# Redundant Bundling in TCP to Reduce Perceived Latency for Time-Dependent Thin Streams

Kristian Evensen, Andreas Petlund, Carsten Griwodz, and Pål Halvorsen

***Abstract*— TCP and UDP are the dominant transport protocols today, with TCP being preferred because of the lack of fairness mechanisms in UDP. Some time-dependent applications with small bandwidth requirements, however, occationally suffer from unnecessarily high latency due to TCP retransmission mechanisms that are optimized for high-throughput streams. Examples of such thin-stream applications are Internet telephony and multiplayer games. For such interactive applications, the high delays can be devastating to the experience of the service. To address the latency issues, we explored application-transparent, sender-side modifications. We investigated whether it is possible to bundle unacknowledged data to preempt the experience of packet loss and improve the perceived latency in time-dependent systems. We implemented and tested this idea in Linux. Our results show that we can reduce the application latency by trading it against bandwidth.**

***Index Terms*— Retransmission latency, thin streams.**

## I. Introduction

TCP is designed to carry variable traffic from sources that demand a lot of bandwidth (greedy sources), and a lot of work has been done to increase throughput without losing control of congestion. It is assumed that a sender will always try to send data as quickly as the flow- and congestion control mechanisms permit. A challenge then is to support the many applications that consume very little bandwidth (*thin* data streams). These thin-stream applications may also have stringent latency requirements. Many time-dependent applications use TCP, due to the need for reliability and because of firewall policy issues. Important examples of interactive, thin-stream applications are multiplayer online games, audio conferences, sensor networks, virtual reality systems, augmented reality systems and stock exchange systems. As representative examples that have millions of users world wide, we examine online games and audio conferencing. In a packet trace from Funcom's massively multiplayer online role playing game (MMORPG) *Anarchy Online*, the average packet payload size was 93 bytes and the average interarrival time (IAT) 580 ms. For the online first-person shooter (FPS) game *Counter Strike*, the average payload size was 142 bytes, and the packet IAT was about 50 ms. Standard-compliant voice-over-IP (VoIP) telephony systems using the G.7xx audio compression formats have an average payload size of 160 bytes and a packet IAT of 20 ms [1]. In a sample packet trace of Skype, we found average payloads of 110 bytes and packet IATs of 24 ms.

These applications are also highly interactive and thus depend on the timely delivery of data. Different applications have different latency requirements. For example, the required latency is approximately 100 ms for FPS games, 500 ms for role playing games (RPGs) and 1000 ms for real-time strategy (RTS) games [2]. Latency in audio conferences must stay below 150-200 ms to achieve user satisfaction and below 400 ms to remain usable [3]. TCP variations that assume greedy sources do not accommodate thin-stream needs. It is unlikely that a separate transport protocol that addresses these needs will succeed any time soon, and it is also undesirable to make this distinction because applications may, at different times, be both greedy and interactive. It is, therefore, important to find solutions that react dynamically and use appropriate mechanisms according to the stream characteristics [4].

To address the low latency requirements of thin streams and the problems posed by the TCP retransmission mechanisms, we aim for application-transparent, sender-side modifications so that neither existing applications nor various multi-platform (OS) clients will need modifications. In particular, we propose a dynamically applicable bundling technique that includes unacknowledged data in the next packet if there is room. It is similar to several existing mechanisms. For example, TCP merges small user writes using Nagle's algorithm [5], and the stream control transmission protocol (SCTP) [6] has a multiplexing operation whereby more than one user message may be carried in the same packet. Thus, the number of sent packets is reduced.

Sending redundant data has been proposed both for latency reduction and error control. TCP and SCTP already bundle unacknowledged packets in some retransmission situations. For conferencing systems, the real-time transport protocol (RTP) packets may carry redundant audio data [7]. This allows a lost packet to be recovered earlier at the receiving side, because the data is also included in the next. However, to the best of our knowledge, no system performs such dynamic bundling for thin streams by packing unacknowledged data in a fully TCP-compatible manner before a loss is detected. We implemented a bundling mechanism and tested it in the Linux 2.6.19 kernel. The test results show that we can reduce the application latency experienced due to retransmissions by trading bandwidth for lower latency.

