Data Communications & Networks

Session 8 – Main Theme
Networks: Part II
Circuit Switching, Packet Switching, The Network Layer

Dr. Jean-Claude Franchitti
New York University
Computer Science Department
Courant Institute of Mathematical Sciences

Adapted from course textbook resources
Computer Networking: A Top-Down Approach, 5/E
Copyright 1996-2013
J.F. Kurose and K.W. Ross, All Rights Reserved
1. Session Overview
2. Networks Part 2
3. Summary and Conclusion
What is the class about?

- **Course description and syllabus:**
  - [http://www.nyu.edu/classes/jcf/csci-ga.2262-001/](http://www.nyu.edu/classes/jcf/csci-ga.2262-001/)
  - [http://cs.nyu.edu/courses/Spring13/CSCI-GA.2262-001/index.html](http://cs.nyu.edu/courses/Spring13/CSCI-GA.2262-001/index.html)

- **Textbooks:**
    - James F. Kurose, Keith W. Ross
    - Addison Wesley
Course Overview

- Computer Networks and the Internet
- Application Layer
- Fundamental Data Structures: queues, ring buffers, finite state machines
- Data Encoding and Transmission
- Local Area Networks and Data Link Control
- Wireless Communications
- Packet Switching
- OSI and Internet Protocol Architecture
- Congestion Control and Flow Control Methods
- Internet Protocols (IP, ARP, UDP, TCP)
- Network (packet) Routing Algorithms (OSPF, Distance Vector)
- IP Multicast
- Sockets
Course Approach

- Introduction to Basic Networking Concepts (Network Stack)
- Origins of Naming, Addressing, and Routing (TCP, IP, DNS)
- Physical Communication Layer
- MAC Layer (Ethernet, Bridging)
- Routing Protocols (Link State, Distance Vector)
- Internet Routing (BGP, OSPF, Programmable Routers)
- TCP Basics (Reliable/Unreliable)
- Congestion Control
- QoS, Fair Queuing, and Queuing Theory
- Network Services – Multicast and Unicast
- Extensions to Internet Architecture (NATs, IPv6, Proxies)
- Network Hardware and Software (How to Build Networks, Routers)
- Overlay Networks and Services (How to Implement Network Services)
- Network Firewalls, Network Security, and Enterprise Networks
Networks Part 2 Session in Brief

- Understand principles behind network layer services:
  - Routing (path selection)
  - Dealing with scale
  - Advanced topics: IPv6, mobility
- Instantiation, implementation in the Internet
- Conclusion
1. Session Overview
2. Networks Part 2
3. Summary and Conclusion
Networks Part 2 Session in Brief

- Routing algorithms
  - Link state
  - Distance Vector
  - Hierarchical routing
- Routing in the Internet
  - RIP
  - OSPF
  - BGP
- Broadcast and multicast routing
Interplay between routing, forwarding

Routing algorithm

Local forwarding table

<table>
<thead>
<tr>
<th>header value</th>
<th>output link</th>
</tr>
</thead>
<tbody>
<tr>
<td>0100</td>
<td>3</td>
</tr>
<tr>
<td>0101</td>
<td>2</td>
</tr>
<tr>
<td>0111</td>
<td>2</td>
</tr>
<tr>
<td>1001</td>
<td>1</td>
</tr>
</tbody>
</table>

Value in arriving packet’s header

0111
Graph abstraction

Graph: $G = (N, E)$

$N =$ set of routers = $\{ u, v, w, x, y, z \}$

$E =$ set of links $\{ (u,v), (u,x), (v,x), (v,w), (x,w), (x,y), (w,y), (w,z), (y,z) \}$

Remark: Graph abstraction is useful in other network contexts

Example: P2P, where $N$ is set of peers and $E$ is set of TCP connections
Graph abstraction: costs

- $c(x,x') =$ cost of link $(x,x')$
  - e.g., $c(w,z) = 5$
- cost could always be 1, or inversely related to bandwidth, or inversely related to congestion

Cost of path $(x_1, x_2, x_3, \ldots, x_p) = c(x_1,x_2) + c(x_2,x_3) + \ldots + c(x_{p-1},x_p)$

Question: What’s the least-cost path between $u$ and $z$?

