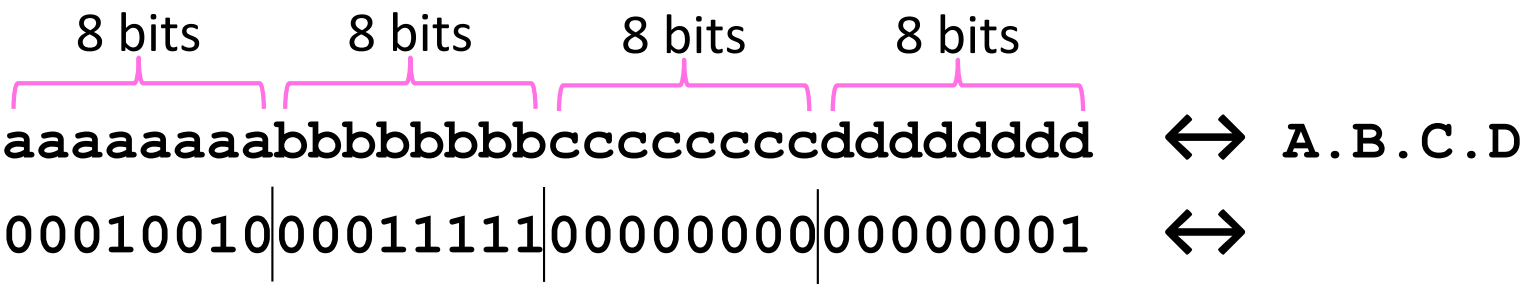
Computer Networks

IP Forwarding

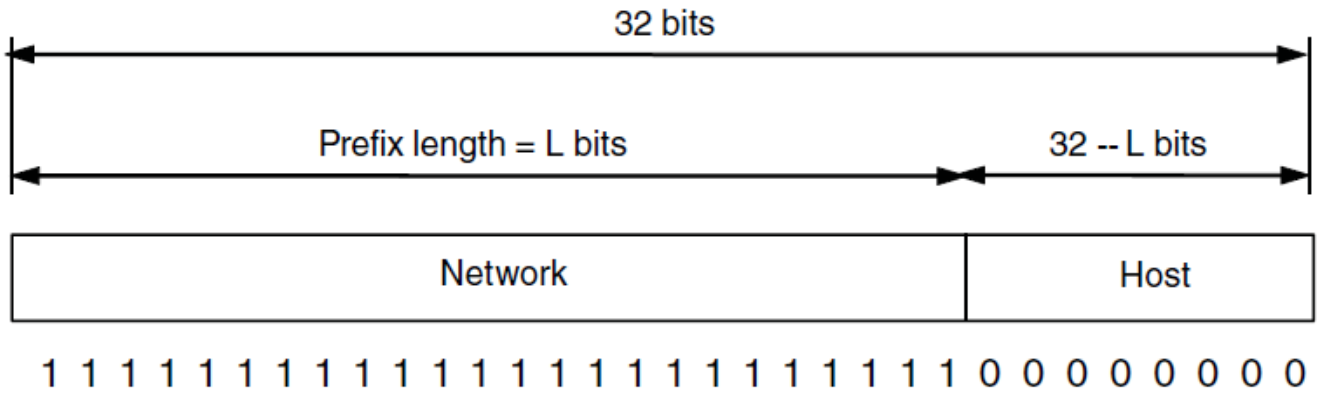
IP Addresses

- IPv4 uses 32-bit addresses
  - IPv6 uses 128-bit addresses
- Written in “dotted quad” notation
  - Four 8-bit numbers separated by dots



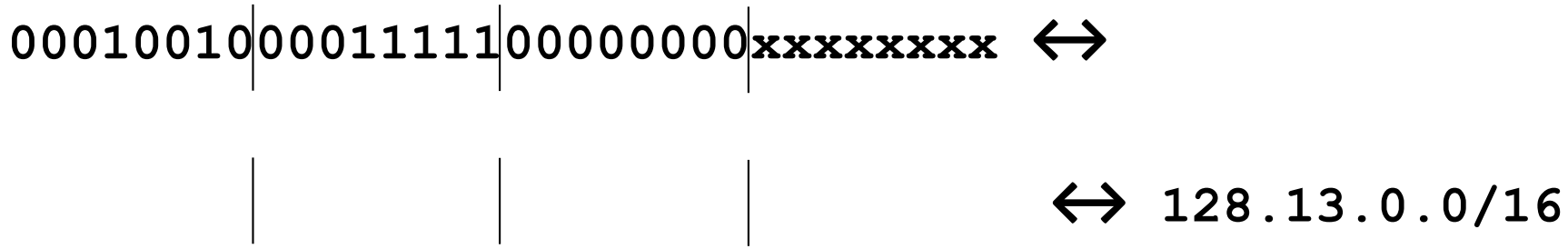
IP Prefixes

- Addresses are allocated in blocks called prefixes
  - Addresses in an L-bit prefix have the same top L bits
  - There are 2^{32-L} addresses aligned on 2^{32-L} boundary



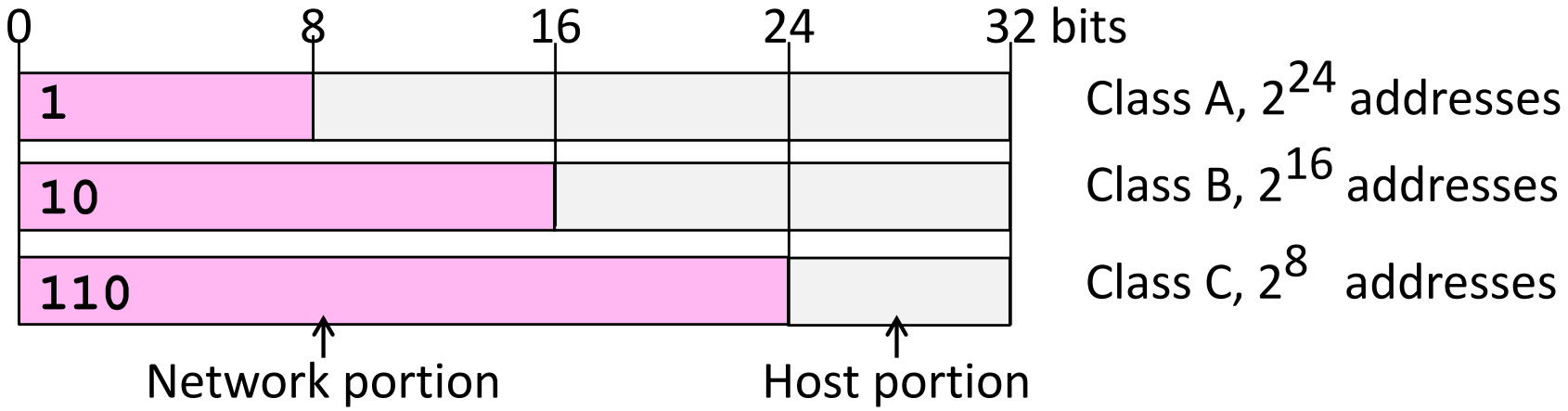
IP Prefixes (2)

- Written in “address/length” notation
  - Address is lowest address in the prefix, length is prefix bits
  - E.g., 128.13.0.0/16 is 128.13.0.0 to 128.13.255.255
  - So a /24 (“slash 24”) is 256 addresses, and a /32 is one address



Classful IP Addressing

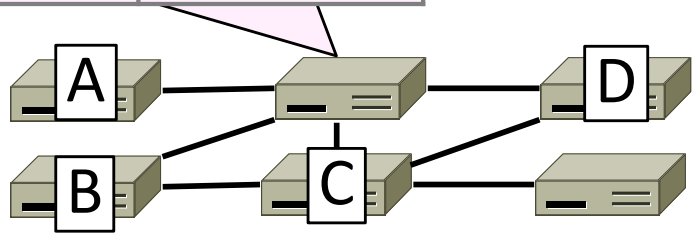
- Originally, IP addresses came in fixed size blocks with the class/size encoded in the high-order bits
  - They still do, but the classes are now ignored



IP Forwarding

- All addresses on one network belong to the same prefix
- Node uses a table that lists the next hop for prefixes

Prefix	Next Hop
192.24.0.0/19	D
192.24.12.0/22	B



Longest Matching Prefix

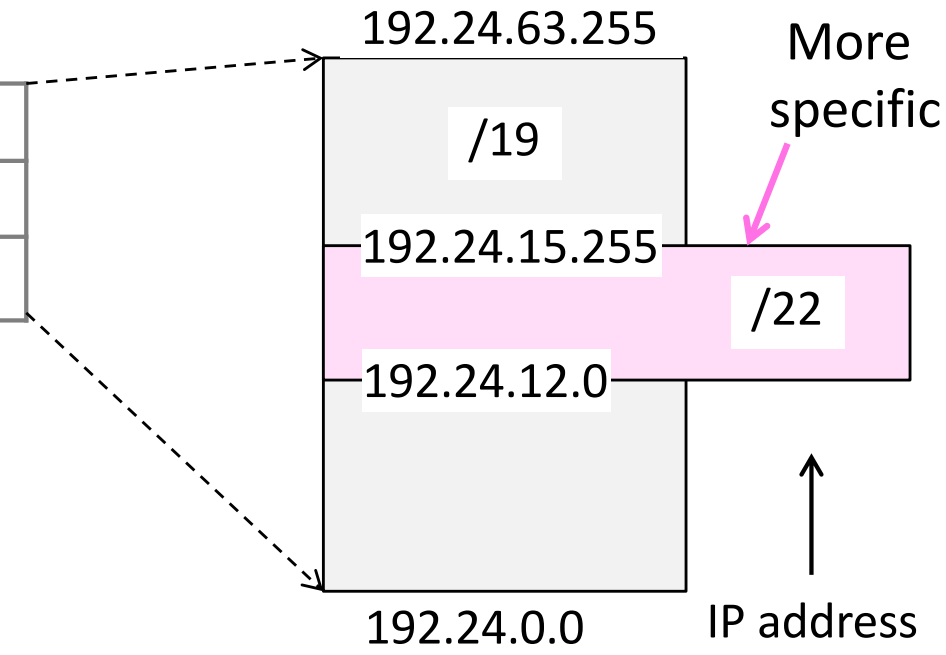
- Prefixes in the table might overlap!
  - Combines hierarchy with flexibility

- Longest matching prefix forwarding rule:
  - For each packet, find the longest prefix that contains the destination address, i.e., the most specific entry
  - Forward the packet to the next hop router for that prefix

Longest Matching Prefix (2)

Prefix	Next Hop
192.24.0.0/19	D
192.24.12.0/22	B

192.24.6.0 →
192.24.14.32 →
192.24.16.32 →



- What are the use cases or implications of longest prefix matching?

Flexibility of Longest Matching Prefix

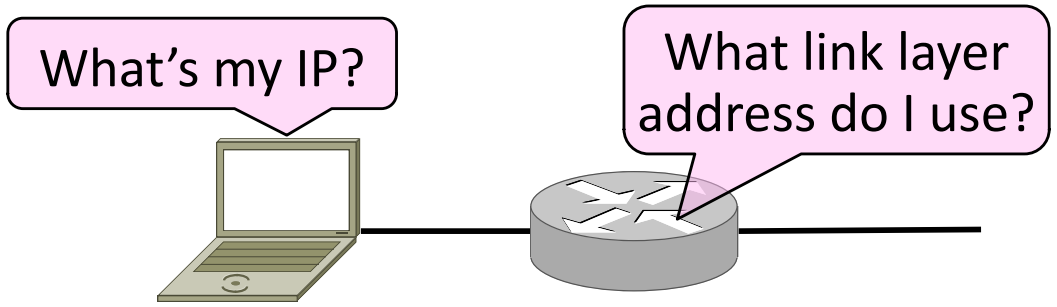
- Can provide default behavior, with less specifics
  - To send traffic going outside an organization to a border router
- Can special case behavior, with more specifics
  - For performance, economics, security, …

Performance of Longest Matching Prefix

- Uses hierarchy for a compact table
  - Relies on use of large prefixes
- Lookup more complex than table
  - Used to be a concern for fast routers
  - Not an issue in practice these days

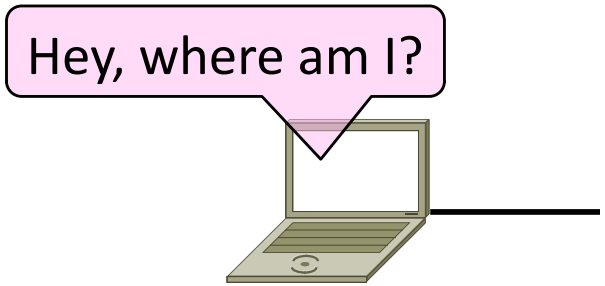
Topic

- Filling in the gaps we need to make for IP forwarding work in practice
  - Getting IP addresses (DHCP) »
  - Mapping IP to link addresses (ARP) »



Getting IP Addresses

- Problem:
  - A node wakes up for the first time …
  - What is its IP address? What's the IP address of its router? Etc.
  - At least Ethernet address is on NIC



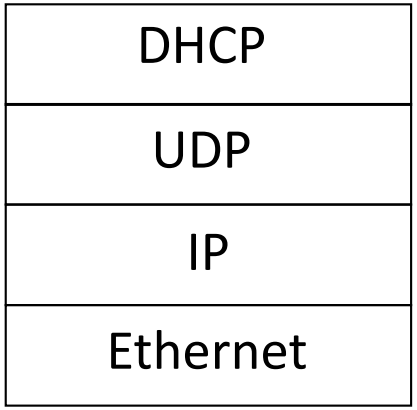
DHCP

- DHCP (Dynamic Host Configuration Protocol), from 1993, widely used

- It leases IP address to nodes
- Provides other parameters too
  - Network prefix
  - Address of local router
  - DNS server, time server, etc.

DHCP Protocol Stack

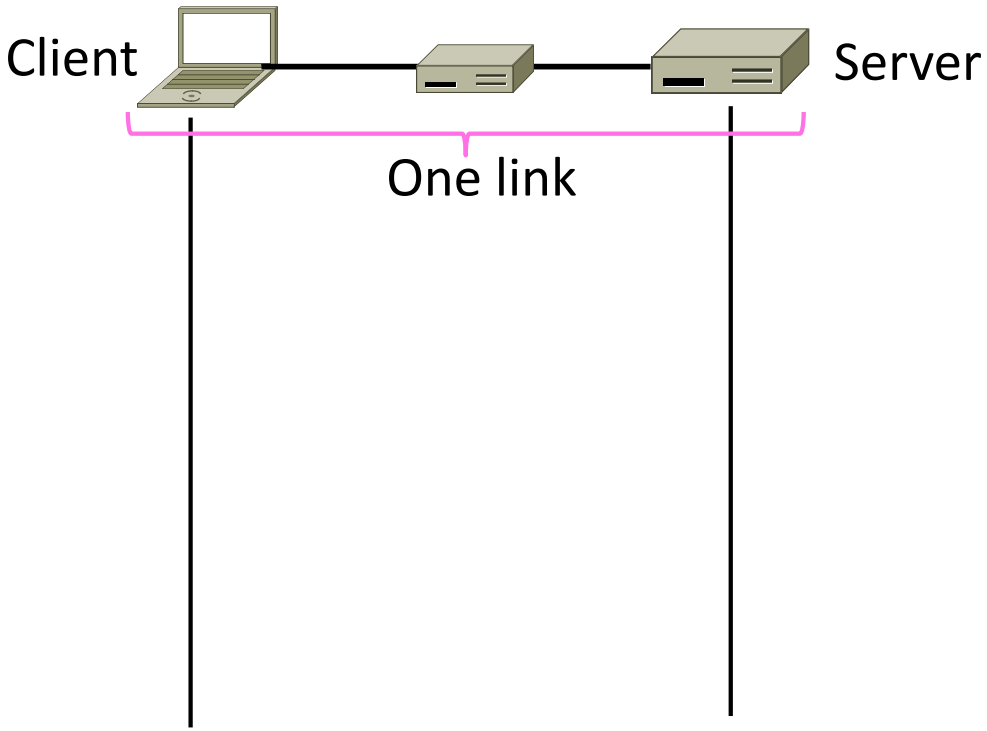
- DHCP is a client-server application
  - Uses UDP ports 67, 68



