15-441
15-641
Computer Networking

Lecture 9: Routing
Peter Steenkiste

Fall 2016
www.cs.cmu.edu/~prs/15-441-F16

To Autolab or Not to Autolab

- Autolab is supposed to make life easier
  - Give you confidence that you will credit for tests
- But autolab is an artificial execution environment
  - Isolated: no contact with other servers, file systems, ..
  - Strange timing: packet latencies can become very unpredictable and (possibly) unrealistic
  - It is very difficult (~impossible) to debug in autolab
- Recommendation: try to get tests to work in autolab but limit how much time you spend on it
- You will get a chance to run the tests over Andrew, without loss of credit

To Take or Not to Take a Late Penalty

- Nobody likes to miss a deadline
  - But is happens, especially when you are overcommitted
- Put it in perspective: the penalty (in this course) is generally not a big deal
  - For assignment A: % of A x weight of A x 15%
  - CP2 of P1: ~0.75% of points for the course
- Think strategically:
  - Final project submissions: you can use a late day
  - The rest: move on and learn from the experience
- Biggest concern: cascading missed deadlines

To Use or Not to Use a Late Day

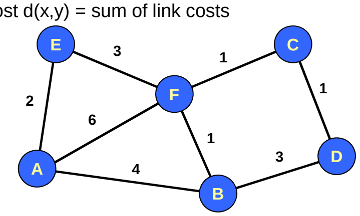
- You don't take late days
- At the end of the course, we look at what assignments you were late for and we use your late days so it maximizes your grade
  - You can also "suggest" what you think is optimal
- It is to your advantage to keep track of this so you can make good decisions during the semester

Outline

- Routing intro
- Distance Vector
- Link State

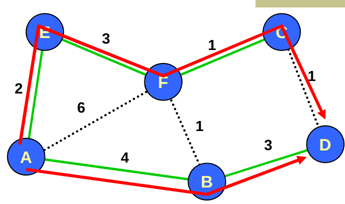
Graph Model

- Represent each router as node
- Direct link between routers represented by edge
  - Symmetric links ⇒ undirected graph
- Edge "cost" c(x,y) denotes measure of difficulty of using link
  - delay, $ cost, or congestion level
- Task
  - Determine least cost path from every node to every other node
    - Path cost d(x,y) = sum of link costs



Routes from Node A

Forwarding 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

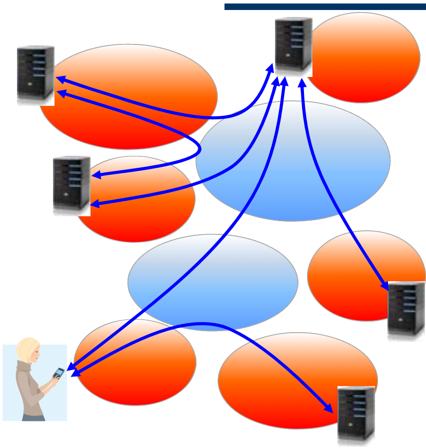


- Set of shortest paths forms a tree
  - Shortest path spanning tree
- Solution is not unique
  - E.g., A-E-F-C-D also has cost 7

Ways to Compute Shortest Paths

- Centralized
  - Collect graph structure in one place
  - Use standard graph algorithm
  - Disseminate routing tables
- Link-state
  - Every node collects complete graph structure
  - Each computes shortest paths from it
  - Each generates its own routing table
- Distance-vector
  - No one has copy of graph
  - Nodes construct their own tables iteratively
  - Each sends information about its table to neighbors

Routing Hierarchy



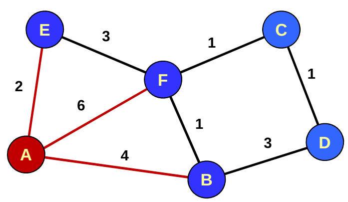
- IP packets must travel across domains – inter-domain routing
  - Primary role of IP
  - Based on CIDR prefix
- Must also travel through domains – intro-domain routing
  - Across subnets
  - Based on subnet ID or longer prefix

