



Routing

Reading: Ch. 4.2, 4.3

Goals of Today's Lecture

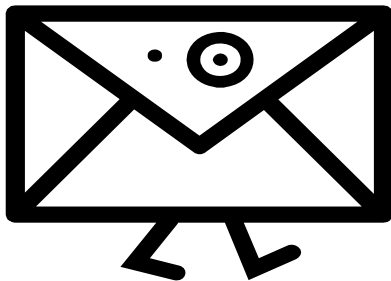
- **Link-state routing**
 - Dijkstra Algorithm
 - Open Shortest Path First Protocol (OSPF)
- **Distance-vector routing**
 - Bellman-Ford algorithm
 - Routing Information Protocol (RIP)
- **Path-vector routing**
 - Faster convergence than distance vector
 - More flexibility in selecting paths
- **Interdomain routing**
 - Autonomous Systems (AS)
 - Border Gateway Protocol (BGP)

What is Routing?

- A famous quotation from RFC 791

“A *name* indicates what we seek.
An *address* indicates where it is.
A *route* indicates how we get there.”

-- Jon Postel



Routing vs. Forwarding

- **Routing:**
 - Computing paths the packets will follow
 - Routers talking amongst themselves
 - Individual router *creating* a forwarding table
- **Forwarding:**
 - Directing a data packet to an outgoing link
 - Individual router *using* a forwarding table

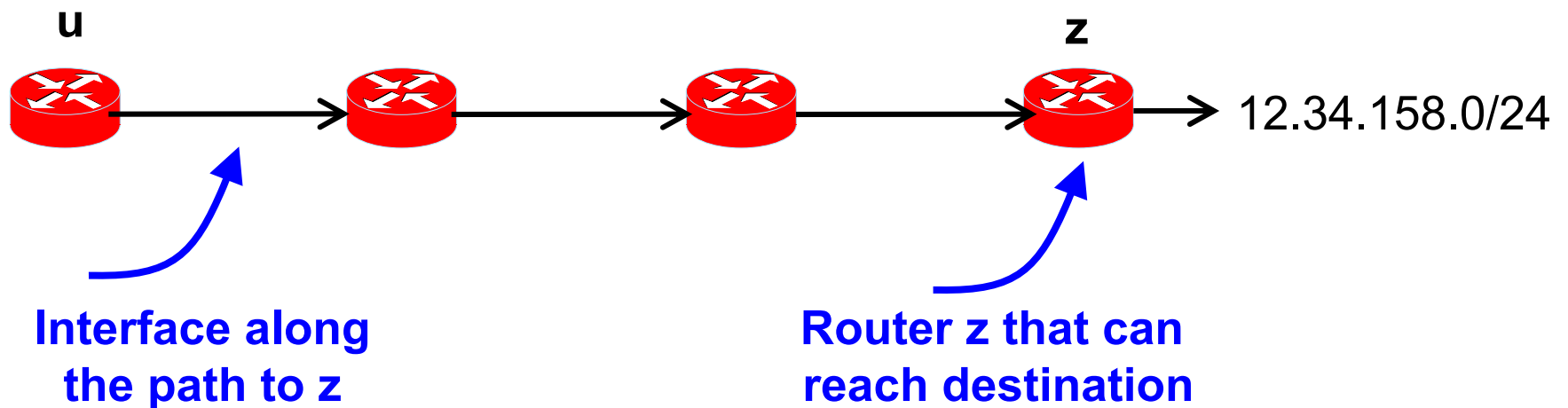


Where do Forwarding Tables Come From?

- Routers have forwarding tables
 - Map IP prefix to outgoing link(s)
- Entries can be statically configured
 - E.g., “map 12.34.158.0/24 to Serial0/0.1”
- But, this doesn't adapt
 - To failures
 - To new equipment
 - To the need to balance load
- That is where routing protocols come in

Computing Paths Between Routers

- Routers need to know
 - Which router to use to reach a destination prefix
 - Which outgoing interface to use to reach that router



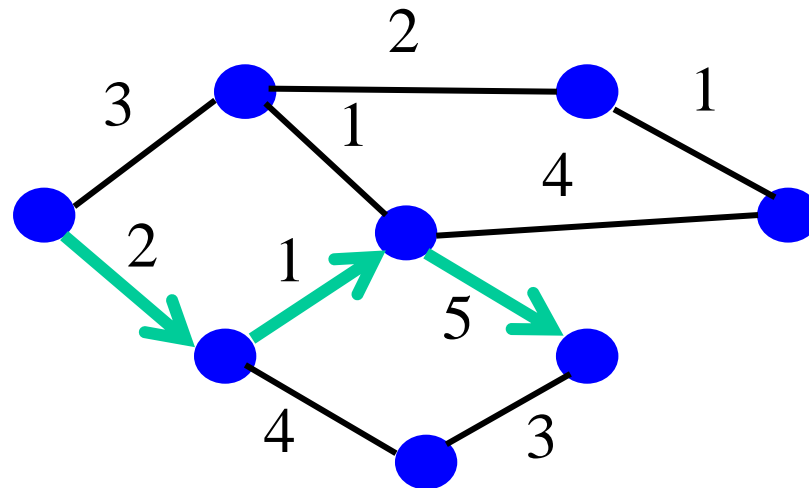
- Today's class: just how routers reach each other
 - How u knows how to forward packets toward z

Computing the Shortest Paths

(assuming you already know the topology)

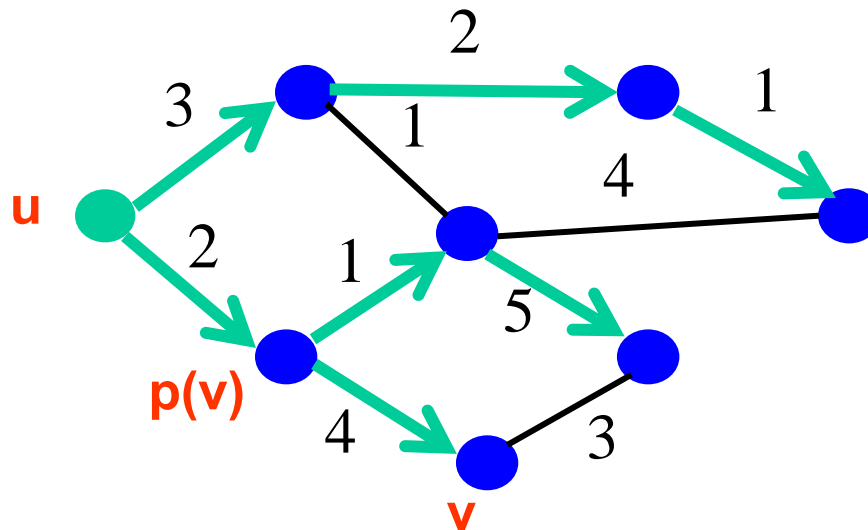
Shortest-Path Routing

- Path-selection model
 - Destination-based
 - Load-insensitive (e.g., static link weights)
 - Minimum hop count or sum of link weights



Shortest-Path Problem

- Given: network topology with link costs
 - $c(x,y)$: link cost from node x to node y
 - Infinity if x and y are not direct neighbors
- Compute: least-cost paths to all nodes
 - From a given source u to all other nodes
 - $p(v)$: predecessor node along path from source to v



Dijkstra's Shortest-Path Algorithm

- Iterative algorithm
 - After k iterations, know least-cost path to k nodes
- **S**: nodes whose least-cost path definitively known
 - Initially, $\mathbf{S} = \{\mathbf{u}\}$ where u is the source node
 - Add one node to S in each iteration
- **D(v)**: current cost of path from source to node v
 - Initially, $\mathbf{D}(\mathbf{v}) = \mathbf{c}(\mathbf{u}, \mathbf{v})$ for all nodes v adjacent to u
 - ... and $\mathbf{D}(\mathbf{v}) = \infty$ for all other nodes v
 - Continually update $D(v)$ as shorter paths are learned

Dijkstra's Algorithm

1 **Initialization:**

2 $S = \{u\}$

3 for all nodes v

4 if (v is adjacent to u)

5 $D(v) = c(u,v)$

6 else $D(v) = \infty$

7

8 **Loop**

9 find w not in S with the smallest $D(w)$

10 add w to S

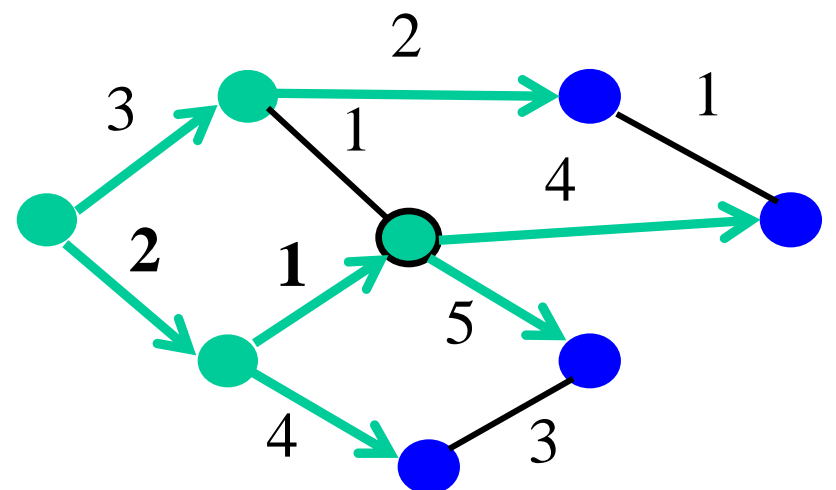
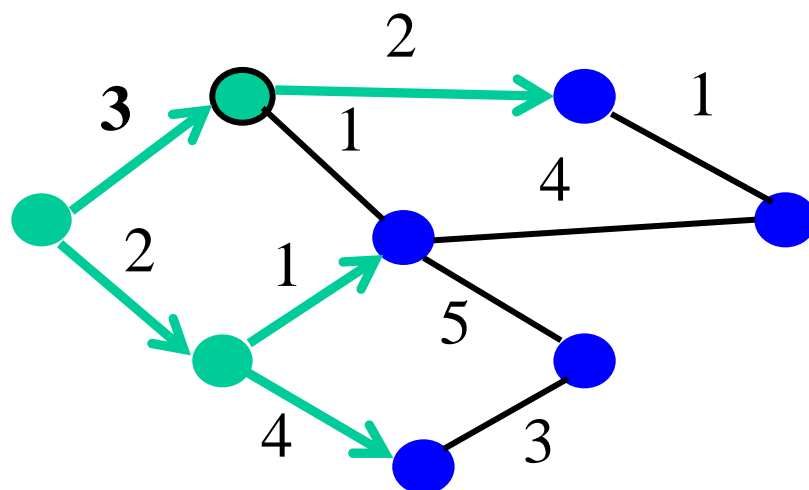
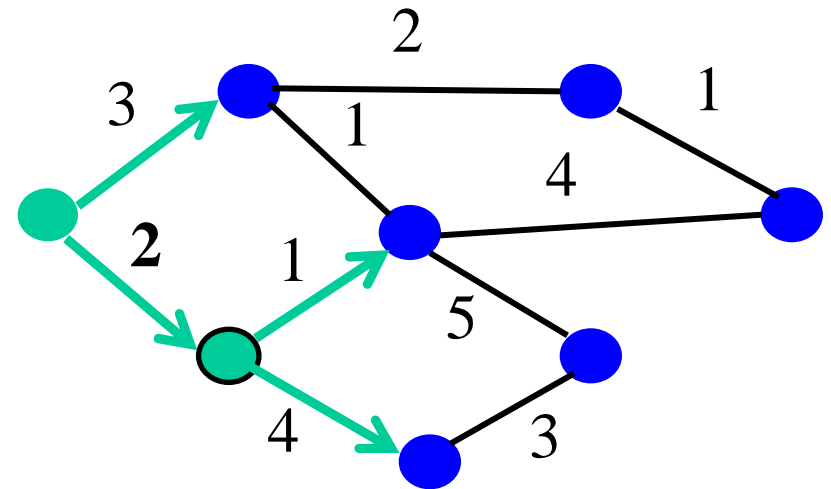
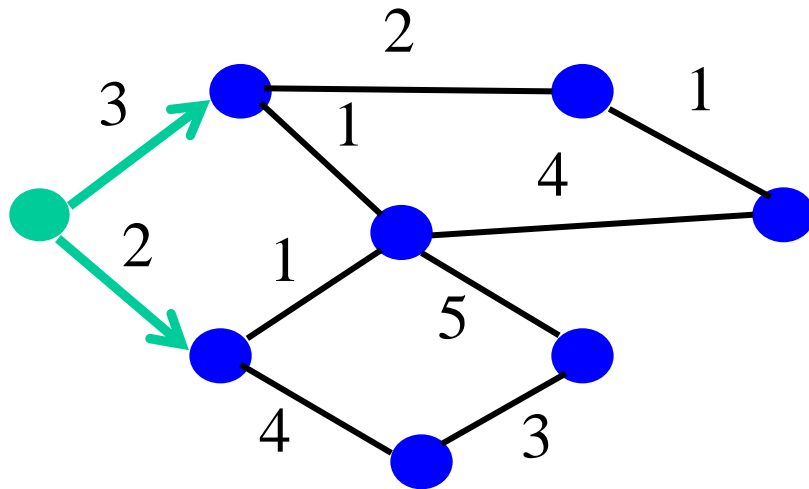
11 update $D(v)$ for all v adjacent to w and not in S :

12 $D(v) = \min\{D(v), D(w) + c(w,v)\}$

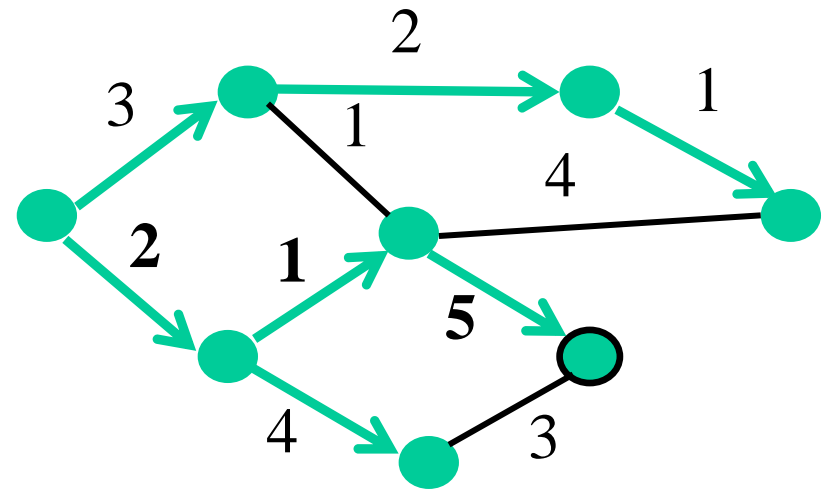
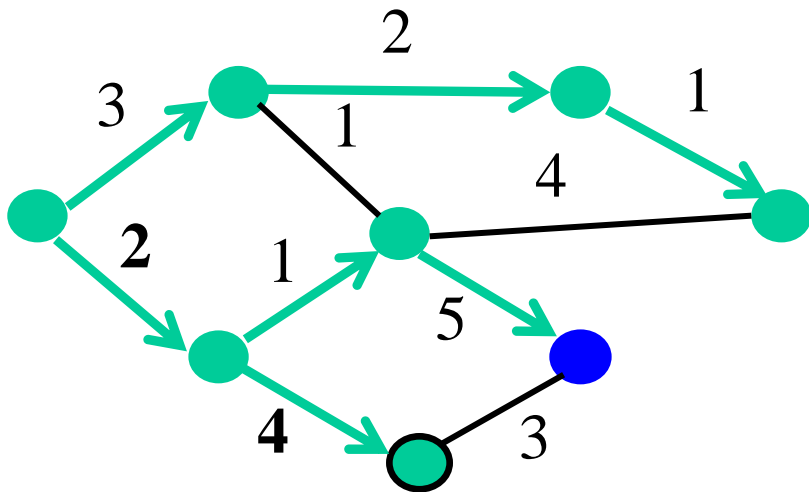
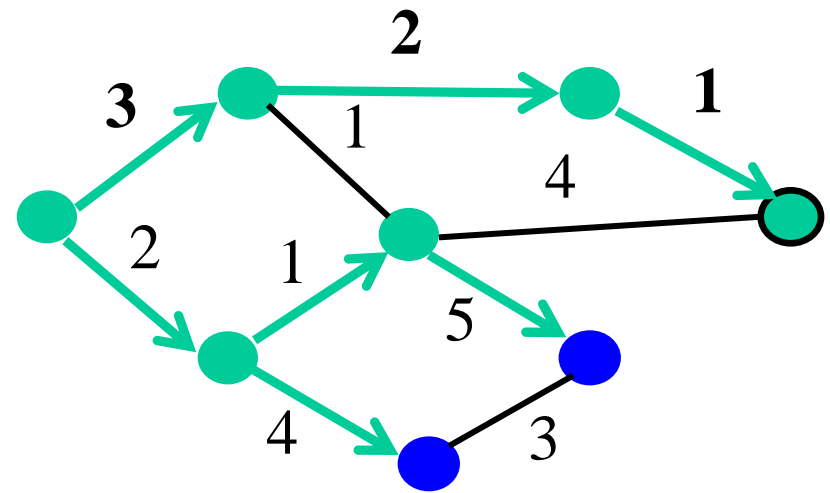
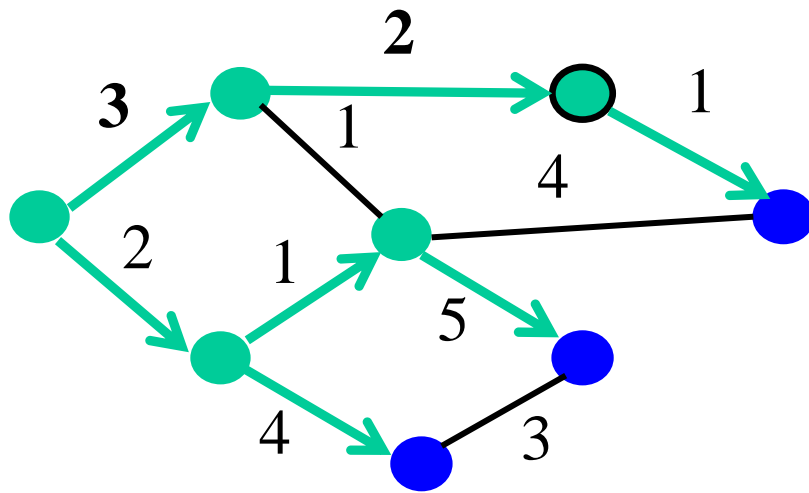
13 **until all nodes in S**



Dijkstra's Algorithm Example

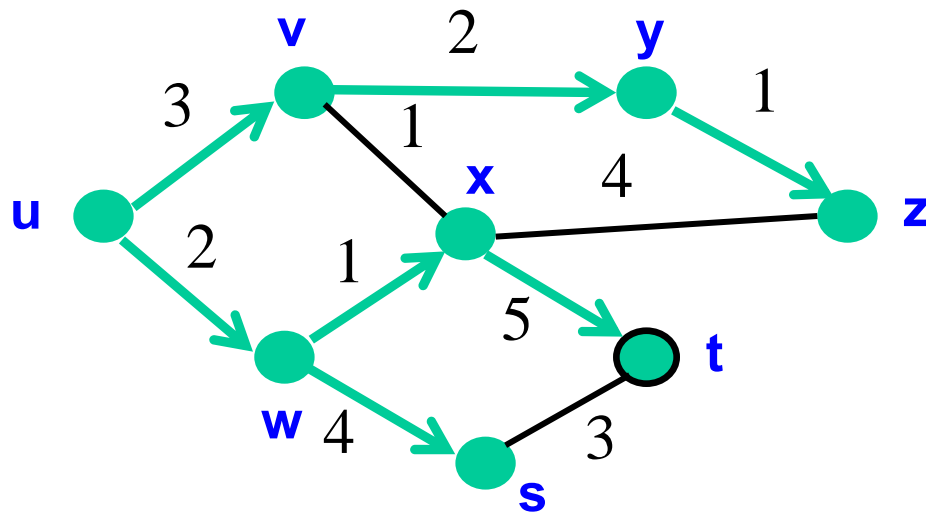


Dijkstra's Algorithm Example



Shortest-Path Tree

- Shortest-path tree from u
- Forwarding table at u



	link
v	(u,v)
w	(u,w)
x	(u,w)
y	(u,v)
z	(u,v)
s	(u,w)
t	(u,w)

Learning the Topology

(by the routers talking amongst themselves)

Link-State Routing

- Each router keeps track of its incident links
 - Whether the link is up or down
 - The cost on the link
- Each router broadcasts the link state
 - To give every router a complete view of the graph
- Each router runs Dijkstra's algorithm
 - To compute the shortest paths
 - ... and construct the forwarding table
- Example protocols
 - Open Shortest Path First (OSPF)
 - Intermediate System – Intermediate System (IS-IS)