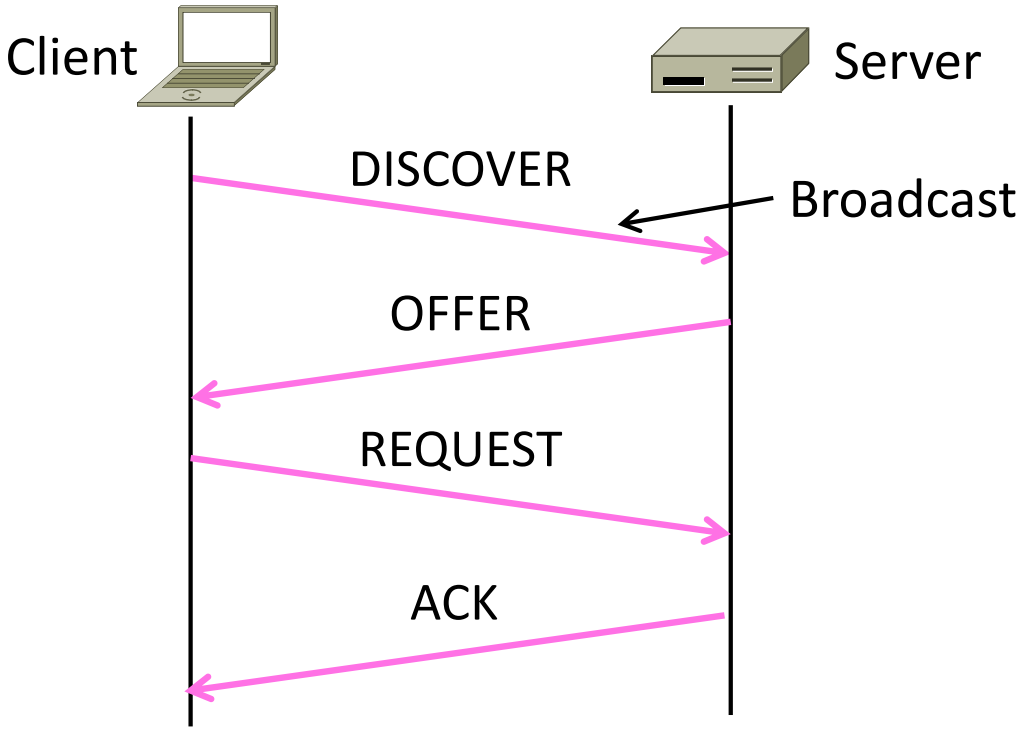
DHCP Addressing

- Bootstrap issue:
  - How does node send a message to DHCP server before it is configured?
- Answer:
  - Node sends broadcast messages that delivered to all nodes on the network
  - Broadcast address is all 1s
  - IP (32 bit): 255.255.255.255
  - Ethernet (48 bit): ff:ff:ff:ff:ff:ff

DHCP Messages



DHCP Messages (2)

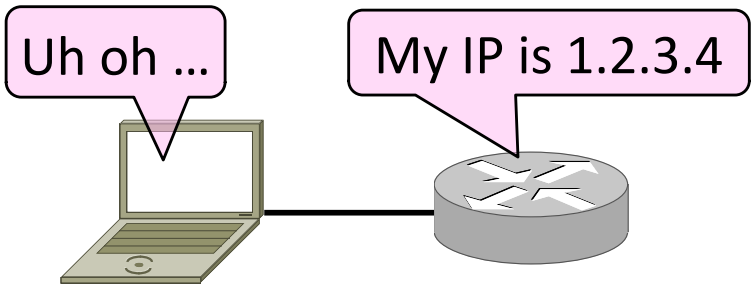


DHCP Messages (3)

- To renew an existing lease, an abbreviated sequence is used:
  - REQUEST, followed by ACK
- Protocol also supports replicated servers for reliability

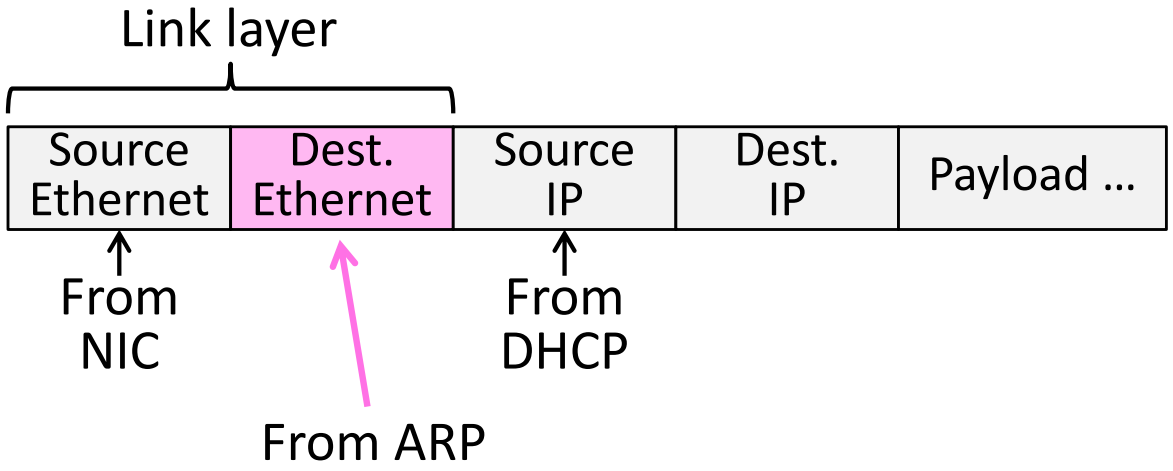
Sending an IP Packet

- Problem:
  - A node needs Link layer addresses to send a frame over the local link
  - How does it get the destination link address from a destination IP address?



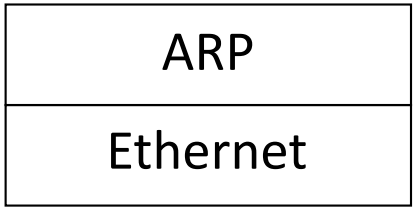
ARP (Address Resolution Protocol)

- Node uses to map a local IP address to its Link layer addresses

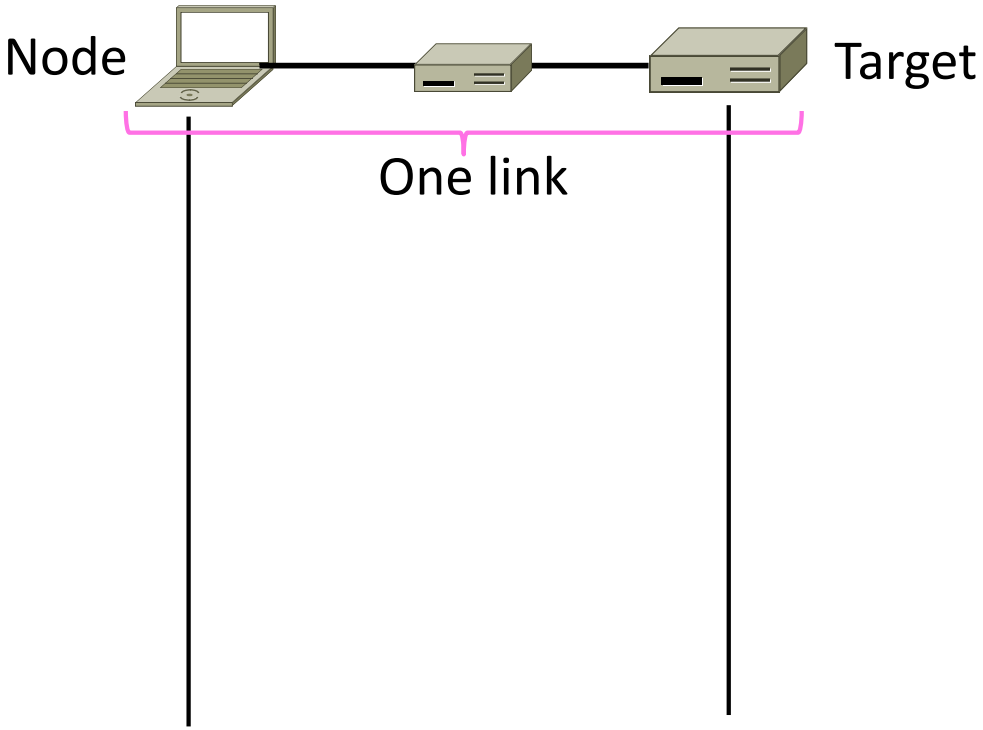


ARP Protocol Stack

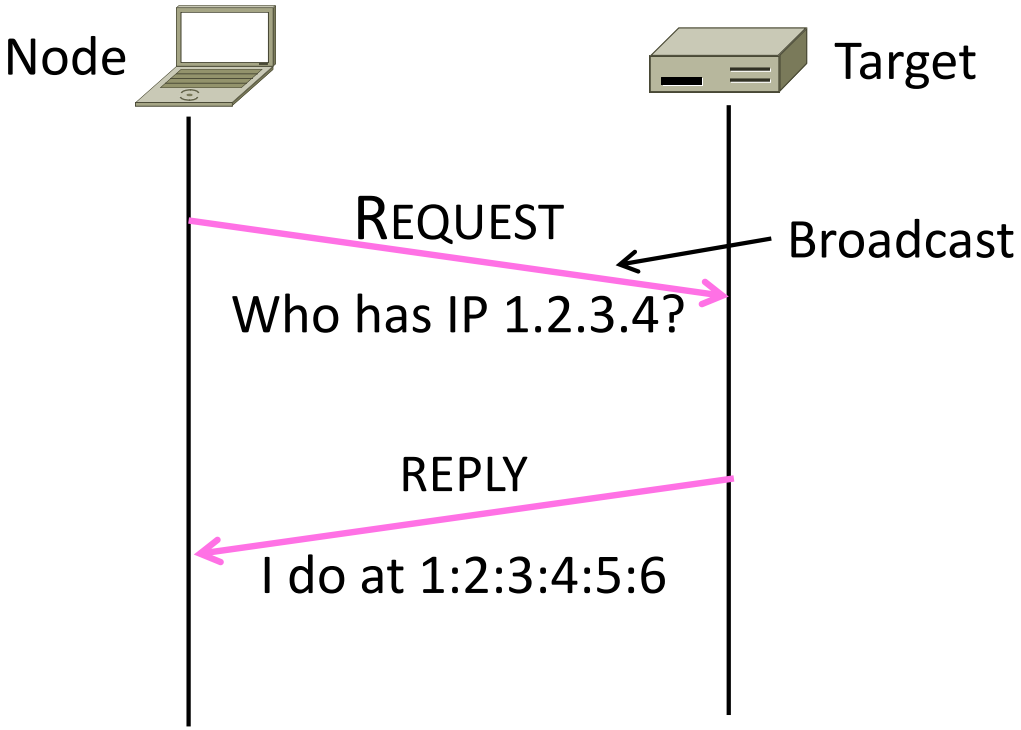
- ARP sits right on top of link layer
  - No servers, just asks node with target IP to identify itself
  - Uses broadcast to reach all nodes



ARP Messages

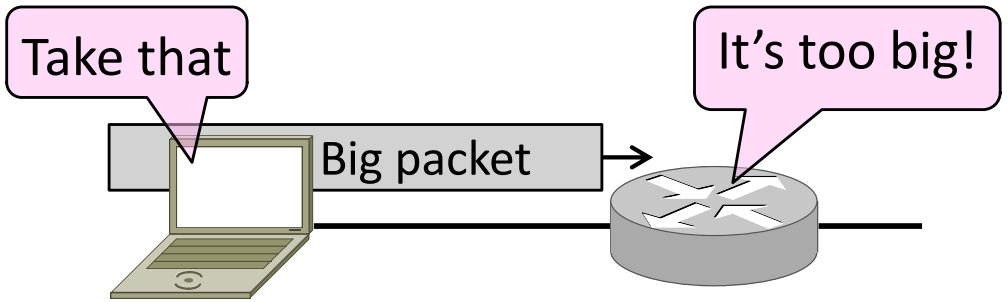


ARP Messages (2)



Topic

- How do we connect networks with different maximum packet sizes?
  - Need to split up packets, or discover the largest size to use



Packet Size Problem

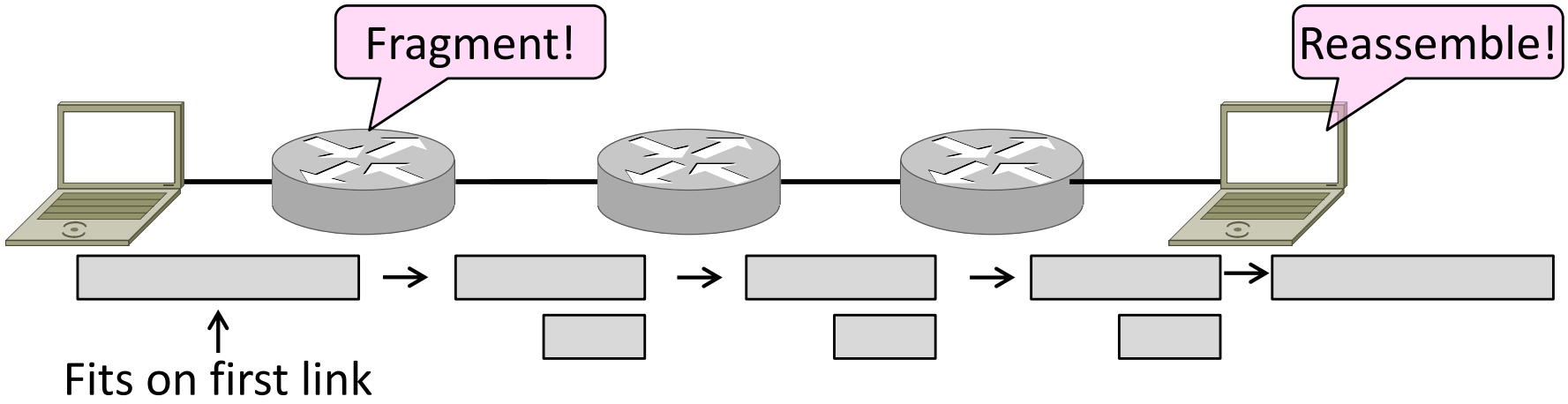
- Different networks have different maximum packet sizes
  - Or MTU (Maximum Transmission Unit)
  - E.g., Ethernet 1.5K, WiFi 2.3K
- Prefer large packets for efficiency
  - But what size is too large?
  - Difficult because node does not know complete network path

Packet Size Solutions

- Fragmentation
  - Split up large packets in the network if they are too big to send
  - Classic method, dated
- Discovery
  - Find the largest packet that fits on the network path and use it
  - IP uses today instead of fragmentation

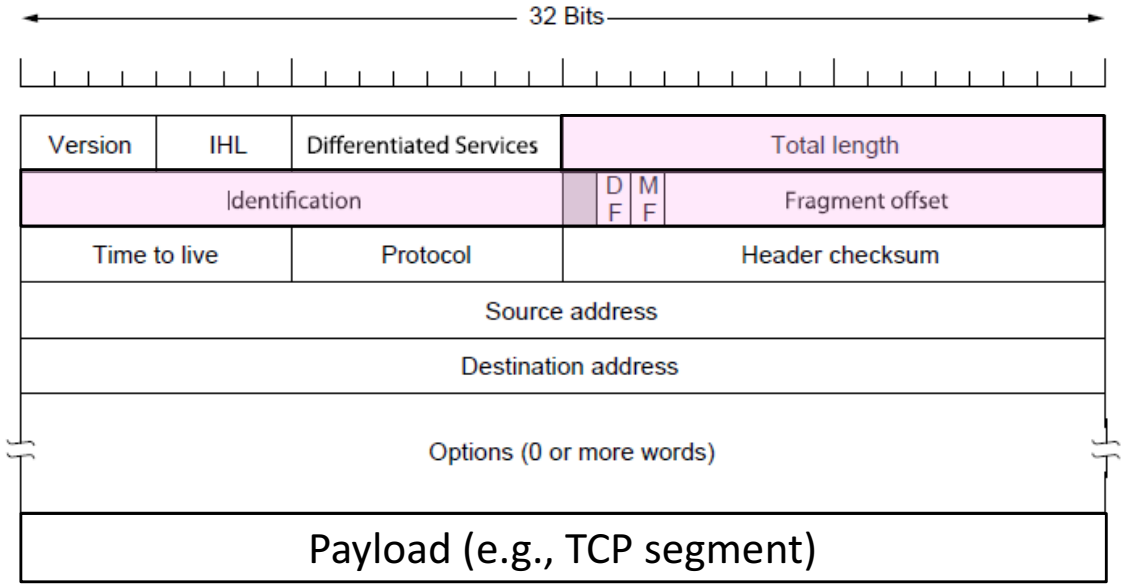
IPv4 Fragmentation

- Routers fragment packets that are too large to forward
- Receiving host reassembles to reduce load on routers



IPv4 Fragmentation Fields

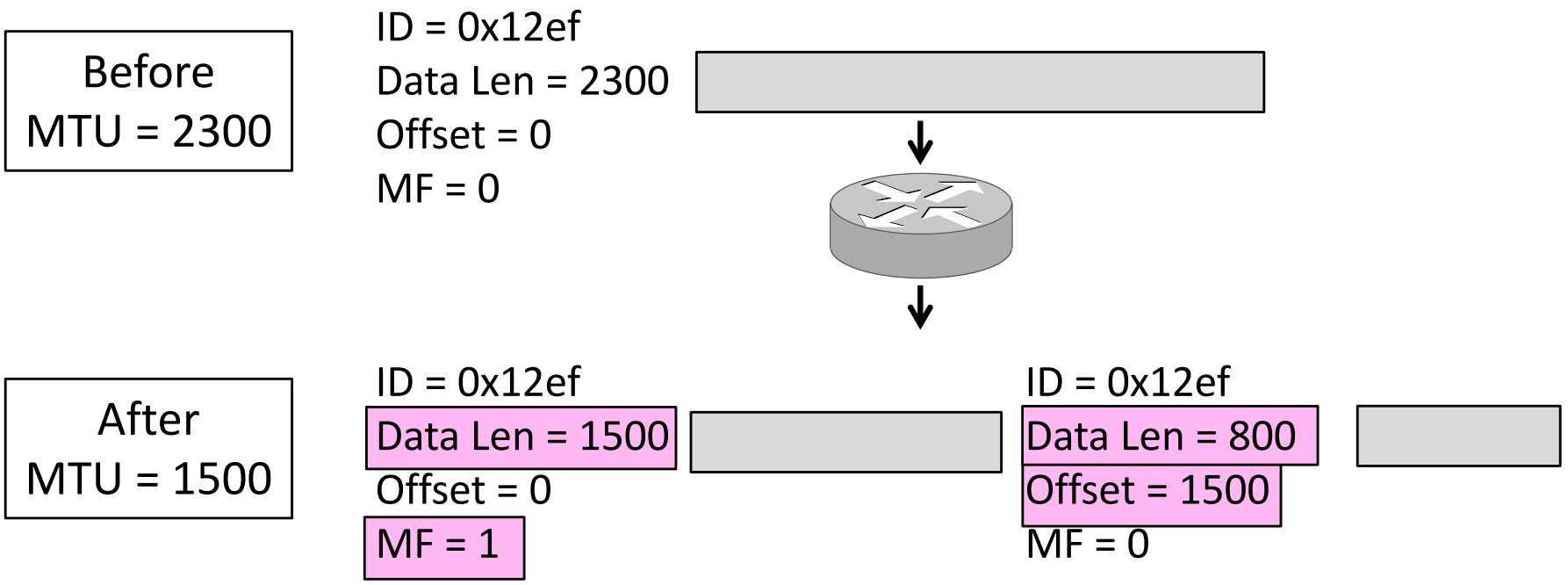
- Header fields used to handle packet size differences
  - Identification, Fragment offset, MF/DF control bits



IPv4 Fragmentation Procedure

- Routers split a packet that is too large:
  - Typically break into large pieces
  - Copy IP header to pieces
  - Adjust length on pieces
  - Set offset to indicate position
  - Set MF (More Fragments) on all pieces except last
- Receiving hosts reassembles the pieces:
  - Identification field links pieces together, MF tells receiver when it has all pieces

IPv4 Fragmentation (3)



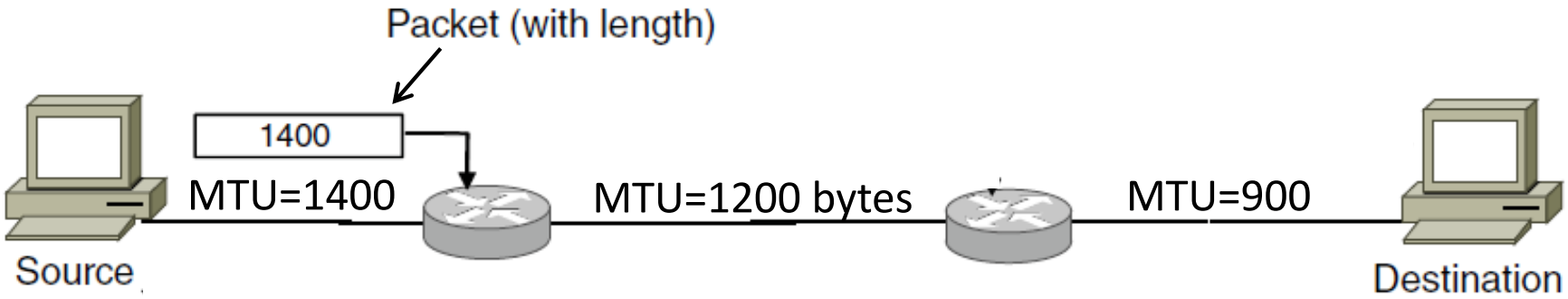
IPv4 Fragmentation (4)

- It works!
  - Allows repeated fragmentation
- But fragmentation is undesirable
  - More work for routers, hosts
  - Tends to magnify loss rate
  - Security vulnerabilities too

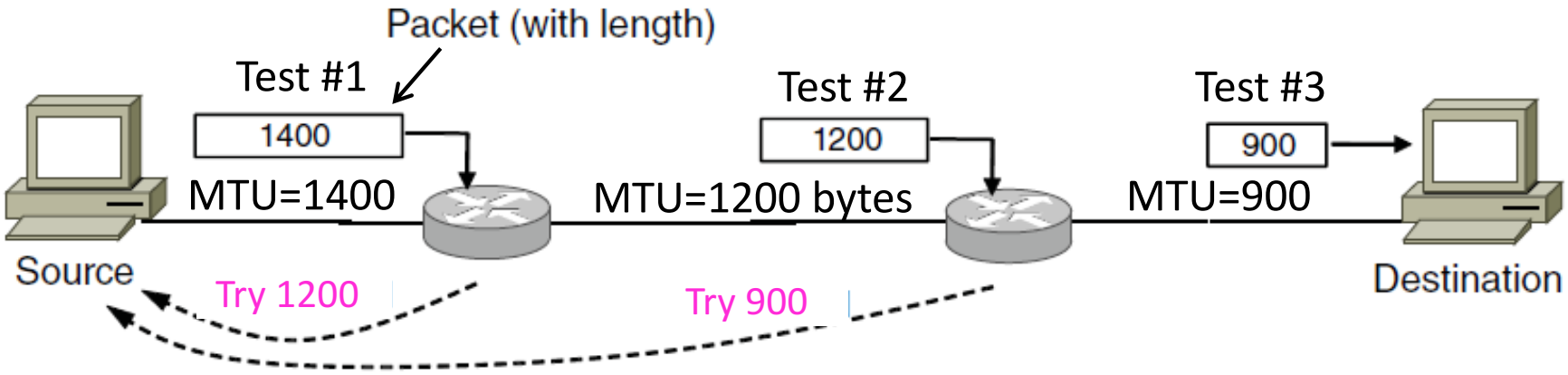
Path MTU Discovery

- Discover the MTU that will fit
  - So we can avoid fragmentation
  - The method in use today
- Host tests path with large packet
  - Routers provide feedback if too large; they tell host what size would have fit

Path MTU Discovery (2)



Path MTU Discovery (3)

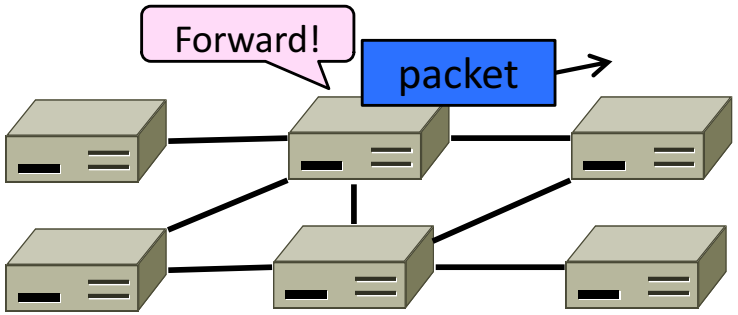


Computer Networks

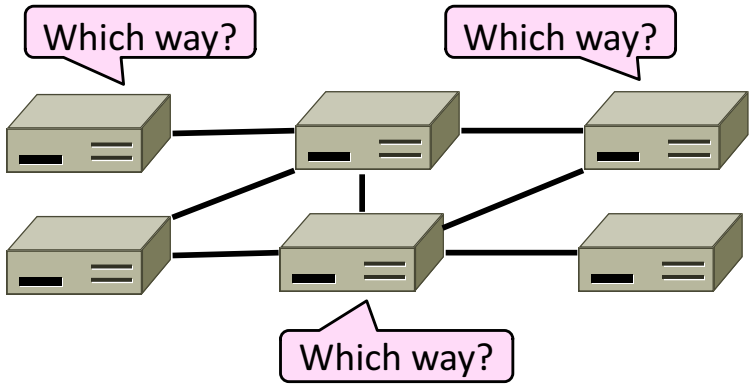
Routing Overview

Routing versus Forwarding

- Forwarding is the process of sending a packet on its way

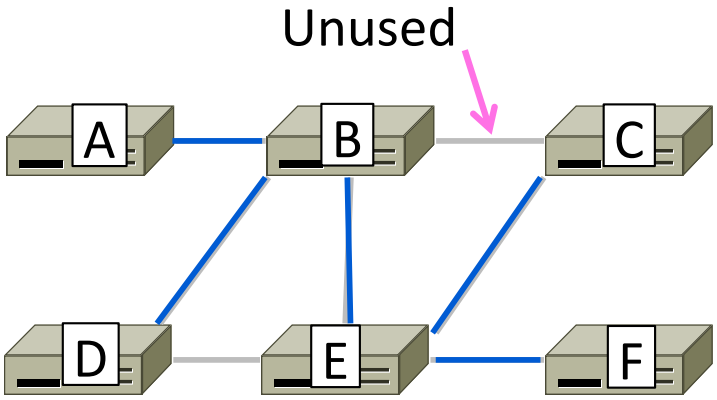


- Routing is the process of deciding in which direction to send traffic

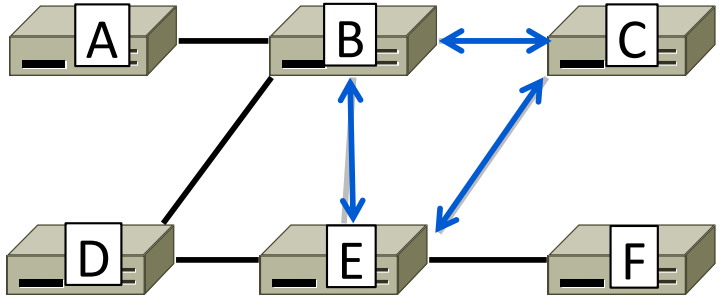


Improving on the Spanning Tree

- Spanning tree provides basic connectivity
  - e.g., some path B→C



- Routing uses all links to find "best" paths
  - e.g., use BC, BE, and CE



Perspective on Bandwidth Allocation

- Routing allocates network bandwidth adapting to failures; other mechanisms used at other timescales

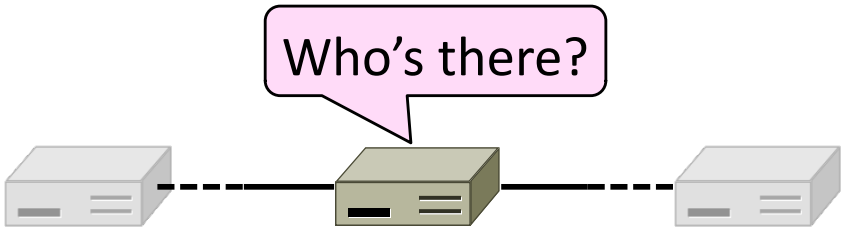
Mechanism	Timescale / Adaptation
Load-sensitive routing	Seconds / Traffic hotspots
Routing	Minutes / Equipment failures
Traffic Engineering	Hours / Network load
Provisioning	Months / Network customers

Goals of Routing Algorithms

- What are the properties we want of any routing scheme?

Rules of Routing Algorithms

- Decentralized, distributed setting
  - All nodes are alike; no controller
  - Nodes only know what they learn by exchanging messages with neighbors
  - Nodes operate concurrently
  - May be node/link/message failures

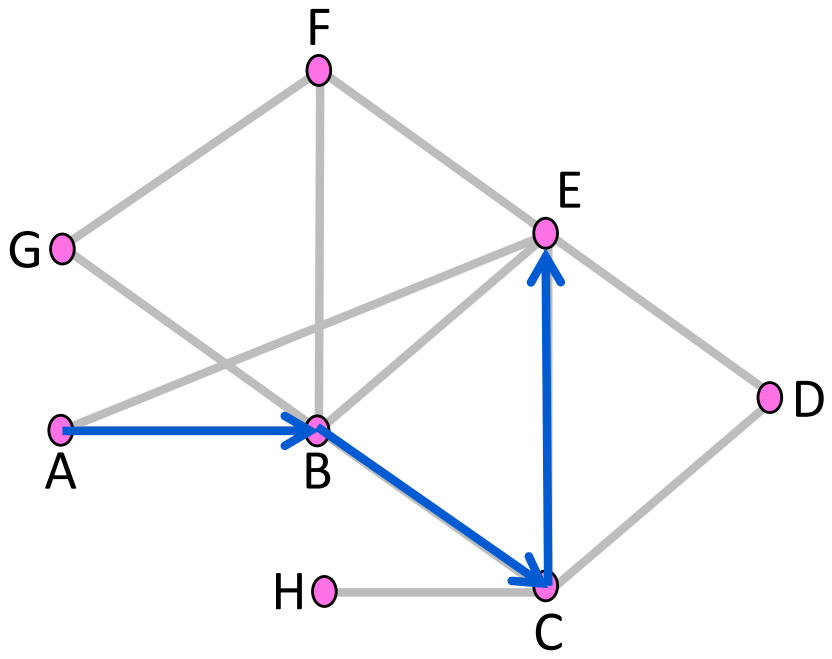


Computer Networks

Shortest Path Routing

What are "Best" paths anyhow?

- Many possibilities:
  - Latency, avoid circuitous paths
  - Bandwidth, avoid slow links
  - Money, avoid expensive links
  - Hops, to reduce switching
- But only consider topology
  - Ignore workload, e.g., hotspots



Shortest Paths

We'll approximate "best" by a cost function that captures the factors

- Often call lowest "shortest"

1. Assign each link a cost (distance)
2. Define best path between each pair of nodes as the path that has the lowest total cost (or is shortest)
3. Pick randomly to break ties

Dijkstra's Algorithm

Algorithm:

- Mark all nodes tentative, set distances from source to 0 (zero) for source, and ∞ (infinity) for all other nodes

- While tentative nodes remain:
  - Extract N, the one with lowest distance
  - Add link to N to the shortest path tree
  - Relax the distances of neighbors of N by lowering any better distance estimates

Dijkstra Comments

- Dynamic programming algorithm; leverages optimality property
- Runtime depends on efficiency of extracting min-cost node
- Gives us complete information on the shortest paths to/from one node
  - But requires complete topology

Distance Vector Routing

Distance Vector Routing

- Simple, early routing approach
  - Used in ARPANET, and “RIP”
- One of two main approaches to routing
  - Distributed version of Bellman-Ford
  - Works, but very slow convergence after some failures
- Link-state algorithms are now typically used in practice
  - More involved, better behavior

Distance Vector Setting

Each node computes its forwarding table in a distributed setting:

1. Nodes know only the cost to their neighbors; not the topology
2. Nodes can talk only to their neighbors using messages
3. All nodes run the same algorithm concurrently
4. Nodes and links may fail, messages may be lost

Distance Vector Algorithm

Each node maintains a vector of distances to all destinations

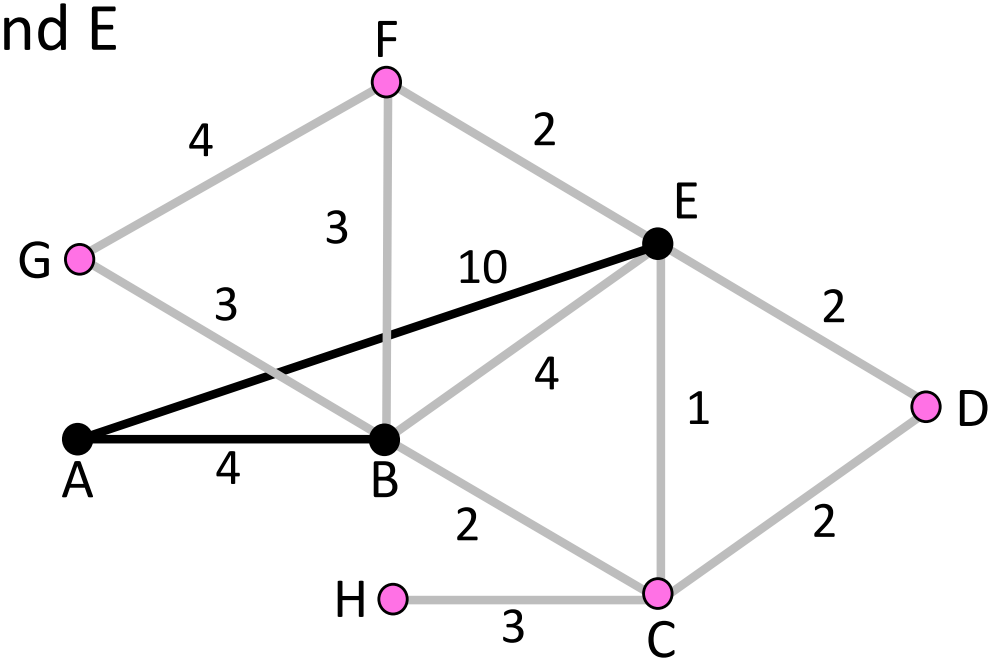
1. Initialize vector with 0 (zero) cost to self, ∞ (infinity) to other destinations
2. Periodically send vector to neighbors
3. Update vector for each destination by selecting the shortest distance heard, after adding cost of neighbor link
   - Use the best neighbor for forwarding

Distance Vector (2)

- Consider from the point of view of node A
  - Can only talk to nodes B and E

Initial vector →

To	Cost
A	0
B	∞
C	∞
D	∞
E	∞
F	∞
G	∞
H	∞



Distance Vector (3)

- First exchange with B, E; learn best 1-hop routes

To	B says	E says
A	∞	∞
B	0	∞
C	∞	∞
D	∞	∞
E	∞	0
F	∞	∞
G	∞	∞
H	∞	∞

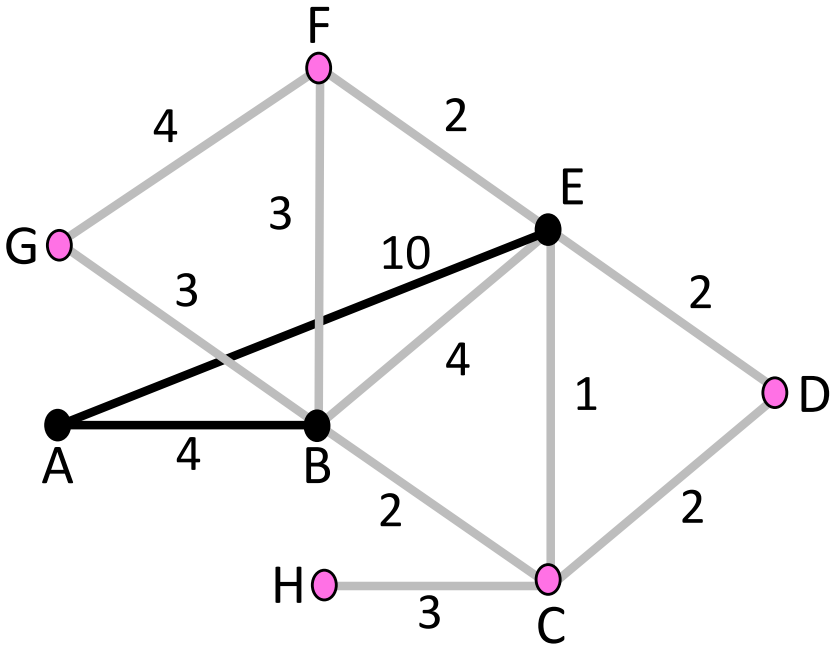
→

B +4	E +10
∞	∞
4	∞
∞	∞
∞	∞
∞	10
∞	∞
∞	∞
∞	∞

→

A's Cost	A's Next
0	--
4	B
∞	--
∞	--
10	E
∞	--
∞	--
∞	--

Learned better route



Distance Vector (4)

- Second exchange; learn best 2-hop routes

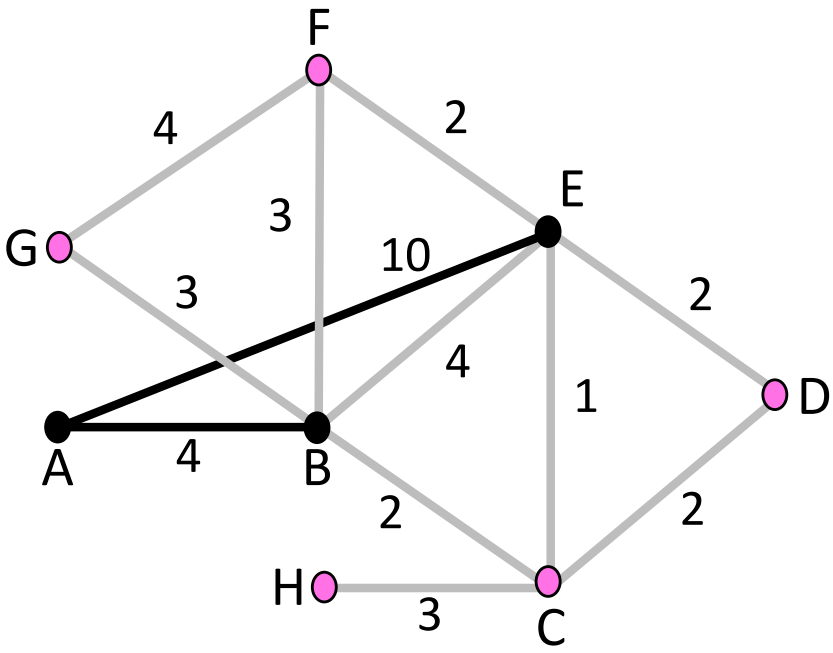
To	B says	E says
A	4	10
B	0	4
C	2	1
D	∞	2
E	4	0
F	3	2
G	3	∞
H	∞	∞

→

B +4	E +10
8	20
4	14
6	11
∞	12
8	10
7	12
7	∞
∞	∞

→

A's Cost	A's Next
0	--
4	B
6	B
12	E
8	B
7	B
7	B
∞	--



Distance Vector (4)

- Third exchange; learn best 3-hop routes

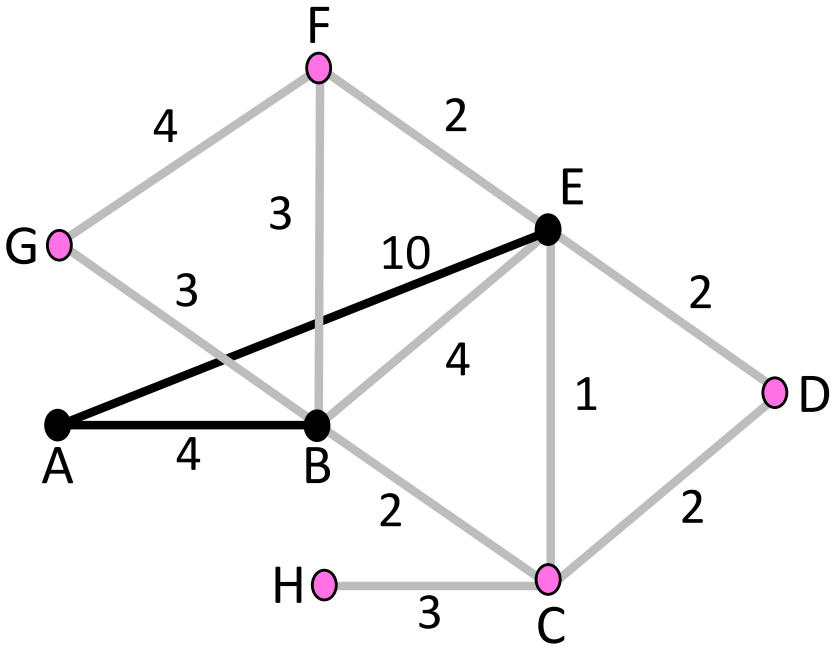
To	B says	E says
A	4	8
B	0	3
C	2	1
D	4	2
E	3	0
F	3	2
G	3	6
H	5	4

→

B +4	E +10
8	18
4	13
6	11
8	12
7	10
7	12
7	16
9	14

→

A's Cost	A's Next
0	--
4	B
6	B
8	B
7	B
7	B
7	B
9	B



Distance Vector (5)

- Subsequent exchanges; converged

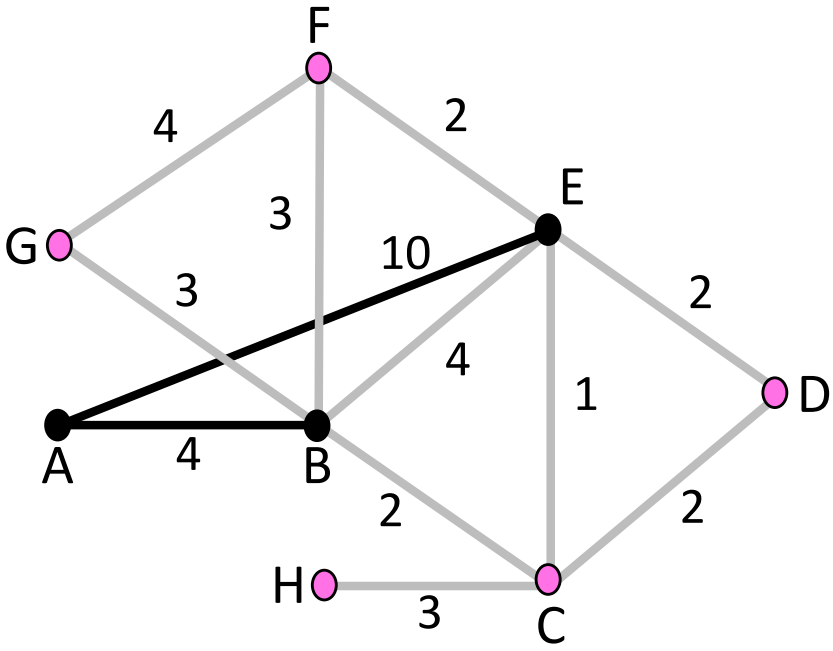
To	B says	E says
A	4	7
B	0	3
C	2	1
D	4	2
E	3	0
F	3	2
G	3	6
H	5	4

→

B +4	E +10
8	17
4	13
6	11
8	12
7	10
7	12
7	16
9	14

→

A's Cost	A's Next
0	--
4	B
6	B
8	B
8	B
7	B
7	B
9	B



Computer Networks

Link State Routing

Link-State Algorithm

Proceeds in two phases:

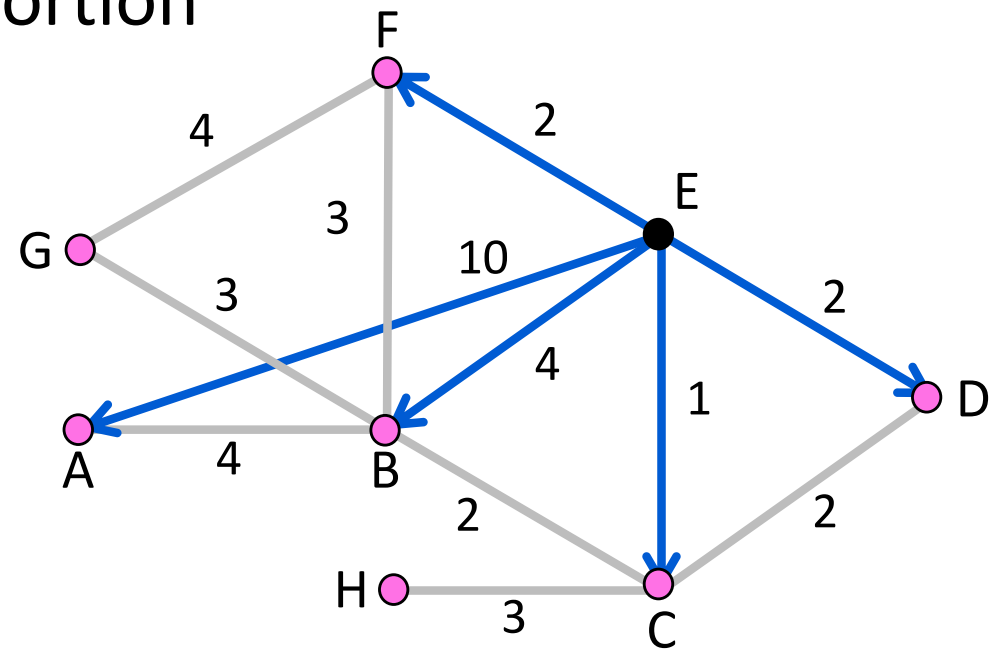
1. Nodes flood topology in the form of link state packets
   - Each node learns full topology
2. Each node computes its own forwarding table
   - By running Dijkstra (or equivalent)

Topology Dissemination

- Each node floods link state packet (LSP) that describes their portion of the topology

Node E's LSP flooded to A, B, C, D, and F

Seq. #	
A	10
B	4
C	1
D	2
F	2



Handling Changes

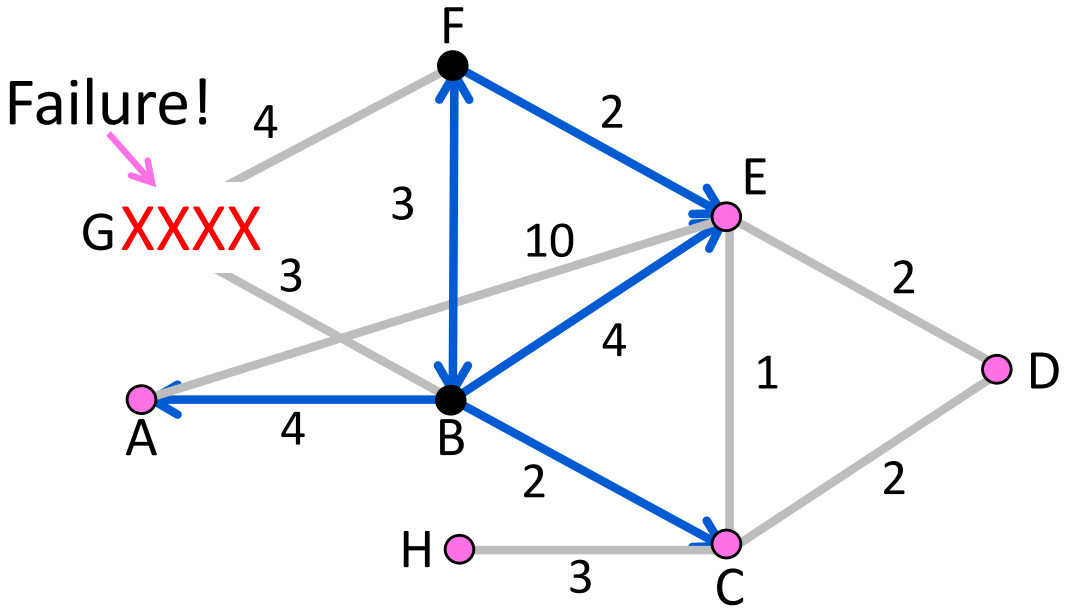
- Nodes adjacent to failed link or node will notice
  - Flood updated LSP with less connectivity

B's LSP

Seq. #	
A	4
C	2
E	4
F	3
~~G~~	~~3~~

F's LSP

Seq. #	
B	3
E	2
~~G~~	~~4~~



Handling Changes (2)

- Link failure
  - Both nodes notice, send updated LSPs
  - Link is removed from topology
- Node failure
  - All neighbors notice a link has failed
  - Failed node can’t update its own LSP
  - But it is OK: all links to node removed

Handling Changes (3)

- Addition of a link or node
  - Add LSP of new node to topology
  - Old LSPs are updated with new link
- Additions are the easy case …

DV/LS Comparison

- How do the two compare?
- What can go wrong?

