SIMULATION-BASED STUDY OF ECN PERFORMANCE IN RED NETWORKS

Marek Małowidzki
Military Communication Institute
05-130 Zegrze, Poland
malowidz@wil.waw.pl

Keywords: Explicit Congestion Notification (ECN), TCP, congestion control, IP networks, ns-2

Abstract

Explicit Congestion Notification is a recently proposed congestion avoidance scheme for IP networks. ECN uses marking packets instead of dropping them in case of incipient congestion. Avoiding packet drops saves network bandwidth and allows congestion signals to be propagated faster. In the paper, we present the results of detailed simulations that we have performed to assess the gain TCP applications could achieve from using ECN. We analyze both a single congested link and a more complex configuration. We also discuss head marking, a modification of router's behavior that allows to faster propagate congestion signals.

I. INTRODUCTION

Internet routers have been traditionally dropping packets arriving to a full queue. This approach is known as the TailDrop method. It was shown in many works that TailDrop does not perform well. First, it allows queues in routers to remain full for long periods of time, which increases delays [7] and may cause unnecessary retransmissions. Second, the so-called Global Synchronization phenomenon was observed [7], when due to tail drops multiple TCP flows backed off at the same time, causing link underutilization and resource wasting. Finally, TailDrop may seriously impair fairness between TCP flows [7] (the so-called Lock Out effect).

To overcome problems with TailDrop, active queue management (AQM) methods were proposed. AQM mechanisms detect congestion before router queues overflow and try to prevent the state of heavy congestion. One of AQM mechanisms is Random Early Detection (RED) [1].

A RED router maintains two levels of thresholds: the minimum one (*minth*) and maximum one (*maxth*). When the average queue size stays below *minth*, no action is performed. When the average queue size is above *minth* but below *maxth*, packets are dropped probabilistically, with dropping probability increasing linearly from 0 to a configured maximum drop probability, *maxprob*. If the average queue size exceeds *maxth*, all packets are dropped. Since RED starts dropping packets before congestion becomes severe, the Global Synchronization effect is avoided.

Explicit Congestion Notification (ECN) [4] refines this mechanism and uses packets marking instead of dropping when queue remains between *minth* and *maxth*. The avoidance of packet drops is especially important for reliable transports, such as TCP, since every lost packet must be retransmitted. In an ideal, marking-based network, drops could be completely eliminated and congestion avoidance would be based on a cooperative actions of responsive sources.

In the paper, we use simulations to verify the gain an IP network could get from ECN deployment. The paper is organized as follows. In section II, we describe ECN in greater detail. In section III, we present earlier works related to ECN performance. The subsequent sections describe our simulations and results achieved for both single link and multiple links topology. In the next section, we describe and assess *head marking*, a modification of router's behavior that allows to faster propagate congestion signals. The final section contains conclusions.

II. EXPLICIT CONGESTION NOTIFICATION

ECN mechanism ([4], see also Fig. 1) uses two bits in the IP header (in the former ToS byte, now redefined by DiffServ to be the DSCP field) and two (or, as proposed in [6], three) bits in the TCP header. The two IP bits carry information about whether the flow to which the packet belongs is ECN-capable (the two equally valid *ECT codepoints*) and, if it is, whether congestion has been experienced (the CE codepoint). The two TCP header bits, ECE (ECN-Echo) and CWR (Congestion Window Reduced), are used to inform the sender about congestion or whether the receiver has reduced its congestion window, respectively. Moreover, these bits are used during TCP connection setup phase to

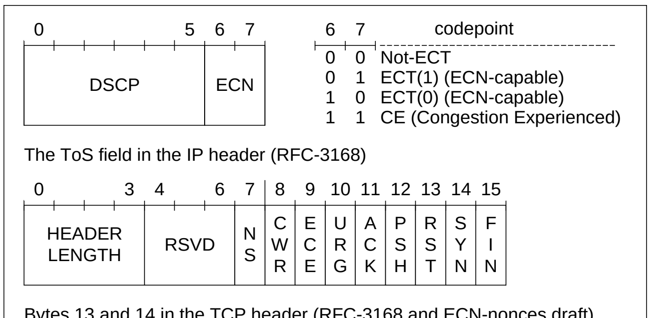

Fig. 1: The new definitions of some fields in the IP and TCP headers

negotiate the willingness to use ECN. The third proposed bit, NS (Nonces Sum), is an additional protection against malicious ECN endpoints (which do not cooperate in ECN).