Detecting Topology Changes

- **Beaconing**

- Periodic “hello” messages in both directions
- Detect a failure after a few missed “hellos”



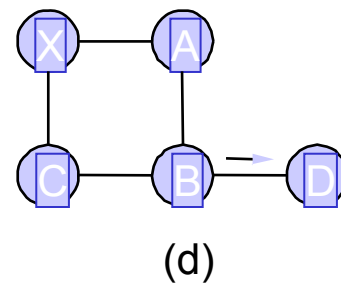
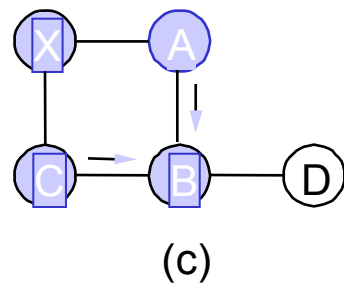
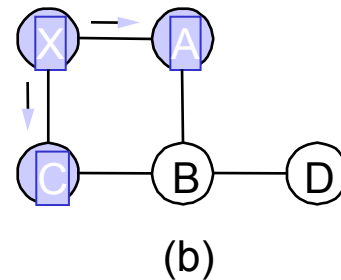
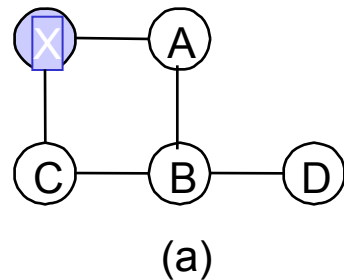
- **Performance trade-offs**

- Detection speed
- Overhead on link bandwidth and CPU
- Likelihood of false detection

Broadcasting the Link State

- Flooding

- Node sends link-state information out its links
- And then the next node sends out all of its links
- ... except the one where the information arrived



When to Initiate Flooding

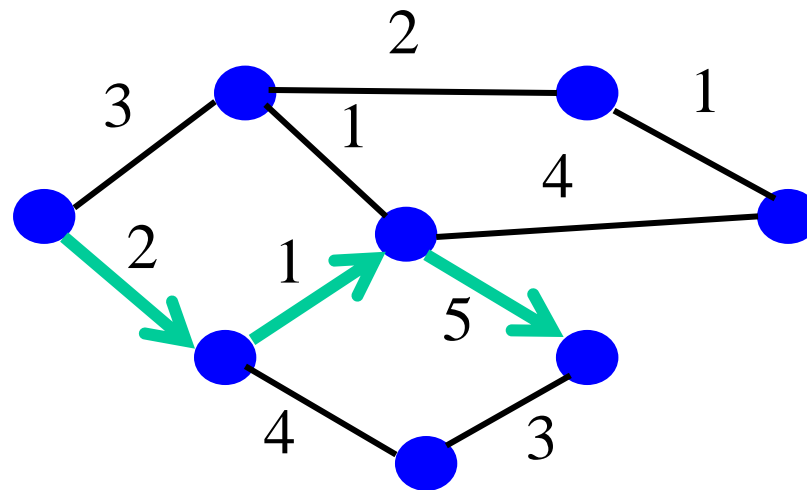
- **Topology change**
 - Link or node failure
 - Link or node recovery
- **Configuration change**
 - Link cost change
- **Periodically**
 - Refresh the link-state information
 - Typically (say) 30 minutes
 - Corrects for possible corruption of the data

When the Routers Disagree

(during transient periods)

Convergence

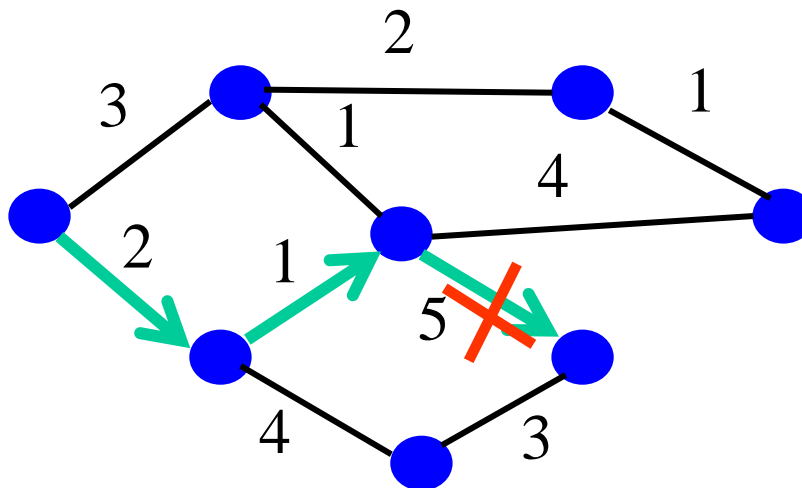
- Getting consistent routing information to all nodes
 - E.g., all nodes having the same link-state database
- Consistent forwarding after convergence
 - All nodes have the same link-state database
 - All nodes forward packets on shortest paths
 - The next router on the path forwards to the next hop



Transient Disruptions

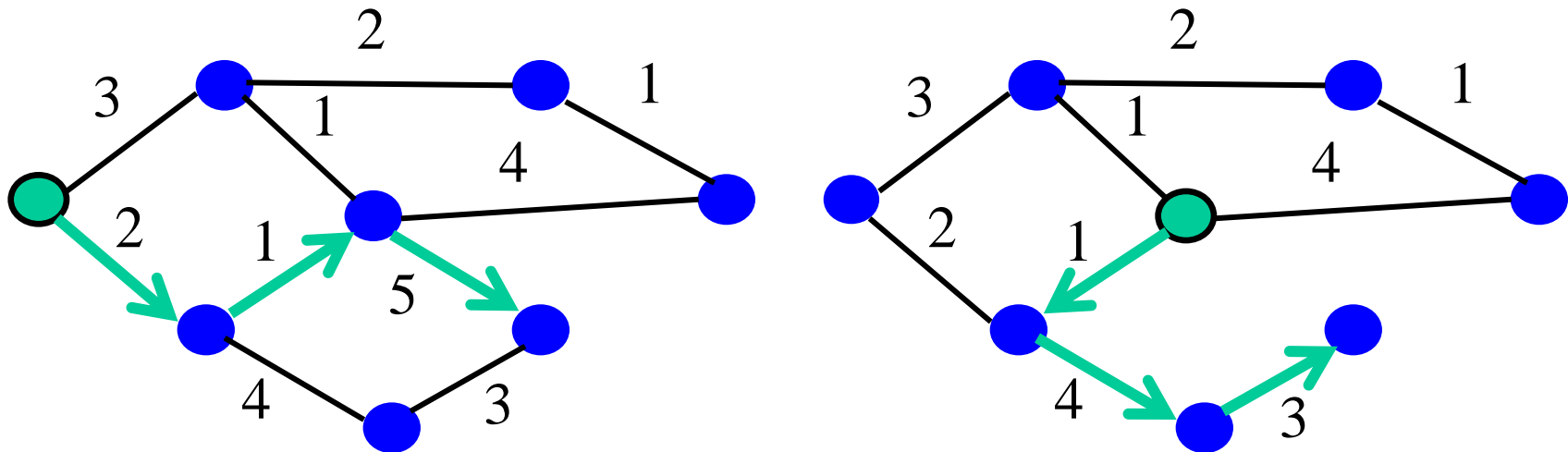
- Detection delay

- A node does not detect a failed link immediately
- ... and forwards data packets into a “blackhole”
- Depends on timeout for detecting lost hellos



Transient Disruptions

- Inconsistent link-state database
 - Some routers know about failure before others
 - The shortest paths are no longer consistent
 - Can cause transient forwarding loops



Convergence Delay

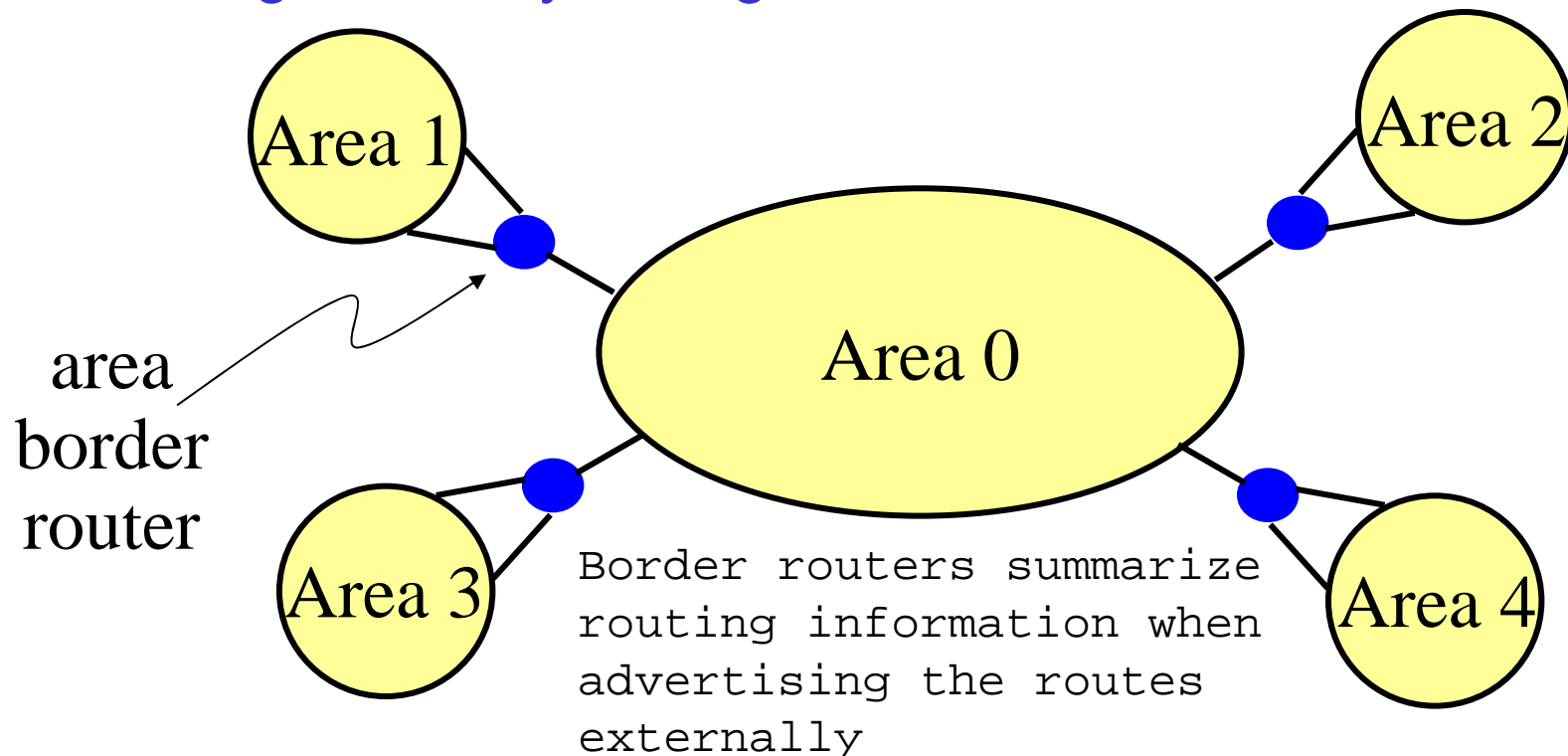
- Sources of convergence delay
 - Detection latency
 - Flooding of link-state information
 - Shortest-path computation
 - Creating the forwarding table
- Performance during convergence period
 - Lost packets due to blackholes and TTL expiry
 - Looping packets consuming resources
 - Out-of-order packets reaching the destination
- Very bad for VoIP, online gaming, and video

Reducing Convergence Delay

- **Faster detection**
 - Smaller hello timers
 - Link-layer technologies that can detect failures
- **Faster flooding**
 - Flooding immediately
 - Sending link-state packets with high-priority
- **Faster computation**
 - Faster processors on the routers
 - Incremental Dijkstra's algorithm
- **Faster forwarding-table update**
 - Data structures supporting incremental updates

Scaling Link-State Routing

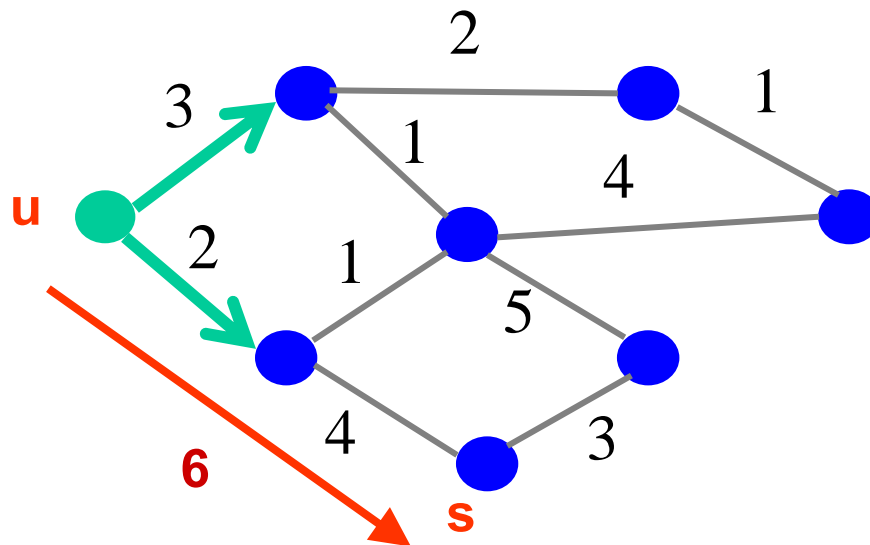
- Overhead of link-state routing
 - Flooding link-state packets throughout the network
 - Running Dijkstra's shortest-path algorithm
- Introducing hierarchy through "areas"



Distance-Vector Routing

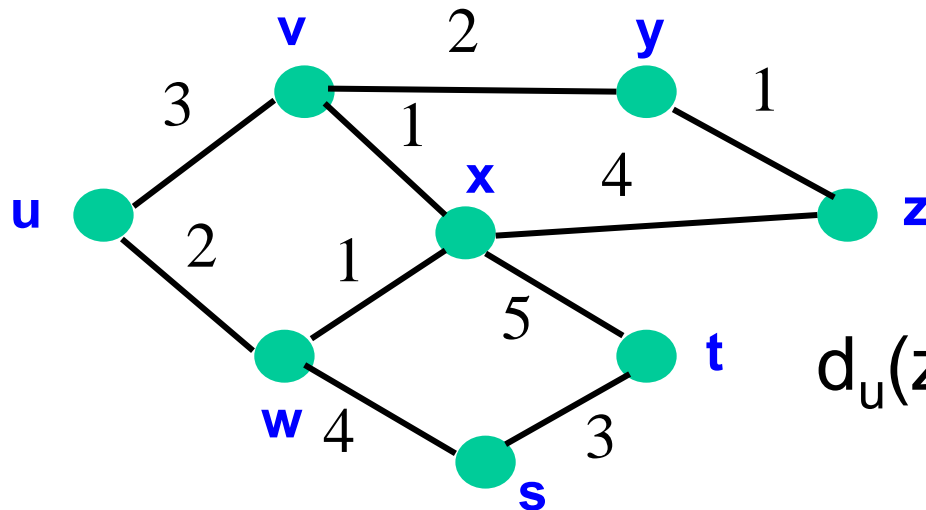
Review: Shortest-Path Problem

- 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



Bellman-Ford Algorithm

- 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



$$d_u(z) = \min\{c(u,v) + d_v(z), \\ c(u,w) + d_w(z)\}$$

Distance Vector Algorithm

- $c(x,v)$ = cost for direct link from x to v
 - Node x maintains costs of direct links $c(x,v)$
- $D_x(y)$ = estimate of least cost from x to y
 - Node x maintains distance vector $\mathbf{D}_x = [D_x(y): y \in N]$
- Node x maintains its neighbors' distance vectors
 - For each neighbor v , x maintains $\mathbf{D}_v = [D_v(y): y \in N]$
- Each node v periodically sends \mathbf{D}_v to its neighbors
 - And neighbors update their own distance vectors
 - $D_x(y) \leftarrow \min_v \{c(x,v) + D_v(y)\}$ for each node $y \in N$
- Over time, the distance vector \mathbf{D}_x converges

Distance Vector Algorithm

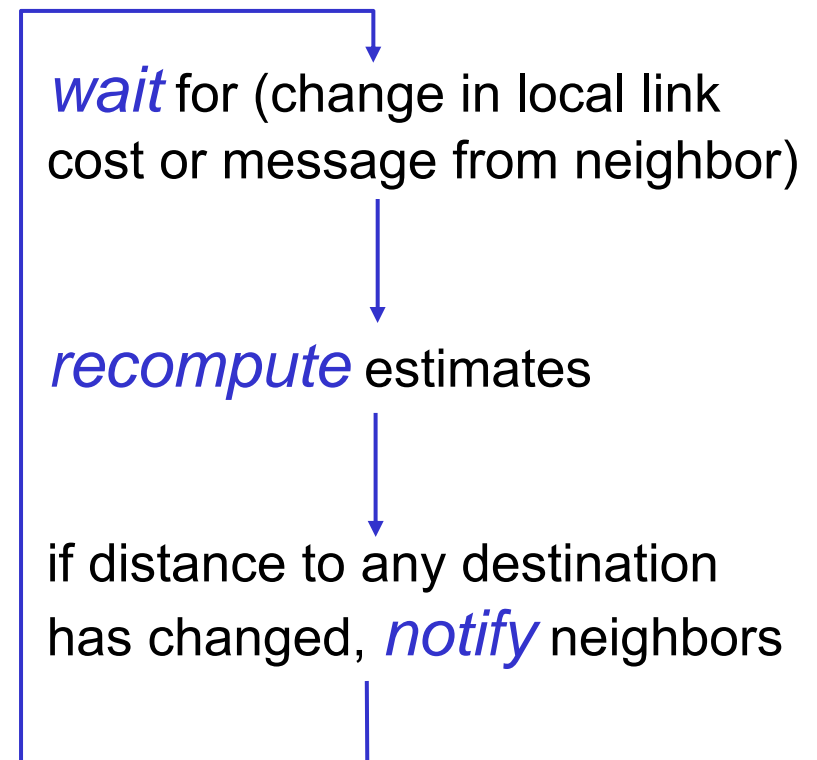
Iterative, asynchronous: each local iteration caused by:

- Local link cost change
- Distance vector update message from neighbor

Distributed:

- Each node notifies neighbors *only* when its DV changes
- Neighbors then notify their neighbors if necessary

Each node:



Distance Vector Example: Step 1

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

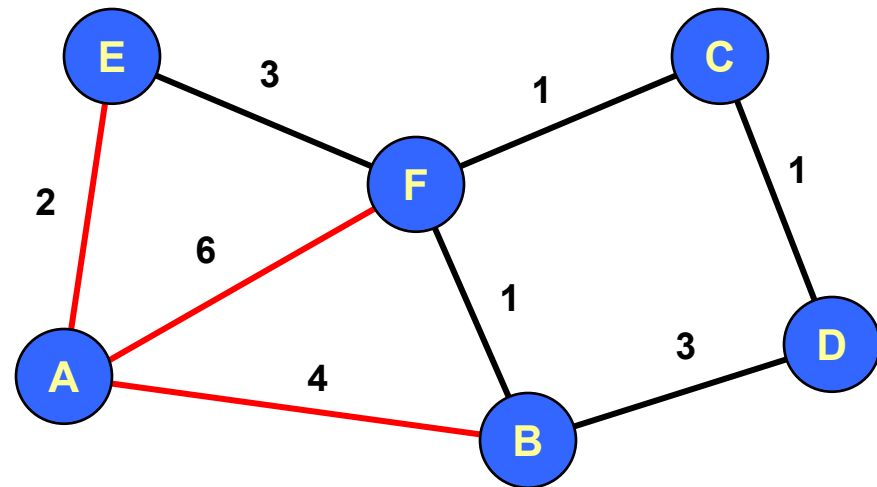


Table for C			Table for D			Table for E			Table for F		
Dst	Cst	Hop	Dst	Cst	Hop	Dst	Cst	Hop	Dst	Cst	Hop
A	∞	-	A	∞	-	A	2	A	A	6	A
B	∞	-	B	3	B	B	∞	-	B	1	B
C	0	C	C	1	C	C	∞	-	C	1	C
D	1	D	D	0	D	D	∞	-	D	∞	-
E	∞	-	E	∞	-	E	0	E	E	3	E
F	1	F	F	∞	-	F	3	F	F	0	F

