Chapter 4: outline

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

Routing algorithm determines end-end-path through network.
Forwarding table determines local forwarding at this router.

IP destination address in arriving packet’s header.

<table>
<thead>
<tr>
<th>dest address</th>
<th>output link</th>
</tr>
</thead>
<tbody>
<tr>
<td>address-range 1</td>
<td>3</td>
</tr>
<tr>
<td>address-range 2</td>
<td>2</td>
</tr>
<tr>
<td>address-range 3</td>
<td>2</td>
</tr>
<tr>
<td>address-range 4</td>
<td>1</td>
</tr>
</tbody>
</table>
Graph abstraction

Graph: \( G = (N,E) \)

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

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

*aside:* graph abstraction is useful in other network contexts, e.g., 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 related to congestion

cost of path (x_1, x_2, x_3, ..., x_p) = c(x_1, x_2) + c(x_2, x_3) + ... + c(x_{p-1}, x_p)

**key question:** what is the least-cost path between u and z?

**routing algorithm:** algorithm that finds that least cost path
Routing algorithm classification

Q: 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

Q: static or dynamic?

static:
- routes change slowly over time

dynamic:
- routes change more quickly
  - periodic update
  - in response to link cost changes
Chapter 4: outline

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
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 \[ \text{if } v \text{ adjacent to } u \]
5 \[ \text{then } D(v) = c(u,v) \]
6 \[ \text{else } D(v) = \infty \]
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)</th>
<th>D(w)</th>
<th>D(x)</th>
<th>D(y)</th>
<th>D(z)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>p(v)</td>
<td>p(w)</td>
<td>p(x)</td>
<td>p(y)</td>
<td>p(z)</td>
</tr>
<tr>
<td>0</td>
<td>u</td>
<td>7,u</td>
<td>3,u</td>
<td>5,u</td>
<td>∞</td>
<td>∞</td>
</tr>
<tr>
<td>1</td>
<td>uw</td>
<td>6,w</td>
<td>5,u</td>
<td>11,w</td>
<td>∞</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>uwx</td>
<td>6,w</td>
<td>11,w</td>
<td>14,x</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>uwxv</td>
<td>6,w</td>
<td>10,v</td>
<td>14,x</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>uwxvy</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>uwxvzy</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Notes:**
- construct shortest path tree by tracing predecessor nodes
- ties can exist (can be broken arbitrarily)
**Dijkstra’s algorithm: another 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></td>
<td>3,y</td>
<td>4,y</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>uxyvw</td>
<td></td>
<td></td>
<td>4,y</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>uxyvww</td>
<td></td>
<td></td>
<td></td>
<td>4,y</td>
<td></td>
</tr>
</tbody>
</table>

![Diagram of a network with nodes u, v, w, x, y, and z showing the step-by-step application of Dijkstra's algorithm.](Network Layer 4-10)
Dijkstra’s algorithm: example (2)

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>
Chapter 4: outline

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)

let
\[ d_x(y) := \text{cost of least-cost path from } x \text{ to } y \]
then
\[ d_x(y) = \min_v \{ c(x,v) + d_v(y) \} \]

- cost from neighbor \( v \) to destination \( y \)
- cost to neighbor \( v \)

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

B-F equation says:

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

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

node achieving minimum is next hop in shortest path, used in forwarding table
Distance vector algorithm

- $D_x(y) = \text{estimate of least cost from } x \text{ to } y$
  - $x$ maintains distance vector $D_x = [D_x(y): y \in N]$

- node $x$:
  - knows cost to each neighbor $v$: $c(x,v)$
  - maintains its neighbors’ distance vectors. For each neighbor $v$, $x$ maintains $D_v = [D_v(y): y \in N]$
key idea:

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

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

- under minor, natural conditions, the estimate \( D_x(y) \) converge to the actual least cost \( d_x(y) \)
Distance vector algorithm

**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:**

1. wait for (change in local link cost or msg from neighbor)
2. recompute estimates
3. if DV to any dest has changed, *notify* neighbors
\[ D_x(y) = \min\{c(x,y) + D_y(y), c(x,z) + D_z(y)\} = \min\{2+0, 7+1\} = 2 \]

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

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

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 600 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:**
- at "edge" of its own AS
- has 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 which gateway it should forward packets towards for dest x
  - this is also job of inter-AS routing protocol!
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!
- *hot potato routing:* send packet towards closest of two routers.

```
<table>
<thead>
<tr>
<th>learn from inter-AS protocol that subnet ( x ) is reachable via multiple gateways</th>
</tr>
</thead>
<tbody>
<tr>
<td>use routing info from intra-AS protocol to determine costs of least-cost paths to each of the gateways</td>
</tr>
<tr>
<td>hot potato routing: choose the gateway that has the smallest least cost</td>
</tr>
<tr>
<td>determine from forwarding table the interface ( I ) that leads to least-cost gateway. Enter ( (x,I) ) in forwarding table</td>
</tr>
</tbody>
</table>
```