For ECN to work it is required that TCP endpoints and routers cooperate (Fig. 2). Routers using ECN RED in the case of incipient congestion mark ECN-capable packets (with one of the ECT codepoints set) or drop non-ECN-capable packets (with the Not-ECT codepoint). Packets already marked are simply forwarded. If the TCP endpoint receives a marked packet, it sets the ECE bit in TCP acknowledgements traveling back to the sender until it receives a packet with the CWR bit set. On the other hand, if the TCP endpoint sees the ECE bit set, it should react in the same way as in the case of a single packet being dropped, i.e. halve its congestion window *cwnd* and reduce the slow start threshold *ssthresh*.

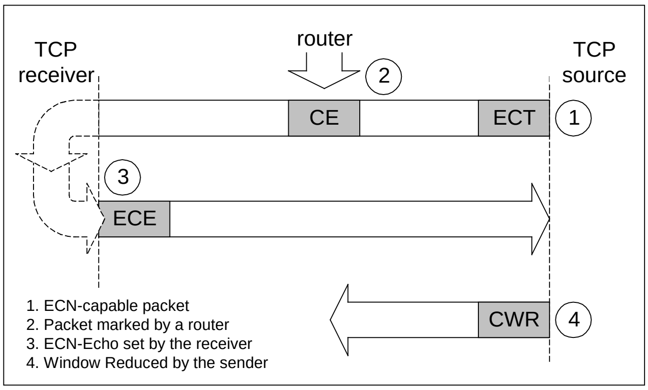

Fig. 2: ECN in action.

ECN starts to be supported in TCP implementations as well as in networking equipment. For example, Linux kernel 2.4 or later has full support for ECN; Cisco routers also support ECN from IOS Release 12.2(8)T. On the other hand, there exist both software and hardware incompatible with ECN, e.g., some firewalls disallow packets with ECN-related flags set [14].

III. RELATED WORK

The immediate observation is that ECN allows the avoidance of packet drops (at least while congestion is still relatively mild). Reliable transports, such as TCP, have to retransmit lost packets, so dropping packets wastes bandwidth. Moreover, with ECN, TCP endpoints can distinguish between packet loss due to transmission errors and congestion signals – this could be especially useful in networks with relatively high bit error rates (e.g., radio networks). Besides, IP-level marking could cooperate with link-level ECN techniques (e.g., EFCI bits in ATM). Finally, marking is more flexible than dropping and could carry different meaning (e.g., some works proposed that a stream of marks be interpreted as a price for sending traffic).

The advantages of ECN have been verified in [2], [5] and [12]. In [2], one set of tests employs a single WAN link. The results show that ECN improves longest packet delays over drop-based TCP Tahoe and Reno while throughput remains similar.

In [5], Linux boxes with experimental ECN implementations were used to verify that ECN improves both bulk and transactional TCP traffic when compared with TCP NewReno and SACK (Selective Acknowledgment, [9]); the gain was more significant for short connections (so-called mice). In [12], fast Cisco routers were employed. The results were less obvious but generally for a larger topology ECN performed better. However, in both cases, a *mix* of ECN-capable and non-ECN-capable flows was examined; in most cases, ECN flows performed better. In this paper, we are rather interested in a situation when ECN is widely deployed. Thus, we compare a scenario with *all flows* using ECN against a pure non-ECN case.

In [11], we tested a user-level gain from ECN. We used the FTP service to measure goodput difference for a single congested link in two cases, when all flows used ECN or all used plain, drop-based TCP/SACK. That test employed Cisco routers with IOS Release 12.2(8)T, with support for ECN, and ECN-enabled Linux boxes. To our surprise, we observed no practical difference in both cases. That is, from applications' point of view, it did not matter if ECN was in use. However, we admit that we checked only a limited set of parameters and a very simple topology (similar to that from Fig. 3).