Distance Vector Example: Step 2

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

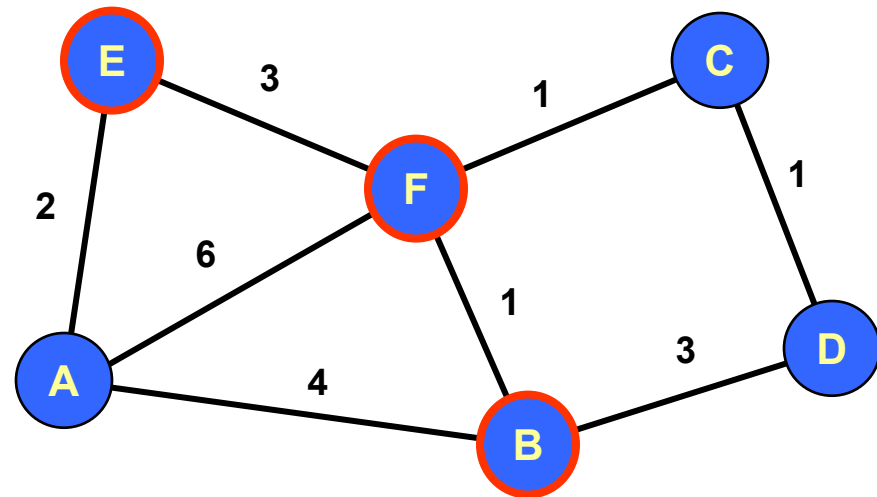


Table for C			Table for D			Table for E			Table for F		
Dst	Cst	Hop	Dst	Cst	Hop	Dst	Cst	Hop	Dst	Cst	Hop
A	7	F	A	7	B	A	2	A	A	5	B
B	2	F	B	3	B	B	4	F	B	1	B
C	0	C	C	1	C	C	4	F	C	1	C
D	1	D	D	0	D	D	∞	-	D	2	C
E	4	F	E	∞	-	E	0	E	E	3	E
F	1	F	F	2	C	F	3	F	F	0	F

Distance Vector Example: Step 3

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

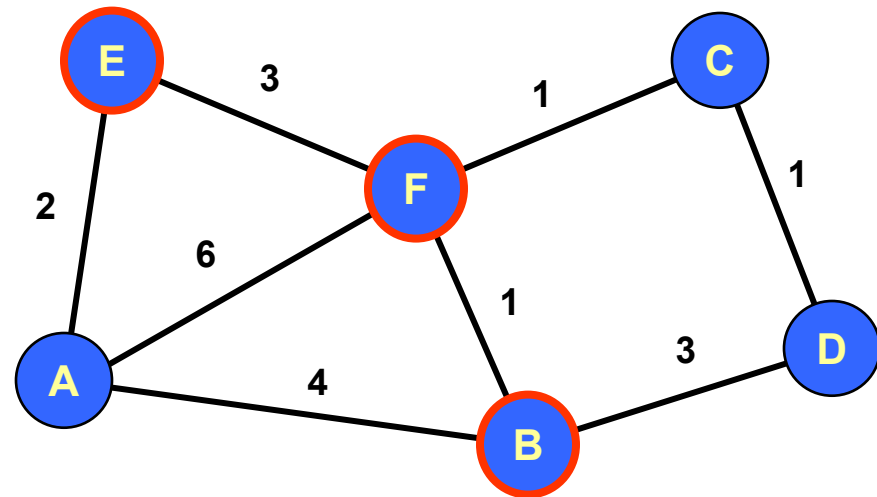
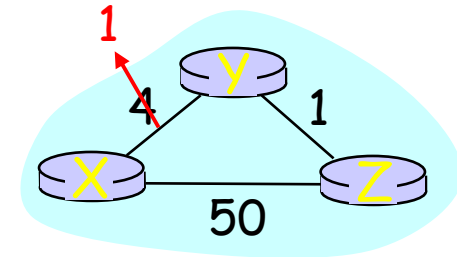


Table for C			Table for D			Table for E			Table for F		
Dst	Cst	Hop	Dst	Cst	Hop	Dst	Cst	Hop	Dst	Cst	Hop
A	6	F	A	7	B	A	2	A	A	5	B
B	2	F	B	3	B	B	4	F	B	1	B
C	0	C	C	1	C	C	4	F	C	1	C
D	1	D	D	0	D	D	5	F	D	2	C
E	4	F	E	5	C	E	0	E	E	3	E
F	1	F	F	2	C	F	3	F	F	0	F

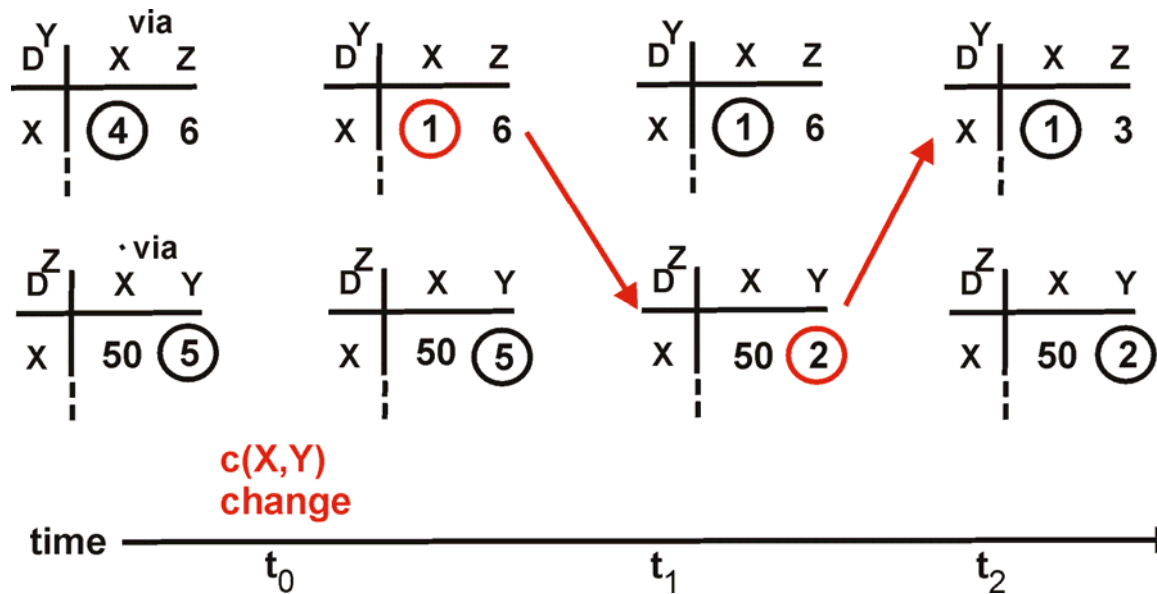
Distance Vector: Link Cost Changes

Link cost changes:

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



“good news travels fast”

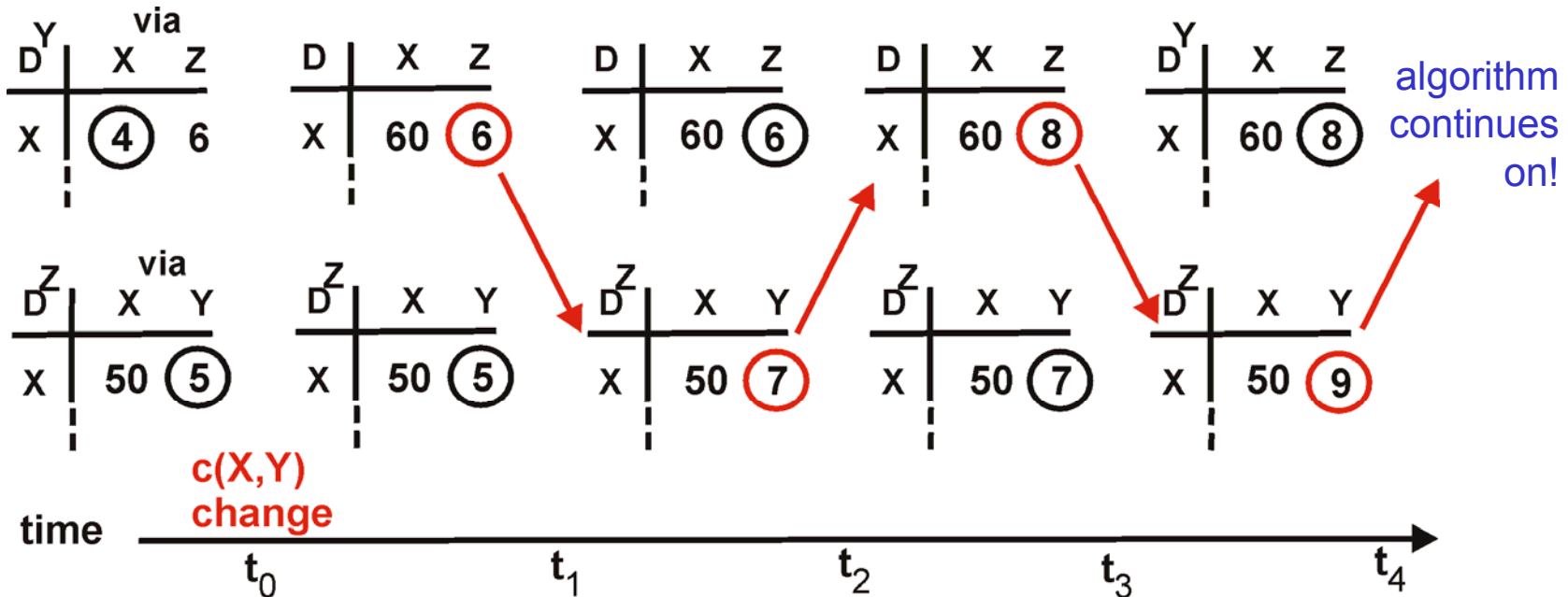
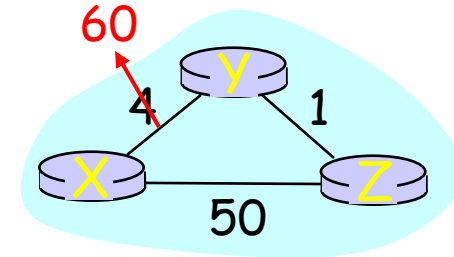


algorithm terminates

Distance Vector: Link Cost Changes

Link cost changes:

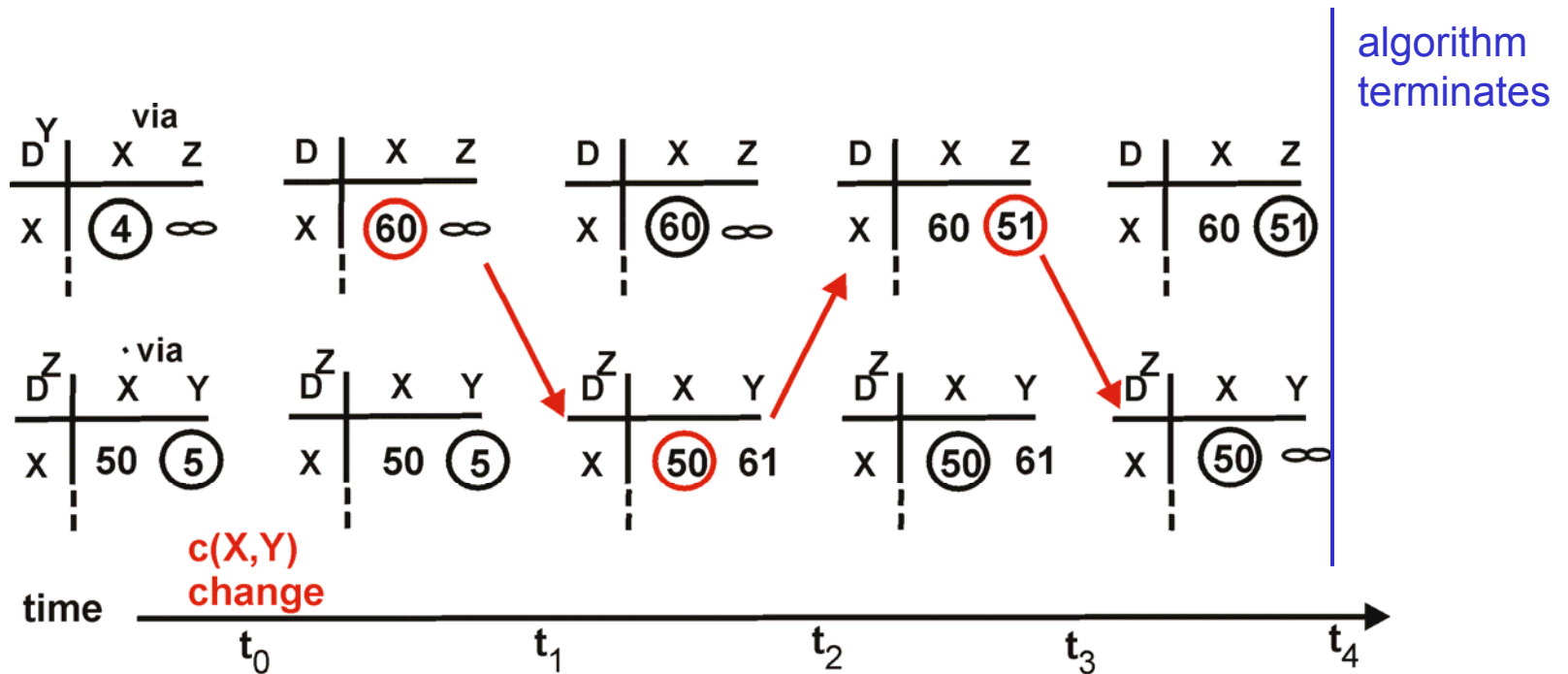
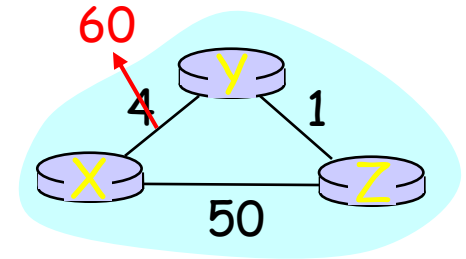
- Good news travels fast
- Bad news travels slow - “count to infinity” problem!



Distance Vector: Poison Reverse

If Y routes through Z to get to X :

- Y tells Z its (Y's) distance to X is infinite (so Z won't route to X via Y)
- Still, can have problems when more than 2 routers are involved



Routing Information Protocol (RIP)

- Distance vector protocol
 - Nodes send distance vectors every 30 seconds
 - ... or, when an update causes a change in routing
- Link costs in RIP
 - All links have cost 1
 - Valid distances of 1 through 15
 - ... with 16 representing infinity
 - Small “infinity” → smaller “counting to infinity” problem
- RIP is limited to fairly small networks
 - E.g., used in the Princeton campus network

Comparison of LS and DV Routing

Message complexity

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

Speed of Convergence

- LS: relatively fast
- DV: convergence time varies
 - May be routing loops
 - Count-to-infinity problem

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 (error propagates)

Similarities of LS and DV Routing

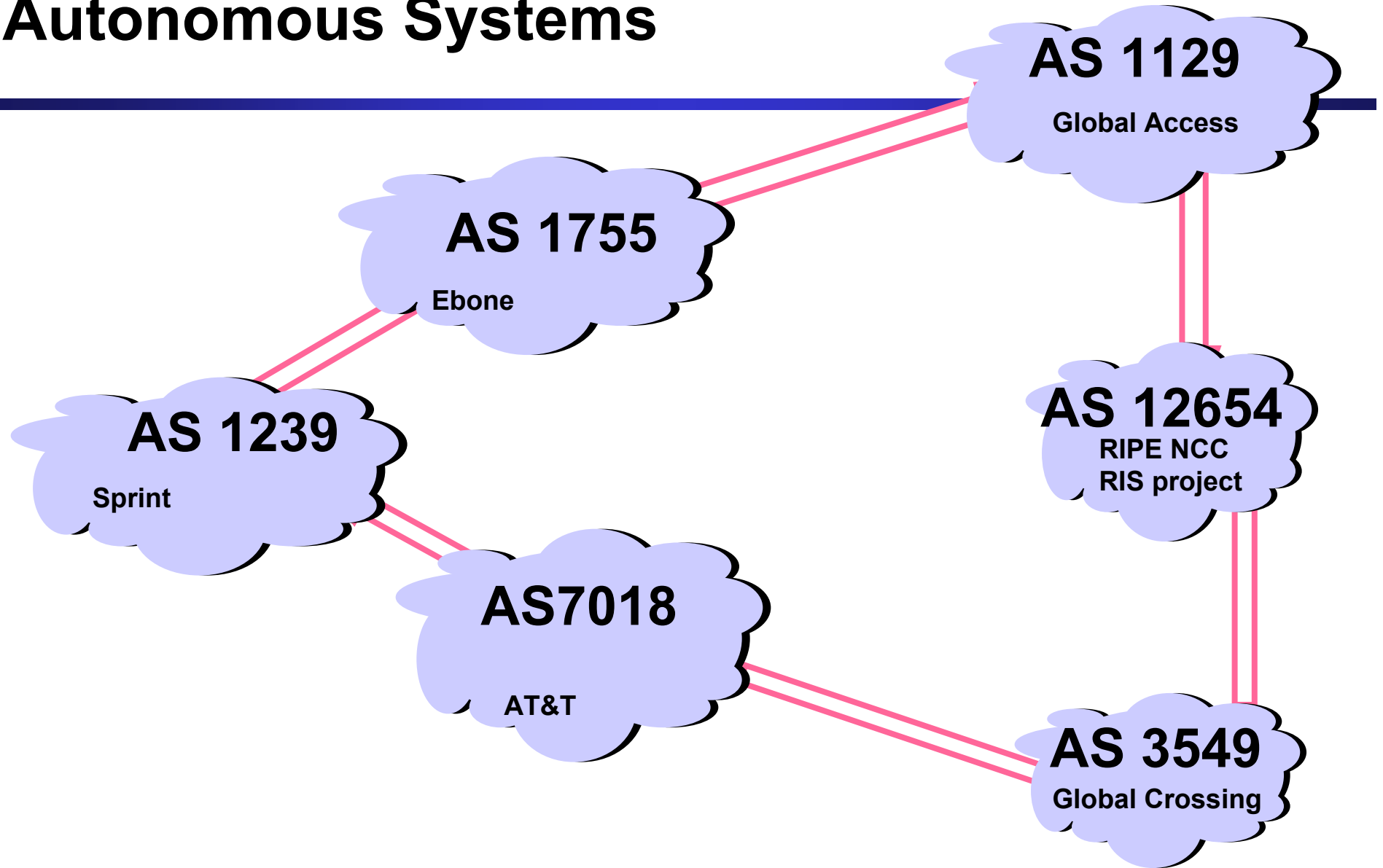
- Shortest-path routing
 - Metric-based, using link weights
 - Routers share a common view of how good a path is
- As such, commonly used *inside* an organization
 - RIP and OSPF are mostly used as *intradomain* protocols
 - E.g., Princeton uses RIP, and AT&T uses OSPF
- But the Internet is a “network of networks”
 - How to stitch the many networks together?
 - When networks may not have common goals
 - ... and may not want to share information

Interdomain Routing and Autonomous Systems (ASes)

Interdomain Routing

- Internet is divided into Autonomous Systems
 - Distinct regions of administrative control
 - Routers/links managed by a single “institution”
 - Service provider, company, university, ...
- Hierarchy of Autonomous Systems
 - Large, tier-1 provider with a nationwide backbone
 - Medium-sized regional provider with smaller backbone
 - Small network run by a single company or university
- Interaction between Autonomous Systems
 - Internal topology is not shared between ASes
 - ... but, neighboring ASes interact to coordinate routing

Autonomous Systems



Autonomous System Numbers

AS Numbers are 16 bit values.

Currently over 20,000 in use.

