not achieved early. Fig. 10 also shows that this effect causes considerable fluctuation in average queue size. ECN however is less severely affected due to reliable delivery of ECN-Echo ACKs. Second, Fig. 11 and Fig. 12 for our experiment with the ECN+BECN mechanism shows that the ECN+BECN scheme enjoys this merit of ECN and hence is not adversely affected by loss of BECN ISQs unlike pure BECN.

D. Short-lived web transfers - [Fig. 13 - Fig. 17]

This test case assesses the performance of ECN+BECN with short-lived web-traffic workloads. In the experiment homogeneous ECN and ECN+BECN web sessions are set up between 10 web servers and 10 web clients, while homogeneous background FTP connections are varied using 5, 10, 15, 20, 25, 30, and 35 flows, to establish different levels of congestion. The FTP flows start randomly within the initial 5s of simulation while the web-traffic connections start after 50s. Short-lived traffic sources were emulated using the built-in web-traffic model in NS with parameters in Table 1. According to Internet traffic studies in [12] the average

TABLE I
WEB TRAFFIC PARAMETERS

Parameter value	Distribution	Mean	Shape
Intersession time	Exponential	0.5s	-
Session size	Constant	10 pages	-
Inter-page time	ParetoII	10s	2
Pagesize	ParetoII	3 objects	1.5
Inter-object time	ParetoII	0.1s	1.5
Object size	Constant	8 packets	-
Number of sessions	Constant	1000	-

size for an HTTP object is between 8KB - 10KB hence we used a constant web object size of 8KB in our experiments with HTTP traffic. Having all objects with same size in a particular experiment permitted measurement of the fairness index for object transfer time. The intersession time, number of sessions, session sizes, and inter-object time were chosen to ensure that several sessions were active throughout simulation time, while other parameters were chosen based on recommendations in [13].

We record the average object transfer delay for the web transfers, fairness in web object transfer times, the percentage loss for HTTP packets, the mean and variance of queue size for the congested router. Fig. 13 - Fig. 17 show the observed results alongside results for ECN. Fig. 14 shows that

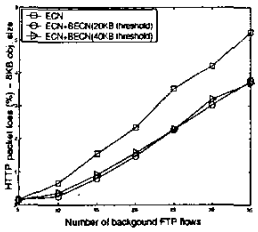

Fig. 14. HTTP packet loss

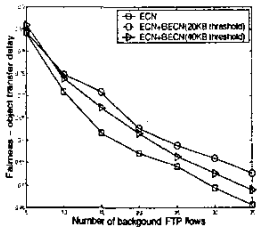

Fig. 15. Fairness - Object transfer delay

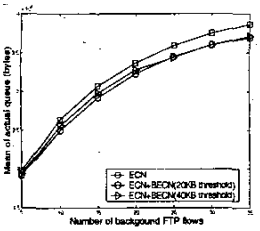

Fig. 16. Mean of queue - web transfers

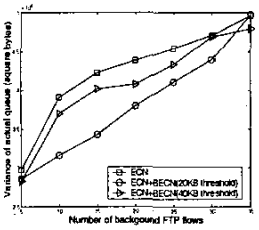

Fig. 17. Variance of queue - web transfers

the burstiness of the web-traffic workloads causes packet loss for both ECN+BECN and ECN connections. Contributors to transfer delay (Fig. 13) include queue size (delay), packet sending rate, and packet loss which leads to TCP timeout and retransmission while variations in queue size contribute to unfairness in average transfer delay. First, we observe that due to ISQ early notification ECN+BECN experiences lower mean queue size with less queue fluctuations (Fig. 16 and Fig. 17) compared to the ECN. Second, in comparison to the ECN+BECN scheme with 20KB *becnthresh*, ECN experiences up to 18.9% greater variance in queue size with 15 background FTP flows (Fig. 17) resulting in 14.8% poorer fairness index (Fig. 15). Third, as a result of a 4.9% higher mean queue size with 15 background FTP flows (Fig. 16) ECN experiences 47.4% greater packet loss (Fig. 14) and 7.47% greater average transfer delay (Fig. 13) compared to ECN+BECN with a 20KB *becnthresh*.