Outline

- Routing intro
- Distance Vector
- Link State

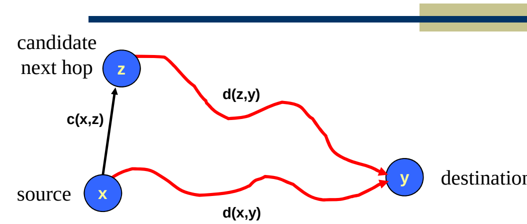
Distance-Vector Method

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



- Idea
  - At any time, have cost/next hop of best known path to destination
  - Use cost ∞ when no path known
- Initially
  - Only have 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 for destination y

  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

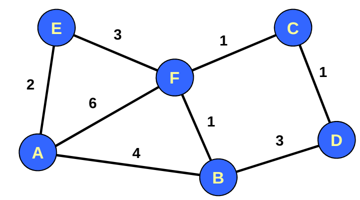
  d(x,y) ← Update(x,y,z)
- Until converge

Start

Optimum 1-hop paths

Table for A			Table for B		
Dst	Cst	Hop	Dst	Cst	Hop
A	0	A	A	4	A
B	4	B	B	0	B
C	∞	–	C	∞	–
D	∞	–	D	3	D
E	2	E	E	∞	–
F	6	F	F	1	F

Table for C			Table for D			Table for E			Table for F		
Dst	Cst	Hop	Dst	Cst	Hop	Dst	Cst	Hop	Dst	Cst	Hop
A	∞	–	A	∞	–	A	2	A	A	6	A
B	∞	–	B	3	B	B	∞	–	B	1	B
C	0	C	C	1	C	C	∞	–	C	1	C
D	1	D	D	0	D	D	∞	–	D	∞	–
E	∞	–	E	∞	–	E	0	E	E	3	E
F	1	F	F	∞	–	F	3	F	F	0	F

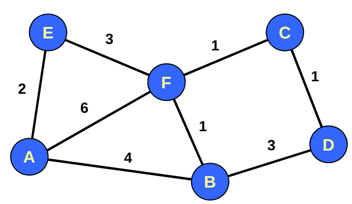


Iteration #1

Optimum 2-hop paths

Table for A			Table for B		
Dst	Cst	Hop	Dst	Cst	Hop
A	0	A	A	4	A
B	4	B	B	0	B
C	7	F	C	2	F
D	7	B	D	3	D
E	2	E	E	4	F
F	5	E	F	1	F

Table for C			Table for D			Table for E			Table for F		
Dst	Cst	Hop	Dst	Cst	Hop	Dst	Cst	Hop	Dst	Cst	Hop
A	7	F	A	7	B	A	2	A	A	5	B
B	2	F	B	3	B	B	4	F	B	1	B
C	0	C	C	1	C	C	4	F	C	1	C
D	1	D	D	0	D	D	∞	–	D	2	C
E	4	F	E	∞	–	E	0	E	E	3	E
F	1	F	F	2	C	F	3	F	F	0	F

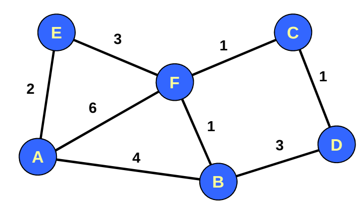


Iteration #2

Optimum 3-hop paths

Table for A			Table for B		
Dst	Cst	Hop	Dst	Cst	Hop
A	0	A	A	4	A
B	4	B	B	0	B
C	6	E	C	2	F
D	7	B	D	3	D
E	2	E	E	4	F
F	5	E	F	1	F

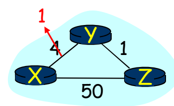
Table for C			Table for D			Table for E			Table for F		
Dst	Cst	Hop	Dst	Cst	Hop	Dst	Cst	Hop	Dst	Cst	Hop
A	6	F	A	7	B	A	2	A	A	5	B
B	2	F	B	3	B	B	4	F	B	1	B
C	0	C	C	1	C	C	4	F	C	1	C
D	1	D	D	0	D	D	5	F	D	2	C
E	4	F	E	5	C	E	0	E	E	3	E
F	1	F	F	2	C	F	3	F	F	0	F

