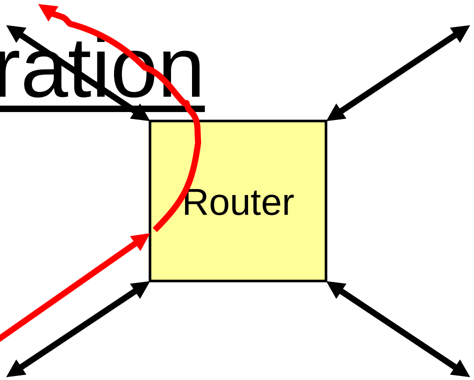
ECE 463
Introduction to Computer Networks

Lecture: Routing

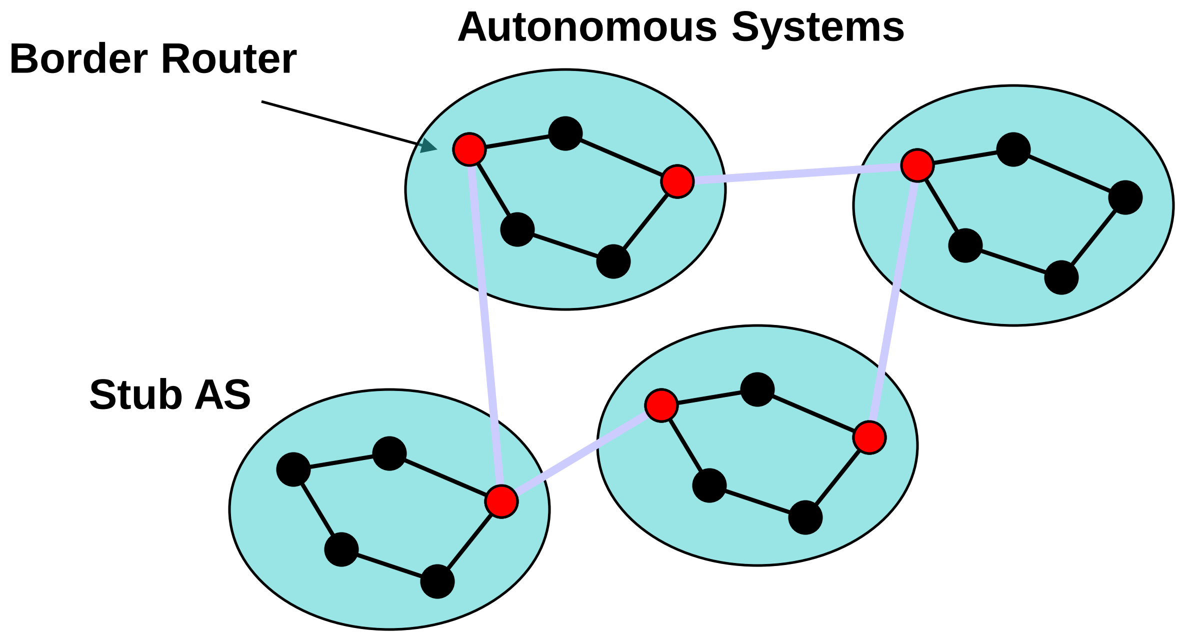
Sanjay Rao

Router Operation



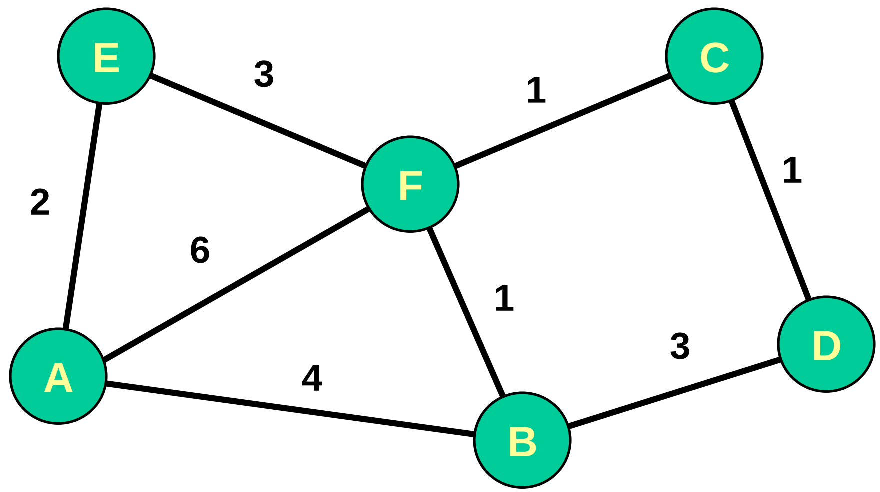
- When Packet Arrives at Router
  - Examine header to determine intended destination
  - *Look up in table to determine next hop in path*
  - Send packet out appropriate port
- Terminology
  - Each router *forwards* packet to next router
  - Overall goal is to *route* packet from source to destination
- Today's task
  - How to generate the routing table

Routing Hierarchy



- Autonomous System
  - Network administered by single entity
- Intradomain Routing
  - Routing within single AS
  - Typically ~ 200 nodes
- Interdomain Routing
  - Routing between AS's

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

- Task
  - Determine least cost path from every node to every other node
    - Path cost d(x,y) = sum of link costs

Ways to Compute Shortest Paths

- Centralized
  - Collect graph structure in one place
  - Use standard graph algorithm
  - Disseminate routing tables
- Partially Distributed
  - Every node collects complete graph structure
  - Each 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