Link-State Complications

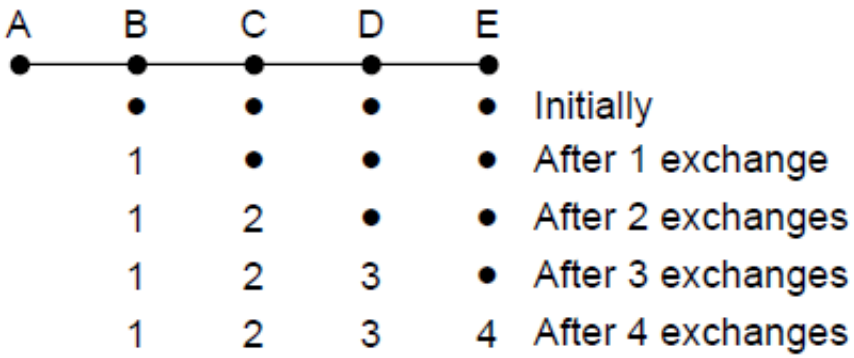
- Things that can go wrong:
  - Corrupt seq. number, or hits max.
  - Node crashes and loses seq. number
  - Network partitions then heals
- Strategy:
  - Include age on LSPs and forget old information that is not refreshed
- Much of the complexity is due to handling corner cases (as usual!)

Distance Vector Dynamics

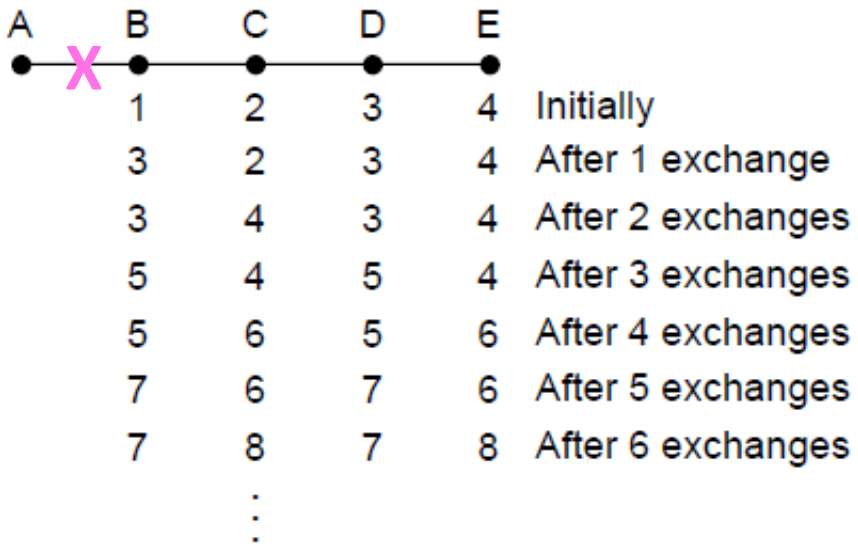
- Adding routes:
  - News travels one hop per exchange
- Removing routes
  - When a node fails, no more exchanges, other nodes forget
- But partitions (unreachable nodes in divided network) are a problem
  - “Count to infinity” scenario

Dynamics (2)

- Good news travels quickly, bad news slowly (inferred)



Desired convergence



"Count to infinity" scenario

Dynamics (3)

- Various heuristics to address
  - e.g.,“Split horizon, poison reverse” (Don’t send route back to where you learned it from.)
- But none are very effective
  - Link state now favored in practice
  - Except when very resource-limited

Computer Networks

Equal-Cost Multi-Path Routing

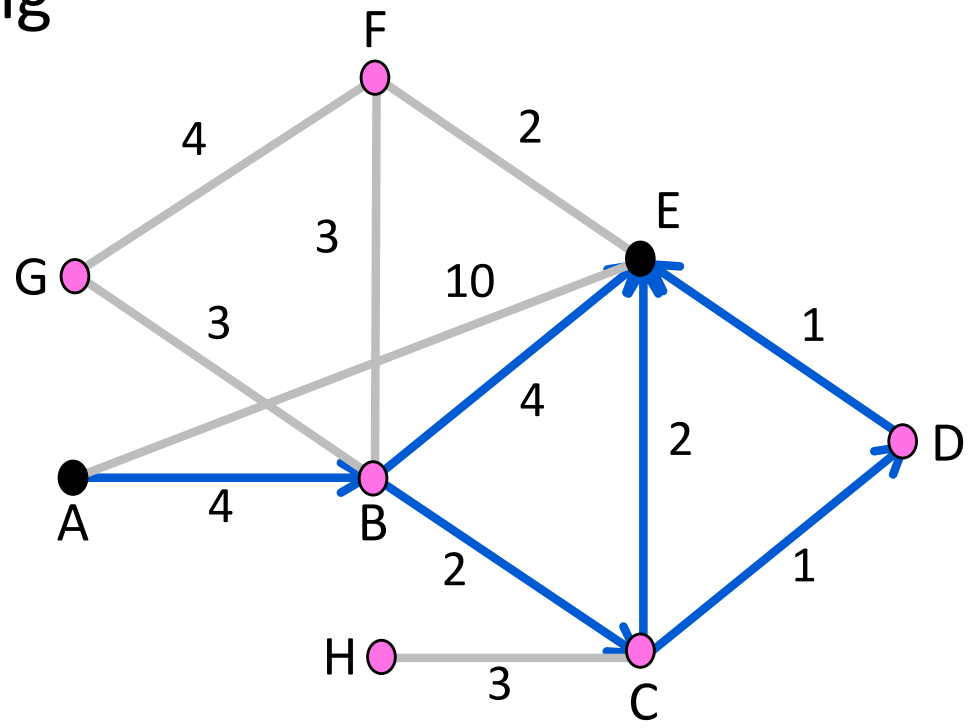
Multipath Routing

- Allow multiple routing paths from node to destination be used at once
  - Topology has them for redundancy
  - Using them can improve performance
- Questions:
  - How do we find multiple paths?
  - How do we send traffic along them?

Equal-Cost Multipath Routes

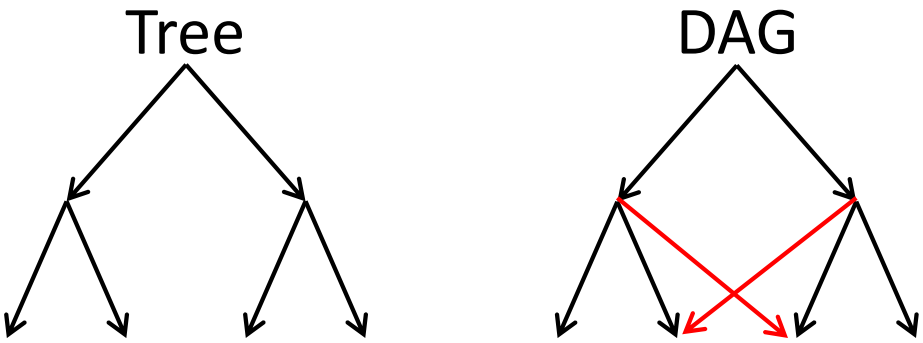
- One form of multipath routing
- Extends shortest path model
  - Keep set if there are ties

- Consider A→E
  - ABE = 4 + 4 = 8
  - ABCE = 4 + 2 + 2 = 8
  - ABCDE = 4 + 2 + 1 + 1 = 8
  - Use them all!



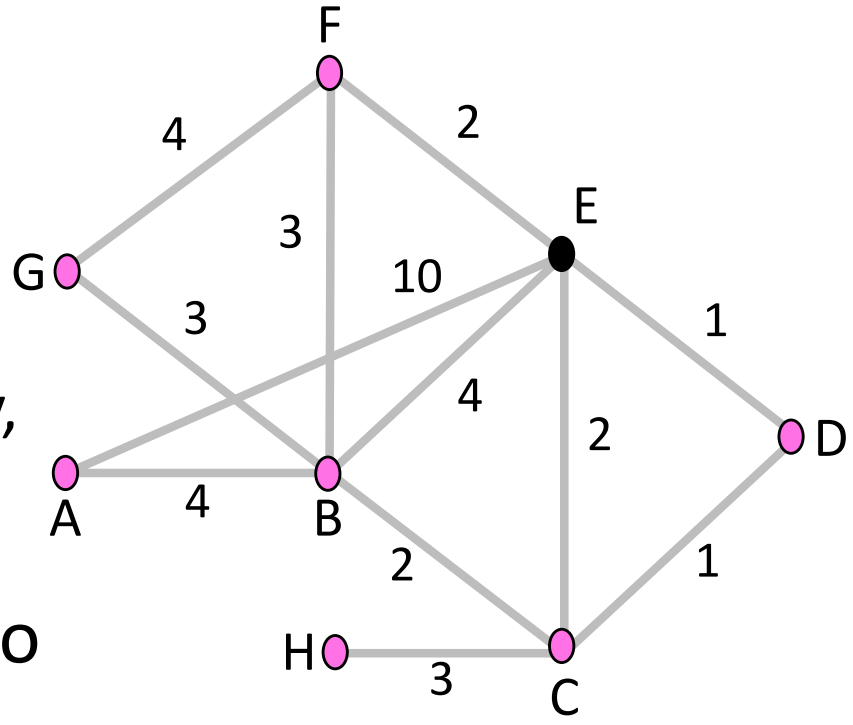
Source “Trees”

- With ECMP, source/sink “tree” is a directed acyclic graph (DAG)
  - Each node has set of next hops
  - Still a compact representation

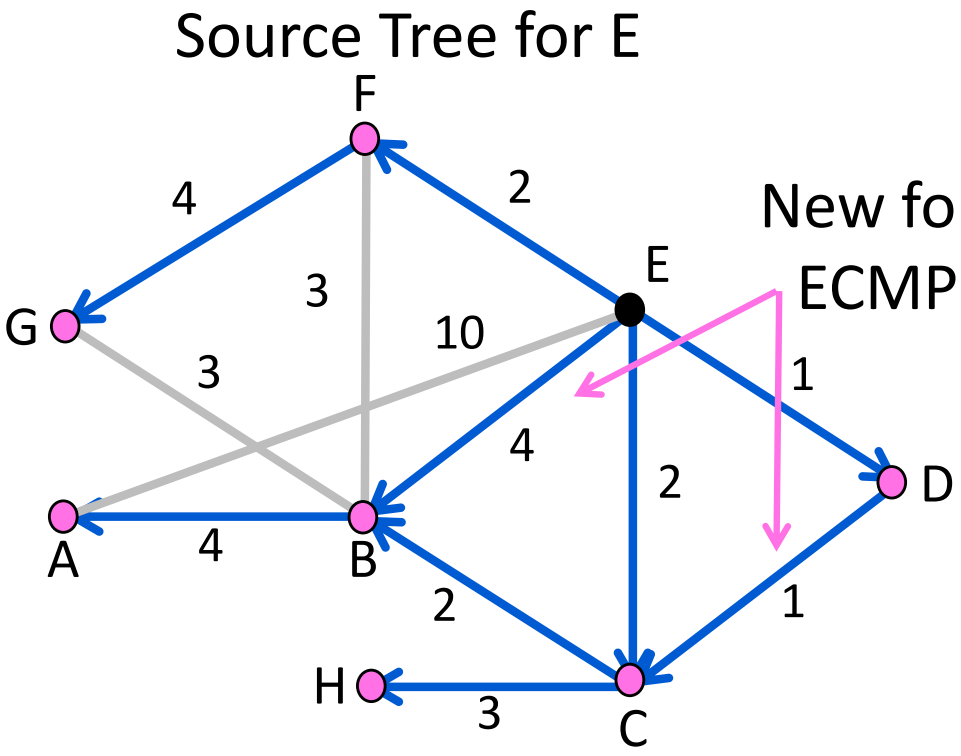


Source "Trees" (2)

- Find the source "tree" for E
  - Procedure is Dijkstra, simply remember set of next hops
  - Compile forwarding table similarly, may have set of next hops
- Straightforward to extend DV too
  - Just remember set of neighbors



Source "Trees" (3)



E's Forwarding Table

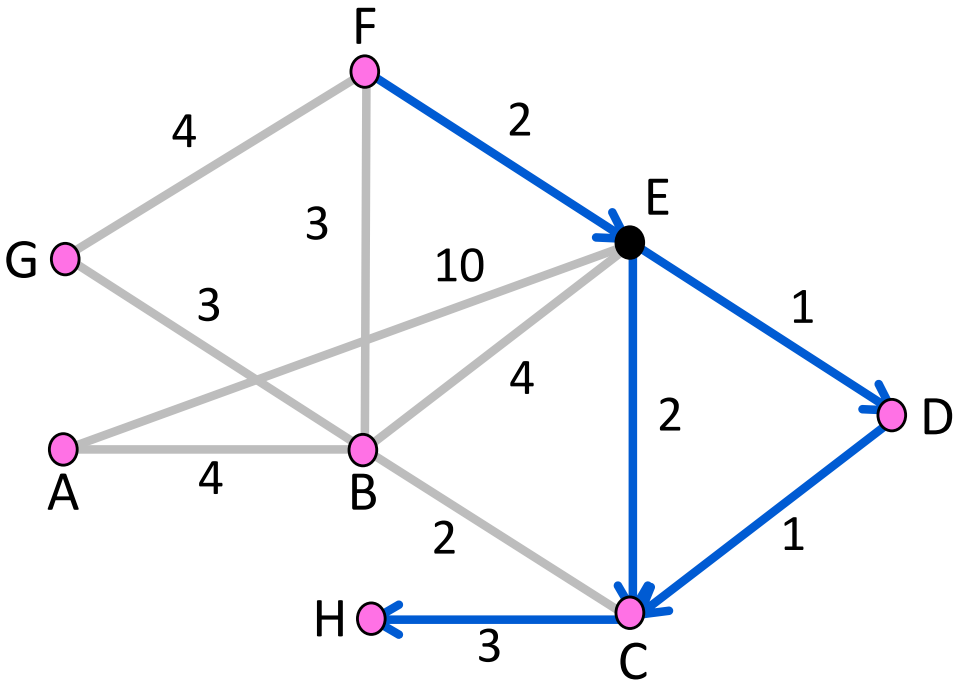
Node	Next hops
A	B, C, D
B	B, C, D
C	C, D
D	D
E	--
F	F
G	F
H	C, D

ECMP Forwarding

- Could randomly pick a next hop for each packet based on destination
  - Balances load, but adds jitter
- Instead, try to send packets from a given source/destination (and ports) on the same path
  - Source/destination pair is called a flow
  - Hash flow identifier to next hop
  - No jitter within flow, but less balanced

ECMP Forwarding (2)

Multipath routes from F to H



E's Forwarding Choices

Flow	Possible next hops	Example choice
F → H	C, D	D
F → C	C, D	D
E → H	C, D	C
E → C	C, D	C

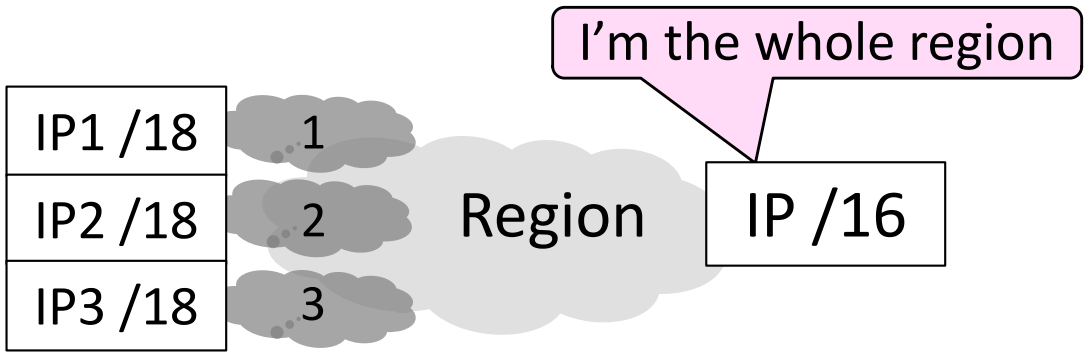
Use both paths to get to one destination

Computer Networks

IP Prefix Aggregation and
Subnets

Prefixes and Hierarchy

- IP prefixes already help to scale routing, but we can go further
  - We can use a less specific (larger) IP prefix as a name for a region

