[simula . research laboratory]

UNIVERSITY OF OSLO

TCP Enhancements for Interactive Thin-Stream Applications

Andreas Petlund, Kristian Evensen, Carsten Griwodz and Pål Halvorsen

A wide range of Internet-based applications that use reliable transport protocols displays what we call thin-stream properties. This means that the application sends data at such a low rate that the retransmission mechanisms of the transport protocol are not fully effective. In time-dependent scenarios, where the user experience depends on the data delivery latency, packet loss can be devastating for the user experience.

In order to reduce the perceived latency when packets are lost, we have implemented modifications to the TCP retransmission mechanisms in the Linux kernel. The changes are only active when thin-stream properties are detected, thus not affecting TCP behaviour when the stream is "thick".

THIN-STREAM MECHANISMS

If the kernel detects a thin stream, by defining thresholds for packet size and packets in flight, we trade a small amount of bandwidth for latency reduction and apply:

DISABLING OF EXPONENTIAL BACKOFF (DEB):

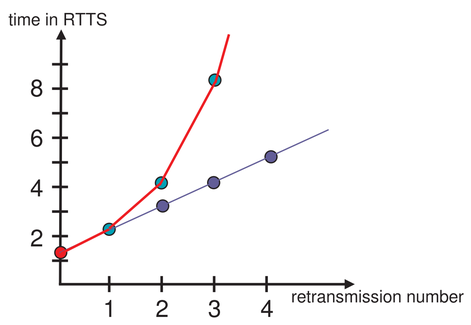
To prevent an exponential increase in retransmission delay for a repeatedly lost packet, we disable the exponential factor.

FASTER FAST RETRANSMIT (FFR):

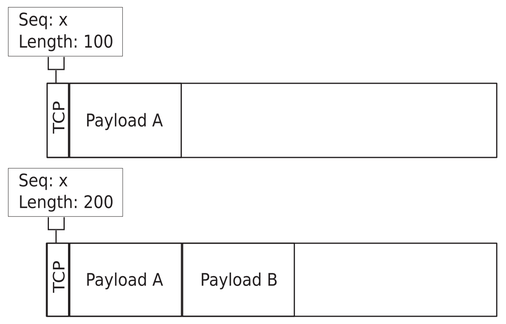
Instead of waiting for 3 duplicate acknowledgments before sending a fast retransmission, we retransmit after receiving only one.

REDUNDANT DATA BUNDLING (RDB):

We copy (bundle) data from the unacknowledged packets in the send buffer into the next packet if space is available.



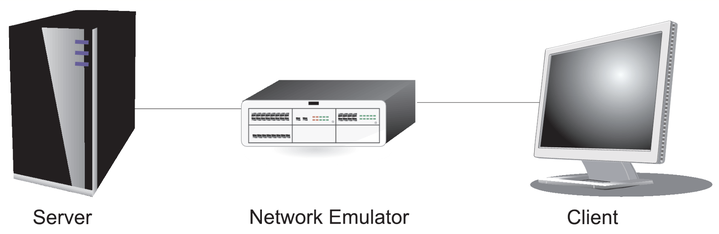
Exponential backoff



Bundling of unacknowledged data

	Small Packets	Large Packets
High IA	RDB / REB / FFR	REB / FFR
Low IA	RDB	Thick

Applicability of thin-stream mechanisms



Demonstration setup

DEMONSTRATION CONFIGURATION

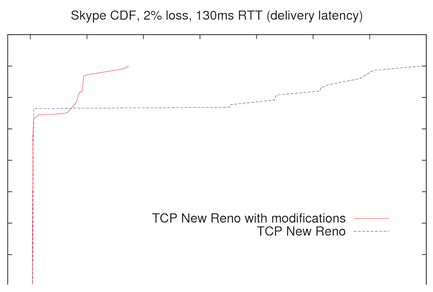
To emulate network loss and delay, we send data from a server to a client through a network emulator. The server uses the modified 2.6.23.8 Linux kernel. The thin-stream modifications can be enabled on the server either on a per-stream basis using IOCTL or globally using SYSCTL.

RESULT OF PACKET ANALYSIS

We have analysed packet traces made while performing thin-stream experiments on different applications. The first CDF (CDF - Transport layer) shows delivery latency statistics for Skype sessions using TCP fallback. We can see that TCP with thin-stream modifications delivers lost payloads much earlier than standard TCP.

When applied to the application layer, we take into account that received segments have to wait for the lost ones before they can be delivered. The second CDF (CDF - Application layer) shows the delivery times for the application layer, and it is clear that when using TCP, this will have an impact on performance when loss occurs.

We also see that when the modifications are active, transport layer latency and application layer latency are nearly identical, indicating that the delay imposed by the in-order requirement is minimized.



CDF - Transport layer

CDF - Application layer

POSITION UPDATES

In order to demonstrate the effects of TCP retransmission mechanisms on a thin stream (like position updates in a game), we have created a proof-of-concept application. The server displays a dot moving in a circle (in order to have a predictable trajectory). When a client connects, the server will send position updates with regular intervals. A network emulator positioned between the server and the client creates loss and delay.

The client, once connected, will move the dot when a new position update is received. From the client user interface, several parameters can be changed in order to observe the effect on how position updates are received. Interarrival time and packet size can be changed, and each of the thin-stream mechanisms described above can be turned on and off.

In order to test different client settings with different loss rates and delays, we also have a GUI for configuring the network emulator.

SKYPE

As shown in the above graphs, Skype has a transmission pattern that is typical for thin streams. When we performed a survey where people were asked to rate audio clips sent over a Skype connection with or without the thin-stream modifications, a large majority deemed the quality better with the modifications turned on.

In the demonstration, you will be able to to listen to audio through a Skype connection with emulated loss and delay. You may then evaluate the quality with or without the modifications under different network conditions.

BZFLAG

BZFlag, a popular first person shooter, multiplayer game, generates typical thin streams. In the case of lost or delayed packets, BZFlag predicts the movement of player(s) until the position update(s) arrive. If the prediction is wrong, the tank(s) will jump into the correct position when the update arrives. Also, as the links get poorer, the prediction has to cover a longer time period.

We demonstrate that by using our modifications, lost data is delivered faster to the receivers. Thus, the difference between the perceived and actual position will be reduced, leading to a better user experience.