Lecture 12
Routing

Peter Steenkiste
Departments of Computer Science and
Electrical and Computer Engineering
Carnegie Mellon University

15-441 Networking, Spring 2006
http://www.cs.cmu.edu/~prs/15-441

Packet Forwarding

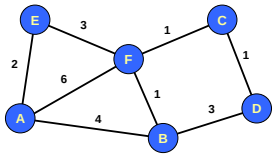
- Use IP address in the header to determine where to forward the packet
  - » Longest prefix match in the forwarding table
  - » Send packet out appropriate port
- Today's lecture: how to create and manage the forwarding table
  - » Focus on intra-domain routing
  - » Route selection is based on the optimization of a routing metric

Outline

- Distance vector routing
- RIP
- Link state routing
- OSPF

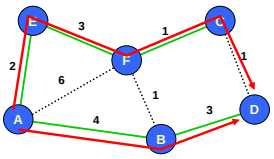
Abstraction: Represent Network as a Graph

- Represent each router as node
- Direct link between routers represented by edge
  - » Symmetric links ⇒ undirected graph
- Edge "cost" c(x,y) denotes measure of ``cost'' of using link
  - » delay, $ cost, or congestion level
- Must determine least cost path for every node pair
  - – Path cost d(x,y) = sum of link costs



Routes from Node A

Table for A		
Dest	Cost	Next Hop
A	0	A
B	4	B
C	6	E
D	7	B
E	2	E
F	5	E



- Properties
  - » Some set of shortest paths forms tree
    - – Shortest path spanning tree
  - » Solution not unique
    - – E.g., A-E-F-C-D also has cost 7

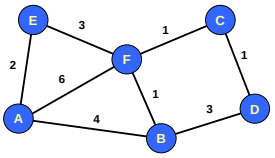
Classes of Routing Solutions

- Centralized
  - » Collect graph structure in one place
  - » Use standard graph algorithm
  - » Disseminate routing tables

  Does this work?
- Partially Distributed
  - » Every node collects complete graph structure
  - » Each locally computes shortest paths from it
  - » Each generates own routing table
  - » "Link-state" algorithm
- Fully Distributed
  - » No one has copy of graph
  - » Nodes construct their own tables iteratively
  - » Each sends information about its table to neighbors
  - » "Distance-Vector" algorithm

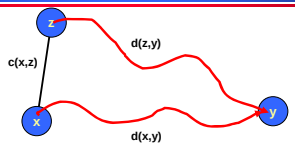
Distance-Vector Method

Initial Table for A		
Dest	Cost	Next Hop
A	0	A
B	4	B
C	∞	–
D	∞	–
E	2	E
F	6	F



- Idea is to iteratively improve least-cost path to any destination based on information learned from neighbors
  - Table lists cost/next hop of best known path to destination
  - Initially table only has entries for directly connected nodes

Distance-Vector Update



- **Update(x,y,z)**

```
d ← c(x,z) + d(z,y)        # Cost of path from x to y with first hop z
if d < d(x,y)
    # Found better path
    return d,z             # Updated cost / next hop
else
    return d(x,y), nexthop(x,y)    # Existing cost / next hop
```


Algorithm

- **Bellman-Ford algorithm**
- **Repeat**

```
For every node x
    For every neighbor z
        For every destination y
            d(x,y) ← Update(x,y,z)
```

- **Until Converge**
- **Nodes x can run the algorithm in fully distributed fashion**

