Network Layer: Part II

- Basic Routing Principles and Routing Algorithms
  - Link State vs. Distance Vector
- Routing in the Internet
  - Intra-AS vs. Inter-AS routing
  - Intra-AS: RIP and OSPF
  - Inter-AS: BGP and Policy Routing
- Broadcast and Multicast Routing

Readings: Textbook: Chapter 4, Sections 4.5-4.8
Routing & Forwarding: Logical View of a Router

Control Plane (Routing Processor)
- routing agent (daemon)
- routing table
  - update

Data Plane (Forwarding Engine)
- switching fabric
- input interfaces (ports)
- output interfaces (ports)

Routing information
- lookup
- forwarding table
  - update

Input links

Diagram showing network nodes A, B, C, D, E, and F with numbered connections.
IP Forwarding & IP/ICMP Protocol

Transport layer: TCP, UDP

Routing protocols
• path selection
• RIP, OSPF, BGP

IP protocol
• addressing conventions
• packet handling conventions

ICMP protocol
• error reporting
• router “signaling”

Data Link layer (Ethernet, WiFi, PPP, …)

Physical Layer (SONET, …)
Routing: Issues

- How are routing tables determined?
- Who determines table entries?
- What info used in determining table entries?
- When do routing table entries change?
- Where is routing info stored?
- How to control routing table size?

Answer these questions, we are done!
Routing Paradigms

• Hop-by-hop Routing
  - Each packet contains destination address
  - Each router chooses next-hop to destination
    • routing decision made at each (intermediate) hop!
    • packets to same destination may take different paths!
  - Example: IP’s default datagram routing

• Source Routing
  - Sender selects the path to destination precisely
  - Routers forward packet to next-hop as specified
    • Problem: if specified path no longer valid due to link failure!
  - Example:
    • IP’s loose/strict source route option (you’ll see later)
    • virtual circuit setup phase in ATM (or MPLS)
Routing Algorithms/Protocols

Issues Need to Be Addressed:

• Route selection may depend on different criteria
  - Performance: choose route with smallest delay
  - Policy: choose a route that doesn’t cross .gov network

• Adapt to changes in network topology or condition
  - Self-healing: little or no human intervention

• Scalability
  - Must be able to support large number of hosts, routers
Centralized vs. Distributed Routing Algorithms

Centralized:
• A centralized route server collects routing information and network topology, makes route selection decisions, then distributes them to routers

Distributed:
• Routers cooperate using a distributed protocol
  - to create mutually consistent routing tables
• Two standard distributed routing algorithms
  - Link State (LS) routing
  - Distance Vector (DV) routing
Link State vs Distance Vector

• Both assume that
  - The address of each neighbor is known
  - The cost of reaching each neighbor is known

• Both find **global information**
  - By exchanging routing info among neighbors

• Differ in info exchanged and route computation
  - LS: tells every other node its distance to neighbors
  - DV: tells neighbors its distance to every other node
Link State Algorithm

• Basic idea: Distribute to all routers
  - Topology of the network
    • Cost of each link in the network

• Each router independently computes optimal paths
  - From itself to every destination
  - Routes are guaranteed to be loop free if
    • Each router sees the same cost for each link
    • Uses the same algorithm to compute the best path
Topology Dissemination

- Each router creates a set of link state packets (LSPs)
  - Describing its links to neighbors
  - LSP contains
    - Router id, neighbor’s id, and cost to its neighbor
- Copies of LSPs are distributed to all routers
  - Using controlled flooding
- Each router maintains a topology database
  - Database containing all LSPs
Topology Database: Example

link state database
Constructing Routing Table: Dijkstra’s Algorithm

• **Given the network topology**
  - How to compute *shortest* path to each destination?