## II. Redundant Bundling

Supporting time-dependent thin streams over TCP is a challenge, because current loss recovery and congestion control mechanisms are aimed at greedy streams that try to utilize the full bandwidth as efficiently as possible. To achieve this with TCP's set of mechanisms, it is appropriate to use buffers that are as large as possible and to delay packet retransmission for as long as possible. Large buffers counters the throttling of throughput by flow control, allowing huge application layer delays. Delayed transmission strives to avoid retransmissions of out of order packets, promoting these delays actively as a

Manuscript received November 20, 2007. The associate editor coordinating the review of this letter and approving it for publication was J. Holliday.

The authors are affiliated with both Simula Research Laboratory, Norway and the Department of Informatics, University of Oslo, Norway (e-mail: {kristrev, apetlund, griff, paalh}@ifi.uio.no).

Digital Object Identifier 10.1109/LCOMM.2008.071957.

desired feature. Both methods come with an associated cost of increased delay, which means that interactive thin-stream applications experience greater delays in the application.

In our redundant data bundling (RDB) technique for TCP, we trade off space in a packet for reduced latency by reducing the number of required retransmissions. For important protocols such as gigabit Ethernet, this may be performed without exceeding the minimum slot size in our target scenarios, which means that RDB does not waste link bandwidth. It is implemented as a sender-side modification that copies (bundles) data from the previous unacknowledged packets in the send/retransmission-queue into the next packet. For example, assume that two packets of 100 bytes each should be sent. The first is sent using TCP sequence number X and a payload length of 100. Then, when the second packet is processed and the first is not yet acknowledged, the two packets are bundled, which results in a packet that has the same sequence number X but a payload length of 200. This scheme creates packets that could also appear with multiple lost acknowledgments (ACKs) and is fully TCP-compatible. An unmodified receiver should be able to handle streams with redundant data correctly.

The results of our experiments show that by sacrificing bandwidth (increasing packet size) without increasing the number of packets, time dependency can be supported better because a lost packet can be recovered when the next packet arrives. The approach has no effect at all on greedy sources that fill packets to their maximum transmission unit (MTU) size.

## III. Experiments

To test our bundling scheme, we modified the 2.6.19 (and, later, 2.6.22) Linux kernel and transmitted thin data streams over 1) an emulated network, using the *netem* emulator in Linux to create loss and delay, and 2) over the Internet from Oslo to the Universities in Amherst, UMASS, (USA) and Hong Kong. Each test in the emulated network setting had a fixed round-trip time (RTT) (between 50 and 250 ms) and a fixed packet-loss rate (between 0 and 10 %). The packet size was 120 bytes for all experiments and we used packet IATs between 25 and 500 ms. These parameters match our analysis of packet traces from the thin-stream applications that were described briefly in section I. For the real Internet test where the RTTs and loss rates varied, we used real game traffic from World of Warcraft (WoW). For comparison, we used TCP New Reno with selective acknowledgement (SACK) because it performed the best out of the existing TCP variants in the thin-stream scenario [4]. We also turned off Nagle's algorithm [5].

Fig. 1 shows the percentage of retransmissions experienced using the emulator when we vary the packet IAT using an RTT of 100 ms and a packet loss rate of 1% each way. The retransmission percentage for traditional TCP adheres to the packet loss rate and the retransmission scheme. At packet IATs below the minimum retransmission timeout (minRTO) value, the number of retransmissions depends on the number of lost packets. A lost ACK will not trigger a retransmission because the next one implicitly ACKs the previous packet before the timeout. However, when the packet IAT increases, packets are also retransmitted because of lost ACKs. This explains the

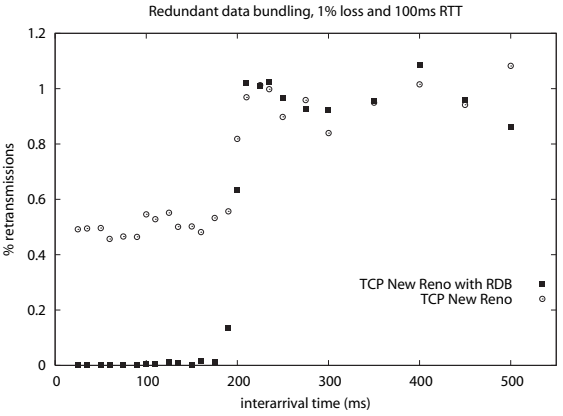


Fig. 1. Retransmissions versus packet IAT, loss = 1%, RTT = 100 ms.