- Villanova: 10448
- MIT: 3
- Harvard: 11
- Yale: 29
- Princeton: 88
- AT&T: 7018, 6341, 5074, ...
- UUNET: 701, 702, 284, 12199, ...
- Sprint: 1239, 1240, 6211, 6242, ...
- ...

whois -h whois.arin.net as10448

OrgName: Villanova University
OrgID: VILLAN
Address: 800 Lancaster Avenue
City: Villanova
StateProv: PA
PostalCode: 19085
Country: US

ASNumber: 10448
ASName: VILLANOVA-UNIV
ASHandle: AS10448
Comment:
RegDate: 1997-08-06
Updated: 2001-07-11

RTechHandle: ZV26-ARIN
RTechName: Villanova University
RTechPhone: +1-610-519-4400
RTechEmail: hostmaster@villanova.edu

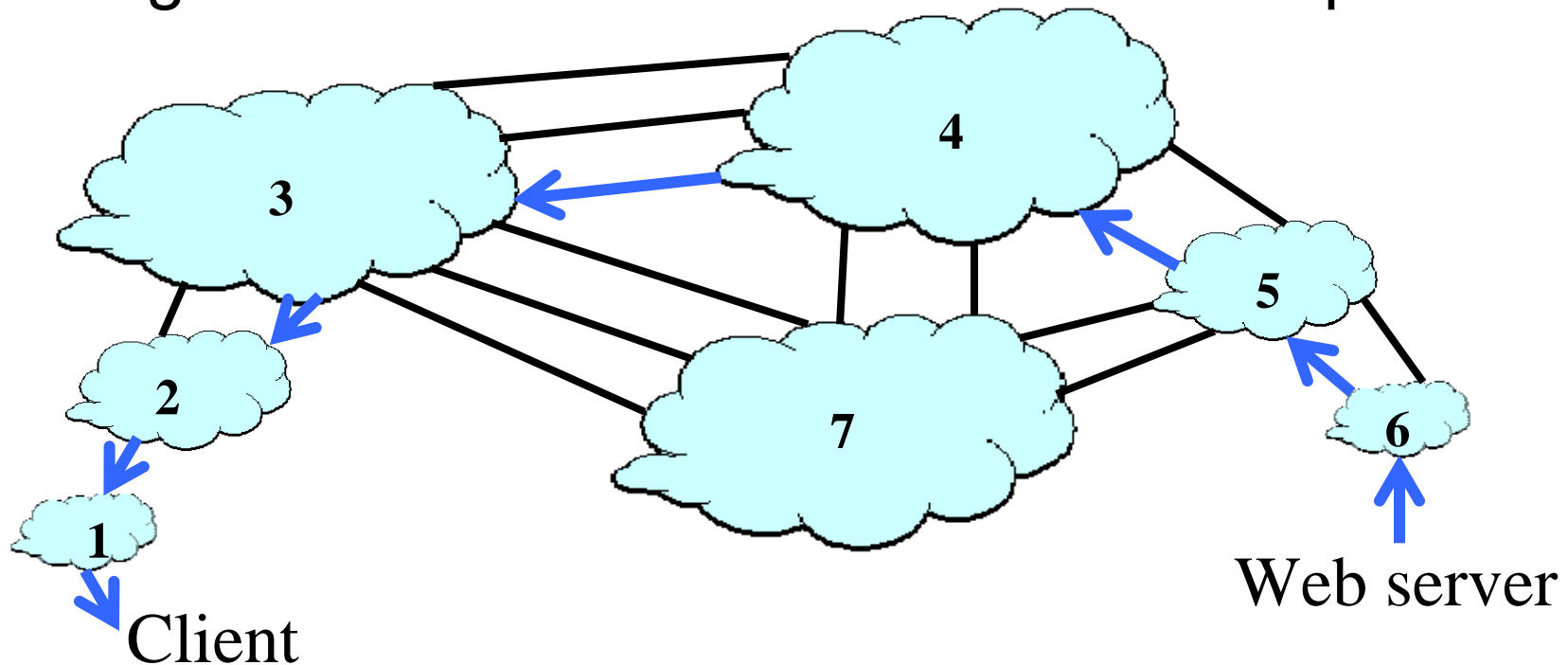
...

AS Number Trivia

- AS number is a 16-bit quantity
 - So, 65,536 unique AS numbers
- Some are reserved (e.g., for private AS numbers)
 - So, only 64,510 are available for public use
- Managed by Internet Assigned Numbers Authority
 - Gives blocks of 1024 to Regional Internet Registries
 - IANA has allocated 39,934 AS numbers to RIRs (Jan'06)
- RIRs assign AS numbers to institutions
 - RIRs have assigned 34,827 (Jan'06)
 - Only 21,191 are visible in interdomain routing (Jan'06)
- Recently started assigning 32-bit AS #s (2007)

Interdomain Routing

- AS-level topology
 - Destinations are IP prefixes (e.g., 12.0.0.0/8)
 - Nodes are Autonomous Systems (ASes)
 - Edges are links and business relationships



Challenges for Interdomain Routing

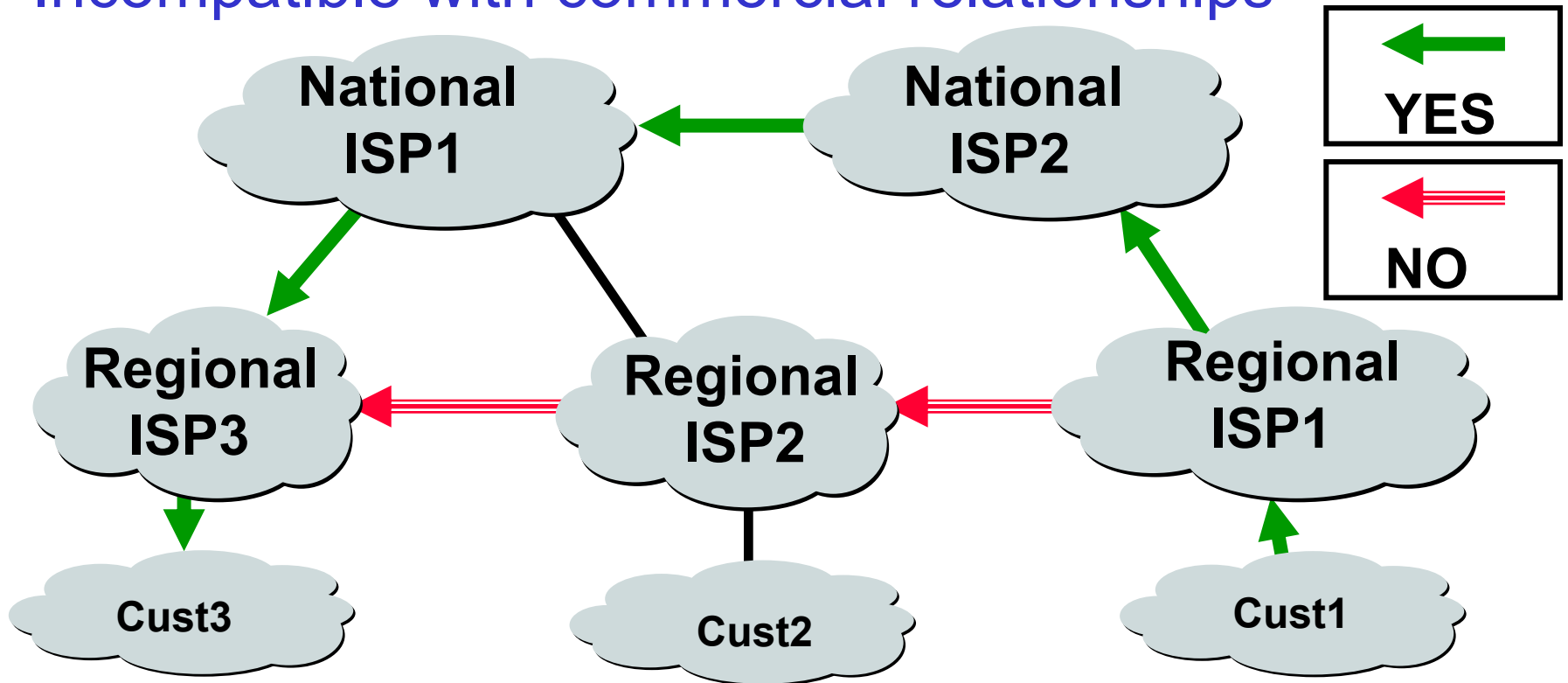
- **Scale**
 - ASes: 20,000+ visible ones, and 40K allocated
 - Routers: at least in the millions...
- **Privacy**
 - ASes don't want to divulge internal topologies
 - ... or their business relationships with neighbors
- **Policy**
 - No Internet-wide notion of a link cost metric
 - Need control over where you send traffic
 - ... and who can send traffic through you

Path-Vector Routing

Slides by Rexford @ Princeton, slightly altered by M.D.

Shortest-Path Routing is Restrictive

- All traffic must travel on shortest paths
- All nodes need common notion of link costs
- Incompatible with commercial relationships



Link-State Routing is Problematic

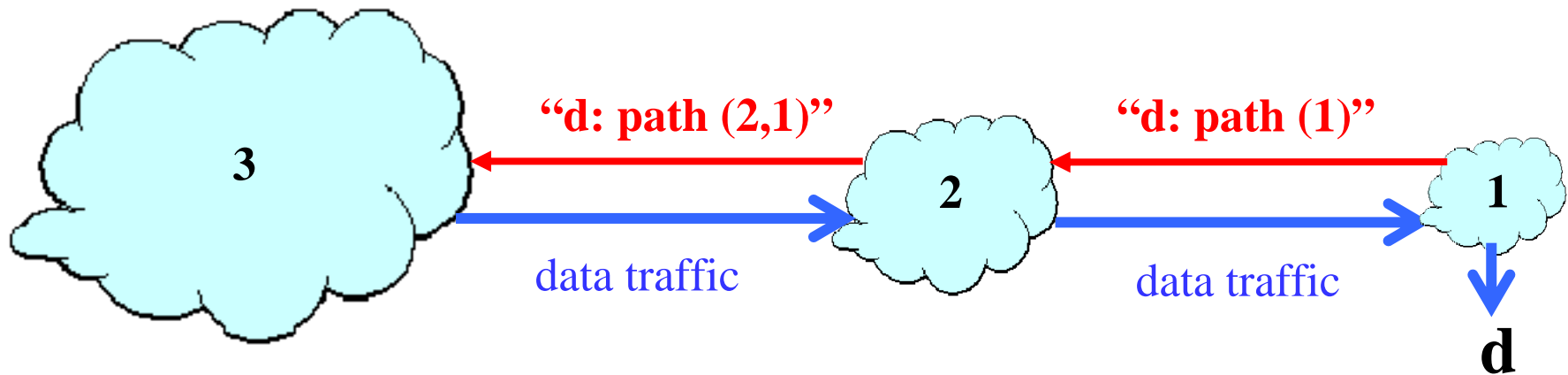
- Topology information is flooded
 - High bandwidth and storage overhead
 - Forces nodes to divulge sensitive information
- Entire path computed locally per node
 - High processing overhead in a large network
- Minimizes some notion of total distance
 - Works only if policy is shared and uniform
- Typically used only inside an AS
 - E.g., OSPF and IS-IS

Distance Vector is on the Right Track

- **Advantages**
 - Hides details of the network topology
 - Nodes determine only “next hop” toward the dest
- **Disadvantages**
 - Slow convergence due to the counting-to-infinity problem (“bad news travels slowly”)
- **Idea: extend the notion of a distance vector**
 - To make it easier to detect loops

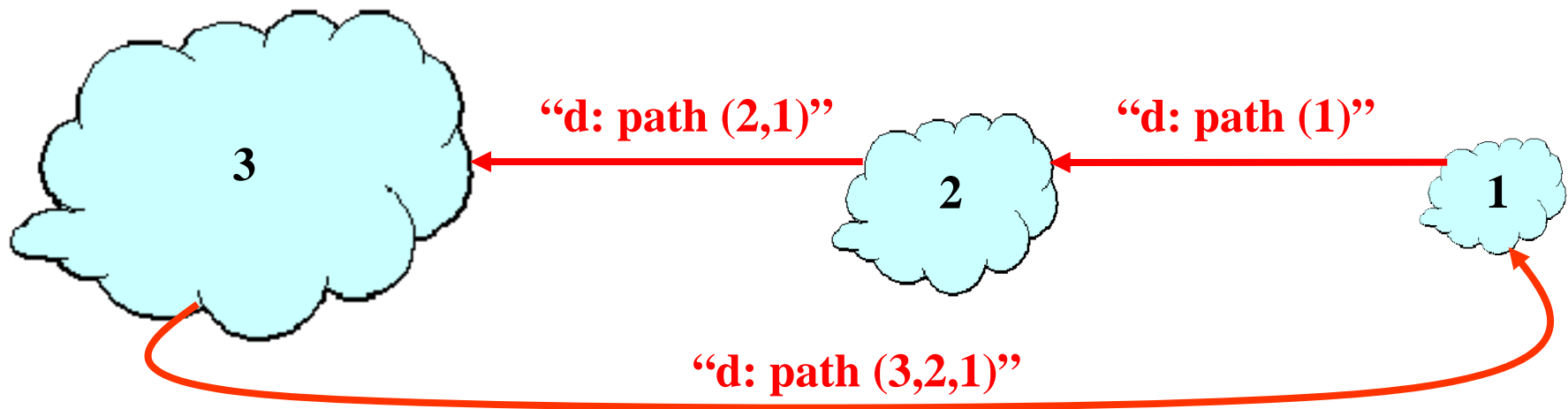
Path-Vector Routing

- Extension of distance-vector routing
 - Support flexible routing policies
 - Avoid count-to-infinity problem
- Key idea: advertise the entire path
 - Distance vector: send *distance metric* per dest d
 - Path vector: send the *entire path* for each dest d



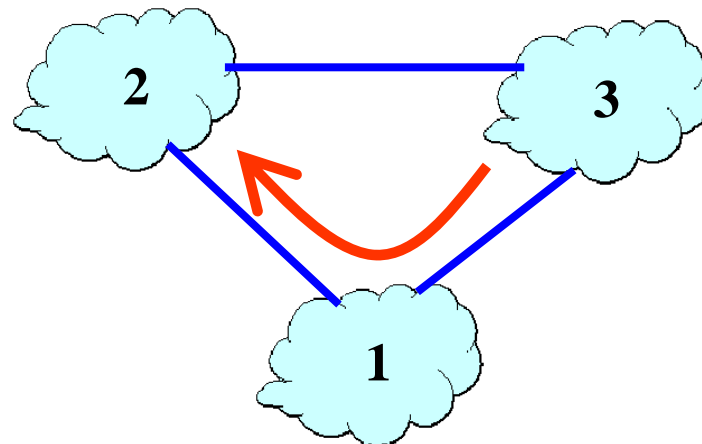
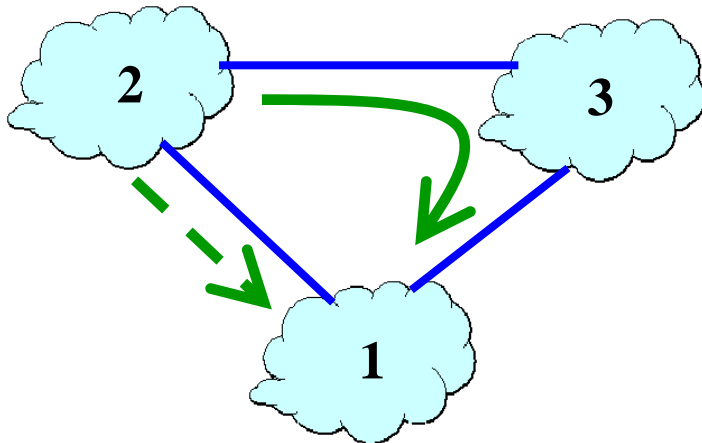
Faster Loop Detection

- Node can easily detect a loop
 - Look for its own node identifier in the path
 - E.g., node 1 sees itself in the path “3, 2, 1”
- Node can simply discard paths with loops
 - E.g., node 1 simply discards the advertisement



Flexible Policies

- Each node can apply local policies
 - Path selection: Which path to use?
 - Path export: Which paths to advertise?
- Examples
 - Node 2 may prefer the path “2, 3, 1” over “2, 1”
 - Node 1 may not let node 3 hear the path “1, 2”

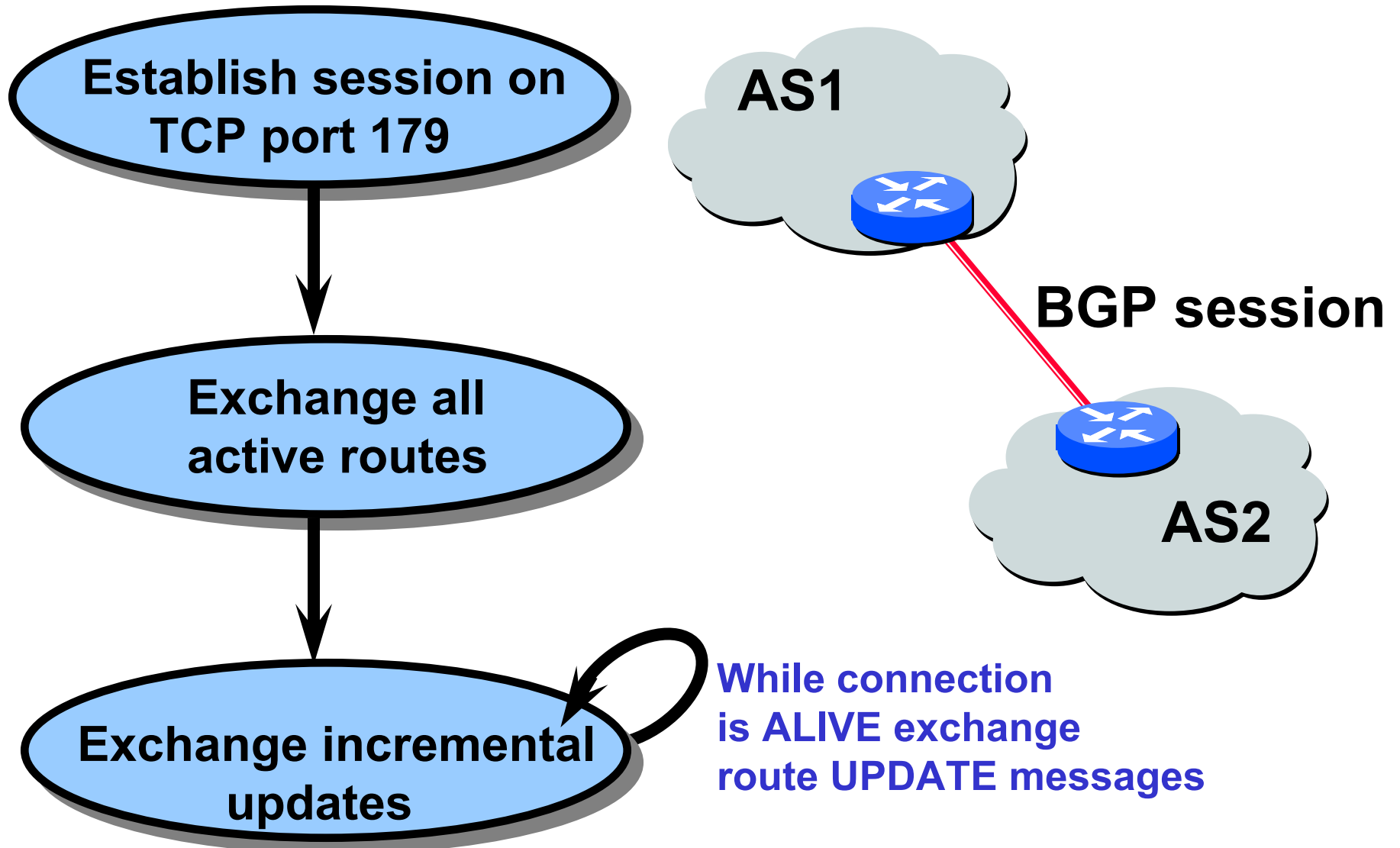


Border Gateway Protocol (BGP)

Border Gateway Protocol

- Interdomain routing protocol for the Internet
 - Prefix-based path-vector protocol
 - Policy-based routing based on AS Paths

