Future Network and MobileSummit 2013 Conference Proceedings
Paul Cunningham and Miriam Cunningham (Eds)
IIMC International Information Management Corporation, 2013
ISBN: 978-1-905824-36-6

Poster Paper

Reducing Internet Transport Latency for Thin Streams and Short Flows

Abstract: The present Internet limits the performance of applications that need real-time interaction. This is in part because the design of the network has been optimised to boost throughput, maximising efficiency for bulk applications. However, changes in use have resulted in that an increasing number of applications now depend on timely delivery. One of the targets of the RITE project is to reduce internet transport latency in support of such applications. Initial results from the project on how end nodes can be optimized for more timely error recovery are presented in this poster.

Keywords: Latency, TCP, thin streams, short flows, loss recovery

1. Introduction

Historically the Internet community has worked to improve throughput and resource utilisation. We believe that the time is now ripe for research to focus instead on latency, because there are more and more applications and scenarios where low latency is critical: In financial trading, a millisecond less latency is worth up to tens of millions of Euros per year to a major brokerage firm; in multiplayer online games, increased end-to-end delays severely harm the perceived service quality; in interactive multimedia services, like immersive video conferencing, quality and understanding is reduced if latency is high; in general interactive use of the Internet (e.g. Web 2.0 applications), which typically involves sequences of numerous small objects, each one can get held up in buffers occupied by other traffic, resulting in cumulative delays of seconds rather than milliseconds. The common feature of all the above is that latency matters. The focus on throughput at all costs is no longer appropriate partly because the extra capacity being deployed means that increased utilisation is no longer the most critical research topic, and partly because delay-sensitive applications often need to send less than the network allows.

To meet the needs of an increasing share of flows that require low latency transfer the RITE FP7 project aims to develop, deploy and evaluate novel mechanisms that can reduce end-to-end transmission latency experienced by interactive applications in every leg of the network. This poster introduces initial results on some of the mechanisms that are currently under study within RITE, focusing on mechanisms that can be deployed at the end nodes in support of thin streams and short flows. In particular, two ongoing works to optimize error recovery are introduced and initial performance results are presented.

2. Loss Recovery Optimizations

TCP (and SCTP) uses two mechanisms to detect segment loss. First, if a segment is not acknowledged within a certain amount of time, a retransmission timeout (RTO) occurs, and the segment is retransmitted. The RTO is based on measured round-trip times (RTTs) between the sender and receiver. Second, when a sender receives duplicate acknowledgments, the fast retransmit algorithm infers segment loss and triggers a

retransmission. Duplicate acknowledgments are generated by a receiver when out-of-order segments arrive. As both segment loss and segment reordering cause out-of-order arrival, fast retransmit waits for three duplicate acknowledgments before considering the segment as lost.

Application-limited streams will in many cases not probe for bandwidth and expand the congestion window when using congestion-controlled transport. The effect is that mechanisms that were built in for speedy recovery of lost segments, like fast retransmit, will not be triggered for such streams, leaving all losses to be recovered by slow timeouts. For time-dependent applications, this is not good [1]. Below, we describe two ongoing works for improving this situation.

2.1 RTO restart

The current RTO management algorithm in TCP (RFC6298 [2]) recommends that the retransmission timer is restarted when an ACK that acknowledges new data is received and there is still outstanding data. The restart is conducted to guarantee that unacknowledged segments will be retransmitted after approximately RTO seconds.

This approach causes retransmissions to occur RTO seconds after the last ACK has been received, not RTO seconds after the transmission of the lost segment(s). In most cases, this adds approximately one round-trip time (RTT) to the loss recovery time. If the ACK that triggers the restart also is a delayed ACK, then the total loss recovery time can become RTO + RTT + delACK, where delACK corresponds to the receiver's delayed ACK setting. This delay can be significant for time-sensitive applications.

To enable faster loss recovery for thin streams and short flows we have proposed an alternative restart algorithm (RTO restart) for situations where fast retransmit does not apply (less than four packets are outstanding). By resetting the timer to "RTO - T_earliest", where T_earliest is the time elapsed since the earliest outstanding segment was transmitted, retransmissions will always occur after exactly RTO seconds. Further details on the RTO restart algorithm can be found in [3].

The RTO restart algorithm has been implemented in the Linux 3.7 kernel. Figure 1 illustrates the effect of the algorithm. The experiment was conducted in a setup with three machines, where the traffic between the sender and the receiver passed through the third machine where the KauNet[1] network emulator introduced 1% random packet loss and a varying path propagation delay with a mean of 200ms. The figure compares the results with and without RTO restart in runs where a loss occured for at least one of the last three packets of the flow (i.e. flows where RTO restart can have an effect). As can be seen in the figure, RTO restart reduces the recovery time by roughly one RTT.

