BECN for Congestion Control in TCP/IP Networks: Study and Comparative Evaluation

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

Nabil Seddigh, Biswajit Nandy
Tropic Networks
Kanata, ON, Canada
nseddigh, bnandy@tropicnetworks.com

***Abstract*— This paper evaluates the suitability of Backward Explicit Congestion Notification (BECN) for IP networks. The BECN mechanism has previously been used in non-IP networks, but there has been limited experimental investigation into the application of the BECN scheme as congestion control mechanism in IP networks. In this paper, we consider an enhanced algorithm for BECN which uses Internet Control Message Protocol (ICMP) Source Quenches for backward congestion notification in IP networks and undertake comparative performance evaluation of RED, ECN and our enhanced BECN mechanism using both long-lived TCP bulk transfers and short-lived webtraffic workloads. Our results show that for webtraffic workloads, BECN offers only slight improvement in transfer delay while average goodput for bulk transfers is no worse than that of ECN. For paths that have a high bandwidth delay product our results show that not only can BECN offer significant improvement in average goodput for bulk transfers over the ECN mechanism, but packet drops and transfer delay for short-lived webtraffic connections are also comparatively reduced. Additional observations show that on such paths TCP (NewReno) can offer higher goodput for bulk transfers compared to ECN.**

We investigate the overhead due to Source Quenches in a BECN capable network and find that for scenarios considered in this paper it does not significantly impact performance of BECN.

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

I. Introduction

Over the years, the Internet has evolved into a global heterogeneous network that is used for mission-critical and leisure purposes. Its successful evolution is in part due to flow and congestion control mechanisms that aim to avoid "congestion collapse" [8]. Sender flow control is achieved with TCP conservatively injecting packets into the network based on feedback of the congestion state of the network. The TCP sender considers packet loss as an indication of congestion and relies on three duplicate acknowledgements or a retransmission timeout (RTO) to confirm occurrence of a loss. The sender responds to this feedback by limiting its packet injection rate into the network.

Recently, Random Early Detection (RED) [6] has been recommended [14] as the active queue management scheme for use on the Internet. More recently, 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 [11]. Recent studies show that RED with ECN support gives definite improvement in time delays for interactive traffic over packet drop schemes [9][19].

In this paper, we examine use of a Backward Explicit Congestion Notification (BECN) mechanism to inform the sender of the congestion situation in the network. This mechanism uses Internet Control Message Protocol (ICMP) Source Quenches for backward notification. The ICMP Source Quench (ISQ) messaging was originally standardized as the mechanism of choice for notifying an end host of network congestion [1]. However, while the messaging for ICMP Source Quenches (ISQs) was defined, the conditions for ISQ generation at the router and the appropriate reaction at the end host were not implemented in a standardized way nor could they be considered mature algorithms. RFC 1812 [3] deprecates generation of ISQ messages from a router but specifies that a router that originates ISQs must be able to limit the rate at which they are generated. Floyd in [9], gives due consideration to ISQs as a mechanism for explicitly notifying the TCP sender about congestion. However, for a number of reasons explained in the next section, ISQs were considered unsuitable for congestion notification. Nevertheless, we believe that in particular network scenarios BECN might have an advantage over other congestion notification schemes.

We provide in this paper a modification to an originally proposed BECN algorithm [7], and evaluate performance implication of using the modified BECN with FTP bulk transfers and HTTP-like workloads.

The rest of this paper is organized as follows: Section II discusses the arguments in favor and against use of the BECN mechanism for congestion control. Section III gives a brief overview of previous related work on BECN. Section IV provides our modification to the original 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.

II. Issues with use of BECN for TCP/IP networks

In this section, we discuss issues that have been raised concerning the use of the BECN mechanism for congestion control

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

in TCP/IP networks.

