Time-Driven Early Discard (TED) to Improve the Fairness of TCP Congestion Control

Mario Baldi and Andrea Vesco
Dipartimento di Automatica e Informatica — Politecnico di Torino
Corso Duca degli Abruzzi, 24 - 10129 Torino (Italy)
{mario.baldi,andrea.vesco}@polito.it

Abstract— ***This paper proposes a novel adaptive AQM (advanced queue management) approach called Time-Driven Early Discard (TED). The basic underlying idea is to set a deadline on packet service time in routers, beyond which packets are discarded. TED is shown to improve fairness among TCP connections sharing congested links when the deadline is chosen proportional to their round trip time (RTT). TED is adaptive in limiting the throughput of only those connections that traverse congested links. In fact, as demonstrated by the presented results, TCP connections traversing parts of the network with enough available capacity can achieve the maximum throughput enabled by their transmission window as corresponding packets do not reach their deadline . Finally, the paper shows how TED can be instrumental in enabling TCP to deploy shorter retransmission timeouts, which results in prompter reactivity to loss, hence improved performance overall in terms of achieved goodput .***

I. INTRODUCTION

Internet traffic is currently dominated by TCP connections carrying data generated by applications such as the web, file transfers, or peer-to-peer file sharing. A distinctive feature of TCP-based applications is their elastic nature as they can operate at a wide range of rates depending on availability of network resources. TCP sources increase the sending rate up to the capacity of their access link to match the maximum throughput available from the network path. Since available bandwidth changes over time, TCP uses a window-based, congestion control algorithm that increases and decreases the sending rate in order to match these variations.

The TCP window limits the maximum number of outstanding — transmitted and not yet acknowledged — bytes and slides over a segment of transmitted data once their reception acknowledgement is received. In addition, the window size, hence the sending rate, is increased if packets are correctly delivered and decreased if packets are lost according to the well known, widely studied additive increase and multiplicative decrease (AIMD) algorithm.

A round trip time (RTT) separates events (e.g., transmission of a packet) from their respective effect on the window (i.e., sliding or size changing), which takes place once the corresponding acknowledgement is (not) received. Consequently, throughput achieved by TCP is inversely proportional to the RTT, hence the well known *fairness* problem of TCP as connections with short RTT achieve higher throughput than connections with longer RTT when they share a congested link. To be more specific, TCP unfairness has two causes:

1. Flow control: a window limits the amount of outstanding data, which is outstanding for at least a RTT;
2. Congestion control: TCP senders identify congestion (as a consequence of packet loss) and react to it within a time proportional to their RTT.

The solution proposed in this work focuses on congestion control to mitigate the fairness problem and improve goodput by penalizing TCP connections with a shorter RTT in case of congestion and decreasing the time for reacting when congestion is identified, respectively.

It is widely accepted that flow control and congestion control mechanisms, albeit critical, are essential for packet networks — specifically the Internet — to function in a stable way. Consequently, a significant amount of work has been devoted to the design, performance evaluation, and optimization of flow control and congestion control algorithms, among which the ones of TCP. Being an end-to-end mechanism, TCP infers network congestion from clues it gathers from events, such as a missing packet. Significant research has been devoted to giving routers an active role in congestion control. According to the well known RED (Random Early Detection) algorithm [2] routers monitor the average occupancy of their buffers and when it grows beyond a given threshold they "notify" selected TCP sources by dropping their outstanding packets.

Due to the multiplicative decrease algorithm, packet loss can lead to a potentially significant degradation of network performance in terms of overall amount of information successfully transferred by TCP-based applications. Explicit Congestion Notification (ECN) was proposed as a way of notifying TCP sources without dropping packets. Routers use two bits in the DiffServ Field of the IP header to mark packets of selected connections when the network is congested. The TCP sender receiving a notification reacts accordingly without triggering actions typically performed in reaction to a packet loss. Specific packet marking and window updating algorithms are required in network nodes and TCP senders, respectively, to benefit from ECN in fairness control [3]. For example FRED (Fair Random Early Detection) [4] uses per-flow information and core-stateless fair queuing (CSFQ) [5] in network nodes that operate differently at the network edge and in the core.