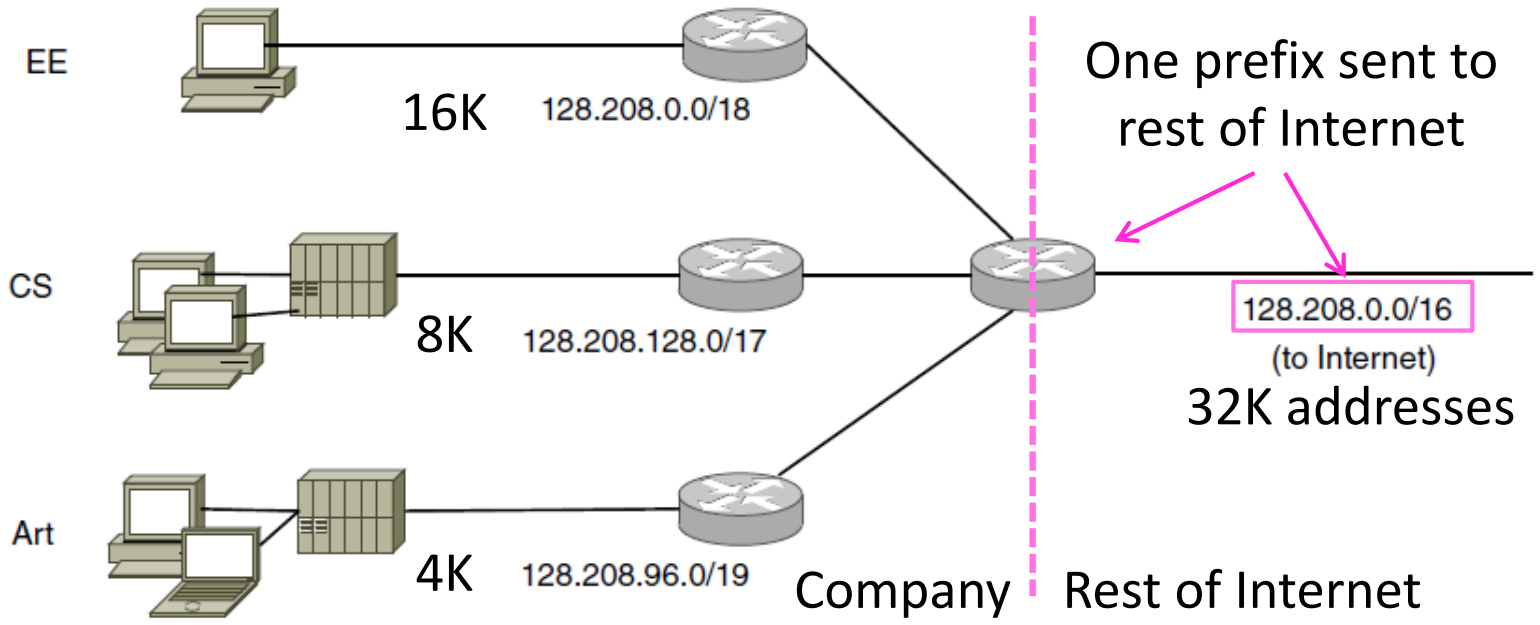


Subnets and Aggregation

1. Subnets
   - Internally split one large prefix into multiple smaller ones
2. Aggregation
   - Externally join multiple smaller prefixes into one large prefix

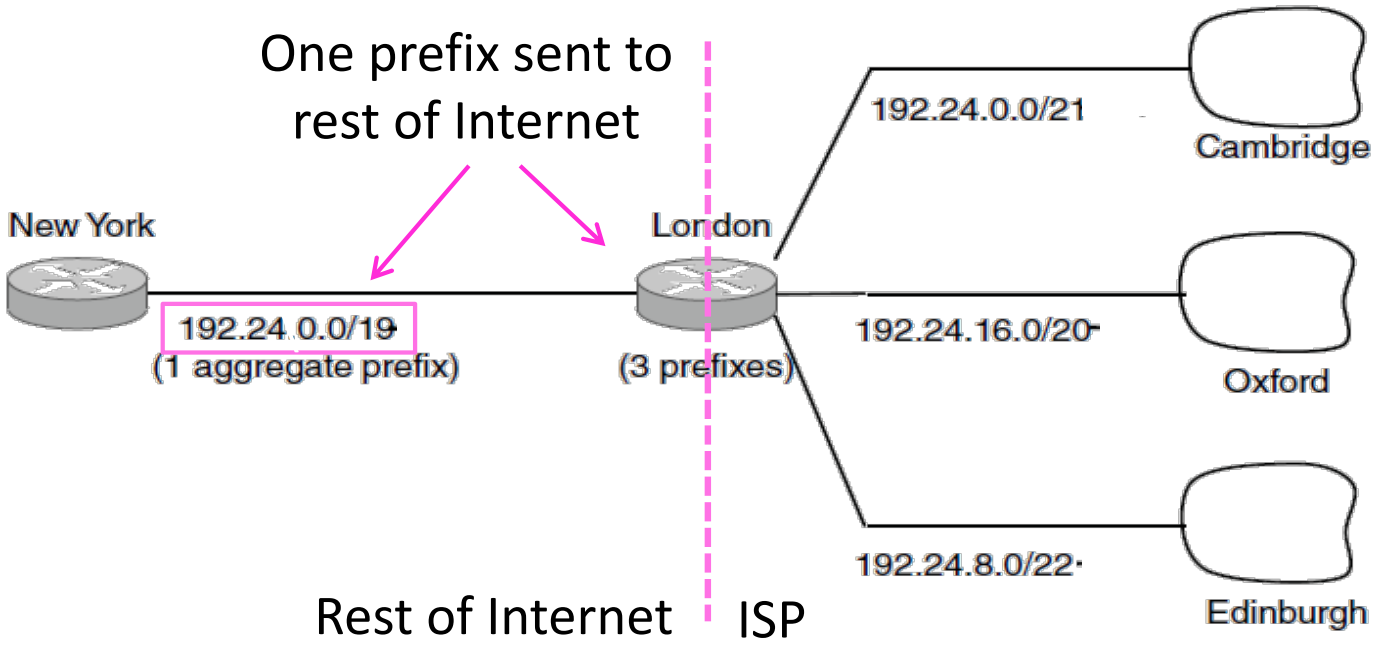
Subnets

- Internally split up one IP prefix



Aggregation

- Externally join multiple separate IP prefixes



Computer Networks

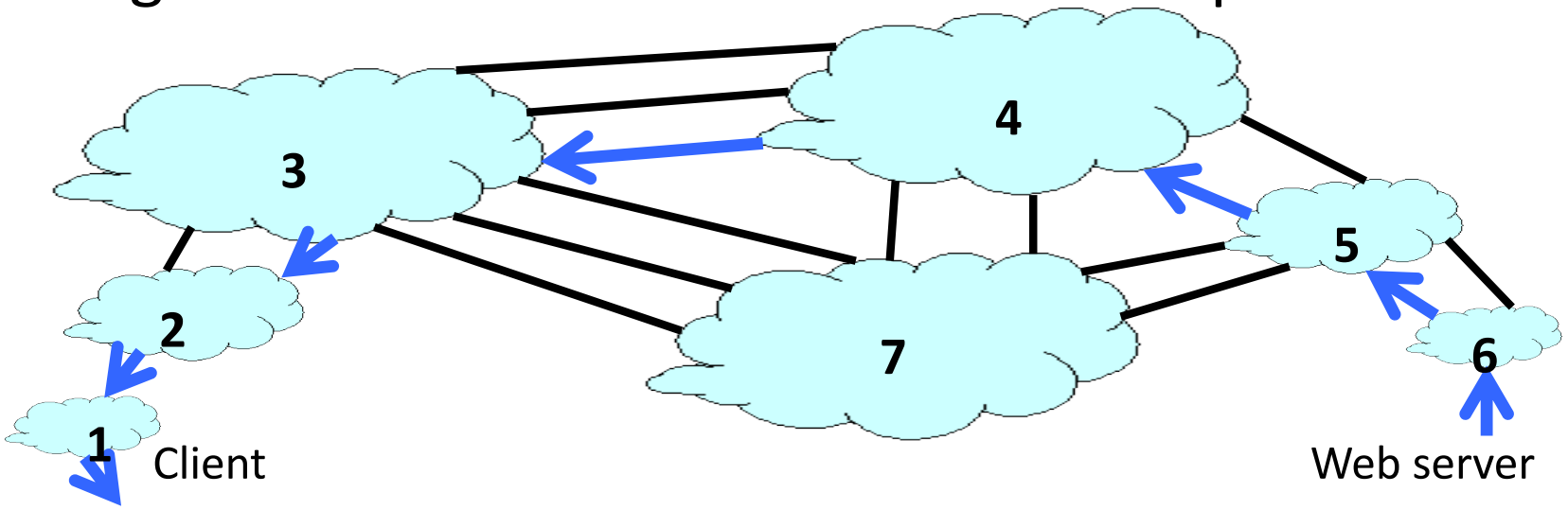
Routing with Policy (BGP)

Outline

- Interdomain routing
  - Autonomous Systems (ASes)
- Path-vector routing
  - Flexible path selection
- Business relationships
  - Customer-provider and peer-peer
  - Hierarchy from tier-1 ASes to stubs
- Border Gateway Protocol (BGP)
  - Announcements and withdrawals
  - Import and export policies

Interdomain Routing: Between Networks

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



AS Numbers (ASNs)

ASNs are 16 bit values.

Currently around 30,000 in use.

- Level 3: 1
- Harvard: 11
- AT&T: 7018, 6341, 5074, …
- UUNET: 701, 702, 284, 12199, …
- Sprint: 1239, 1240, 6211, 6242, …
- …

ASNs represent units of routing policy

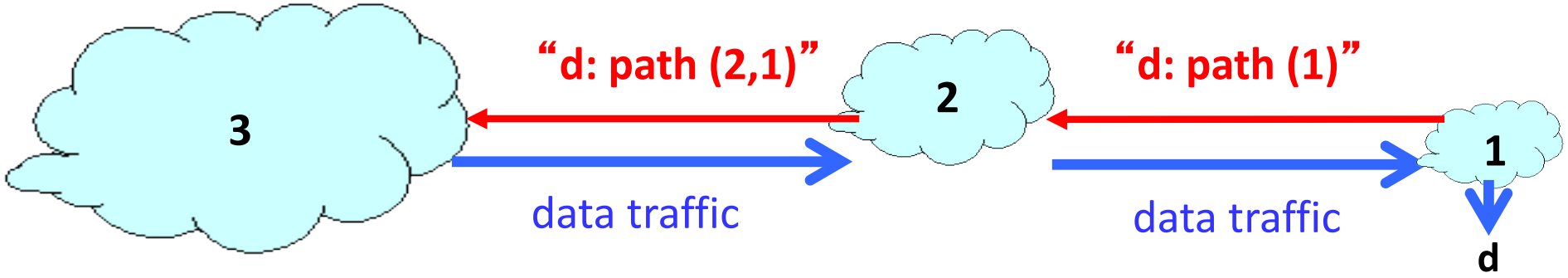
Challenges for Interdomain Routing

- Scale
  - Prefixes: 500,000, and growing
  - ASes: 30,000, and growing
- Privacy
  - ASes don't want to divulge internal topologies
  - … or their business relationships with neighbors
- Policy
  - Need control over where you send traffic
  - … and who can send traffic through you

Policy-Based Path-Vector Routing

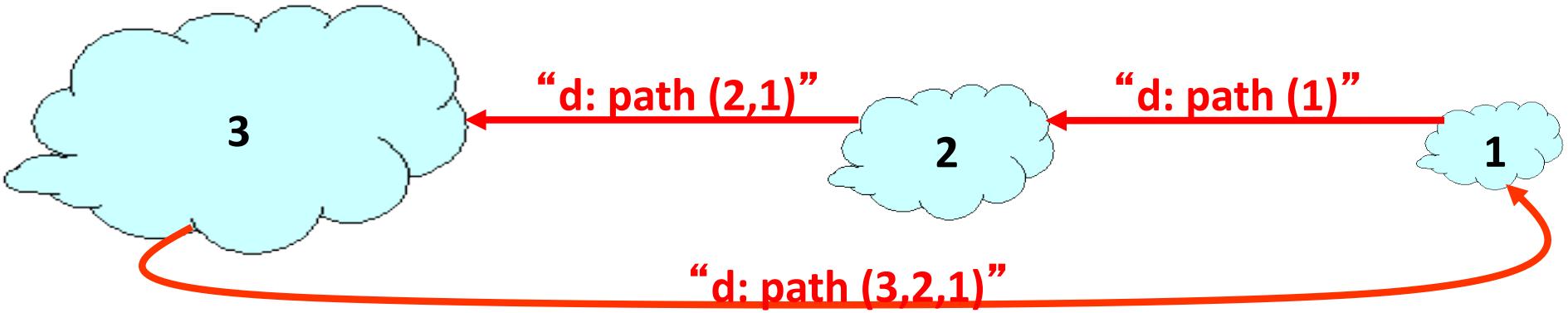
Path-Vector Routing

- 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



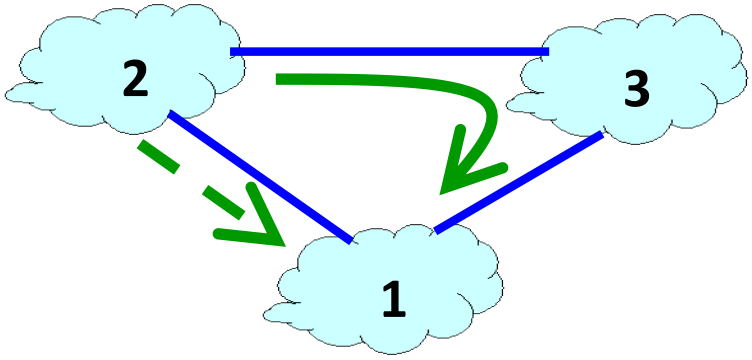
Faster Loop Detection

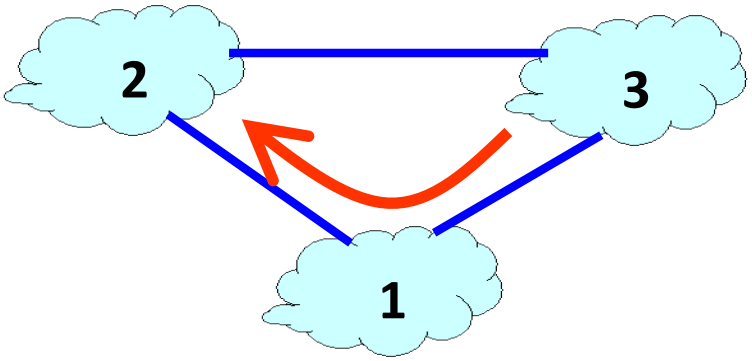
- Node can easily detect a loop
  - Look for its own node identifier in the path
  - E.g., node 1 sees itself in the path "3, 2, 1"
- Node can simply discard paths with loops
  - E.g., node 1 simply discards the advertisement



Flexible Policies

- Each node can apply local policies
  - Path selection: Which path to use?
  - Path export: Whether to advertise the path?
- 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"





Business Relationships

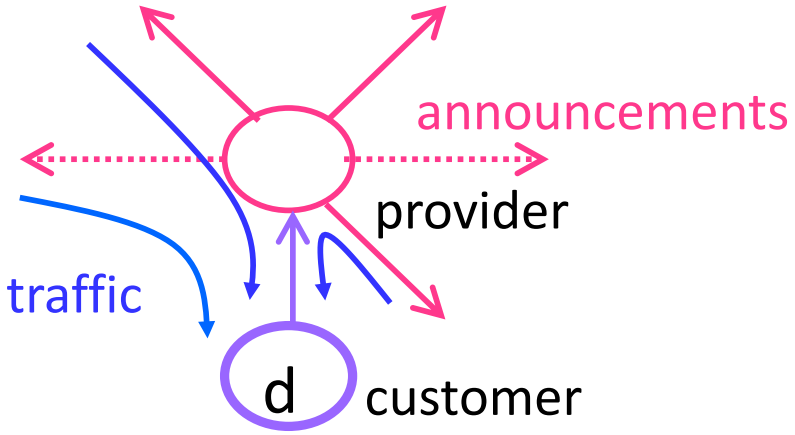
Business Relationships

- 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., MIT is a customer of Level3
  - Peer-peer
    - E.g., UUNET is a peer of Sprint

