

On the Performance Gains of TCP with ECN

Kostas Pentikousis, Hussein Badr, and Bilal Kharmah
Department of Computer Science
Stony Brook University
Stony Brook, NY 11794-4400
{kostas, badr, bilal}@cs.sunysb.edu

Abstract- We study the effect of Explicit Congestion Notification (ECN) on TCP performance for large file transfers and compare it with other congestion avoidance mechanisms, namely Drop Tail and RED. In contrast to previous work, we focus on situations in which all nodes in the network operate uniformly under the same mechanism (DT or RED or ECN). We observe no significant improvement in TCP goodput when ECN is supported. On the other hand, our results show that with ECN, TCP flows benefit from lower overhead for unsuccessful transmissions, and lockouts are largely avoided.

Keywords- TCP, Explicit Congestion Notification, Random Early Detection

I. INTRODUCTION

The Transmission Control Protocol (TCP) [1] has been studied and improved over the last decade in order to leverage faster and “fatter” networks on the one hand [2], and to accommodate different application needs on the other. Towards these ends there has been considerable effort to improve the efficiency of TCP’s congestion control algorithms [3], starting with the introduction of Fast Recovery in TCP Reno and progressing to selective acknowledgements in TCP SACK [4]. All these algorithms are based on a simple framework that equates packet loss with congestion. Specifically, since network routers use buffers to smooth out spikes in traffic, as contention increases packets will have to be dropped when a router queue runs out of space. Although one may choose to drop any packet from the queue, the most widely used mechanism is tail-drop or drop tail (DT), which drops the packet that arrives when the queue is full [5]. These packet drops can be detected by TCP (through duplicate acknowledgements and timeouts) and are used as a clear indication of congestion. Even though such an assumption is more or less safe for wired networks, it has been pointed out that it is problematic for networks that are more transmission-media heterogeneous [6]. Despite the fact that dropping a packet is a very cheap operation for a router (in terms of processing), one may argue that it is nevertheless a sub-optimal way of “informing” TCP senders of congestion in the network.

The addition of Explicit Congestion Notification (ECN) to IP [7] has recently advanced to a proposed standard status [8] by the Internet Engineering Task Force

(IETF). ECN aims at providing TCP with an alternative mechanism for detecting (incipient) congestion in the network. That is, a TCP sender that supports ECN (TCP/ECN) does not have to solely depend on packet drops to detect congestion and limit its sending rate.

Several studies have shown that TCP/ECN achieves superior goodput when compared to “standard” TCP (see Section IV below). In fact, the performance gains reported are so high that one is tempted to suppose that the issue of TCP/ECN efficiency has been conclusively settled. This paper, however, presents a study of an all-ECN network in which TCP does not achieve better goodput. But we do show that TCP/ECN is very successful in economizing network resources. The following section provides a brief overview of how ECN works. We refer the reader to [7] for further details.

II. AN OVERVIEW OF ECN FOR IP NETWORKS

ECN requires support from both routers and hosts. Hosts negotiate ECN capability during TCP connection setup. If both are ECN-capable, the TCP sender indicates this by setting a bit in each outgoing packet. ECN-capable routers are responsible for monitoring congestion levels and “marking” packets of (ECN-capable) sources as congestion grows critical, instead of passively waiting until buffer space runs out and resorting to drops. Clearly, ECN relies on the ability of the router to detect incipient congestion, unlike DT. Therefore the router must use an active queue management (AQM) mechanism, such as the one employed in Random Early Detection (RED) [9].

Routers can efficiently mark packets by setting the “CE” bit before forwarding, as explained in [7]. Upon receipt of a packet with the CE bit set, the TCP receiver sends back an acknowledgment with the “ECN-Echo” bit set. The TCP sender exercises congestion avoidance mechanisms upon receipt of the (first) acknowledgement carrying the ECN-Echo bit. The sender reduces the congestion window and sets the “CWR” bit in the first new data segment transmitted. Because TCP acknowledgments are not guaranteed delivery, it is important to make the ECN-Echo mechanism robust against acknowledgment loss. For this reason, the receiver continues to set the ECN-Echo bit in subsequent acknowledgments sent until it receives notification from the sender, via the CWR bit, that the congestion window has been reduced.

TCP/ECN reduces its congestion window when (a) it receives an ECN signal, (b) Fast Retransmit is triggered, or (c) a timeout occurs. The TCP sender signals the receiver that it has reduced its congestion window in all of these cases. A salient point that further differentiates a TCP/ECN sender from a standard TCP one is that the former does not reduce its congestion window more than once per window of data, unless a retransmitted packet has been dropped.

In fact, ECN for IP networks borrows elements from forward explicit congestion notification defined for frame relay and ATM networks [10]. For one thing, both mechanisms require active network components to monitor queue sizes to detect congestion buildup. But there are some important differences. In frame relay, for example, the receiver uses a more sophisticated algorithm in order to determine whether to pass a congestion notification back to the sender. In TCP/ECN, on the other hand, the sender's response to congestion notification is more radical than in frame relay.

III. THE ROLE OF AQM IN ECN

In principle, a RED gateway can signal incipient congestion either by dropping a packet, or by setting a bit if the transport protocol is capable of reacting to ECN [9]. In practice, RED usually refers to the mechanism with no ECN capability, in which a RED router initiates early packet dropping when contention increases, thereby implicitly informing TCP senders that the network is unable to handle the current sending rates, and triggering their congestion control mechanisms. This is the sense in which we shall consistently use the term "RED" in this paper. With the advent of ECN-capable hosts, a RED router that supports ECN can use early packet marking instead of dropping, effectively notifying senders to slow down without imposing the costs associated with actual packet drops (which include, but are not limited to, retransmission of dropped packets). Even with ECN, however, some packet dropping is to be expected to the extent that buffers continue to overflow.

RED was shown to provide better TCP performance than DT in many cases [9], and to deal well with problems like global synchronization and lockouts, prompting the IETF to recommend its deployment [5]. Yet, some network operators seemed reluctant to proceed with it [11]. Subsequent studies confirmed that RED degrades TCP performance under a variety of scenarios [12]. Apparently, the performance gains that can be achieved through ECN depend largely on the effectiveness of a given AQM to detect incipient congestion.

