The Interaction between TCP performing a Bandwidth Estimation and RED queue management

ANTONIO CAPONE, FABIO MARTIGNON
Dipartimento Elettronica e Informazione
Politecnico di Milano
Piazza Leonardo da Vinci 32, 20133 Milan
ITALY
capone, martigno@elet.polimi.it

Abstract: - The Time Intervals based Bandwidth Estimation Technique (TIBET) proposes to change the TCP congestion control algorithm adding a sender-side estimation of the bandwidth available to the connection. TCP sources implementing the TIBET Algorithm can achieve higher performances than actually deployed TCP versions, especially in presence of links affected by random losses. In this paper we study the interaction between TCP sources performing a direct bandwidth estimation and RED queue management, and we compare their performances with those of existing TCP versions.

Key-Words: - TCP, Bandwidth Estimation, RED, Wireless Networks.

1 Introduction

The actual versions of the Transmission Control Protocol (TCP), i.e TCP Reno and NewReno, have shown to achieve poor performances over links affected by random, sporadic losses, such as wireless links. The main reason to these degraded performances is the confusion the classical TCP sources make between random losses, due for example to sudden fading over wireless channels, and true congestion signals due to high network load ([1, 2]). To solve this problem, various modifications to the TCP algorithm have been proposed as those implemented by TCP Vegas [3] and TCP Westwood [4]. They both perform an explicit estimation of the bandwidth used by the connection, and use this estimate to avoid unnecessary reductions of the transmission rate by making a distinction between random losses and true congestion.

However, we have shown in [5] that all these schemes do not perform well, either because the estimation scheme is highly biased (TCP Westwood), or because they cannot fairly share network resources with other connections (TCP Vegas). For these reasons, we have proposed a new scheme, the Time Intervals based Bandwidth Estimation Technique (TIBET) which succeeds in obtaining good estimates of the bandwidth used by a TCP source. As TIBET needs no cooperation from the peer TCP, it can be implemented by modifying the sender-side only of a TCP connection, thus allowing the possibility of its immediate deployment in the Internet. Moreover, TIBET doesn't show any unfair behavior towards existing versions of TCP, such as TCP Reno and NewReno.

In this paper we study the performance of TIBET in presence of an algorithm of queue management which will be probably deployed in the future Internet, Random Early Discard (RED) [6]. RED routers discard incoming packets with a probability which increases with the buffer occupation, thus making the wired links over which it runs exhibit a behavior similar to wireless ones.

The rest of this paper is structured as follows: in Section 2 we briefly illustrate the TIBET algorithm. Section 3 studies the performance and the behavior of TIBET and other TCP performing a direct bandwidth estimate in presence of RED queues, comparing their performances with existing versions of TCP. Finally, Section 4 concludes the paper.

2 TIBET algorithm

In this section we illustrate the Time Intervals based Bandwidth Estimation Technique (TIBET), which succeeds in obtaining correct estimates of the bandwidth used by the TCP source. TIBET is able to cope efficiently with many of the bandwidth estimation problems known in literature, including Clustering and ACK Compression [7].

To explain the rationale of TIBET let us refer to the example in Figure 1 where transmissions occurring in a period $T$ are considered. Let $n$ be the number of packets belonging to a connection and
$L_1, L_2, ..., L_n$ the lengths, in bits, of these packets.

![Packet timing structure](image)

Fig. 1: Packet timing structure

The average bandwidth used by the connection is simply given by $\frac{1}{T} \sum_{i=1}^{n} L_i$. If we define $L = \frac{1}{n} \sum_{i=1}^{n} L_i$, we can express the bandwidth ($Bwe$) used by the connection as:

$$Bwe = \frac{nL}{T} = \frac{L}{\frac{T}{n}}$$

(1)

The basic idea is to perform a run-time senderside estimate of the average packet length, $L$, and the average interarrival, $\frac{T}{n}$, separately. This can be done in two ways:

- by measuring and low-pass filtering the length of acked packets and the interarrival times between consecutive ACKs;

- by measuring and low-pass filtering directly the packets' length and their interdeparture times, according to the self-clocking nature of TCP[8].

If we consider the minimum RTT ever recorded by the TCP source ($RTT_{\text{min}}$) as a good estimator of the end-to-end propagation delay, then we can set:

$$ssthresh = Bwe \times RTT_{\text{min}}$$

(2)

which has been proposed as an optimal value for $ssthresh$ in [9].

The $ssthresh$ is set to the value of equation (2) only after three duplicate ACK's are received by the source, or after a coarse-grained timeout expiration, following the guidelines of [4]. This choice is supported by the worst performance observed with more frequent updating of $ssthresh$ as discussed in [5].

The congestion window, instead, is set to the same value of $ssthresh$ after three duplicate ACK's, and it’s reset to 1 segment after timeout expirations.

Simulation results presented in [5] show that TIBET is not biased, and obtains bandwidth estimates which oscillate around the fair-share value when all TCP sources experience almost the same path conditions.

The issue of fairness has also been considered in [5], showing the ability of TIBET sources to fairly share network resources in presence of different versions of the TCP protocol. Simulation results have also shown that the strength of TIBET lies in its scalability: as more connections share the bottleneck link, as the estimate variance reduces.

The presence of constant rate flows, such as UDP flows for IP telephony or video conference, makes TIBET perform better as it reduces the dimension of packet clusters.

3 Red queues

The RED algorithm has been proposed to provide a more fair queuing policy than drop-tail, which is actually deployed by almost every IP router all over the Internet. Several studies and measurements have stressed the limits of drop-tail policy, which simply consists in discarding packets which reach the router when its buffer is full.