Routing algorithm: algorithm that finds least-cost path
Routing Algorithm classification

Global or decentralized information?

Global:
- all routers have complete topology, link cost info
- “link state” algorithms

Decentralized:
- router knows physically-connected neighbors, link costs to neighbors
- iterative process of computation, exchange of info with neighbors
- “distance vector” algorithms

Static or dynamic?

Static:
- routes change slowly over time

Dynamic:
- routes change more quickly
  - periodic update
  - in response to link cost changes
Networks Part 2 Session in Brief

- Routing algorithms
  - Link state
  - Distance Vector
  - Hierarchical routing
- Routing in the Internet
  - RIP
  - OSPF
  - BGP
- Broadcast and multicast routing
A Link-State Routing Algorithm

Dijkstra’s algorithm

- net topology, link costs known to all nodes
  - accomplished via “link state broadcast”
  - all nodes have same info
- computes least cost paths from one node (‘source”) to all other nodes
  - gives forwarding table for that node
- iterative: after k iterations, know least cost path to k dest.’s

Notation:

- \( c(x,y) \): link cost from node \( x \) to \( y \); \( = \infty \) if not direct neighbors
- \( D(v) \): current value of cost of path from source to dest. \( v \)
- \( p(v) \): predecessor node along path from source to \( v \)
- \( N' \): set of nodes whose least cost path definitively known
Dijsktra’s Algorithm

1  *Initialization:*
2    N' = \{u\}
3    for all nodes v
4      if v adjacent to u
5         then D(v) = c(u,v)
6      else D(v) = \infty
7
8  *Loop*
9    find w not in N' such that D(w) is a minimum
10   add w to N'
11   update D(v) for all v adjacent to w and not in N' :
12      D(v) = \min(D(v), D(w) + c(w,v))
13      /* new cost to v is either old cost to v or known
14        shortest path cost to w plus cost from w to v */
15  until all nodes in N'
Dijkstra’s algorithm: example

<table>
<thead>
<tr>
<th>Step</th>
<th>N'</th>
<th>D(v),p(v)</th>
<th>D(w),p(w)</th>
<th>D(x),p(x)</th>
<th>D(y),p(y)</th>
<th>D(z),p(z)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>u</td>
<td>2,u</td>
<td>5,u</td>
<td>1,u</td>
<td>∞</td>
<td>∞</td>
</tr>
<tr>
<td>1</td>
<td>ux</td>
<td>2,u</td>
<td>4,x</td>
<td>2,x</td>
<td>4,y</td>
<td>4,y</td>
</tr>
<tr>
<td>2</td>
<td>uxy</td>
<td>2,u</td>
<td>3,y</td>
<td>3,y</td>
<td>4,y</td>
<td>4,y</td>
</tr>
<tr>
<td>3</td>
<td>uxyv</td>
<td>2,u</td>
<td>3,y</td>
<td>3,y</td>
<td>4,y</td>
<td>4,y</td>
</tr>
<tr>
<td>4</td>
<td>uxyvw</td>
<td>2,u</td>
<td>3,y</td>
<td>3,y</td>
<td>4,y</td>
<td>4,y</td>
</tr>
<tr>
<td>5</td>
<td>uxyvwz</td>
<td>2,u</td>
<td>3,y</td>
<td>3,y</td>
<td>4,y</td>
<td>4,y</td>
</tr>
</tbody>
</table>

The diagram shows the network with nodes u, x, y, w, v, and z, and the edges with their respective weights. The table summarizes the steps of Dijkstra’s algorithm, showing the distances (D) and the predecessors (p) for each node at each step.
### Resulting shortest-path tree from u:

![Diagram of the shortest-path tree](image)