Along these lines, this paper presents and assesses a possible solution to improve fairness among TCP connections

with different RTT based on a novel advanced queue management (AQM) algorithm called Time-driven Early Discard (TED). Network nodes drop packets, possibly even when their buffers are not yet completely full, based on the time a packet has spent in the network. A different discard deadline can be set for each TCP connection sharing a link. The AQM algorithm aims at ensuring that the TCP connections fairly share the link bandwidth without keeping any per-flow information inside network nodes.

Section II introduces the basic operating principles of TED and some possible TED-based approaches to improve fairness among different TCP connections and analytically devises the maximum queue size with TED is devised analytically. In order to assess the proposed approach to TCP fairness improvement simulations were run with NS2. Section III describes the simulation scenario and results obtained using both conventional TCP Reno sources and TCP senders with improved loss recovery and RTT estimation. Section IV draws some conclusions and delineates possible future work.

II. TIME-DRIVEN EARLY DISCARD

This paper presents a solution that aims at controlling TCP sources to ensure fairness while maximizing aggregate throughput. Time-driven early discard (TED), a novel AQM, is at the core of such solution: a *deadline* is set on packet service time and a packet is dropped if it is not forwarded before its deadline. In the context of this work output queues with a first-in-first-out (FIFO) scheduling discipline are deployed. However, TED is independent from the specific scheduling technique.

A. Operating Principles

The general, basic idea for ensuring fairness is to set the deadline (i.e., upper bound on packet service time) proportional to the RTT of TCP connections. Given that a shorter deadline will result in more losses, this should compensate the natural disadvantage of TCP connections with long RTT, thus enabling competing flows to fairly share available bandwidth.

The deadline D, can be set on a per-queue basis — Locally Bounded Delay (LBD) — or globally on the whole path — End to End Bounded Delay (EBD). The deadline, whether local or global, can be the same for all packets or set on a per-flow basis. In the latter case the deadline can be associated to packets, e.g., stored in their header in order to avoid nodes to handle per-flow information. As the goal of this paper is to present the basic idea of deploying TED to enforce fairness and assess it, implementation related issues and trade-offs are not addressed here and are left as future work.

EBD requires each packet to include either the time already spent inside the network or the time remaining before the deadline expiration[1]. Each node, before forwarding a packet, updates this information based on the time spent in the node. Packets are dropped by the first node finding an expired deadline.

[1] Specification of additional information and details on its inclusion in packet headers (i.e., field format, protocol level) are beyond the scope of paper.

TED guarantees that, if delivered, packets will reach their destination within a maximum service time $S_{\max}$ that is obtained by adding:

1. The propagation time on the links of the path
2. The processing and switching time of the traversed routers
3. The deadline, i.e., overall maximum amount of time that the packet can wait in the queues on the path.

Assuming that the first two addends are constant[2], they provide the minimum service time $S_{\min}$, i.e., the time a packet takes to travel across the network when queues are empty.

The total service time with EBD is:

$$S_{\max} = S_{\min} + D_{EBD}$$

The LBD deadline limits the time a packet can spend in each node on the path from source to destination. Each node when receiving a packet time stamps it with its time of arrival; if the packet cannot be forwarded within the deadline, it is discarded. Being the deadline a local queue parameter, per-packet LBD does not require any information to be included into packet headers and the resulting total service time is:

$$S_{\max} = S_{\min} + K \cdot D_{LBD}$$

where K is the number of nodes on the path from source to destination.