Example Routing Tables

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

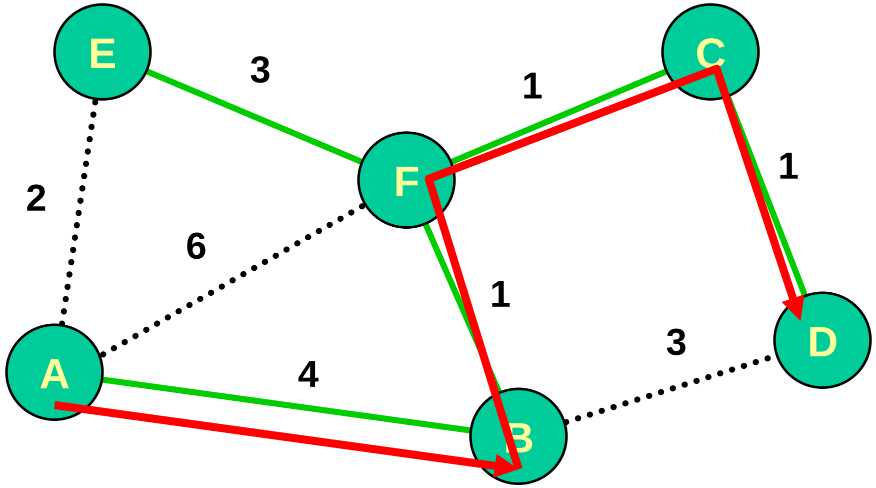


Table for B		
Dest	Cost	Next Hop
A	4	A
B	0	B
C	2	F
D	3	F
E	4	F
F	1	F

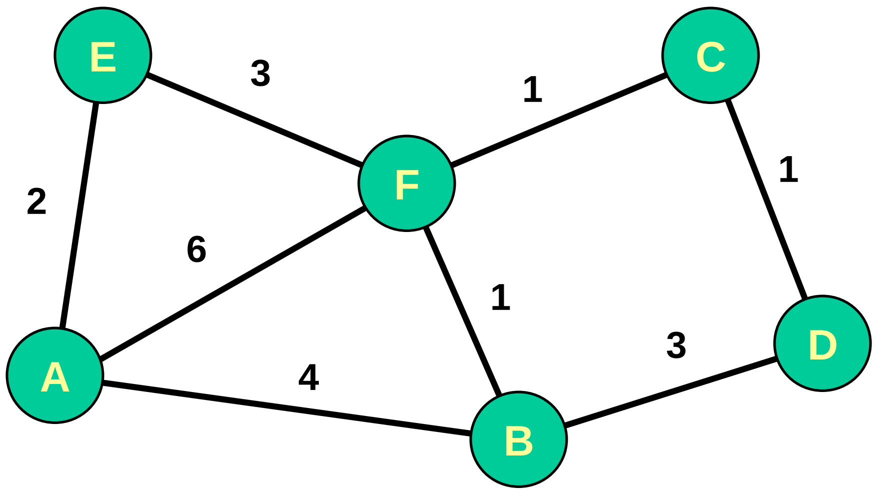
- Multiple choices for A
  - A-E-F-C-D, A-B-D
- Multiple choices for B as well
  - B-D, B-F-C-D
- A may not know route B takes

Issues

- Ensure packets don’t get stuck in a loop
- Find “good” paths
- Adapt to changes in edge costs
- Adapt to failure of nodes

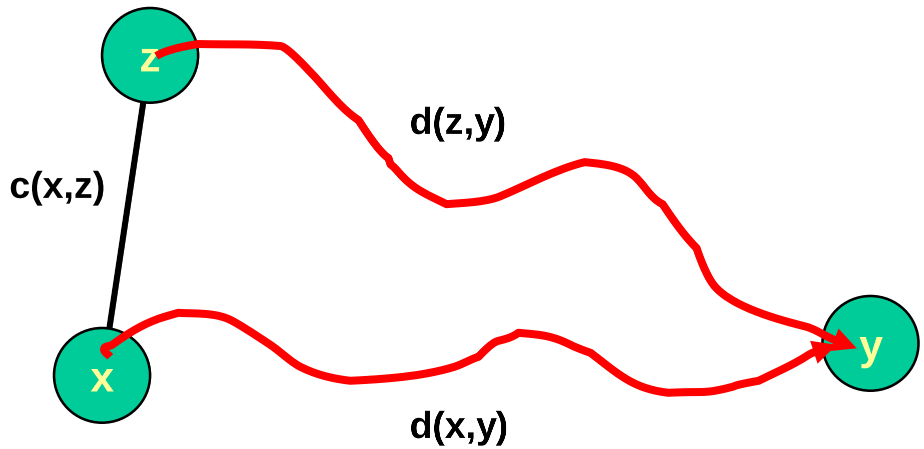
Distance-Vector Method

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

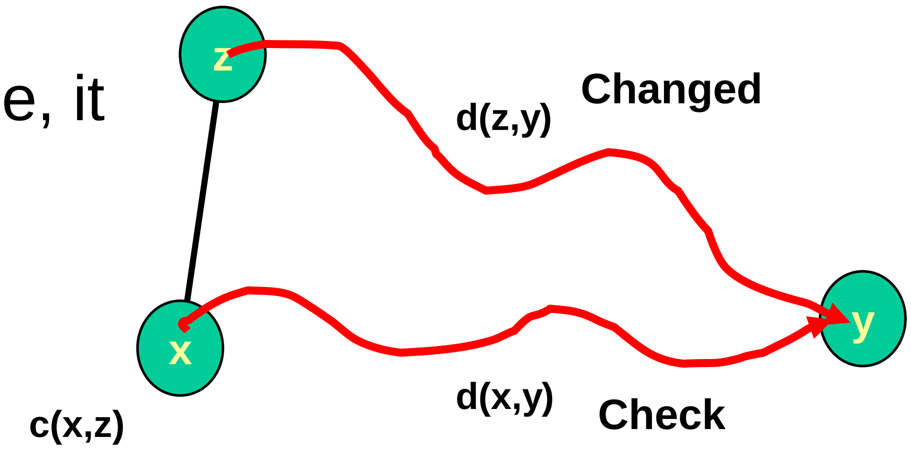
  return d,z   # Updated cost / next hop

  else

  return d(x,y), nexthop(x,y)   # Existing cost / next hop

Routing Algorithm

- Periodically, every node z sends each neighbor x a copy of its routing table
- When x receives the table, it
  - runs Update(x,y,z) for every destination y
- Process occurs in "asynchronous" fashion
- Routing tables "eventually" converge



What if Node Fails?