This paper attempts to shed more light on the performance implications for large TCP transfers with respect to the congestion avoidance mechanisms used in the network, and the potential gains if ECN is used. Our primary interest is to compare TCP when complemented by a "pure" marking mechanism to the more usual situation of "standard" TCP with a dropping mechanism. The former is repre-

sented by TCP/ECN, and the latter by TCP over DT. TCP over RED, as such, is ancillary to our focus because it still uses dropping as a congestion notification mechanism. ECN is dependent on an AQM mechanism, in this case RED's. As such, results from TCP over RED provide a "control" group which enable us to apportion the gains achieved by TCP/ECN between, on the one hand, its AQM scheme and, on the other, the marking mechanism per se.

IV. RELATED WORK

Slim and Ahmed [13] used a Linux-based test bed network to study TCP/ECN. They demonstrated that when an ECN-capable sender competes with a non-ECN-capable one in the presence of various levels of background traffic, the TCP/ECN sender always performs better. In particular, they showed that both for bulk and transactional transfers there are substantial gains in TCP goodput if ECN is used.

Zhang and Qiu [14] used simulation to evaluate TCP performance under RED and DT exploring different scenarios, which included long and short transfers, different round trip times (RTTs), a mix of TCP and UDP traffic, and two-way traffic, but were mainly interested in evaluating RED with dropping. Their observations include that, in mixed traffic where n ECN-capable TCP senders compete with n non-ECN-capable ones the average increase in goodput can be up to 30% in favor of TCP/ECN. The best relative performance gains are achieved when bulk transfers are considered and the size of the congestion window is 3-4 packets: This is the most favorable for ECN because Fast Retransmit can still be employed and TCP senders do not always resort to timeouts for loss recovery.

More recently, Athuraliya *et al.* [15] evaluated TCP/ECN as part of their simulation study of Random Exponential Marking (REM). They show that for increasing levels of congestion and for large transfers TCP achieves better goodput under REM, followed by DT and then RED. They also illustrate that ECN reduces losses in the network while preserving high goodput. Finally, they consider the case where random drops are introduced in the network (roughly simulating errors in a wireless environment) and conclude that ECN can increase TCP's performance in hybrid wired/wireless networks.

A common, determinant characteristic of the published literature known to us is that it examines scenarios in which TCP/ECN flows compete with ECN-unaware ones. The literature makes clear that, under these circumstances, the TCP/ECN sender is virtually assured better goodput than the TCP sender relying only on segment losses to detect congestion in the network. All other things being equal, a TCP/ECN sender possesses more information about network conditions and avoids decreasing its congestion window more than once per window of data.

Another point worth highlighting is that many studies use "background", non-ECN traffic in order to maintain a level of congestion that ensures routers operate in the "RED region": that is, with average queue sizes always

within the range of values for which random early marking is in effect. This permits ECN to display its best potential with respect to the ECN-capable flows present. However, it is known that tuning RED parameters so that routers in real networks, experiencing a variety of traffic mixes, are always operating in the RED region is difficult.

This paper presents case studies using simulation in which there is no background traffic. Unless mentioned otherwise, traffic in our studies is either all ECN-capable or all ECN-unaware, and is not constructed to ensure any particular, *a priori* relationship with respect to the RED parameters in effect. We also focus on relatively large transfers.

V. EXPERIMENTAL CONFIGURATION

We used ns-2 [16] to simulate a scenario involving a number of clients connecting to a FTP server (Figure 1). Our network topology is similar to the one used in many protocol evaluations, including the ones presented in the previous section. Each client is connected to router A with a LAN-like link at 10 Mbps bandwidth and 0.5 ms delay. Router A is connected to router B through a (bottleneck) link of 1.5 Mbps capacity and 20 ms propagation delay. Finally, the server is connected to router B over a LAN-like link. The server uses TCP Reno under all scenarios, while the clients support delayed acknowledgements [3]. TCP time granularity is set to 100 ms, and the maximum segment size is 1460 B. The receiver’s advertised window (τ_{wnd}) is set to 64 KB, or approximately 44 segments. For the scenarios presented in this paper, all clients have the same TCP configuration (τ_{wnd} , time granularity, *etc.*) and simultaneously initiate 20-MB downloads from the server. These large transfers allow TCP/ECN to display its best potential. The congestion avoidance mechanism used at router B is either DT or RED, as indicated in the results.

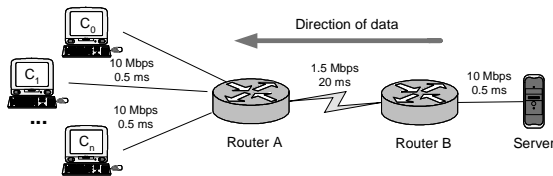


Figure 1. The network topology used in our experiments.

If DT is used at router B, the only parameter we can set is the queue limit (QL). Van Jacobson proposed as early as 1988 that QL should no be less than the bandwidth \times delay product [17]: Bandwidth is the capacity of the bottleneck link along the connection path (1.5 Mbps in our setting), and delay is the roundtrip propagation delay of the bottleneck link. It was also proposed that delay should be the average end-to-end roundtrip time across all flows sharing the bottleneck link [17]. Calculating this kind of delay is rather difficult (if at all possible) in real networks,

but it is straightforward for the network of Figure 1. Other researchers have proposed that QL should be even larger, set to two to four times the above-mentioned bandwidth \times delay product of the bottleneck link. If we consider the end-to-end bandwidth \times delay product for the network of Figure 1, then this translates to approximately 6-24 segments, or 5-20 packets if we consider the propagation delay of the bottleneck link only. Large queue limits can increase the delays experienced by TCP connections, leading to worse performance [5]. On the other hand, larger queue limits (in general) allow higher link utilization. Since there is no universally accepted criterion for the size of QL we conducted our tests with QL ranging from 4 to 256 packets.

If RED is used at router B, four additional parameters need to be specified: the minimum (\min_{th}) and maximum (\max_{th}) thresholds of the “RED region”, the maximum dropping probability (\max_p), and the weight factor used to calculate the average queue size (w_q) [9]. We experimented with, and present here the results for, RED with and without ECN support. Several studies have shown that RED performance depends to some extent on parameter setting [11-13, 17]. The RED parameter settings for all tests presented here are: \max_{th} is three times \min_{th} , w_q was set to 0.002, and \max_p to 2% and 10%, as per [9] and [18]. The only free parameters in our experiments are \min_{th} and QL. In the remainder of this paper we will use the terms “conservative” and “aggressive” to denote a \max_p of 2% and 10%, respectively.

In our evaluation, we consider global (“aggregate”) network performance, as well as individual host performance. Choosing the metrics to judge end host performance is rather straightforward. Goodput per connection, *i.e.* the application payload divided by the total connection time, is a good candidate. Others include total number of packets sent, total number of packets dropped at the gateway, and per-packet transfer time. Global network performance is more difficult to determine. Link utilization interests network operators and can be used to measure the efficiency of the network, but could lead to inaccurate conclusions for our study. Instead, we define goodput efficiency as the ratio of the sum of application payload across all flows divided by the amount of data that could have been sent during the maximum connection time. Link utilization is always greater or equal to goodput efficiency because it includes duplicate and retransmitted packets. Another metric can be the amount of buffer space needed in order to maintain a given level of overall performance, *e.g.* keep the link full and packet drops to a minimum.

It should be noted that very little in the simulation is actually driven by random numbers: ns-2 runs actual TCP code, and packet transfer times are deterministic for a given link, once its transmission rate and propagation delay are defined. The only aspect of the simulation that receives generated random variate input is the RED marking at the router B buffer. The following sections present our evaluation of TCP performance for the topology of Figure 1

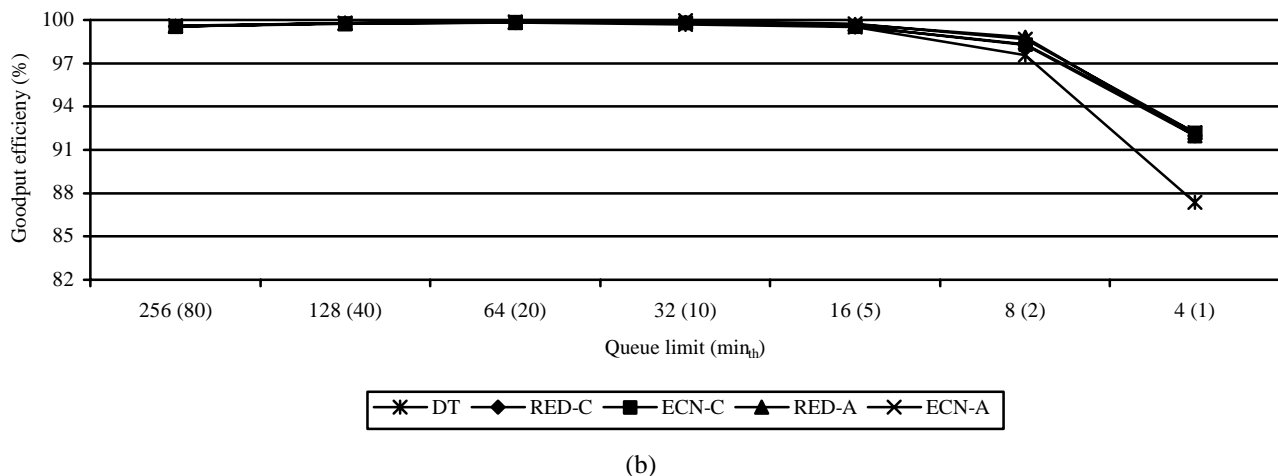
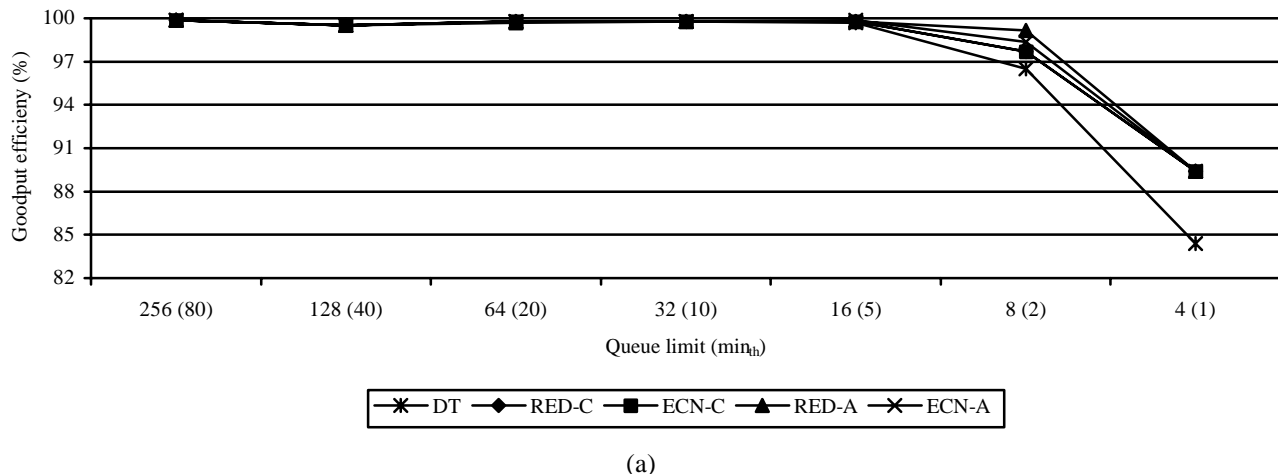


Figure 2 Goodput efficiency (%) for five (a) and ten clients (b).

when router B uses DT or RED (with and without ECN support).

