DTL: Dynamic Transport Library for Peer-To-Peer Applications

Bernardo Gonzalez Riede Xin Ren

I. Motivation

Although bandwith has increased vastly since the introduction of the internet, so have the file sizes of media, e.g. pictures and videos. Asynchronous internet connections, i.e. different upload and download speeds, are very common. Therefore upload speeds are still a bottleneck in private households stymieing adoption of p2p application. In the beginning of BitTorrent a reputation as bandwith hog was gained which had to be overcome. For this end new transport protocols have been proposed. Their purposes range from using unused bandwith in the case of LEDBAT to more efficient use of the available bandwith coupled with prioritization like MulTFRC.

Moreover, an application might require flexibility in the form of inter- or intra-protocol prioritization and changes during run-time. The widespread usage of NAT makes it a imperative consideration when designing a new protocol.

II. Contributions

The primary contribution is the design and implementation of a new transport protocol called *Dynamic Trasport Library*, abbreviated DTL. The implementation is done in JAVA using the NIO package for platform portability. It is a combination of a modified LEDBAT and MulTCP, covering the following target requirements:

- Support of existing congestion control.
- Control about bandwith behaviour towards other applications in the form of *Inter-protocol prioritization*.
- Based on UDP for more efficient NAT traversal.
- Implementation in user space.
- Priority changes at run-time without dropping existing connections.

A contribution of smaller magnitude is the tweaked LEDBAT algorithm used in the final implementation. The additive-decrease of the reference LEDBAT might lead to an unfair system. The modified equation for dealing with late-comer connections improves the stability. This confirmes findings of another study [1].

III. Solution

The DTL is implemented using Java NIO and provides reliability, congestion control and flow control over UDP in the same way TCP does by appending an application level TCP-like header, carrying the receiver's advertised window, the sequence number and the acknowledge number, to each datagram. A *priority* parameter, intended as a positive floating point number is provided to applications for each socket for controlling the aggressiveness of the flow.

When the value of *priority* is zero, *polite* mode is activated and LEDBAT algorithm is used, on top of which, a simple AIMD algorithm is implemented to overcome the late-comer advantage referring to the work of Chiu et al##########. As a result, if the estimated queuing delay exceeds the target delay, the *cwnd* shrinks by a factor smaller than 1.

When the value of *priority* is greater than zero, *variable* mode is activated and the congestion control switches to the MulTCP algorithm, implemented on top of the normal TCP SACK algorithm using the *priority* value as number of virtual flows N which one transfer must virtualize. The smooth slow start algrithm introduced in ########## is implemented to avoid sending large bursts of packets if N is too large, causing packet losses. Moveover, the MulTCP2 algorithm modification proposed by Nabeshima et al ########## is implemented to make it possible, in case of packet loss, to achieve a better virtualized behavior than the original MulTCP specification, considered to be too aggressive.

IV. Strong Points

1) References are well listed and noted.
2) The differences between the paper and other related work are clearly stated with descriptions and comparison table.
3)

V. Weak Points

1) There are some grammar faults in the paper. For example, "The serious shortcoming is the that the best know techniques are scattered around in different libraries".
2) The are some format issues in the paper. For example, in Introduction, Run-Time Dynamic Prioritization is introduced in the same paragraph as the description for Portability.
3) Table 1 is confusing in the way that X means Yes and blank means No.
4) There needs to be a consistency about the notation of the fixed target delay, either or TARGET or tau.

VI. Questions

A. What is TCP slow start? How do Internet applications get around the TCP slow start?

orsnto

REFERENCES

[1] Giovanna Carofiglio, Luca Muscariello, Dario Rossi, and Silvio Valenti. The quest for LEDBAT fairness. *CoRR*, abs/1006.3018, 2010.