6.896 Lecture 17: Reliable Multicast II

Hari Balakrishnan

November 9, 1998

1 Outline

- **Readings: PPV98, BLMR98, PGM98**
- Towards better scaling
- Replier-based scheme
- PGM from cisco
- Digital fountain

2 How to achieve better scaling in reliable multicast?

- First, what are we trying to scale?
- In the last class, I said that feedback implosion was the key problem and the key to better scaling is greatly reducing implosion.
- Recap: how does SRM handle this? And what are its drawbacks?
- The following are the key metrics along which to evaluate a reliable multicast scheme:
  1. Request implosion
  2. Duplicate replies
  3. Recovery latency
  4. Recovery isolation (exposure)
  5. Adaptability o dynamic membership changes
- Two (orthogonal) approaches to better scalability:

1. FEC
2. Local recovery

- Numerous proposals have been published on how these can be used.
- Consider FEC
  1. n packet payload of which k packets are redundant. If A loses one of these and B another of these, both can be recovered using the same retransmission.
  2. Can use FEC (e.g., Reed-Solomon) to layer data across multiple layers. Then, multicast congestion control (and loss recovery too) can be performed based on ideas similar to RLM for video. *Erasure codes* are well-suited for this.
  3. *Multiple multicast groups (MMG)* are not strictly FEC... But can use them to minimize request implosion and exposure.
  4. What we'll study is recent work on *digital fountains.*
- Consider local recovery: again, many proposals here.
  1. Elect local representatives (e.g., SRM proposal, STORM, etc.).
  2. Tree-based approaches (e.g., RMTP, TMTP).
  3. Replier-based scheme.
  4. Use active networks...
  5. Replier-based scheme from wustl.
  6. PGM from cisco.
  7. Subcast and randomcast ("search party").
- Key questions to remember:
  1. How to use FEC schemes to improve scalability?
  2. What's the "right" (i.e., minimalist yet useful and general) set of forwarding services routers should export to enable better reliable multicast?

3 Replier-based scheme

- Use minimal router support to do "local" recovery.
- Observation: Any time a loss occurs, it affects a *subtree* of the multicast distribution tree. Q: why?

- So, in the ideal case, the router just below the loss link asks the one just above for the missing data, and then multicasts it to the loss subtree.
- Problem with this ideal view of the world is that you end up putting lots of cruft into the routers.
- Challenge: what's the minimal set of things routers can do to approximate this ideal situation?
- Note: this is still an open question. Papadopoulos' replier-based scheme is one approach. There are others.
- Overview
  1. Each router picks a replier link from contenders. Receivers willing to be repliers periodically multicast control messages saying so.
  2. Repair request arriving on any other link is sent down replier link (this is the *turning point*. Request on replier link sent up the tree.
  3. Responses are unicast to requesting turning point. From there, they are *subcast* to all receivers behind the requesting link.
- There are some simple details that need to be worked out!
  1. How to add a replier to each router.
  2. How to send/forward requests?
  3. How to send replies?
  4. How to subcast replies?
  5. How to handle changing memberships and replier failures?
  6. How to handle source-based trees?
  7. And how to analyze resulting protocol?!

3.1 Details

- Replier selection: each router selects one of its downward links to be the replier link. I.e., replier chosen from one of its children subtrees. Router only keeps track of link, not particular receiver who may be the replier. Multiple choices dealt with using a metric (loss rate, distance,...).
- Request forwarding: A request is forwarded onto the replier link, unless it came from the replier link, in which case it is forwarded upward.

  Q: So how many requests "escape" subtree for a given loss? A: at most one! This is a key point to remember about this scheme.

- *Turning point:* If you get a request from a non-replier link, turn it toward replier link. This router is the turning point for the request.

  The turning point puts its address and requesing link id into the turned request. Routers between turning point and replier forward packet unchanged.

- Sending replies: If you're a replier and have the requested data, you can reply. But you want to make sure that it only gets to the people who've requested it, not everybody. How do you do this?

  Unicast it to turning point. That's why we put its address in turned request. Turning point gets reply, decapsulates it and gets inner multicast header. It directs this down loss subtree. How does it know which subtree? Well, that's why it had the corresponding requesting link id in turned request!

- *Dynamic memberships:* Routers only have soft state. A leaving replier sends a *replier gone* message. Replier link information also times out.

  They claim that this scheme handles dynamic memberships very well. I have my doubts. There are several examples where it wouldn't do that well. On the other hand, considering their elegant, minimalist approach with as little router state as possible, I think they do a good job.

- DVMRP has source-based trees, so to make sure that the subcast propagates down the correct tree, they spoof the address of the source!

- *Analysis:* not as complete as it should be. I think that an analysis of this protocol (and its improvements) is tractable in more generality than they did in this paper.

3.2 Opinions

- I thought it was a good paper describing an elegant scheme to improve RM scaling. There are some cases when the scheme doesn't work that well, but overall it is a workable solution.

- I liked their emphasis on minimal router state and support, focusing on what's necessary rather than putting everything into routers.

- The analysis needs more work. So does the part handling dynamic memberships, despite the soft-state arguments.

4 Announcements

- HW due tomorrow. Hard copy in my office.

- Exams graded. Will hand out in last 10 mins; solns to be posted soon.

- No class wednesday. Meet again next monday to talk about programmable networks.

- Start thinking actively about projects!

5 Digital Fountain

- Application: new release of widely used software. Reliable bulk distribution (tens/hundreds of MB). Many, many, many clients. Widespread heterogeneity. Asynchronous arrivals. Lossy network.
- A digital fountain is a data carousel that is an ideal solution to this problem. Draw picture of how this works ideally.
- Key idea: use an erasure code to encode data and continually transmit it. Receiving *any* k distinct packets allows receiver to reconstruct entire message.
- FEC has received a bad rap in the congestion control community. Important to realize that FEC isn't inherently bad—FEC only tells you *what* to send, not *what rate* to send it at.
- Approximating a digital fountain: encode stream of k packets. Receiving $(1 + \epsilon)k$ packets allows message to be reconstructed.
- Focus analysis on decoding time and on decoding inefficiency (how many extra packets to reconstruct message?).
- Two important classes of erasure (or almost-erasure) codes: Reed-Solomon and Tornado.
- *Reed-Solomon:*
  1. Based on dense system of linear equations.
  2. Decoding complexity quadratic in k.
  3. But has optimal decoding efficiency of 1.
- *Tornado*
  1. Based on sparse system of equations.
  2. Fast decoding complexity (linear in k).
  3. But decoding efficiency is sub-optimal.
- Particular Tornado used in paper called *Tornado Z code.* Think of message packets as left side nodes in bipartite graph and actual transmitted packets as right side nodes. Edges between two sides. Right node is XOR of neighbor nodes on left side.
- Q: Why is it called a tornado code? A: while decoding, nothing happens for a while, then all of a sudden a tornado of decoded messages occurs!

- Far superior to R-S in terms of time to decode. Allows very large block sizes.
- In implementation, use Tornado codes and maximum ineffiecieny on the order of 5%. Use MMG-based congestion control by layering data.
- Tornado codes may have applications in areas other than reliable multicast too...