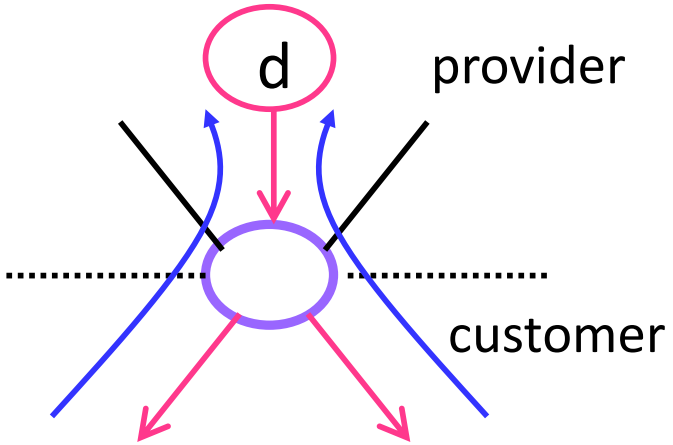
Customer-Provider Relationship

- 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

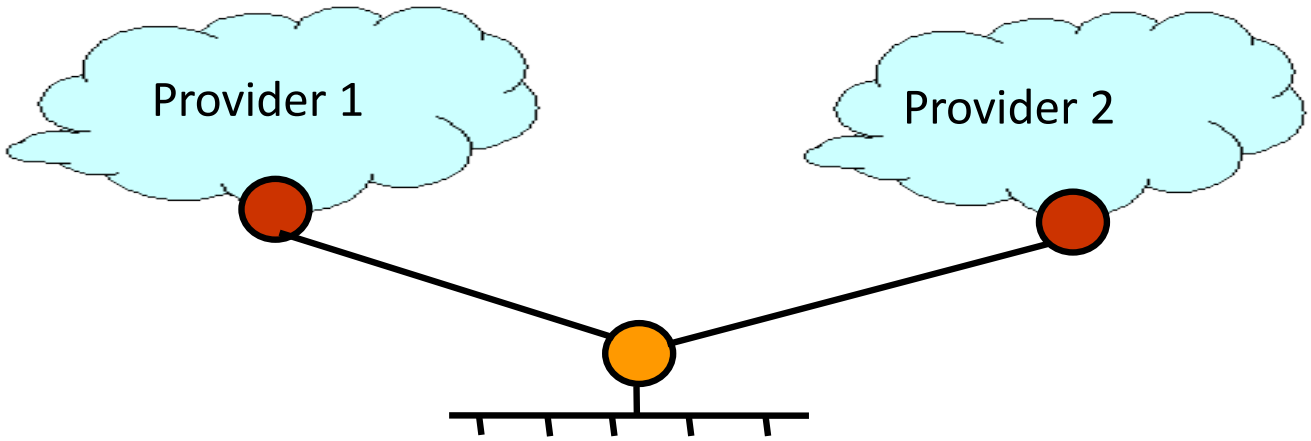


Traffic **from** the customer



Multi-Homing: Two or More Providers

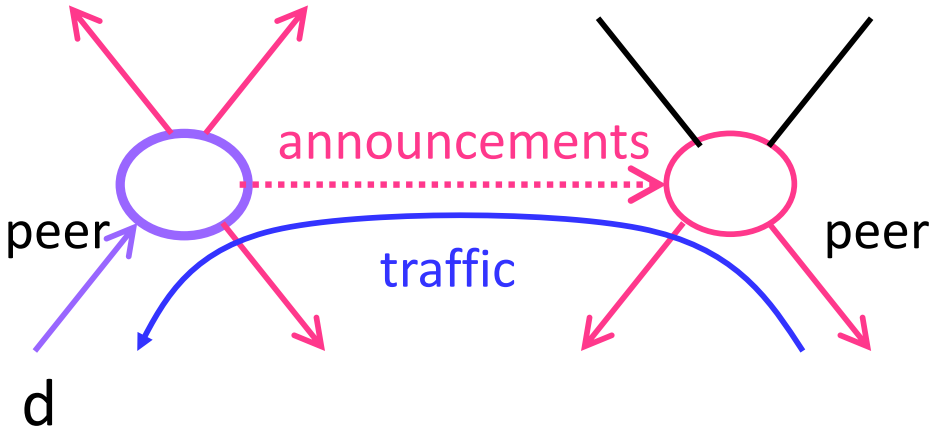
- Motivations for multi-homing
  - Extra reliability, survive single ISP failure
  - Financial leverage through competition
  - Better performance by selecting better path
  - Gaming the 95th-percentile billing model



Peer-Peer Relationship

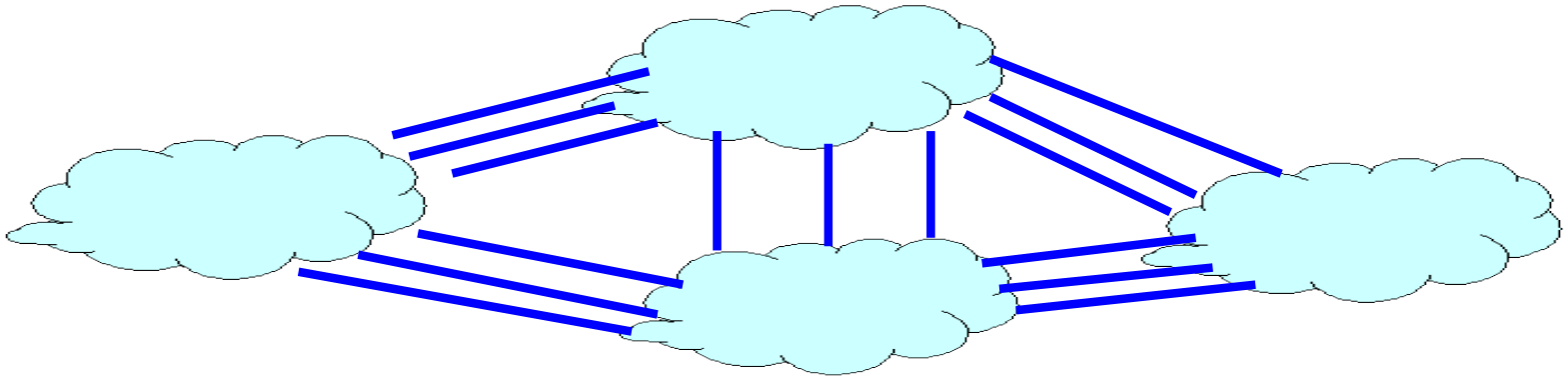
- Peers exchange traffic between customers
  - AS exports *only* customer routes to a peer
  - AS exports a peer's routes *only* to its customers
  - Often the relationship is settlement-free (i.e., no $$$)

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. and Sprint
  - Full peer-peer connections between tier-1 providers

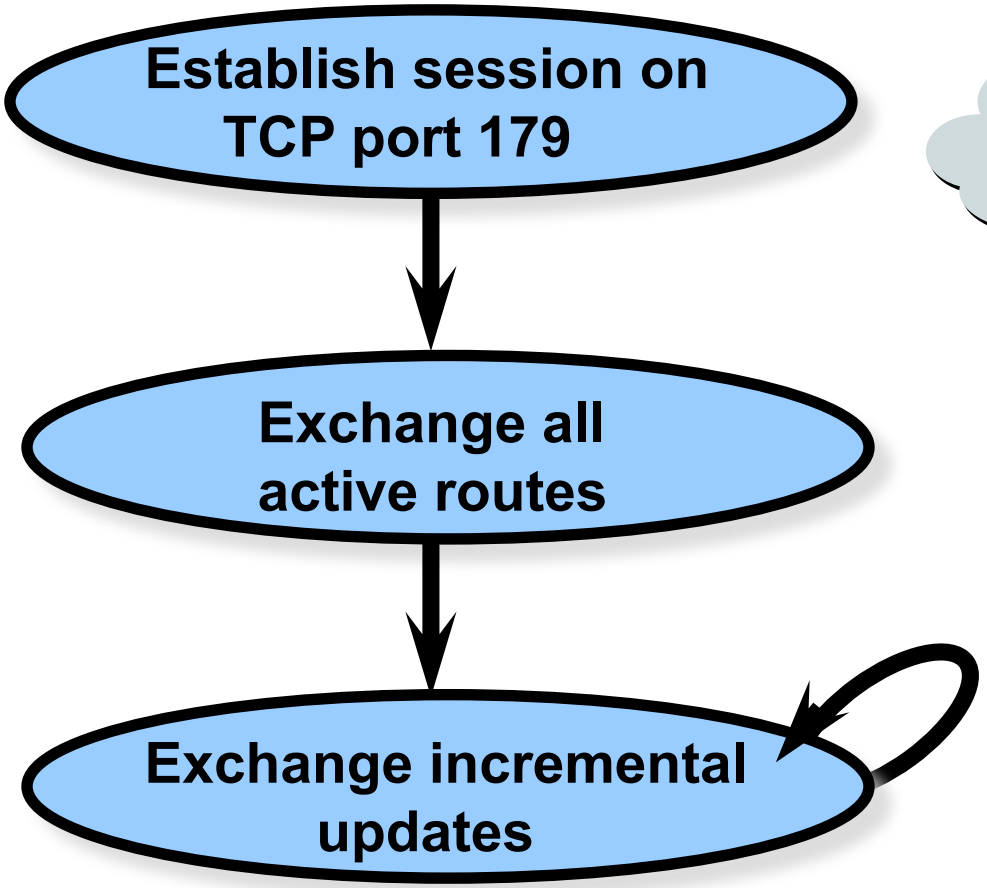


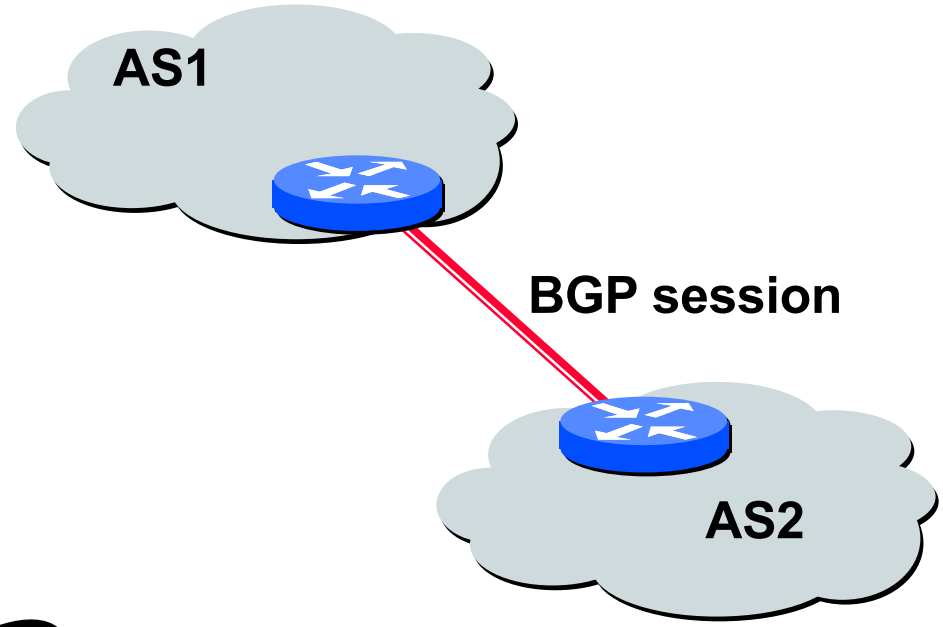
AS Structure: Other ASes

- 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 vast majority (e.g., 85-90%) of the ASes

Border Gateway Protocol

BGP Operations





While connection
is ALIVE exchange
route UPDATE messages

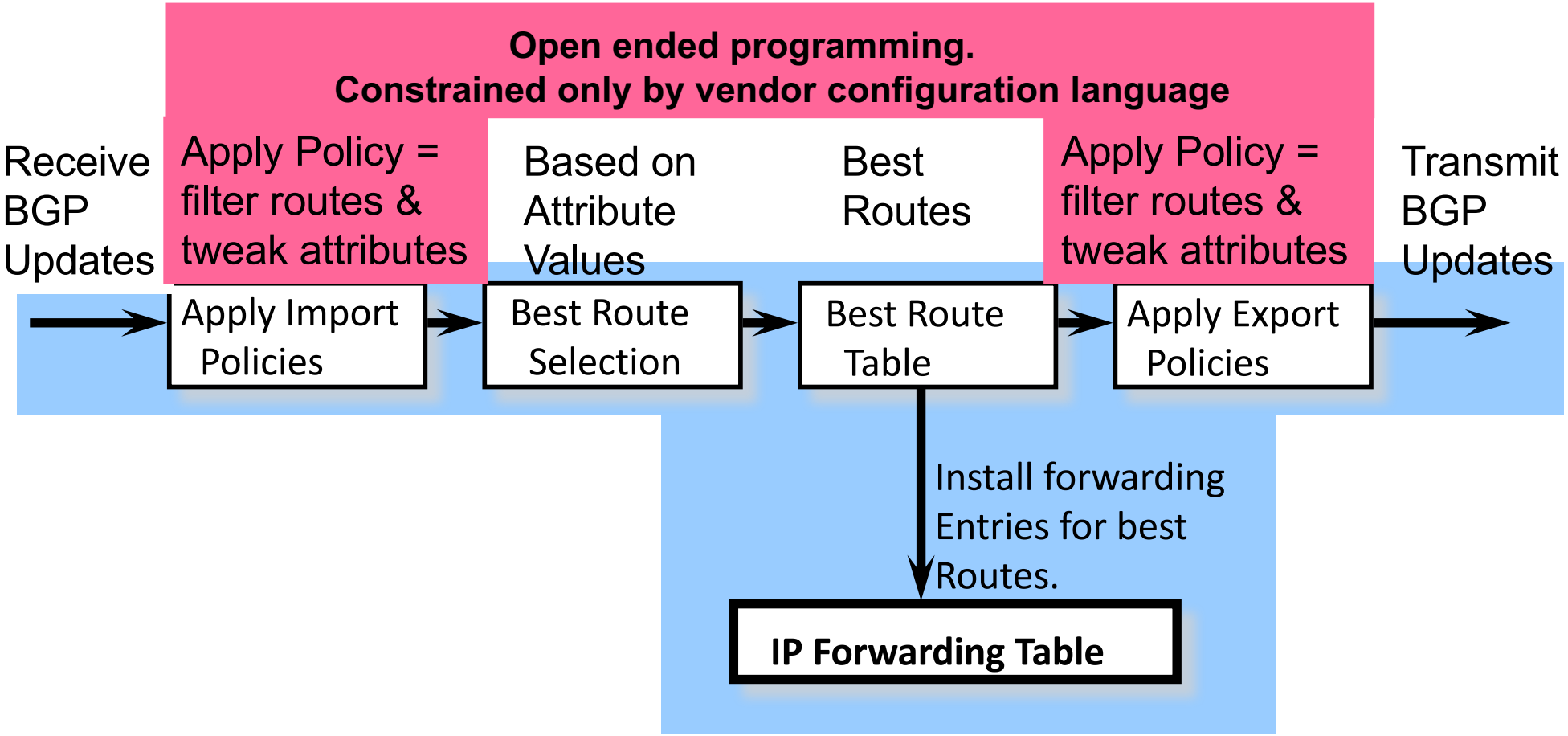
Incremental Protocol

- 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 Policy: Applying Policy to Routes

