Multicast

EECS 122: Lecture 16

Department of Electrical Engineering and Computer Sciences
University of California
Berkeley

Midterm

Exam was not meant to be easy...

Performance quite good!
Worth understanding the answers…
30Mb burst from B being served
10Mb burst from B being served

Area of two triangles = 125Mb = 125K packets

Max delay = 7.5s

B's 10Mb burst takes 2.5s to serv
B's 30Mb burst takes 7.5s to serv

Avg Occupancy = 125K/20 = 6.25K

Avg Delay: Q/α = 6.25K/2K = 3.125s

Max Q = 15Mb

Playout buffer = 7.5*2 = 15Mb
Today

- Multicast
- RSVP

Broadcasting to Groups

- Many applications are not one-one
  - Webcasts
  - Group collaboration
  - Proxy/Cache updates
- Packets must reach a **Group** rather than a single destination
  - Group membership may be dynamic
  - More than one group member might be a source
- Idea: Notion of a Group Address
  - First sender establishes group, \( G \)
  - Interested receivers join the group
  - The network takes care of group management
The Multicast service Model

- To join a multicast
  - Host finds out the group address (a class D address)
  - Sends a “join” message to its local router
- To send to a group
  - Just send to the group
- Packet delivery is best effort

Internet Group Management Protocol
IGMP

- Operates between Router and local Hosts, typically attached via a LAN (e.g., Ethernet)
  - Query response architecture
1. Router periodically queries the local Hosts for group membership information
   - Can be specific or general
2. Hosts receiving query set a random timer before responding
3. First host to respond sends membership reports
4. All the other hosts observe the query and suppress their own reports.
- To Join send a group send an unsolicited Join
  - Start a group by joining it
  - To leave don’t have to do anything
  - Soft state
Naïve Routing Option: Don’t change anything

Point-to point routing

Group abstraction not implemented in the network

This approach does not scale…
Instead build trees

Copy data at routers
At most one copy of a data packet per link

- Routers keep track of groups in real-time
- "Path" computation is Tree computation

Issues

- How many trees per group?
  - Source-based
  - Shared

- Tree computation
  - Steiner Tree
  - Spanning Tree
  - Center based Shared Trees
  - Source based Trees with Reverse Path Forwarding

- Routing Protocols
  - DVMRP
  - PIM
How Many Trees per Group?

- **Source-based:** One tree for each source
  - **Pros**
    - Works well for webcasts
    - Can compute “Good” trees
  - **Cons**
    - In conference type applications everyone could be a source
      - Too many trees!
    - Router state explodes

- **Shared Trees:** one tree per group
  - **Pros**
    - Works well for multisource groups
    - Easier to maintain and compute
  - **Cons**
    - Link choice may not be the best for minimizing delay, maximizing throughput

---

Steiner Trees

*A tree which connects all the group nodes is a Steiner Tree*
Steiner Tree

- A tree which connects all the group nodes is a Steiner Tree
- Finding the min cost Steiner Tree is NP hard

Spanning Tree

- A tree that connects all of the nodes in the graph is a spanning tree
- Finding a minimum spanning tree is much easier
A tree that connects all of the nodes in the graph is a spanning tree.
Finding a minimum spanning tree is much easier.
Prune back to get a multicast tree.
Computing spanning trees

- Many algorithms known
- Two popularly implemented approaches
  - Center based approached
  - Source based Reverse Path Forwarding and pruning

Center-based trees

- single delivery tree shared by all
- one router identified as “center” of tree
- to join:
  - edge router sends unicast join-msg addressed to center router
  - join-msg “processed” by intermediate routers and forwarded towards center
  - join-msg either hits existing tree branch for this center, or arrives at center
  - path taken by join-msg becomes new branch of tree for this router
Spanning Tree: Center Based Approach

- Center node
- Each node sends unicast join message to center node
  - Message forwarded until it arrives at a node already belonging to spanning tree

(a) Stepwise construction of spanning tree
(b) Constructed spanning tree

Source Based Trees

- mcast forwarding tree: tree of shortest path routes from source to all receivers
  - Dijkstra’s algorithm