It has been shown in [10] that drop-tail queues cause synchronization of TCP flows. When the router’s buffer is full, in effect, almost all TCP connections which traverse the router will lose packets contemporarily. All these connections will thus re-enter at the same time the slow-start phase. This causes great oscillations in buffer occupation, as well as a general decay of TCP performances.

![Packet Drop Probability of a RED queue as a function of the Average Queue Size avg](image)

Fig. 2: Packet Drop Probability of a RED queue as a function of the Average Queue Size $\text{avg}$

Drop-tail queues also exhibit the so-called phase effects. The deterministic dropping technique used by these queues, which accept or discard packets provided there’s enough space in the buffer, may constantly penalize some connections, as their packets arrive at the router mainly when its buffer is full. Evidently, this phenomenon is strictly related with flow synchronization we cited above, and may cause great disproportions in the goodput achieved by different connections.

The RED algorithm aims at resolving both the problems, guaranteeing at the same time low oscillations in buffer occupation as well as a more fair share of buffer resources among different TCP connections. The studies and network measurements have confirmed RED’s effectiveness, thus making the IETF recommend its utilization in IP routers.
The whole behavior of RED, depicted in Figure 2, can be summarized as follows: when the average queue length (avg) exceeds a given threshold \( \min_{th} \), the RED algorithm starts discarding new packets with a probability proportional to \( \text{avg} \) itself. The discarding probability increases up to a maximum value, \( p_{\text{max}} \), which is reached when \( \text{avg} = \max_{th} \). Finally, when \( \text{avg} \) exceeds the upper threshold \( \max_{th} \), each incoming packet is discarded.

Hence, RED causes wired, error-free links to behave similarly to wireless lossy links. Thus we expect that, also in this case, TIBET could allow to better exploit link capacities. This is confirmed by the results in Figure 3 which show the goodput achieved by TIBET and Reno versus the maximum dropping probability \( p_{\text{max}} \). The simulation scenario considers a single TCP connection transmitting over a 10 Mb/s link. The other RED parameters are those suggested by its authors in [6]: \( \min_{th} = 5 \) and \( \max_{th} = 15 \) packets.

All the simulated results presented in this paper have been obtained using the Network Simulator, ns’ ver. 2 [11], for which we developed modules implementing the TIBET algorithm.

![Figure 3: Link utilization of TIBET and TCP Reno with RED queue. Link capacity 10 Mbit/s.](image)

Figure 4 furtherly proves TIBET performances, showing the results of simulations involving TIBET and TCP Reno connections running over a 5 Mb/s link, where the RED parameters are the same considered before. We have observed similar results also for different link speeds.

A possible reason behind these results could be a higher buffer occupancy with TIBET due to a more aggressive congestion control scheme. But this is not the main reason. Figure 5 shows a time trace of the queue size and the corresponding drop probability obtained with 5 TCP connections, TCP Westwood, Reno or TIBET, over a 5 Mbit/s link (RTT=50 ms, \( p_{\text{max}} = 0.1 \), \( \min_{th} = 1 \), \( \max_{th} \) equal to the bandwidth-delay product).

The average queue size resulting from 5 TCP Reno connections (which is not shown in the figure for the sake of clarity) oscillates around the value of 6 packets. Consequently, the RED queue will discard incoming packets with probability oscillating around 1.6%.

![Figure 4: Link utilization of TIBET and TCP Reno with RED queue. Link capacity 5 Mbit/s.](image)

![Figure 5: Queue length and corresponding Packet Drop Probability on a 5 Mb/s link in presence of 5 TCP Reno or 5 TCP with TIBET.](image)

Figure 5 shows instead the average queue size with 5 TIBET connections transmitting (the lower line), which oscillates around 7 packets, and with 5 TCP Westwood connections (the upper line), which oscillates around 16 packets and over. Evidently, \( \text{avg} \) is greater in both the last situations than TCP Reno, but TCP Westwood shows an overly aggressive behavior due to its overestimate of the available bandwidth, as we have pointed out in [5]. Hence, TCP Westwood sources induce a packet drop probability much greater than TIBET ones, i.e. about 4.9% for TCP Westwood against 2% for TIBET, in this scenario. Evidently, it's necessary to correctly estimate the available bandwidth, as TIBET does, or else an unfair division
of network resources will be obtained. Finally, we have to underline how the difference in \( \text{avg} \) is not so stressed between TIBET and TCP Reno (i.e. 7 against 6 packets).

These results appear satisfactory for TIBET since they prove its ability to achieve a higher link utilization with RED queues. However, one could also consider them from a different point of view: since the new algorithm allows a higher link utilization, it may be unfair in mixed scenarios. Figure 6 shows the results obtained with two connections, one TCP Reno and one TCP running TIBET, sharing a 5 Mbit/s link (propagation delay equal to 50 ms). The \( \min_{nh} \) parameter has been set to 1 packet, while \( \max_{nh} \) to the bandwidth-delay product of the connection (in packets).

![Figure 6: Link utilization of a 5 Mbit/s link with RED queue](image)

The graph shows for each connection the link utilization, i.e. the ratio between the goodput achieved by the connection and the total goodput, versus \( p_{\max} \). We observe that the link utilization of TIBET is 50% higher than that of Reno in most of the cases. As \( p_{\max} \) diminishes, the difference becomes less remarkable until it disappears when \( p_{\max} = 0 \) and RED algorithm becomes equivalent to the drop-tail one. If most of the Internet routers would adopt the RED algorithm, we should admit that the smooth introduction of TIBET would not be so easy. But, since the use of RED queues is at the moment limited to few experimental testbeds, we believe the behavior of TIBET with RED queues can be considered as a potential advantage.

4 Conclusions

In this paper we have analyzed the performances of TIBET Algorithm, focusing over the interaction between TIBET and RED queue management and showing how TIBET is able to achieve high link utilizations allowing connections to better exploit network resources with little impact over fairness.

References:


