Optimization and Control of Networks

Network Routing

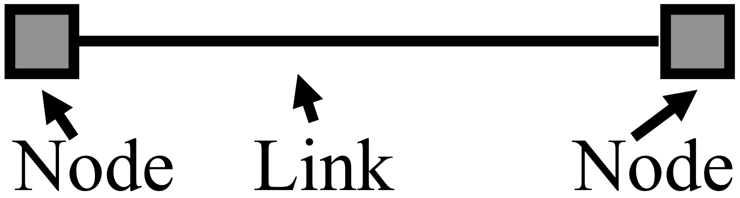
Lijun Chen

02/16/2016

Agenda

- Review on network routing
  - Internetworking
  - Intradomain routing
  - Interdomain routing

Simple Network: Nodes and a Link

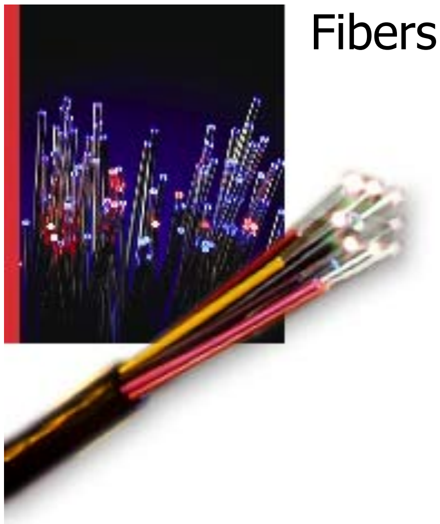


- **Node**: computer
  - End host: general-purpose computer, cell phone, PDA
  - Network node: switch or router
- **Link**: physical medium connecting nodes
  - Twisted pair: the wire that connects to telephones
  - Coaxial cable: the wire that connects to TV sets
  - Optical fiber: high-bandwidth long-distance links
  - Space: propagation of radio waves, microwaves, …

Network Components

Links

Fibers



Coaxial Cable

Interfaces

Ethernet card

Wireless card

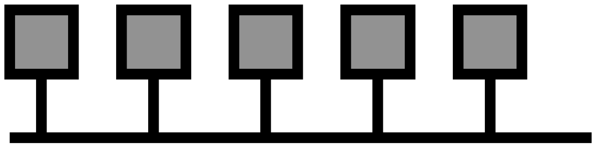
Switches/routers

Large router

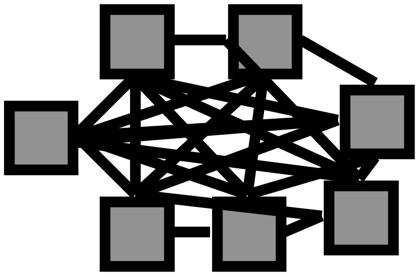
Telephone switch

Connecting More Than Two Hosts

- Multi-access link: Ethernet, wireless
  - Single physical link, shared by multiple nodes
  - Limitations on distance and number of nodes
- Point-to-point links: fiber-optic cable
  - Only two nodes (separate link per pair of nodes)
  - Limitations on the number of adapters per node

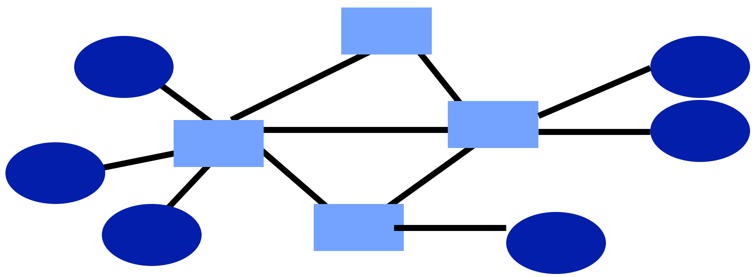


multi-access link



point-to-point links

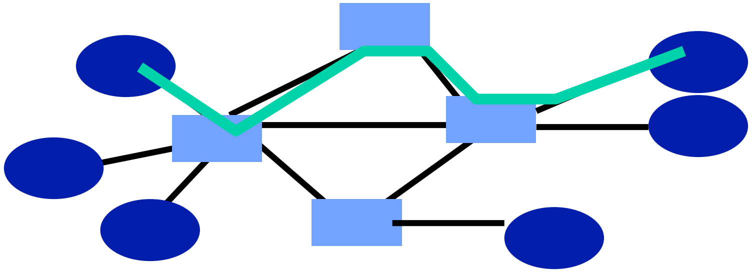
Beyond Directly-Connected Networks



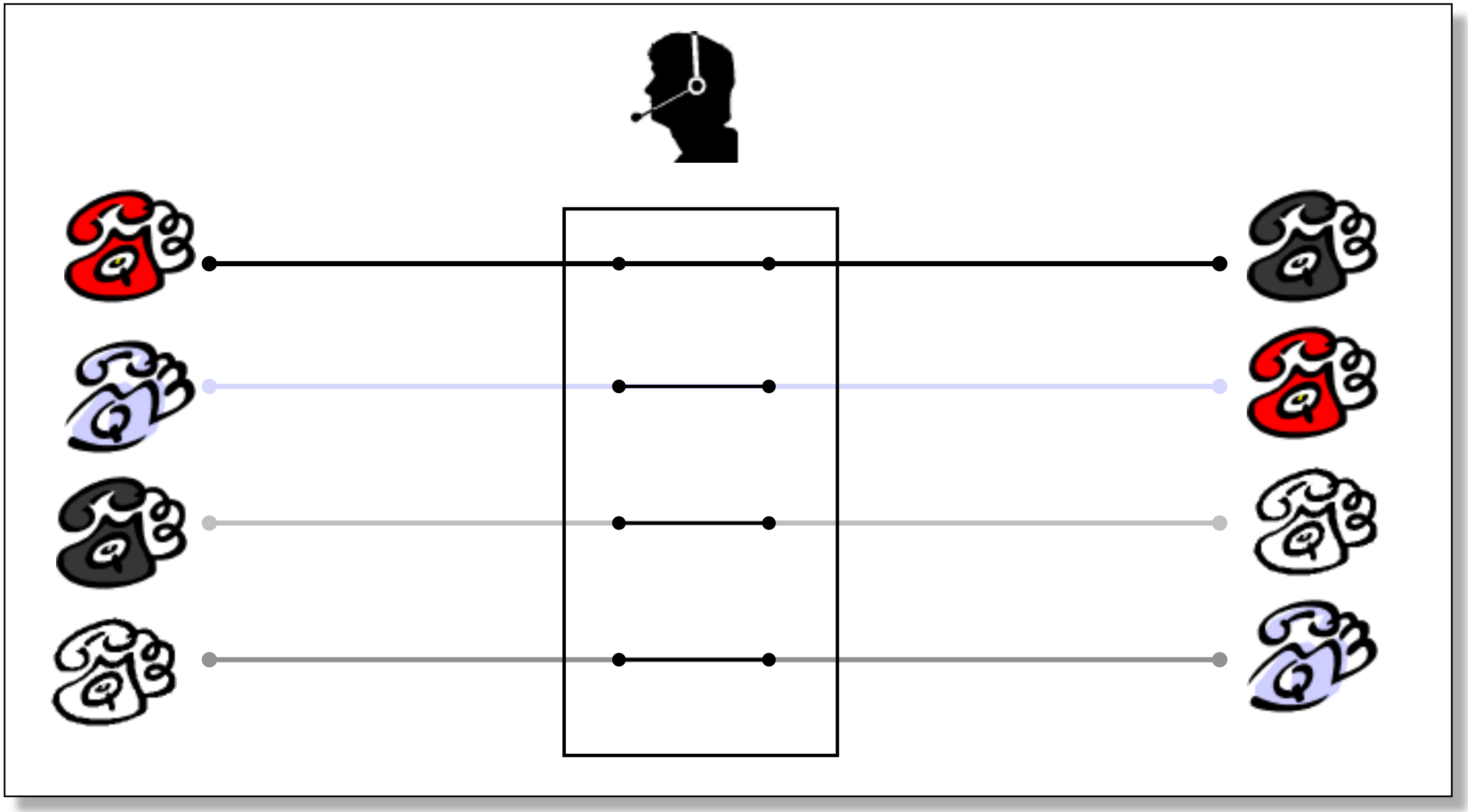
- Switched network
  - End hosts at the edge
  - Network nodes that switch traffic
  - Links between the nodes
- Multiplexing
  - Many end hosts communicate over the network
  - Traffic shares access to the same links

Circuit Switching (e.g., Phone Network)

- Source establishes connection to destination
  - Nodes along the path store connection info
  - Nodes may reserve resources for the connection
- Source sends data over the connection
  - No destination address, since nodes know path
- Source tears down connection when done

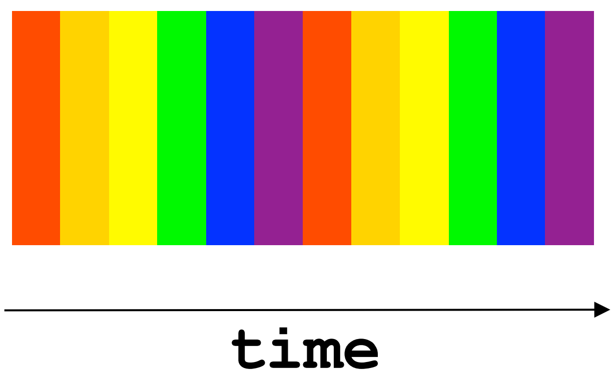


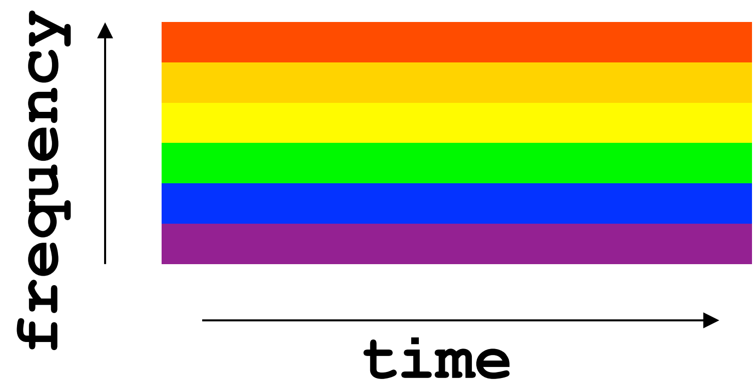
Circuit Switching With Human Operator



Circuit Switching: Multiplexing a Link

- Time-division
  - Each circuit allocated certain time slots
- Frequency-division
  - Each circuit allocated certain frequencies





Advantages of Circuit Switching

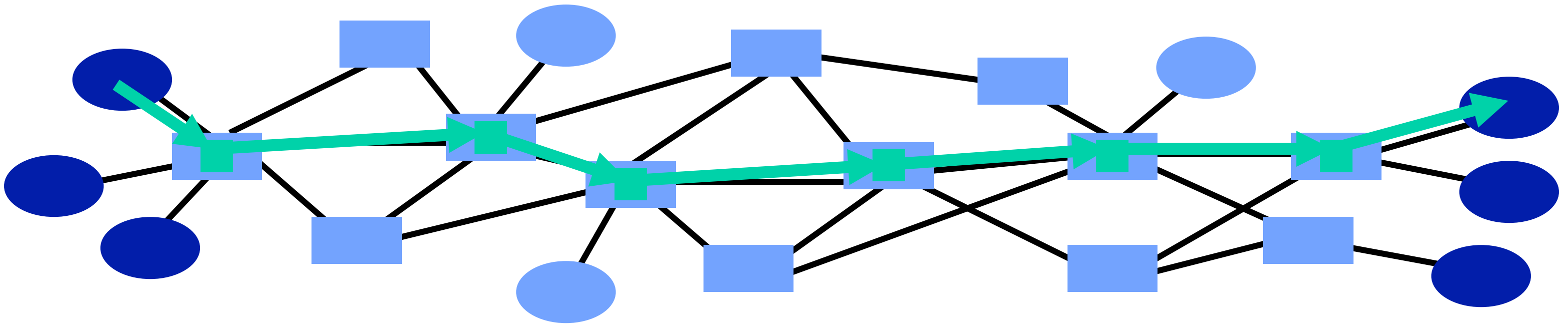
- Guaranteed bandwidth
  - Predictable communication performance
  - Not “best-effort” delivery with no real guarantees
- Simple abstraction
  - Reliable communication channel between hosts
  - No worries about lost or out-of-order packets
- Simple forwarding
  - Forwarding based on time slot or frequency
  - No need to inspect a packet header
- Low per-packet overhead
  - Forwarding based on time slot or frequency
  - No IP (and TCP/UDP) header on each packet