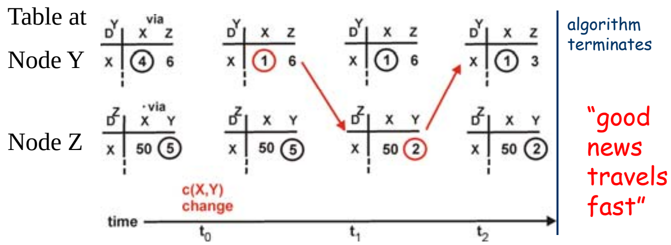


Distance Vector: Link Cost Changes

Link cost changes:

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

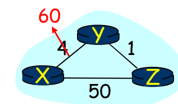


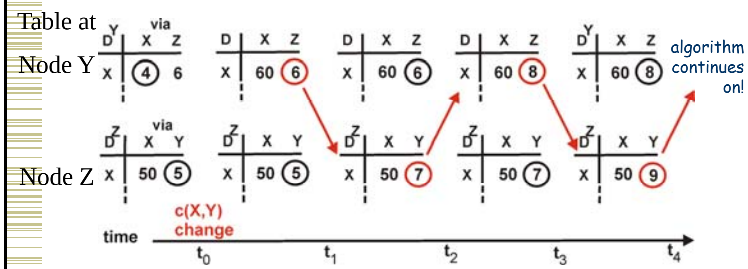


Distance Vector: Link Cost Changes

Link cost changes:

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





Bad News Travels Slowly

Is this a problem? Yes!

- After a path cost increases, it can take a very long time before paths stabilize, and
- During this process, the network has a routing loop

What is the cause?

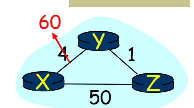
- Nodes refuse to accept the up-to-date information, because they prefer the older, better cost
- Outdated information based on the older, lower path cost loops around the network

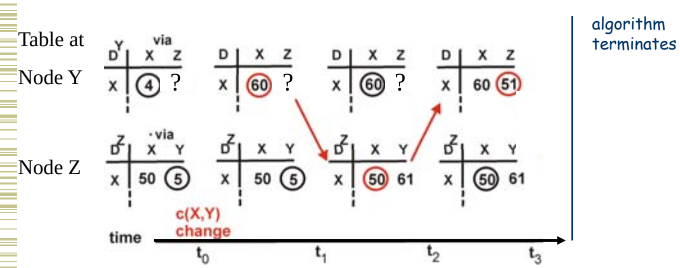
Distance Vector: Split Horizon

Problem: if Z routes through Y to get to X, it still advertises its path back to Y

- This serves no purpose and causes the loops

Solution: Z does not advertise its route back to Y





Poison Reverse Failures

Table for A			Table for B			Table for D			Table for F		
Dst	Cst	Hop	Dst	Cst	Hop	Dst	Cst	Hop	Dst	Cst	Hop
C	7	F	C	8	A	C	9	B	C	1	C

Table for A		
Dst	Cst	Hop
C	∞	–

Forced Update

Forced Update

Table for F		
Dst	Cst	Hop
C	∞	–

Table for A		
Dst	Cst	Hop
C	13	D

Better Route

Forced Update

Table for B		
Dst	Cst	Hop
C	14	A

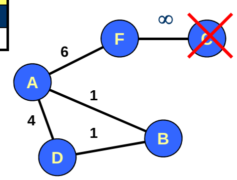
Forced Update

Table for D		
Dst	Cst	Hop
C	15	B

Table for A		
Dst	Cst	Hop
C	19	D

Forced Update

⋮



- Split horizon does not help!
- Especially bad if a link goes down: "Count to infinity"
- Solution:
  - Make "infinity" smaller
  - Force the cost "infinity" to all interfaces and wait
  - Helps network converge faster

