A Congestion Control Architecture for Bulk Data Transfer

Mark Handley,

USC/ISI

The Multicast Congestion Control Problem

- **Multicast traffic needs to compete with TCP traffic to obtain a fair share of the bandwidth**
- **"Fair share" is ill-defined, but a *conservative* definition is that the multicast traffic sent to any receiver should be no greater than the bandwidth a TCP connection would achieve to that receiver.**
- **Problem: how to determine an appropriate rate?**

Loss rate against bandwidth

If you measure loss rate against transmitted bandwidth for realistic data rates, there is usually no good correlation.

As bandwidth tends to zero, loss rate is still positive.

What does this mean?

A flow cannot determine the effect of its own traffic on the network because of the "noise" caused by the many many TCP connections it is competing with.

Three solutions seem possible:

- **devise a scheme that can deal with time-averaged loss data and still achieve a fair share.**
- **devise a scheme that divides the problem up so that it can run a TCP-like algorithm on TCP timescales**
- **devise a scheme that utilises information from the routers to determine an appropriate rate**

Working on TCP timescales

- **To adjust bandwidth on TCP timescales with an RM scheme requires *immediate* feedback of loss reports.**
- **To prevent NACK implosions requires *delaying NACKs* to give other sites time to respond.**
- **How to reconcile this contradiction?**

Warning! Warning! Warning! Half Baked Ideas coming up....

- **We have network measurements that imply this is a valid approach.**
- **We do not have sufficient simulation or real-world tests that show this all works!**

Overview

Stage 1: ***representative "election"***

Stage 2: ***subgroup formation***

Stage 3: ***congestion controlled data flow***

Startup Mode

Everyone starts off in one group.

Data is sent slowly, and is grouped into small (eg 5-10 packet) blocks.

Use *SRM-style random timers* weighted by number of losses to determine which sites send NACKs

A NACK indicates the *number* of packets lost from the block and a mask of which packets they were as a loss fingerprint.

Sender transmits *parities* (cf Nonnenmacher et al) to repair the loss.

This means that only sites with significant loss send NACKs at the start. Other sites don't need to, and so only significant bottlenecks are emphasized.

Representative Election

A site that sends a NACK and then sees a duplicate NACK with the same or greater number of losses increases its base value for calculating random timers.

A site that sends a NACK and sees no duplicate NACK, or duplicate NACKs with fewer losses decreases the value for calculating random timers.

- Exponential increase, linear decrease seems to have the right selection properties for rapid election.

Other sites send session messages with high rate if they match the loss masks of a "representative", and low rate if they do not.

- Eg 60% of a fixed session message bandwidth budget is reserved for sites matching representatives.

Subgroup formation

If a site is sending frequent NACKS it is a representative, and if there is demand from other sites (via session messages), it can inititate subgroup formation.

Firstly it performs an expanding ring search requesting a site that is prepared to be a relay and that is seeing significantly lower loss rates.

When it finds one, it obtains a new multicast address and sends session messages to indicate that it is forming a new group.

The relay site then starts relaying data to the new group, and the new subgroup members leave the original group.

Congestion Control in a Subgroup

Now, if all went well, there is only one significant bottleneck between the relay and the subgroup members, and almost all loss seen in the subgroup is now correlated.

The relay is buffering data from the main group and playing it out slower to the subgroup.

The representative can now ACK/NACK immediately knowing that a duplicate is very unlikely, and the relay can perform a rate-based variant of TCP congestion control on TCP timescales.

- should the representative die, the adaptive timers ensure a new one takes over rapidly.

Congestion Control in a Subgroup

Sometimes however, the topology will be different in the subgroup, and correlated loss may not occur. Sites uncorrelated with the representative should then return to the original group.

Optimisations and Details

- **This scheme does a serial search for representatives. It can be parallelised by allowing multiple NACKs to not cause backoff so long as their fingerprint is "significantly different".**
- **Expanding ring searches can be performed on a special control group so subgroups can be chained.**
- **A relay may receive all its data and wish to leave the group. Thus the original sender must be prepared to receive subgroups back and to transmit to them data that has already been sent to the sites with good connectivity.**

Summary

- **Self-elect receiver representatives for each significant bottleneck.**
- **Sites downstream of each bottleneck form a new subgroup fed by a site upstream of the bottleneck.**
- **Each subgroup is fed at its own appropriate rate using a variant of the TCP congestion control algorithm.**

We need a lot more work before we can be sure of some of the details though!

Limitations and Requirements

- **Needs sufficiently long data flow to be able to self-organise.**
- **Needs bounded data flow to prevent requiring infinite buffering at sender and relays.**
- **Makes some assumptions about loss distributions that appear to be justified but may not always be.**
- **May use a lot of groups unless a subtree multicast mechanism is available or it gets lucky with admin scoping.**
- **Needs TTL mechanisms (or similar) to perform expanding ring search to find a relay.**