• **Some notation**
  - **X**: source node
  - **N**: set of nodes to which shortest paths are known so far
    - N is initially empty
  - **D(V)**: cost of *known* shortest path from source X
  - **C(U,V)**: cost of link U to V
    - C(U,V) = \(\infty\) if not neighbors
Algorithm (at Node X)

• **Initialization**
  - \( N = \{X\} \)
  - For all nodes \( V \)
    • If \( V \) adjacent to \( X \), \( D(V) = C(X,V) \) else \( D(V) = \infty \)

• **Loop**
  - Find \( U \) not in \( N \) such that \( D(U) \) is smallest
  - Add \( U \) into set \( N \)
  - Update \( D(V) \) for all \( V \) not in \( N \)
    • \( D(V) = \min\{D(V), D(U) + C(U,V)\} \)
  - Until all nodes in \( N \)
# Dijkstra’s Algorithm: Example

<table>
<thead>
<tr>
<th>Step</th>
<th>start N</th>
<th>D(B),p(B)</th>
<th>D(C),p(C)</th>
<th>D(D),p(D)</th>
<th>D(E),p(E)</th>
<th>D(F),p(F)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>A</td>
<td>2,A</td>
<td>5,A</td>
<td>1,A</td>
<td>infinity</td>
<td>infinity</td>
</tr>
<tr>
<td>1</td>
<td>AD</td>
<td>2,A</td>
<td>4,D</td>
<td>2,D</td>
<td>infinity</td>
<td>infinity</td>
</tr>
<tr>
<td>2</td>
<td>ADE</td>
<td>2,A</td>
<td>3,E</td>
<td></td>
<td>4,E</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>ADEB</td>
<td></td>
<td>3,E</td>
<td></td>
<td>4,E</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>ADEBC</td>
<td></td>
<td></td>
<td></td>
<td>4,E</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>ADEBCF</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

![Graph Diagram](image-url)
Routing Table Computation

<table>
<thead>
<tr>
<th>dest</th>
<th>next</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>B</td>
</tr>
<tr>
<td>C</td>
<td>D</td>
</tr>
<tr>
<td>D</td>
<td>D</td>
</tr>
<tr>
<td>E</td>
<td>D</td>
</tr>
<tr>
<td>F</td>
<td>D</td>
</tr>
</tbody>
</table>
Distance Vector Routing

• A router tells neighbors its distance to every router
  - Communication between neighbors only
• Based on Bellman-Ford algorithm
  - Computes “shortest paths”
• Each router maintains a distance table
  - A row for each possible destination
  - A column for each neighbor
    • $D^X(Y,Z)$: distance from $X$ to $Y$ via $Z$
• Exchanges distance vector with neighbors
  - Distance vector: current least cost to each destination
Distance Table: Example

cost to destination via

<table>
<thead>
<tr>
<th></th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>(</td>
<td>()</td>
</tr>
<tr>
<td>D</td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>1</td>
<td>14</td>
</tr>
<tr>
<td>B</td>
<td>7</td>
<td>8</td>
</tr>
<tr>
<td>C</td>
<td>6</td>
<td>9</td>
</tr>
<tr>
<td>D</td>
<td>4</td>
<td>11</td>
</tr>
</tbody>
</table>
## Distance Table to Routing Table

<table>
<thead>
<tr>
<th>Destination</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>1</td>
<td>14</td>
<td>5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>7</td>
<td>8</td>
<td>5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>6</td>
<td>9</td>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>4</td>
<td>11</td>
<td>2</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Distance table**

<table>
<thead>
<tr>
<th>Destination</th>
<th>Outgoing link to use, cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>A,1</td>
</tr>
<tr>
<td>B</td>
<td>D,5</td>
</tr>
<tr>
<td>C</td>
<td>D,4</td>
</tr>
<tr>
<td>D</td>
<td>D,2</td>
</tr>
</tbody>
</table>

**Routing table**

- Distance table
- Routing table
Distance Vector Routing Algorithm

iterative:
• continues until no nodes exchange info.
• self-terminating: no "signal" to stop

asynchronous:
• nodes need not exchange info/iterate in lock step!