- D & F notice that C isn't responding
- Set entries to ∞
- Iterate

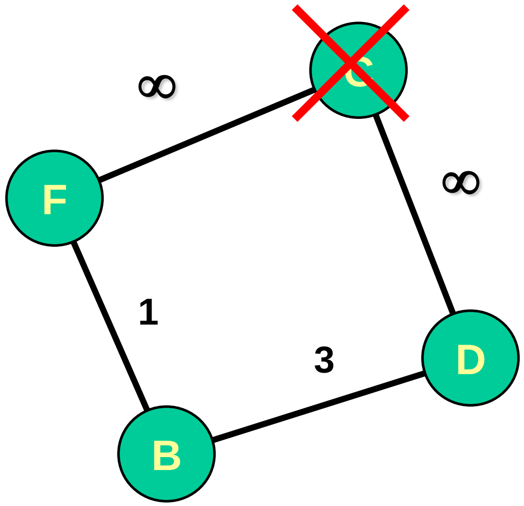


Table for D		
Dst	Cst	Hop
C	∞	–

Table for F		
Dst	Cst	Hop
C	∞	–

Failing Node Iterations

- Stale entry in B propagates to D & F
- What Happened?

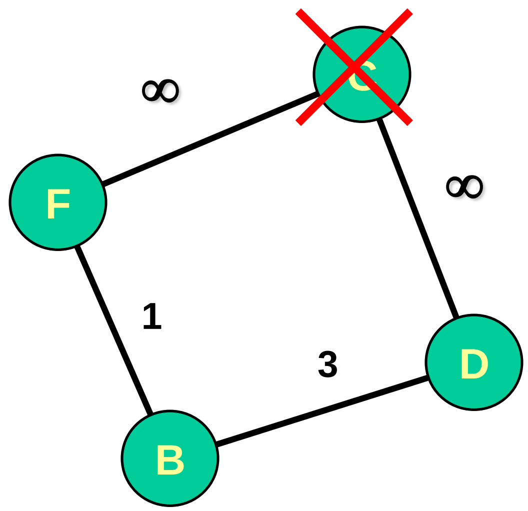


Table for B		
Dst	Cst	Hop
C	2	F

Table for D		
Dst	Cst	Hop
C	∞	–

Table for F		
Dst	Cst	Hop
C	∞	–

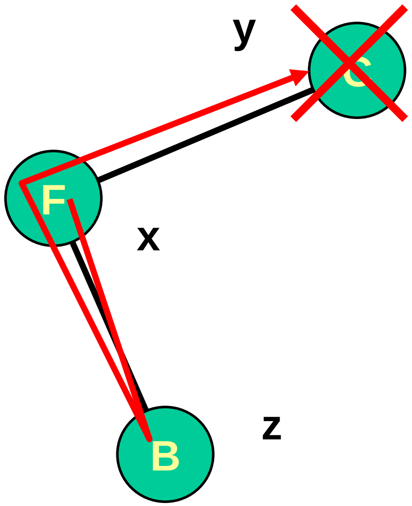
Better Route

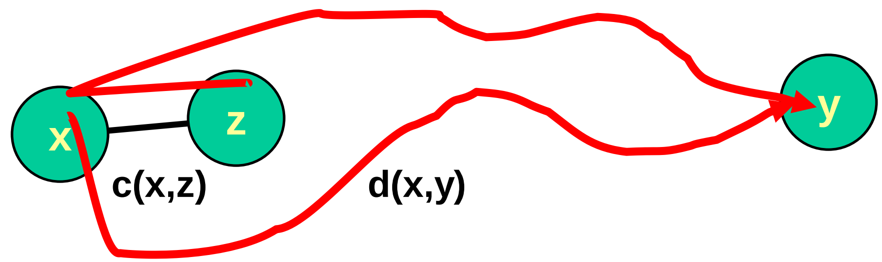
Table for D		
Dst	Cst	Hop
C	5	B

Better Route

Table for F		
Dst	Cst	Hop
C	3	B

Revised Update Rule #1





Variants: "Split Horizon Rule",
"Split Horizon with Poison Reverse"

- Update(x,y,z)

  d $\leftarrow$ c(x,z) + d(z,y) # Cost of path from x to y with first hop z

  if d < d(x,y) & x $\neq$ nexthop(z,y)

  # Found better path

  return d,z

  else

  return d(x,y), nexthop(x,y)

Iterations with Revision #2

- Stale entry in B still propagates

- What Happened?

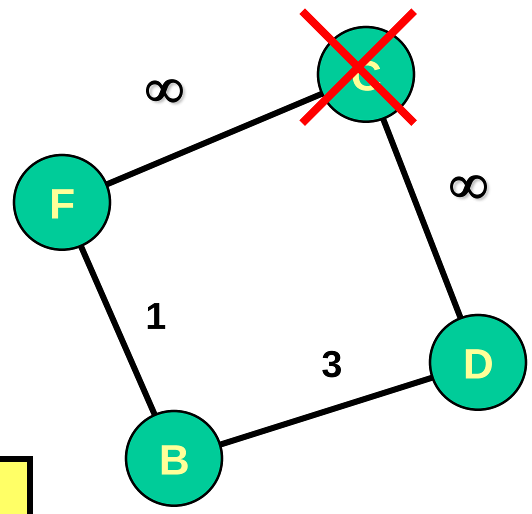


Table for B		
Dst	Cst	Hop
C	2	F

Table for D		
Dst	Cst	Hop
C	∞	–

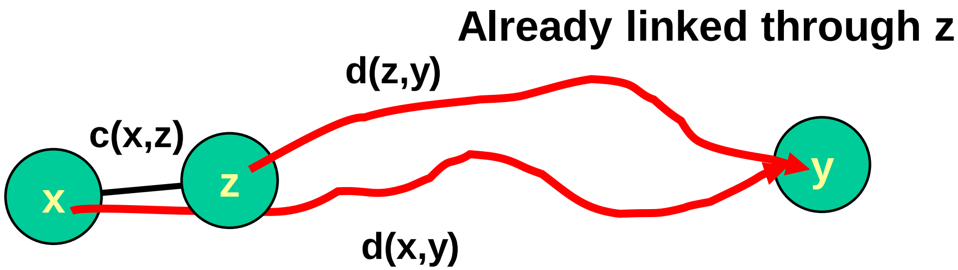
Table for F		
Dst	Cst	Hop
C	∞	–

No Change

Better Route

Table for D		
Dst	Cst	Hop
C	5	B

Revised Update Rule #2



- Update(x,y,z)

  d $\leftarrow$ c(x,z) + d(z,y) # Cost of path from x to y with first hop z

  if nexthop(x,y) = z | # Forced update

  (d < d(x,y) & x $\neq$ nexthop(z,y))