VI. BASELINE TEST

The roundtrip bandwidth×delay product of the bottleneck link is 7500 B, which enables the server to have a minimum of five segments buffered in the network. A single TCP sender needs a sending window of at least six segments to saturate the end-to-end connection. TCP sets its sending window as the minimum of $rwnd$ and the congestion window $cwnd$ [3]. That is, even if no losses occur the sending window can never exceed 44 segments. Therefore, if we use DT at the bottleneck link with a QL of 39 packets no drops should occur. Our baseline test, consisting of a single FTP transfer from the server to a client, confirms this. The minimum QL for which no drops occurred was 39 packets. The file transfer took 112.5 seconds, yielding a goodput of 1491.3 Kbps, and a “total” goodput efficiency of 99.4%.

VII. GOODPUT EFFICIENCY

Figure 2 presents the goodput efficiency results for five and ten clients for various QLs, and different congestion avoidance mechanisms used at router B. The results shown in the figure make clear that in our experiments there are no significant differences in goodput efficiency when the queue limit is sufficiently large, *i.e.* $QL \geq 16$, regardless of the congestion avoidance mechanism used at router B. Further increases of QL (not shown in Figure 2) increase goodput efficiency only slightly, if the number of connections increases as well. At the other end, when QL shrinks to eight or fewer packets, goodput efficiency with DT deteriorates considerably, especially if fewer clients are active.

Global synchronization and lockouts are a known deficiency of TCP in conjunction with DT [5]. Global synchronization (of losses) occurs when a router drops many consecutive packets in a short period, forcing a number of TCP senders to slow down at the same time, retransmit the

dropped packets virtually in unison (global synchronization of *retransmissions*), enjoy successful transmissions for a short period (because the backlog at the bottleneck queue has been cleared in the meantime), and then experience more consecutive drops. This repetitive cycle comprises periods of high congestion, followed by poor aggregate TCP performance and underutilization of scarce network resources such as the bottleneck link. Global synchronization has been observed in simulation studies (*e.g.* [19]) of TCP over DT, but not in real networks [20], nor subsequent studies [11-15], yet many still consider it in the nature of a performance disaster waiting to happen.

Figure 2 shows that unless QL is unrealistically small, there is no collapse in aggregate TCP performance, even for small numbers of clients, and despite the fact that the file downloads start simultaneously. Hence, while global synchronization of losses might be taking place, the basic symptom of global synchronization is not observed in our experiments: An examination of the simulation data explicitly shows that global synchronization of retransmissions does not occur. This is because TCP Reno handles congestion-related losses better than Tahoe, which always resorts to Slow Start [3]. Reno, on the other hand, proceeds with Fast Recovery after Fast Retransmit, which helps to keep the pipe full and mitigates link underutilization¹.

Global synchronization was studied in [19], with a TCP version sometimes referred to as Old Tahoe; Old Tahoe does not include Fast Retransmit, Fast Recovery, or the delayed acknowledgements algorithm. Fast Retransmit enables a TCP sender to recover from a loss without waiting for a timeout, thus making global synchronization (of retransmissions) less likely. Fast Recovery prevents the unnecessary reduction of the congestion window to one segment, thereby allowing for the sender to keep the pipe full after a congestion incident. The combination of Fast Retransmit and Fast Recovery causes less oscillation in transmission behavior than that described in [19]. Moreover, we conjecture that the delayed acknowledgements algorithm introduces some degree of randomization in the sender’s transmission patterns, making global synchronization even less likely. This “randomization” is closely related to the multiplexing of acknowledgements from multiple clients at router B, and to the fact that these clients sometimes acknowledge two segments and sometimes only one. To the extent that newer TCP versions (*e.g.* TCP SACK) handle multiple drops from a single window of data better than Reno, we would expect global synchronization of losses to become even less of a problem.

On the other hand, lockouts, which are defined as the phenomenon where a small number of connections are allowed to “take over” the bottleneck link, do occur. Lockouts do not translate into lower link utilization if the router has sufficient buffer size. If QL is sufficiently large, routers can effectively smooth out traffic spikes, allowing “favored” connections to transmit enough packets to keep the

the link utilized. Lockouts *can* diminish goodput efficiency if QL is limited (see Figure 2, for QL = 4, 8) because traffic spikes cannot be accommodated, forcing even the “favored” TCP senders to maintain very small sending windows. For small QLs, RED (with and without ECN) seems to be capable of maintaining a higher level of goodput efficiency than DT.

To conclude, for our configuration setup, in a network where all clients support ECN, TCP/ECN does not achieve better goodput efficiency than standard TCP over RED or DT (except for extreme and rather unrealistic QLs).

VIII. TOTAL TRANSFER TIMES

The literature points out the unfairness that different RTTs cause to a TCP sender’s performance. In other words, longer RTTs translate into slower congestion window expansion, leading to worse TCP transfer times, *i.e.* goodput. In this section, we show that the queue management employed at the gateway can induce unfairness even when there should be none.

Consider the scenario with five clients initiating 20-MB file downloads at the same time and QL is set to 64 packets. The following results are typical for other scenarios as well. Figure 3 presents the progress of the file transfer with time (in terms of acknowledgements received at the server) for the fastest (C_4) and slowest (C_2) downloads when DT is used at router B. DT discards a total of 370 packets due to buffer overflow. The difference in download times, which can be used as an implicit metric of fair bandwidth sharing, is 11.1%. This is very similar to the difference in download times when RED is used: 10.76% (Figure 4). RED induces 842 packet drops (including packet discards due to early packet dropping and buffer overflow), *i.e.* packet drops double.

If all five clients support ECN then both the network as a whole and the individual clients perform better. First, goodput efficiency stays the same (Figure 2), and packet drops are drastically reduced to 39. Second, fairness increases, as measured by the difference in download times between the fastest (C_3) and slowest (C_0) transfers, which shrinks to 2.56% (Figure 5).

It is interesting to investigate what happens to these metrics as we migrate from scenarios in which none or all of the clients support ECN to scenarios where some, but not all, support ECN. Slim and Ahmed [13] studied the performance gains of TCP for bulk FTP transfers. However, they consider the case where a single ECN-capable host competes with an ECN-unaware one. These two flows compete in a network where a varied number of ECN-unaware TCP senders introduce background traffic. The background flows are used to introduce different levels of congestion at the gateway in order to keep the gateway always operating in the RED region.