distributed:
• each node talks only with directly-attached neighbors

Distance Table data structure
• each node has its own row for each possible destination
• column for each directly-attached neighbor to node
• example: in node X, for dest. Y via neighbor Z:

$$D (X, Z) = \text{distance from } X \text{ to } Y, \text{ via } Z \text{ as next hop}$$

$$= c(X, Z) + \min_{w} \{D (Y,w)\}$$
Distance Vector Routing: Overview

Iterative, asynchronous:
  each iteration caused by:
  • local link cost change
  • message from neighbor: its least cost path change from neighbor

Distributed:
  • each node notifies neighbors only when its least cost path to any destination changes
    - neighbors then notify their neighbors if necessary

Each node:
  \( \text{wait for (change in local link cost or msg from neighbor)} \)
  \( \text{recompute distance table} \)
  if least cost path to any dest has changed, \( \text{notify neighbors} \)
**Distance Vector Algorithm: Example**

\[
D^X(Y,Z) = c(X,Z) + \min_w \{D^Z(Y,w)\}
\]

\[
= 7 + 1 = 8
\]

\[
D^X(Z,Y) = c(X,Y) + \min_w \{D^Y(Z,w)\}
\]

\[
= 2 + 1 = 3
\]
Distance Vector Algorithm: Example

![Distance Vector Algorithm Diagram]

- **Nodes**: X, Y, Z
- **Connections**: X to Y (2), X to Z (7), Y to Z (1)
- **Costs**:
  - X to Y: 2
  - X to Z: 7
  - Y to Z: 1

The algorithm updates the cost via neighboring nodes based on the received information.
Convergence of DV Routing

- router detects local link cost change
- updates distance table
- if cost change in least cost path, notify neighbors

“good news travels fast”

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>4</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>D'</td>
<td>X</td>
<td>Z</td>
<td></td>
</tr>
<tr>
<td>Y</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>1</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>D'</td>
<td>X</td>
<td>Z</td>
<td></td>
</tr>
<tr>
<td>Y</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>1</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>D'</td>
<td>X</td>
<td>Z</td>
<td></td>
</tr>
<tr>
<td>Y</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

\[ c(X,Y) \] change

t_0 \quad t_1 \quad t_2

algorithm terminates
Problems with DV Routing

Link cost changes:
  - good news travels fast
  - bad news travels slow
    - “count to infinity” problem!

![Diagram showing routing changes over time with link costs updating]
Count-to-Infinity Problem

X \quad 1 \quad Y \quad 1 \quad Z

\quad 2
“Fixes” to Count-to-Infinity Problem

- **Split horizon**
  - A router never advertises the cost of a destination to a neighbor
    - If this neighbor is the next hop to that destination

- **Split horizon with poisonous reverse**
  - If X routes traffic to Z via Y, then
    - X tells Y that its distance to Z is infinity
      - Instead of not telling anything at all
  - Accelerates convergence
Split Horizon with 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)

![Diagram showing the process of split horizon with poisoned reverse]
Count-to-Infinity Problem Revisited

X -> Y -> Z

W
Link State vs Distance Vector

- Tells everyone about neighbors
- Controlled flooding to exchange link state
- Dijkstra’s algorithm
- Each router computes its own table
- May have oscillations
- Open Shortest Path First (OSPF)

- Tells neighbors about everyone
- Exchanges distance vectors with neighbors
- Bellman-Ford algorithm
- Each router’s table is used by others
- May have routing loops
- Routing Information Protocol (RIP)
Routing in the Real World

Our routing study thus far - idealization
- all routers identical
- network "flat"

How to do routing in the Internet
- scalability and policy issues

**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
Routing in the Internet

• The Global Internet consists of Autonomous Systems (AS) interconnected with each other:
  - Stub AS: small corporation: one connection to other AS’s
  - Multihomed AS: large corporation (no transit): multiple connections to other AS’s
  - Transit AS: provider, hooking many AS’s together

• Two-level routing:
  - Intra-AS: administrator responsible for choice of routing algorithm within network
  - Inter-AS: unique standard for inter-AS routing: BGP
Internet Architecture

Internet: “networks of networks”!