It is worth noting that although network nodes deploy time information in order to implement TED, no synchronization among them is required. Specifically

- Given that only local time is deployed, the value of the time-of-day does not need to be the same on different nodes,
- Given the deadline values practically deployed (see Section III for examples)

the accuracy of oscillators installed on commercial routers suffices and no frequency synchronization is required.

B. Queue Size

The size of a queue operating TED does not diverge; the upper bound on the number of packets in the queue $Q_{\max}$ devised below. Let

- K be the number of input links of a node,
- C_i the capacity of link i,
- $D_{\max}$ the maximum TED deadline, whether local or global, associated to a packet.

The worst case as far as accumulation of packets in a buffer is when:

1. All flows entering the node from its input interfaces must be forwarded through the same output port j,
2. Port j is temporary unavailable (e.g., because it is transmitting packets from a higher priority queue), and
3. Each arriving packet can spend in the queue of port j the maximum deadline $D_{\max}$.

The total amount of bits entering the buffer from input link i before the deadline of the first received packet expires and it is

[2] Although the processing and switching times are not constant, it is reasonable not to consider their variation in this context as in all practical cases it is negligible compared to propagation delay (on a global scale network), queuing delay, and queuing delay variation.

eliminated from the buffer is $D_{\max} \cdot C_i$. Consequently, the total amount of bits possibly accumulated in the queue of output port j is:

$$Q_{\max,j} = D_{\max} \cdot \sum_{1<i<k, i \neq j} C_i$$

If all the J links of the node have the same capacity C

$$Q_{\max} = (J-1) \cdot D_{\max} \cdot C$$

TABLE I shows maximum size reached by a TED queue on a router with 10 Gb/s interfaces for various port counts (rows) and maximum deadline values (columns). Most high-end routing products currently available on the market — the so-called terabit routers — feature buffers between 100 MB and 1 GB on 10 Gb/s interfaces. Consequently, if TED AQM were activated on such routers in most of their hardware configurations no packet would be dropped due to buffer overflow, i.e., packets would be lost only when their deadline expires. Moreover, adopting TED would enable reducing the amount of memory in routers without compromising network utilization and TCP performance.

TABLE I
MAXIMUM SIZE [MBYTE] OF QUEUES OPERATED WITH TED ON A ROUTER WITH 10 GB/S INTERFACES

Deadline (µs)	**Interfaces**				
	10	***20***	***30***	***40***	***50***
125	1,41	2,97	4,53	6,09	7,66
500	5,63	11,88	18,13	24,38	30,63
1000	11,25	23,75	36,25	48,75	61,25
2000	22,50	47,50	72,50	97,50	122,50
4000	45	95	145	195	245

TED operation is not compromised if a queue is smaller than $Q_{\max}$. However, packet losses due to overflow could affect the effectiveness of the presented approach to improve fairness of TCP congestion control. In fact, such losses do not follow the rationale of TED with deadlines set as discussed above and trigger a reaction from possibly "wrong" TCP connections. In this first work on TED the effects of buffer overflow are not considered and consequently the above equations are devised to calculate the size $Q_{\max}$ of each buffer. Future work will be devoted to the issues of buffer overflow and possibly to developing a TED variant in which when a queue reaches its maximum capacity packets are discarded based on their deadline and the time they have already spent in the network.

III. Simulation Results

Simulations were run to compare TED AQM and the DropTail queue management policy in terms of goodput and fairness in a typical high network load scenario. Simulations results not reported in the paper show that (*i*) in the given scenario (i.e., a number of TCP connections sharing a single bottleneck link, as it is likely to happen in real-life scenarios) and (*ii*) for the objective addressed by this work (i.e., improving the fairness of TCP congestion control) EBD-TED and LBD-TED provide practically identical outcomes and lead to the same conclusions. Consequently, although the presented simulation results were devised with EBD-TED this section generically refers to TED as the observations made and conclusions drawn equally apply to LBD-TED.

