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**

Interplay between routing, forwarding

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

value in arriving packet’s header
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

$\cdot c(x,x') = \text{cost of link } (x,x')$
- e.g., $c(w,z) = 5$

$\cdot \text{cost could always be 1, or inversely related to bandwidth, or inversely related to congestion}$

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

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
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>∞</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>uxy</td>
<td>2,u</td>
<td>3,y</td>
<td>4,y</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>uxyv</td>
<td>3,y</td>
<td></td>
<td>4,y</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>uxyvw</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>uxyvwz</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Resulting shortest-path tree from u:

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

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
**Distance Vector Algorithm**

**Bellman-Ford Equation (dynamic programming)**

Define

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

Then

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

where \( \min \) is taken over all neighbors \( v \) of \( x \)

---

**Bellman-Ford example**

Clearly, \( d_v(z) = 5 \), \( d_x(z) = 3 \), \( d_w(z) = 3 \)

B-F equation says:

\[ d_u(z) = \min \{ c(u,v) + d_v(z), c(u,x) + d_x(z), c(u,w) + d_w(z) \} \]

\[ = \min \{ 2 + 5, 1 + 3, 5 + 3 \} = 4 \]

Node that achieves minimum is next hop in shortest path \( \rightarrow \) forwarding table
Distance Vector Algorithm

- $D_x(y) =$ estimate of least cost from $x$ 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 $D_v = [D_v(y): y \in N ]$

Distance vector algorithm (4)

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:
  $D_x(y) \leftarrow \min_v \{ c(x,v) + D_v(y) \}$ for each node $y \in N$
- Under minor, natural conditions, the estimate $D_x(y)$ converge to the actual least cost $d_x(y)$
**Distance Vector Algorithm (5)**

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

---

Node x table

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

Node y table

<table>
<thead>
<tr>
<th>Cost to</th>
<th>x</th>
<th>y</th>
<th>z</th>
</tr>
</thead>
<tbody>
<tr>
<td>From</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Node z table

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

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

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

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)
Example: Choosing among multiple ASes

- now suppose AS1 learns from inter-AS protocol that subnet $x$ is reachable from AS3 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!

```
Learn from inter-AS protocol that subnet $x$ is reachable via multiple gateways
Use routing info from intra-AS protocol to determine costs of least-cost paths to each of the gateways
Hot potato routing: Choose the gateway that has the smallest least cost
Determine from forwarding table the interface $I$ that leads to least-cost gateway. Enter $(x, I)$ in forwarding table
```

Network Layer 4-97