  # Forced update or found better path

  return d,z

  else

  return d(x,y), nexthop(x,y)

Iterations with Revision #2

Table for B		
Dst	Cst	Hop
C	2	F

Table for D		
Dst	Cst	Hop
C	∞	–

Table for F		
Dst	Cst	Hop
C	∞	–

No Change

Better Route

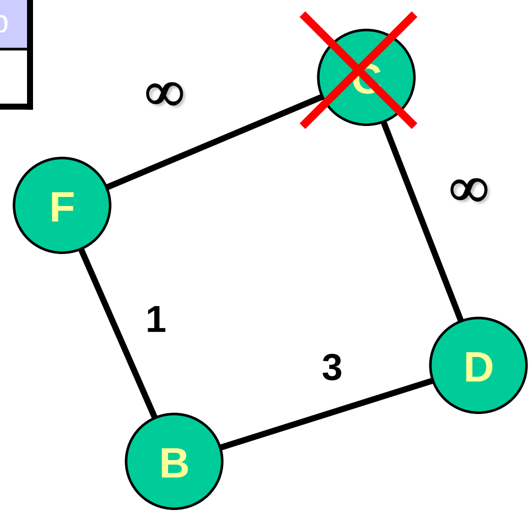
Table for D		
Dst	Cst	Hop
C	5	B

Forced Update

Table for B		
Dst	Cst	Hop
C	∞	–

Forced Update

Table for D		
Dst	Cst	Hop
C	∞	–



– Forced updates will eliminate false entries

– Forced update rule violates monotonicity
  - Increases d(x,y)

Routing Information Protocol (RIP)

- Earliest IP routing protocol (1982 BSD)
  - Ideas in first Arpanet protocols (late 60's)
- 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 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 entry changes, send copy of entry to neighbors
    - Except for one causing update
  - Neighbors use to update their tables

RIP Staleness / Oscillation Control

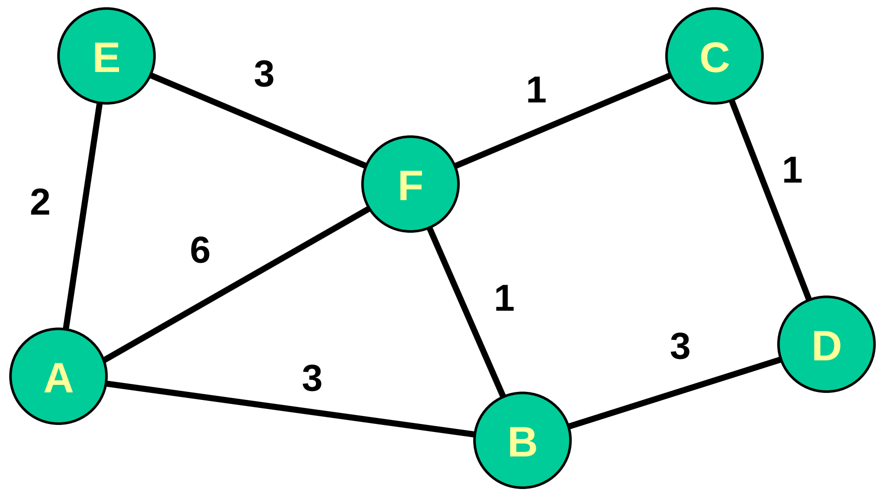
- Small Infinity
  - Count to infinity doesn't take very long
- Route Timer
  - Every route has timeout limit of 180 seconds
    - Reached when haven't received update from next hop for 6 periods
  - If not updated, set to infinity
- Behavior
  - When router or link fails, can take minutes to stabilize
  - Lots of subtlety to get good implementation (RFC 1058).

Features of Distributed Algorithms

- Desirable in Network Setting
  - Every node operates in purely local way
    - No central control or global synchronization
  - Only communication between direct neighbors
- Not Difficult to Handle Static System
  - Monotonicity guarantees convergence
- Difficult in Dynamically-Changing System
  - Anything that reduces link cost OK
    - Iterations will converge to reflect reduced costs
  - Anything that increases link cost problematic
    - Iterations will converge, but possibly to wrong values
    - Changing update rule can lead to convergence problems
      - Violate monotonicity

---Link-State Routing---

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

- Task
  - Determine least cost path from every node to every other node
    - Path cost d(x,y) = sum of link costs

Optimal Routing Table

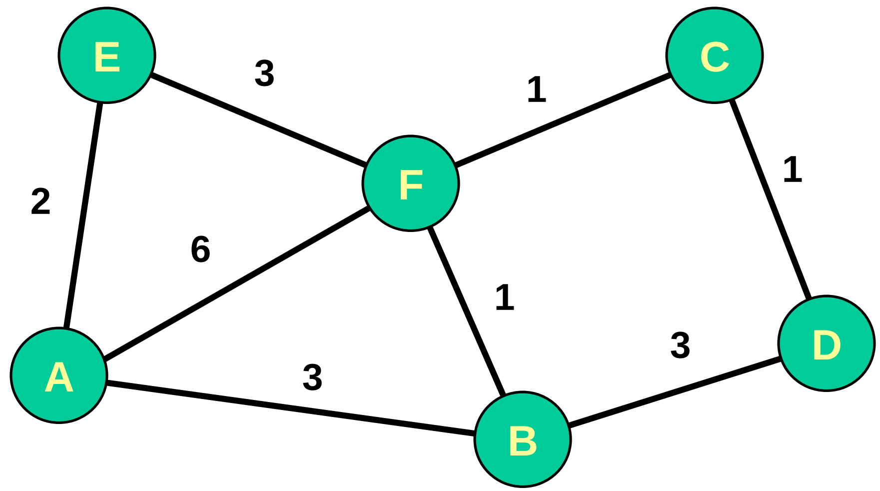


Table for A		
Dst	Cst	Hop
A	0	A
B	3	B
C	5	B
D	6	B
E	2	E
F	4	B

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

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

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

Ways to Compute Shortest Paths

- Centralized
  - Collect graph structure in one place
  - Use standard graph algorithm
  - Disseminate routing tables
- Partially Distributed
  - Every node collects complete graph structure
  - Each 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

Why Not Fully Distributed?

