Routing

Mike Freedman
COS 461: Computer Networks
Lectures: MW 10-10:50am in Architecture N101

http://www.cs.princeton.edu/courses/archive/spr13/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

Locally 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
- 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

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

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
  \( d_x(y) = \min (c(x,v) + d_v(y)) \) over all neighbors v

Distance Vector Example

Distance Vector Example (Cont.)
Distance Vector Example (Cont.)

\[ d_u(z) = \min(1 + d_v(z), 4 + d_w(z), 2 + d_y(z)) \]
\[ = 5 \]

\[ d_v(z) = \min(3 + d_x(z), 2 + d_u(z)) \]
\[ = 6 \]

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 \( d \)

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

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?
  - Learning/flooding
  - Injecting into the routing protocol
  - Dissemination using a different protocol
  - 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

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

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)
  - disseminate route to other routers
  - 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

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