Our experiments provide no such guarantees: we believe that in real networks it is neither easy nor reasonable to expect a gateway to always operate in the RED region –

¹ We would like to thank the anonymous reviewer for drawing our attention to this point.

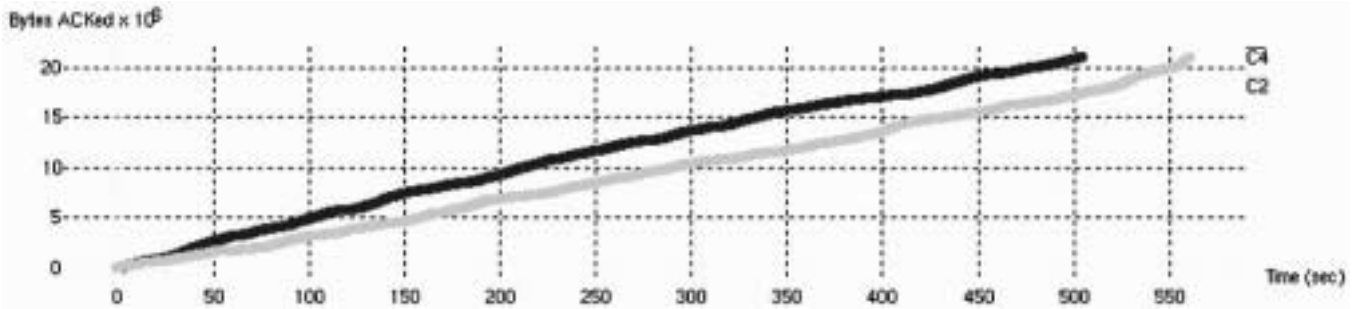


Figure 3. Download progress for the fastest (C₄) and slowest (C₂) clients; DT with QL = 64.

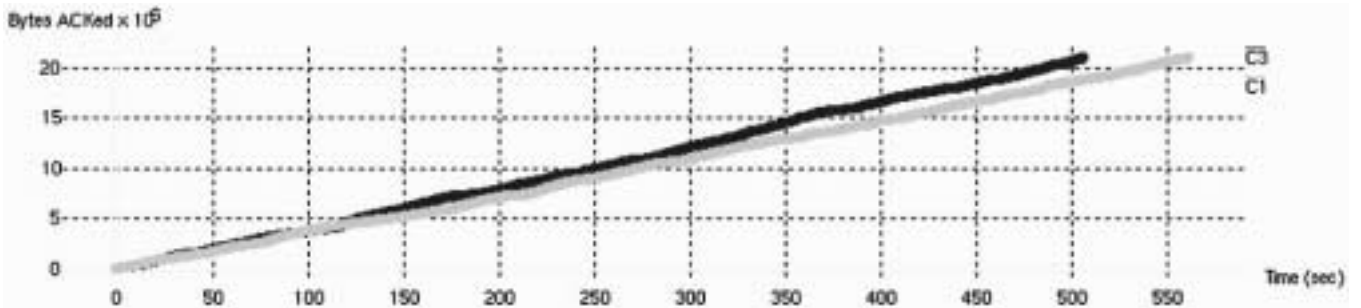


Figure 4. Download progress for the fastest (C₃) and slowest (C₁) clients; aggressive RED ($\min_{th} = 20$, QL = 64).

after all this is not the goal of RED. In order to see the performance gains both for individual clients and the network as a whole we have one client supporting ECN, while the rest do not. For example, in the case of two clients, we first conduct the experiments for varying QLs with both clients being ECN-unaware. The experiments are then repeated, but with ECN enabled for the underperforming client. As expected (see Section IV), the ECN-capable client suffers fewer packet drops and always achieves better goodput than the ECN-unaware one; this result is consistent across all scenarios [21].

Nevertheless, in experiments with one ECN-capable client amongst a total of five competing clients, TCP/ECN did not seem to have a guaranteed advantage. In particular, we enabled ECN support for client C₁ (which had the worst performance in Figure 4 and experienced 153 packet drops throughout the file transfer). As may be inferred from Figure 6, C₁ no longer experiences the slowest download, but neither the fastest of the five clients. C₁ received the file 14 seconds earlier than it did without ECN support and lost only 7 packets (these losses provide a clear indication that router B was not always operating in the RED region). The gain in terms of packets that were marked at the congested router and delivered, instead of being dropped, constituted only 1% of C₁'s total packets. Overall, the addition of a single ECN-capable host increased the minimum goodput, kept goodput efficiency high, and saved network and host resources. Our results are far less striking than those reported in the literature, but still leave the advantage with ECN.

Our experiments point out that the presence of an ECN-capable client influences the performance of other non-ECN-capable hosts sharing the network in a non-uniform way. For example, the performance of the non-ECN-capable C₀ (which was not, as can be inferred from Figure 4, the worst performing client) degraded in the presence of an ECN-capable C₁ (though not by so much that it became the worst performing one, as may be surmised from Figure 6). When ECN was enabled for both C₀ and C₁, C₀'s *absolute* download time stayed the same, indicating that a host is not bound to achieve better goodput simply because it uses ECN. Of course, C₀ experienced fewer packet drops, which resulted in improved *relative* performance. The minimum goodput achieved by the five clients improved as the number of ECN-capable hosts increased, indicating that bandwidth sharing became fairer.

We repeated these scenarios, maintaining QL at 64 but increasing the competing downloads to ten. The results were more favorable for ECN. With no ECN-capable client amongst the ten, C₉ was the underperforming client, experiencing the most segment drops and finishing the download in 1120.5 seconds. Had C₉ be the only ECN-capable client, it completed the download faster than the other nine (in 1029 sec.) and lost 334 fewer packets. Once again we observed that the performance of the other downloads was influenced by C₉ capability to react to ECN marks.

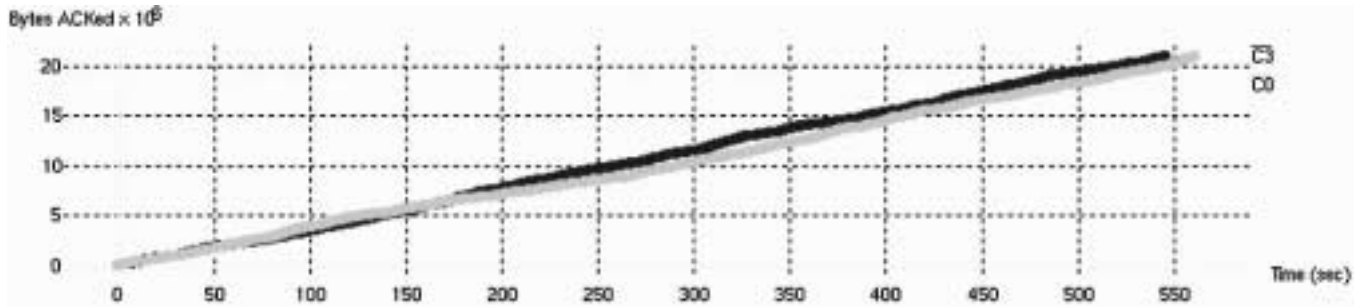


Figure 5. Download progress for the fastest (C_3) and slowest (C_0) clients; aggressive ECN ($\text{min}_{th} = 20$, $QL = 64$).

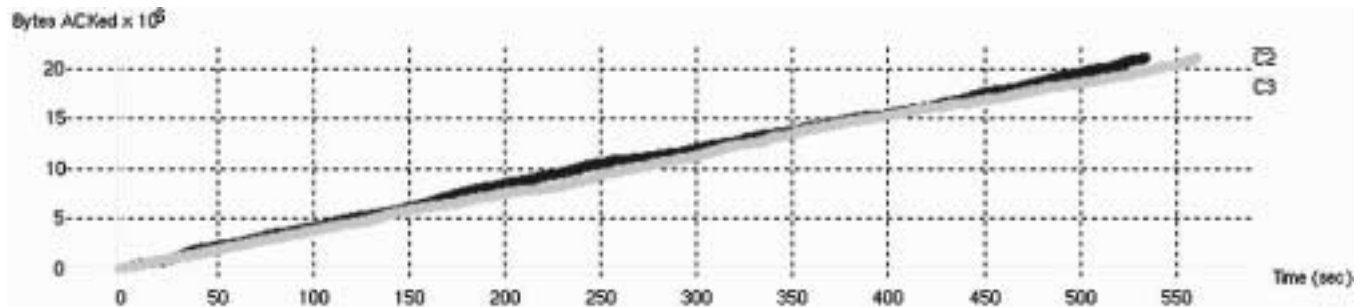


Figure 6. Download progress for the fastest (C_2) and slowest (C_3) clients. C_1 is the only out of five clients using TCP/ECN; aggressive ECN ($\text{min}_{th} = 20$, $QL = 64$).

IX. TCP/ECN FRUGALITY

This section focuses on the number of packets sent by the FTP server but dropped at router B. Figure 7 presents the total number of packets sent by the server, the number of packets delivered to the ten clients (in black) and the number of packets dropped by the congestion avoidance mechanism at the gateway (in white). Clearly, DT, conservative and aggressive RED, and conservative ECN cause more packet drops as QL decreases. On the other hand, aggressive ECN drops fewer packets with $QL = 64$ than $QL = 128$. It is also remarkable that with $QL = 32$, aggressive ECN drops an extremely small fraction of packets when compared with the others. In fact, with $QL = 32$ and aggressive marking, TCP/ECN achieves the highest goodput efficiency measured (Figure 2). That same goodput efficiency is attained again only when QL is increased to 512 packets, a value that is sufficiently large to avoid packet drops altogether. ECN cannot prevent packet drops altogether, but it can reduce them dramatically. In general, TCP performs better when aggressive ECN is used: the sender sends fewer total segments, with, furthermore, a higher proportion of delivered to dropped packets.

The high level of goodput efficiency is due to a combination of factors, including the increased level of multiplexing and TCP's competency with large file transfers. ECN, too, plays an important role by notifying TCP senders of congestion build-up in a manner more effective than "traditional" packet drops. The lack of packet drops bene-

fits the TCP sender in two ways. First, segments are delivered to the receiver, and retransmissions are avoided. Second, the TCP sender does not rely on packet drops alone to initiate congestion avoidance, and thus its ability to react to incipient congestion improves. For example, (without ECN) TCP Reno may infer a single segment drop after at least three round trip times, provided that there are enough packets on the fly to allow Fast Retransmit to kick in. TCP/ECN needs only one RTT to become cognizant of the congestion buildup.

X. TCP/ECN PERFORMANCE WITH LONGER RTTs

We repeated the experiments of the previous section but increased the propagation delay of the bottleneck link (Figure 1) from 20 to 100 ms. Introducing longer delays at the bottleneck link increases the delays in the TCP congestion control feedback loop forcing TCP senders to become less aggressive. Meanwhile, the amount of packets that can be buffered in the network increases as well. Thus, for a given QL the increase in propagation delay causes fewer packets drops. This is illustrated in Figure 8, which also shows that when QL is 32 or 16, the relative advantage for a TCP/ECN sender is actually improving in terms of packet losses. Aggressive ECN is still the best choice in terms of packets dropped across all configurations.

Longer latencies translate into longer periods before a packet drop is detected by Fast Retransmit, a timeout, or even ECN. Therefore, TCP spends longer periods of time

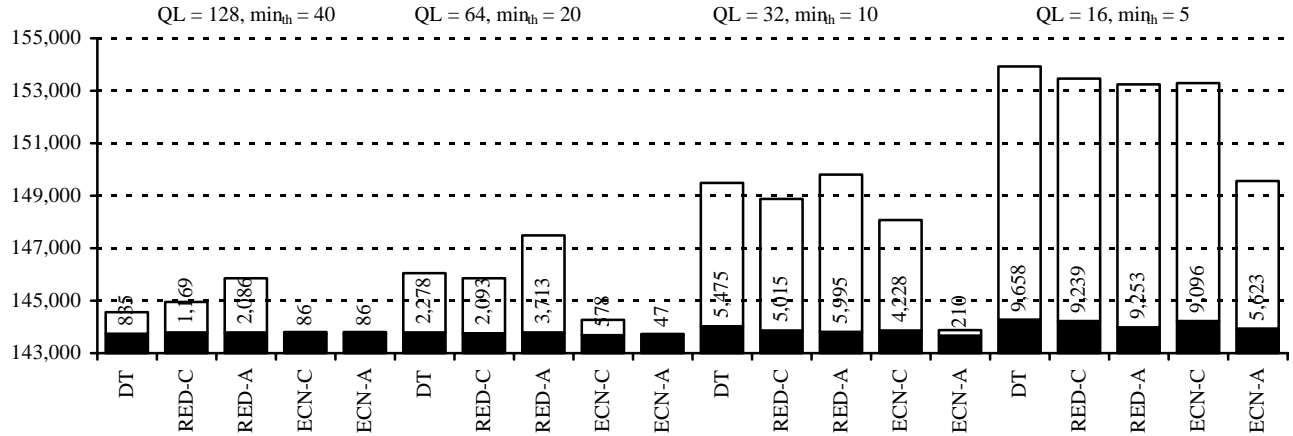


Figure 7. Total number of packets sent by the FTP server to all ten clients. The number of packets dropped at the gateway is shown in white.

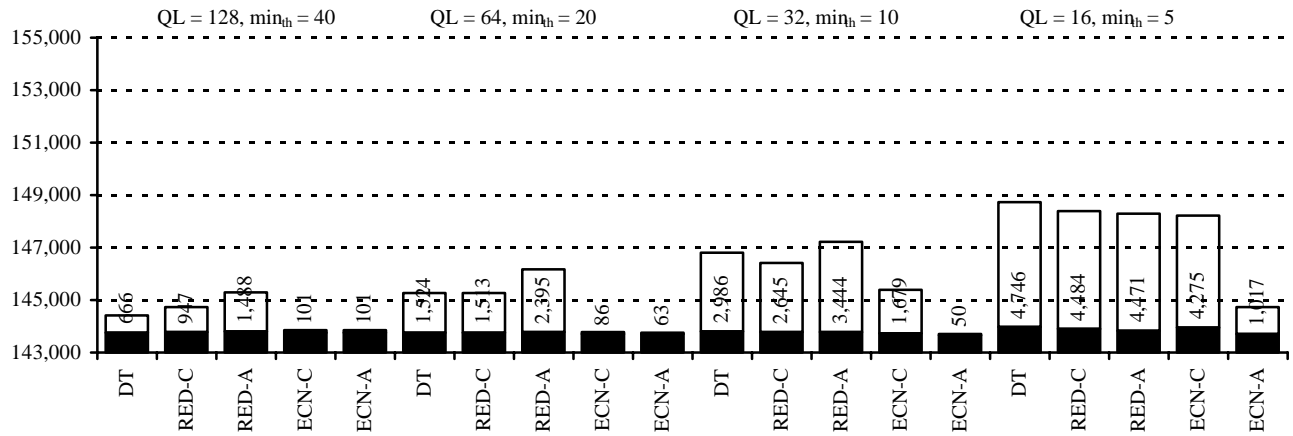


Figure 8. Total number of packets sent by the FTP server to all ten clients (longer delays). The number of packets dropped at the gateway is shown in white.

without transmitting. In addition, the Slow Start phase has a higher toll on goodput efficiency. Although goodput efficiency remains high (Figure 9), it is less than for smaller latencies (Figure 2(b)).

XI. DISCUSSION

Many network operators charge their customers by the amount of traffic they carry through their routers. The foregone revenues incurred by dropping a packet under RED can be significant in many cases: a packet dropped while the router is in the RED region is a packet that will not be charged to the customer under current pricing models. On the other hand, network costs can be reduced with aggressive ECN, which yields high goodput efficiency and fewer packet drops, and, hence, higher operating margins.

ECN is already being deployed with Linux 2.4, and we believe that it will soon be deployed in other TCP stacks as well. Section VIII demonstrated that even partial ECN deployment improves overall performance. Yet the results presented in this paper are a lot less impressive than the ones presented in earlier work. So far as TCP goodput gains are concerned, one may argue that they are actually disappointing. Note, though, that our results are based on a different approach for evaluating TCP/ECN. Our experiments compare situations in which, in any given case, *all* clients are operating in either a DT or a RED (with dropping) or an ECN environment. As such, no client has a competitive advantage over its peers. In contrast, previous studies focused on TCP/ECN senders that co-exist with other, ECN-unaware ones. Our results show that ECN is a very effective congestion “warning” mechanism that can drastically reduce the amount of wasted effort (as meas-

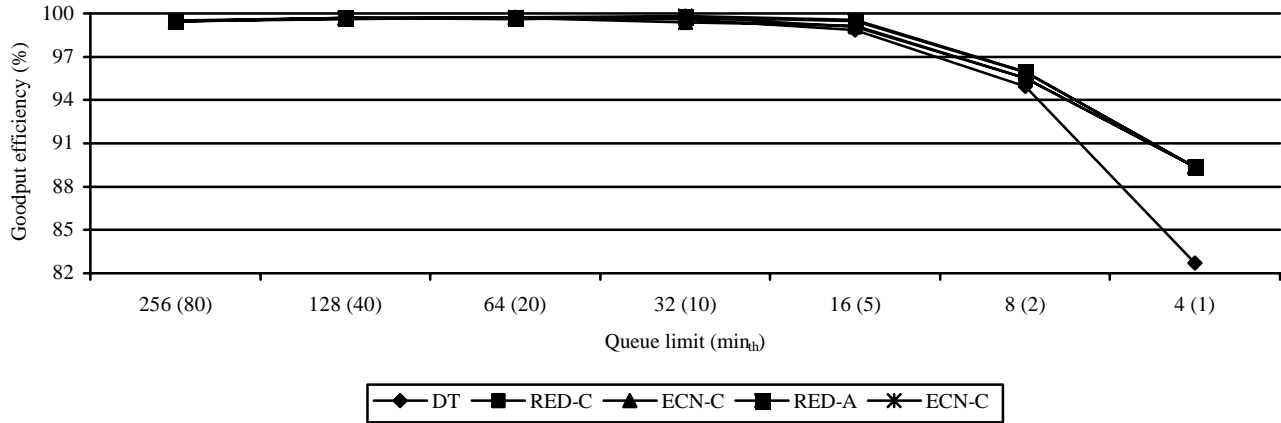


Figure 9. Goodput efficiency (%) with ten clients and long delays.

ured by the number of duplicate and dropped packets), and foster a fairer sharing of network resources. Be that as it may, TCP/ECN goodput is not improved, because it has the same mechanisms for detecting improved network conditions as does standard TCP.

