Broadcast Routing

deliver packets from source to all other nodes
source duplication is inefficient:

source duplication: how does source determine recipient addresses?
In-network duplication

flooding: when node receives brdcst pkt, sends copy to all neighbors
  - Problems: cycles & broadcast storm

controlled flooding: node only brdcsts pkt if it hasn’t brdcst same packet before
  - Node keeps track of pckt ids already brdcsted
  - Or reverse path forwarding (RPF): only forward pckt if it arrived on shortest path between node and source

spanning tree
  - No redundant packets received by any node
Spanning Tree

First construct a spanning tree
Nodes forward copies only along spanning tree

(a) Broadcast initiated at A
(b) Broadcast initiated at D
Spanning Tree: Creation

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
Multicast Routing: Problem Statement

• **Goal:** find a tree (or trees) connecting routers having local mcast group members
  - **tree:** not all paths between routers used
  - **source-based:** different tree from each sender to rcvrs
  - **shared-tree:** same tree used by all group members

![Shared tree](image1)
![Source-based trees](image2)
Approaches for building mcast trees

Approaches:

• **source-based tree:** one tree per source
  shortest path trees
  reverse path forwarding

• **group-shared tree:** group uses one tree
  minimal spanning (Steiner) center-based trees

...we first look at basic approaches, then specific protocols adopting these approaches.
mcast forwarding tree: tree of shortest path routes from source to all receivers

Dijkstra’s algorithm

**LEGEND**

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

S: source
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

- result is a source-specific reverse SPT
  - may be a bad choice with asymmetric links
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

LEGEND
- router with attached group member
- router with no attached group member
- prune message
- links with multicast forwarding
Shared-Tree: Steiner Tree

• **Steiner Tree:** minimum cost tree connecting all routers with attached group members

problem is NP-complete

excellent heuristics exists

not used in practice:

computational complexity

information about entire network needed

monolithic: re-run whenever a router needs to join/leave
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
Center-based trees: an example

Suppose R6 chosen as center:

![Diagram of network with routers R1 to R7, showing path order in which join messages are generated.]

**LEGEND**
- Router with attached group member
- Router with no attached group member
- Path order in which join messages generated
• **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
**Q:** How to connect “islands” of multicast routers in a “sea” of unicast routers?

- mcast datagram encapsulated inside “normal” (non-multicast-addressed) datagram
- normal IP datagram sent thru “tunnel” via regular IP unicast to receiving mcast router
- receiving mcast router unencapsulates to get mcast datagram
flood-and-prune RPF, similar to DVMRP but

- underlying unicast protocol provides RPF info for incoming datagram
- less complicated (less efficient) downstream flood than DVMRP reduces reliance on underlying routing algorithm
- has protocol mechanism for router to detect it is a leaf-node router
center-based approach
router sends *join* msg
to rendezvous point (RP)
intermediate routers update state and forward *join*
after joining via RP, router can switch to source-specific tree
increased performance: less concentration, shorter paths
sender(s):
unicast data to RP, which distributes down RP-rooted tree
RP can extend mcast tree upstream to source
RP can send *stop* msg if no attached receivers

![Diagram](attachment:image.png)

- R1
- R2
- R3
- R4
- R5
- R6
- R7

all data multicast from rendezvous point
rendezvous point
Consequences of Sparse-Dense Dichotomy:

**Dense**
- group membership by routers *assumed* until routers explicitly prune
- *data-driven* construction on mcast tree (e.g., RPF)
- bandwidth and non-group-router processing *profligate*

**Sparse:**
- no membership until routers explicitly join
- *receiver-driven* construction of mcast tree (e.g., center-based)
- bandwidth and non-group-router processing *conservative*