- Original RIP Protocol
  - Link cost = 1
  - Treat 16 like infinity
- Good Features
  - Simple to implement
  - Completed distributed
- Bad Features
  - Potentially slow convergence
    - Count-to-infinity
    - Tables could have loops when not yet converged
  - Had to use very weak metrics & features
  - Restricted to spanning-tree routes

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
- Every Node Updates Own Routing Table
- Process Performed Whenever Needed
  - When connections die / reappear
  - Periodically

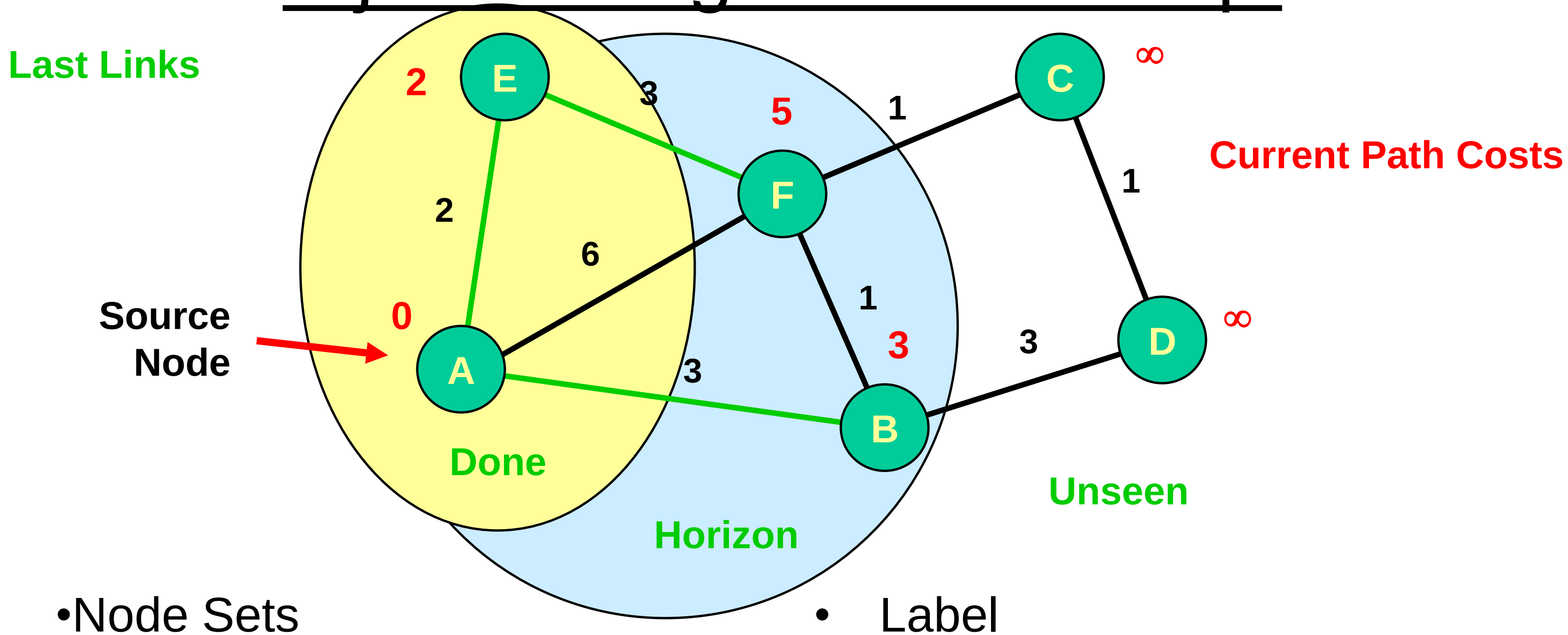
Link State Pros & Cons

- Advantages
  - Rapidly adapts to changes in network
  - Can afford to use more sophisticated link costs (“metrics”)
  - Can incorporate multiple paths
- Disadvantages
  - Difficult to make reliable
    - Must make sure all nodes get consistent copy of graph
  - Tempting to add lots of features
    - Sources of complexity & bugs

Dijkstra's Algorithm

- Edsgar Dijkstra (1930--2002)
  - Pioneer in understanding mathematical basis for computer science
  - Fundamental ideas in concurrency (e.g., semaphores)
- 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