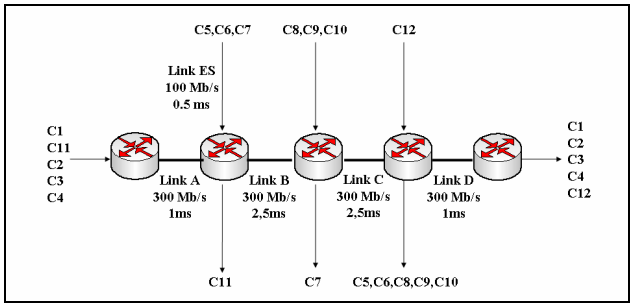

Figure 1 Network Topology

Figure 1 shows the network topology used for the simulations where TCP connections with different RTT are routed through a bottleneck link, i.e., link C with capacity 300 Mb/s. The overload condition of the bottleneck link was assessed in advance with a TCP throughput model [6] that provides the throughput of each TCP connection as a function of loss rate and round trip time. The starting time of TCP connections — modeled as having an unlimited amount of data to send — is calculated randomly and the total simulation time is 120 sec.

TABLE II
DEADLINE VALUES DEPLOYED IN THE SIMULATIONS

Simulations	***Deadlines (µs)***			
	4ms	***7ms***	***12ms***	***16ms***
S1	120	500	625	1000
S2	120	550	600	900
S3	120	650	750	900

The deadlines to be used with TED AQM were set according to the RTT of TCP connections. Specifically, the 12 TCP flows have one of four RTTs: 4 ms, 7 ms, 12 ms, and 16 ms. A different deadline was associated to each RTT and chosen with the aim of limiting the throughput of connections with short RTT. In case of congestion, this enables TCP connections with longer RTT to grab a fairer share of the bottleneck link bandwidth. As shown in TABLE II the simulations were run with three sets of deadlines.

The buffer associated to each link j has size $Q_{\max,j}$ as derived in Section II.B to avoid packet loss due to buffer overflow, i.e., in order to isolate the effects of TED and measure their impact on TCP performance. Simulations with the DropTail queue management policy using infinite buffers were run in order to devise an (ideal) TCP goodput to be used as a baseline for the comparison.

The graphs shown in this section (e.g., Figure 2 and Figure 3) plot the goodput of TCP connections normalized to their optimal — in terms of fairness — goodput O_i over the bottleneck link. The TCP connections in the figures are sorted in decreasing RTT order. The optimal goodput of a TCP connection is calculated as follow:

Let active connections traversing the bottleneck be sorted by RTT so that $RTT_j \geq RTT_i$ for all $i > j$, with $0 \leq i \leq N-1$ and $0 \leq j \leq N-1$,

$$O_0 = \min\left\{\frac{C_b}{N} \cdot \left(1 - \frac{h_o}{p_o}\right), \frac{W_o}{RTT_0}\right\}$$

$$O_i = \min\left\{\frac{C_b \cdot \left(1 - \frac{h_i}{p_i}\right) - \sum_{j=0}^{i-1} O_j}{N - i}, \frac{W_i}{RTT_i}\right\} \qquad 0 < i \le N - 1$$

where C_b is the capacity of the bottleneck link, N is the number of active connections, p_i and h_i are the mean packet length and the packet header length, respectively, RTT_i is the round trip time and W_i is the dimension of the maximum transmission window in bits for TCP connection *i*. Basically, the optimal goodput is either the maximum goodput allowed by the transmission window, or a fair share of the link bandwidth, whichever is the minimum. In the simulation scenario presented in Figure 1 the goodput of all TCP connections traversing the bottleneck link is not limited by the transmission window, i.e., $O_i = C_b / N, \forall i$.

The reminder of this section presents two sets of simulation results; the first set of simulations (Section III.A) deploys common TCP sources, while in the second set (Section III.B) the TCP loss recovery algorithm was modified in order to take advantage of the specificity of a network in which TED is being deployed.