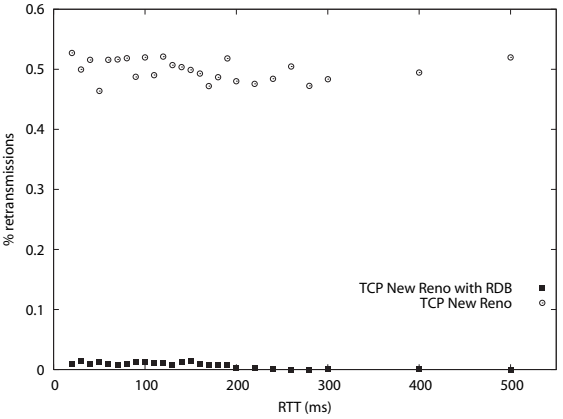


Fig. 2. Retransmissions versus RTT, loss = 1 %, IAT = (150±10) ms

jump from 0.5% to 1% for regular TCP in Fig. 1. On the other hand, Fig. 1 also shows that our bundling enhancement requires very few retransmissions, which again reduces the perceived latency. When there are (relatively) many packets that can be bundled, i.e., when IATs are low, no packets are retransmitted because a copy of the data arrives in one of the subsequent packets. However, the bundling gain lessens as the IATs increase. The reason for this is the reduced amount of data sent between the timeouts. Since less than one data segment is unacknowledged, there is no opportunity to bundle. Fig. 1 shows that the stream starts to retransmit at an IAT of roughly 200 ms (which is higher than in most of the applications we have analyzed). At this point, the performance of TCP with RDB quickly converges to that of the unmodified TCP because the minRTO is 200 ms. The results using other RTTs and loss rates show the same pattern.

Fig. 2 shows the percentage of retransmissions when the RTT increases. For these tests, we have varied the IAT randomly between 140 and 160 ms to reflect the traffic that we want to emulate. We can see that a very small number of retransmissions occurs, and as the RTT increases towards 500 ms, the tests stabilize at no retransmissions. The change from occasional retransmissions to none at all may be caused by changes in the retransmission timeout (RTO). In summary, the reduction in the number of retransmissions by timeout in TCP with RDB increases with the number of packets that are in transit and, thus, the ability to ACK data before a timeout occurs.

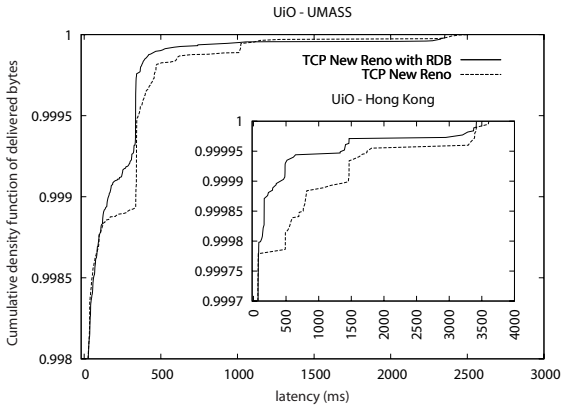


Fig. 3. Application latency differences running WoW 24 hours (19/11-2007).

Of course, the performance gain of the bundling technique depends on the packet loss rate. This claim is confirmed in our experiments. The number of timeout retransmissions in TCP increases as the loss rate increases. In our tested scenarios, TCP with RDB hardly experiences retransmissions. However, for (unrealistically) high loss rates (such as 10%), the number of packets sent using our technique increases because the packets do not have room for more bundling. Each retransmission will then be sent by the standard TCP retransmission scheme as a separate packet. Here, standard TCP bundles the unacknowledged data in one retransmission and thereby reduces the number of packets sent. Nevertheless, in our bundling scheme, the latency is still less than in TCP New Reno.

Finally, the latency at the application level, which influences the end-user satisfaction directly, is the most important measure. Fig. 3 shows the measured application latencies using real WoW game traffic over the Internet when RDB and New Reno are used back to back to have equal conditions. In general, the data recovery latency is reduced when using TCP with RDB compared to plain New Reno (both to Amherst and Hong Kong), but we also see some few large latencies for both mechanisms which are due to larger burst losses. However, there are vital differences in the latency where we see that RDB usually delivers lost data much faster, i.e., normally in the next packet, compared to Reno's retransmitted data.