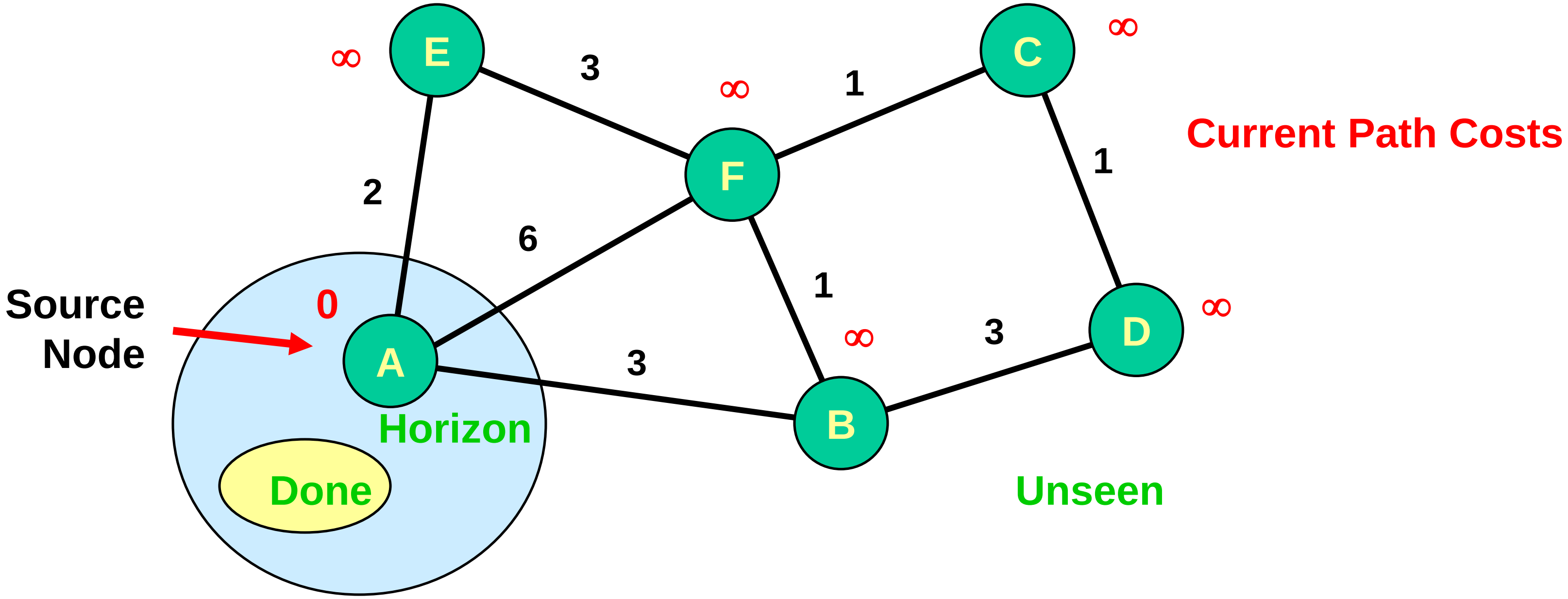
Dijkstra's Algorithm: Concept



- 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

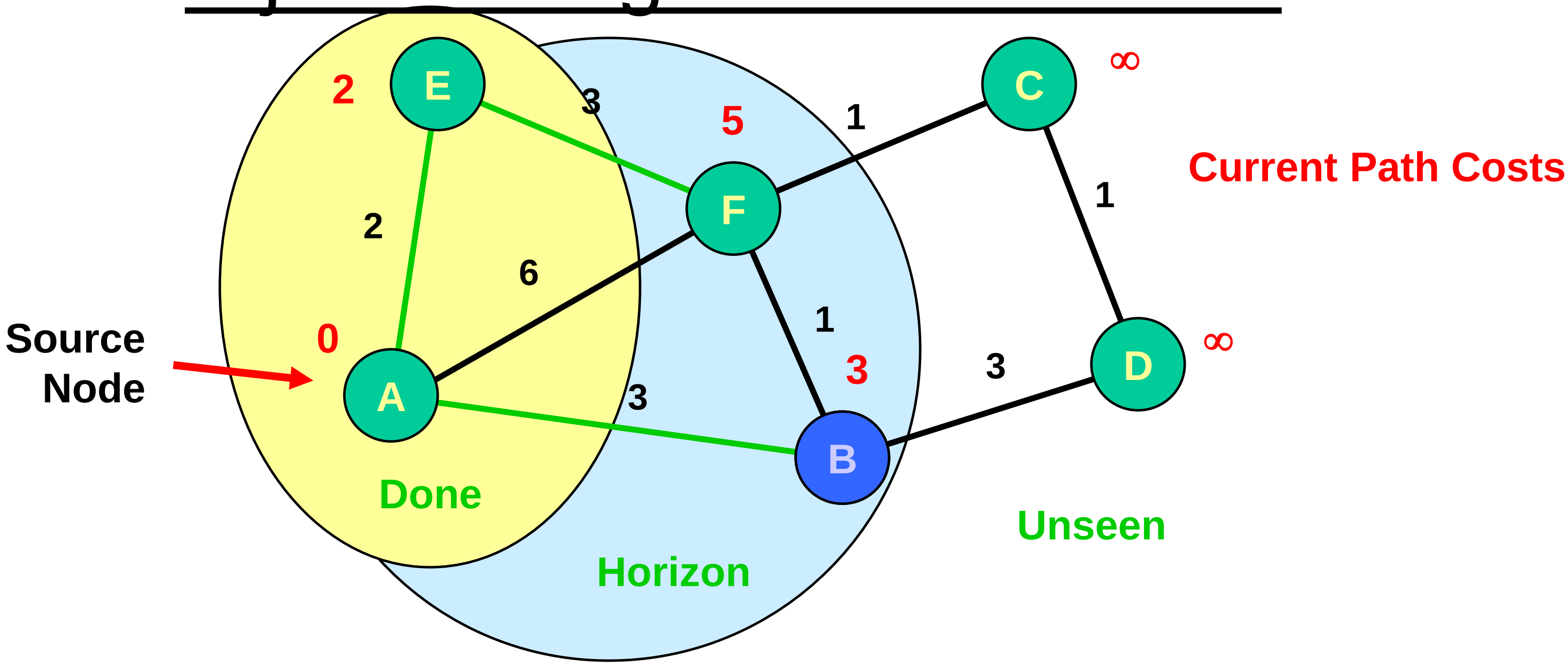
- Label
  - d(v) = path cost
    - From s to v
    - All but v are done nodes
    - Otherwise optimal
- Path
  - Keep track of last link in path