A. Concerns with use of BECN

i) There is concern that no standard algorithm exists for response of a TCP source to an ISQ nor are the conditions for ISQ generation well defined. BECN algorithm in [7] addresses this by defining conditions for ISQ generation and TCP source response. More details are given in subsequent sections.

ii) There is the issue of how network stability would be affected when BECN ISQ messages are lost on the reverse path [10]. We point out that during persistent congestion this condition is no worse than loss of an ECN-Echo ACK [11] in the case of ECN since ISQs continues to be generated irrespective of whether a previous one was sent. The BECN sender only responds once per window. For very temporal congestion situations more work is required to evaluate the impact of loss of an ISQ message.

iii) In [10], the amount of extra reverse network traffic generated by the BECN ISQ messages was a concern. This was a valid concern that existed with drop-tail buffer management as during times of congestion lots of ISQs are generated. However in the BECN proposal [7], BECN ISQs are generated only when the computed RED probability requires a packet dropping or marking. We show in our results that for the scenarios considered in this paper the contribution of ISQs to reverse traffic in a BECN capable network does not significantly impact performance of BECN.

iv) It has been argued that BECN is non-generic for multicasting as there can be receiver or sender based congestion control [10]. However, it has also been pointed out that with sender-based multicast congestion control, the BECN feedback mechanism is more scalable than earlier proposals for feedback control in multicast environments since it is provided by the router not by all the recievers in a multicast session [12].

v) The issue of ATM's "beat-down" problem has also been raised with regards to IP BECN. It is worth pointing out that this is really a problem of traffic that passes through multiple hops versus traffic that passes through fewer hops. This issue is present even for TCP flows in a RED or ECN enabled network and was dealt with in [13]. As a result, it is not considered in this paper.

The concerns with IP BECN need further investigation to better understand them. To date, there has been limited quantitative investigation into the performance of IP BECN in comparison with other IP Congestion control mechanisms.

B. Benefits of using BECN

i) BECN enjoys all the advantages of ECN over TCP with RED. This stems from the fact that for both ECN and BECN, packets are marked probabilistically and not dropped. Such advantages include lower loss rates, reduction in number of TCP timeouts and retransmissions, faster congestion notification, and lower packet delay.

ii) BECN uses existing network layer signalling and does not require the use of any transport layer protocol for congestion notification. It is therefore protocol independent and can be used by other transport protocols such as UDP. Also, there may be value in providing a common mechanism for notifying all transport protocols about congestion.

iii) BECN provides faster congestion notification as compared to the ECN mechanism. This could be particularly useful in networks with large delay such as satellite networks. There is a clear need to investigate the possibility of a BECN advantage in this scenario.

iv) Finally, the BECN scheme allows the development in the future of multi-level congestion feedback schemes. Till now, this has not been possible since both the duplicate ACKs and ECN schemes cannot carry multi-level congestion feedback notification. However, with use of ISQs there is possibility of such a mechanism.

III. Related Work

The ISQ message format was originally defined in [1]. In [2] it is documented that a disadvantage of the ISQ mechanism is that its details are discretionary stating that it is impossible for the end-system user to be sure about the conditions under which the ISQ was generated. RFC 896 [4] discusses in general terms approaches for generating ISQs and reacting to them. Among the approaches is one that considers generating ISQs adaptively before the queue is full. RFC 1016 [5] described Source Quench Introduced Delay (SQuID) where ISQs were to be generated based on threshold levels of the physical queue in the router. Packets are clocked by the sender based on inter-arrival times adjusted in response to the ISQ arrival rate. More recently, [15] explored the the use of Source Quenches for controlling unresponsive sources that inject more than their fair share bandwidth into the network.

There has also been proposals for using BECN within ATM networks [16]. In [17] it was affirmed that though the indiscriminate use of BECN can cause problems in ATM networks, BECN may help reduce the feedback time for paths with large delays.

The proposal for BECN in IP networks [7] provided guidelines for generating ISQs and responding to them in a TCP/IP network. According to [7], a BECN TCP sender responds to ISQ congestion notification by halving its TCP window. We observed that with this proposal the BECN sender also starts increasing its window upon receipt of the next ACK after a window reduction. The immediate reaction of increasing rate of injecting packets into the network makes the proposed BECN algorithm unduly aggressive. We suggest a modification to the BECN algorithm and evaluate its performance.