Disadvantages of Circuit Switching

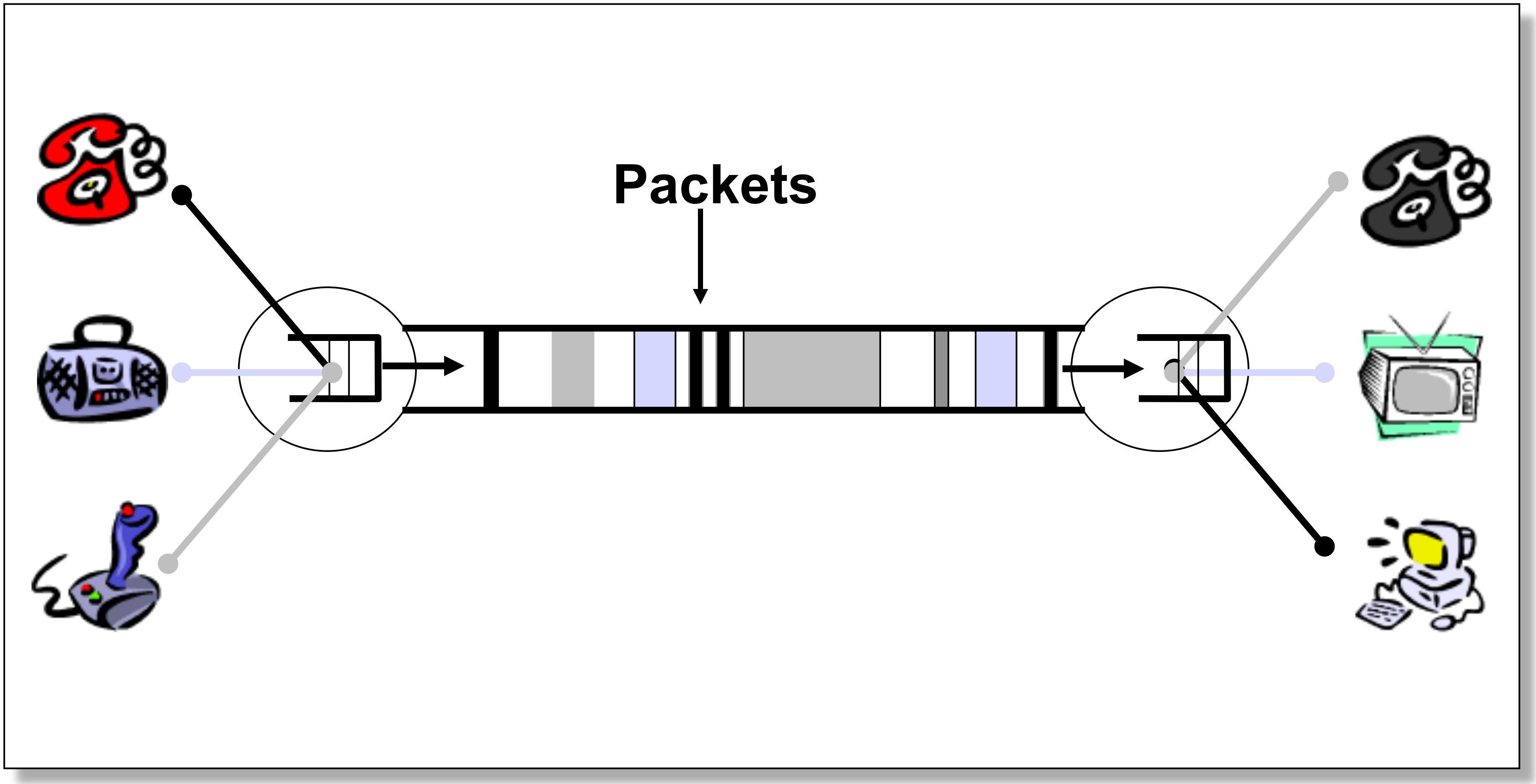
- Wasted bandwidth
  - Bursty traffic leads to idle connection during silent period
  - Unable to achieve gains from statistical multiplexing
- Blocked connections
  - Connection refused when resources are not sufficient
  - Unable to offer “okay” service to everybody
- Connection set-up delay
  - No communication until the connection is set up
  - Unable to avoid extra latency for small data transfers
- Network state
  - Network nodes must store per-connection information
  - Unable to avoid per-connection storage and state

Packet Switching (e.g., Internet)

- Data traffic divided into packets
  - Each packet contains a header (with address of the source and destination)
- Packets travel separately through network
  - Packet forwarding based on the header
  - Network nodes may store packets temporarily
- Destination reconstructs the message

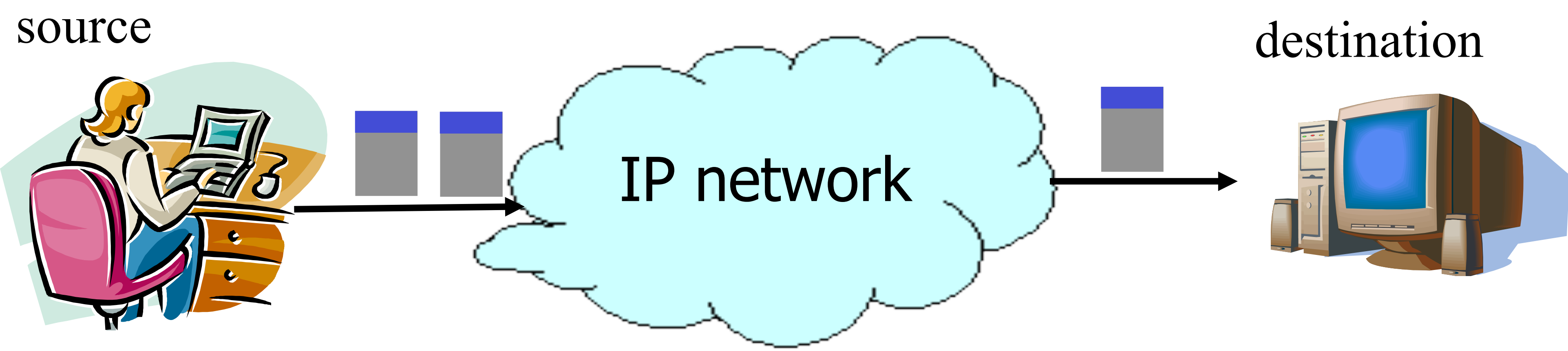


Packet Switching: Statistical Multiplexing



IP Service: Best-Effort Packet Delivery

- Packet switching
  - Divide messages into a sequence of packets
  - Headers with source and destination address
- Best-effort delivery
  - Packets may be lost
  - Packets may be corrupted
  - Packets may be delivered out of order



IP Service Model: Why Packets?

- Data traffic is bursty
  - Logging in to remote machines
  - Exchanging e-mail messages
- Don't want to waste reserved bandwidth
  - No traffic exchanged during idle periods
- Better to allow multiplexing
  - Different transfers share access to same links
- Packets can be delivered by almost anything
  - RFC 2549: IP over Avian Carriers (aka birds)
- … still, packet switching can be inefficient
  - Extra header bits on every packet

IP Service Model: Why Best-Effort?

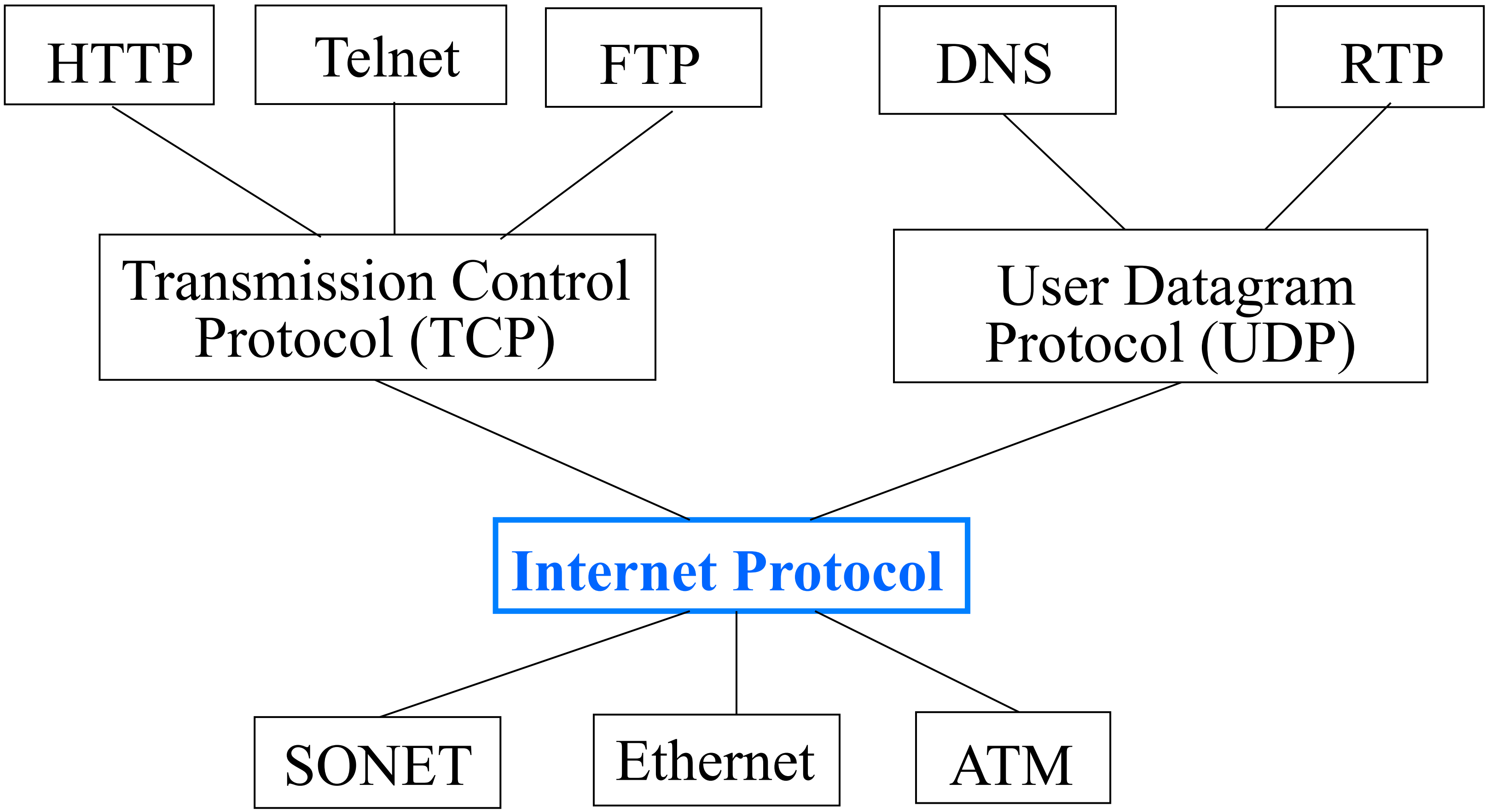
- IP means never having to say you're sorry…
  - Don't need to reserve bandwidth and memory
  - Don't need to do error detection & correction
  - Don't need to remember from one packet to next
- Easier to survive failures
  - Transient disruptions are okay during failover

- … but, applications *do* want efficient, accurate transfer of data in order, in a timely fashion

IP Service: Best-Effort is Enough

- No error detection or correction
  - Higher-level protocol can provide error checking
- Successive packets may not follow the same path
  - Not a problem as long as packets reach the destination
- Packets can be delivered out-of-order
  - Receiver can put packets back in order (if necessary)
- Packets may be lost or arbitrarily delayed
  - Sender can send the packets again (if desired)
- No network congestion control (beyond "drop")
  - Sender can slow down in response to loss or delay

Layering in the IP Protocols



How to get to a destination: hop-by-hop Packet Forwarding

- Each router has a forwarding table
  - Maps destination addresses…
  - … to outgoing interfaces
- Upon receiving a packet
  - Inspect the destination IP address in the header
  - Index into the table
  - Determine the outgoing interface
  - Forward the packet out that interface
- Then, the next router in the path repeats
  - And the packet travels along the path to the destination

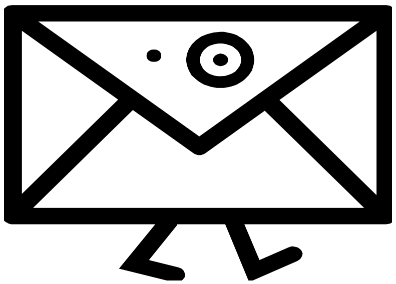
Where do Forwarding Tables Come From?

- Routers have forwarding tables
  - Map address (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…

What's routing?

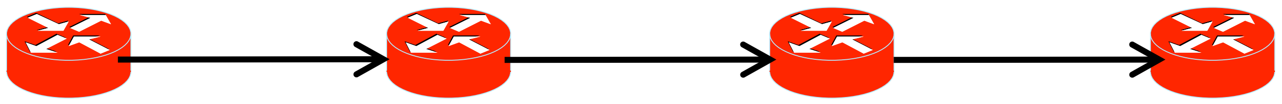
- A famous quotation from RFC 791

  "A *name* indicates what we seek.
  An *address* indicates where it is.
  A *route* indicates how we get there."
  -- Jon Postel



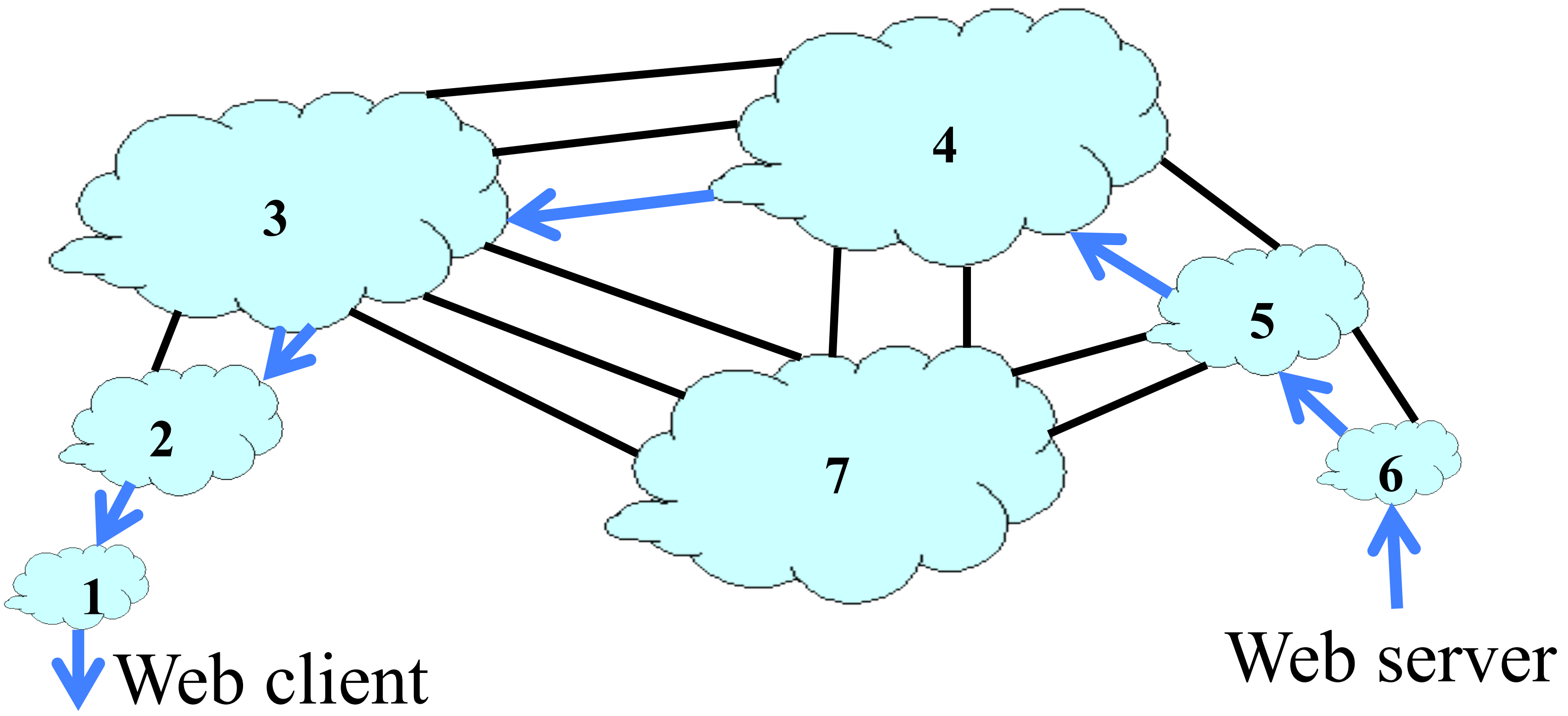
Routing vs. Forwarding

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



Internet Structure

- Federated network of Autonomous Systems
  - Routers and links controlled by a single entity
  - Routing between ASes, and within an AS

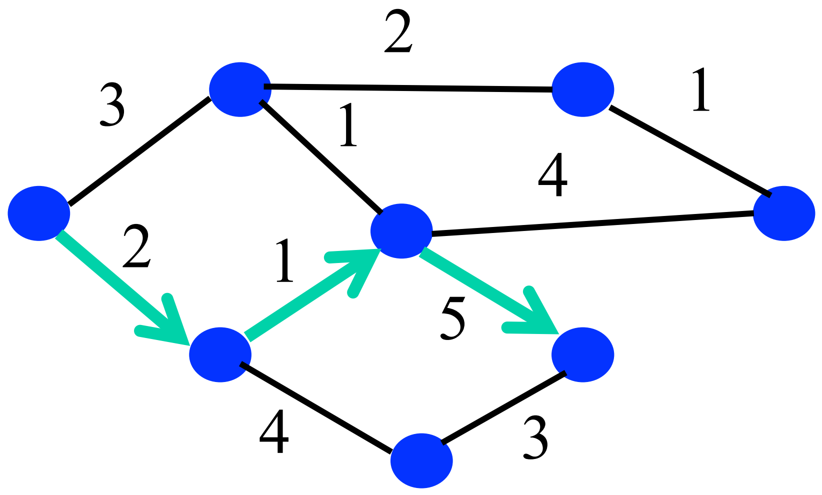


Two-Tiered Internet Routing System

- Interdomain routing: between ASes
  - Routing policies based on *business relationships*
  - No common metrics, and limited cooperation
  - BGP: policy-based, path-vector routing protocol
- Intradomain routing: within an AS
  - Shortest-path routing based on *link metrics*
  - Routers all managed by a single institution
  - OSPF and IS-IS: link-state routing protocol
  - RIP and EIGRP: distance-vector routing protocol

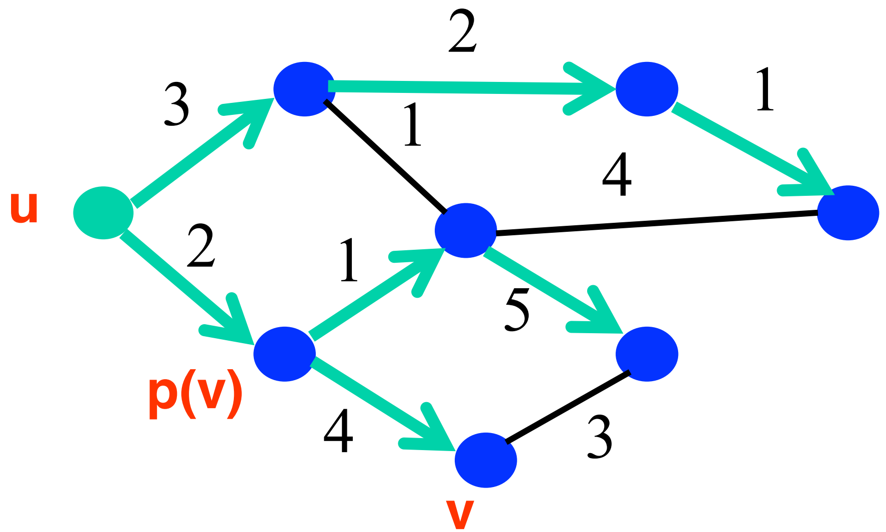
Intradomain routing

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



Shortest-Path Problem

- Given: network topology with link costs/weights
  - **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



Ways to Compute Shortest Paths

- Link-state
  - Every node collects complete information about network topology and link costs
  - Each computes shortest paths from it
  - Each generates 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

Dijkstra's Shortest-Path Algorithm

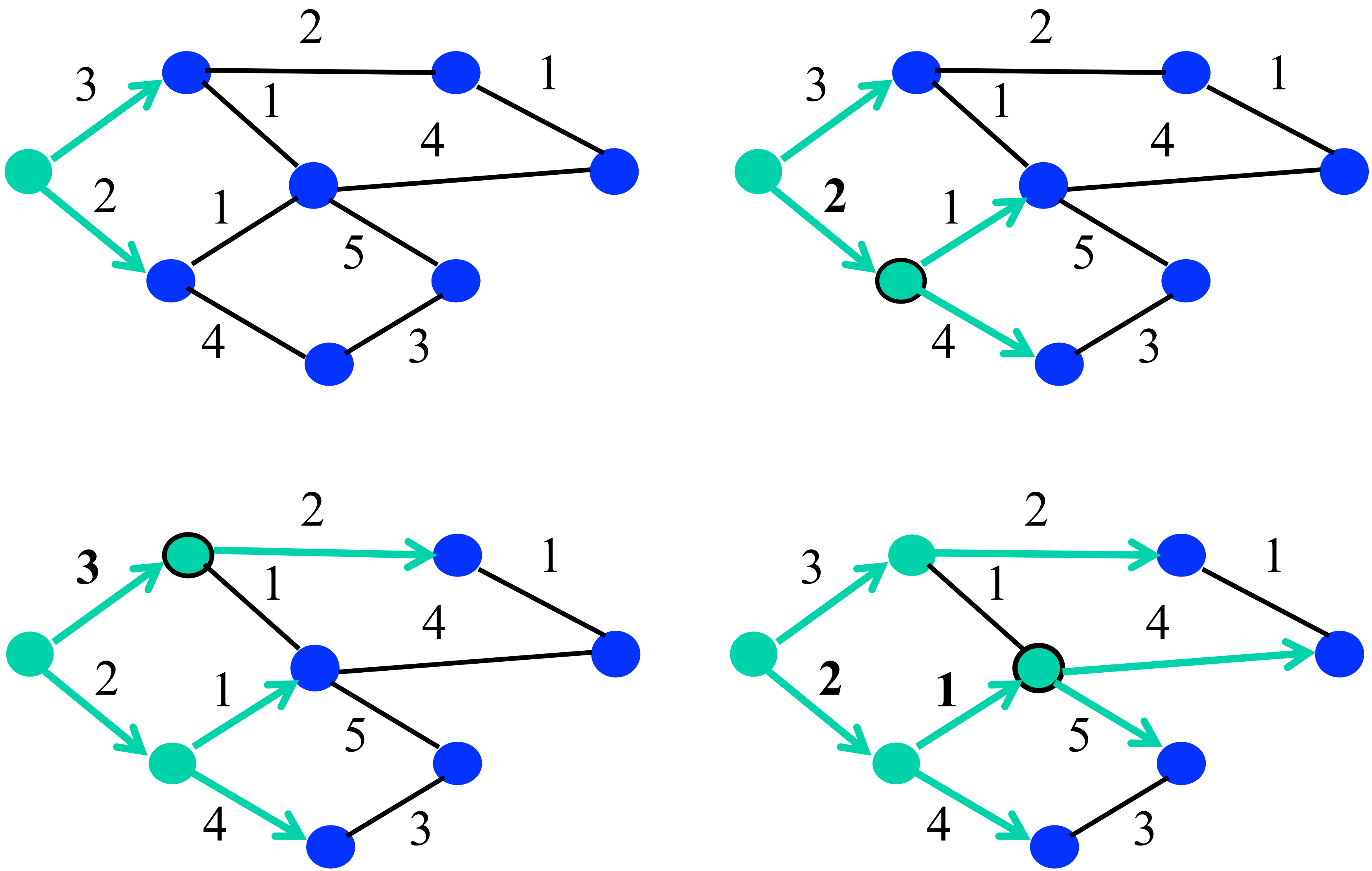
- Iterative algorithm
  - After k iterations, know least-cost path to k nodes
- **S**: nodes whose least-cost path definitively known
  - Initially, **S = {u}** where u is the source node
  - Add one node to S in each iteration
- **D(v)**: current cost of path from source to node v
  - Initially, **D(v)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

Dijsktra's Algorithm

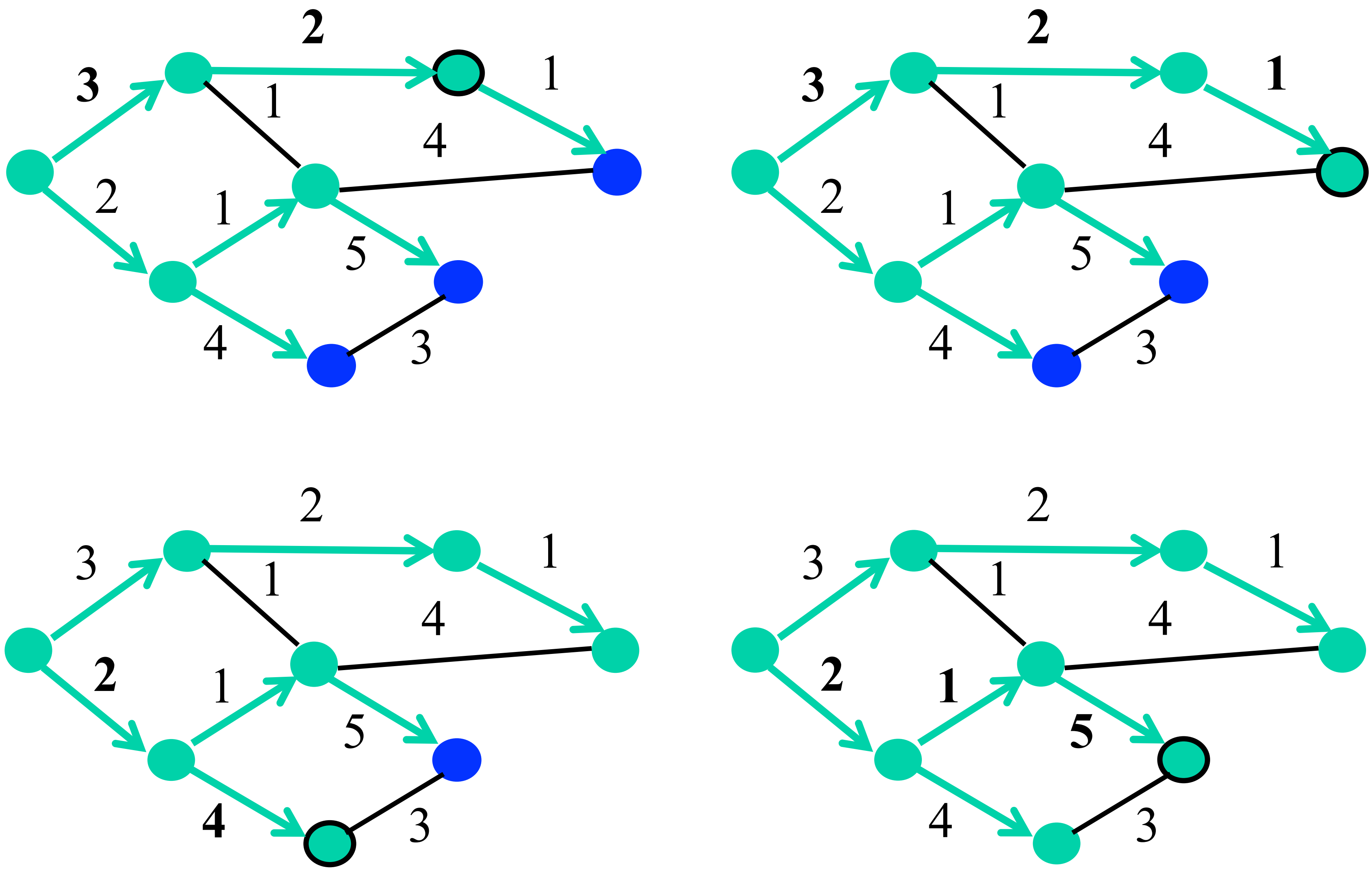
```
Initialization:
  S = {u}
  for all nodes v
    if v adjacent to u {
        D(v) = c(u,v)
    else D(v) = ∞

Loop
  find w not in S with the smallest D(w)
  add w to S
  update D(v) for all v adjacent to w and not in S:
      D(v) = min{D(v), D(w) + c(w,v)}
until all nodes in S
```

Dijkstra’s Algorithm Example

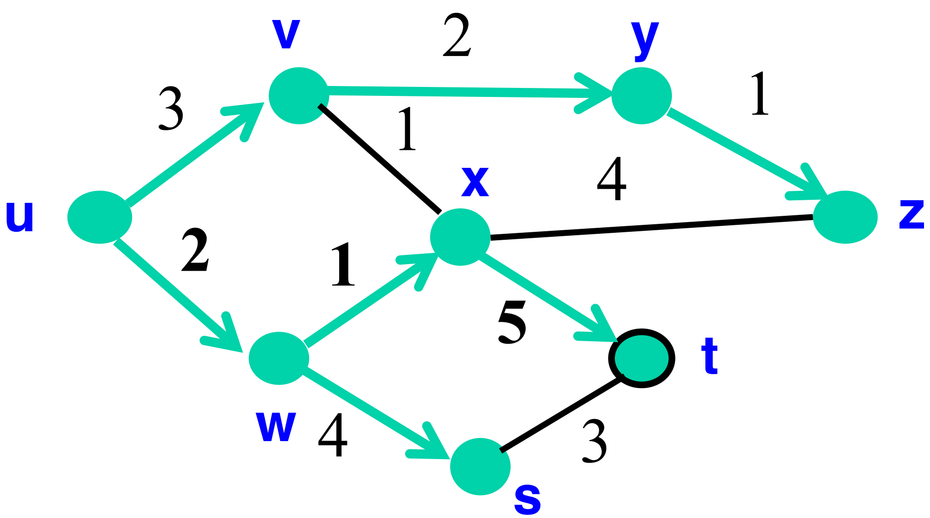


Dijkstra's Algorithm Example



Shortest-Path Tree

- Shortest-path tree from u



- Forwarding table at u

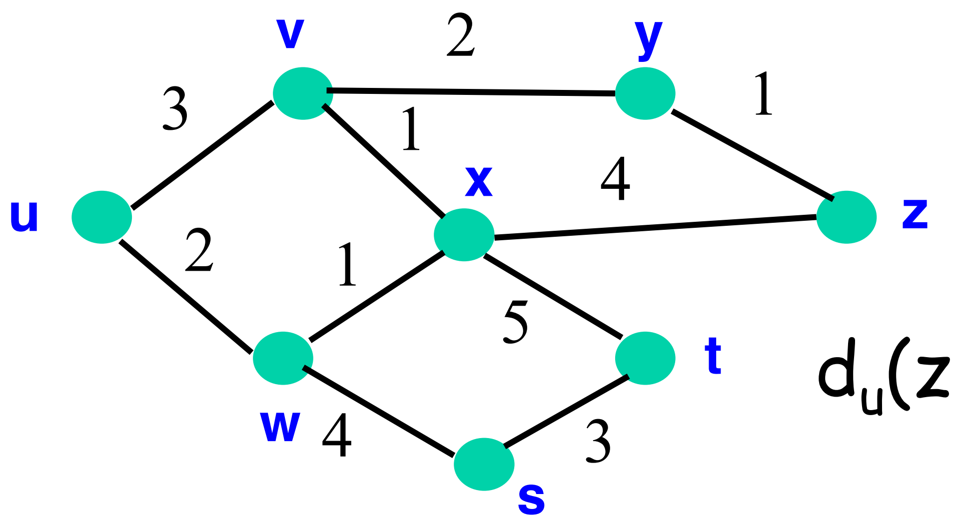
	link
v	(u,v)
w	(u,w)
x	(u,w)
y	(u,v)
z	(u,v)
s	(u,w)
t	(u,w)

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)

Bellman-Ford Algorithm

- Define distances at each node x
  - $d_x(y)$ = cost of least-cost path from x 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 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 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:

wait for (change in local link cost or message from neighbor)

recompute estimates

if DV to any destination has changed, *notify* neighbors

Distance Vector Example: Step 0

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

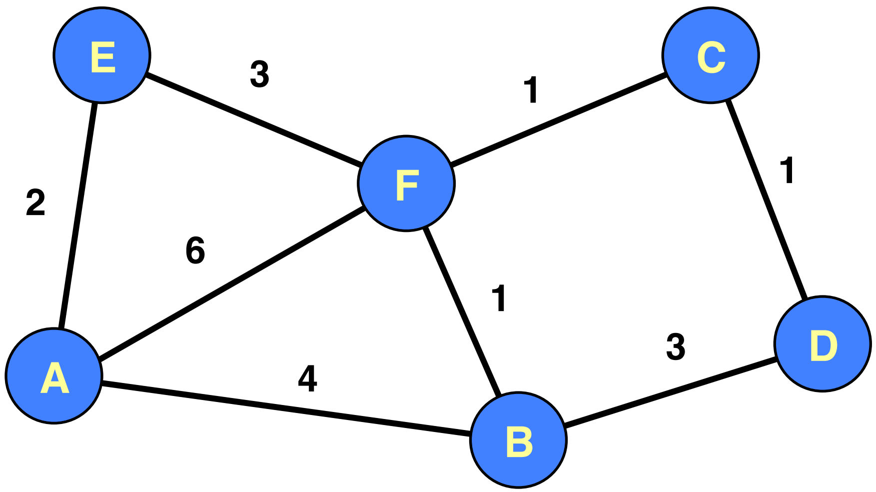


Table for C			Table for D			Table for E			Table for F		
Dst	Cst	Hop	Dst	Cst	Hop	Dst	Cst	Hop	Dst	Cst	Hop
A	∞	-	A	∞	-	A	2	A	A	6	A
B	∞	-	B	3	B	B	∞	-	B	1	B
C	0	C	C	1	C	C	∞	-	C	1	C
D	1	D	D	0	D	D	∞	-	D	∞	-
E	∞	-	E	∞	-	E	0	E	E	3	E
F	1	F	F	∞	-	F	3	F	F	0	F

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

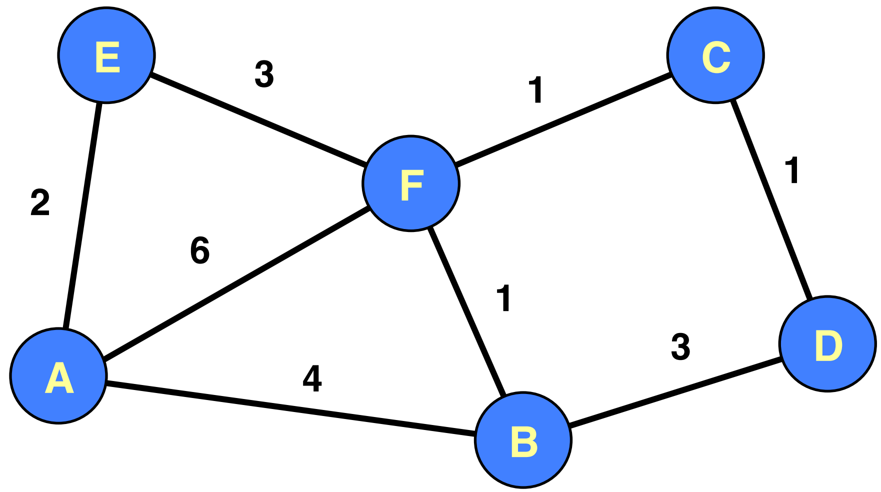


Table for C			Table for D			Table for E			Table for F		
Dst	Cst	Hop	Dst	Cst	Hop	Dst	Cst	Hop	Dst	Cst	Hop
A	7	F	A	7	B	A	2	A	A	5	B
B	2	F	B	3	B	B	4	F	B	1	B
C	0	C	C	1	C	C	4	F	C	1	C
D	1	D	D	0	D	D	∞	-	D	2	C
E	4	F	E	∞	-	E	0	E	E	3	E
F	1	F	F	2	C	F	3	F	F	0	F

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

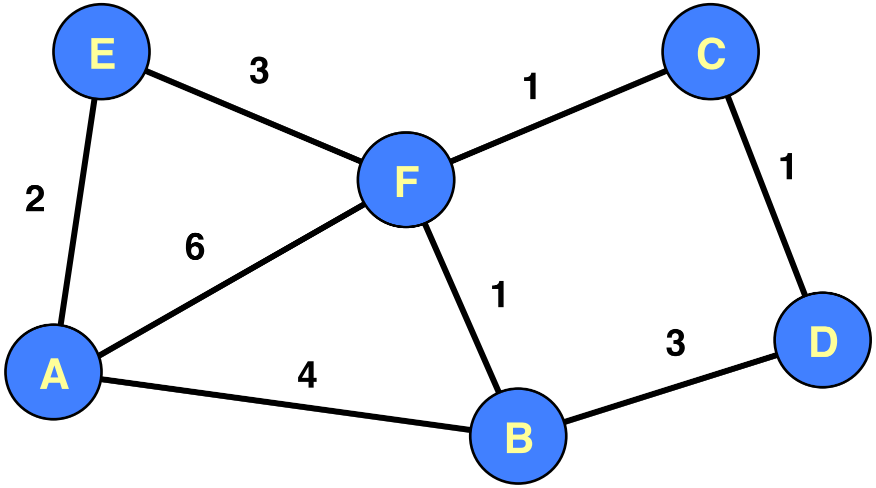
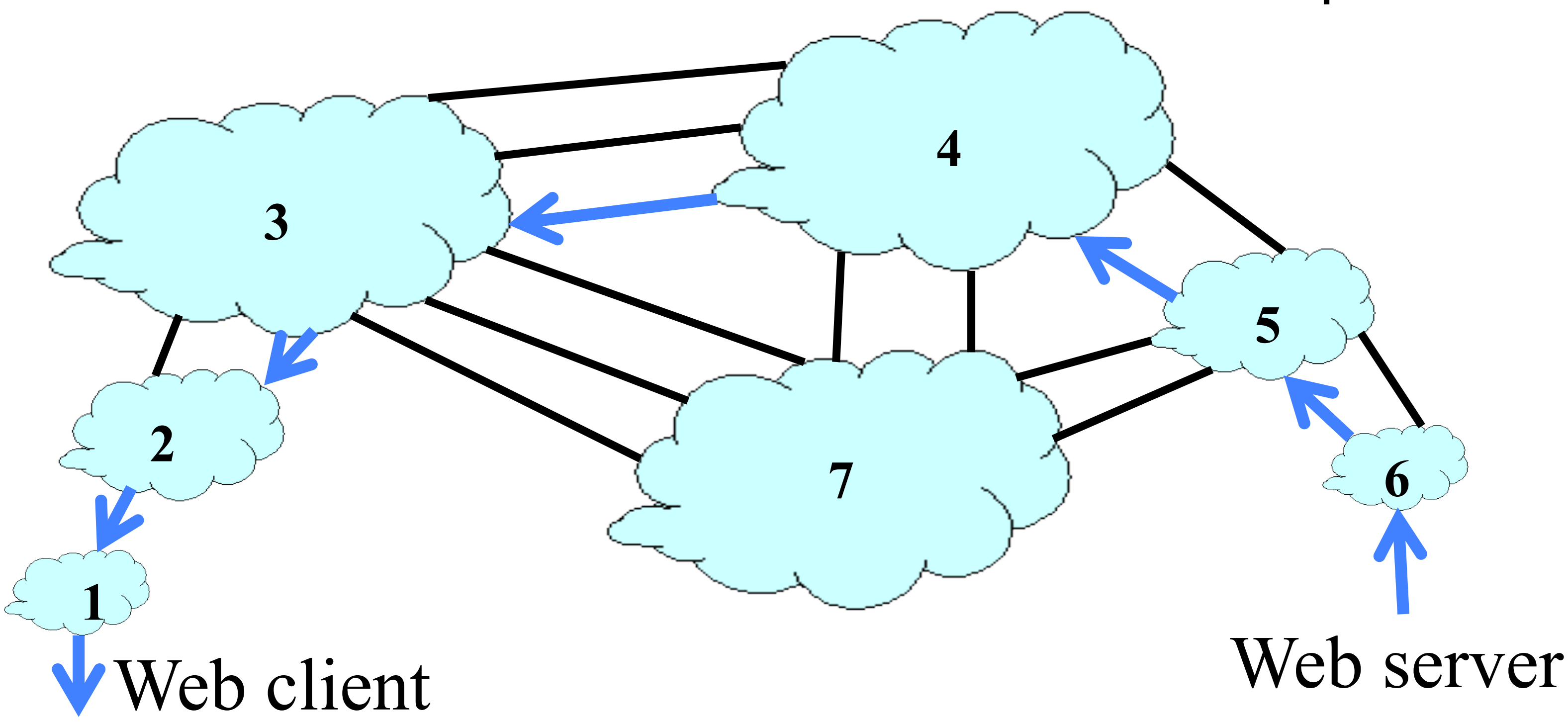