Diagram:
- NAP
- Internic
- National network
- Regional network
- On-line services
- ISP
- Company
- University
- LANs
- Access via modem

Key terms:
- ISP
- Company
- University
- National network
- Regional network
- International lines
Number of Used ASNs

Source: Geoff Huston, http://bgp.potaroo.net
Growth of Destination Net Prefixes
(measured by # of BGP routes)

Internet AS Hierarchy

Intra-AS border (exterior gateway) routers

Inter-AS interior (gateway) routers
Intra-AS vs. Inter-AS Routing

Intra-AS routing within AS A

Inter-AS routing between A and B

Intra-AS routing within AS B

Host h1

Host h2
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, update traffic

Performance:
• Intra-AS: can focus on performance
• Inter-AS: policy may dominate over performance
Intra-AS and Inter-AS Routing

“Gateways”:
• perform inter-AS routing amongst themselves
• perform intra-AS routers with other routers in their AS

inter-AS, intra-AS routing in gateway A.c

Network Layer: Part II
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
  - IS-IS: Intermediate System to Intermediate System (OSI Standard)
  - EIGRP: Extended Interior Gateway Routing Protocol (Cisco proprietary)
RIP (Routing Information Protocol)

- Distance vector algorithm
- Included in BSD-UNIX Distribution in 1982
- Distance metric: # of hops (max = 15 hops)
  - Can you guess why?

- Distance vectors: exchanged among neighbors every 30 sec via Response Message (also called advertisement)
- Each advertisement: list of up to 25 destination nets within AS
RIP: Link Failure and Recovery

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
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 (“Type-of-Services”)
  - e.g., satellite link cost set “low” for best effort; high for real time)
- **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.
Inter-AS Routing in the Internet: BGP

Figure 4.5.2-new2: BGP use for inter-domain routing
BGP (Border Gateway Protocol)

- The de facto standard (BGP-4)
- **Path Vector** protocol:
  - similar to Distance Vector protocol
  - each Border Gateway broadcast to neighbors (peers) entire path (i.e., sequence of AS’s) to destination
  - BGP routes to networks (ASs), not individual hosts
- E.g., Gateway X may announce to its neighbors it “knows” a (AS) path to a destination network, Z, via a series of ASs:
  \[ \text{Path } (X, Z) = X, Y_1, Y_2, Y_3, \ldots, Z \]
- BGP border gateways referred to as BGP speakers
BGP Operations: Policy Routing

Q: What does a BGP border gateway do?

• Receiving and *filtering* route advertisements from directly attached neighbor(s)
  - To accept or not accept route advertisements depends on policies (e.g., whether you “trust” your neighbors)

• Route selection.
  - to route to destination X, which path (of several advertised) will be taken?
  - route selection based on policies (e.g., always prefer route advertisement from “good old” neighbor Y)

• Sending route advertisements to neighbors
  - what/whether to advertise to your neighbors also depends on policies (e.g., don’t tell your neighbor Z that you know a route to destination X)
Customers and Providers

Customer pays provider for access to the Internet

Customer pays provider for access to the Internet
The Peering Relationship

Peers provide transit between their respective customers

Peers do not provide transit between peers

Peers (often) do not exchange $$$

peer ⬇️ peer

provider ⬇️ customer

traffic allowed ➡️ traffic NOT allowed
Peering also allows connectivity between the customers of “Tier 1” providers.
U of Minnesota Neighborhood

- AS 1
  - Genuity
- AS 7018
  - AT&T
- AS 3908
  - SuperNet (Qwest)
- AS 57
  - UMN
  - GigaPoP
- AS 1998
  - State of Minnesota
- AS 217
  - UMN