IV. Enhanced BECN algorithm

In this section, we describe our improved BECN algorithm explaining the guidelines for the behavior of all hosts in a BECN-capable TCP/IP network.

A. Behavior of a BECN-capable router.

If *(arriving packet causes the average queue size to go above maximum RED threshold)*
{
check IP header of packet and 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 and maximum RED thresholds)*{
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) 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 BECN-capable TCP end hosts

BECN TCP end hosts do not need initial end-to-end negotiations to establish BECN capability as the TCP receiver is not involved in the congestion notification mechanism. The TCP sender on receipt of an ISQ due to a marked packet sets its congestion window and *ssthresh* to one-half of current congestion window and waits a full roundtrip time (RTT) before it starts increasing the window. An ISQ due to a dropped packet causes the sender to set its congestion window and *ssthresh* to one-half of current congestion window and follows the TCP congestion control algorithm thereafter. The sender does not react to ISQs more than once per RTT.

Our suggested algorithm introduces some modifications to the original BECN algorithm [7]. It ensures that the BECN sender is not unduly aggressive by creating a delay of one RTT before sender starts to increase its congestion window after a window reduction. This gives the network time to alleviate the state of congestion before packet injection rate is increased.

The modified algorithm 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. The advantage for BECN here is early notification of packet drops since it does not wait for duplicate acknowledgements before responding to congestion.

We propose the use of a single bit in the 32 bit unused field in the ICMP source quench packet header for the ISQ differentiation. A bit within the fifth octet of the header can be used.

[1] The ECN-Capable Transport (ECT) bit is set in the IP header of an ECN or BECN IP packet [11] for identification at the router.

[2] The Congestion Experienced (CE) bit is set in the IP header of an ECN or BECN IP packet [11] as an indication of congestion.

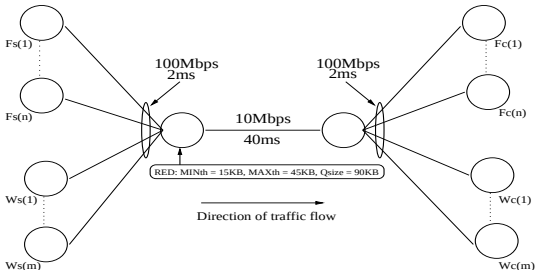

Fig. 1. Network Topology

V. Simulation Setup

Study and comparative evaluation of the modified BECN and ECN mechanisms is done using the Network Simulator (ver 2.1b8a) [21]. In this section, we describe the network scenarios, simulation parameters and performance metrics used in evaluating the algorithms.

A. Simulation Topology

Fig. 1 illustrates a single bottleneck topology that we use in most of our simulations. The bottleneck bandwidth is 10Mbps with a propagation delay of 40ms. All other links have 100Mbps capacity with 2ms propagation delay. Link capacities of 10Mbps and 100Mbps are common in Local Area Networks while the link propagation delays were chosen to ensure that the bandwidth delay product permitted the range of network loading used in our experiments. 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 web servers. Traffic flow is from servers to clients.

B. Queue parameters

RED queue management was used in all simulations with or without support for ECN or BECN. We follow guidelines by Floyd [6] in setting the RED parameters as follows: *minth* = 15KB, *maxth* = 3**minth*, *buffer size* = 2**maxth*, *maxp* = 0.1, *wq* = 0.002. We use byte-based dropping for RED.

C. Traffic sources

In our experiments two types of traffic sources were used:

1) Long-lived TCP traffic sources: 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. Though a number of older systems use 500ms as TCP clock granularity, most current systems have no problem coping with 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.

2) Short-lived web-traffic sources: Here we use the built-in web-traffic model in NS with parameters in Table 1.

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	ParetoII	12 packets	1.2
Number of sessions	Constant	1000	-

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 [18].

D. 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) 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. In the NS webtraffic model, since the TCP three-way handshake is not implemented, computing the object transfer time requires identification of the two seperate flow pairs used in NS to represent the GET flow and the data transfer flow.

