6.896 Lecture 16: Reliable Multicast I

Hari Balakrishnan

October 26, 1998

1 Outline

- **Readings: LP96, FJM+95**
- The problem
- Approaches
- RMTP
- SRM

2 The problem

- Problem: transmit data reliably from one or more sources to multiple receivers over an unreliable multicast delivery channel (e.g., IP multicast) in an efficient manner.
- Two issues:
    1. What is *reliable*?
    2. What is *efficient*?
- *Reliable:* The unicast (TCP) definition is easy. But multicast semantics are harder promarily because "ordering" has many possible meanings.
- Birman et al.'s ISIS work pioneered many definitions (in addition to many other contributions):
    - Totally ordered: all messages delivered to all receivers in identical order.
    - Causally ordered: all messages delivered according to Lamport's "happened before" relationship (see Lamport's 77 paper). Here, "concurrent" messages can be delivered in arbitrary order to all receivers.

- Concurrently "ordered": don't care about relative ordering, just focus on reliabile delivery.
- The famous CATOCS debate: Cheriton/Skeen (SOSP '93) vs. Birman's ISIS scheme.

- Most work today in the context of IP multicast doesn't worry too much about the above and focuses on *efficient.*

- *Efficiency:* Key problems

  1. *Scalability*. Q: What does this mean? A: avoid message implosion.
  2. *Congestion management:* Problem hard because of heterogeneity.
  3. *Generality:* Is there a general protocol (toolkit) possible?

- Area of very active research. At least 16(!) different protocols proposed for different contexts, work on congestion control and some work on developing general toolkits. RMRG research group of the IRTF (linked from class Web page).

3 Approaches

- This is incomplete and is based on a paper by Levine et al.

- Token ring schemes: form a logical ring using some formation protocol and pass token around; piggyback ACKs on token; easy to provide total/causal ordering and release messages.

  Examples: TRP, RMP, Totem

  Q: disadvantages? A: scaling; failures, token loss,...

- Sender-based: each receiver ACKs every packet; doesn't scale well. Examples: TCP generalization, XTP.

- Receiver-based: Each receiver NAKs missing data; randomized damping and suppression; session messages; persistent data model.

  Q: Disadvantages. A: doesn't scale well to large numbers of uncorrelated loss neighborhoods; persistent data model.

- Tree-based: Arrange receivers into a logical tree; send ACKs up tree; sender sees few ACKs if tree is built well.

  Q: disadvantages: how to build tree; if receivers come and go, what are ACK semantics?

4 RMPT: Reliable Multicast Transport Protocol

- Comprehensive design and implementation of a large-scale reliable multicast scheme using a hierarchy of *designated receivers (DR)* into an ACK tree. Being used in AT&T's billing network.
- Overview:
  - Sender multicasts data using simple window flow control.
  - Send ACKs up logical tree to be processed by ACK processors (DRs and sender).
  - DRs cache and retransmit "locally" to subtree. Only pass one ACK up if successful.
  - Upon network partitions, invoke application-defined recovery procedure.
  - Q: how are DRs configured? Statically. (Dynamic: future work).
  - End-receivers discover DRs automatically using separate protocol.
  - Meant for large bulk transfers.
  - Opposite of ALF model; here, everyone gets everything.
- What really happens:
  - All packets have sequence numbers.
  - Receiver periodically sends (L, V) ACK, where L is cumulative ACK and V is SACK bit vector.
  - ACKs sent to DR. DR waits some time and queues retransmission; if number of lossy hosts larger then threshold, multicast retransmission, else machnine-gun unicast to each lossy host. Q: How long to wait for ACK at DR?
  - Late joiners can recover up to what DR has.
  - RMTP flow control like TCP's, goes at rate of slowest receiver.

5 SRM: Scalable Reliable Multicast

- ALM model: application in control of reliability and ordering.
- Motivated by wb.
- Receiver-based protocol.
- Contributions:
  1. ALF
  2. Distributed and randomized approach to solving message implosion.

3. *Multicast everything!* Highly robust, session actually very hard to kill!
4. Real app (wb) on the MBone.

- ALF: key problem is *naming data.* Explain in detail using wb and webcast example. No linear sequence space of data.
- Two types of losses: "regular" and "tail".
- How to recover from regular losses? Send RREQ with missing name, if app wants it.
- Timer backoff $= (C_1 + C_2 r)d$, where r is $U(0, 1)$.
- Explain motivation and scaling behavior for simple topologies asking questions.
- Same algorithm for RRESP. *Anyone in session can respond.* Data is *persistent.*
- Tail losses more frequent than in protocol like TCP. Why?
- Recovered using periodic session announcements. Soft-state announcements.
- Elegant protocol, real app, initial solution to data naming, nice scaling in many cases.
- But severe scaling problem when you have a number of uncorrelated loss neighborhoods.
- Next class: two approaches to better scaling: local recovery and FEC.