Routing

Mike Freedman
COS 461: Computer Networks

http://www.cs.princeton.edu/courses/archive/spr20/cos461/

Routing: Mapping Link to Path

Data and Control Planes

Routing vs. Forwarding

• Routing: control plane
  — Computing paths the packets will follow
  — Routers talking amongst themselves
  — Creating the forwarding tables

• Forwarding: data plane
  — Directing a data packet to an outgoing link
  — Using the forwarding tables
Three Issues to Address

• What does the protocol compute?
  — E.g., shortest paths

• What algorithm does the protocol run?
  — E.g., link-state routing

• How do routers learn end-host locations?
  — E.g., injecting into the routing protocol

What Does the Protocol Compute?

Different Types of Paths

• Static model
  — What is computed, not how computation performed

• Trade-offs
  — State to represent the paths
  — Efficiency of the paths
  — Ability to support multiple paths
  — Complexity of path computation

Spanning Tree

• One tree that reaches every node
  — Single path between each pair of nodes
  — No loops, so can support broadcast easily
  — But, paths are long, and some links not used
Shortest Paths

- Shortest path(s) between pairs of nodes
  - A shortest-path tree rooted at each node
  - Min hop count or min sum of edge weights
  - Multipath routing is limited to Equal Cost MultiPath

Local Policy at Each Hop

- Locally best path
  - Local policy: each node picks the path it likes best
  - ... among the paths chosen by its neighbors

End-to-End Path Selection

- End-to-end path selection
  - Each node picks its own end to end paths
  - ... independent of what other paths other nodes use
  - More state and complexity in the nodes
How to Compute Paths?

Spanning Tree Algorithm

• Elect a root
  – The switch with the smallest identifier
  – And form a tree from there

• Algorithm
  – Repeatedly talk to neighbors
    • “I think node Y is the root”
    • “My distance from Y is d”
  – Update based on neighbors
    • Smaller id as the root
    • Smaller distance d+1

Spanning Tree Example: Switch #4

• Switch #4 thinks it is the root
  – Sends (4, 0, 4) message to 2 and 7
  – Notation: (my root, my distance, my ID)

• Switch #4 hears from #2
  – Receives (2, 0, 2) message from 2
  – Thinks #2 is root and it’s one hop away

• Switch #4 hears from #7
  – Receives (2, 1, 7) from 7
  – But, this is a longer path, so 4 prefers 4-2 over 4-7-2
  – And removes 4-7 link from the tree

Shortest-Path Problem

• Compute: path costs to all nodes
  – From a given source u to all other nodes
  – Cost of the path through each outgoing link
  – Next hop along the least-cost path to s

Used in Ethernet LANs
Link State: Dijkstra’s Algorithm

- Flood the topology information to all nodes
- Each node computes shortest paths to other nodes

**Initialization**

\[
S = \{u\}
\]

for all nodes \( v \)

if \( v \) is adjacent to \( u \)

\[
D(v) = c(u,v)
\]

else \( D(v) = \infty \)

**Loop**

add \( w \) with smallest \( D(w) \) to \( S \)

update \( D(v) \) for all adjacent \( v \):

\[
D(v) = \min(D(v), D(w) + c(w,v))
\]

until all nodes are in \( S \)

Used in OSPF and IS-IS

Link-State Routing Example

Link-State Routing Example (cont.)

Link State: Shortest-Path Tree

- Shortest-path tree from \( u \)
- Forwarding table at \( u \)

<table>
<thead>
<tr>
<th>dest</th>
<th>link</th>
</tr>
</thead>
<tbody>
<tr>
<td>( v )</td>
<td>( (u,v) )</td>
</tr>
<tr>
<td>( w )</td>
<td>( (u,w) )</td>
</tr>
<tr>
<td>( x )</td>
<td>( (u,w) )</td>
</tr>
<tr>
<td>( y )</td>
<td>( (u,v) )</td>
</tr>
<tr>
<td>( z )</td>
<td>( (u,v) )</td>
</tr>
<tr>
<td>( s )</td>
<td>( (u,w) )</td>
</tr>
<tr>
<td>( t )</td>
<td>( (u,w) )</td>
</tr>
</tbody>
</table>
**Link State: Shortest-Path Tree**