A. TCP Reno

As it is graphically shown in Figure 2, the TCP congestion control algorithm divides the capacity of a fully loaded (i.e., bottleneck) link among active TCP connections in inverse proportion to their RTT. Consequently, unfairness among active connections arises. On the other hand, as shown in Figure 3, deployment of TED AQM with properly chosen deadlines results in long TCP connections (C1, C2, C3, and C4) enhancing their goodput and short ones (C8, C9, and C10) decreasing it.

Fairness can be more quantitatively assessed using the well known *Jain's Fairness Index*:

$$f = \left(\sum_{i=1}^{N} G_i\right)^2 \Big/ N \cdot \sum_{i=1}^{N} G_i^2$$

where N is the number of active connections and G_i is the normalized goodput of i_{th} TCP connection. This fairness index ranges from 0 to 1, with $f = 1$ representing optimal fairness.

As shown in TABLE III, TED with the deadlines set in simulation S3 yields a 15% and 24% fairness improvement respectively over DropTail queue management with infinite and finite (i.e., of size $Q_{\max,j}$ for each output interface j calculated as discussed in Section II.B for the chosen deadlines) buffers, respectively.

Figure 4 plots the goodput of TCP connections not traversing the bottleneck link C (i.e., C7, C11, and C12) in all simulation scenarios.

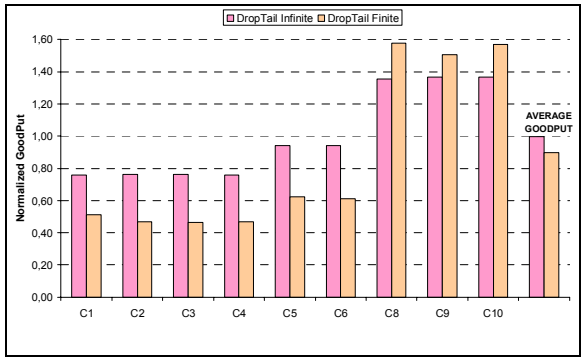

Figure 2 Normalized goodput of each TCP connection traversing the bottleneck and their average with DropTail queue management.

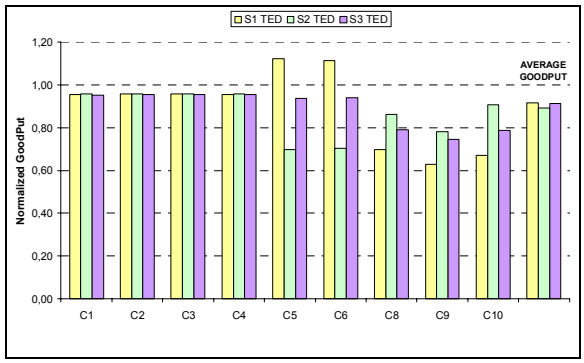

Figure 3 Normalized goodput of each TCP connection traversing the bottleneck link and their average with TED AQM.

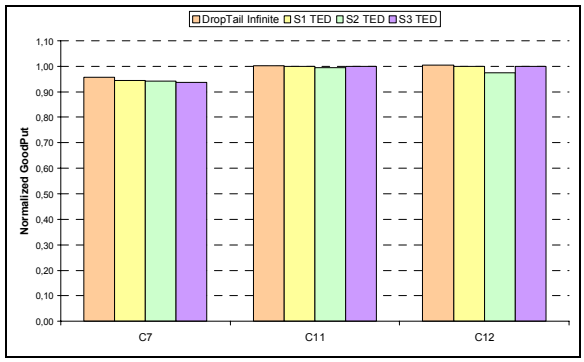

Figure 4 Normalized goodput of TCP connections not traversing bottleneck.