Table for C			Table for D			Table for E			Table for F		
Dst	Cst	Hop	Dst	Cst	Hop	Dst	Cst	Hop	Dst	Cst	Hop
A	6	F	A	7	B	A	2	A	A	5	B
B	2	F	B	3	B	B	4	F	B	1	B
C	0	C	C	1	C	C	4	F	C	1	C
D	1	D	D	0	D	D	5	F	D	2	C
E	4	F	E	5	C	E	0	E	E	3	E
F	1	F	F	2	C	F	3	F	F	0	F

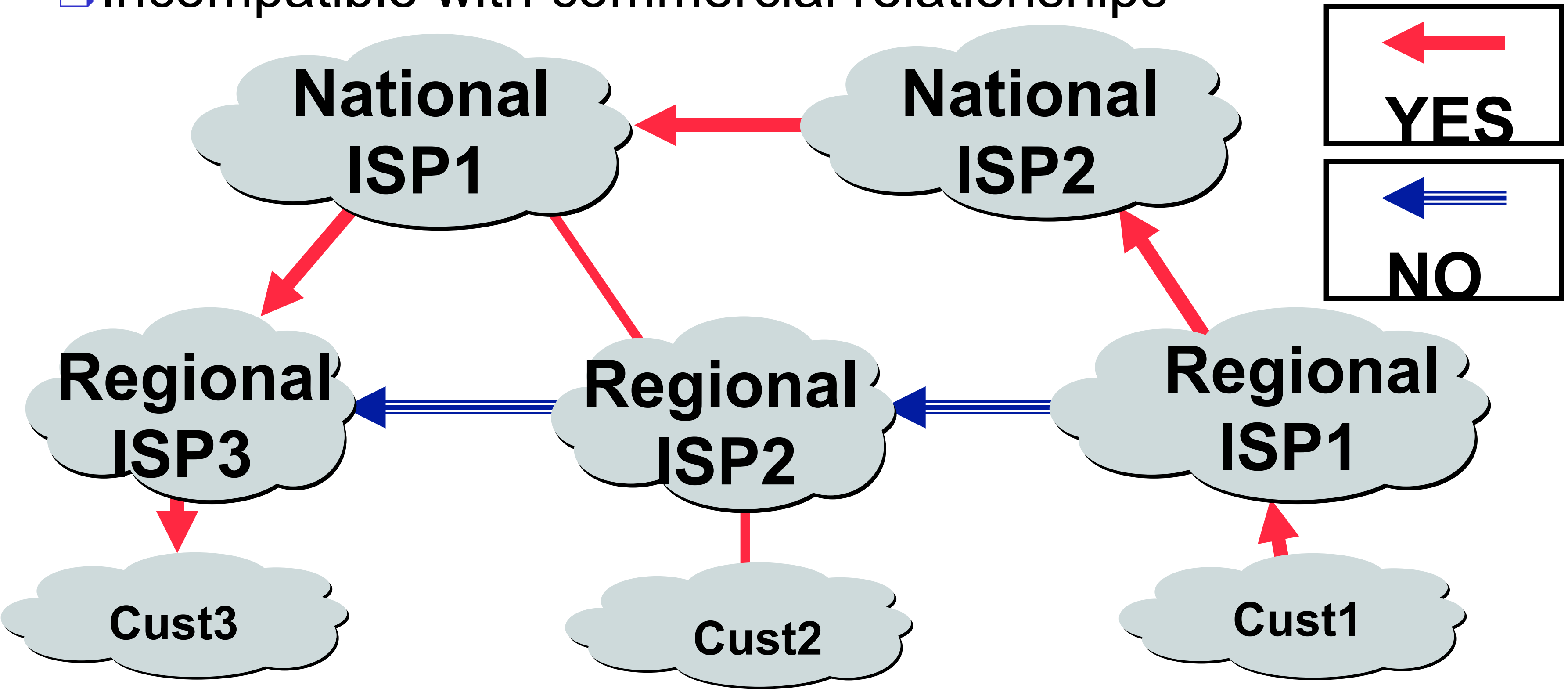
Interdomain Routing

- AS-level topology
  - Nodes are Autonomous Systems (ASes)
  - Links are connections & business relationships



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

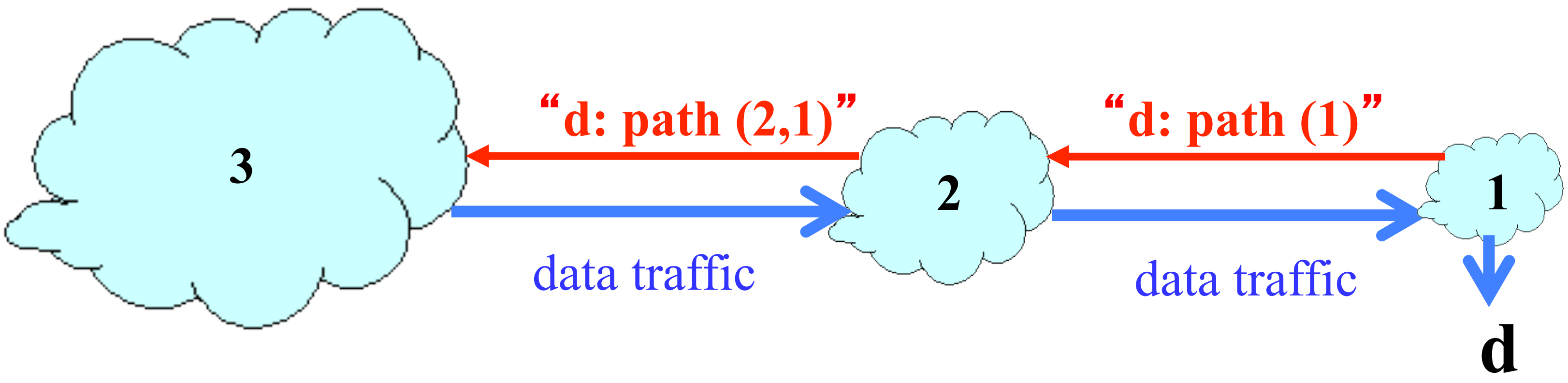
Distance Vector is on the Right Track

- Advantages
  - Hides details of the network topology
  - Nodes determine only "next hop" toward the dest
- Disadvantages
  - Minimizes some notion of total distance, which is difficult in an interdomain setting

- Idea: extend the notion of a distance vector

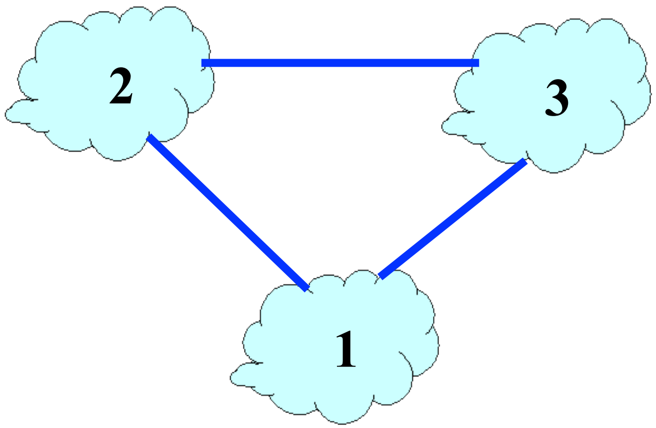
Path-Vector Routing

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



Flexible Policies

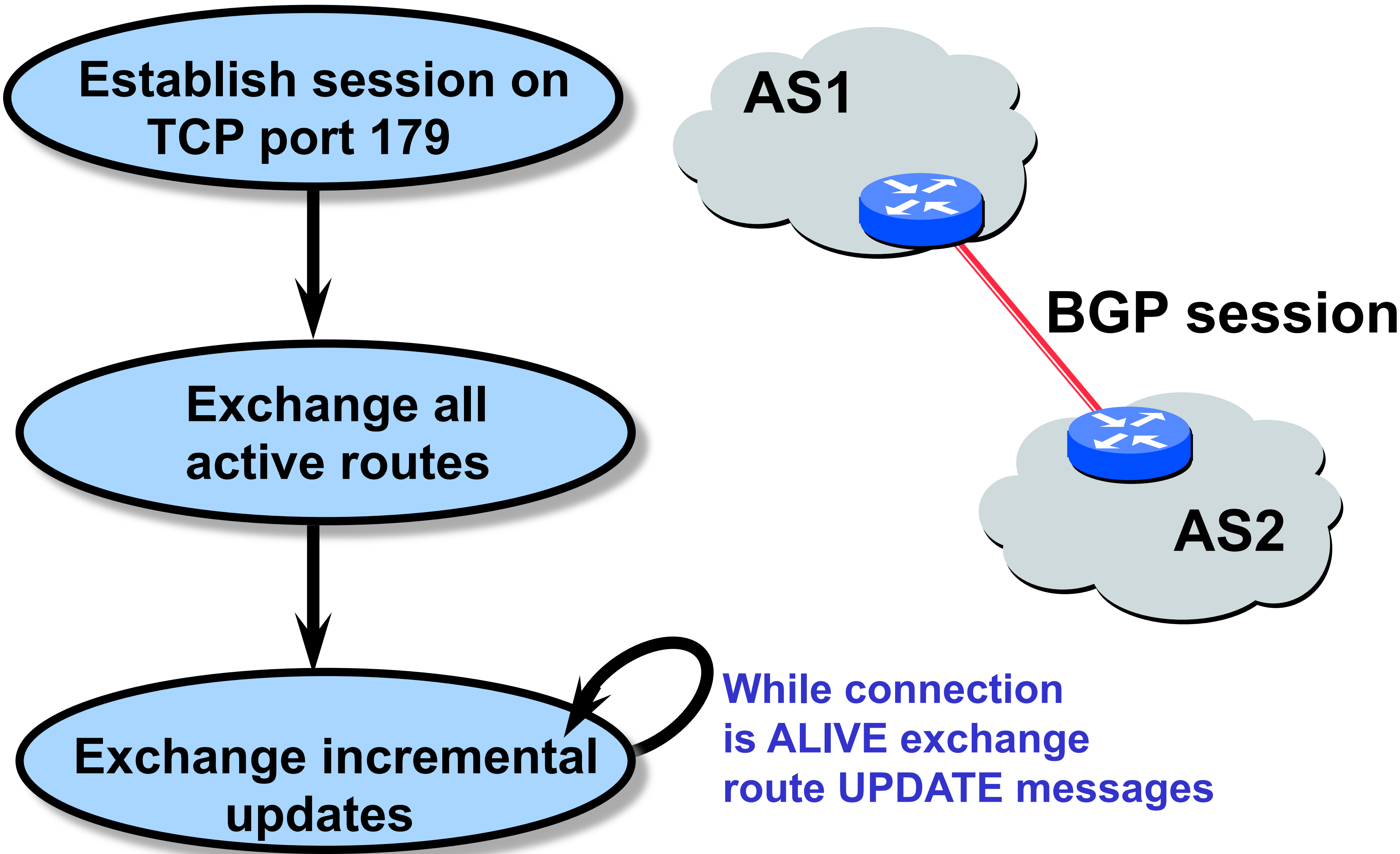
- Each node can apply local policies
  - Path selection: Which path to use?
  - Path export: Which path to advertise?
- Examples
  - Node 2 may prefer the path "2, 3, 1" over "2, 1"
  - Node 1 may not let node 3 hear the path "1, 2"



Border Gateway Protocol

- Interdomain routing protocol for the Internet
  - Path-vector protocol
  - Policy-based routing based on AS Paths
  - Evolved during the past 15 years
    - **1989 : BGP-1 [RFC 1105]**
      - **Replacement for EGP (1984, RFC 904)**
    - **1990 : BGP-2 [RFC 1163]**
    - **1991 : BGP-3 [RFC 1267]**
    - **1995 : BGP-4 [RFC 1771]**
      - **Support for Classless Interdomain Routing (CIDR)**