S: source

LEGEND
- router with attached group member
- router with no attached group member
- link used for forwarding, i indicates order link added by algorithm
Reverse Path Forwarding

- rely on router’s knowledge of unicast shortest path from it to sender
- each router has simple forwarding behavior:

  \[
  \text{if (mcast datagram received on incoming link on shortest path back to center)} \]

  \[
  \text{then flood datagram onto all outgoing links} \]

  \[
  \text{else ignore datagram} \]

---

Reverse Path Forwarding: example

![Diagram of network with routers and flows]

**Legend**

- router with attached group member
- router with no attached group member
- datagram will be forwarded
- datagram will not be forwarded

**Note:** Every link carries every multicast packet!
Reverse Path Forwarding: pruning

- forwarding tree contains subtrees with no mcast group members
  - no need to forward datagrams down subtree
  - “prune” msgs sent upstream by router with no downstream group members

S: source

Tunneling

**Q:** How to connect “islands” of multicast routers in a “sea” of unicast routers?

- mcast datagram encapsulated inside “normal” (non-mcast-addressed) datagram
- normal IP datagram sent thru “tunnel” via regular IP unicast to receiving mcast router
- receiving mcast router unencapsulates to get mcast datagram
Internet Multicasting Routing: DVMRP

- **DVMRP**: distance vector multicast routing protocol, RFC1075
- **flood and prune**: reverse path forwarding, source-based tree
  - RPF tree based on DVMRP’s own routing tables constructed by communicating DVMRP routers
  - no assumptions about underlying unicast
  - initial datagram to mcast group flooded everywhere via RPF
  - routers not wanting group: send upstream prune msgs

DVMRP: continued…

- **soft state**: DVMRP router periodically (1 min.) “forgets” branches are pruned:
  - mcast data again flows down unpruned branch
  - downstream router: reprune or else continue to receive data
- routers can quickly regraft to tree
  - following IGMP join at leaf
- odds and ends
  - commonly implemented in commercial routers
  - Mbone routing done using DVMRP
PIM

- Popular intradomain method
  - UUNET streaming using this
- Recognizes that most groups are very sparse
  - Why have all of the routers participate in keeping state?
- Two modes
  - Dense mode: flood and prune
  - Sparse mode: Center(core)-based shared tree approach

Resource Reservation Protocol: RSVP

- Signaling protocol to set up virtual circuits and trees in Intserv
  - Allows heterogeneous hosts to participate in the same group
  - Leverages multicast routing
  - Designed to keep protocol overhead linear in the number of receivers
RSVP: overview of operation

- Senders, receiver join a multicast group
- Path message from sender
  - make sender presence known to routers
  - path teardown: delete sender’s path state from routers
- Reservation message from receiver
  - reservation message: reserve resources from sender(s) to receiver
  - reservation teardown: remove receiver reservations

RSVP: simple audio conference

- H1, H2, H3, H4, H5 both senders and receivers
- multicast group m1
- no filtering: packets from any sender forwarded
- audio rate: \( b \)
- only one multicast routing tree possible
RSVP: building up path state

- H1, ..., H5 all send path messages on $m1$:
  (address=$m1$, Tspec=b, filter-spec=no-filter, refresh=100)
- Suppose H1 sends first path message

Suppose H1 sends first path message

RSVP: building up path state

- next, H5 sends path message, creating more state in routers
RSVP: building up path state

- H2, H3, H5 send path msgs, completing path state tables

Reservation Msgs: *receiver-to-network* signaling

- reservations flow upstream from receiver-to-senders, reserving resources, creating additional, *receiver-related* state at routers
RSVP: *receiver* reservation

H1 wants to receive audio from all other senders
- H1 reservation msg flows uptree to sources
- H1 only reserves enough bandwidth for 1 audio stream
- reservation is of type "no filter" – any sender can use reserved bandwidth

RSVP: *receiver* reservation

- H1 reservation msgs flows uptree to sources
- routers, hosts reserve bandwidth \( b \) needed on downstream links towards H1
RSVP: *receiver* reservation

- next, H2 makes reservation for bandwidth \( b \)
- H2 forwards to R1, R1 forwards to H1 and R2 (?)
- R2 takes no action, since \( b \) already reserved on L6