2.2 Live game server evaluation of thin-stream retransmission mechanisms

Below we describe two mechanisms to optimize thin-stream retransmissions and evaluate them on a "live" game server for Funcom's Massively Multiplayer Online Game (MMORPG) "Age of Conan"[2]. In our implementation of the two evaluated mechanisms, the sender performs a check for how many packets the stream has "on the wire". If the number of outstanding packets is less than or equal to 4, the stream is considered

[1] http://www.kau.se/KauNet
[2] http://www.ageofconan.com

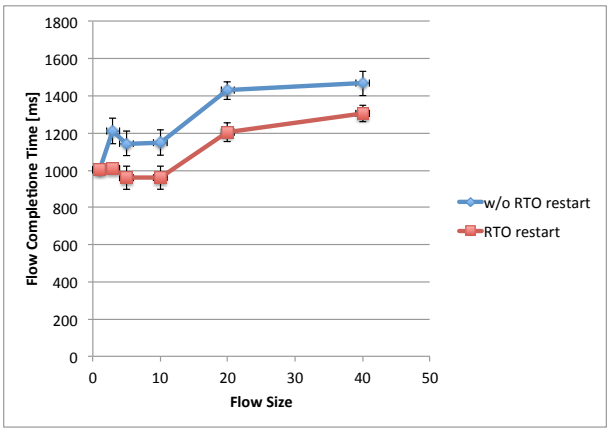

Figure 1: Flow completion time as a function of flow length, with and without RTO restart.

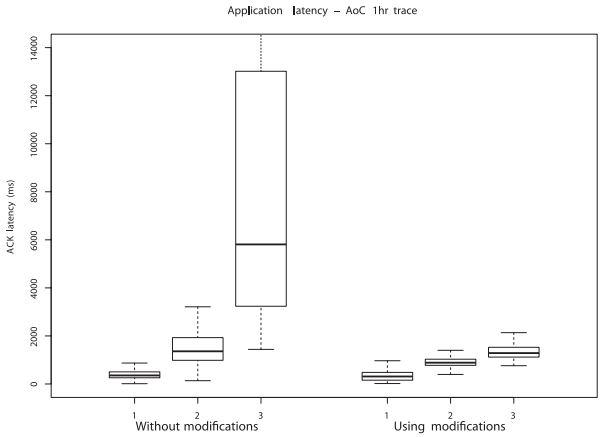

Figure 2: Modified vs. Traditional TCP in Age of Conan. The box shows the upper and lower quartiles and the average values. Maximum and minimum values (excluding outliers) are shown by the drawn line. The plot shows statistics for the first, second and third retransmissions.

as unable to trigger fast retransmit, and will enable the thin-stream retransmission mechanisms. The two evaluated mechanisms are as follows:

1. Perform a "fast retransmit" on the first dupACK, thus reacting on the first indication that loss has happened.

2. Do not exponentially increase the RTO until 6 retransmissons using the base RTO has been performed, thus increasing the chance of recovering the segment without extreme delays.

For a detailed description and lab evaluations of the mechanisms, see [4].

For the evaluation presented below, the mechanisms were deployed in a "live" game server. Two traces were captured on the server side by Norwegian game company Funcom[3]. One of the traces used the regular TCP settings, in the other, the thin-stream modifications were enabled.

From the captured one-hour trace from the Funcom server, we saw more than 700 players (746 for the traditional and 722 for the modified TCP tests), where about 300

[3]http://www.funcom.com

streams in each experiment experienced loss rates between 0.001% and 10%. Figure 2 shows the results from an analysis of the three first retransmissions. When the data segment is recovered after only one retransmission, the mechanisms have no effect. In this case, the average and worst-case latencies are still within the bounds of a playable game. However, as the users start to experience second and third retransmissions, severe latencies are observed in the traditional TCP scenario, whereas the latencies in the modified TCP test are significantly lower. The effect for the end-user is that the effect of application layer stalls caused by subsequent losses of the same data segment is drastically reduced.

3. Conclusions

We have provided a short introduction to the importance of reducing latency for interactive Internet applications, which is the foundation for the RITE FP7 project. To exemplify the project's work, we have presented two examples of preliminary work that focuses on reducing latency from an end-system perspective. Our results show that the traffic patterns generated by interactive applications need special consideration in the transport protocols, and that significant latency reduction can be achieved by considering such needs in the protocol design.

4. Acknowledgements

This work was funded by the European Community under its Seventh Framework Programme through the Reducing Internet Transport Latency (RITE) project (ICT-317700). The views expressed are solely those of the author.

References

[1] C. Griwodz and P. l. Halvorsen, "The fun of using TCP for an MMORPG," in *Proceedings of the 2006 international workshop on Network and operating systems support for digital audio and video NOSSD AV 06*, p. 1, ACM Press, 2006.

[2] V. Paxson, M. Allman, J. Chu, and M. Sargent, "Computing TCP's retransmission timer." IETF RFC 6298, june 2011.

[3] P. Hurtig, A. Brunstrom, A. Petlund, and M. Welzl, "TCP and SCTP RTO restart." Internet Draft draft-ietf-tcpm-rtorestart, *work in progress*, Feb. 2013.

[4] A. Petlund, *Improving latency for interactive, thin-stream applications over reliable transport*. PhD thesis, Simula Research Laboratory / University of Oslo, Unipub, Kristian Ottosens hus, Pb. 33 Blindern, 0313 Oslo, December 2009.