### Resulting forwarding table in u:

<table>
<thead>
<tr>
<th>destination</th>
<th>link</th>
</tr>
</thead>
<tbody>
<tr>
<td>v</td>
<td>(u,v)</td>
</tr>
<tr>
<td>x</td>
<td>(u,x)</td>
</tr>
<tr>
<td>y</td>
<td>(u,x)</td>
</tr>
<tr>
<td>w</td>
<td>(u,x)</td>
</tr>
<tr>
<td>z</td>
<td>(u,x)</td>
</tr>
</tbody>
</table>
Dijkstra’s algorithm, discussion

Algorithm complexity: n nodes
- each iteration: need to check all nodes, w, not in N
- \( n(n+1)/2 \) comparisons: \( O(n^2) \)
- more efficient implementations possible: \( O(n \log n) \)

Oscillations possible:
- e.g., link cost = amount of carried traffic

Initially

... recompute routing

... recompute

... recompute
Networks Part 2 Session in Brief

- Routing algorithms
  - Link state
  - Distance Vector
  - Hierarchical routing
- Routing in the Internet
  - RIP
  - OSPF
  - BGP
- Broadcast and multicast routing
**Bellman-Ford Equation (dynamic programming)**

Define
\[ d_x(y) := \text{cost of least-cost path from } x \text{ to } y \]

Then
\[ d_x(y) = \min_v \{ c(x,v) + d_v(y) \} \]

where min is taken over all neighbors v of x
Clearly, \( d_v(z) = 5, \) \( d_x(z) = 3, \) \( d_w(z) = 3 \)

B-F equation says:
\[
d_u(z) = \min \{ c(u,v) + d_v(z), \quad c(u,x) + d_x(z), \quad c(u,w) + d_w(z) \}
= \min \{2 + 5, \quad 1 + 3, \quad 5 + 3 \} = 4
\]

Node that achieves minimum is next hop in shortest path ➜ forwarding table
Distance Vector Algorithm

- $D_x(y) = \text{estimate of least cost from } x \text{ to } y$
- Node $x$ knows cost to each neighbor $v$: $c(x,v)$
- Node $x$ maintains distance vector $D_x = [D_x(y): y \in N ]$
- Node $x$ also maintains its neighbors’ distance vectors
  » For each neighbor $v$, $x$ maintains $D_v = [D_v(y): y \in N ]$
Basic idea:

- From time-to-time, each node sends its own distance vector estimate to neighbors
- Asynchronous
- When a node $x$ receives new DV estimate from neighbor, it updates its own DV using B-F equation:

$$D_x(y) \leftarrow \min_v \{ c(x,v) + D_v(y) \} \quad \text{for each node } y \in N$$

- Under minor, natural conditions, the estimate $D_x(y)$ converge to the actual least cost $d_x(y)$
Iterative, asynchronous: each local iteration caused by:
- local link cost change
- DV update message from neighbor

Distributed:
- each node notifies neighbors *only* when its DV changes
  » neighbors then notify their neighbors if necessary

Each node:

wait for (change in local link cost or msg from neighbor)

recompute estimates

if DV to any dest has changed, *notify* neighbors
\[ D_x(y) = \min \{ c(x,y) + D_y(y), c(x,z) + D_z(y) \} \]
\[ = \min \{ 2 + 0, 7 + 1 \} = 2 \]

\[ D_x(z) = \min \{ c(x,y) + D_y(z), c(x,z) + D_z(z) \} \]
\[ = \min \{ 2 + 1, 7 + 0 \} = 3 \]
\[ D_x(y) = \min\{c(x,y) + D_y(y), c(x,z) + D_z(y)\} = \min\{2+0, 7+1\} = 2 \]

\[ D_x(z) = \min\{c(x,y) + D_y(z), c(x,z) + D_z(z)\} = \min\{2+1, 7+0\} = 3 \]