BGP Operations

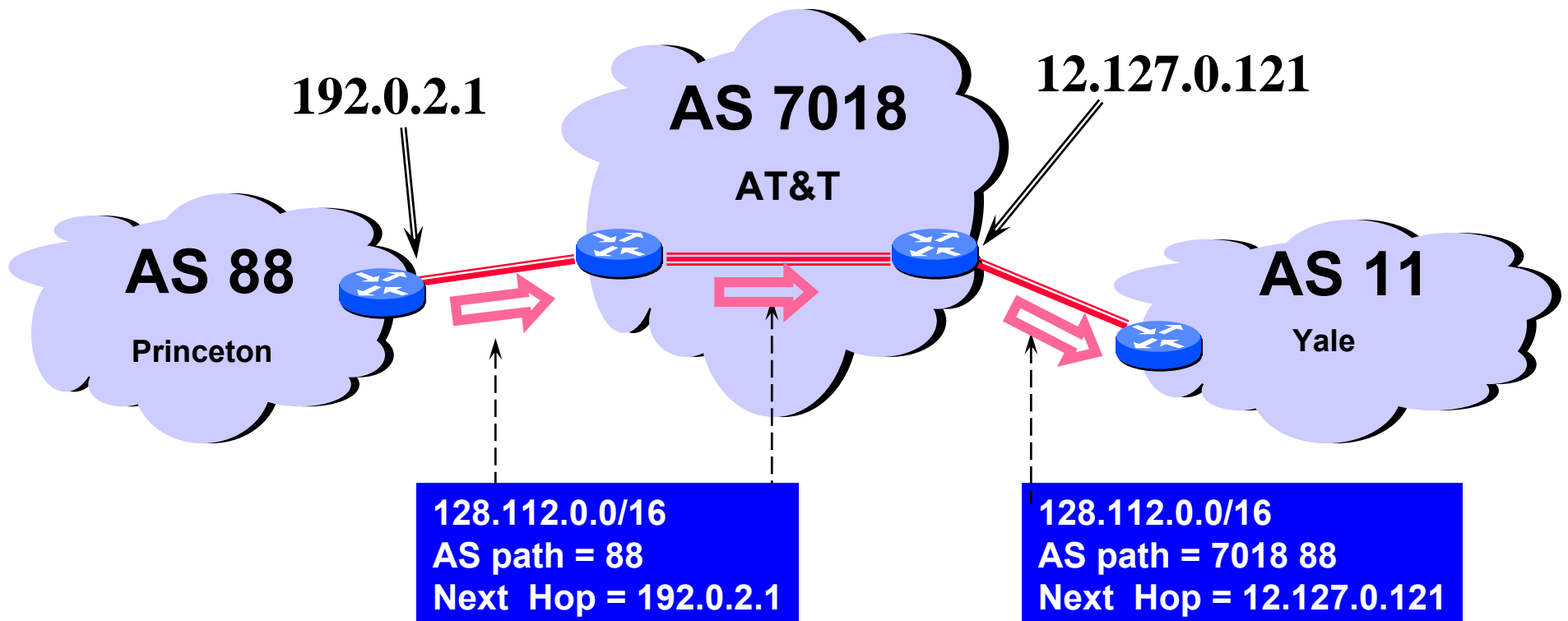


Incremental Protocol

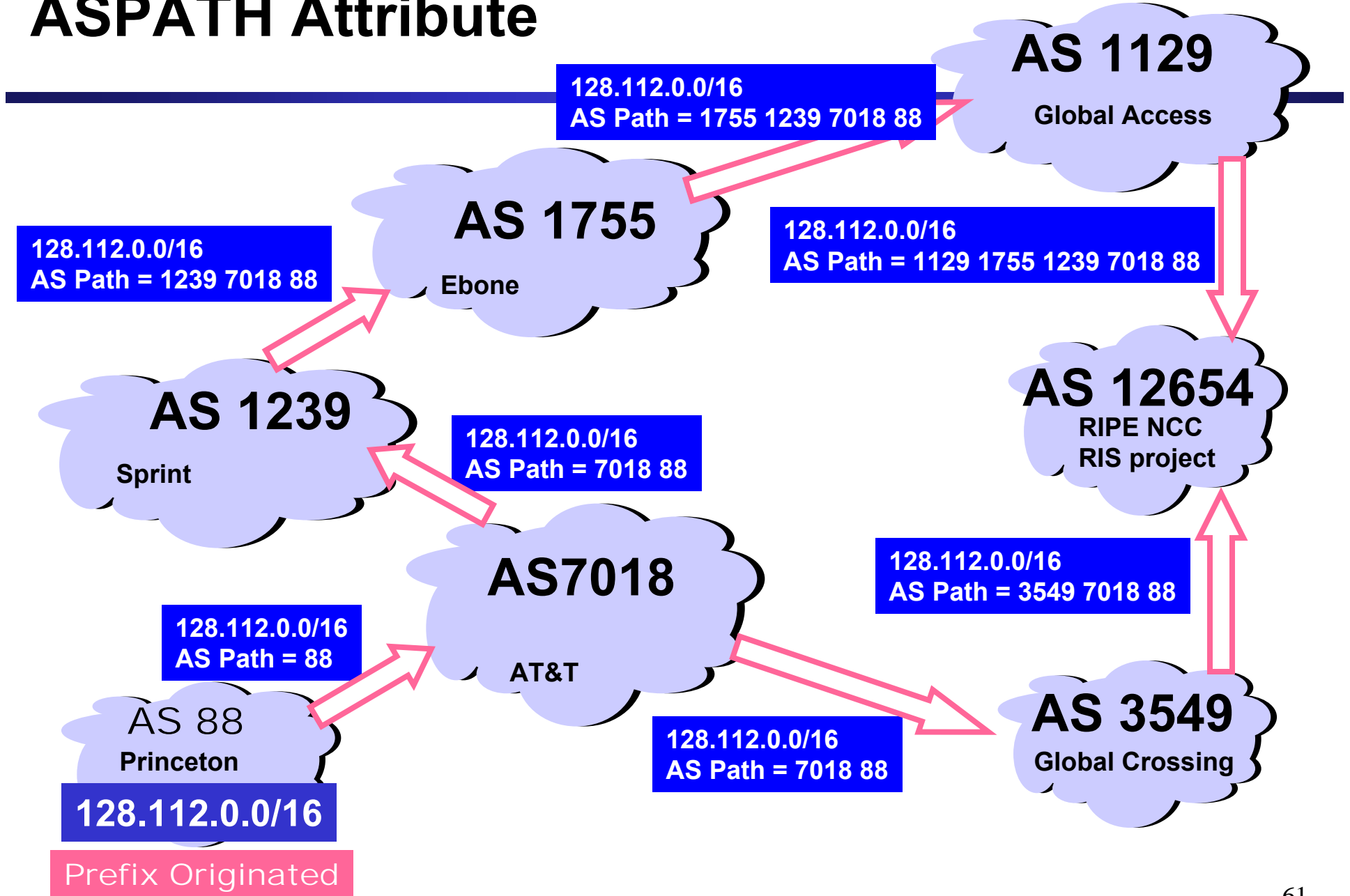
- A node learns multiple paths to destination
 - Stores all of the routes in a routing table
 - Applies policy to select a single active route
 - ... and may advertise the route to its neighbors
- Incremental updates
 - Announcement
 - Upon selecting a new active route, add node id to path
 - ... and (optionally) advertise to each neighbor
 - Withdrawal
 - If the active route is no longer available
 - ... send a withdrawal message to the neighbors

BGP Route

- Destination prefix (e.g., 128.112.0.0/16)
- Route attributes, including
 - AS path (e.g., “7018 88”)
 - Next-hop IP address (e.g., 12.127.0.121)

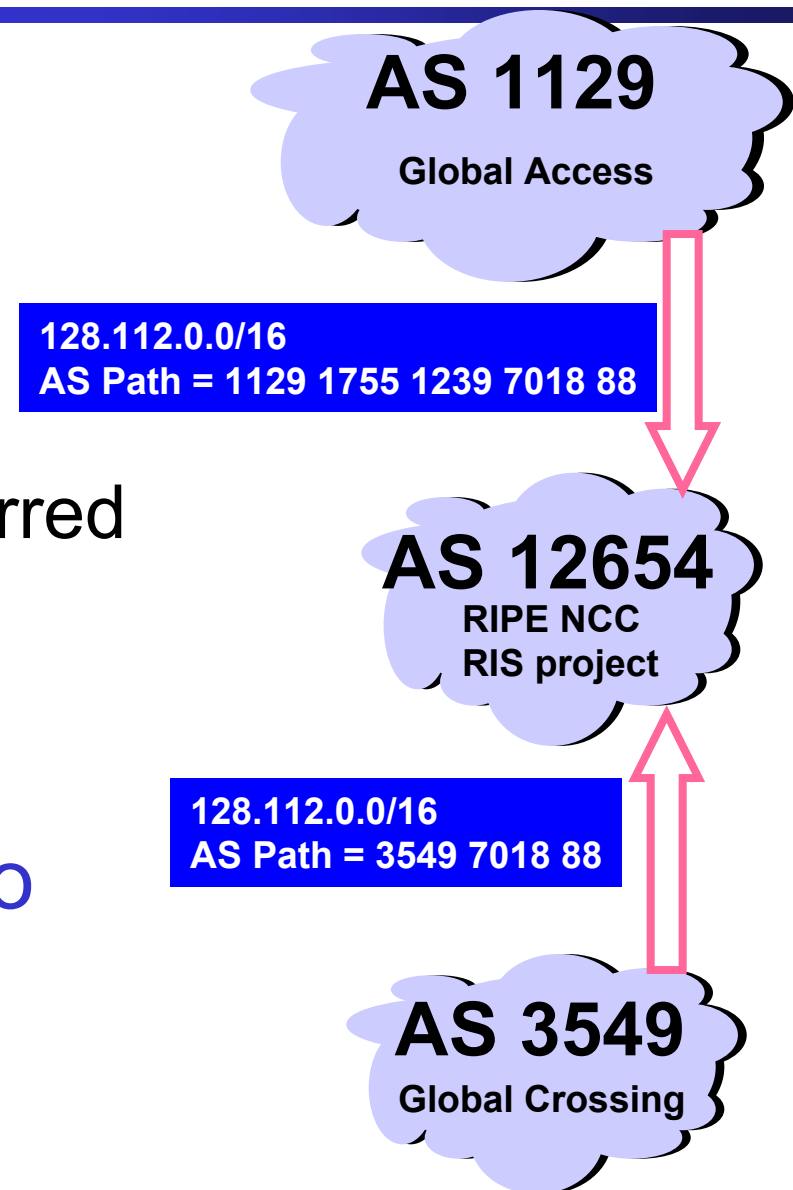


ASPATH Attribute



BGP Path Selection

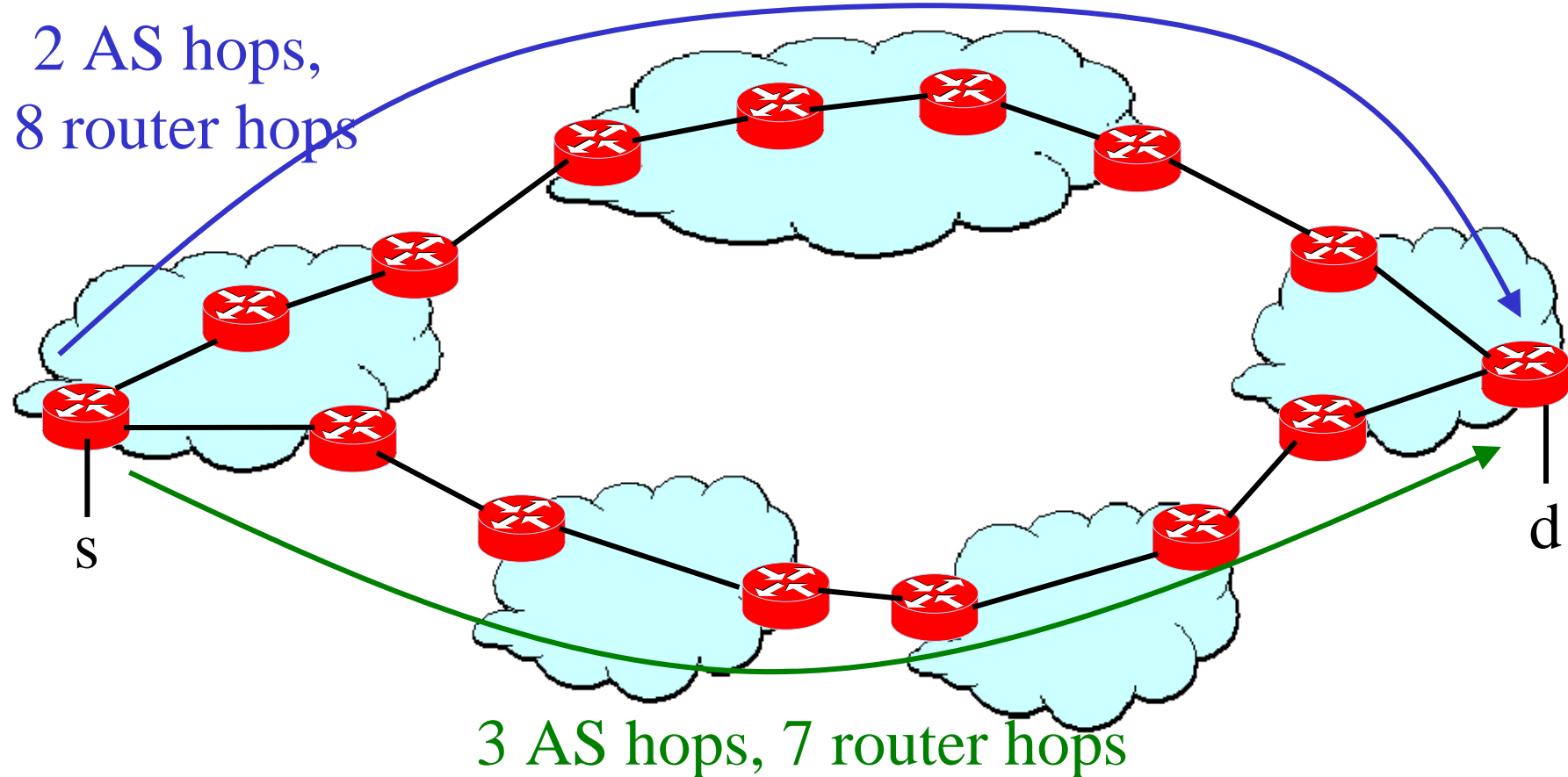
- Simplest case
 - Shortest AS path
 - Arbitrary tie break
- Example
 - Three-hop AS path preferred over a five-hop AS path
 - AS 12654 prefers path through Global Crossing
- But, BGP is not limited to shortest-path routing
 - Policy-based routing



Multiple Routers in an AS

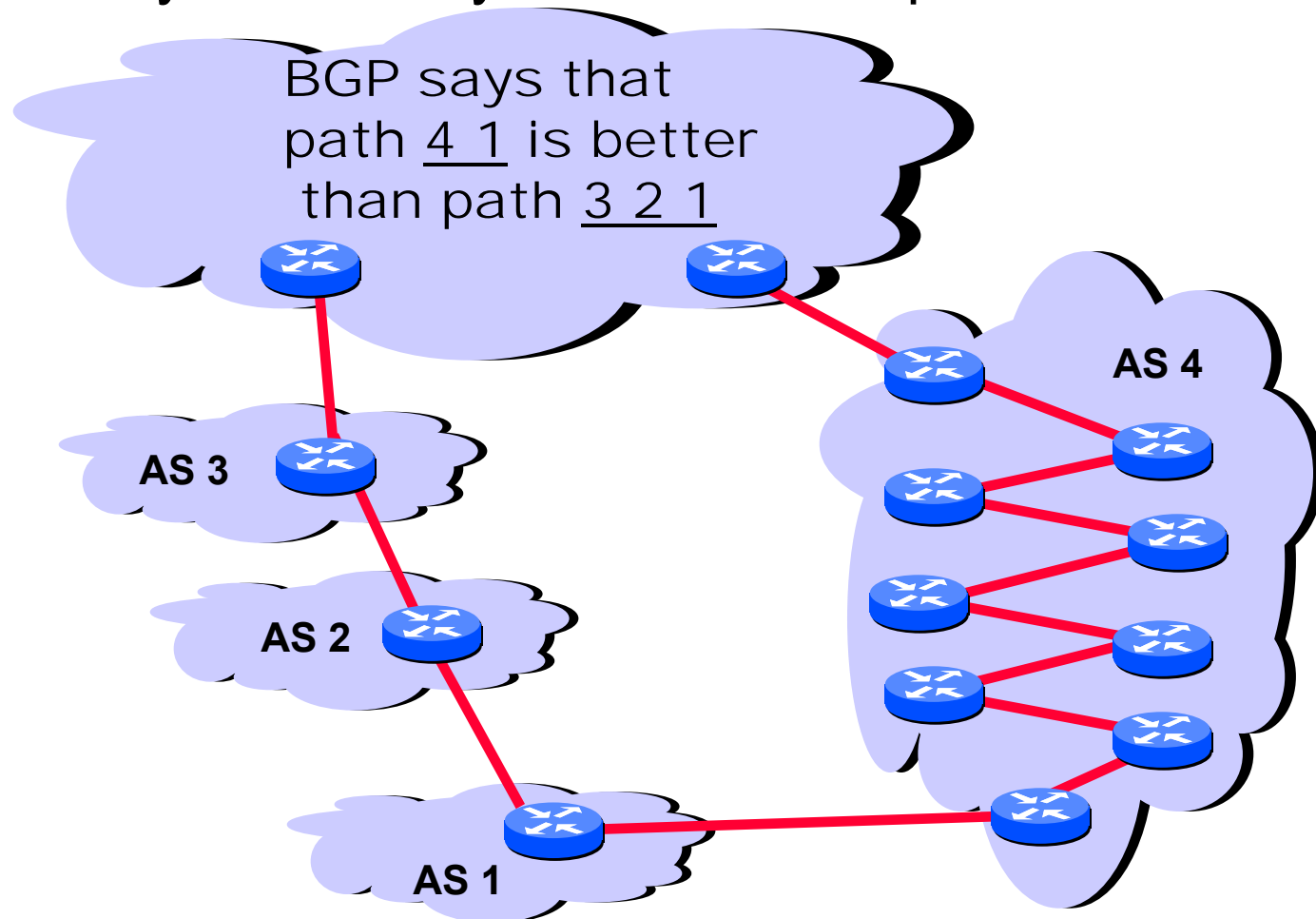
AS Path Length != Router Hops

- AS path may be longer than shortest AS path
- Router path may be longer than shortest path



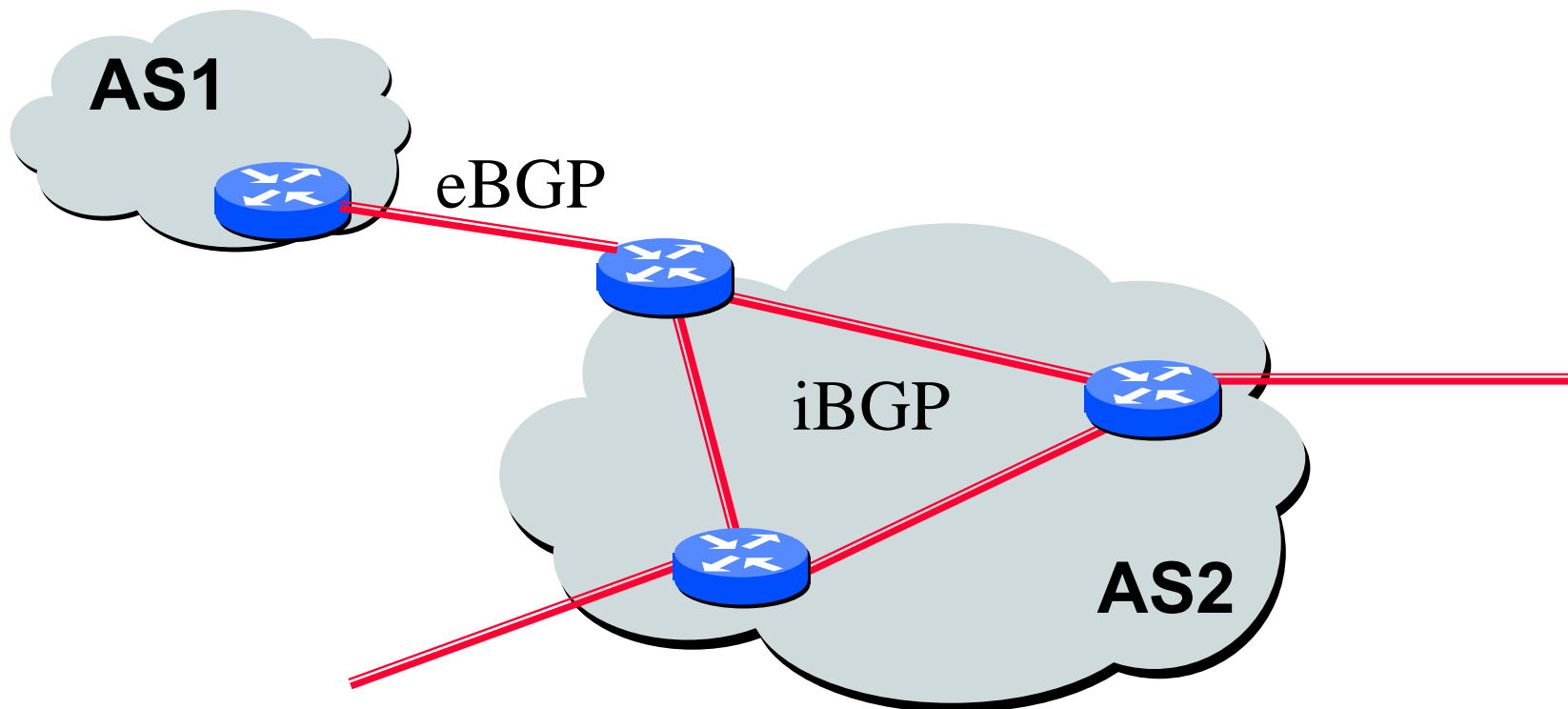
AS is Not a Single Node

- AS path length can be misleading
 - An AS may have many router-level hops



An AS is Not a Single Node

- Multiple routers in an AS
 - Need to distribute BGP information within the AS
 - Internal BGP (iBGP) sessions between routers



BGP Convergence

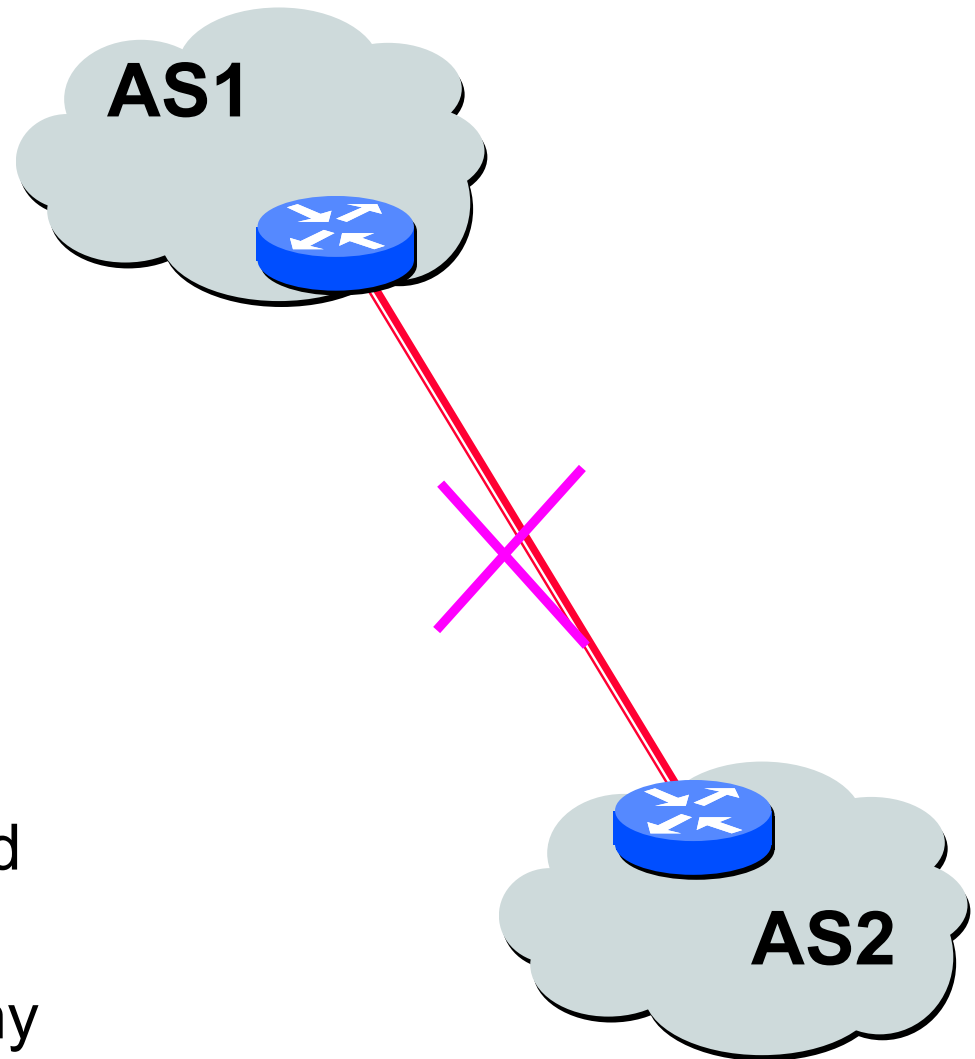
Slides by Rexford @ Princeton, slightly altered by M.D.