Dijkstra's Algorithm: Initially



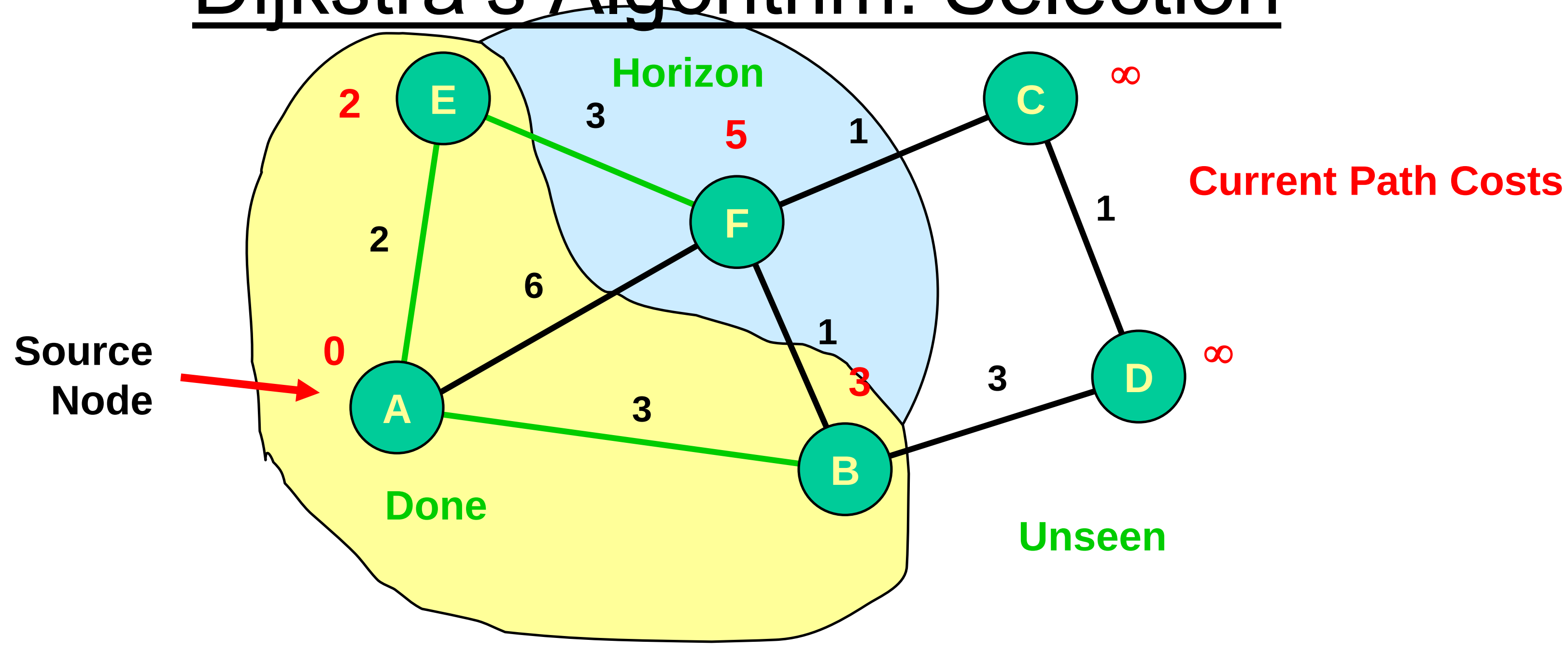
- No nodes done
- Source in horizon

Dijkstra's Algorithm: Selection



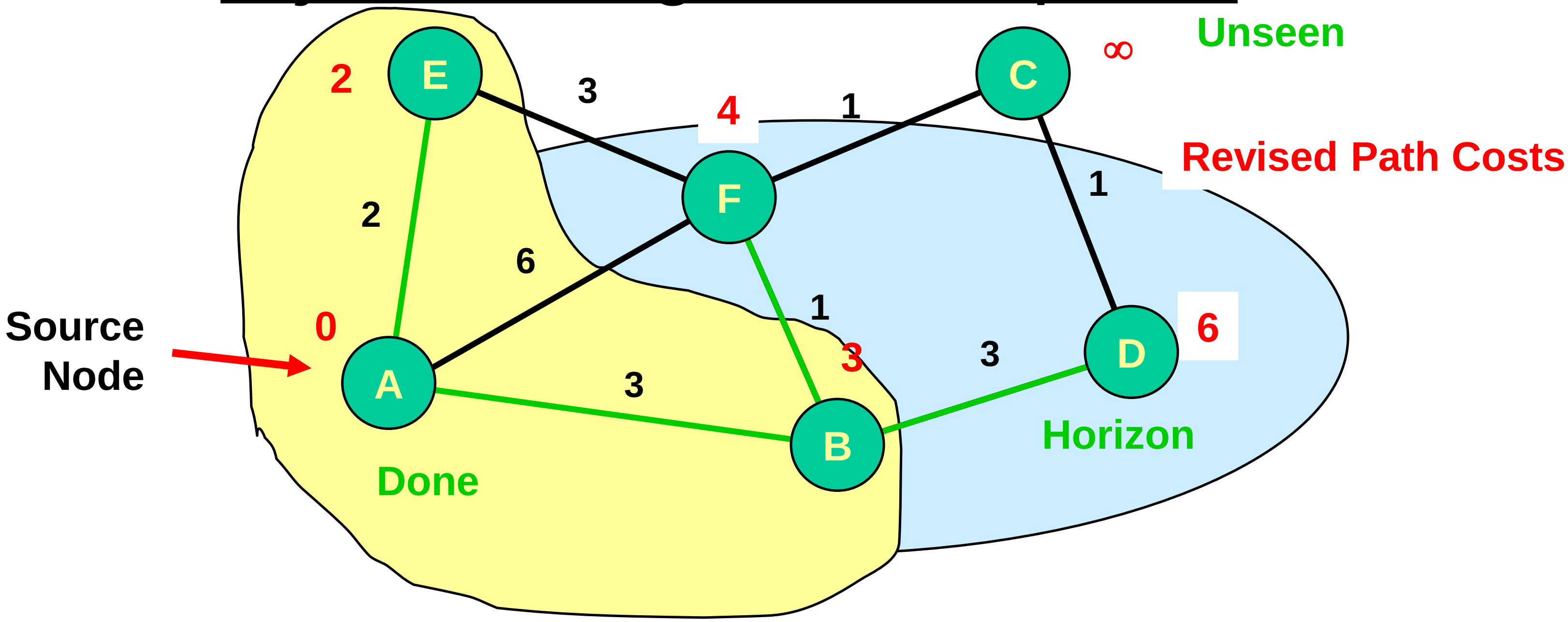
– Select node v in horizon with minimum d(v)

Dijkstra's Algorithm: Selection



– Add selected node to Done

Dijkstra's Algorithm: Update



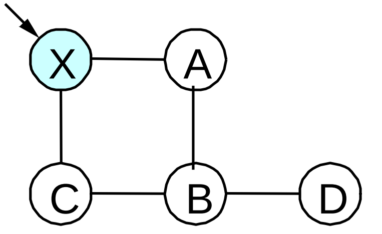
– Update costs based on paths having last link from newly added Done node
  - Could change values of nodes in horizon
  - Could add new nodes to horizon
– Update link information

OSPF Routing Protocol

- Open
  - Open standard created by IETF
- Shortest-Path First
  - Another name for Dijkstra's algorithm
- Most Prevalent Intradomain Routing Protocol

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
- Naïve Approach:
  - Floods indefinitely.
  - Prevent through sequence numbers