BGP Operations

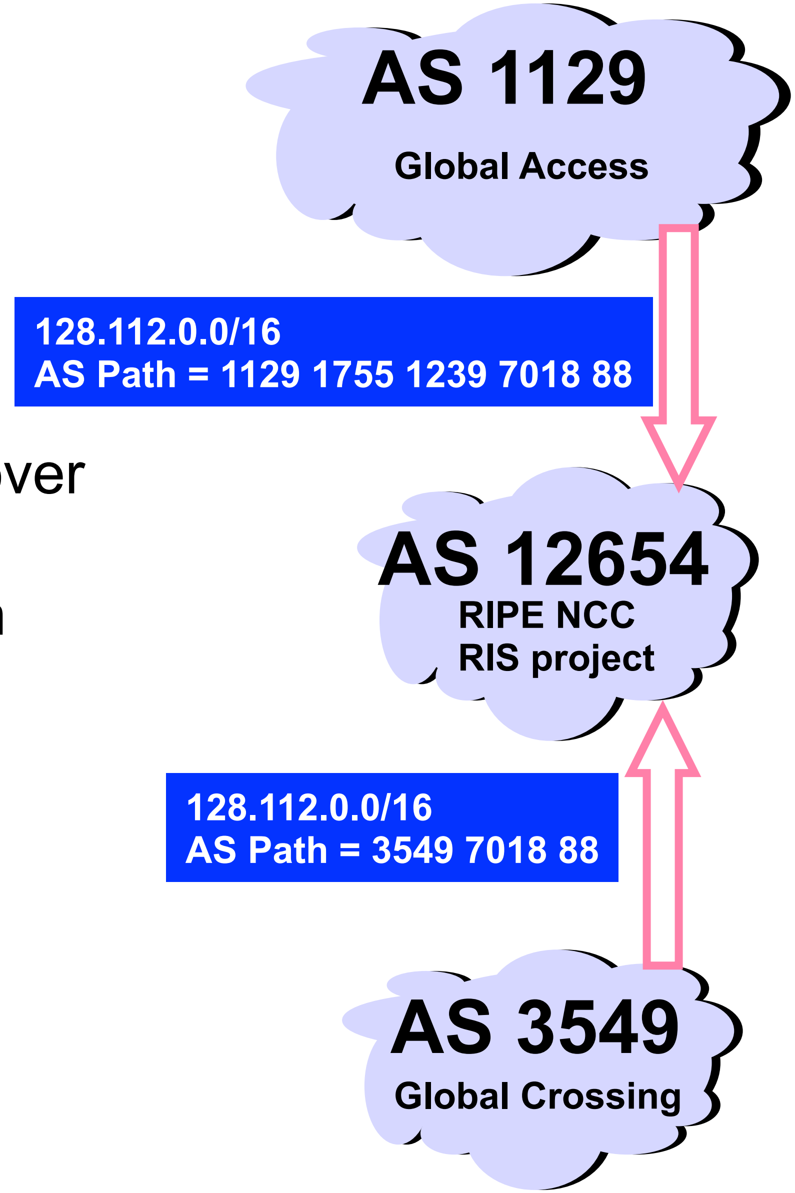


Incremental Protocol

- 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 Path Selection

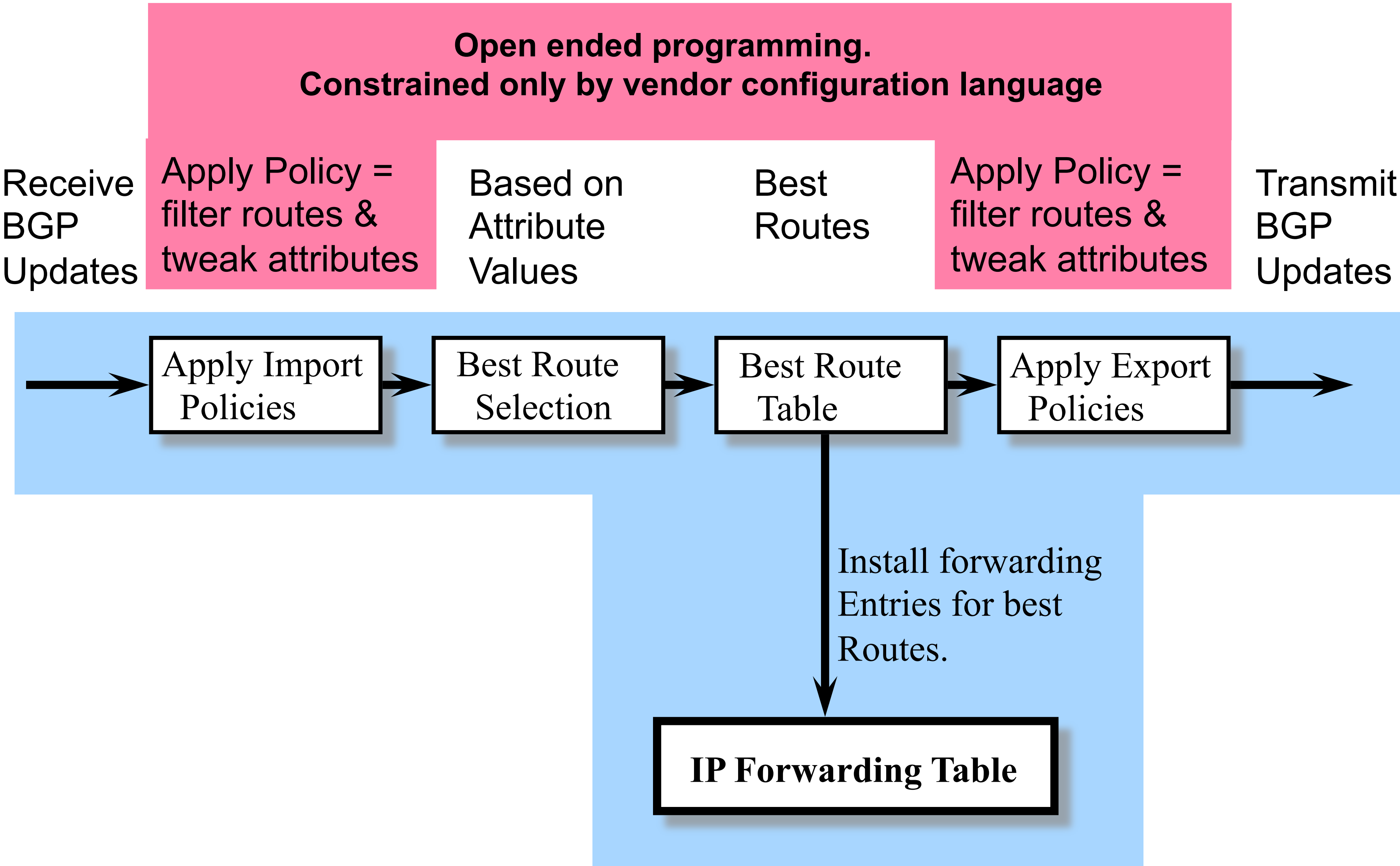
- Simplest case
  - Shortest AS path
  - Arbitrary tie break
- Example
  - Three-hop AS path preferred over a four-hop AS path
  - AS 12654 prefers path through Global Crossing
- But, BGP is not limited to shortest-path routing
  - Policy-based routing



BGP Policy: Applying Policy to Routes

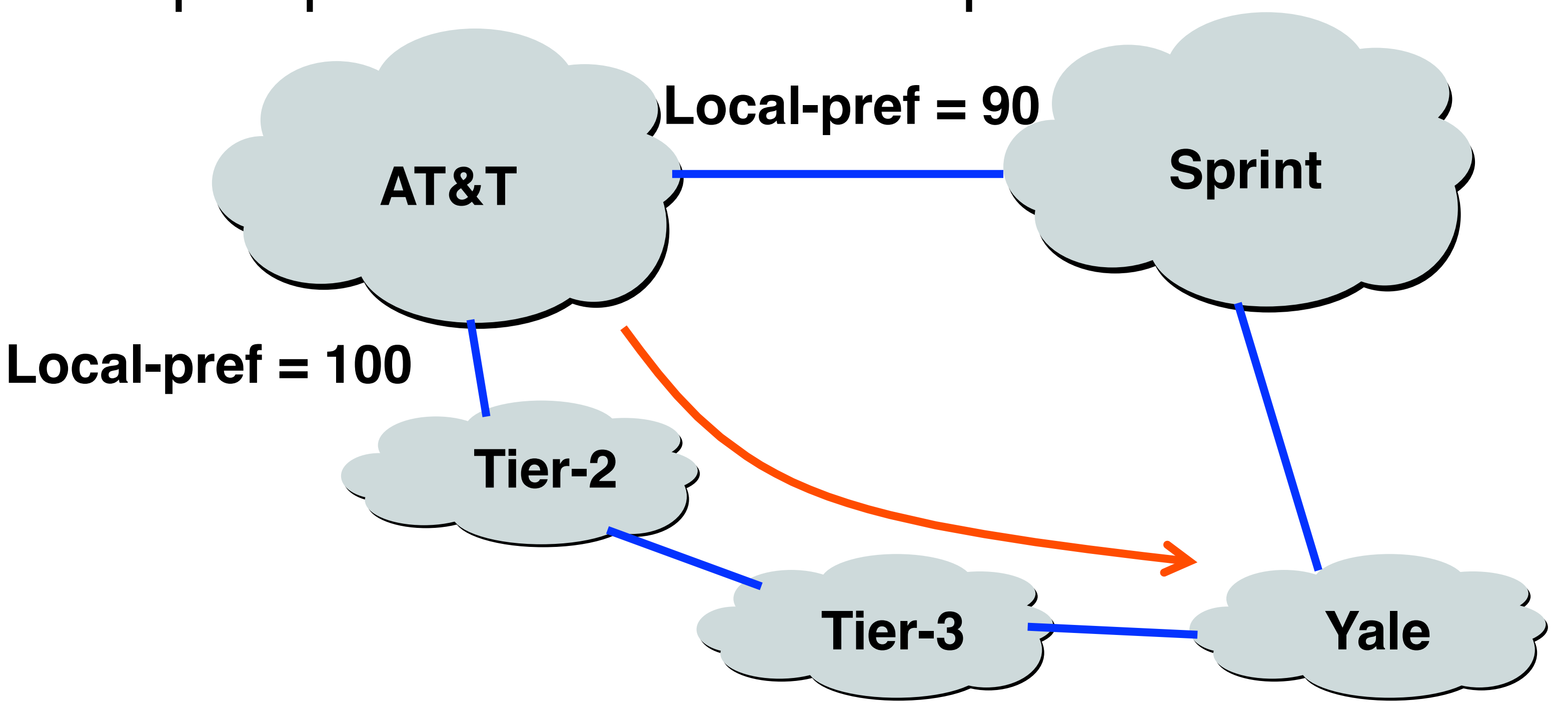
- Import policy
  - Filter unwanted routes from neighbor
    - E.g. prefix that your customer doesn't own
  - Manipulate attributes to influence path selection
    - E.g., assign local preference to favored routes
- Export policy
  - Filter routes you don't want to tell your neighbor
    - E.g., don't 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