4) 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.

VI. EXPERIMENTS AND RESULTS

In this section, we describe five sets of experiments and present explanations for the observed behavior of the algorithms (RED, ECN and the modified BECN). All simulations

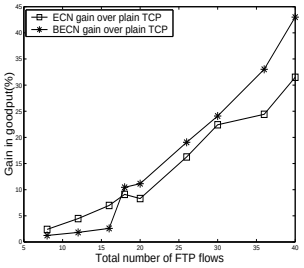

Fig. 2. Goodput gain over TCP

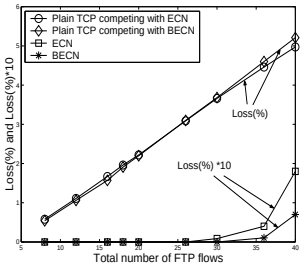

Fig. 3. Percentage loss

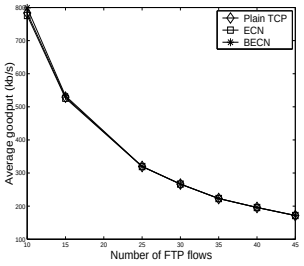

Fig. 4. Goodput - Homogeneous flows

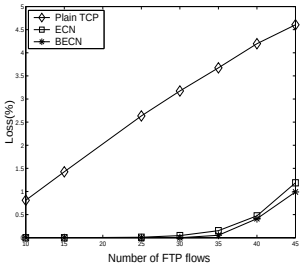

Fig. 5. Loss - Homogeneous flows

are run for 500s and data for results were collected after a period of 100s to avoid bias due to initial conditions. For each of the algorithms, the same seed was used in all random number generators to ensure all algorithms are tested based on same input variables.

A. Competing long-lived BECN/ECN flows with plain TCP flows - [Fig. 2 and Fig. 3]

This experiment captures the scenario where some Internet users decide to use either the BECN or ECN algorithm for congestion control in a Internet where RED is widely deployed. We describe flows as "plain TCP" if the are treated based on pure RED algorithm at the routers. ECN and BECN flows are treated based on the ECN and enhanced BECN algorithms respectively. In this experiment, a number of either BECN or ECN FTP flows compete with an equal number of plain TCP FTP flows. The total number of competing flows is varied using 8, 12, 16, 18, 20, 26, 30, 36 and 40 flows (i.e. n = 8, 12, 16, 18, 20, 26, 30, 36, 40 in section 5A), 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. 2 shows the measured gain in goodput for BECN and ECN over plain TCP flows.

Fig. 2 shows that the BECN algorithm offers upto 43% gain in average goodput under high congestion over plain TCP flows. Recall that unlike plain TCP packets, the BECN packets are not dropped probabilistically but marked. As congestion increases plain TCP flows experience more packet drops while the BECN flows continue to send packets successfully across the network. Under high congestion while BECN suffers a 0.07% packet loss, plain TCP flows suffer 5.21% loss

(Fig. 3) - thus, higher goodput for BECN. For same reasons, ECN offers upto 31.5% gain in average goodput during high congestion with a loss of 0.18%. BECN therefore offers greater gain over plain TCP during high congestion as *it suffers less losses due to early notification compared to ECN*.

However, during very low congestion we observe that ECN offers higher gain over plain TCP compared to BECN. The reason is that during low congestion, the dropping and marking probability is small, therefore plain TCP flows not only experience less drops but also do not respond to packet drop notification early since they have to wait for duplicate acknowledgements. On the other hand, BECN senders are quenched quickly and this reduces their average goodput in this case. ECN senders in this scenario are also quenched but not as early as the BECN senders since they have to wait more than half RTT before responding to congestion notification due to a marked packet. The advantage of BECN's early notification become more obvious under higher congestion as explained above.