OSPF Reliable Flooding

- Transmit Link State Advertisements
  - Originating Router
  - List of directly connected neighbors of that node with the cost of the link to each one
  - Sequence Number
    - Incremented each time sending new link information
  - Link State Age
    - Packet expires when a threshold is reached,

OSPF Flooding Operation

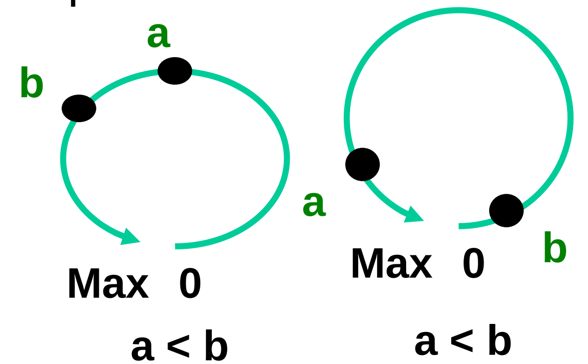
- Node X Receives LSA from Node Y
  - With Sequence Number q
  - Looks for entry with same origin/link ID
- Cases
  - No entry present
    - Add entry, propagate to all neighbors other than Y
  - Entry present with sequence number $p < q$
    - Update entry, propagate to all neighbors other than Y
  - 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

Flooding Issues (Cont.)

- What if Sequence Number Wraps Around
  - OSPF V1:
    - Restrict LSAs to same semi-circle by regulating generation
    - But difficult to enforce with data corruption



  - OSPF V2:
    - Linear rather than circular space
    - Once sequence number reaches maximum, reset count to min
      - Flush out old sequence number by advertising LSA with MAXAGE
    - With 32-bit counter, doesn’t happen very often

OSPF – Load Balancing

- Extensions that allow multiple paths to be taken for the same destination when they have equal costs
  - "ECMP" – Equal Cost Multi-Path.
- Key issue: packet reordering
  - Hash on IP header to ensure packets of the same flow take the same path.
- Recent body of research
  - ECMP does not work well
  - E.g., Multiple large flows hashed to same path and might cause congestion.
  - New solutions based on SDNs.

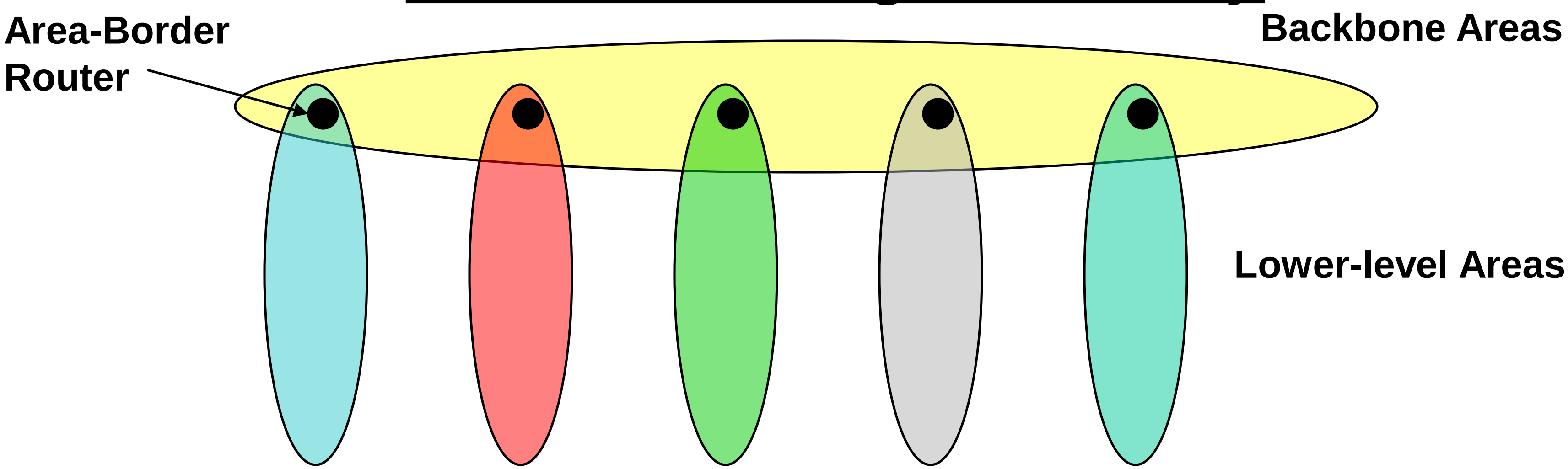
OSPF – Load Balancing

- Extensions that allow multiple paths to be taken for the same destination when they have equal costs
  - "ECMP" – Equal Cost Multi-Path.
- Key issue: packet reordering
  - Hash on IP header to ensure packets of the same flow take the same path.
- Recent body of research
  - ECMP does not work well
  - E.g., Multiple large flows hashed to same path and might cause congestion.
  - New solutions based on SDNs.

OSPF: Richer cost metrics

- How do we set costs?
  - Common recommendation: inversely proportional to link capacities
- Static cost or latency metrics not sufficient
  - Handling network congestion important
  - Traffic patterns might change over time
- Research in 2000's: (e.g., AT&T deployments)
  - Dynamic setting of weights in a manner that takes traffic demands into account
  - Changing weight settings: slower time-scale operation
- SDNs offer new opportunities.

OSPF Routing Hierarchy



- Partition Network into “Areas”
  - Router maintains link states of nodes within its area
  - Nodes in lower-level area use area-border router as default router
  - Backbone nodes can “summarize” routes within area

OSPF External Routes

- Limited Connectivity to Rest of Internet
  - Stub AS
    - Single border router
    - Can use border router as default for all addresses outside AS
  - Non-stub AS
    - External addresses need to be routed to appropriate border router
    - Can often summarize set of addresses by giving CIDR address

Emerging trends

- RIP Viewed as Outmoded
  - Good when networks small and routers had limited memory & computational power
  - Does not handle large networks with complex features
- OSPF Advantages
  - Fast convergence when configuration changes
  - Able to use more sophisticated metrics
- New trends
  - OSPF viewed overly complicated for more sophisticated traffic engineering tasks (load balancing, adjusting to congestion etc.)
  - Interest in SDNs to revisit routing