|       | x   | y   | z   | x   | y   | z   | x   | y   | z   | x   | y   | z   | x   | y   | z   | x   | y   | z   | x   | y   | z   | x   | y   | z   | x   | y   | z   | x   | y   | z   | x   | y   | z   |
|-------|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| from  |     |     |     | x   | 0   | 2   | 7   | y   | 0   | 2   | 3   | x   | 0   | 2   | 3   | y   | 0   | 2   | 3   | z   | 7   | 1   | 0   | x   | 0   | 2   | 3   | y   | 0   | 2   | 3   | z   | 7   | 1   | 0   | x   | 0   | 2   | 3   | y   | 0   | 2   | 3   | z   | 7   | 1   | 0   |
| x     | x   | ∞   | ∞   | y   | 2   | 0   | 1   | z   | ∞   | ∞   | ∞   | x   | ∞   | ∞   | ∞   | y   | 2   | 0   | 1   | z   | ∞   | ∞   | ∞   | x   | ∞   | ∞   | ∞   | y   | 2   | 0   | 1   | z   | ∞   | ∞   | ∞   | x   | ∞   | ∞   | ∞   | y   | 2   | 0   | 1   |
| y     | x   | ∞   | ∞   | y   | 2   | 0   | 1   | z   | 7   | 1   | 0   | x   | 0   | 2   | 7   | y   | 2   | 0   | 1   | z   | 7   | 1   | 0   | x   | 0   | 2   | 7   | y   | 2   | 0   | 1   | z   | 7   | 1   | 0   | x   | 0   | 2   | 7   | y   | 2   | 0   | 1   |
| z     | x   | ∞   | ∞   | y   | 2   | 0   | 1   | z   | 7   | 1   | 0   | x   | 0   | 2   | 7   | y   | 2   | 0   | 1   | z   | 7   | 1   | 0   | x   | 0   | 2   | 7   | y   | 2   | 0   | 1   | z   | 7   | 1   | 0   | x   | 0   | 2   | 7   | y   | 2   | 0   | 1   |

Diagram: Node X, Y, Z with edges and costs.
Distance Vector: link cost changes

Link cost changes:
- node detects local link cost change
- updates routing info, recalculates distance vector
- if DV changes, notify neighbors

"good news travels fast"

At time $t_0$, $y$ detects the link-cost change, updates its DV, and informs its neighbors.

At time $t_1$, $z$ receives the update from $y$ and updates its table. It computes a new least cost to $x$ and sends its neighbors its DV.

At time $t_2$, $y$ receives $z$'s update and updates its distance table. $y$'s least costs do not change and hence $y$ does not send any message to $z$. 
Distance Vector: link cost changes

Link cost changes:

- good news travels fast
- bad news travels slow - “count to infinity” problem!
- 44 iterations before algorithm stabilizes: see text

Poisoned reverse:

- If Z routes through Y to get to X:
  - Z tells Y its (Z’s) distance to X is infinite (so Y won’t route to X via Z)
- will this completely solve count to infinity problem?
Comparison of LS and DV algorithms

Message complexity

- **LS**: with n nodes, E links, \( O(nE) \) msgs sent
- **DV**: exchange between neighbors only
  - convergence time varies

Speed of Convergence

- **LS**: \( O(n^2) \) algorithm requires \( O(nE) \) msgs
  - may have oscillations
- **DV**: convergence time varies
  - may be routing loops
  - count-to-infinity problem

Robustness: what happens if router malfunctions?

- **LS**:
  - node can advertise incorrect link cost
  - each node computes only its own table

- **DV**:
  - DV node can advertise incorrect path cost
  - each node’s table used by others
    - error propagate thru network
Networks Part 2 Session in Brief

- Routing algorithms
  - Link state
  - Distance Vector
  - Hierarchical routing
- Routing in the Internet
  - RIP
  - OSPF
  - BGP
- Broadcast and multicast routing
Hierarchical Routing

Our routing study thus far - idealization

- all routers identical
- network “flat”

... not true in practice

scale: with 200 million destinations:

- can’t store all dest’s in routing tables!
- routing table exchange would swamp links!

administrative autonomy

- internet = network of networks
- each network admin may want to control routing in its own network
Hierarchical Routing