Find shortest path \( t \) to \( v \)
- Forwarding table entry at \( t \)
  \( (Y) \ (t, x) \quad (M) \ (t, s) \)
- Distance from \( t \) to \( v \)
  \( (Y) \ 6 \quad (M) \ 7 \quad (C) \ 8 \quad (A) \ 9 \)
- Rounds to find shortest path
  \( (Y) \ 5 \quad (M) \ 6 \quad (C) \ 7 \quad (A) \ 8 \)

**Distance Vector: Bellman-Ford Algo**

- Define distances at each node \( x \)
  \(- \ d_x(y) = \text{cost of least-cost path from } x \text{ to } y \)
- Update distances based on neighbors
  \(- \ d_x(y) = \min \{ c(x, v) + d_v(y) \} \text{ over all neighbors } v \)

\[
\begin{align*}
  d_u(z) &= \min \{ c(u, v) + d_v(z), c(u, w) + d_w(z) \} \\
  \text{Used in RIP and EIGRP}
\end{align*}
\]

**Distance Vector Example**

\[
\begin{align*}
  d_u(z) &= 1 \\
  d_u(z) &= 4 \\
  d_u(z) &= \min \{ 2 + d_v(z), 1 + d_v(z) \} = 3
\end{align*}
\]
**Distance Vector Example (Cont.)**

\[
d_w(z) = \min \{ 1 + d_w(z), 4 + d_v(z), 2 + d_u(z) \}
\]

\[
d_u(z) = \min \{ 3 + d_u(z), 2 + d_w(z) \}
\]

= 5

**Path-Vector Routing**

- Extension of distance-vector routing
  - Support flexible routing policies

- Key idea: advertise the entire path
  - Distance vector: send *distance metric* per dest \( d \)
  - Path vector: send the *entire path* for each dest \( d \)

**Path-Vector: Flexible Policies**

- Each node can apply local policies
  - Path selection: Which path to use?
  - Path export: Which paths to advertise?

- Node 2 prefers “2, 3, 1” over “2, 1”
- Node 1 doesn’t let 3 hear the path “1, 2”
End-to-End Signaling

- Establish end-to-end path in advance
  - Learn the topology (as in link-state routing)
  - End host or router computes and signals a path
    - Signaling: install entry for each circuit at each hop
    - Forwarding: look up the circuit id in the table

Source Routing

- Similar to end-to-end signaling
  - But the data packet carries the hops in the path
- End-host control
  - Tell the end host the topology
  - Let the end host select the end-to-end path
- Variations of source routing
  - Strict: specify every hop
  - Loose: specify intermediate points
    - Used in IP source routing (but almost always disabled)

Learning Where the Hosts Are

Finding the Hosts

- Building a forwarding table
  - Computing paths between network elements
  - ... and figuring out where the end-hosts are
- How to find the hosts?
  1. Learning/flooding
  2. Injecting into the routing protocol
  3. Dissemination using a different protocol
  4. Directory service
Learning and Flooding

• When a frame arrives
  – Inspect the source address
  – Associate address with the incoming interface

• When the frame has an unfamiliar destination
  – Forward out all interfaces
  – ... except incoming interface

When in doubt, shout!

Switch learns how to reach A

Used in Ethernet LANs

Inject into Routing Protocol

• Treat the end host (or subnet) as a node
  – And disseminate in the routing protocol
  – E.g., flood information about where addresses attach

Used in OSPF and IS-IS, especially in enterprise networks

Disseminate With Another Protocol

• Distribute using another protocol
  – One router learns the route
  – ... and shares the information with other routers

Learn a route to d (e.g., via BGP)

Internal BGP (iBGP) used in backbone networks

Directory Service

• Contact a service to learn the location
  – Look up the end-host or subnet address
  – ... to determine the label to put on the packet

“Host d is at egress e”

Encapsulate packet to send to egress e

Used in some data centers
Conclusions: Many Different Solutions

- **Ethernet LAN and home networks**
  - Spanning tree, MAC learning, flooding

- **Enterprise**
  - Link-state routing, injecting subnet addresses

- **Backbone**
  - Link-state routing inside, path-vector routing with neighboring domains, and iBGP dissemination

- **Data centers**
  - Many different solutions, still in flux