B. Homogeneous long-lived flows - [Fig. 4 - Fig. 8]

In this experiment, we call the flows "homogeneous" as all flows in a particular experiment are either plain TCP, ECN capable or BECN capable. Three sets of simulations are done in the experiment. In each simulation, FTP flows are either all plain TCP, all ECN-capable or all 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 (i.e. n = 10, 15, 25, 30, 35, 40, 45; m = 0 in section 5A). Average goodput for the TCP flows is computed. Fig. 4 - Fig. 8 summarizes the observed results. We noted that though plain TCP homogeneous flows experience higher losses - 4.6% (Fig. 5), their average queue size (31392 bytes) remains less than that of BECN (39247 bytes) or ECN (40034 bytes) under high congestion (45 background flows). This is because arriving plain TCP packets are dropped probabilistically not marked as with BECN and ECN packets. However, the plain TCP flows achieve between 97.44% and 99.7% utilization on the bottleneck link (Fig. 6), and their average goodput remain comparable to that of homogeneous BECN and ECN flows (Fig. 4). This is due not only to the improved TCP NewReno fast recovery mechanism which ensures that when multiple packets are lost from a single window of data, TCP can recover without a retransmission timeout [20], but also because all flows in individual sets of simulations are same type and therefore equally competing for available bandwidth.

However, we observe that due to earlier notification BECN flows have lower average and instantaneous queue size with 15 background flows (Fig. 7 and Fig. 8) compared to ECN flows. In this case, even though packets are not dropped for either BECN or ECN flows (Fig. 5), the BECN flows achieve higher bottleneck link utilization (Fig. 6) since the probability of quenching BECN flows due to marked packets (which is proportional to average queue size) is less than that of ECN. With higher congestion (45 background flows), BECN flows

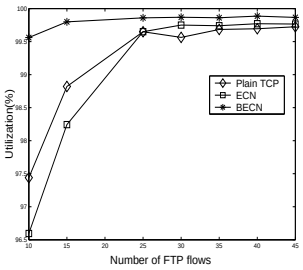

Fig. 6. Bottlebeck utilization

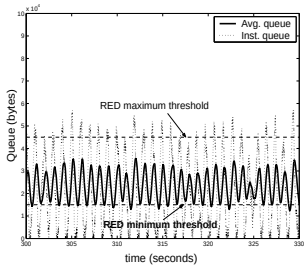

Fig. 7. Queue - 15 ECN FTP

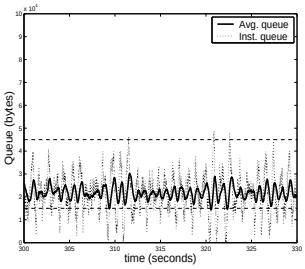

Fig. 8. Queue - 15 BECN FTP

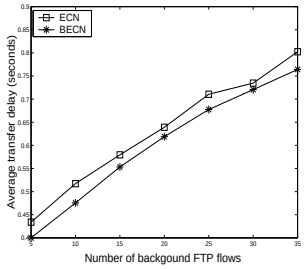

Fig. 9. Average object transfer delay

suffer slightly lower losses - 0.99% compared to the ECN flows - 1.19% (Fig. 5).

C. Short-lived web transfers - [Fig. 9 - Fig. 12]

This test case assesses the performance of BECN and ECN with short-lived web-traffic workloads. In this experiment, there are either 10 BECN or ECN capable web servers and 10 BECN or ECN capable web clients, while homogeneous BECN or ECN background FTP connections are varied using 5, 10, 15, 20, 25, 30, and 35 flows (i.e. n = 5, 10, 15, 20, 25, 30, and 35; m = 10 in section 5A), 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. We record the average object transfer delay for the web transfers. Fig. 9 - Fig. 12 show the observed results. Fig. 10 shows that the burstiness of the web-traffic workloads causes packet loss for both BECN and ECN connections especially during high congestion. Contributors to transfer delay Fig. 9 include queue size (delay), packet sending rate, and packet loss which leads to TCP timeout and retransmission. Fig. 11 and Fig. 12 show

