CS423: Lecture 26, SRM Multicast

George Varghese

December 5, 1996
Just flooding
Reverse Path Forwarding
Deering’s Reverse Path Broadcasting

[Diagram of a network with nodes A, B, C, D and edges x, y, z with labels 1, 2, 15 and 15, 1, 1, 1, 2, 1, 1, 1, 1, z, x, y, z]
Reverse Path Broadcasting with Pruning
Scalable Reliable Multicast

• Model: thousands of endnodes in a multicast connection. Application (say a whiteboard) wants all messages sent to be delivered to all listening endnodes. Like a transport.

• Endnodes use the MBONE for communication but the routers do not want to keep transport state like sequence numbers. Routers only do multicast forwarding.

• How to do error recovery?
Positive Acks do not work well

• If when a source endnode sends a multicast packet to a group, all receivers (e.g., 10,000) send an ack, we have many problems.

• First, source must know all receivers. Not easily possible in IP model. Source must keep state about all receivers, large amount of state.

• Second, large amount of acks for each message sent. Bandwidth expense. Called ack implosion problem.

• Third if a few receivers have lost packet, can’t tell IP to multicast to only those.
Negative Acks and Gap Detection

- Source attaches unique sequence number to each data item: (S, seq). Even with no data to send, source sends a low-rate multicast messages with last sequence number sent.

- Receiver who has sequence number $x$ and receives a message with sequence number $y > x$ knows its missing numbers $x + 1 \ldots y$.

- Receiver that detects gap can request missing data from source. But if all receivers are missing packet, we get request implosion again.
Handling Request Implosion

- Easy to handle request implosion if routers were to keep state about last sequence number received and pass up only one request for every connection. Bad idea for routers to keep transport state.

- Instead we use two ideas:
  - Any endnode in connection that has data from source can “repair” the gap, not just the source.
  - We try and pick a unique endnode to request and repair based on a combination of distance from the source and randomization.
Distance Based Election of Requestors/Responders
Need Randomization

- Suppose we have a star topology in which all destinations are the same distance from source. Then distance based election, does not prevent request or repair implosion.

- So receivers that detect a gap wait a random amount of time distributed between $C_1d$ and $(C_1 + C_2)d$. Any potential requestor that receives identical request cancels its own timer.

- Similarly, any endnode that receives a request for data it has sets a distance-based random timer value before it sends data for “repair”.