Chapter 4: Roadmap

4.1 Introduction
4.2 Virtual circuit and datagram networks
4.3 What’s inside a router
4.4 IP: Internet Protocol
4.5 Routing algorithms
  - Link state
  - Distance Vector
  - Hierarchical routing
4.6 Routing in the Internet
4.7 Broadcast and multicast routing
Distance Vector Algorithm

- Two metrics known to each node $x$
  - Estimate $D_x(y)$ of least cost from $x$ to $y$
  - Link cost $c(x,v)$ to reach $x$’s immediate neighbors
- Each node maintains a distance vector:
  $$\vec{D}_x = \{D_x(y) : y \in V\}$$
- Node $x$ periodically asks its neighbors for their distance vectors
  - Thus, $x$ has access to the following for each neighbor $v$
  $$\vec{D}_v = \{D_v(y) : y \in V\}$$
**Distance Vector Algorithm (cont’d)**

**Basic idea (Bellman-Ford):**

- When a node $x$ receives new DV estimate from neighbor $v$, it updates its own DV using the Bellman-Ford equation:

  \[
  D_x(y) \leftarrow \min\{D_x(y), c(x,v) + D_v(y)\}, \forall \ y \in V
  \]

- Centralized Bellman Ford requires $O(|V| \cdot |E|)$ time
  - Dijkstra’s algorithm was $O(|V| \cdot \log|V|)$
  - Convergence of decentralized version depends on topology, link weights, update delays, and timing of events

- Bellman Ford allows negative weights
Distance Vector Algorithm (cont’d)

Iterative, asynchronous

Each 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
### node x table

<table>
<thead>
<tr>
<th>From</th>
<th>Cost to</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>0 2 7</td>
</tr>
<tr>
<td>y</td>
<td>∞ ∞ ∞</td>
</tr>
<tr>
<td>z</td>
<td>∞ ∞ ∞</td>
</tr>
</tbody>
</table>

### node y table

<table>
<thead>
<tr>
<th>From</th>
<th>Cost to</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>0 2 3</td>
</tr>
<tr>
<td>y</td>
<td>2 0 1</td>
</tr>
<tr>
<td>z</td>
<td>7 1 0</td>
</tr>
</tbody>
</table>

### node z table

<table>
<thead>
<tr>
<th>From</th>
<th>Cost to</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>0 2 7</td>
</tr>
<tr>
<td>y</td>
<td>2 0 1</td>
</tr>
<tr>
<td>z</td>
<td>3 1 0</td>
</tr>
</tbody>
</table>

---

**Graph with Costs**

- From x to y: 2
- From y to z: 1
- From z to x: 7

---

**Table of Costs**

<table>
<thead>
<tr>
<th>From</th>
<th>Cost to</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>0 2 3</td>
</tr>
<tr>
<td>y</td>
<td>2 0 1</td>
</tr>
<tr>
<td>z</td>
<td>3 1 0</td>
</tr>
</tbody>
</table>

---

**Legend**

- x, y, z: Nodes
- 2, 1, 7: Costs
- Time: Sequence of events

---

**Graph Details**

- Nodes x, y, z are connected.
- Costs associated with each edge indicate the time taken for movement or event occurrence.
Distance Vector: Link Cost Changes

Link cost changes:
- Node detects local link cost change
- Recalculates distance vector, updates routing info if needed
- If DV changes, notifies neighbors

"good news travels fast"
- Node $y$ detects link-cost change, updates its distance to $x$, and informs its neighbors
- Node $z$ receives $y$'s message and updates its table; computes a new least-cost to $x$ and sends its DV to $x$ and $y$
- Finally, node $y$ receives $z$'s vector and updates its distance table; $y$'s least costs do not change and hence $y$ does not send any messages after that
**Distance Vector: Link Cost Changes**

Link cost changes:
- Good news travels fast
- Bad news travels slow – “count to infinity” problem!
- 46 iterations before algorithm stabilizes

Poisoned reverse (“split horizon”):
- If $z$ routes through $y$ to get to $x$:
  - $z$ tells $y$ that 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
  - Depends on convergence time

Time to Convergence

- **LS**: $O(n \log n)$ algorithm + delay to send $O(nE)$ msgs
  - Oscillations (cost = congestion)
- **DV**: convergence time varies
  - May have routing loops
  - Count-to-infinity problem

Robustness: what happens if router malfunctions?

- **LS**:
  - Node can advertise incorrect *link* cost
  - Affects only a small portion of the graph

- **DV**:
  - DV node can advertise incorrect *path* cost
  - Each node’s table used by others
  - Errors propagate thru network
Chapter 4: Roadmap

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

Problems in practice:

- **Memory**: can’t store paths to all destinations in a routing table (several billion links)
- **CPU time**: can’t overload routers with such huge computational expense
- **Message overhead**: routing table exchanges would overload links

- **Competitiveness**: ISPs not willing to share their topology with others

Solution: administrative autonomy

- Internet = network of networks
- Network admins control routing in their own networks, export reachable subnets to outside world
Hierarchical Routing

• Aggregate routers into regions called **AS** (Autonomous Systems)

• Routers in the same AS run the same algorithm
  - Accomplished via **intra-AS** routing protocols