Also in [11], we discussed security issues related to ECN. ECN opens a new hole in the security of the Internet, since congestion signals may be ignored by malicious TCP endpoints. It remains open to discussion of how serious this hole is (given existing TCP's security flaws [8]); anyway, we showed in that work, using a modified Linux kernel, that malicious FTP sessions (ignoring marks) could effectively throttle compliant ones, regardless of whether the compliant sessions used ECN or not. An interesting result was that short-lived ECN FTP sessions were relatively little vulnerable to subverting, but all other kinds of compliant traffic suffered badly.

IV. SIMULATION DETAILS

We performed the simulations using the ns-2 simulator [15] version 2.26. During the simulations, we started multiple FTP sessions (which in fact were just plain TCP flows, as FTP implementation in ns-2 poorly emulates the real FTP service [10]). The sessions were started randomly within the first 0.5 s. of the simulation (which, together with random session pause times, was sufficient to cause the results differ significantly). All tests employed FullTCP implementation with SACK, since this TCP version performs best in most scenarios [3]. TCP segment size was set to 1492 bytes (802.3 LAN MTU). To closely emulate multiple repeated sessions, the *close_on_empty_* option was used – that is, the server closed the session immediately after the client had acknowledged the reception of the whole file. The session was restarted after a short random delay (up to 50 msec.). Each session used a new couple of available port numbers. Additionally, we used UDP flows to limit available link bandwidth; these flows were alternatively active and disabled for 10 second periods. When enabled, they generated 100-byte packets at a rate set to a given part of link bandwidth (usually, 0.25).

Access links had significantly bigger bandwidth (10Mb/s) than the router links (512kb/s) and used the TailDrop discipline while router links used RED with the same settings in both directions. Every single test for a given set of parameters lasted for 5 min. of simulated time and was repeated 5 times to get average values. We performed tests for both bulk (long files) and transactional (short files, a few kBytes) transfers. We were mostly interested in the following values: the total number of packets dropped (or marked) by routers, the average queue length on the most congested interface and finally the aggregate goodput (i.e., the amount of TCP data successfully acknowledged by a TCP client; duplicated packets and packets in transit at the end of simulation were not taken into account) achieved by the TCP connections.

V. SINGLE CONGESTED LINK

In this section, we present the results of the first set of tests, which were performed for a single link topology (Fig. 3). In this test, a number of FTP sessions were started from the server (ftp0) to both clients ftp3 and ftp4. We were observing the output queue of router gw1 at the link to router gw2. The values of parameters are shown in Table 1; we simulated all possible combinations of these parameters (thus, 256 single tests were performed). In this way, we were able to check multiple cases and multiple levels of congestion, from no congestion (no packet drops, no marks) to a heavily congested network, with multiple packets dropped or marked by routers. Also, even if congestion was so severe that RED could degenerate to TailDrop [5], we still expected ECN to work no worse that non-ECN.