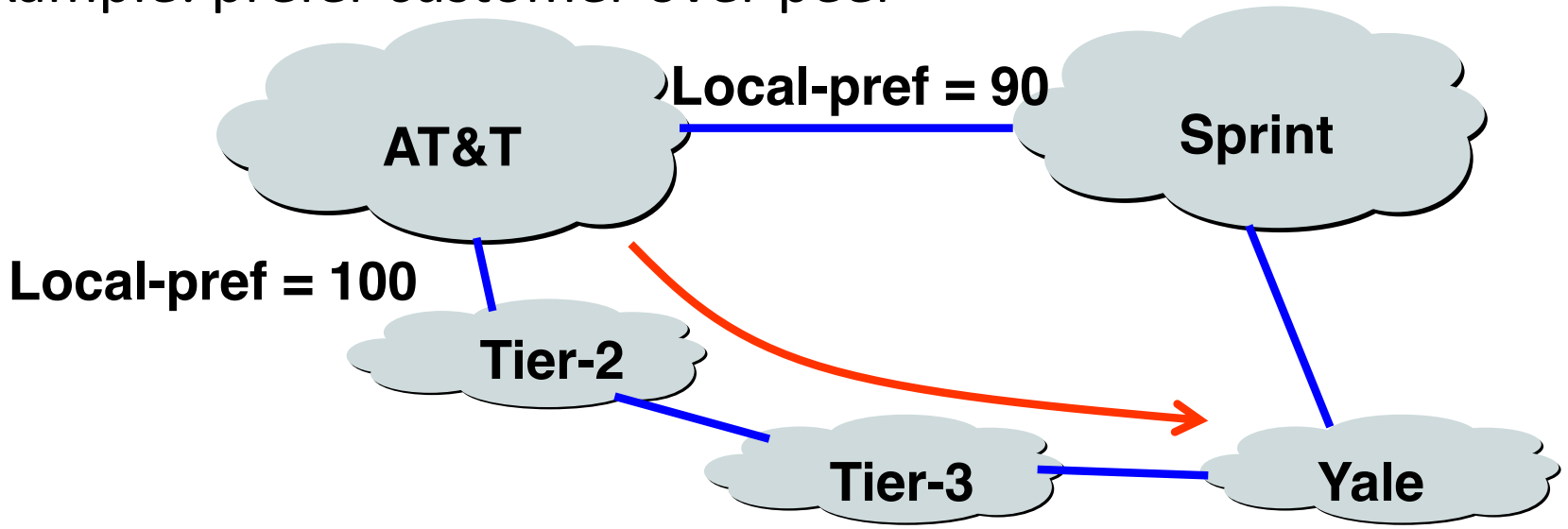
- Import policy
  - Filter unwanted routes from neighbor
  - Manipulate attributes to influence path selection
    - E.g., assign local preference to favored routes
- Export policy
  - Filter routes you don't want to tell your neighbor
    - E.g., don't tell a peer a route learned from other peer
  -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

BGP Policy: Influencing Decisions



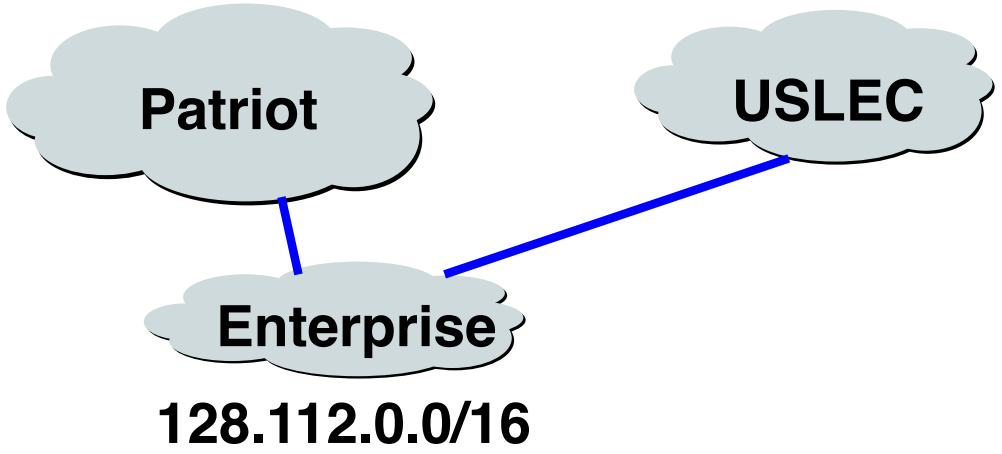
Import Policy: Local Preference

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



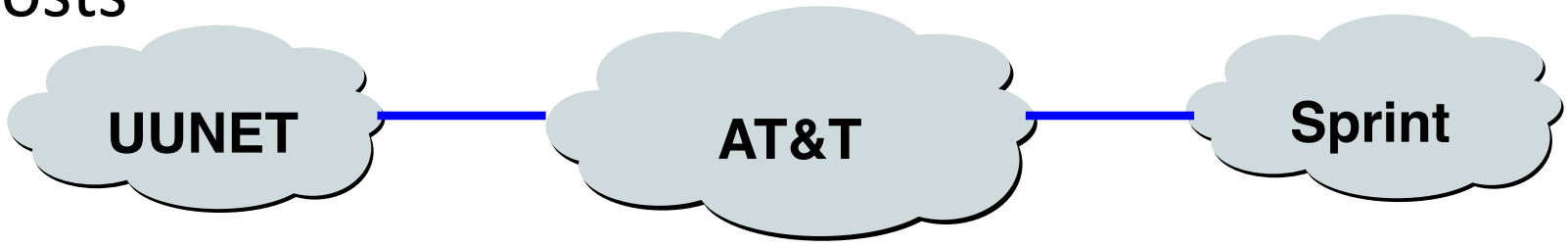
Import Policy: Filtering

- 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



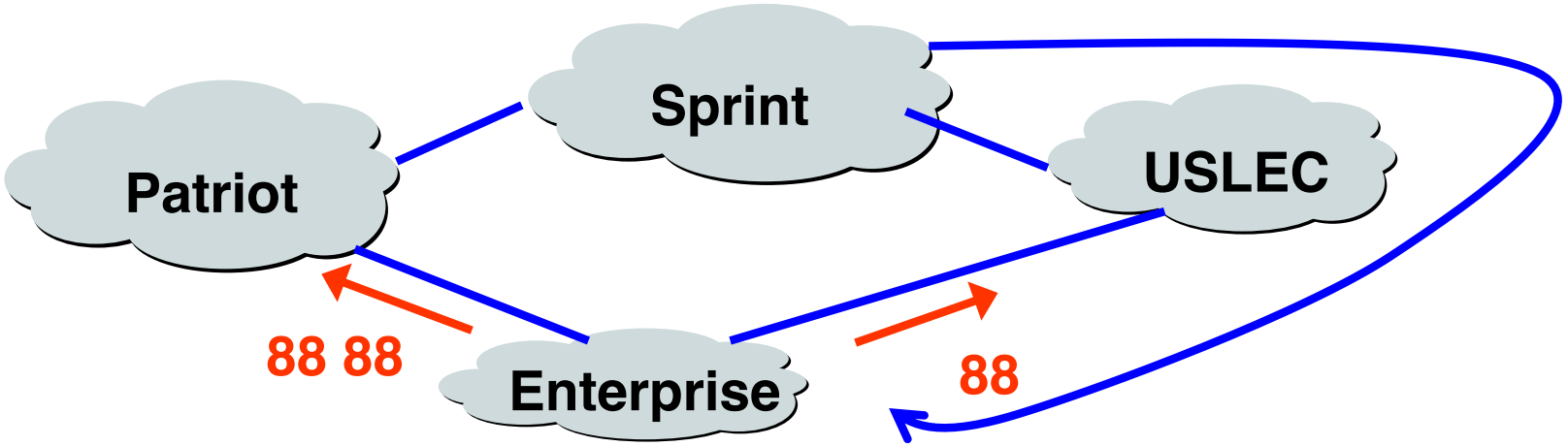
Export Policy: Filtering

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



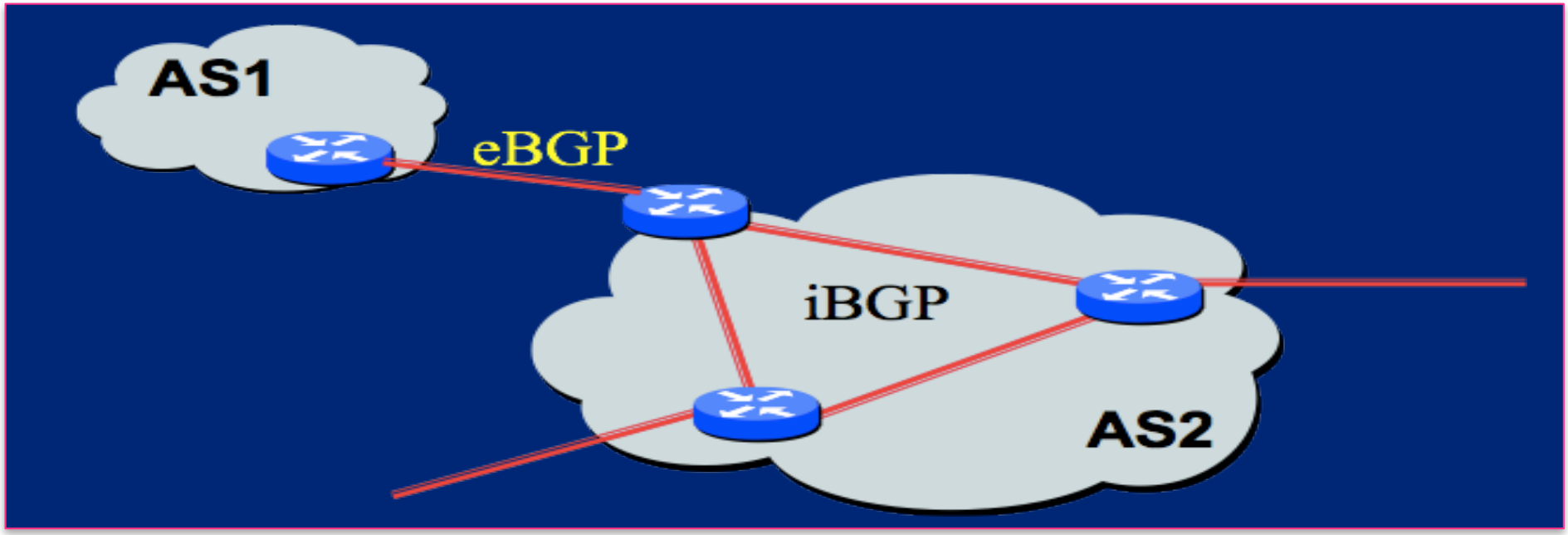
Export Policy: Attribute Manipulation

- 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



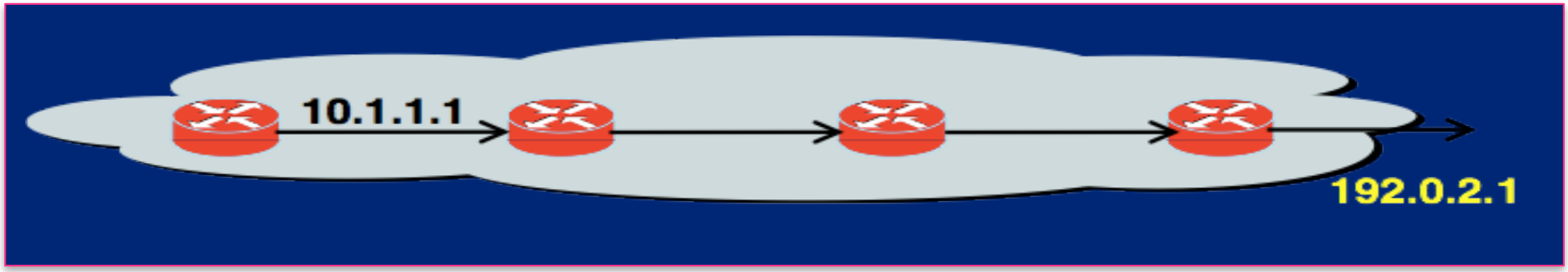
AS is not a single node

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



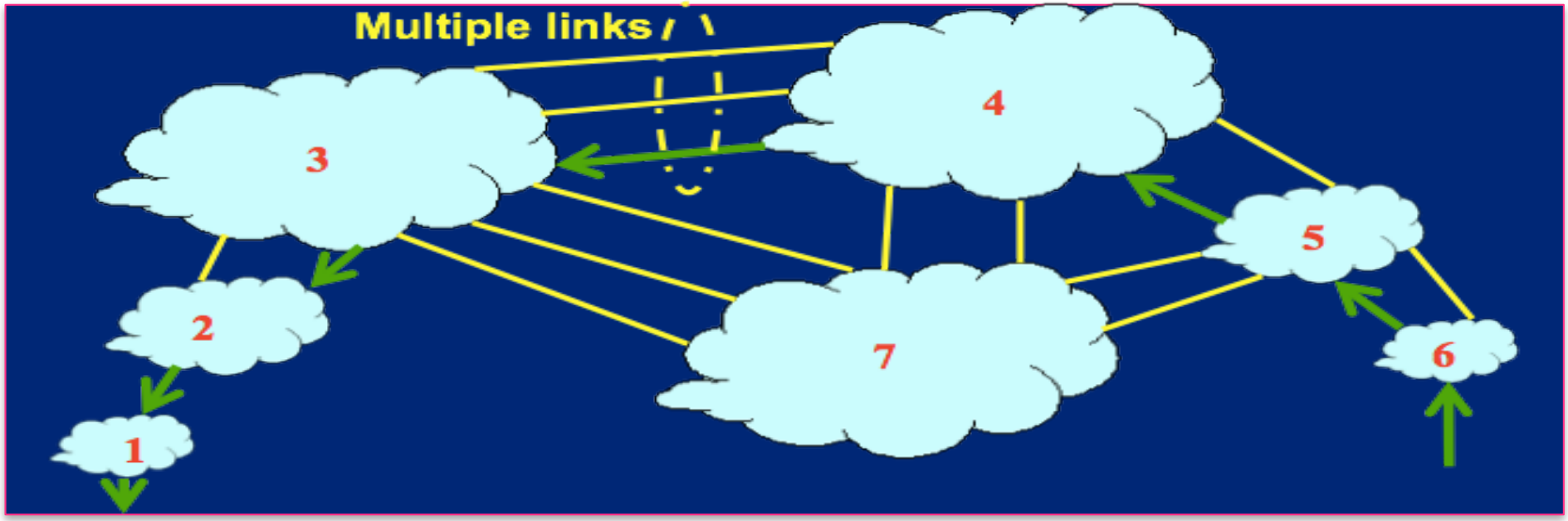
Joining BGP and IGP

- Border Gateway Protocol (BGP)
  - Maps a destination prefix to an egress point
  - 128.112.0.0/16 reached via 192.0.2.1
- Interior Gateway Protocol (IGP)
  - Used to compute paths within the AS
  - Maps an egress point to an outgoing link
  - 192.0.2.1 reached via 10.1.1.1



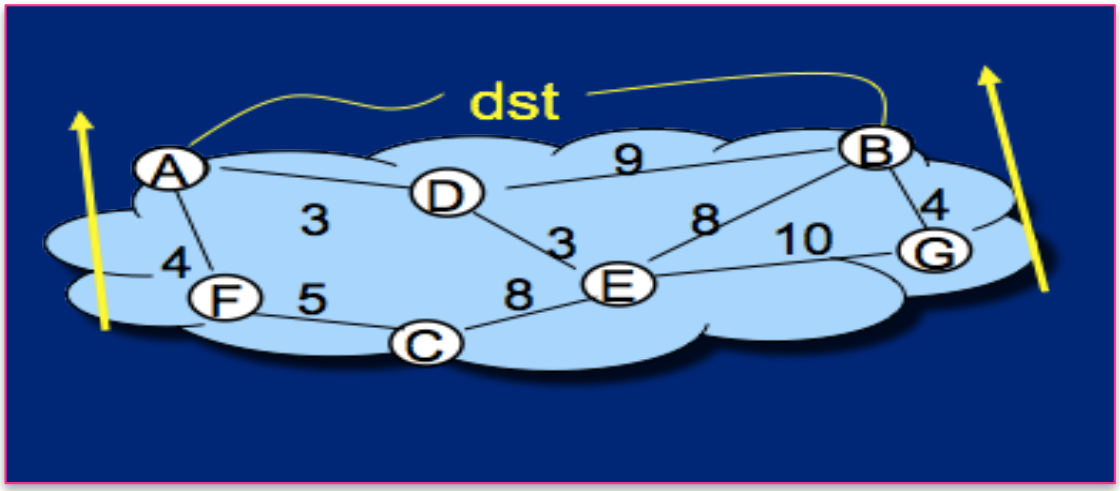
An AS may learn many routes

- Multiple connections to neighboring ASes
  - Multiple border routers may learn good routes
  - … with the same local-pref and AS path length



Hot-Potato (Early-Exit) Routing

- Hot-potato routing
  - Each router selects the closest egress point
  - … based on the path cost in intradomain protocol
- BGP decision process
  - Highest local preference
  - Shortest AS path
  - Closest egress point
  - Arbitrary tie break



BGP Thoughts

- Much more beyond basics to explore!
- Policy is a substantial factor
  - Can we even be sure independent decisions will be sensible overall?
- Other important factors:
  - Convergence effects
  - How well it scales
  - Integration with intradomain routing
  - And more …