Causes of BGP Routing Changes

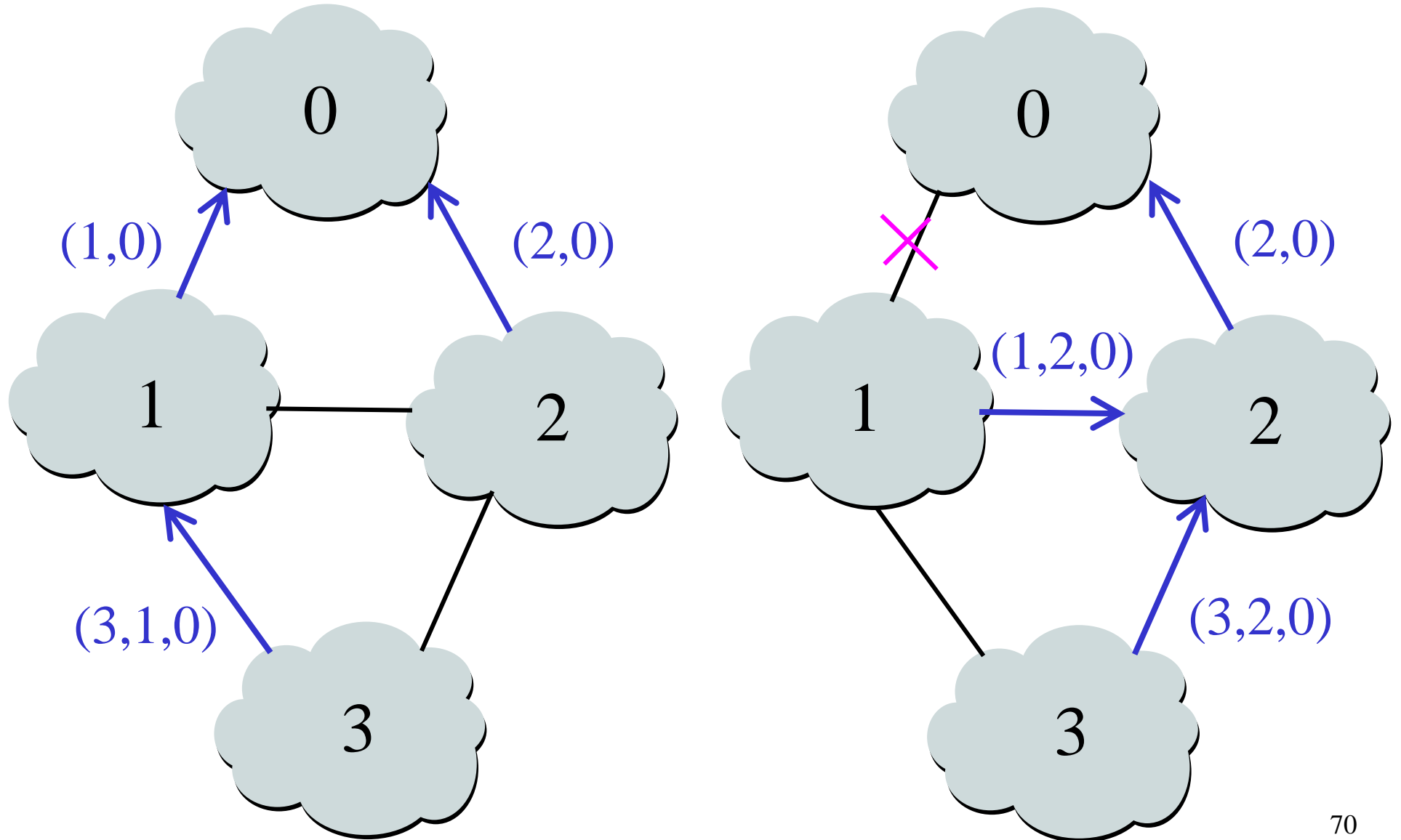
- **Topology changes**
 - Equipment going up or down
 - Deployment of new routers or sessions
- **BGP session failures**
 - Due to equipment failures, maintenance, etc.
 - Or, due to congestion on the physical path
- **Changes in routing policy**
 - Changes in preferences in the routes
 - Changes in whether the route is exported
- **Persistent protocol oscillation**
 - Conflicts between policies in different ASes

BGP Session Failure

- BGP runs over TCP
 - BGP only sends updates when changes occur
 - TCP doesn't detect lost connectivity on its own
- Detecting a failure
 - Keep-alive: 60 seconds
 - Hold timer: 180 seconds
- Reacting to a failure
 - Discard all routes learned from the neighbor
 - Send new updates for any routes that change

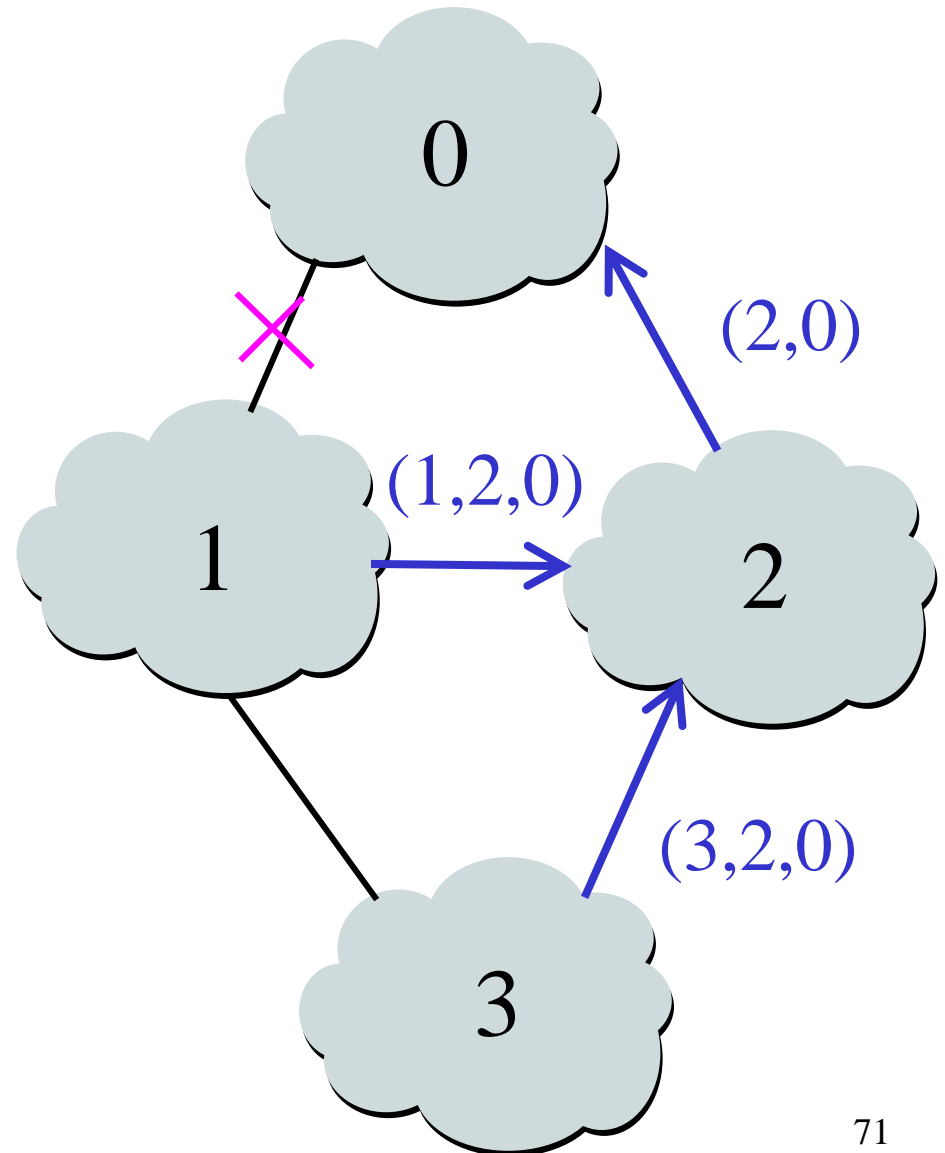


Routing Change: Before and After



Routing Change: Path Exploration

- AS 1
 - Delete the route (1,0)
 - Switch to next route (1,2,0)
 - Send route (1,2,0) to AS 3
- AS 3
 - Sees (1,2,0) replace (1,0)
 - Compares to route (2,0)
 - Switches to using AS 2



BGP Converges Slowly

- Path vector avoids count-to-infinity
 - But, ASes still must explore many alternate paths
 - ... to find the highest-ranked path that is still available
- Fortunately, in practice
 - Most popular destinations have very stable BGP routes
 - And most instability lies in a few unpopular destinations
- Still, lower BGP convergence delay is a goal
 - Can be tens of seconds to tens of minutes
 - High for important interactive applications
 - ... or even conventional application, like Web browsing

BGP Policies

Slides by Rexford @ Princeton, slightly altered by M.D.

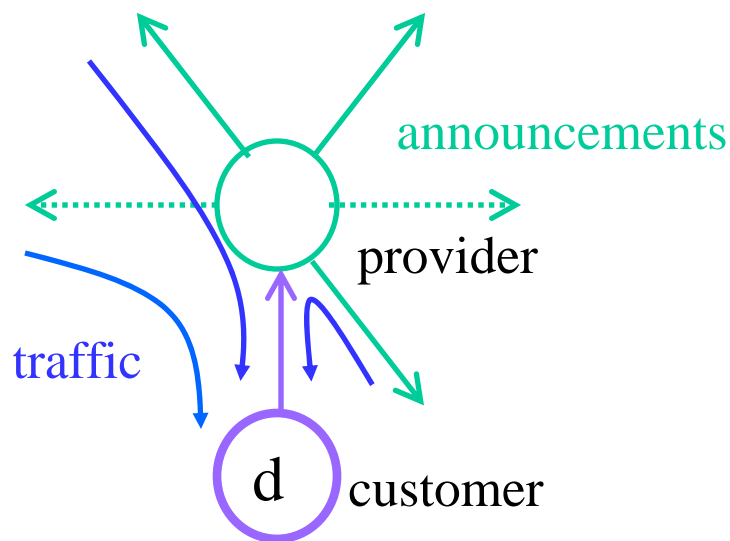
Business Relationships

- Neighboring ASes have business contracts
 - How much traffic to carry
 - Which destinations to reach
 - How much money to pay
- Common business relationships
 - Customer-provider
 - E.g., Princeton is a customer of USLEC
 - E.g., Villanova is a customer of Level3
 - Peer-peer
 - E.g., UUNET is a peer of Sprint
 - Visit <http://www.robtex.com/as/as10448.html> (graph)

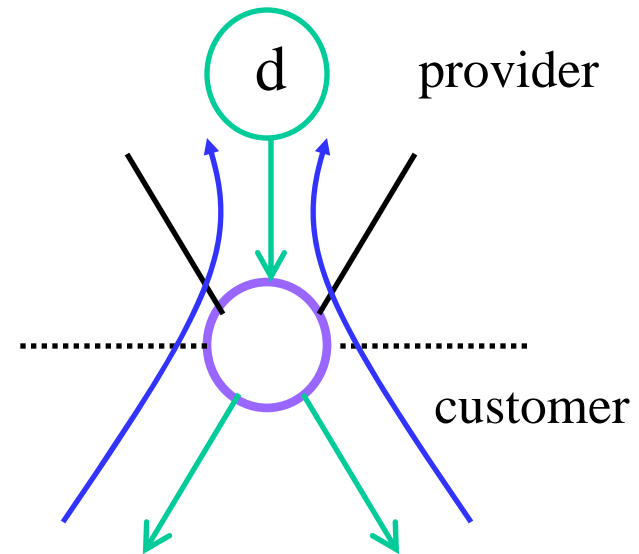
Customer-Provider Relationship

- Customer needs to be reachable from everyone
 - Provider tells all neighbors how to reach the customer
- Customer does not want to provide transit service
 - Customer does not let its providers route through it

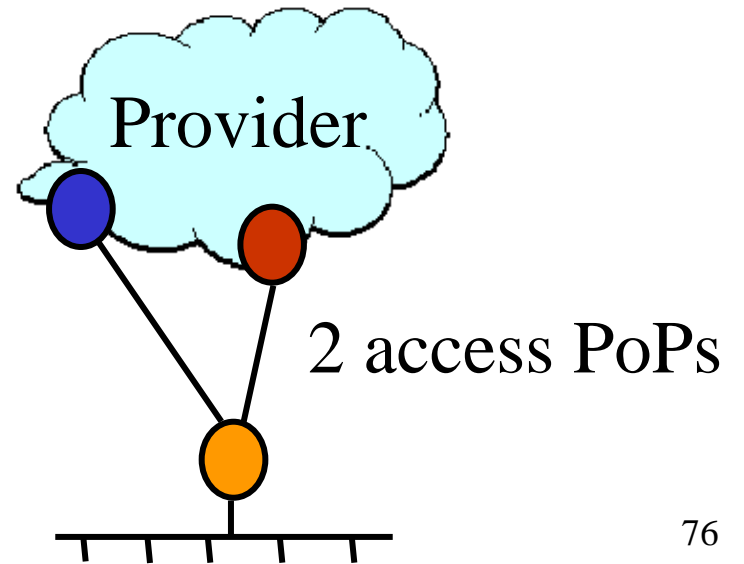
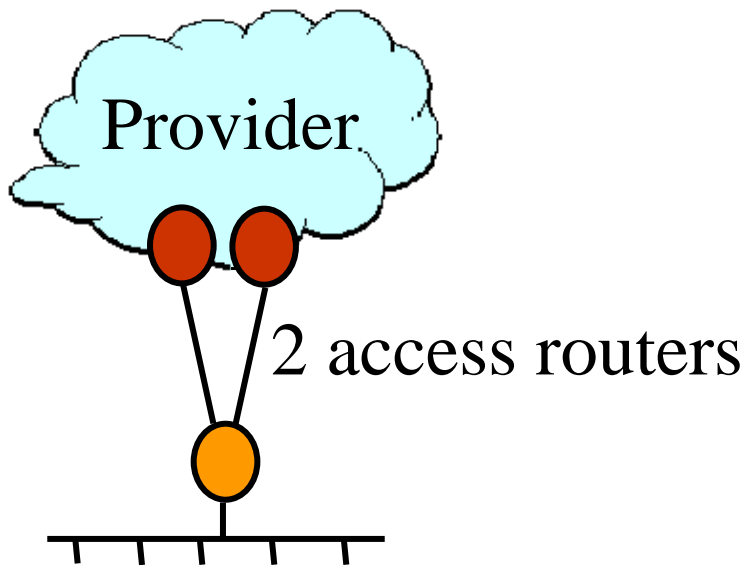
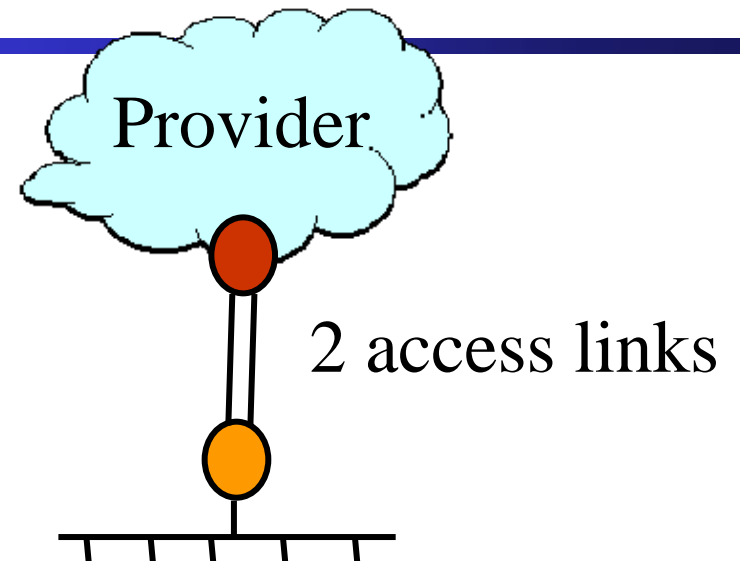
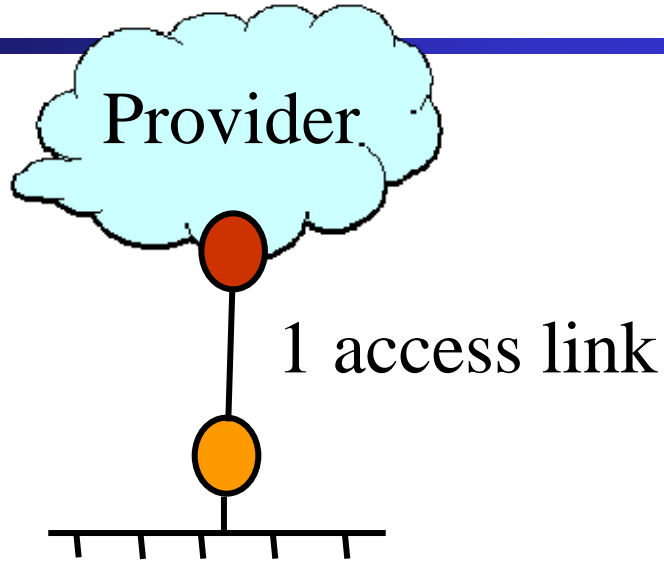
Traffic to the customer



