Template for Alternative Transport Protocols

Title: CADPC / PTP

Contacts (Include institutions)

Michael Welzl, University of Innsbruck
email: michael.welzl@uibk.ac.at web: http://come.to/michael.welzl or
http://informatik.uibk.ac.at/users/c70370/

URLs / RFCs / Papers

The PTP website is at http://fullspeed.to/ptp or
http://informatik.uibk.ac.at/users/c70370/research/projects/ptp/

CADPC / PTP is mainly the result of a Ph.D. thesis, which is finished and currently in print; a .pdf file is available upon request. A published paper containing some details about CADPC is:

Welzl, M.: "Traceable Congestion Control", ICQT 2002 (International Workshop on Internet Charging and QoS Technologies), Zürich, Switzerland, 16-18 October 2002. Springer LNCS 2511, available from the PTP website.

Principle / Description of Operation

PTP, the “Performance Transparency Protocol”, is a lightweight signaling protocol that queries routers along a path for performance related information; when used for congestion control purposes, this information consists of:

- the router address
- the MIB2-ifSpeed object (nominal link bandwidth)
- the MIB2-ifOutOctets object (traffic counters)
- a timestamp

At the receiver, it is possible to calculate the bandwidth that was available at the bottleneck during a certain period from two consecutive packets carrying this information for all routers along the path. This operation resembles ATM ABR Explicit Rate Feedback, but work in routers is minimized, all calculations are moved need to network endpoints.

CADPC, “Congestion Avoidance with Distributed Proportional Control”, is a congestion control scheme that is solely based on PTP feedback. Since it does not rely on packet loss, it works seamlessly over wireless links. It is slowly responsive in that it utilizes a small amount of feedback, but it shows quick convergence.

Among its outstanding features / properties are:

- good scalability

- fully distributed convergence to max-min-fairness irrespective of RTTs
- designed for heterogeneous links and links with a large bandwidth X delay product
- since it only relies on PTP, it is easily traceable
- simple underlying control law (logistic growth) which is known to be stable
- very smooth rate

Supported operation mode:

memory to memory (general transport)

Authentication: no

Implementations / API: Code is available from the website; future releases will be available from this site, too. Currently, there is:

- A PTP end system and router implementation for Linux
- PTP code for the ns-2 simulator

CADPC was only implemented for the ns-2 simulator so far and will be made available via the PTP website soon.

Congestion Control Algorithms: CADPC is the congestion control algorithm.

Fairness: max-min fairness, could probably be extended to support other forms of fairness (such as proportional fairness) too.

TCP Friendly: No.

Predictable Performance Model: In a network with n users, CADPC converges to 1/(n+1) for each user; thus, the total traffic converges to n/(n+1), which quickly converges to 1 with a growing number of users (these calculations are normalized with the bottleneck capacity). With a VERY small number of users (say, 2 or 3), CADPC is inefficient.

Results:

In simulations, CADPC outperformed several TCP variants and TCP-friendly mechanisms in a large variety of scenarios and in several aspects; in particular, it showed greater throughput than its competitors with close to zero loss. Please see the "traceable congestion control" paper for more details.

Target Usage Scenario:

This protocol is intended for bulk data transport of large data sets. It will work well in scenarios with highly asymmetric links, noisy links and links with a large bandwidth X

delay product. It will have trouble if this product is very small. In its present form, it must be isolated from other traffic and will not work well in the presence of short web-like flows or long-term TCP flows. A closer look at CADPC in isolation (usage to control traffic management, or isolated via QoS mechanisms – e.g., by using it within a DiffServ class) is currently under research.