Start

Optimum 1-hop paths

Table for A		
Dst	Cst	Hop
A	0	A
B	4	B
C	∞	–
D	∞	–
E	2	E
F	6	F

Table for B		
Dst	Cst	Hop
A	4	A
B	0	B
C	∞	–
D	3	D
E	∞	–
F	1	F

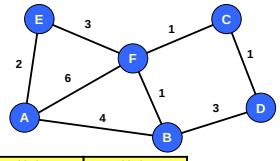


Table for C		
Dst	Cst	Hop
A	∞	–
B	∞	–
C	0	C
D	1	D
E	∞	–
F	1	F

Table for D		
Dst	Cst	Hop
A	∞	–
B	3	B
C	1	C
D	0	D
E	∞	–
F	∞	–

Table for E		
Dst	Cst	Hop
A	2	A
B	∞	–
C	∞	–
D	∞	–
E	0	E
F	3	F

Table for F		
Dst	Cst	Hop
A	6	A
B	1	B
C	1	C
D	∞	–
E	3	E
F	0	F

Iteration #1

Optimum 2-hop paths

Table for A		
Dst	Cst	Hop
A	0	A
B	4	B
C	7	F
D	7	B
E	2	E
F	5	E

Table for B		
Dst	Cst	Hop
A	4	A
B	0	B
C	2	F
D	3	D
E	4	F
F	1	F

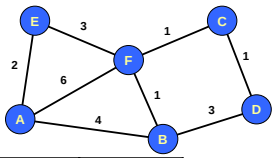


Table for C		
Dst	Cst	Hop
A	7	F
B	2	F
C	0	C
D	1	D
E	4	F
F	1	F

Table for D		
Dst	Cst	Hop
A	7	B
B	3	B
C	1	C
D	0	D
E	∞	–
F	2	C

Table for E		
Dst	Cst	Hop
A	2	A
B	4	F
C	4	F
D	∞	–
E	0	E
F	3	F

Table for F		
Dst	Cst	Hop
A	5	B
B	1	B
C	1	C
D	2	C
E	3	E
F	0	F

Iteration #2

Optimum 3-hop paths

Table for A		
Dst	Cst	Hop
A	0	A
B	4	B
C	6	E
D	7	B
E	2	E
F	5	E

Table for B		
Dst	Cst	Hop
A	4	A
B	0	B
C	2	F
D	3	D
E	4	F
F	1	F

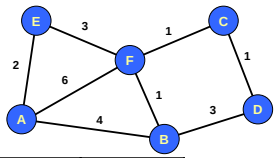


Table for C		
Dst	Cst	Hop
A	6	F
B	2	F
C	0	C
D	1	D
E	4	F
F	1	F

Table for D		
Dst	Cst	Hop
A	7	B
B	3	B
C	1	C
D	0	D
E	5	C
F	2	C

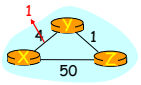
Table for E		
Dst	Cst	Hop
A	2	A
B	4	F
C	4	F
D	5	F
E	0	E
F	3	F

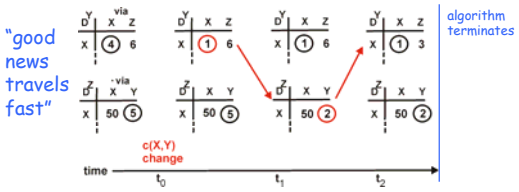
Table for F		
Dst	Cst	Hop
A	5	B
B	1	B
C	1	C
D	2	C
E	3	E
F	0	F

Distance Vector: Link Cost Changes

Link cost changes:

- Node detects local link cost change
- Updates distance table
- If cost change in least cost path, notify neighbors

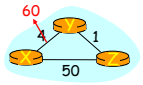


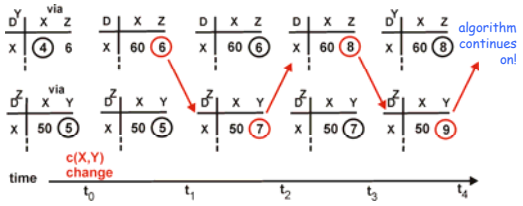


Distance Vector: Link Cost Changes

Link cost changes:

- Good news travels fast
- Bad news travels slow - "count to infinity" problem!

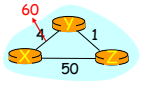


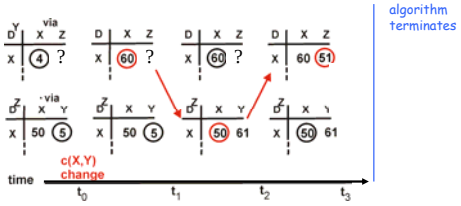


Distance Vector: Split Horizon

If Z routes through Y to get to X :

