AReliableMulticastFramework
forLight-weightSessions
andApplicationLevelFraming
forLight-weightSessions
AReliableMulticastFramework
Errata:
Figure 3 in the proceedings contains the wrong figures.
• Graphs all show Delay/RTT that is 0.5 too big (e.g., 3.0 should be 2.5).
• Corrected paper and tech report (longer version) available at:
Why Multicast?

- Efficiency (only one copy of data per link).
- Group queries (can request data without knowing who has it).
- Independent of number of receivers.
The world used to be so simple...
but multicast changes the rules.

Model of communication as "conversation" breaks down.

... Sender can't keep state for unknown number of receivers.

Algorithms based on estimating path properties don't generalize to trees.

RTT, congestion window (RTT, congestion window) don't generalize to trees.

Sender can't keep "state" for unknown number of receivers.

but multicast changes the rules.
Most work on reliable multicast attempts to leverage unicast transport models so unicast transport models will work. (Forming ring or electing leader require group-wide agreement which is expensive and problematic when membership changes frequently.)

These approaches have serious scaling problems.

E.g., Chang & Maxemchuk (and derivatives like RMP) form members into token rings; MTP elects a central controller.
At SIGCOMM 90, Clark and Tennenhouse proposed a new communication model, Application-Level Framing (ALF), that easily generalizes to multicast.

Some key parts are to let applications manage the specific namespace for data (e.g., filename & sector offset) and use an application-specific namespace for data (e.g., video frames, disk blocks) and in "application data units" (e.g., application, communication, speak in application data units).
Since 1991, we have been trying to elaborate the ALF model. One piece we've developed is a scalable, reliable multicast framework, SRM. It is fully decentralized (no central controller) and handles arbitrarily large groups. A complete protocol using the framework has been implemented in the LBL whiteboard tool, wb, and tested on the MBone. WB has been in widespread use since 1993 for conferences with anywhere from two to several thousand participants.
rate limited to 3% of session bandwidth. Each session has a bandwidth limit. Anyone can send if they have data and aggregate traffic is under limit.

All members send low-rate reports that contain their current state. Report sends randomized and send low-rate reports that contain their current state, randomized and rate limited to 3% of session bandwidth. Each session has a bandwidth limit. Any member can send it if they have data and aggregate traffic is under limit.

All traffic is multicast.

SRM Reliability Machinery
Receivers learn they're missing data either from someone's report or a hole in sequence space. They multicast a 'repair request' to ask for missing data. Anyone that has data can reply, not just the original source of data.

SRM Reliability Machinery (cont.)
Ack Implosions (cont.)

synchronized 'repair' requests

Fmz-SRM-12
Avoiding ack implosions

Every node estimates distance (in time) from every other node. (Info for this carried in session reports.)

Nodes use randomized function of distance to decide when they should request repair (or reply to a repair request).

Receipt of request or reply causes node to suppress its own attempt.

•
When $j$'s report arrives at $i$, distance from $j$ is calculated as:

$$\frac{R_i}{2} - \frac{S_j}{2}.$$
Worst case topology (star) and Randomization
Request and repair timers set to random number in intervals.

$\log_{10} (\text{members}) = \frac{2}{p} = \frac{1}{p}$

$z = cz = c_1$

Simpliest SRM uses fixed values for constants:

$\lfloor \frac{2}{p} + \frac{1}{p}, \frac{1}{p} \rfloor$

$\lfloor c_1, c_2 \rfloor$
(random trees; all nodes members)

Session Size vs. Number of Requests

Session Size vs. Number of Repairs

Session Size vs. Delay/RTT
(1000 node, bounded degree trees)

- Session Size vs. Number of Requests
- Session Size vs. Number of Repairs
- Session Size vs. Delay/RTT
Random interval constants (weakly) sensitive to both topology and location of loss. Can get better repair response, fewer duplicates, or both, if $c$ and $p$ are dynamically adjusted.
(1000 node, bounded degree trees, adaptive algorithm)

---

**Session Size**

- **Number of Requests**
  - 20
  - 40
  - 60
  - 80
  - 100

- **Number of Repairs**
  - 0
  - 5
  - 10
  - 15
  - 20

- **Delay/RTT**
  - 0
  - 1
  - 2
  - 3
  - 4
  - 5
  - 6

---

(1000node,bounded degree trees,adaptive algorithm)
Other SRM Applications

- Self-configuring cache hierarchies for e.g., Web or FTP data.
- Almost any large-scale data distribution — BGP routes, DNS zone transfers, Usenet news, stock quotes, etc.
- Almost any large-scale data distribution — BGP routes, DNS zone transfers, Usenet news, stock quotes, etc.
Some Open Questions

Local repair to avoid crying baby problem.

Other forms of bandwidth adaptation / congestion control.