TABLE III
FAIRNESS INDEX PER SIMULATION

Simulations	Jain's Fairness Index
DropTail (Finite buffers)	0,759
DropTail (Infinite buffers)	0,84
S1	0,963
S2	0,985
S3	0,991

TED does not affect the performance of TCP connections when bandwidth is not limited: no significant difference can be observed when comparing the goodput TCP connections achieve when TED AQM and then DropTail[3] queue management are deployed. This results from TED being adaptive, i.e., affecting network operation only when congestion kicks in. Specifically, although in the considered

[3] Since the links traversed by the TCP connections being considered are not congested, there is no loss with the buffer size chosen for the simulation, hence the same goodput was measured in both DropTail simulation scenarios.

scenarios the same deadline is set for all TCP connections with the same RTT, packets flowing through parts of the network with enough available capacity (links A, B and D) do not reach their deadline and the corresponding TCP connections (i.e., C7, C11, and C12) achieve the maximum throughput allowed by their transmission window. Instead, packets of connections traversing the bottleneck link spend more time in its buffer, their deadline possibly expires, and the corresponding TCP senders reduce their sending rate.

A comparison of the average goodput plotted in Figure 2 and Figure 3 shows a decrease when TED AQM is deployed with respect to DropTail queue management with infinite buffers. However, in authors' opinion this is not a major drawback for the following reasons:

- A trade-off between fairness and goodput is common to most approaches to improve fairness (see for example [8]).
- In the presented network scenarios a 5% network goodput decrease is traded for a 15% fairness improvement
- The average goodput achieved with TED is higher than with DropTail queue managements with the same buffer dimension (i.e., $Q_{\max,j}$ for each output interface j as devised in Section II.B).

To conclude, it is important mentioning that TCP source synchronization phenomena could not be observed in the traces of any presented simulation.

B. Improved Loss Recovery

In its basic principle of operation TED controls TCP connections by dropping packets in the network, which could lead to a potentially significant degradation of network performance in terms of overall amount of information successfully transferred by TCP-based applications. Although the presented simulation results show that goodput decrease with TED AQM is not significant, this negative side effect could be further limited by modifying the TCP loss recovery algorithm in order to achieve higher goodput notwithstanding packet loss. In the context of TED this is particularly important for connections with shorter round trip time that are given a shorter deadline.

TED operation is simpler and more effective when packets of the same flow follow the same route. This is not strictly required, but it anyway happens in the not uncommon cases of stable routing and label switched path (LSP) provisioning over MPLS clouds. Under such circumstances a TCP receiver does not experience out of order arrivals, making it possible to discover a packet loss at time of arrival of the first duplicated acknowledgment. Consequently, in the context of this work the TCP Reno behavior was modified so that the first duplicated acknowledgment triggers the fast retransmit algorithm, thus improving reactivity to packet loss, hence goodput. In the following, simulation results are presented for TCP connections with modified loss recovery with TED AQM (TED 1 ACK), where buffer dimension is $Q_{\max,j}$ and deadlines of simulation scenario S3 are deployed.

Simulation results devised for TCP connections with modified loss recovery and DropTail queue management (not reported here) are for all practical purposes identical to the ones devised with conventional TCP Reno and previously presented. This is due to the fact that in the given network scenario, as discussed in [6], retransmissions are triggered by the expiration of the retransmission time-out (RTO). In particular, since TCP senders are assumed to have an unlimited amount of data to send, packets are transmitted in bursts (of a congestion window size); when a burst hits a overflowing queue, its whole tail (or possibly the whole burst) is discarded. Consequently, the sender will wait in vain for the corresponding acknowledgements until the RTO expires.

When DropTail queue management is deployed shortening the reaction time of TCP senders to packet loss is not necessarily an advantage. In fact, when congestion arises queues fill up completely and bursts of packets are being discarder. Immediate retransmission can result in retransmitted packets finding buffers still (almost) full, hence perpetuating the congestion state and causing more packets to be discarded. Instead, with TED AQM packets are dropped based on their deadline, most likely before queues become full and, as confirmed by the results presented below, a prompt reaction of TCP senders can improve its goodput without perpetrating the network congestion state.

