Chapter 4
Network Layer

A note on the use of these ppt slides:
We’re making these slides freely available to all (faculty, students, readers). They’re in PowerPoint form so you can add, modify, and delete slides (including this one) and slide content to suit your needs. They obviously represent a lot of work on our part. In return for use, we only ask the following:

- If you use these slides (e.g., in a class) in substantially unaltered form, that you mention their source (after all, we’d like people to use our book!)
- If you post any slides in substantially unaltered form on a www site, that you note that they are adapted from (or perhaps identical to) our slides, and note our copyright of this material.

Thanks and enjoy! JFK/KWR

All material copyright 1996-2010
J.F Kurose and K.W. Ross, All Rights Reserved
Chapter 4: Network Layer

4.1 Introduction
4.2 Virtual circuit and datagram networks
4.3 What’s inside a router
4.4 IP: Internet Protocol
   - Datagram format
   - IPv4 addressing
   - ICMP
   - IPv6
4.5 Routing algorithms
   - Link state
   - Distance Vector
   - Hierarchical routing
4.6 Routing in the Internet
   - RIP
   - OSPF
   - BGP
4.7 Broadcast and multicast routing
Broadcast Routing

- deliver packets from source to all other nodes

- source duplication is inefficient!
- source duplication: how does source determine recipient addresses?
Broadcast Algorithms: bcst 1.0

- “Whenever you receive a message, duplicate it and send it again on all neighboring link”
- What is wrong with this approach?
  The previous neighbor received it! It never stops!
Broadcast Algorithms: bcst 1.1

- “Whenever you receive a message, duplicate it and send it again on all neighboring link, except the one you received it from”
- What is wrong with this approach? If the graph has any loop, it also never stops!
Broadcast Algorithms: bcst 1.2

- “Whenever you first receive a message, duplicate it and send it again on all neighboring link, except the one you received it from.”
- At least it is correct, called PI (Prop. Info)
- Still not very satisfying?
  1. Uses all links in the network (cost $\sim O(M)$)
  2. Need to remember all messages sent
  3. Does not provide feedback (was it received?)
Broadcast Algorithms: bcst 1.3

- Can we modify bcst 1.2 to solve feedback
- First guess: YES, perhaps
  - Broadcast an ACK for all nodes? Unicast to src?
  - It's costly! (~ as much as source duplication)
  - Only works if full set known in advance
- Second guess: Exploit the broadcast phase
  - Let p(i) be nodes i received the message first
  - “When received message or ACK from all links, this node “terminates”; it sends an ACK to p(i).”
  - Node i terminates before p(i), all terminate
Example
From flooding to spanning tree

- Key feature: collection of links \( (p(i),i) \)
  - It is a subgraph of the original graph
  - It does not contain cycle
  - It contains all nodes
  This is called a spanning tree

- Properties of spanning trees
  1. They have exactly \( N-1 \) edges and \( N \) nodes
     (This is the minimum number to connect all)
  2. Many possible ways to construct one
     But also, spanning trees are fragile!
Spanning Tree

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

(a) Broadcast initiated at A
(b) Broadcast initiated at D
Center-based tree: bcst 2.0

- 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
A general construction

- Given a connected graph
  - Start with $S=\{s\}$, $A=\{\}$
  - While (S does not contain all nodes) {
    - Pick an edge $(i, j)$ such that $i$ in $S$, $j$ not in $S$
    - do $S<- S+\{j\}$ ; $A<- A+\{(i, j)\}$
  }

- Proof
  1. There always exists such edge
  2. $S,A$ is always a tree
  3. After N-1 steps, all nodes are included
Optimal Spanning Tree

- Is there a “best” spanning tree?
  - The one whose entire cost is minimum
  - Minimum Spanning Tree (MST)
- A greedy algorithm finds MST
  - Start with $S=\{s\}$, $A=\{}$
  - While ($S$ does not contain all nodes) {
    - Pick the edge $(i,j)$ such that
      - $i$ in $S$, $j$ not in $S$
      - and $(i,j)$ has minimum weight
    - do $S\leftarrow S+\{j\}$ ; $A\leftarrow A+\{(i,j)\}$
  }
Example

- Original topology

- Algorithm follows the following 6 steps

- Proof: At any time, a link in A belongs to the spanning tree
Reverse Path Forward: bcst3.0

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

  
  ```
  if (mcast datagram received on incoming link on shortest path back to center)
      then flood datagram onto all outgoing links
  else ignore datagram
  ```
Broadcast Algorithms

- flooding: when node receives broadcast packet, sends copy to all neighbors
  - problems: cycles & broadcast storm
- controlled flooding: node only broadcasts pkt if it hasn’t broadcasted same packet before
  - node keeps track of packet ids already broadcasted
  - or reverse path forwarding (RPF): only forward packet if it arrived on shortest path between node and source
- spanning tree
  - No redundant packets received by any node
Chapter 4: Network Layer

4.1 Introduction

4.2 Virtual circuit and datagram networks

4.3 What’s inside a router

4.4 IP: Internet Protocol
   - Datagram format
   - IPv4 addressing
   - ICMP
   - IPv6

4.5 Routing algorithms
   - Link state
   - Distance Vector
   - Hierarchical routing

4.6 Routing in the Internet
   - RIP
   - OSPF
   - BGP

4.7 Broadcast and multicast routing
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
Why Johnny can’t multicast?

Broadcast
- Flooding (PI)
- Spanning trees
  - Avoid duplication
  - Allow termination (PIF)
  - Reduces memory
- Optimal:
  - Needs full information
  - Minimum spanning tree
  - Prim-Dijkstra or Kruskal
- Distributed:
  - Reverse Path Forwarding
  - Center based tree

Multicast
- Flooding = overshooting
- Spanning trees
- Optimal: Steiner Tree
  - Needs full information
  - NP Hard, approximable
- Distributed:
  - All the others + Pruning
- Not really used
  - Inter-domain routing hard
  - Congestion control hard
  - What is used instead: mesh, epidemic, p2p
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: rerun whenever a router needs to join/leave
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
Shortest Path Tree

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

![Diagram]

**LEGEND**
- S: source
- 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:

```plaintext
if (mcast datagram received on incoming link on shortest path back to center)
  then flood datagram onto all outgoing links
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
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:
**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
Tunneling

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

physical topology
logical topology
**PIM: Protocol Independent Multicast**

- not dependent on any specific underlying unicast routing algorithm (works with all)
- two different multicast distribution scenarios:
  
  **Dense:**
  - group members densely packed, in “close” proximity.
  - bandwidth more plentiful

  **Sparse:**
  - # networks with group members small wrt # interconnected networks
  - group members “widely dispersed”
  - bandwidth not plentiful
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*
PIM- Dense Mode

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
PIM - Sparse Mode

- center-based approach
- router sends \textit{join} msg to rendezvous point (RP)
  - intermediate routers update state and forward \textit{join}
- after joining via RP, router can switch to source-specific tree
  - increased performance: less concentration, shorter paths

![Diagram of PIM Sparse Mode](image-url)
**PIM - Sparse Mode**

**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
  - “no one is listening!”
Chapter 4: summary

4.1 Introduction
4.2 Virtual circuit and datagram networks
4.3 What’s inside a router
4.4 IP: Internet Protocol
   - Datagram format
   - IPv4 addressing
   - ICMP
   - IPv6
4.5 Routing algorithms
   - Link state
   - Distance Vector
   - Hierarchical routing
4.6 Routing in the Internet
   - RIP
   - OSPF
   - BGP
4.7 Broadcast and multicast routing