• **ISPs** gain flexibility
  - Routers in different ASes can run different intra-AS protocols that cannot directly speak to each other, which is OK

Gateway routers

• Direct links to routers in other ASes

• Exchange routing view of each AS using an **inter-AS** protocol
  - Summary of subnets to which this AS is willing to route

Texas A&M owns AS3794 with two subnets: 128.194/16 and 165.91/16
**Interconnected ASes**

- Intra-AS sets entries for all internal dests
  - E.g., 1a plots shortest path to 1b using link-state alg
- Inter-AS accepts external dests from neighbor ASes
  - E.g., 1b learns 128.194/16 is reachable via AS2
- Inter-AS broadcasts pairs (subnet, exit router)
  - E.g., 1b notifies all routers in AS1 that it can reach 128.194/16

Terminology: exit router = border router = gateway
Example: Choosing Among Multiple ASes

- Now suppose AS1 learns from the inter-AS protocol that 128.194/16 is reachable from AS3 and from AS2
  - To configure forwarding table, routers in AS1 must determine towards which exit (1c or 1b) they must forward packets
- This is also the job of inter-AS routing protocol!
  - Usually based on ISP policy, SLAs, prior traffic engineering
- **Hot potato routing:** send packet towards closest of two exit points (other options discussed later)

Learn from inter-AS protocol that subnet \( x \) is reachable via multiple gateways

Use routing info from intra-AS protocol to determine least-cost paths to each of the gateways

Hot potato routing: choose the gateway that has the smallest cost

Determine from forwarding table the interface \( l \) that leads to least-cost gateway. Enter \( (x, l) \) in forwarding table
Chapter 4: Roadmap

4.1 Introduction
4.2 Virtual circuit and datagram networks
4.3 What’s inside a router
4.4 IP: Internet Protocol
4.5 Routing algorithms
4.6 Routing in the Internet
   - RIP
   - OSPF
   - BGP
4.7 Broadcast and multicast routing
Intra-AS Routing

• Also known as Interior Gateway Protocols (IGP)
• Common Intra-AS routing protocols:
  – RIP: Routing Information Protocol (DV)
  – OSPF: Open Shortest Path First (LS)
  – IGRP: Interior Gateway Routing Protocol (Cisco proprietary, DV, now obsolete); EIGRP (Extended IGRP)
Chapter 4: Roadmap

4.1 Introduction
4.2 Virtual circuit and datagram networks
4.3 What’s inside a router
4.4 IP: Internet Protocol
4.5 Routing algorithms
4.6 Routing in the Internet
   - RIP
   - OSPF
   - BGP
4.7 Broadcast and multicast routing
RIP (Routing Information Protocol)

- Included in BSD-UNIX distribution in 1982
  - Distance vector algorithm
- Distance metric: # of hops (max = 15)
  - Distance vectors: exchanged among neighbors every 30 sec using advertisement messages
  - Each message: lists of up to 25 destination nets within AS

```
<table>
<thead>
<tr>
<th>Destination Subnet</th>
<th>Hops from A</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: Example

### Routing Table in D

<table>
<thead>
<tr>
<th>Destination Network</th>
<th>Next Router</th>
<th>Number 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>

Diagram:
- **A**: Router A connected to **w** and **x**
- **B**: Router B connected to **y** and **z**
- **C**: Router C connected to **x**
- **D**: Router D connected to **x**, **y**, and **z**
RIP: Example

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

Advertisement from A to D

Routing table in D

<table>
<thead>
<tr>
<th>Destination Network</th>
<th>Next Router</th>
<th>Number 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 A</td>
<td>5</td>
</tr>
<tr>
<td>x</td>
<td>--</td>
<td>1</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
**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 propagates to entire network

- That’s why it is important to assign high priority to packets from routing protocols at ISP routers
  - QoS only applies to specialized packets generated by ISP

- RIP uses poisoned reverse to prevent loops (infinite distance = 16 hops)
RIP Table Processing

- RIP routing tables managed by an application-level process called *routed* (daemon)
- Advertisements sent in UDP packets (port 520)

Note: named, smtpd, etc. are Unix daemons (services)
Chapter 4: Roadmap

4.1 Introduction
4.2 Virtual circuit and datagram networks
4.3 What’s inside a router
4.4 IP: Internet Protocol
4.5 Routing algorithms
4.6 Routing in the Internet
   - RIP
   - OSPF
   - BGP
4.7 Broadcast and multicast routing
OSPF (Open Shortest Path First)

- “Open”: protocol specifications publicly available
- Uses Link State (LS) algorithm
  - LS packet dissemination
  - Topology map at each node
  - Route computation using Dijkstra’s algorithm
- Advertisements disseminated to entire AS (via flooding)
  - Carried in OSPF messages directly over IP (rather than TCP or UDP) using protocol number 89
  - Layer 3.5 similar to ICMP
  - Handles own error detection/correction
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)
- **Integrated uni- and multicast** support:
  - Multicast OSPF (MOSPF) uses same topology database 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 a detailed topology for the area it belongs to and shortest paths to all destinations therein
- Area border routers: “summarize” distances to networks in their own area, advertise to other area border routers
- Backbone routers: run OSPF routing limited to the backbone
- Boundary routers: connect to other AS’s