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 network diameter to 15 hops
- Routers exchange different types of updates
  - Initial: asks for copy of table for every neighbor when it starts
    - Uses it to iteratively generate own table
  - Periodic: sends copy of its table to each neighbor every 30 sec
    - Neighbors use it to iteratively update their tables
  - Triggered: send copy of entry to neighbors when entry changes
    - Except for one causing update (split horizon rule)
    - Neighbors use it to update their tables

Outline

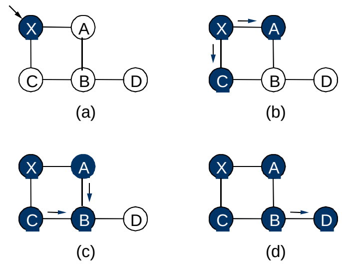
- Routing intro
- Distance Vector
- Link State

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

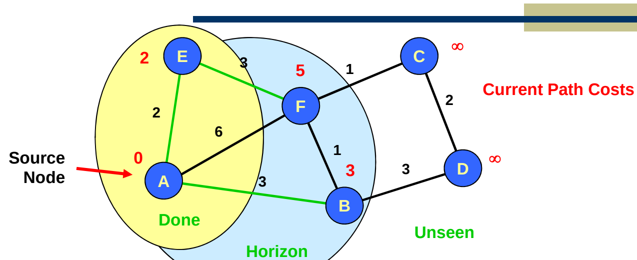
- X Wants to Send Information
  - Sends on all outgoing links
- When Node Y Receives Information from Z
  - Send on all links other than Z



Dijkstra's Algorithm

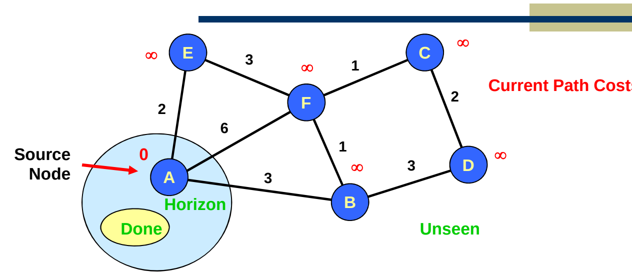
- Given
  - Graph with source node *s* and edge costs *c(u,v)*
  - Determine least cost path from *s* to every node *v*
- Shortest Path First Algorithm
  - Traverse graph in order of least cost from source

Dijkstra's Algorithm: Concept



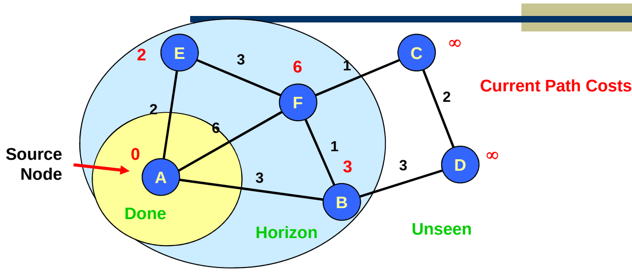
- Node Sets
  - Done
    - Already have least cost path to it
  - Horizon:
    - Reachable in 1 hop from node in Done
  - Unseen:
    - Cannot reach directly from node in Done
- Label
  - *d(v)* = path cost from s to v
- Path
  - Keep track of last link in path