VII. Conclusion and Future Work

In this study we proposed the combination of the ECN and BECN mechanisms to offer a robust congestion notification algorithm for TCP/IP networks. Significant contributions of this paper include the following observations: (1) For long lived TCP flows, there is a significant improvement in goodput for competing flows under heavy congestion. (2) The ECN+BECN scheme can result in significant reduction in ISQ reverse traffic compared to the BECN mechanism. (3) The combined ECN+BECN scheme results in reduced queue size variation as compared to ECN. (4) Unlike BECN, the ECN+BECN mechanism is not adversely affected by the loss of ICMP Source Quench packets on the reverse path. This is due to reliable congestion notification via ECN-Echo ACKs. (5) Finally, we show that for web traffic, the ECN+BECN mechanism can offer improved average transfer delay and fairness compared to ECN.

Therefore, we conclude that a robust mechanism which combines the merits of BECN and ECN algorithms for congestion control in IP networks can result in improved performance over the individual ECN and BECN schemes.

The results in this paper warrant further investigation into the combined ECN+BECN scheme. One such area of future work is to study the impact of different *becnthresh* settings and develop guidelines for its preferred values. Secondly, the use of ISQs opens the possibility of multi-level congestion indication mechanisms where the router can provide a finer granularity notification of congestion. In such a case, the end system response to congestion can be different for different congestion levels - rather than always halving the congestion window. Finally, the ECN+BECN mechanism would benefit from further study, investigation and prototyping in routers and end systems.

References

[1] S. Floyd, Ramakrishnan K, *"A proposal to add Explicit Congestion Notification to IP"*, RFC 2481, January 1999.

[2] S. Floyd, *"TCP and Explicit Congestion Notification"*, ACM Computer Communication review, V.24, N.5, p.10-23, October 1994.

[3] J. Hadi, U. Ahmed, *"Performance Evaluation of Explicit Congestion Notification (ECN) in IP Networks"*, RFC 2884, July 2000.

[4] F. Akujobi, I. Lambadaris, R. Makkar, N. Seddigh, B. Nandy, *"BECN for Congestion Control in TCP/IP Networks: Study and Comparative Evaluation"*, IEEE/Globecom 2002, Taipei, Taiwan, November 2002.

[5] *ftp://ftp.ee.lbl.gov/email/sf.98may7.txt.*

[6] M. Kwon, S. Fahmy, *"TCP Increase/Decrease Behavior for Explicit Congestion Notification (ECN)"*, Proceedings of IEEE ICC (Symposium on High-Speed Networks). volume 4, pp. 2335-2340, April 2002.

[7] A. Rangarajan, A. Acharya, *"Early regulation of unresponsive Best Effort Traffic"*, ICNP 99, Oct 31-Nov 3, 1999, Toronto, Canada.

[8] J. Nagle, *"Congestion Control in IP/TCP Internetworks"*, RFC 896, January 1984.

[9] U. Black, *"Frame relay networks: specifications and implementations"*, McGraw-Hill, NewYork, 1994.

[10] S. Floyd, V. Jacobson, *"Random Early Detection Gateways for Congestion Avoidance"*, IEEE/ACM Transactions on Networking, V.1, N.4, August 1993.

[11] R. Jain, *"The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling"*, Wiley - Interscience, New York, NY, April 1991.

[12] B. Mah, *"An Empirical Model of HTTP Network Traffic"*, Proceedings of INFOCOM '97, Kobe, Japan, April 7-11, 1997.

[13] A. Feldmann, A. C. Gilbert, P. Huang, W. Willinger, *"Dynamic of IP traffic: A study of the role of variability and the impact of control"*, Proceedings of ACM/SIGCOMM, Cambridge, MA, September 1999.