- Z does not advertise its route to X back to Y

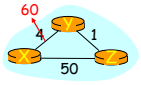


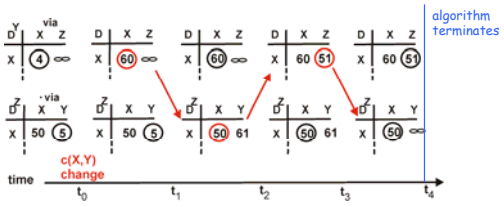


Distance Vector: Poison Reverse

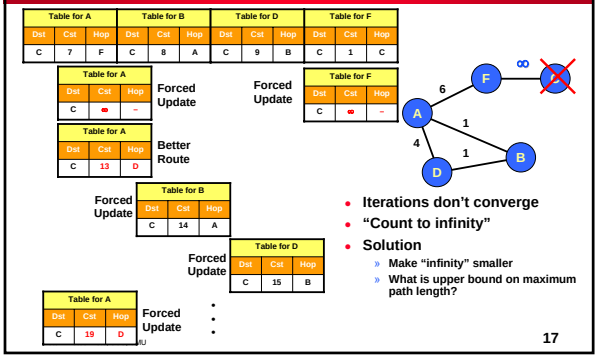
If Z routes through Y to get to X :

- 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?





Poison Reverse Failures



- Iterations don't converge
- "Count to infinity"
- Solution
  - Make "infinity" smaller
  - What is upper bound on maximum path length?

Routing Information Protocol (RIP)

- Earliest IP routing protocol (1982 BSD)
  - Current standard is version 2 (RFC 1723)
- Features
  - Every link has cost 1
  - "Infinity" = 16
    - Limits to networks where everything reachable within 15 hops
- Sending Updates
  - Every router listens for updates on UDP port 520
  - RIP message can contain entries for up to 25 table entries

RIP Updates

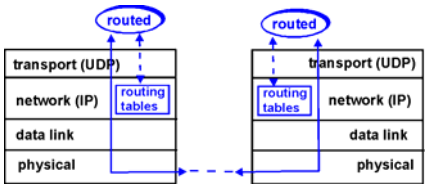
- Initial
  - » When router first starts, asks for copy of table for every neighbor
  - » Uses it to iteratively generate own table
- Periodic
  - » Every 30 seconds, router sends copy of its table to each neighbor
  - » Neighbors use to iteratively update their tables
- Triggered
  - » When every entry changes, send copy of entry to neighbors
    - – Except for one causing update (split horizon rule)
  - » Neighbors use to update their tables

RIP Staleness / Oscillation Control

- Small Infinity
  - » Count to infinity doesn't take very long
- Route Timer
  - » Every route has timeout limit of 180 seconds
    - – Reached when haven't received update from next hop for 6 periods
  - » If not updated, set to infinity
  - » Soft-state refresh → important concept!!!
- Behavior
  - » When router or link fails, can take minutes to stabilize

RIP Table Processing

- RIP routing tables managed by application-level process called route-d (daemon)
- advertisements sent in UDP packets, periodically repeated



Outline

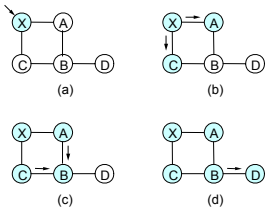
- Distance vector routing
- RIP
- Link state routing
- OSPF

Link State Protocol Concept

- Every node gets complete copy of graph
  - » Every node "floods" network with data about its outgoing links
- Every node computes routes to every other node
  - » Using single-source, shortest-path algorithm
- Process performed whenever needed
  - » When connections die / reappear

Sending Link States by Flooding

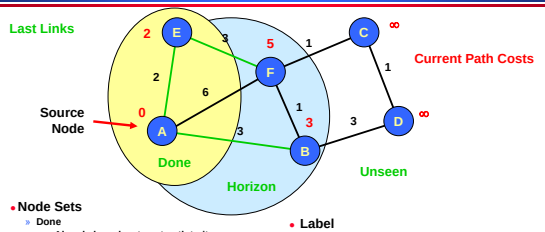
- X wants to send information
  - » Sends on all outgoing links
- When node Y receives information from Z
  - » Send on all links other than Z
- Sequence number used to suppress duplicates



Dijkstra's Algorithm

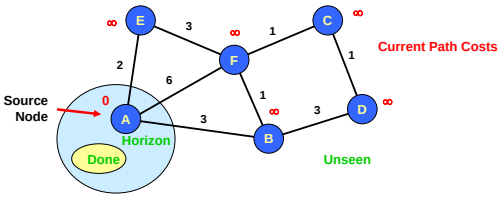
- **Given**
  - Graph with source node s and edge costs c(u,v)
  - Determine least cost path from s to every node v
- **Shortest Path First Algorithm**
  - Traverse graph in order of least cost from source