- aggregate routers into regions, “autonomous systems” (AS)
- routers in same AS run same routing protocol
  - “intra-AS” routing protocol
  - routers in different AS can run different intra-AS routing protocol

Gateway router
- Direct link to router in another AS
Interconnected ASes

- Forwarding table configured by both intra- and inter-AS routing algorithm
  - intra-AS sets entries for internal dests
  - inter-AS & intra-As sets entries for external dests
Inter-AS tasks

- Suppose router in AS1 receives datagram destined outside of AS1:
  - Router should forward packet to gateway router, but which one?

  **AS1 must:**

  1. Learn which dests are reachable through AS2, which through AS3
  2. Propagate this reachability info to all routers in AS1

Job of inter-AS routing!
Example: Setting forwarding table in router 1d

- Suppose AS1 learns (via inter-AS protocol) that subnet $x$ reachable via AS3 (gateway 1c) but not via AS2.
- Inter-AS protocol propagates reachability info to all internal routers.
- Router 1d determines from intra-AS routing info that its interface $I$ is on the least cost path to 1c.

  » Installs forwarding table entry $(x, I)$
now suppose AS1 learns from inter-AS protocol that subnet $x$ is reachable from AS3 \textit{and} from AS2.

- to configure forwarding table, router 1d must determine towards which gateway it should forward packets for dest $x$.

  » this is also job of inter-AS routing protocol!
now suppose AS1 learns from inter-AS protocol that subnet $x$ is reachable from AS3 \textit{and} from AS2.

- to configure forwarding table, router 1d must determine towards which gateway it should forward packets for dest $x$.
  
  » this is also job of inter-AS routing protocol!

- \textbf{hot potato routing:} send packet towards closest of two routers.

\begin{itemize}
  \item Learn from inter-AS protocol that subnet $x$ is reachable via multiple gateways
  \item Use routing info from intra-AS protocol to determine costs of least-cost paths to each of the gateways
  \item Hot potato routing: Choose the gateway that has the smallest least cost
  \item Determine from forwarding table the interface I that leads to least-cost gateway. Enter (x,I) in forwarding table
\end{itemize}
Networks Part 2 Session in Brief

- Routing algorithms
  - Link state
  - Distance Vector
  - Hierarchical routing
- Routing in the Internet
  - RIP
  - OSPF
  - BGP
- Broadcast and multicast routing
Intra-AS Routing

- also known as **Interior Gateway Protocols (IGP)**
- most common Intra-AS routing protocols:
  - RIP: Routing Information Protocol
  - OSPF: Open Shortest Path First
  - IGRP: Interior Gateway Routing Protocol (Cisco proprietary)
Routing algorithms
- Link state
- Distance Vector
- Hierarchical routing

Routing in the Internet
- RIP
- OSPF
- BGP

Broadcast and multicast routing
- distance vector algorithm
- included in BSD-UNIX Distribution in 1982
- distance metric: # of hops (max = 15 hops)

From router A to subnets:

<table>
<thead>
<tr>
<th>destination</th>
<th>hops</th>
</tr>
</thead>
<tbody>
<tr>
<td>u</td>
<td>1</td>
</tr>
<tr>
<td>v</td>
<td>2</td>
</tr>
<tr>
<td>w</td>
<td>2</td>
</tr>
<tr>
<td>x</td>
<td>3</td>
</tr>
<tr>
<td>y</td>
<td>3</td>
</tr>
<tr>
<td>z</td>
<td>2</td>
</tr>
</tbody>
</table>
RIP advertisements

- **distance vectors**: exchanged among neighbors every 30 sec via Response Message (also called advertisement)
- each advertisement: list of up to 25 destination subnets within AS
### Routing/Forwarding table in D