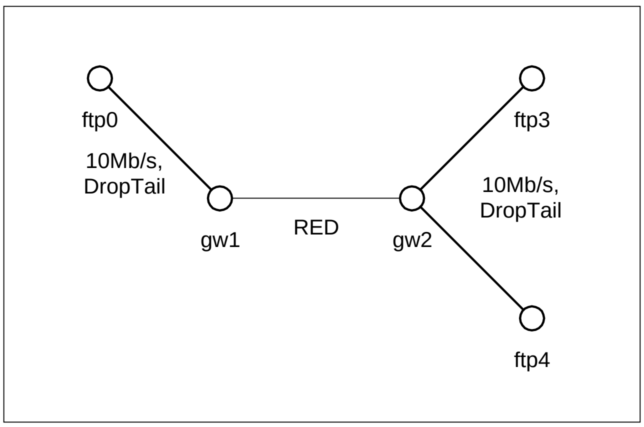

Fig. 3: The topology with a single congested link

	parameter name	**description**	**values**	
link	bandwidth	link bandwidth	512 kb/s	
	delay	link delay	10 ms	25 ms
RED	linterm	1/maxprob	5	10
	minth	minimum threshold	10	
	maxth	maximum threshold	60	120
	q_weight	smoothing factor	0.002	0.02
	limit	queue size	200 packets	
traffic	numftp	number of sessions started between every node pair	6	12
	fsize	file size	10000	1000000
	cbr	rate set to this link bandwidth part	0	0.25
	cbrPeriod	on/off period length	10 s.	
	ecn	enabled ECN	disabled	enabled

Table 1: Parameter values

The results are presented in Fig. 4. The three curves present the relative increase of the aggregate number of marks (and drops), aggregate goodput of all sessions and average queue length. 128 cases were compared with a given set of parameters and ECN disabled or enabled. For all of the three parameters, the value shown in the figure is the value for the ECN case divided by the sum for both cases. That is, the value around 0.5 means that there was no difference, the value of 0,524 means a 10% increase in the ECN case, and the value of 0.545 means a 20% increase. Note that the same number on the x axis usually means a different attempt for all three curves.

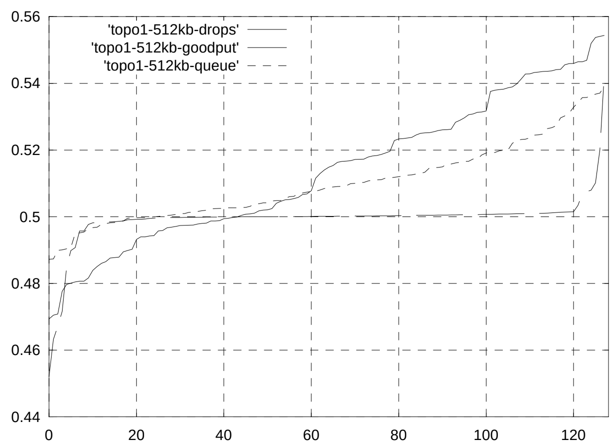

Fig. 4: The gain from ECN: the aggregate number of drops/marks, the aggregare goodput and the average queue length

From this test, we could infer the following observations. First, ECN yields longer queues and more packet marks/drops. This could be easily explained in the following way: When a packet is dropped, it disappears and does not affect the network anymore. However, a marked packet is still inserted into a queue and it affects the queue length. We observed this phenomenon also in our earlier work [11]: an ECN router usually made more marks in the ECN case than drops in the non-ECN case.

There were just about a few runs for which there was a significant difference in bandwidth. Interestingly, sometimes ECN performed noticeably better, and sometimes – significantly worse. The cases that showed a difference above (around) 3% were 13 bulk transfers with heavy congestion (12 sessions in all cases but one between each node pair); CBR traffic was not present in any case. The difference lied in the value of *maxth*: In all non-ECN cases *maxth* was set to 120 while for the winning ECN cases it was set to 60. There was one exception, however: the most successful ECN case used *maxth* set to 120. A possible explanation is that RED with *linterm* 5 and *maxth* 120 behaves (assuming the operation in the "RED region") similarly to a queue with *linterm* set to 10 and *maxth* 60. A general conclusion is that more aggressive RED settings favor ECN, although from the numbers one can see that a more complex relationships reveal. However, these numbers should be taken with care, as for long transfers drops (not marks) dominated in both ECN and non-ECN cases.

CBR traffic seriously impaired performance; ECN did not help in this case, even if the number of marks significantly exceeded the number of drops (for the ECN case, when the average queue length exceeded *maxth*). Even when a test mostly operated in the "RED region," and for the ECN cases the number of marks significantly exceeded the number of drops, there were no differences in goodput (although the queue was usually longer).

A careful conclusion would be that enabling ECN results in increasing the number of drops and marks, lengthen the queues, but does not affect bandwidth except for a few cases. (Maybe, the gain resulting from smaller number of drops is diminished by the fact that there are more congestion signals.) There results are somewhat counterintuitive but this is what the numbers say.