128.101.0.0/16
BGP Messages

- BGP messages exchanged using TCP.
- BGP messages:
  - OPEN: opens TCP connection to peer and authenticates sender
  - KEEPALIVE keeps connection alive in absence of UPDATES; also ACKs OPEN request
    - OPEN/KEEPALIVE establish & maintain BGP neighbor relation
  - UPDATE: advertises new path (or withdraws old)
  - NOTIFICATION: reports errors in previous msg; also used to close connection
BGP Example

• Speaker for AS2 advertises reachability to P and Q
  - network 128.96/16, 192.4.153/24, 192.4.32/24, and 192.4.3/24, can be reached directly from AS2

  \[
  \text{Backbone network (AS 1)} \rightarrow \text{Regional provider A (AS 2)} \rightarrow \text{Customer P (AS 4)} 128.96/16 192.4.153/24 \\
  \rightarrow \text{Customer Q (AS 5)} 192.4.32/24 192.4.3/24 \\
  \rightarrow \text{Regional provider B (AS 3)} \rightarrow \text{Customer R (AS 6)} 192.12.69/24 \\
  \rightarrow \text{Customer S (AS 7)} 192.4.54/24 192.4.23/24
  \]

• Speaker for backbone advertises
  - networks 128.96/16, 192.4.153/24, 192.4.32/24, and 192.4.3/24 can be reached along the path (AS1, AS2).

• Speaker can cancel previously advertised paths (by sending withdrawal messages)
**BGP: AS Path Advertisement**

**Suppose:** gateway X send its path to peer gateway W

- W may or may not select path offered by X
  - cost, policy (don’t route via competitors AS), loop prevention reasons.
- If W selects path advertised by X, then:
  \[ \text{Path (W,Z) = W, Path (X,Z)} \]
- Note: X can control incoming traffic by controlling its route advertisements to peers:
  - e.g., don’t want to route traffic to Z \(\rightarrow\) don’t advertise any routes to Z
# BGP Attributes

<table>
<thead>
<tr>
<th>Value</th>
<th>Code</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>ORIGIN</td>
<td>[RFC1771]</td>
</tr>
<tr>
<td>2</td>
<td>AS_PATH</td>
<td>[RFC1771]</td>
</tr>
<tr>
<td>3</td>
<td>NEXT_HOP</td>
<td>[RFC1771]</td>
</tr>
<tr>
<td>4</td>
<td>MULTI_EXIT_DISC</td>
<td>[RFC1771]</td>
</tr>
<tr>
<td>5</td>
<td>LOCAL_PREF</td>
<td>[RFC1771]</td>
</tr>
<tr>
<td>6</td>
<td>ATOMIC_AGGREGATE</td>
<td>[RFC1771]</td>
</tr>
<tr>
<td>7</td>
<td>AGGREGATOR</td>
<td>[RFC1771]</td>
</tr>
<tr>
<td>8</td>
<td>COMMUNITY</td>
<td>[RFC1997]</td>
</tr>
<tr>
<td>9</td>
<td>ORIGINATOR_ID</td>
<td>[RFC2796]</td>
</tr>
<tr>
<td>10</td>
<td>CLUSTER_LIST</td>
<td>[RFC2796]</td>
</tr>
<tr>
<td>11</td>
<td>DPA</td>
<td>[Chen]</td>
</tr>
<tr>
<td>12</td>
<td>ADVERTISER</td>
<td>[RFC1863]</td>
</tr>
<tr>
<td>13</td>
<td>RCID_PATH / CLUSTER_ID</td>
<td>[RFC1863]</td>
</tr>
<tr>
<td>14</td>
<td>MP_REACH_NLRI</td>
<td>[RFC2283]</td>
</tr>
<tr>
<td>15</td>
<td>MP_UNREACH_NLRI</td>
<td>[RFC2283]</td>
</tr>
<tr>
<td>16</td>
<td>EXTENDED COMMUNITIES</td>
<td>[Rosen]</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
</tr>
<tr>
<td>255</td>
<td>reserved for development</td>
<td></td>
</tr>
</tbody>
</table>

From IANA: [http://www.iana.org/assignments/bgp-parameters](http://www.iana.org/assignments/bgp-parameters)

Most important attributes

Not all attributes need to be present in every announcement.
How to detect loop using AS path?
BGP Route Processing

Receive BGP Updates

Apply Import Policies

Best Route Selection

Best Routes

Based on Attribute Values

Apply Policy = filter routes & tweak attributes

Install forwarding Entries for best Routes.

