Analysis of TCP-Friendly Protocols for Media Streaming

Gábor Hosszú
Budapest University of Technology and Economics, Budapest

Dávid Tegze
Budapest University of Technology and Economics, Budapest

INTRODUCTION

This chapter presents various congestion control schemes for transport protocols together with a number of metrics for the evaluation of these protocols with special emphasis on fairness-related measures. The paper analyzes some properties of the TFRC algorithm, which provides applications with congestion control mechanisms that can be applied for streaming media. Streaming media is a delivery method of media content, which is simultaneously received by, and displayed to, the end-user while it is being delivered by the provider. It is hard to explore the various protocol mechanisms implemented in various protocols in a uniform manner; therefore the SimCast (simulator for multicast) simulator has been developed for traffic analysis of the unicast (one-to-one) and multicast (one-to-many) streams. This article evaluates some TCP and other transport protocol metrics based on results from the SimCast simulator (Orosz & Tegze, 2001). The simulated results are presented along with the evaluated measures. Due to spreading of traffic lacking end-to-end congestion control, congestion collapse may arise in the Internet (Floyd & Fall, 1999). This form of congestion collapse is caused by congested links that are sending packets to be dropped in the network later. The fundamental factor behind this form of congestion collapse is the absence of end-to-end feedback. On the one hand, an unresponsive flow fails to reduce its offered load at a router in response to an increased packet drop rate, and on the other hand, a disproportionate-bandwidth flow uses considerably more bandwidth than other flows in time of congestion. In order to achieve accurate multicast traffic simulation—being not so TCP-friendly yet—the effects of the flow control of the TCP protocol should be determined (Postel, 1981). However, there are many different kinds of TCP and other unicast transport protocol implementations with various flow control mechanisms, which make this investigation rather difficult (He et al., 2005).

Up to now, a lot of comparisons have been done. For example, Wang et al., reviewed the TCP-friendly congestion control schemes in the Internet (Wang et al., 2001). They differentiated two groups of the TCP-friendly congestion control algorithms as follows: (1) end-to-end and (2) hop-by-hop congestion control mechanisms. The end-to-end mechanisms are grouped into (i) AIMD-based schemes (AIMD: additive increase multiplicative decrease) with the window- and rate-adaptation schemes, (ii) modeling-based schemes, including equation-based congestion control schemes and the so-called model-based congestion schemes, and (iii) a combination of AIMD-based and modeling-based mechanisms. Wang’s classification is mostly used in our discussion, too.

Yu proposes another important approach to the survey on TCP-friendly congestion control protocols for media streaming applications (Yu, 2001), in which several TCP-friendly congestion control protocols were discussed via a comparison of many important issues that determine the performance and fairness of a protocol.

It is an important advantage of the simulator SimCast that it implements recent congestion control mechanisms. In this way, the simulator is capable of the examination of cooperation among different TCP protocol entities, or various other transport level protocols (Shalunov, 2005).

This chapter presents the basic congestion control mechanisms of TCP, then it demonstrates the way media streams control their sending rate to avoid congestion. After this, a couple of metrics for various transport layer
attributes are defined. Then some fairness measures of concurrently running flows are evaluated, based on sending rate traces of simulated TCP and TFRC protocol entities. Lastly, conclusions are drawn, and work to be done identified.

OVERVIEW OF THE TCP CONGESTION CONTROL

The Basic Control Mechanisms

TCP congestion control is based on the use of a sliding window. Its main concept is that the sender can only send a limited number of unacknowledged segments to the receiver (Van Jacobson, 1988). The number of segments to be sent without receiving acknowledgement is determined by the congestion window \( Cwnd \). The \( Cwnd \) is given in bytes, which is the total length of the segments that belong to the congestion window (Floyd, 2001).

TCP congestion control is based on additive increase multiplicative decrease (AIMD), halving the \( Cwnd \) for every window containing a packet loss, and increasing the \( Cwnd \) by roughly one segment size per round trip time (RTT) otherwise.

Retransmit timers are of fundamental importance in highly congested systems, which have exponential backoff of the retransmit timer when a retransmitted packet itself is dropped.

The slow-start mechanism is for initial probing of available bandwidth, instead of initially sending it at a high rate that might not be supported by the network (Stevens, 1997). At the beginning of the Slow-Start state, the \( Cwnd \) equals one segment size. During slow-start, the \( Cwnd \) is increased with a squared function in time.

ACK-clocking is the mechanism that uses the arrival of acknowledgements at the sender to clock out the transmission of new data.

CONGESTION CONTROL OF MEDIA STREAMS

TCP-friendly rate control (TFRC) is proposed for equation-based congestion control that explicitly adjusts the sending rate as a function of the measured rate of loss events (Handley et al., 2003). The TFRC is a receiver-based congestion control mechanism, with calculation of the loss event rate performed in the data receiver rather than in the data sender. This is appropriate for an application where the sender is a large server handling many concurrent connections. Therefore, this is suitable as a building block for multicast congestion control. The TFRC is not a complete protocol; it is a congestion control mechanism only. It could be implemented in a transport protocol like real-time transport protocol (Schulzrinne, 1996) or in an application incorporating end-to-end congestion control at the application level.

TFRC uses the following throughput equation directly to determine the allowed sending rate as a function of the loss event rate and the RTT. This equation is a simplified version of the throughput equation for the Reno-TCP (Padhye et al., 1998).

\[
T = \frac{s}{R \cdot \sqrt{\frac{2bp}{3}} + t_{RTO} \left( \frac{3bp}{8} \right) p \left( 1 + 32p^2 \right)},
\]

where \( T \) is the transmit rate in bytes/second, \( s \) is the packet size in bytes, \( R \) is the RTT in seconds, \( p \) is the loss event rate, \( t_{RTO} \) is the TCP retransmission timeout value in seconds, \( b \) is the number of packets acknowledged by a single TCP ACK. For simplicity, \( t_{RTO} = 4R \) and \( b = 1 \) in most cases, however, if the competing TCP implementations use “delayed ACKs,” \( b = 2 \) is a more appropriate value.

During operation, the TFRC receiver measures the loss event rate and feeds this information back to the data sender. Then the sender uses timestamp fields in feedback messages to measure the RTT and feeds these measured values into the throughput equation (1) to get the acceptable transmit rate. The sender then adjusts its transmit rate to match the calculated rate.

The TFRC is reasonably fair when competing for bandwidth with TCP flows, but has a much lower variation of throughput over time compared with TCP, making it more suitable for applications such as streaming media, where a relatively smooth sending rate is of importance. The flow is “reasonably fair” if its sending rate is generally within a factor of two of the sending rate of a TCP flow under comparable conditions (Handley et al., 2003).

The drawback of smoother throughput than TCP while competing fairly for bandwidth is that TFRC