Going forward, ECN seems to point the way to innovations that can change the existing internetworking model. For example, ECN can be used to develop forms of QoS differentiation, and enable schemes that permit the network operator to optimize network utilization while charging users for their marginal contributions to congestion [22].

XII. CONCLUSION AND FUTURE WORK

This paper compares the performance of TCP/ECN vs. standard TCP. In contrast to previous work, we compare situations in which all clients in the network uniformly operate under either DT or RED or ECN. Our conclusions are summarized as follows:

- Global synchronization (of retransmissions) does not occur regardless of the queue management employed. Lockouts do, but ECN seems very effective in avoiding most of them.
- ECN is an effective mechanism for the timely and efficient conveying of congestion information. ECN leads to far fewer packets drops, but does not necessarily lead to improved transfer times, and thus goodput, under uniform conditions.
- Aggressive ECN can reduce packet drops, promote a fairer environment, increase network efficiency, and deliver higher performance to individual connections.

Open issues and future work include:

- The effect of RED parameter settings, other than the ones experimented with here, on TCP/ECN efficiency.

- TCP/ECN performance with short flows, where the shorter connection times might not permit the feedback mechanism to fully deploy its advantage.
- TCP/ECN performance when competing with unresponsive flows, *e.g.* UDP traffic.
- The effect of alternative AQM mechanisms on TCP/ECN efficiency. How might TCP/ECN behave with an AQM such as PI [23], REM [15], BLUE [24], and so on, rather than RED?

XIII. REFERENCES

1. J. B. Postel, *Transmission Control Protocol*, RFC 793, September 1981.
2. V. Jacobson, R. Braden, and D. Borman, *TCP Extensions for High Performance*, RFC 1323, May 1992.
3. M. Allman, V. Paxson, and W. Richard Stevens, *TCP Congestion Control*, RFC 2581, April 1999.
4. M. Mathis, J. Mahdavi, S. Floyd, and A. Romanow, *TCP Selective Acknowledgment Options*, RFC 2018, April 1996.
5. B. Braden, et al. *Recommendations on Queue Management and Congestion Avoidance*, RFC 2309, April 1998.
6. K. Pentikousis, "TCP in wired-cum-wireless environments", in *IEEE Communications Surveys*, vol. 3, no. 4, Fourth Quarter 2000.
7. K.K. Ramakrishnan, S. Floyd, and S. Black, *The Addition of Explicit Congestion Notification (ECN) to IP*, RFC 3168, September 2001.
8. S. Bradner, *The Internet Standards Process – Revision 3*, RFC 2026, October 1996.
9. S. Floyd and V. Jacobson, "Random Early Detection Gateways for Congestion Avoidance", in *ACM/IEEE Transactions on Networking*, vol. 3, no. 1, August 1993.

10. W. Stallings, *ISDN and Broadband ISDN, with Frame Relay and ATM*, Fourth Edition, Prentice Hall, October 1998.
11. M. May, J. Bolot, C. Diot, and B. Lyles., "Reasons Not to Deploy RED", in *Proceedings of 7th International Workshop on Quality of Service (IWQoS'99)*, London, June 1999.
12. M. Christiansen, K. Jeffay, D. Ott, and F. D. Smith, "Tuning RED for Web Traffic". In *Proceedings of ACM SIGCOMM 2000*, Stockholm, Sweden, August 2000.
13. J. H. Slim and U. Ahmed, *Performance Evaluation of Explicit Congestion Notification (ECN) in IP Networks*, RFC 2884, July 2000.
14. Y. Zhang and L. Qiu, "Understanding the End-to-End Impact of RED in a Heterogeneous Environment", Cornell CS Technical Report 2000-1802, July 2000.
15. S. Athuraliya, V. Li, S. Low, and Q. Yin, "REM: Active Queue Management", in *IEEE Network*, May/June 2001.
16. UCB/LBNL/VINT Network Simulator - ns (version 2), <http://www.isi.edu/nsnam/ns>.
17. End-to-end mailing list archive, <ftp://ftp.isi.edu/end2end/end2end-interest-1998.mail>.
18. S. Floyd, *RED: Discussions of Setting Parameters*, <http://www.aciri.org/floyd/REDparameters.txt>.
19. L. Zhang and D. D. Clark, "Oscillating behavior of network traffic: A case study simulation", in *Internetworking: Research and Experience*, vol. 1 no. 2, December 1990.
20. M. May, T. Bonald, and J.-C. Bolot, "Analytic Evaluation of RED Performance", in *Proceedings of IEEE INFOCOM 2000*, Tel-Aviv, Israel, March 2000.
21. K. Pentikousis and H. Badr, "TCP with ECN: the Case of Two Simultaneous Downloads", in *Proceedings of IASTED Advances in Communications (AIC 2001)*, Rhodes, Greece, July 2001.
22. K. Lavens, P. Key, and D. McAuley, "An ECN-based end-to-end congestion-control framework: experiments and evaluation", MSR-TR-2000-104, October 2000.
23. C. Hollot, V. Misra, D. Towsley, and W. Gong, "On designing improved controllers for AQM routers supporting TCP flows", in *Proceedings of IEEE INFOCOM 2001*, April 2001.
24. W. Feng, D. Kandlur, D. Saha, K. Shin, "Stochastic Fair Blue: A Queue Management Algorithm for Enforcing Fairness", in *Proceedings of IEEE INFOCOM 2001*, April 2001.