A second modification to TCP aimed at improving its goodput is related to the calculation of its RTO. Heterogeneous communication networks with their variety of application demands, uncertain time-varying traffic loads, and mixture of wired and wireless links pose several challenging problems in modeling and control. One of the major problems is setting the TCP time out based on a roundtrip time (RTT) estimation, which is a particularly important for efficient end to end congestion control, especially in scenarios where dynamically changing traffic flows cause a bottleneck link to rapidly build up a queue, which in turn induces rapid RTT changes. TCP uses different algorithms to estimate a connection RTT, but all of the only provide an approximation. As demonstrated in [6], in many scenarios the majority of window decrease events is due to time-outs rather then fast retransmit. If the time-out is set too long, possibly due to an over estimation of the RTT, the TCP algorithm does not promptly react to loss and connection performance (in terms of throughput and goodput) decreases.

If TED is deployed in the network, an upper bound on the end to end delay can be calculated based on the deadline set in network nodes. Specifically, $D_{\max}$ is the maximum one way delay of a packet belonging to a TCP connection and the maximum RTT can be calculated as the sum of $D_{\max}$ corresponding to each way. The TCP RTO can thus be set to maximum RTT on the connection path so that TCP operates with a precise, deterministic, and shorter RTO, hence being more reactive in recovering from losses. Figure 5 shows, for connection C11, the maximum RTT — to be possibly used as RTO — resulting from TED AQM with the deadlines set in simulation scenario S3 and the RTO calculated by TCP based on Karn's algorithm [7] with DropTail queue management.

Figure 6 plots the goodput of short TCP connections and the average overall goodput on the bottleneck link with combinations of the two queue management policies and loss recovery variants described so far.

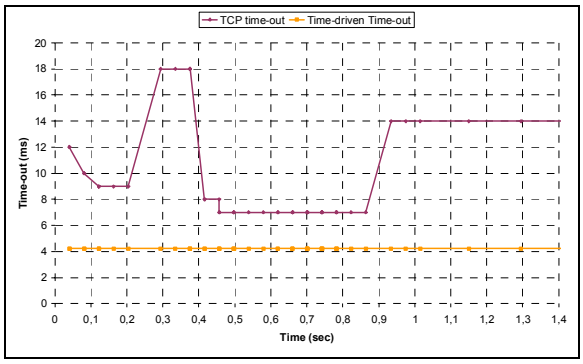

Figure 5 Time-Driven RTO vs RTO estimated by TCP Reno for C2

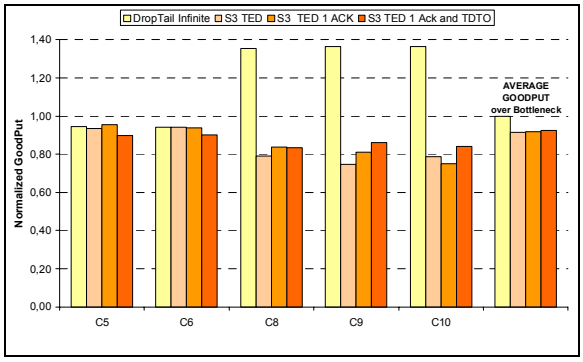

Figure 6 Normalized goodput of short TCP connections and average goodput over bottleneck link.

As expected, modifications to the loss recovery algorithm result in improved goodput of shorter TCP connections (for which a shorter deadline has been set) with TED AQM. This improves the overall network goodput and utilization. The goodput achieved with the DropTail queue management policy with infinite buffer is plotted as an upper bound on the achievable overall goodput on the bottleneck link. As it can be seen in Figure 6, although the goodput with DropTail queue management is still higher, deployment of TED AQM and both the 1ACK and TED-driven RTO (TDTO) modifications result in a 4 Mb/s increase of the average goodput on the bottleneck link, of which 2,62 Mb/s stemming from the 1 ACK modification and the rest stemming from the TED RTO modification. The fairness index shown in TABLE IV demonstrates that the two above described modifications to TCP's loss recovery algorithm do not compromise the capability of TED's to improve fairness.