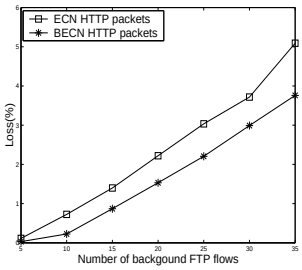

Fig. 10. HTTP data packet loss

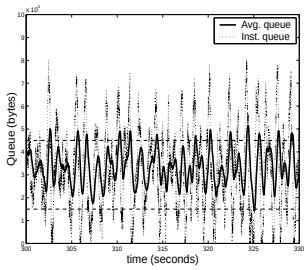

Fig. 11. Queue - ECN web + 15 FTP

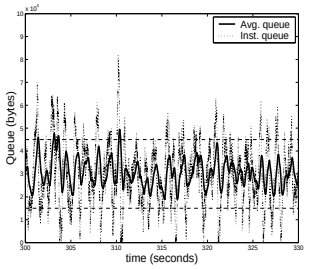

Fig. 12. Queue - BECN web + 15 FTP

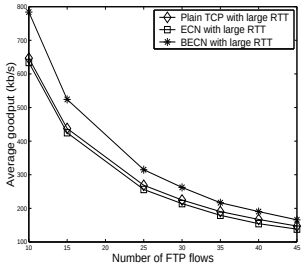

Fig. 13. Goodput - Homogeneous flows

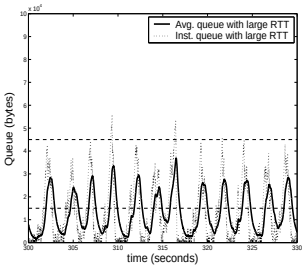

Fig. 18. 45 BECN FTP (large RTT)

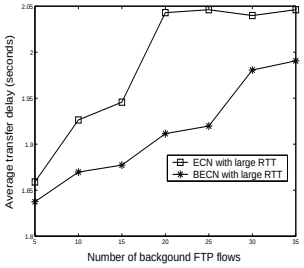

Fig. 19. Average object transfer delay

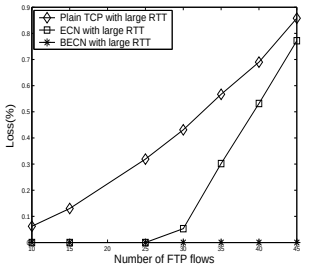

Fig. 14. Loss - Homogeneous flows

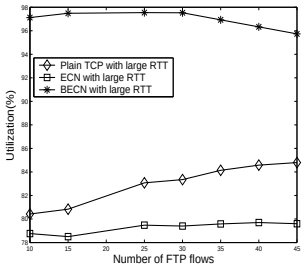

Fig. 15. Bottlebeck utilization

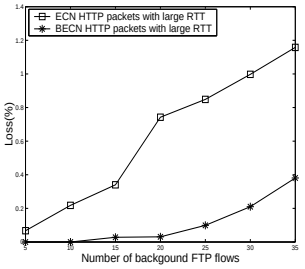

Fig. 20. HTTP data packet loss

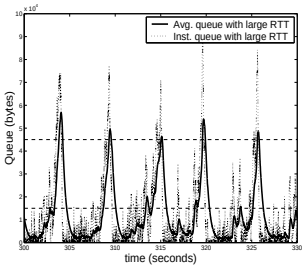

Fig. 21. 25 ECN FTP + web (RTT)

that the queue size for BECN is slightly lower than that of ECN with 15 background FTP flows. For short-lived connections, the BECN advantage of early notification would be enjoyed only if the BECN ISQ reaches the sender while there are still a good number of outstanding packets to be sent. The occurrence of such a situation clearly depends on the all the factors that determine the network condition at that particular time. We observe that while ECN suffers 5.09% loss with 35 background flows, BECN suffers 3.76% loss (Fig. 10). BECN also offers upto 8.1% reduction in the average web object transfer time (with 10 background flows) compared to ECN (Fig. 9).