Traffic from the customer

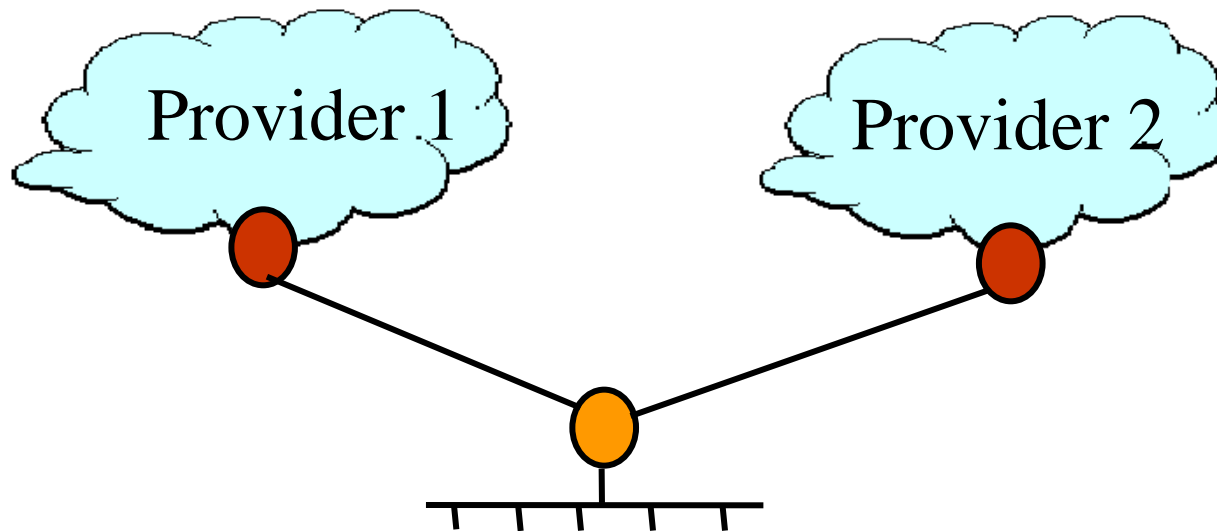


Customer Connecting to a Provider



Multi-Homing: Two or More Providers

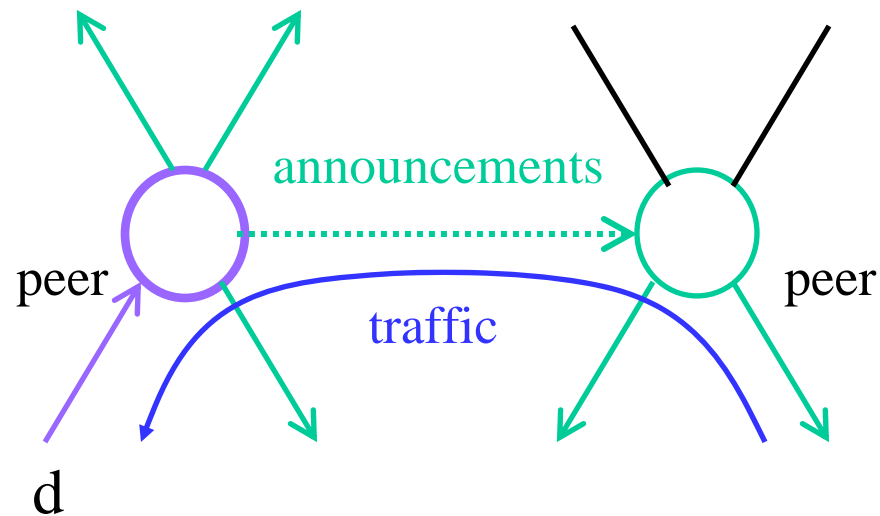
- Motivations for multi-homing
 - Extra reliability, survive single ISP failure
 - Financial leverage through competition
 - Better performance by selecting better path



Peer-Peer Relationship

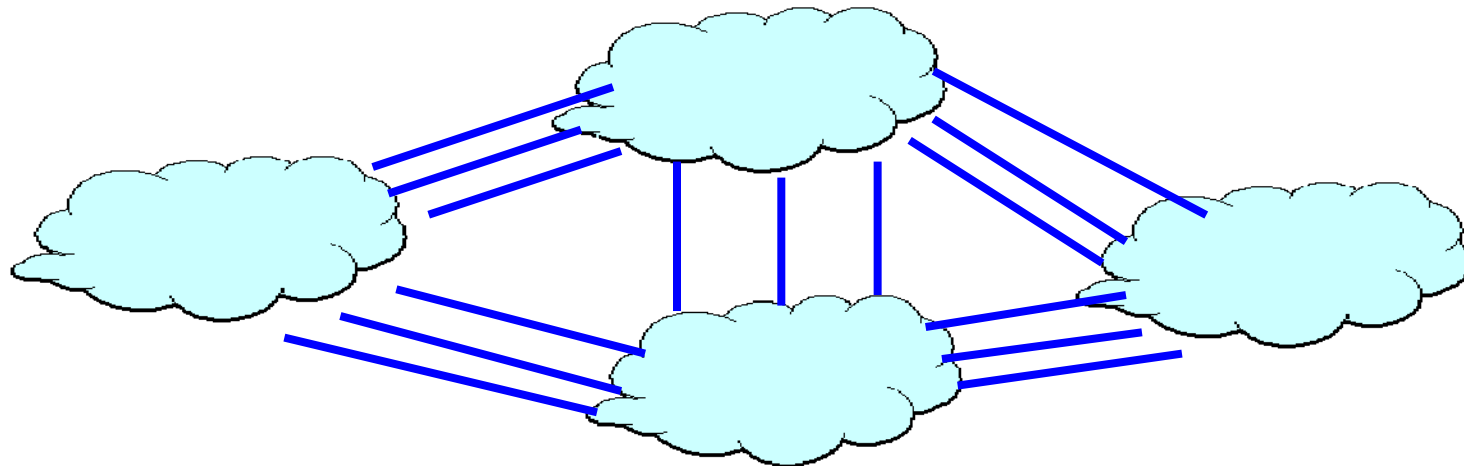
- Peers exchange traffic between customers
 - AS exports *only* customer routes to a peer
 - AS exports a peer's routes *only* to its customers
 - Often the relationship is settlement-free (i.e., no \$\$\$)

Traffic to/from the peer and its customers



AS Structure: Tier-1 Providers

- Tier-1 provider
 - Has no upstream provider of its own
 - Typically has a national or international backbone
- Top of the Internet hierarchy of ~10 ASes
 - AOL, AT&T, Global Crossing, Level3, UUNET, NTT, Qwest, SAVVIS (formerly Cable & Wireless), and Sprint
 - Full peer-peer connections between tier-1 providers



AS Structure: Other ASes

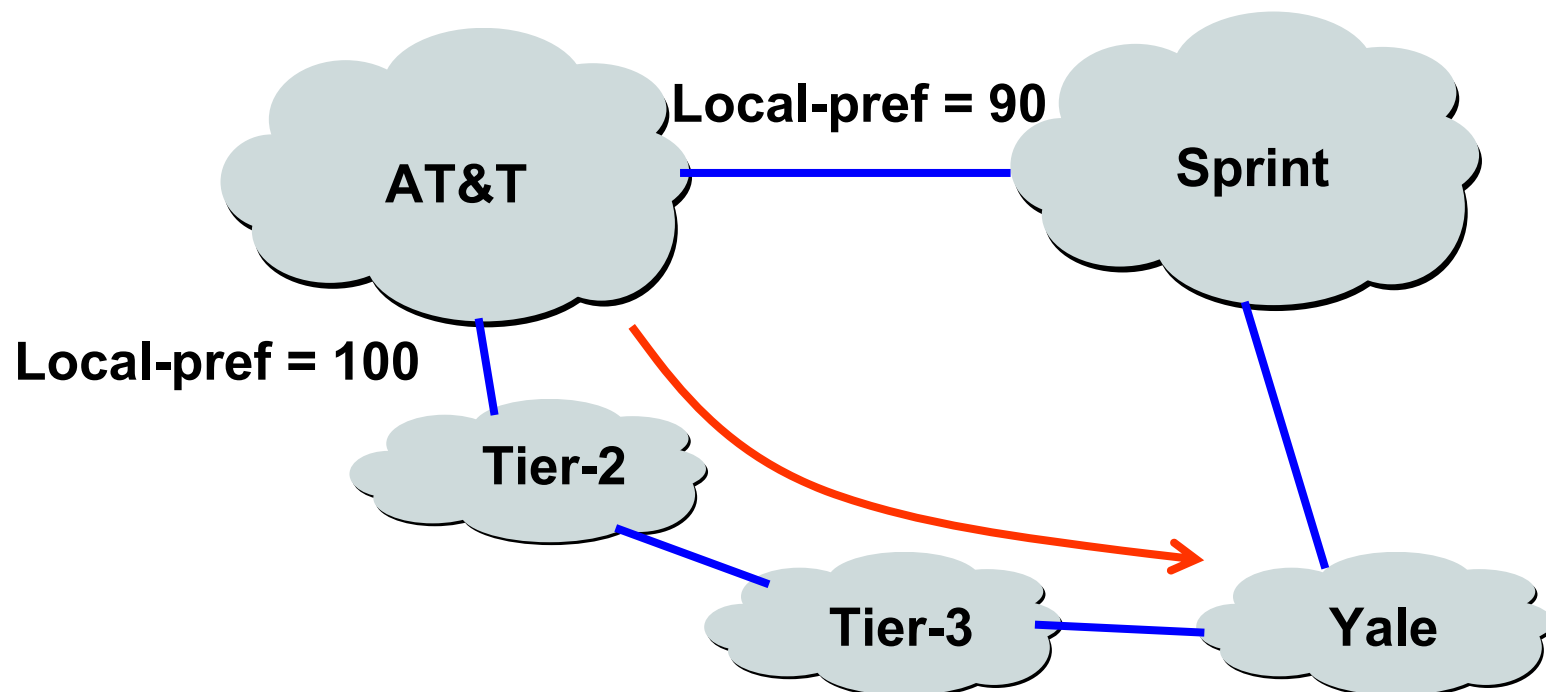
- Other providers
 - Provide transit service to downstream customers
 - ... but, need at least one provider of their own
 - Typically have national or regional scope
 - Includes several thousand ASes
- Stub ASes
 - Do not provide transit service to others
 - Connect to one or more upstream providers
 - Includes the vast majority (e.g., 85-90%) of the ASes

BGP Policy: Applying Policy to Routes

- **Import policy**
 - Filter unwanted routes from neighbor
 - E.g. prefix that your customer doesn't own
 - Manipulate attributes to influence path selection
 - E.g., assign local preference to favored routes
- **Export policy**
 - Filter routes you don't want to tell your neighbor
 - E.g., don't tell a peer a route learned from other peer
 - Manipulate attributes to control what they see
 - E.g., make a path look artificially longer than it is

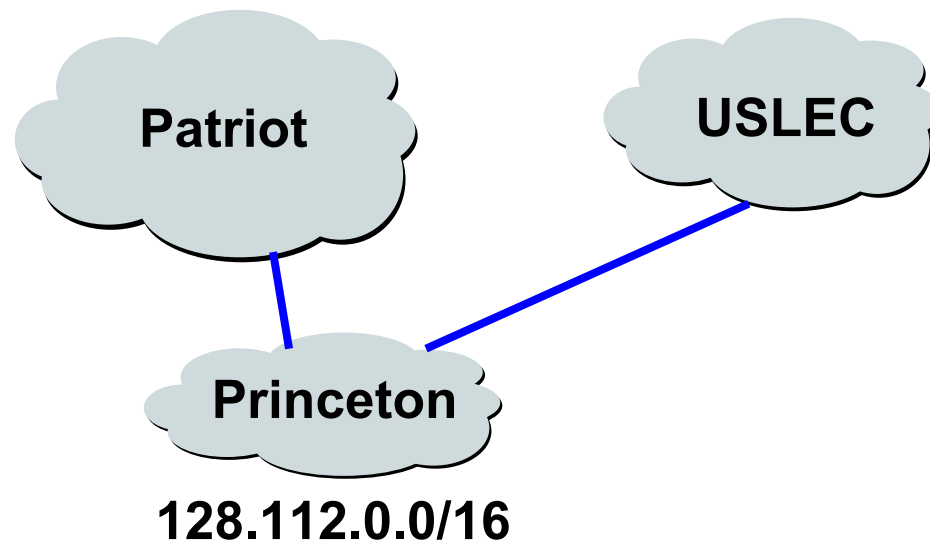
Import Policy: Local Preference

- Favor one path over another
 - Override the influence of AS path length
 - Apply local policies to prefer a path
- Example: prefer customer over peer



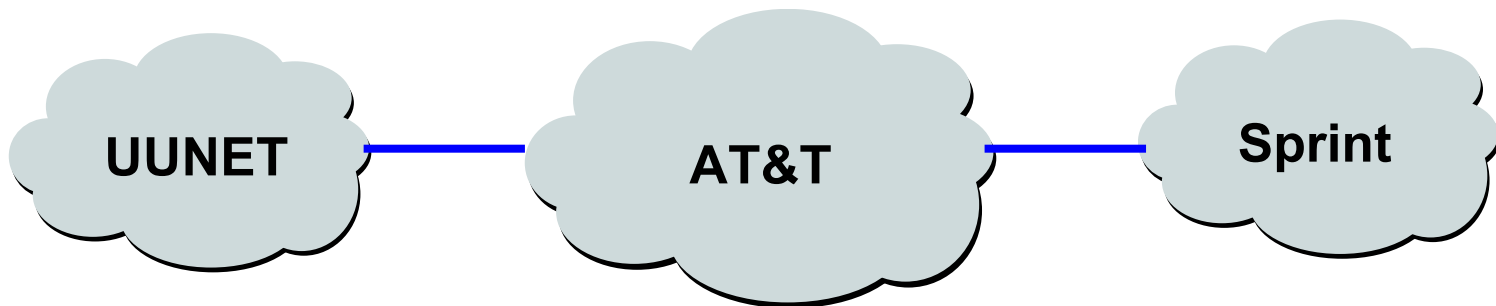
Import Policy: Filtering

- Discard some route announcements
 - Detect configuration mistakes and attacks
- Examples on session to a customer
 - Discard route if prefix not owned by the customer
 - Discard route that contains other large ISP in AS path



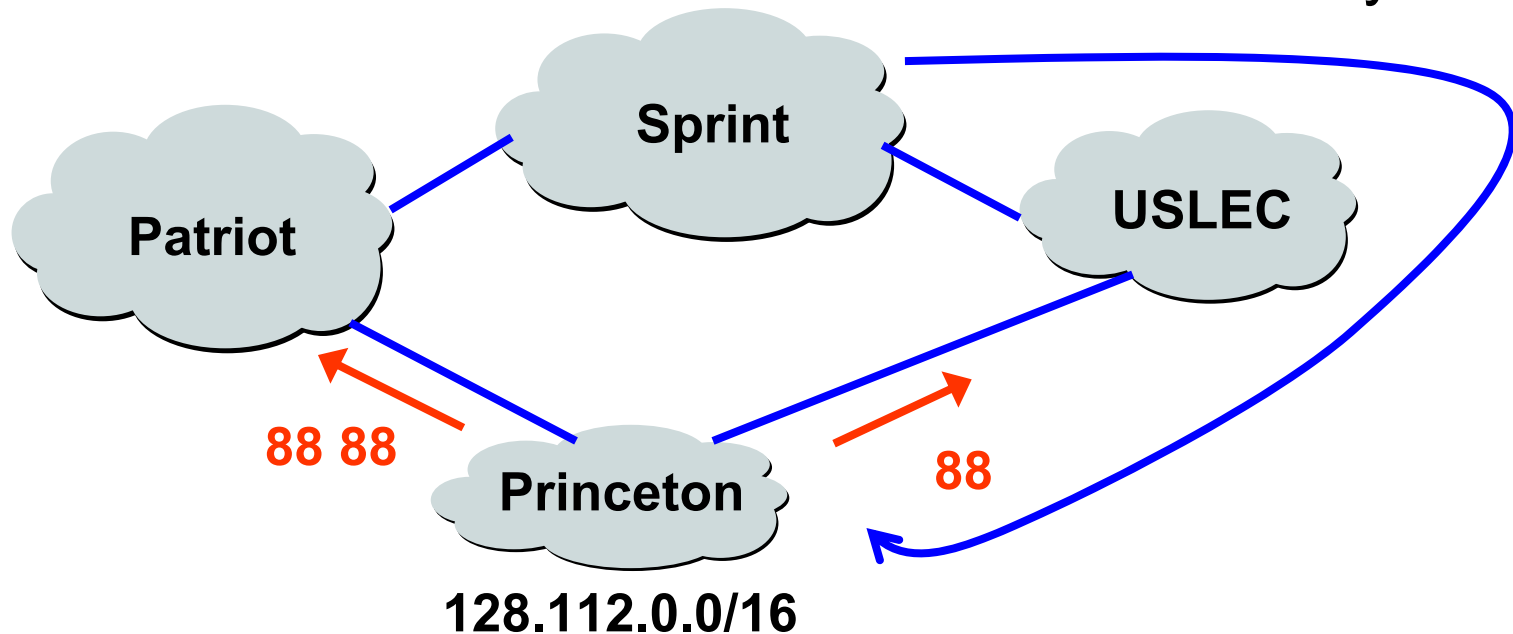
Export Policy: Filtering

- Discard some route announcements
 - Limit propagation of routing information
- Examples
 - Don't announce routes from one peer to another



Export Policy: Attribute Manipulation

- Modify attributes of the active route
 - To influence the way other ASes behave
- Example: AS prepending
 - Artificially inflate the AS path length seen by others
 - To convince some ASes to send traffic another way



BGP Security

Slides by Rexford @ Princeton, slightly altered by M.D.

The February 24 YouTube Outage

- YouTube (AS 36561)
 - Address block 208.65.152.0/22
- Pakistan Telecom (AS 17557)
 - Receives government order to block access to YouTube
 - Starts announcing 208.65.153.0/24 to PCCW (AS 3491), one of its upstream providers
 - All packets directed to YouTube get dropped on the floor
- Mistakes were made
 - AS 17557: announcing to everyone, not just customers
 - AS 3491: not filtering routes announced by AS 17557
- Lasted 100 minutes for some, 2 hours for others

Timeline (UTC Time)

- 18:47:45
 - First evidence of hijacked /24 route propagating in Asia
- 18:48:00
 - Several big trans-Pacific providers carrying the route
- 18:49:30
 - Bogus route fully propagated
- 20:07:25
 - YouTube starts advertising the /24 to attract traffic back
- 20:08:30
 - Many (but not all) providers are using the valid route

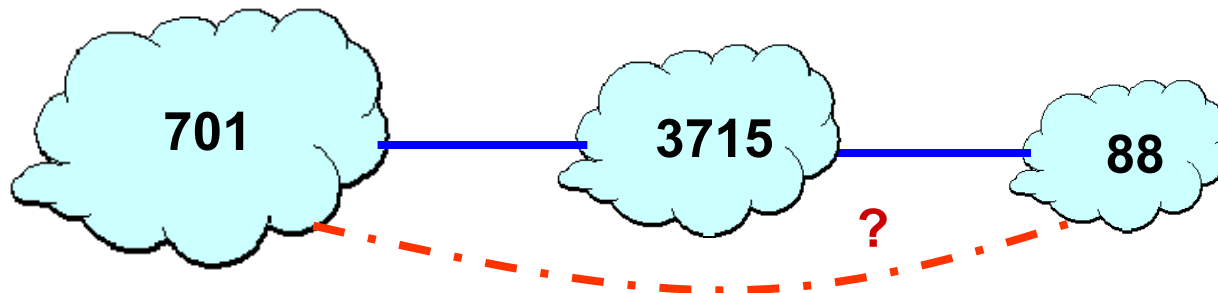
Timeline (UTC Time)

- 20:18:43
 - YouTube starts announcing two more-specific /25 routes
- 20:19:37
 - Some more providers start using the /25 routes
- 20:50:59
 - AS 17557 starts prepending (“3491 17557 17557”)
- 20:59:39
 - AS 3491 disconnects AS 17557
- 21:00:00
 - All is well, videos of cats flushing toilets are available

BGP AS Path

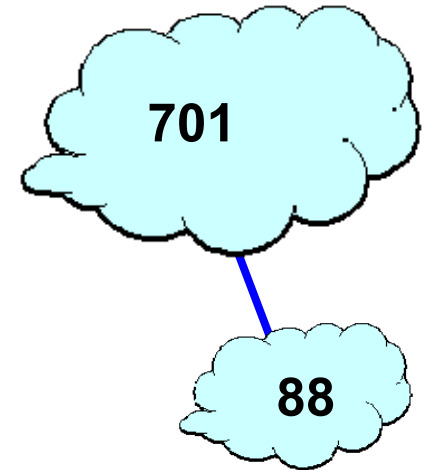
Bogus AS Paths

- Remove ASes from the AS path
 - E.g., turn “701 3715 88” into “701 88”
- Motivations
 - Make the AS path look shorter than it is
 - Attract sources that normally try to avoid AS 3715
 - Help AS 88 look like it is closer to the Internet’s core
- Who can tell that this AS path is a lie?
 - Maybe AS 88 **does** connect to AS 701 directly



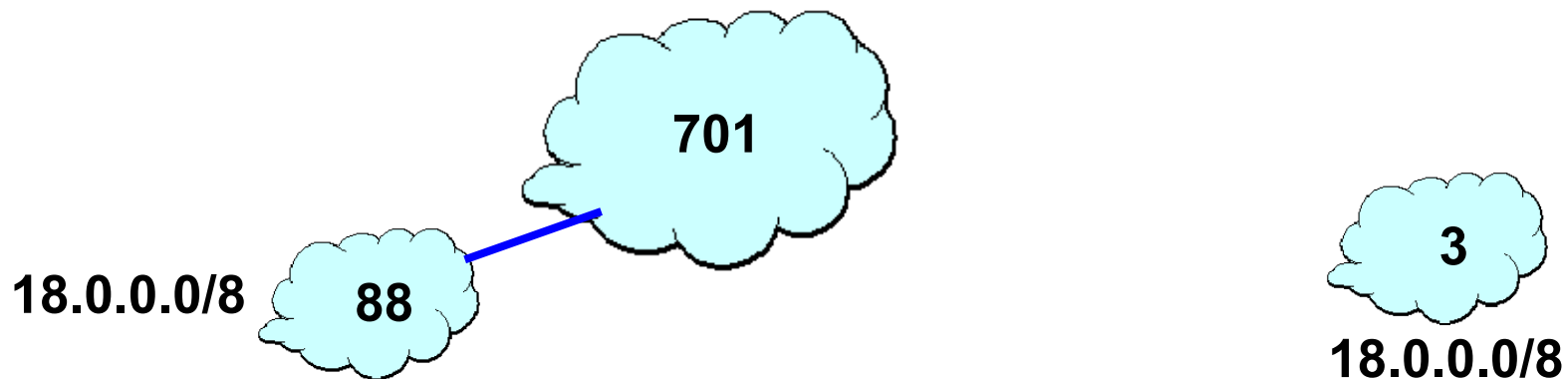
Bogus AS Paths

- Add ASes to the path
 - E.g., turn “701 88” into “701 3715 88”
- Motivations
 - Trigger loop detection in AS 3715
 - Denial-of-service attack on AS 3715
 - Or, blocking unwanted traffic coming from AS 3715!
 - Make your AS look like it has richer connectivity
- Who can tell the AS path is a lie?
 - AS 3715 could, if it could see the route
 - AS 88 could, but would it really care as long as it received data traffic meant for it?