TABLE IV
JAIN'S FAIRNESS INDEX WITH MODIFIED LOSS RECOVERY

Simulations	Jain's Fairness Index
DropTail Finite 1 ACK	0,759
TED 1 ACK	0,997
TED 1 ACK and TDTO	0,997

IV. CONCLUSION AND FUTURE WORK

This paper addresses the fundamental and widely studied issue of fairly sharing congested links with end-to-end congestion control, such as in TCP, and presents a solution based on Time-driven Early Discard (TED), a novel AQM (advanced queue management) technique. Packets waiting in a queue are dropped based on a deadline on the packet service time. Various options have been described for the application of the deadline (i.e., per-node vs. end-to-end deadlines, per-packet vs. per-flow deadlines) and one of them (i.e., per-flow, end-to-end deadlines) assessed by simulation in scenarios featuring different choices of the system parameters. The presented results show the effectiveness of the proposed approach in ensuring that TCP connections fairly share bottleneck link when the deadline associated to a connection is proportional to its round trip time (RTT).

In conditions of light load packets do not spend much time in queues, do not reach their deadline, and no packet is discarded — i.e., TED is adaptive. When congestion arises, packets with shorter deadlines (i.e., packets belonging to TCP connections with shorter RTT) are the first being dropped. These packet losses trigger the TCP congestion control algorithm that reduces the connection throughput, i.e., fairness is enforced by constraining shorter TCP connections. As in most other proposed solutions to the fairness problem (see [8] as an example), the improvement in fairness is traded for a reduction of the overall goodput. However, the presented results show a 7% goodput reduction in face of a 15% fairness improvement.

Moreover, the presented results show that the goodput reduction can be limited to 5% with two simple modifications to the mechanisms that govern TCP reaction to loss: (*i*) detecting packet loss with the reception of a single duplicated acknowledgement (rather then three) and (*ii*) setting the retransmission time out according to the TED deadline associated to the TCP connection.

A delicate issue with regard to the presented approach is the algorithm for choosing the deadline to be associated to each flow and its implementation. This issue together with the assessment of the impact of the choice of non-optimal deadlines will be the object of further work. Another direction for future studies is a variant of this novel AQM that uses service time as a fundamental parameter to enforce fairness and goodput by limiting the oscillation of the congestion control window, while avoiding packet dropping. For example, packet marking or monitoring of the one-way delay variations could be deployed for this purpose.

REFERENCES

[1] S. Floyd, V. Jacobson, et al., "Recommendations on Queue Management and Congestion Avoidance in the Internet," Request for comments: 2309, April 1998.

[2] S. Floyd and V. Jacobson, "Random early detection gateways for congestion avoidance," IEEE/ACM Transactions on Networking, 1(4):397-413, August 1993.

[3] T. Hamann and J. Walrand, "A new fair window algorithm for ECN capable TCP (new-ECN)," IEEE INFOCOM 2000, March 2000.

[4] D. Linand R. Morris, "Dynamics of random early detection," ACM SIGCOMM'97, pp.127-137, October 1997.

[5] I. Stoica, S. Schenker and Zhang, "Core stateless fair queuing: Achiving approximately bandwidth allocations in high speed network," ACM SIGCOMM 1998, pp 118-130, September 1998.

[6] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose. "Modeling TCP throughput: A simple model and its empirical variations," ACM SIGCOMM 1998, September 1998.

[7] V. Paxson, M. Allman, "Computing TCP's Retransmission Timer," Request for Comments: 2988, November 2000.

[8] W.-C. Feng, A. Kapadia, S. Thulasidasan, "GREEN: proactive queue management over a best-effort network," in Proceedings of GLOBECOM 2002, pp. 1774-1778, November 2002.