D. Large round-trip times (RTT) - [Fig. 13 - Fig. 22]

In this section, the experiments in section 6B and section 6C are repeated using plain TCP, ECN and BECN test flows with propagation delay on the bottleneck link increased to 250ms. Roundtrip delays of 500ms are common with geostationary satellite links. Fig. 13 - Fig. 22 summarizes the observed results.

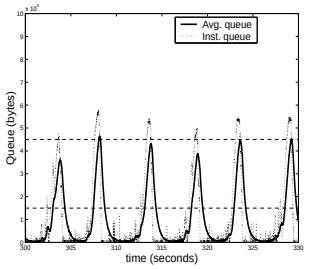

Fig. 16. 45 Plain TCP FTP (large RTT)

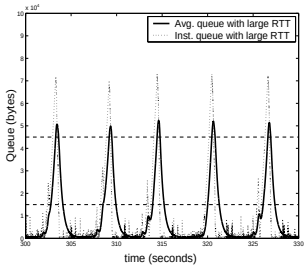

Fig. 17. 45 ECN FTP (large RTT)

For experiments with only homogeneous long-lived flows, we observe that due to early notification, the BECN flows are able to maintain their queue size below maximum RED threshold (Fig. 18), resulting in a 0% packet loss with 45 background flows (Fig. 14). The ECN flows wait more than half RTT before responding to congestion thus we observe that their queue size occassionally exceeds the maximum threshold (Fig. 17) resulting in 0.77% loss (Fig. 14). In this scenario, the BECN algorithm therefore offers better network utilization (Fig. 15) and 20% gain in average goodput over ECN (Fig. 13) under high congestion (45 background flows). Recall that packet marking probability is proportional to average queue size and halving the congestion window reduces packet injection rate which directly affects utilization of the bottleneck link. We also observe in this scenario that plain TCP NewReno flows have lower average and instantaneous queues compared to ECN flows (Fig. 16 and Fig. 17) and achieve higher utilization on the bottleneck link (Fig. 15). Though plain TCP flows experience slightly greater packet drops - only 0.86% (Fig. 14), they achieve average goodput 6.6% higher than that of ECN flows (Fig. 13) with 45 background flows. In addition to the higher probability of quenching the ECN flows due to higher queue size, the TCP flows enjoy the NewReno fast recovery advantage and even have a small probability of experiencing packet drops due to their lower queue size (Fig. 16).

For experiments with short-lived web transfers, we observe that for same reasons as explained above the average queue for BECN is less than that of ECN with 25 background flows (Fig. 21 and Fig. 22) and BECN flows suffer packet losses 88.3% less than ECN flows (Fig. 20). The BECN algorithm in this case offers an average transfer delay 6.4% less than that of ECN with 25 background flows (Fig. 19).

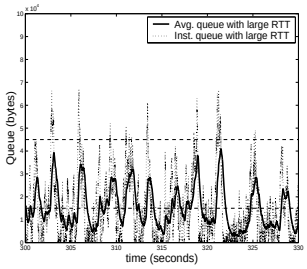

Fig. 22. 25 BECN FTP + web (RTT)

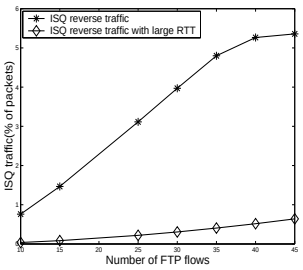

Fig. 23. ISQ reverse traffic

E. ISQ reverse traffic analysis - [Fig. 23]

In this section, the experiments in section 6B with only homogeneous FTP flows are repeated using 40ms propagation delay on the bottleneck link. The experiments are re-done with propagation delay on the bottleneck link increased to 250ms. The ratio of the number of ISQs generated to the total number of packets received in the reverse direction of traffic is computed in each case. Fig. 23 shows that with 40ms bottleneck link propagation delay the ratio of number of ISQs generated under high congestion is only 5.4% of total number of packets in reverse direction. ICMP sourcequench packets are 56 bytes in size (including the IP header) for IPV4 and would therefore consume even less percentage of reverse bandwidth compared to much larger TCP packets in a real life network. Fig. 23 also shows that with larger roundtrip time the computed ratio of number of generated ISQs under high congestion is much less - only 0.64%. This is because with an increase in bandwidth delay product, the bottleneck utilization and average queue size reduces, and this directly impacts probability of marking packets. Our network topology is by no mean representative of the Internet and so further work with more realistic network scenarios is required to better understand the impact of ISQ reverse traffic on BECN performance.