<table>
<thead>
<tr>
<th>Destination Network</th>
<th>Next Router</th>
<th>Num. of hops to dest.</th>
</tr>
</thead>
<tbody>
<tr>
<td>w</td>
<td>A</td>
<td>2</td>
</tr>
<tr>
<td>y</td>
<td>B</td>
<td>2</td>
</tr>
<tr>
<td>z</td>
<td>B</td>
<td>7</td>
</tr>
<tr>
<td>x</td>
<td>--</td>
<td>1</td>
</tr>
<tr>
<td>....</td>
<td>....</td>
<td>....</td>
</tr>
</tbody>
</table>

**RIP: Example**
RIP: Example

<table>
<thead>
<tr>
<th>Dest</th>
<th>Next</th>
<th>hops</th>
</tr>
</thead>
<tbody>
<tr>
<td>w</td>
<td>-</td>
<td>1</td>
</tr>
<tr>
<td>x</td>
<td>-</td>
<td>1</td>
</tr>
<tr>
<td>z</td>
<td>C</td>
<td>4</td>
</tr>
</tbody>
</table>

Routing/Forwarding table in D

<table>
<thead>
<tr>
<th>Destination Network</th>
<th>Next Router</th>
<th>Num. of hops to dest.</th>
</tr>
</thead>
<tbody>
<tr>
<td>w</td>
<td>A</td>
<td>2</td>
</tr>
<tr>
<td>y</td>
<td>B</td>
<td>2</td>
</tr>
<tr>
<td>z</td>
<td>B</td>
<td>7</td>
</tr>
<tr>
<td>x</td>
<td>--</td>
<td>1</td>
</tr>
<tr>
<td>....</td>
<td>....</td>
<td>....</td>
</tr>
</tbody>
</table>

Advertisement from A to D
If no advertisement heard after 180 sec -->
neighbor/link declared dead
» routes via neighbor invalidated
» new advertisements sent to neighbors
» neighbors in turn send out new advertisements (if tables changed)
» link failure info quickly (?) propagates to entire net
» *poison reverse* used to prevent ping-pong loops (infinite distance = 16 hops)
RIP Table processing

- RIP routing tables managed by **application-level** process called route-d (daemon)
- advertisements sent in UDP packets, periodically repeated

<table>
<thead>
<tr>
<th></th>
<th>routed</th>
<th>routed</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Transprt (UDP)</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>network (IP)</td>
<td>forwarding table</td>
<td>forwarding table</td>
</tr>
<tr>
<td>link</td>
<td></td>
<td></td>
</tr>
<tr>
<td>physical</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Transprt (UDP)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>network (IP)</td>
<td>forwarding table</td>
<td>link</td>
</tr>
<tr>
<td>link</td>
<td></td>
<td>physical</td>
</tr>
</tbody>
</table>
Networks Part 2 Session in Brief

- Routing algorithms
  - Link state
  - Distance Vector
  - Hierarchical routing
- Routing in the Internet
  - RIP
  - OSPF
  - BGP
- Broadcast and multicast routing
OSPF (Open Shortest Path First)

- “open”: publicly available
- uses Link State algorithm
  - LS packet dissemination
  - topology map at each node
  - route computation using Dijkstra’s algorithm

- OSPF advertisement carries one entry per neighbor router
- advertisements disseminated to entire AS (via flooding)
  - carried in OSPF messages directly over IP (rather than TCP or UDP)
OSPF “advanced” features (not in RIP)

- **security**: all OSPF messages authenticated (to prevent malicious intrusion)
- **multiple same-cost paths** allowed (only one path in RIP)
- For each link, multiple cost metrics for different TOS (e.g., satellite link cost set “low” for best effort; high for real time)
- integrated uni- and **multicast** support:
  » Multicast OSPF (MOSPF) uses same topology data base as OSPF
- **hierarchical** OSPF in large domains.
Hierarchical OSPF
Hierarchical OSPF

- **two-level hierarchy:** local area, backbone.
  - Link-state advertisements only in area
  - Each node has detailed area topology; only know direction (shortest path) to nets in other areas.
- **area border routers:** “summarize” distances to nets in own area, advertise to other Area Border routers.
- **backbone routers:** run OSPF routing limited to backbone.
- **boundary routers:** connect to other AS’s.
Routing algorithms
  - Link state
  - Distance Vector
  - Hierarchical routing