VI. MORE COMPLEX TOPOLOGY

In this test, we used the topology shown in Fig. 5. We started a number of FTP sessions between the following node pairs: ftp0 and ftp3, ftp0 and ftp4, ftp3 and ftp5, ftp4 and ftp6 (the first node was the server and the second node was the client). Additionally, in some tests we enabled UDP traffic from node 3 to node 4. Parameters were the same as in the previous section.

Note that in this topology we could expect ECN to show even bigger advantage over drop-based TCP, since unlike in the single link case, when packets were dropped at the entrance to the network, some packets will be dropped after they have successfully gone through the first congested link (and used some network resources). The results are shown in Fig. 6 (the queue observed is the most congested one, at the link between gw1 and gw2).

The difference in bandwidth is smaller than previously and the reasons are even more difficult to explain. Simply, for a very small number of bulk transfers, noticeable differences (between 5% and 10%) were observed. A more significant tendency was observed for the number of drops and marks: The decrease in the number of drops and marks in the ECN case was observed for bulk transfers. Interestingly, for the transactional transfers, the situation was completely reversed: Among 60 cases with higher drop/mark values for the ECN case, there was just one bulk transfer. Average queue decrease was observed for bulk transfers while for transactional transfers the queue was longer.

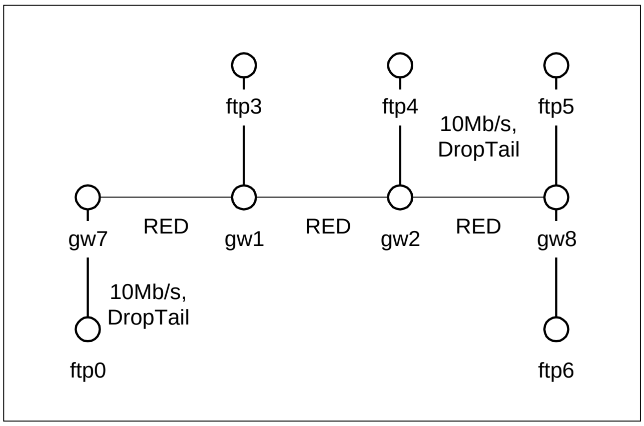

Fig. 5: The topology with three congested links

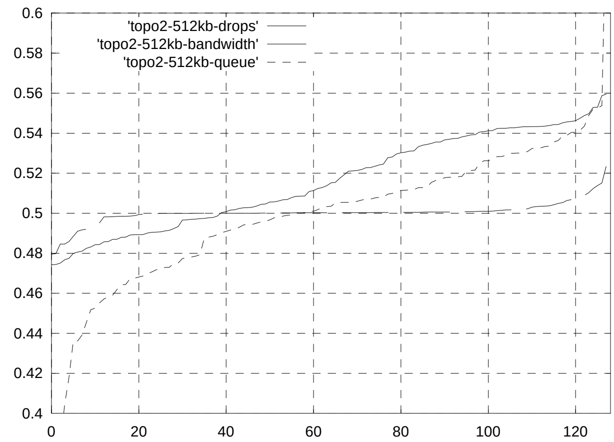

Fig. 6: The case with multiple congested links.

What conclusions could be drawn from these results? Apparently, ECN did not improve much.

VII. HEAD MARKING

Theoretically, the fastest way of congestion notification is marking a packet. When packet dropping is employed, a TCP source could be notified about a congestion (indicated by a packet drop) by a retransmit timer or by duplicate acknowledgments, returned by a receiver after reception of a subsequent packet (provided that the source generates them, which may be not the case for short transfers). So, the second case requires subsequent packets to be successfully delivered to the receiver. Thus, packet marking should deliver congestion signals at least slightly faster.