VII. Conclusion and Future work

We have suggested modifications to a proposed BECN algorithm for IP [7] and used simulations to explore the performance of our modified BECN algorithm. The results in this paper suggest that the BECN mechanisn for congestion control in TCP/IP networks can offer comparable performance to that of the ECN mechanism for both long-lived and short-lived TCP transactions. On links that have a high bandwidth delay product, BECN can offer improved performance compared to the ECN mechanism in terms of average goodput for long-lived traffic and packet loss, transfer delay for short-lived workloads.

Therefore, we conclude that the BECN mechanisn is a viable scheme for congestion control in TCP/IP networks and can be used to offer improved quality of service on links with a high bandwidth delay product.

For future work, we are interested in further examing the performance of our modified BECN in the presence of multiple congested gateways and two-way bulk TCP transfers. We are also interested in evaluating multiple-level BECN (MECN) under a variety of scenarios and workloads. Investigating a mechanism that combines both the ECN and BECN mechanism to give a more robust scheme for congestion control is also under consideration.

References

[1] J. Postel, *"Internet Control Message Protocol"*, RFC 792, September 1981.

[2] A. Mankin, K. Ramakrishnan, *"Gateway Congestion Control Survey"*, RFC 1254, August 1991.

[3] J. Baker, *"Requirements for IP Version 4 Routers"*, Internet RFC 1812, June 1995.

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

[5] W. Prue, J. Postel, *"Something a Host Could do with Source Quench: The Source Quench Introduced Delay (SQuID)"*, Internet RFC 1016, July 1987.

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

[7] S. J. Hadi, B. Nandy, N. Seddigh, *"A proposal for Backward ECN for the Internet Protocol (Ipv4/Ipv6)"*, Internet Draft <draft-salim-jhsbnns-ecn00.txt> July 1998. Available at *http://www.sce.carleton.ca/∼seddigh/publications/draft-salim-jhsbnns-ecn-00.txt.*

[8] V. Jacobson, M. J. Karels, *"Congestion Avoidance and Control"*, In Proceedings of ACM SIGCOMM '88, Stanford, CA, August 1988.

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

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

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

[12] A. Matrawy, I. Lambadaris, C. Huang, *"Multicasting of Adaptively-encoded MPEG4 on a QoS-aware IP Networks"* In Proc. of IEEE ICC 2002.

[13] S. Floyd,*"Connections with Multiple Congested Gateways in Packet-Switched Networks Part 1: One-way traffic"*, Computer Communication Review, Vol. 21, No. 5, pp. 30-47, October 1991.

[14] B. Braden et al, *"Recommendations on Queue Management and Congestion Avoidance in the Internet"*, RFC 2309, April 1998.

[15] A. Rangarajan, A. Acharya,*"Early Regulation of Unresponsive Best-Effort Traffic"*, ICNP 99, October 1999.

[16] P. Newman,*"Traffic Management for ATM Local Area Networks"*, IEEE Communications Magazine, Vol. 32, No. 8, pp. 44-50 August 1994.

[17] R. Jain, S. Kalyanaraman, R. Viswanathan, *"The OSU Scheme for Congestion Avoidance in ATM Networks: Lessons Learnt and Extensions"*, Issue on Traffic Control in ATM Networks, Vol. 31, November 1997.

[18] 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.

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

[20] K. Fall, S. Floyd, *"Simulation-based Comparisons of Tahoe, Reno, and SACK TCP"*, Computer Communications Review, July 1996.

[21] UCB/LBNL/VINT Network Simulator (NS). Available at *http://www.isi.edu/nsnam/ns*