Dijkstra's Algorithm: Concept



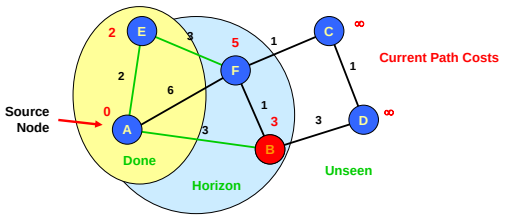
- **Node Sets**
  - Done
    - Already have least cost path to it
  - Horizon:
    - Reachable in 1 hop from node in Done
  - Unseen:
    - Cannot reach directly from node in Done
- **Label**
  - d(v) = path cost
    - From s to v
- **Path**
  - Keep track of last link in path

Dijkstra's Algorithm: Initially



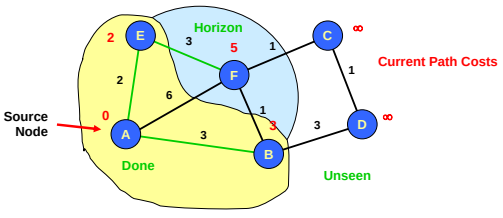
- **No nodes done**
- **Source in horizon**

Dijkstra's Algorithm: Selection



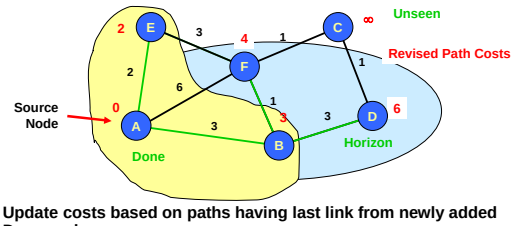
- Select node v in horizon with minimum d(v)

Dijkstra's Algorithm: Selection



- Add selected node to Done

Dijkstra's Algorithm: Update

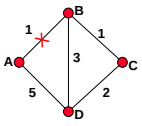


- **Update costs based on paths having last link from newly added Done node**
  - Could change values of nodes in horizon
  - Could add new nodes to horizon
- **Update link information**

Link State Characteristics

- With consistent LSDBs*, all nodes compute consistent loop-free paths
- Can still have transient loops

*Link State Data Base



Packet from C→A may loop around BDC if B knows about failure and C & D do not

OSPF Routing Protocol

- Open
  - Open standard created by IETF
- Shortest-path first
  - Another name for Dijkstra's algorithm
- RIP viewed as outmoded
  - Good when networks small and routers had limited memory & computational power
- OSPF Advantages
  - Fast convergence when configuration changes

OSPF Reliable Flooding

- Transmit link state advertisements
  - Originating router
    - Typically, minimum IP address for router
  - Link ID
    - ID of router at other end of link
  - Metric
    - Cost of link
  - Link-state age
    - Incremented each second
    - Packet expires when reaches 3600
  - Sequence number
    - Incremented each time sending new link information

OSPF Flooding Operation

- Node X Receives LSA from Node Y
  - With Sequence Number q
  - Looks for entry with same origin/link ID
- Cases
  - No entry present
    - Add entry, propagate to all neighbors other than Y
  - Entry present with sequence number p < q
    - Update entry, propagate to all neighbors other than Y
  - Entry present with sequence number p > q
    - Send entry back to Y
    - To tell Y that it has out-of-date information
  - Entry present with sequence number p = q
    - Ignore it

Flooding Issues

- When should it be performed
  - Periodically
  - When status of link changes
    - Detected by connected node
- What happens when router goes down & back up
  - Sequence number reset to 0
    - Other routers may have entries with higher sequence numbers
  - Router will send out LSAs with number 0
  - Will get back LSAs with last valid sequence number p
  - Router sets sequence number to p+1 & resends

Comparison of LS and DV Algorithms

Message complexity

- LS: with n nodes, E links, O(nE) messages
- DV: exchange between neighbors only

Space requirements:

- LS maintains entire topology
- DV maintains only neighbor state

Speed of Convergence

- LS: Complex computation
  - But…can forward before computation
  - may have oscillations
- DV: convergence time varies
  - may be routing loops
  - count-to-infinity problem
  - (faster with triggered updates)

Comparison of LS and DV Algorithms

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
  - errors propagate thru network

- Other tradeoffs
  - Making LSP flood reliable