IP Forwarding Table

Apply Export Policies

Best Route Table

Transmit BGP Updates

Open ended programming. Constrained only by vendor configuration language

Apply Policy = filter routes & tweak attributes

Apply Policy = filter routes & tweak attributes
Tweak Tweak Tweak Tweak

• For **inbound traffic**
  - Filter outbound routes
  - Tweak attributes on **outbound** routes in the hope of influencing your neighbor’s best route selection

• For **outbound traffic**
  - Filter **inbound** routes
  - Tweak attributes on **inbound** routes to influence best route selection

In general, an AS has more control over outbound traffic
BGP: Controlling Who Routes to You

Figure 4.5-BGPnew: a simple BGP scenario

- A, B, C are provider networks
- X, W, Y are customer (of provider networks)
- X is dual-homed: attached to two networks
  - C tells X networks belonging to C, i.e., a route to them via C
  - X does not want to carry traffic from B via X to C
  - .. so X will not advertise to B any route to networks in C learned from C
BGP: Controlling Who Routes to You

• A advertises to B the path AW
• B advertises to X the path BAW
• Should B advertise to C the path BAW?
  - 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!
Network Layer (Unicast) Routing Summary

• Network Layer Routing
  - Basic Issues
  - Distributed Routing Algorithms: LS vs. DV
  - Link State (LS): How does it work?
  - Distance Vector (DV): How does it work? Issues?

• Routing in the Internet
  - Intra-AS vs. Inter-AS routing
  - Intra-AS: RIP and OSPF
  - Inter-AS: BGP and Policy Routing
Routing & Forwarding: Logical View of a Router
IP Forwarding & IP/ICMP Protocol

- **Routing protocols**: path selection, RIP, OSPF, BGP
- **IP protocol**: addressing conventions, packet handling conventions
- **ICMP protocol**: error reporting, router “signaling”

Transport layer: TCP, UDP

Network layer

Data Link layer (Ethernet, WiFi, PPP, …)

Physical Layer (SONET, …)
Broadcast Routing

- Deliver packets from source to all other nodes
- Source duplication is inefficient:

  - Source duplication: how does source determine recipient addresses?

![Diagram showing broadcast routing with source duplication and in-network duplication]

- Source duplication: how does source determine recipient addresses?
In-Network Duplication

- Flooding: when node receives broadcast packet, sends copy to all neighbors
  - Problems: cycles & broadcast storm
- Controlled flooding: node only broadcasts packet if it hasn’t broadcast 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
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
Approaches for Building Multicast 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

- multicast 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:

  **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

S: source

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

• 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
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
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 showing tree structure with routers connected by lines]

**LEGEND**
- Router with attached group member
- Router with no attached group member
- Path order in which join messages generated
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 (cont’d)

• **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
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 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

Diagram:
- R1 to R7
- join messages
- all data multicast from rendezvous point
**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!”
Network Layer Summary

- Network Layer Functions and Service Models
  - Addressing, Routing and Forwarding
  - Virtual Circuit vs. Datagram

- IP Addressing Scheme: CIDR
  - DHCP

- IP Forwarding and IP Protocol
  - IP Datagram Forwarding Model: dest. in same net vs. diff. net
  - IP and ICMP: Datagram Format, IP Fragmentation, ...

- Network Layer Routing
  - Fundamental Issues
  - Two Basic Distributed Algorithms: Link State and Distance Vector
  - Routing in the Internet: Intra-AS vs. Inter-AS routing
    - Intra-AS: RIP and OSPF
    - Inter-AS: BGP and Policy Routing

- IP Broadcast and Multicast