Using TCP with RDB, we make a tradeoff between bandwidth and latency. In the tests for which the results are presented in Fig. 1 and 2, we used the maximum bundling limit, such that we try to bundle as long as there is available space in a packet (up to 1448 bytes for Ethernet). To determine whether we could achieve the same gain by reducing the bandwidth in terms of transmitted bytes, we have also reduced the bundling limit to approximately half the gigabit Ethernet slot size (240 bytes bundling maximum one data element). For a scenario with uncorrelated losses, we did not see any differences. The bundling limit can in many cases be lowered while retaining a reduction in the perceived latency compared to New Reno.

## IV. Dynamic Behavior

The experimental results show that we can reduce application latency for thin streams by introducing redundant bundling in TCP. RDB is efficient when packet sizes are small and IATs are less than the retransmission timeout value. In these cases, the number of packets remains the same; the overhead is in the number of transmitted bytes. Two issues that need to be addressed are when the technique should be applied and how the bundling limit should be chosen. In the current prototype, the bundling scheme is dynamically applied successfully (in terms of the results) when the data stream is thin, i.e. when the data stream has a low rate, small packets, and a low number of packets in transit. However, other factors should also be considered. For example, in a low loss rate scenario, a single loss can often be corrected by the next packet (as shown in Fig. 3). As the loss rate increases, the probability of a burst of losses increases. To compensate for this, and to increase the probability that the data will arrive before a retransmission occurs, it may be necessary to increase the bundling limit.

RDB performs best when it can preempt the retransmission timer with an ACK from a bundled packet. However, there is an intermediate stage in which spurious retransmissions due to timeouts may be experienced, but in which the latency of the application layer can still be improved. This happens when the IAT exceeds the minRTO value. When loss occurs, a timeout will be triggered before the ACK for the bundled packet is received. When the IAT exceeds minRTO + RTT, there is no gain, and the mechanism should be turned off completely. Thus, the RDB mechanism is turned on and off and the bundling limit should be set dynamically on a per connection basis according to the packet size, packet IAT, measured RTT, and estimated loss rate.

## V. Conclusions and Further Work

Many current applications are highly interactive and depend on timely delivery of data. For applications characterized by thin streams, TCP can not guarantee delivery times, and newer variations worsen the problem because they rely on a greedy source. This being so, we implemented a redundant data bundling scheme to reduce the retransmission latency for such scenarios. Our experimental results, both in a lab setting and in real Internet tests, show that we can trade off some bandwidth to greatly lower the perceived latency at the application level in these scenarios.

## References

[1] M. Hassan and D. F. Alekseevich, "Variable packet size of IP packets for VoIP transmission," in *Proc. IASTED International Conference conference on Internet and Multimedia Systems and Applications (IMSA)*. ACTA Press, 2006, pp. 136–141.

[2] M. Claypool and K. Claypool, "Latency and player actions in online games," *Commun. ACM*, vol. 49, no. 11, pp. 40–45, Nov. 2005.

[3] International Telecommunication Union (ITU-T), "One-way Transmission Time, ITU-T Recommendation G.114," 2003.

[4] C. Griwodz and P. Halvorsen, "The fun of using TCP for an MMORPG," in *Proc. International Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV)*.ACM Press, May 2006.

[5] J. Nagle, "Congestion control in IP/TCP internetworks," RFC 896, Jan. 1984. [Online]. Available: http://www.ietf.org/rfc/rfc896.txt

[6] R. Stewart, Q. Xie, K. Morneault, C. Sharp, H. Schwarzbauer, T. Taylor, I. Rytina, M. Kalla, L. Zhang, and V. Paxson, "Stream Control Transmission Protocol," RFC 2960 (Proposed Standard), Oct. 2000, updated by RFC 3309. [Online]. Available: http://www.ietf.org/rfc/rfc2960.txt

[7] C. Perkins, I. Kouvelas, O. Hodson, V. Hardman, M. Handley, J. Bolot, A. Vega-Garcia, and S. Fosse-Parisis, "RTP payload for redundant audio data," RFC 2198 (proposed standard), Sept. 1997. [Online]. Available: http://www.ietf.org/rfc/rfc2198.txt