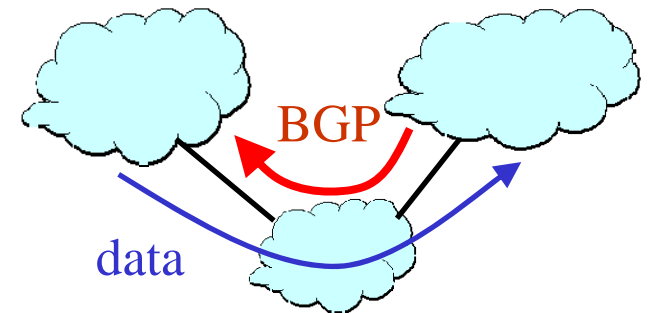
Bogus AS Paths

- Adds AS hop(s) at the end of the path
 - E.g., turns “701 88” into “701 88 3”
- Motivations
 - Evade detection for a bogus route
 - E.g., by adding the legitimate AS to the end
- Hard to tell that the AS path is bogus...
 - Even if other ASes filter based on prefix ownership



Invalid Paths

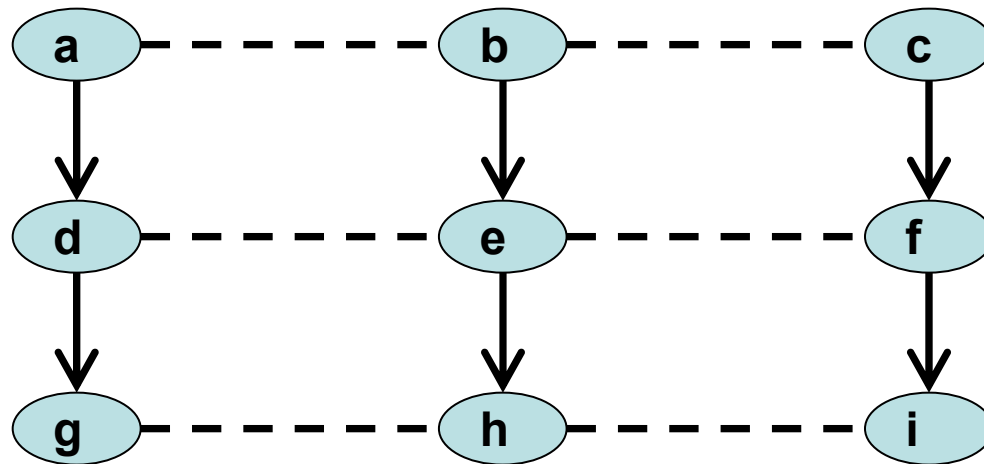
- AS exports a route it shouldn't
 - AS path is a valid sequence, but violated policy
- Example: customer misconfiguration
 - Exports routes from one provider to another
- ... interacts with provider policy
 - Provider prefers customer routes
 - ... so picks these as the best route
- ... leading the dire consequences
 - Directing all Internet traffic through customer
- Main defense
 - Filtering routes based on prefixes and AS path



BGP Security Today

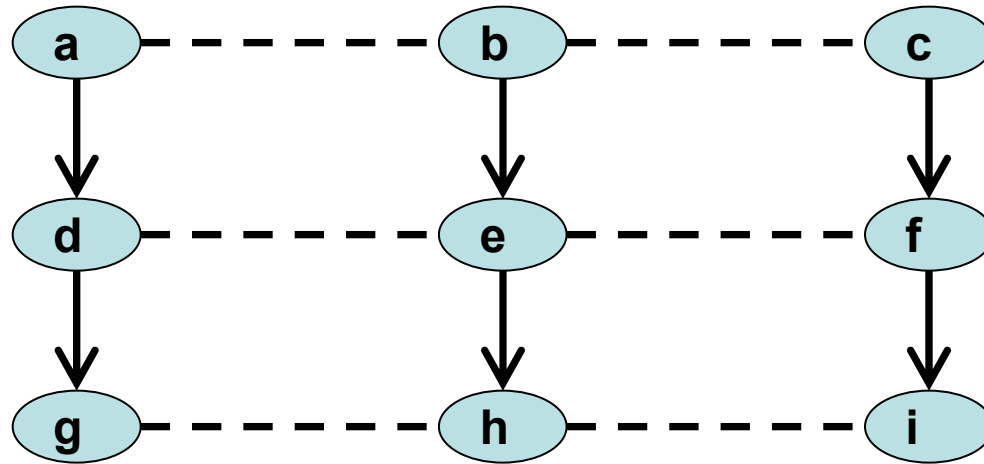
- Applying best common practices (BCPs)
 - Securing the session (authentication, encryption)
 - Filtering routes by prefix and AS path
 - Packet filters to block unexpected control traffic
- This is not good enough
 - Depends on vigilant application of BCPs
 - ... and not making configuration mistakes!
 - Doesn't address fundamental problems
 - Can't tell who owns the IP address block
 - Can't tell if the AS path is bogus or invalid

BGP Exercise



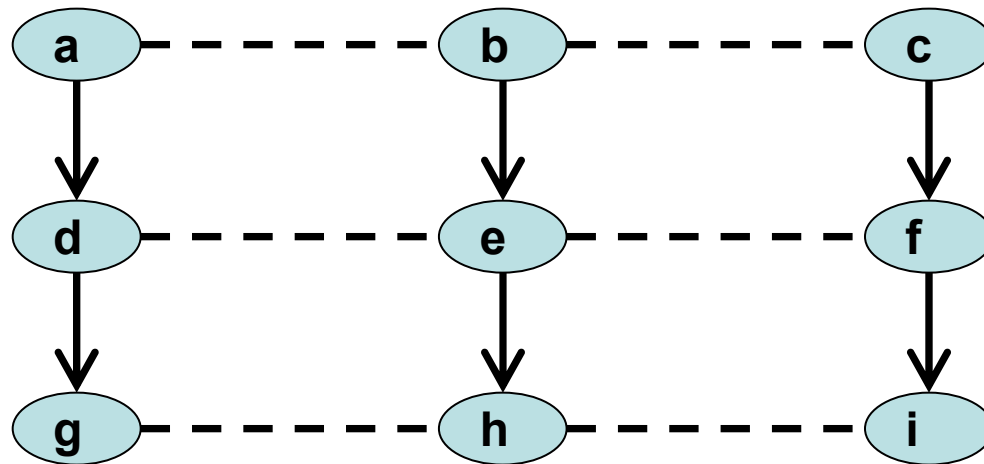
- This exercise explores the role of routing policy on BGP routing. In the picture above, the nodes are Autonomous Systems (ASes). Dotted lines correspond to peer-peer relationships, and solid arrows correspond to provider-customer relationships, with the arrow pointing from provider to customer. Every node prefers customer-learned routes over routes through peer ASes, and peer-learned routes over routes through provider ASes. Among routes through the same kind of neighbor, each AS prefers shorter AS paths over longer ones. An AS does not export routes learned from one peer or provider to other peers or providers.

Q



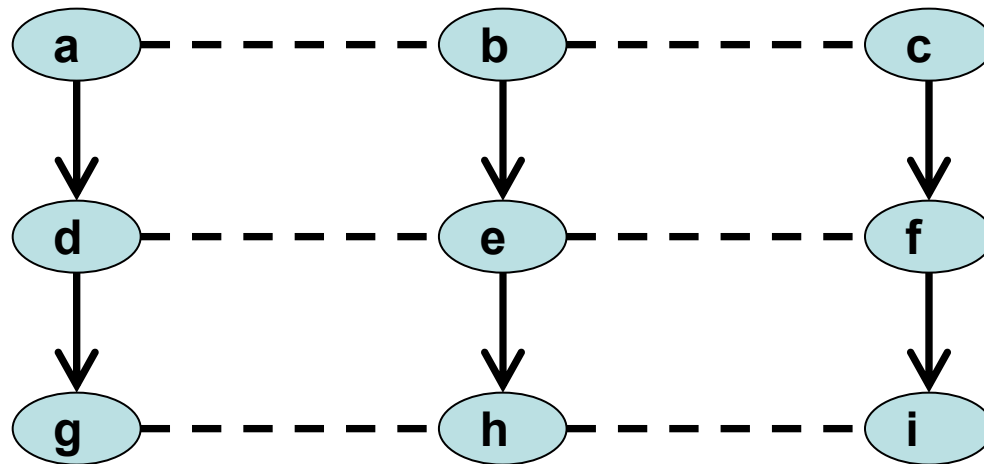
- Draw all of the paths to destination AS d. That is, draw the “sink tree” leading to d. Circle any ASes that do not have any route to d.

Q & A



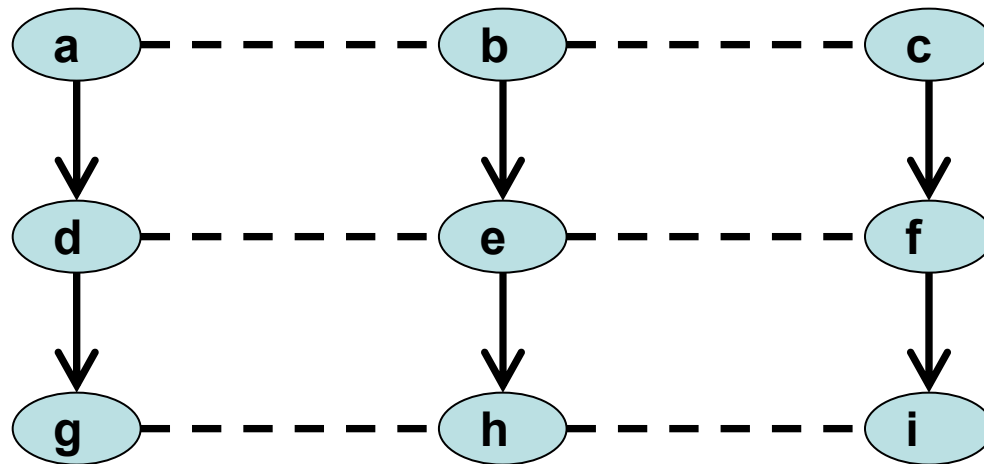
- Draw all of the paths to destination AS d. That is, draw the “sink tree” leading to d. Circle any ASes that do not have any route to d.
- *The paths are b-a-d, g-d, and h-e-d. Nodes c, f, and i cannot reach d.*

Q



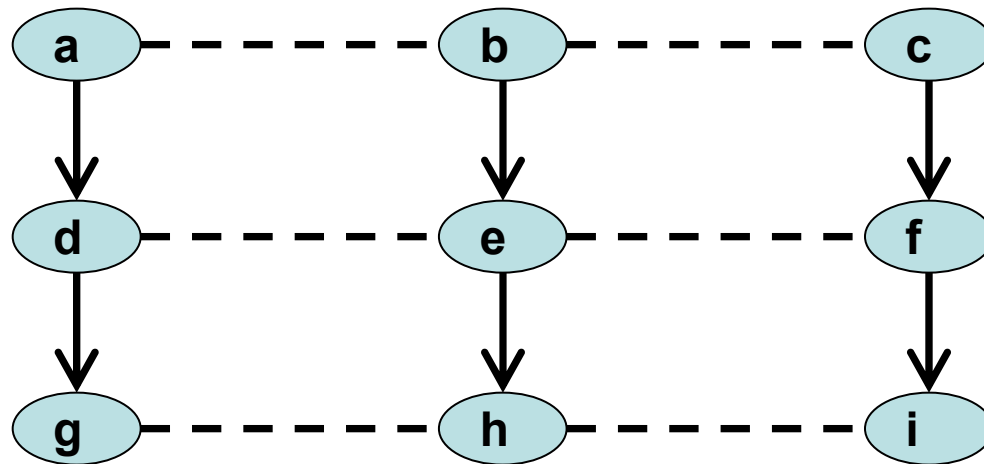
- What path does e take to reach i? Suppose f decides to terminate the peering relationship with AS e. Can AS e still reach destination i? If so, what is the new path?

Q&A



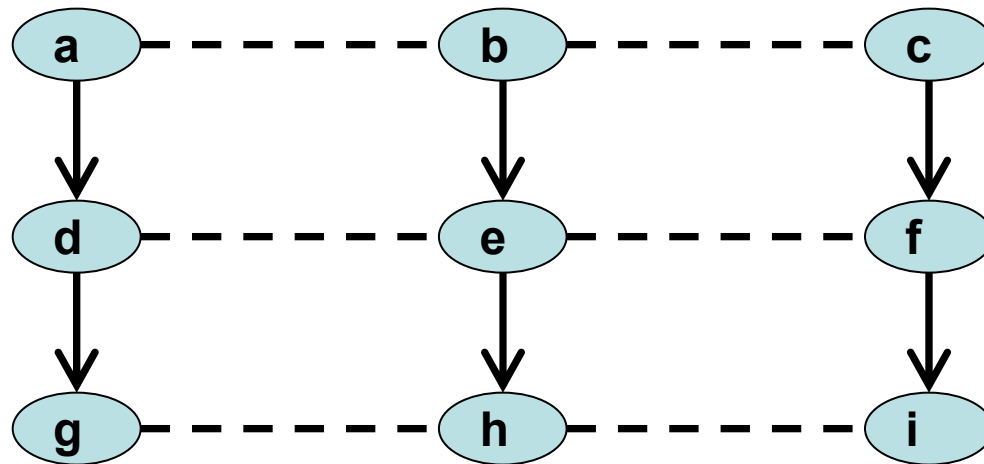
- What path does e take to reach i? Suppose f decides to terminate the peering relationship with AS e. Can AS e still reach destination i? If so, what is the new path?
- *Node e takes the path e-f-i. With the e-f link removed, the path is e-b-c-f-i. The path e-h-i is not used because e's customer h would not advertise a peer-learned route to its provider.*

Q



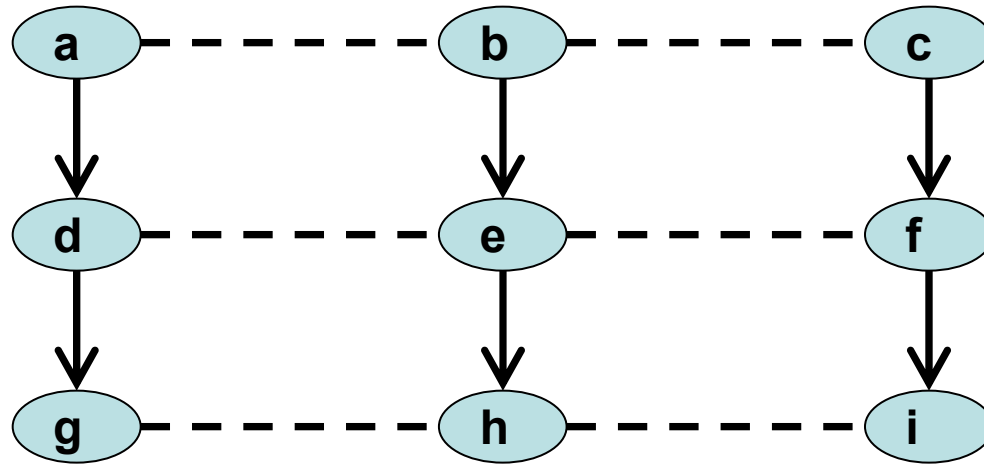
- Suppose ASes a and b provide a “dump” of all BGP routes they learn for every destination to a public measurement repository. Suppose a researcher extracts all links from all AS paths in those “dumps” and uses them to construct a view of the AS-level topology of the network. Draw a picture of the resulting inferred topology.

Q&A



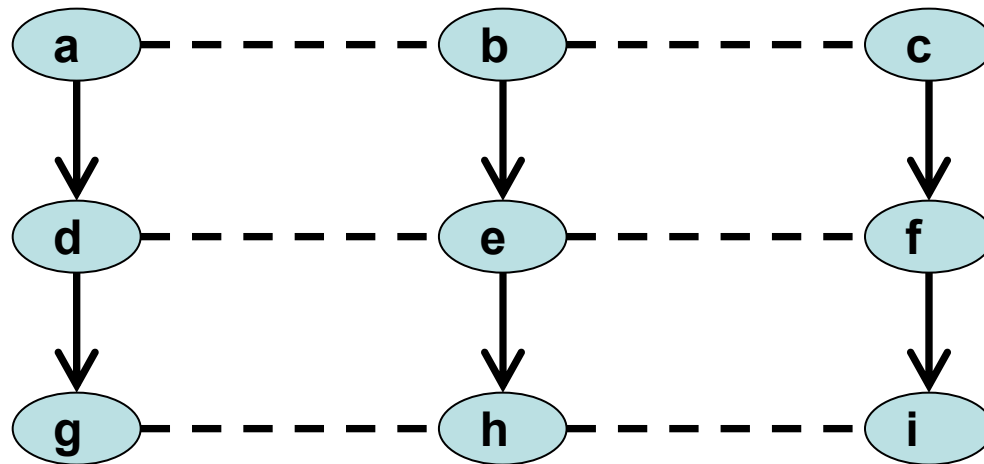
- The graph is missing the peering edges d-e, e-f, g-h, and h-i. Node a learns only customer-learned routes from its customer d and peer b, allowing it to learn a-d, a-d-g, a-b, a-b-e, and a-b-e-h. Node b learns only customer learned routes from its two peers a and c, and its customer e, allowing it to learn the routes that a learned, plus b-e and b-e-h from e, as well as b-c, b-c-f, and b-c-f-i from c.*

Q



- *What is the minimum set of ASes that must provide “dumps” of every AS path they learn for every edge in the graph to be visible in at least one dump? List the ASes.*

Q&A



- *Two nodes: b and h (or e and h). Node h has paths h-g (to g), h-i (to i), h-e (to e). It learns the paths h-e-d, h-e-f, h-e-b, h-e-b-a, and h-e-b-c from e. However, since e is using the path e-d to reach d and the path e-f to reach f, e does not announce the paths e-b-a-d or e-b-c-f, even though e does learn those paths. As such, the dump for node h reveals all edges except a-d and c-f, which are in a routing-table dump at either e or b.*

Conclusions

- **Routing is a distributed algorithm**
 - React to changes in the topology
 - Compute the paths through the network
- **Shortest-path link state routing**
 - Flood link weights throughout the network
 - Compute shortest paths as a sum of link weights
 - Forward packets on next hop in the shortest path
- **Convergence process**
 - Changing from one topology to another
 - Transient periods of inconsistency across routers

Conclusions

- **Distance-vector routing**
 - Compute path costs based on neighbors' path costs
 - Bellman-Ford algorithm & Routing Information Protocol
- **Path-vector routing**
 - Faster convergence than distance-vector protocols
 - While hiding information and enabling flexible policy
- **Interdomain routing**
 - Autonomous Systems (ASes)
 - Policy-based path-vector routing