BGP Policy: Influencing Decisions



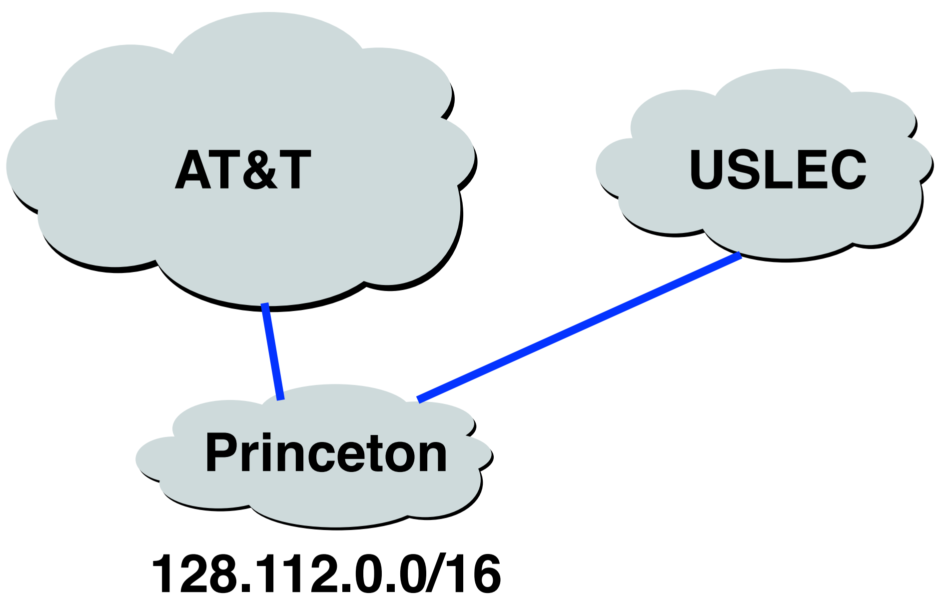
Import Policy: Local Preference

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



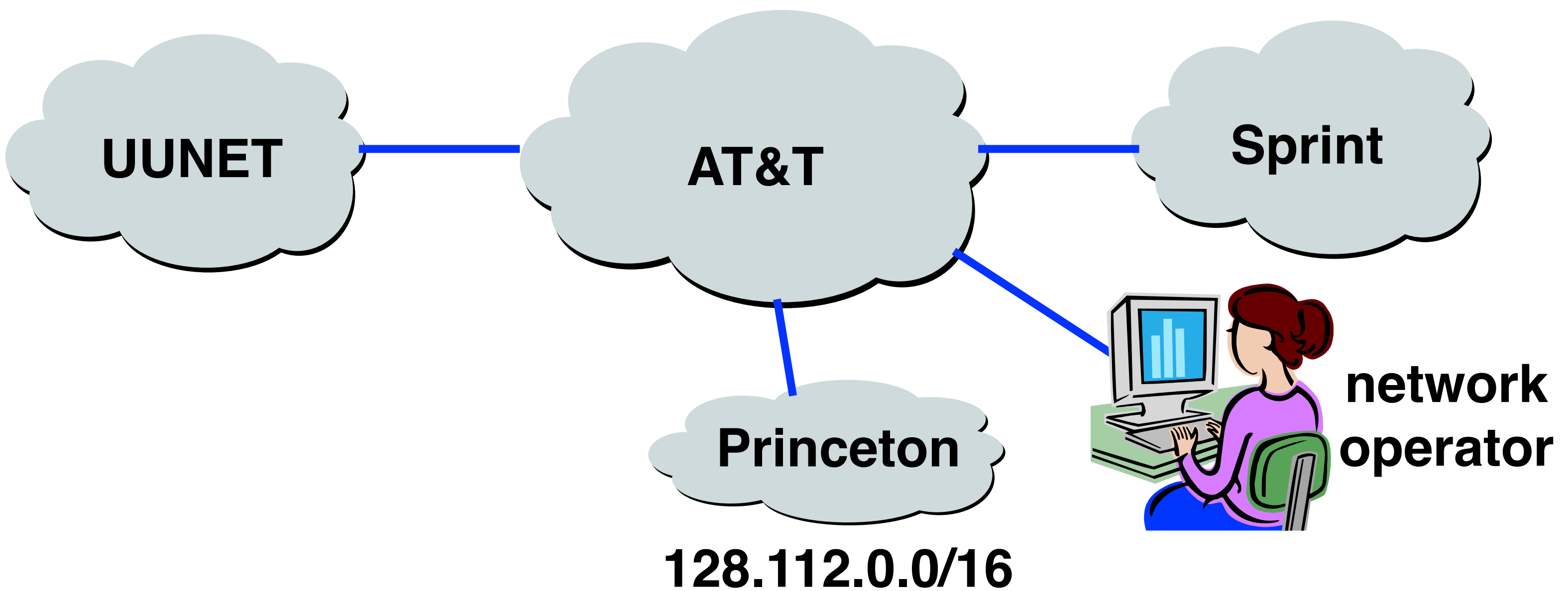
Import Policy: Filtering

- 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



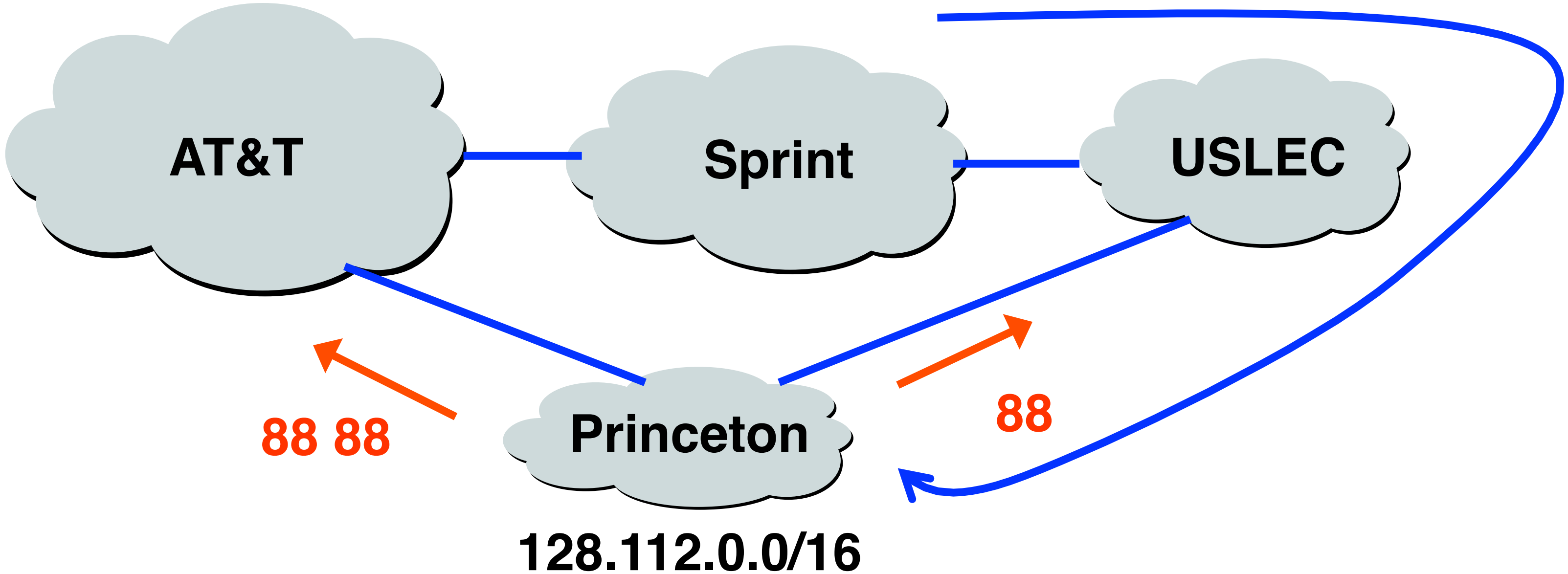
Export Policy: Filtering

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



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



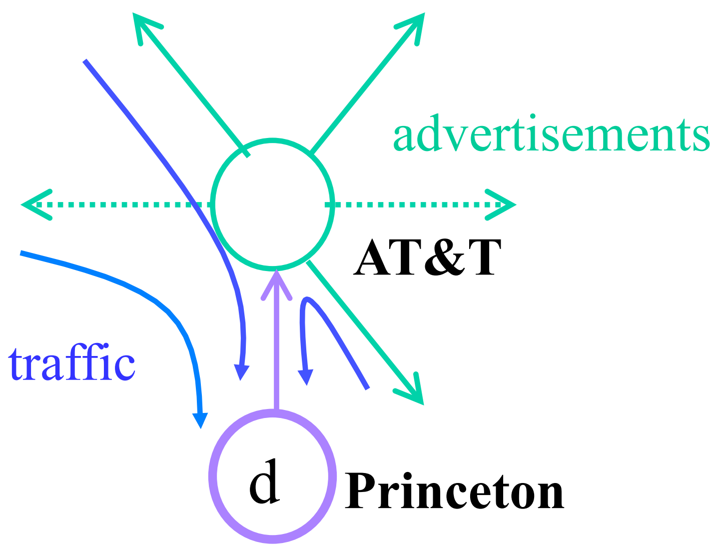
BGP Policies in Practice (Gao-Rexford model)

- Mainly business relationships
  - Customer-provider
  - Peer-peer
  - ……
- Implementing in BGP
  - Import policy
    - Ranking customer routes over peer routes
  - Export policy
    - Export only customer routes to peers and providers

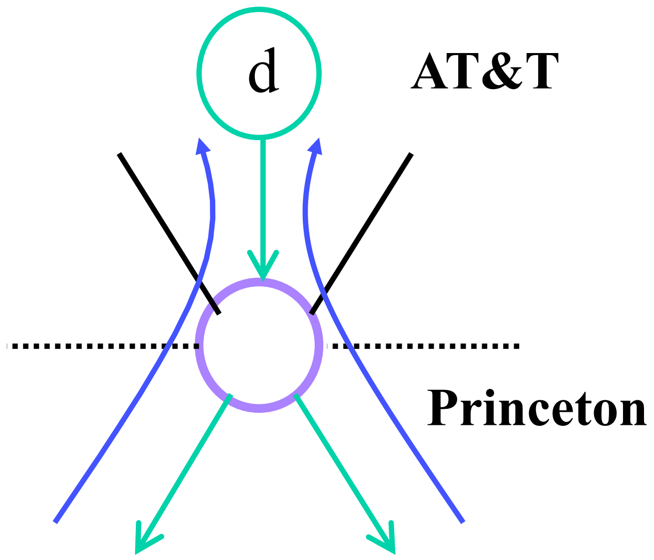
Customer-Provider Relationship

- Customer pays provider for access to Internet
  - Provider exports customer's routes to everybody
  - Customer exports provider's routes to customers

Traffic **to** the customer

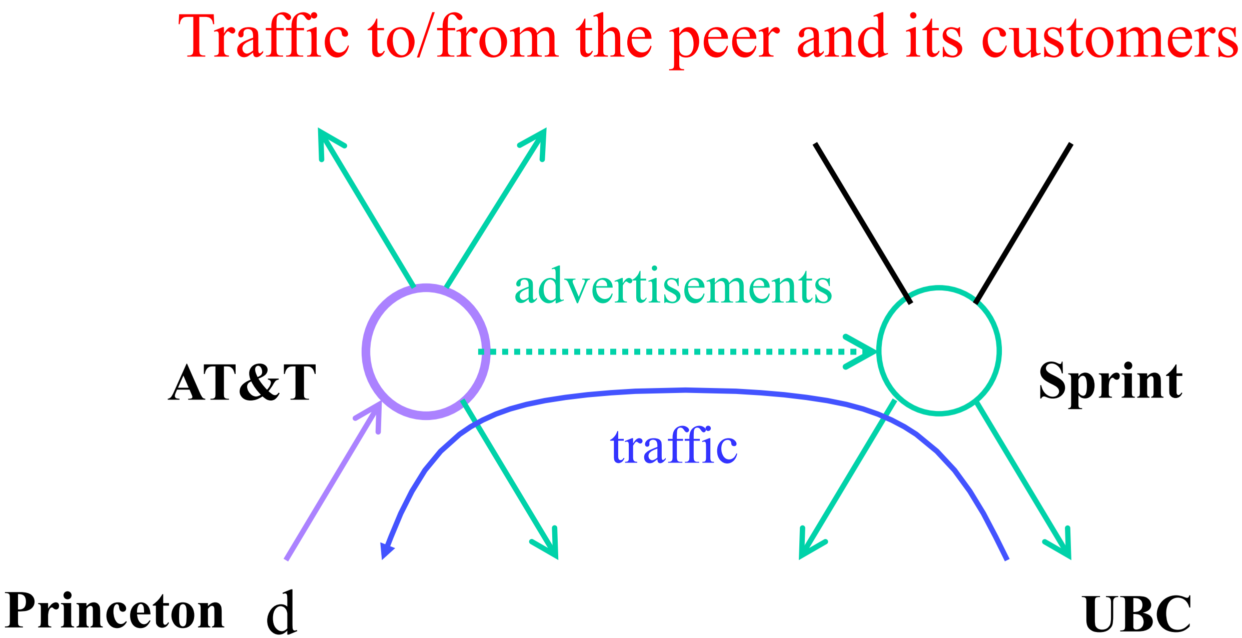


Traffic **from** the customer



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



How Peering Decisions are Made?

Peer

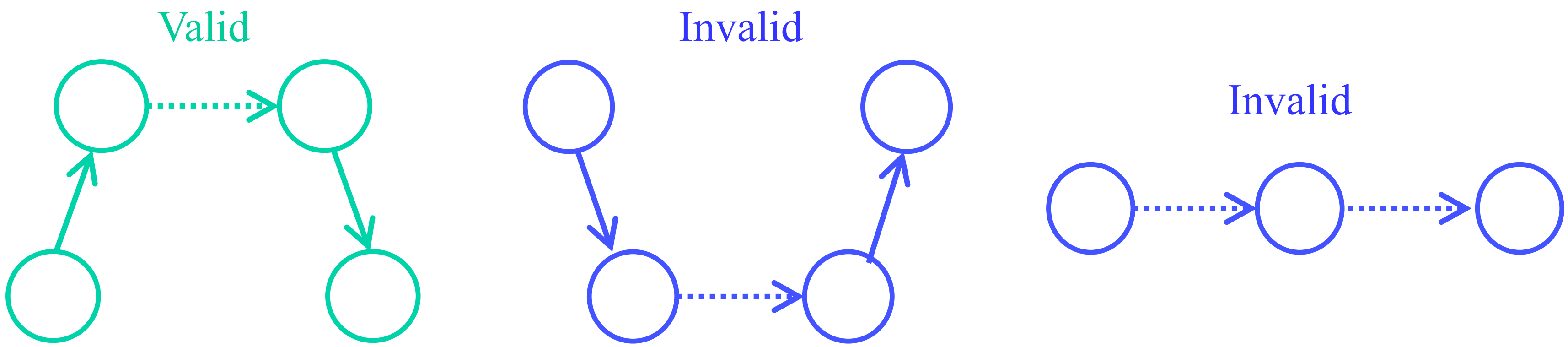
- Reduces upstream transit costs
- Can increase end-to-end performance
- May be the only way to connect your customers to some part of the Internet ("Tier 1")

Don't Peer

- You would rather have customers
- Peers are usually your competition
- Peering relationships may require periodic renegotiation

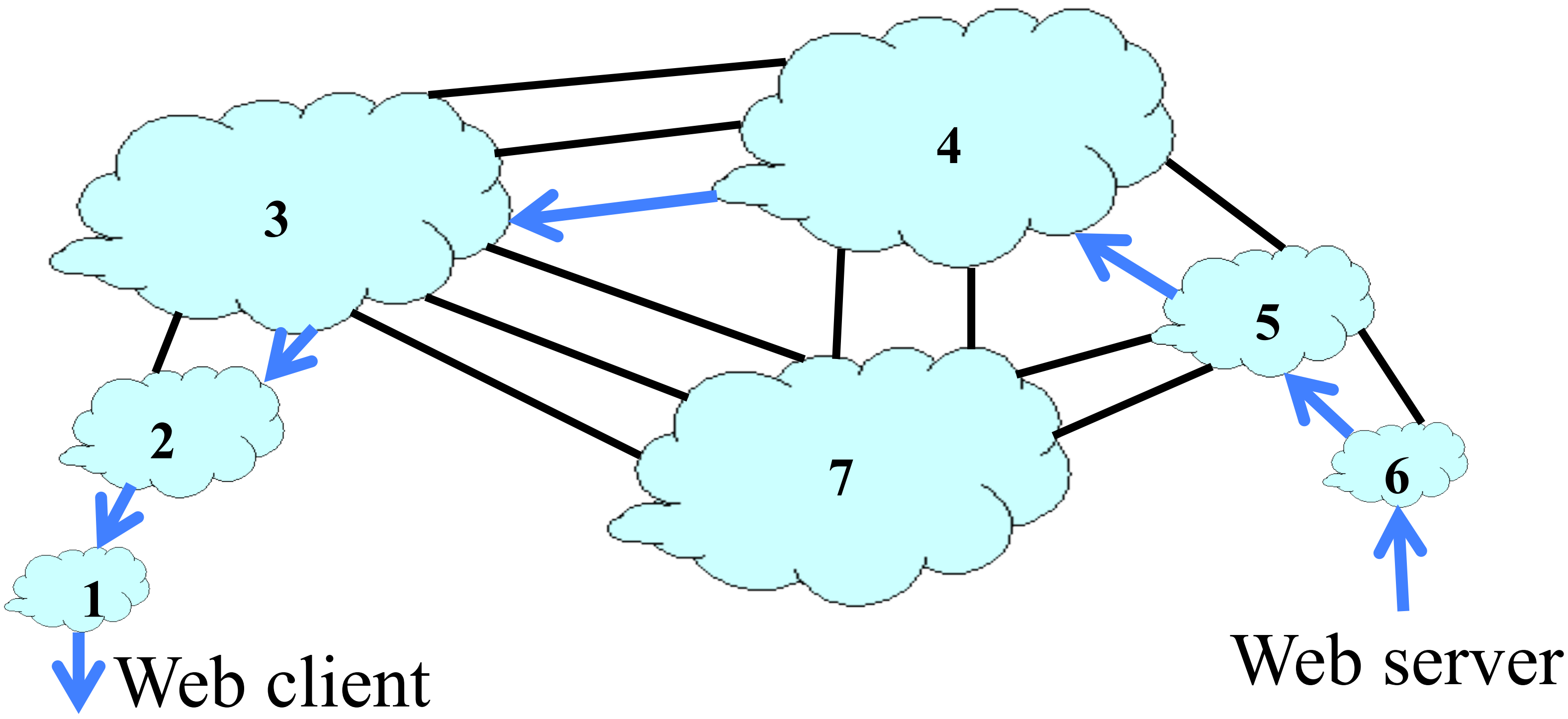
Valid and invalid paths

- AS relationships limit the kinds of valid paths
  - Uphill portion: customer-provider relationships
  - Plateau: zero or one peer-peer edge
  - Downhill portion: provider-customer relationships