Routing in the Internet
  - RIP
  - OSPF
  - BGP

Broadcast and multicast routing
BGP (Border Gateway Protocol): the de facto standard

BGP provides each AS a means to:

1. Obtain subnet reachability information from neighboring ASs.
2. Propagate reachability information to all AS-internal routers.
3. Determine “good” routes to subnets based on reachability information and policy.

allows subnet to advertise its existence to rest of Internet: “I am here”
BGP basics

- pairs of routers (BGP peers) exchange routing info over semi-permanent TCP connections: **BGP sessions**
  - BGP sessions need not correspond to physical links.

- when AS2 advertises a prefix to AS1:
  - AS2 **promises** it will forward datagrams towards that prefix.
  - AS2 can aggregate prefixes in its advertisement.
Distributing reachability info

- using eBGP session between 3a and 1c, AS3 sends prefix reachability info to AS1.
  - 1c can then use iBGP do distribute new prefix info to all routers in AS1
  - 1b can then re-advertise new reachability info to AS2 over 1b-to-2a eBGP session
- when router learns of new prefix, it creates entry for prefix in its forwarding table.
Path attributes & BGP routes

- advertised prefix includes BGP attributes.
  - prefix + attributes = “route”

- two important attributes:
  - **AS-PATH**: contains ASs through which prefix advertisement has passed: e.g, AS 67, AS 17
  - **NEXT-HOP**: indicates specific internal-AS router to next-hop AS. (may be multiple links from current AS to next-hop-AS)

- when gateway router receives route advertisement, uses **import policy** to accept/decline.
BGP route selection

- router may learn about more than 1 route to some prefix. Router must select route.
- elimination rules:
  1. local preference value attribute: policy decision
  2. shortest AS-PATH
  3. closest NEXT-HOP router: hot potato routing
  4. additional criteria
BGP messages

- BGP messages exchanged using TCP.
- BGP messages:
  - **OPEN**: opens TCP connection to peer and authenticates sender
  - **UPDATE**: advertises new path (or withdraws old)
  - **KEEPALIVE**: keeps connection alive in absence of UPDATES; also ACKs OPEN request
  - **NOTIFICATION**: reports errors in previous msg; also used to close connection
- A, B, C are provider networks
- X, W, Y are customer (of provider networks)
- X is dual-homed: attached to two networks
  - X does not want to route from B via X to C
  - .. so X will not advertise to B a route to C
- A advertises path AW to B
- B advertises path BAW to X
- Should B advertise path BAW to C?
  - No way! B gets no “revenue” for routing CBAW since neither W nor C are B’s customers
  - B wants to force C to route to w via A
  - B wants to route *only* to/from its customers!
Why different Intra- and Inter-AS routing?

Policy:
- Inter-AS: admin wants control over how its traffic routed, who routes through its net.
- Intra-AS: single admin, so no policy decisions needed

Scale:
- hierarchical routing saves table size, reduced update traffic

Performance:
- Intra-AS: can focus on performance
- Inter-AS: policy may dominate over performance
Networks Part 2 Session in Brief

- Routing algorithms
  - Link state
  - Distance Vector
  - Hierarchical routing
- Routing in the Internet
  - RIP
  - OSPF
  - BGP
- Broadcast and multicast 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 pckt, 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
- First construct a spanning tree
- Nodes forward copies only along spanning tree

(a) Broadcast initiated at A  
(b) Broadcast initiated at D
- 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
**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
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
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
- 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
**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
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
Suppose R6 chosen as center:

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
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*
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
  » “no one is listening!”
Summary

- Routing algorithms
  - Link state
  - Distance Vector
  - Hierarchical routing
- Routing in the Internet
  - RIP
  - OSPF
  - BGP
  - Broadcast and multicast routing
Assignments & Readings

- Readings
  - Chapter 4
- No Assignment