However, as RED uses exponentially smoothed average queue length, it is possible that a marked packet will be added at the end of the queue that temporary holds many packets. The packet will have to wait for its turn to be forwarded, which will delay the congestion signal propagation. To overcome this problem, we propose *head marking*, a slight modification of marking rules for ECN-enabled RED. A router maintains a *mark pointer*, originally positioned at the head of the queue. When the router decides to mark an ECN-capable packet, it first appends the packet to the tail of the queue and then marks the first ECN-capable packet placed after the mark pointer. After marking, the pointer is advanced to the next packet after the one just marked. While we may end up with marking the original packet, it is more likely that we will mark a packet placed close to the queue head and in this way the congestion notification will be able to "skip" part of the queue. Note that although we mark different packets that the original RED would, the basic RED feature, that the probability of marking a packet belonging to a particular flow is approximately proportional to the flow's share of link bandwidth, is kept.

We implemented head marking in ns-2 and repeated the tests for both topologies. Unfortunately, the results did not improved much over the plain ECN. Even worse, for the multi-link topology it seems that head marking reduces goodput in more cases and in deeper degree as the plain ECN.

VIII. CONCLUSIONS

The dynamics of RED is probably more difficult to understand than one could expect. For the project described in the paper, we made a great number of simulations to assure that the results are not affected by some random effects. Unfortunately, the results are difficult to explain in a simple and intuitive way.

The main assessment factor used in our test, the aggregate goodput of all TCP sources, in most cases was the same for both ECN and non-ECN cases. It is possible that ECN in many cases could not help, since link utilization was near 100%. This suggests, however, that the pure drop-based RED was sufficiently good.

Of course, the simulator may be wrong (e.g., due to a bug). Thus, to finally verify the gain from ECN, detailed tests with network equipment are needed. Note one interesting point: A bug in a network simulator is just a bug, while a bug in network equipment is often a fact.

We believe that the future of congestion control belongs to ECN-type congestion control, i.e., using marking packets instead of dropping them to signal congestion to the sources, provided that some security issues will be addressed. However, our work failed to verify that the deployment of ECN in a RED-based network will give any noticeable improvement. Surprisingly, our results suggest that it is sometimes better to resort to dropping. Since we do not know any results that would show, in a convincing way, the advantage of ECN, we infer that either the RED mechanism should be tuned or it should be replaced by a new AQM-type approach.

REFERENCES

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

[2] S. Floyd, TCP and Explicit Congestion Notification Available: http://www.icir.org/floyd/papers/tcp_ecn.4.pdf

[3] K. Fall, S. Floyd, Simulation-based Comparisons of Tahoe, Reno and SACK TCP, available: http://www-nrg.ee.lbl.gov/papers/sacks.pdf

[4] K. Ramakrishnan, S. Floyd, D. Black, The Addition of Explitic Congestion Notification (ECN) to IP, RFC-3168, September 2001

[5] J. H. Salim, U. Ahmed, Performance Evaluation of Explicit Congestion Notification (ECN) in IP Networks, RFC-2884, July 2000

[6] N. Spring, D. Ely, Robust ECN Signaling with Nonces, IETF Internet Draft, October 2001

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

[8] S. Savage, N. Cardwell, D. Wetherall, T. Anderson, TCP Congestion Control with a Misbehaving Receiver, *ACM Computer Communication Review*, October 1999

[9] M. Mathis, J. Mahdavi, S. Floyd, A. Romanow, TCP Selective Acknowledgement Options, RFC-2018, October 1996

[10] W. Stevens, *TCP/IP Illustrated*, *Vol. I*, Addison-Wesley 1994

[11] M. Małowidzki, ECN Is Fine – But Will It Be Used?, *in Proceedings of Artificial Intelligence – Parallel and Distributed Computing and Networks 2003*, Innsbruck, Austria

[12] S. Alessandrini, T. Ferrari, Experimental evaluation of TCP performance with ECN optimization, available: http://www.cnaf.infn.it/~ferrari/tfngn/tcp/ecn/

[13] ECN (Explicit Congestion Notification) in TCP/IP: http://www.icir.org/floyd/ecn.html

[14] ECN-under-Linux Unofficial Vendor Support Page: http://gtf.org/garzik/ecn/

[15] The ns-2 Simulator Home Page: http://www.isi.edu/nsnam/ns/