Internet Structure

- Federated network of Autonomous Systems (AS)
  - Routers and links controlled by a single entity
  - Routing between ASes, and within an AS



Two-Tiered Internet Routing System

- Intradomain routing: within an AS
  - Shortest-path routing based on *link metrics*
  - Routers all managed by a single institution
  - OSPF and IS-IS: link-state routing protocol
  - RIP and EIGRP: distance-vector routing protocol
- Interdomain routing: between ASes
  - Routing policies based on *business relationships*
  - No common metrics, and limited cooperation
  - BGP: policy-based, path-vector routing protocol

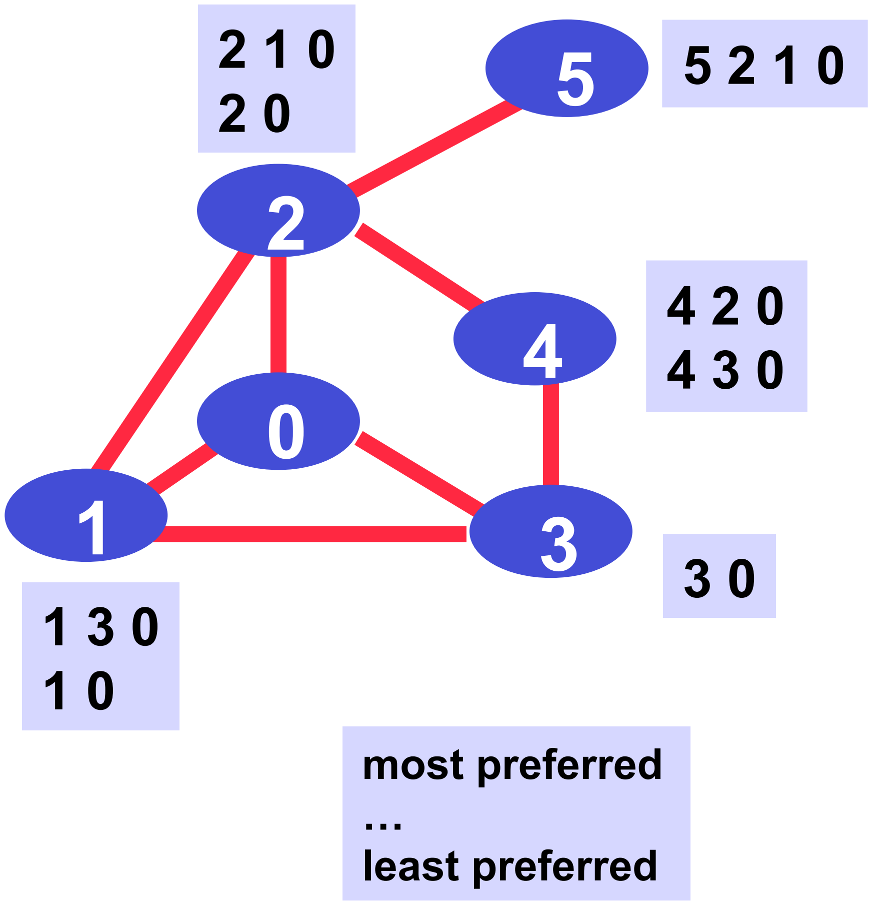
BGP Modeling: What Problem Does BGP Solve?

- Intradomain routings do shortest-path routing
  - Shortest path as sum of link weights
    - Link-state routing (e.g., OSPF and IS-IS)
    - Distance vector routing (e.g., RIP)
- Policy makes BGP more complicated
  - An AS might not tell a neighbor about a path
    - E.g., Sprint can't reach UUNET through AT&T
  - An AS might prefer one path over a shorter one
    - E.g., ISP prefers to send traffic through a customer

What is a good model for BGP?

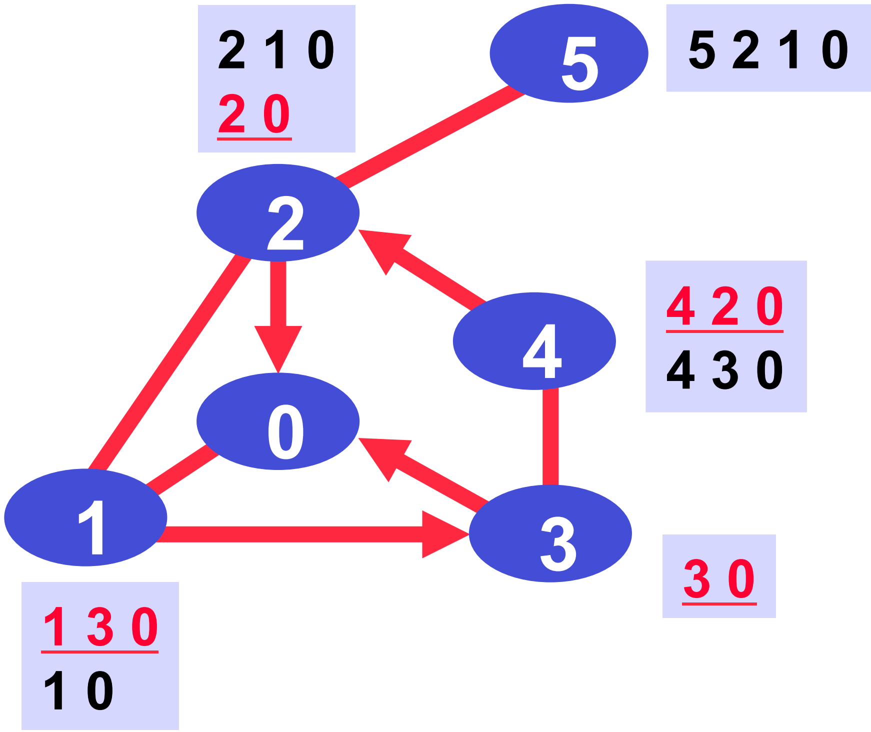
Stable Paths Problem (SPP)

- Node
  - BGP-speaking router
  - Node 0 is destination
- Edge
  - BGP adjacency
- Permitted paths
  - Set of routes to 0 at each node
  - Ranking of the paths



A Solution to a Stable Paths Problem

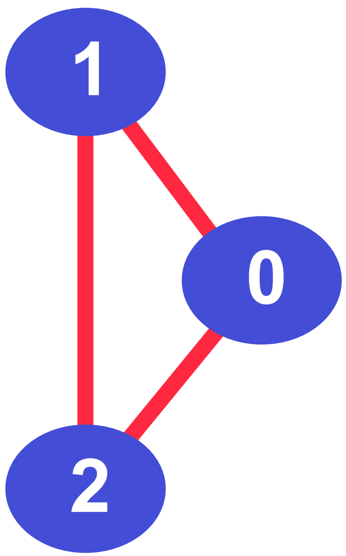
- Solution
  - Path assignment per node
  - Can be the “null” path
- If node u has path uwP
  - {u,w} is an edge in the graph
  - Node w is assigned path wP
- Each node is assigned
  - The highest ranked path consistent with the assignment of its neighbors
- A solution is an in-tree rooted at the destination



A solution need not represent a shortest path tree, or a spanning tree.

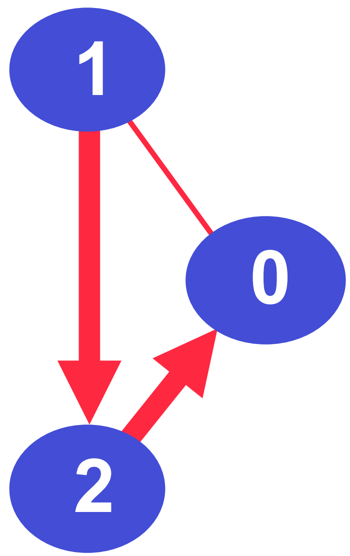
A SPP May Have Multiple Solutions

1 2 0
1 0



2 1 0
2 0

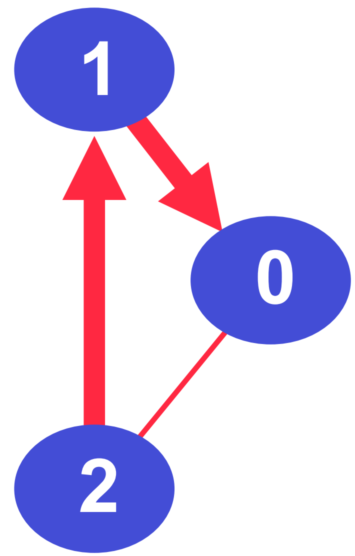
1 2 0
1 0



2 1 0
2 0

First solution

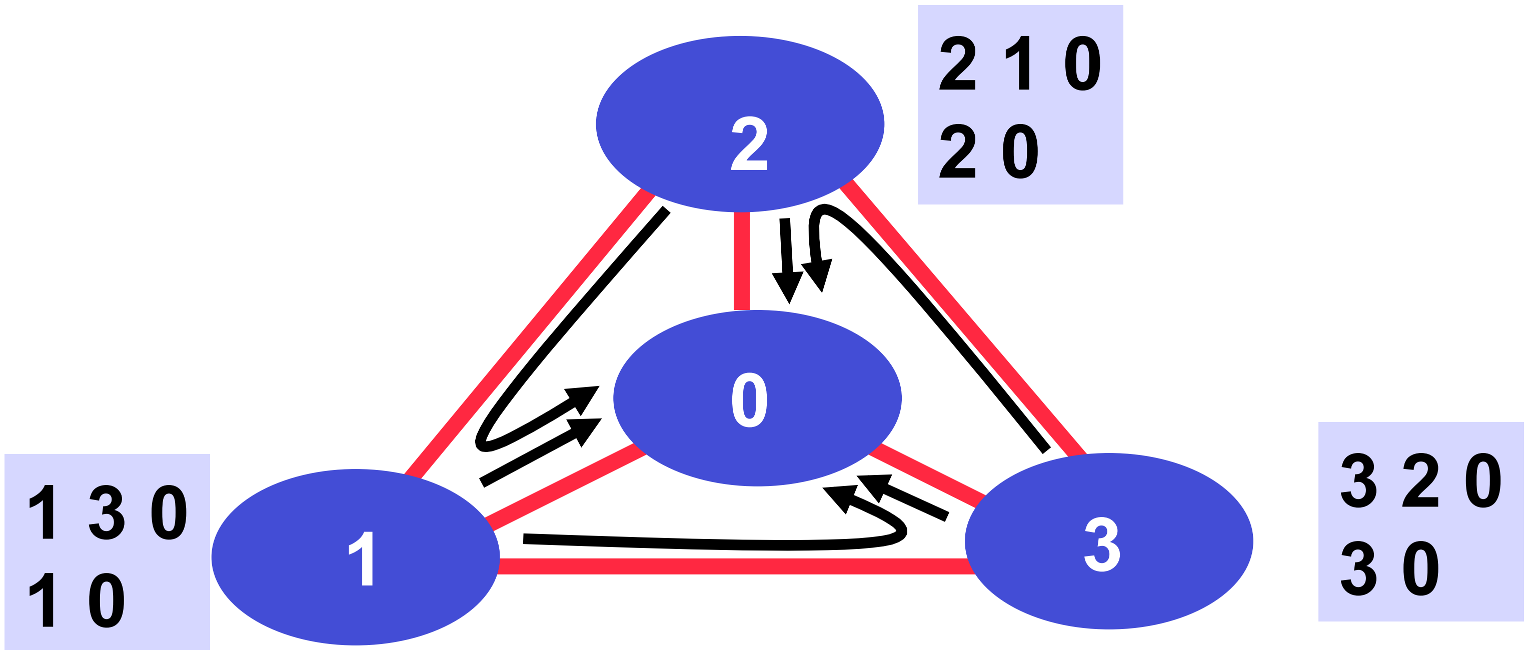
1 2 0
1 0



2 1 0
2 0

Second solution

A SPP May Have No Solution



Ensuring Convergence is Difficult

- Create a global Internet routing registry
  - Difficult to keep up to date
- Require each AS to publish its routing policies
  - Difficult to get them to participate
- Check for conflicting policies, and resolve conflicts
  - Checking is NP-complete
  - Re-checking for each failure scenario

- Sufficient conditions for global convergence
  - Restrictions on the topology and routing policies that can be checked or ensured locally
  - E.g., based on common types of biz relationships
- Game theoretic point of view
  - Stable paths are a dominant strategy equilibrium
  - BGP is a distributed algorithm (iterated elimination of dominated strategies) to seek dominant strategy equilibrium
  - Find sufficient condition for the existence and convergence of dominant strategy equilibrium