Dijkstra's Algorithm: Initially



- No nodes done
- Source in horizon

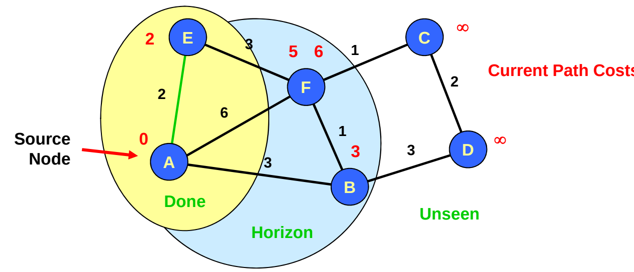
Dijkstra's Algorithm: Initially



- d(v) to node A shown in red
  - Only consider links from done nodes

30

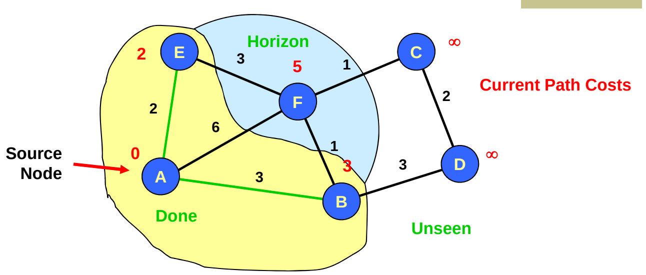
Dijkstra's Algorithm



- Select node v in horizon with minimum d(v)
- Add link used to add node to shortest path tree
- Update d(v) information

31

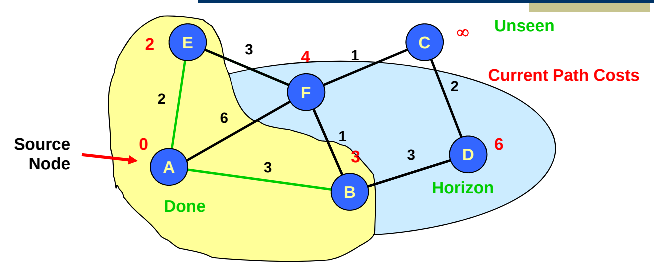
Dijkstra's Algorithm



- Repeat…

32

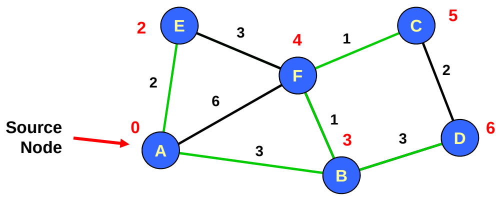
Dijkstra's Algorithm



- Update d(v) values
  - Can cause addition of new nodes to horizon

33

Dijkstra's Algorithm

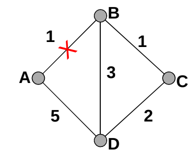


- Final tree shown in green

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 standard created by IETF
- Shortest-path first
  - Another name for Dijkstra's algorithm
- Replaced RIP
  - RIP is dated, given today's requirements
  - OSPF has fast convergence when configuration changes
  - OSPF can scale to very large networks using "areas"

OSPF Reliable Flooding

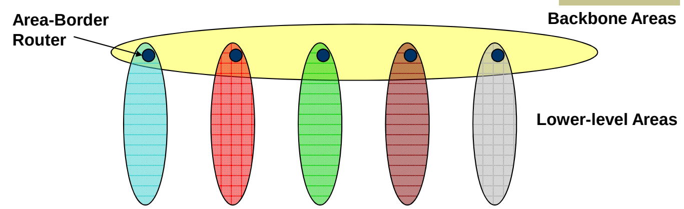
- Transmit link state advertisements
  - Originating router
    - Typically, minimum IP address for router
  - Link ID
    - ID of router at other end of link
  - Metric
    - Cost of link
  - Link-state age
    - Incremented each second
    - Packet expires when reaches 3600
  - Sequence number
   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

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

38

Areas: Scaling to Larger Networks



- Within area: Each node has routes to every other node
- Outside area: Each node has routes for other top-level areas only
  - Inter-area packets are routed to nearest border router
  - Constraint: no path between two sub-areas of an area can exit that area
  - May no longer have shortest path routes

39

Comparison of LS and DV Algorithms

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

Speed of Convergence
- LS: Relatively fast
  - Complex computation, but can forward before computation
  - may have transient loops
- DV: convergence time varies
  - may have routing loops
  - count-to-infinity problem
  - faster with triggered updates

Space requirements:
- LS maintains entire topology
- DV maintains only neighbor state

Robustness: router malfunctions
- LS: Node can advertise incorrect link cost
  - Each node computes its own table
- DV: Node can advertise incorrect path cost
  - Each node's table used by others (error propagates)

40