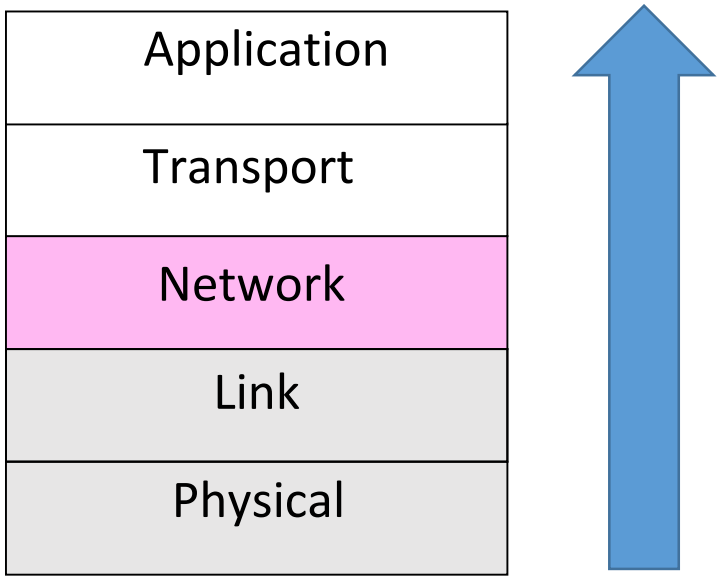
Network Layer (Routing)

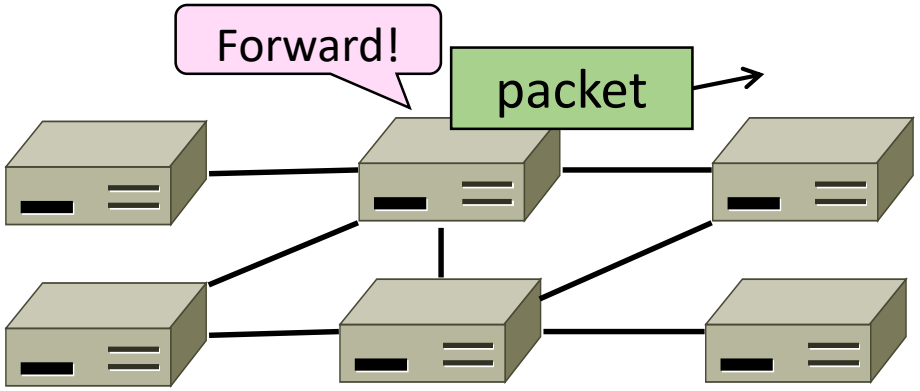
Where we are in the Course

- Moving on up to the Network Layer!

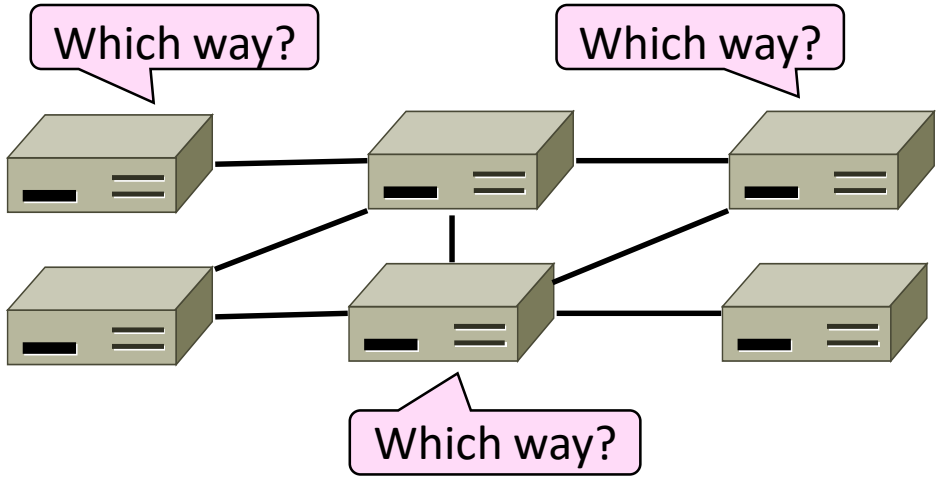


Routing versus Forwarding

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

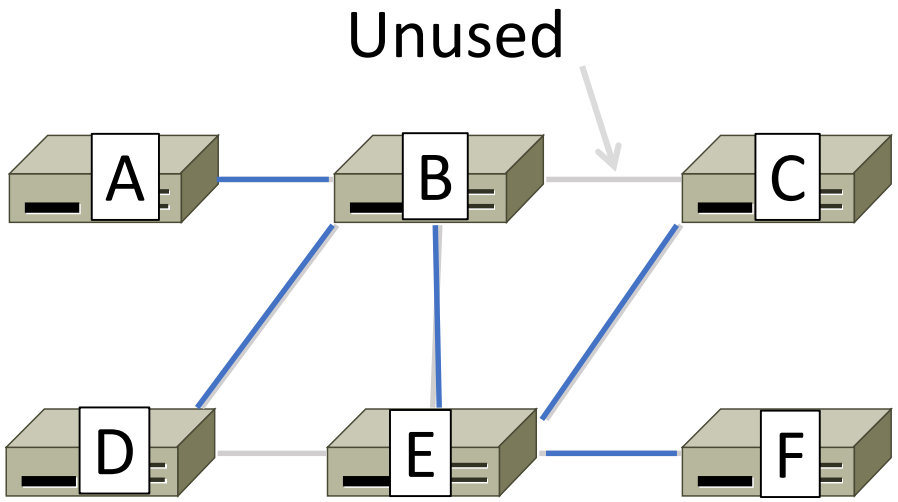


- Routing is the process of deciding in which direction to send traffic

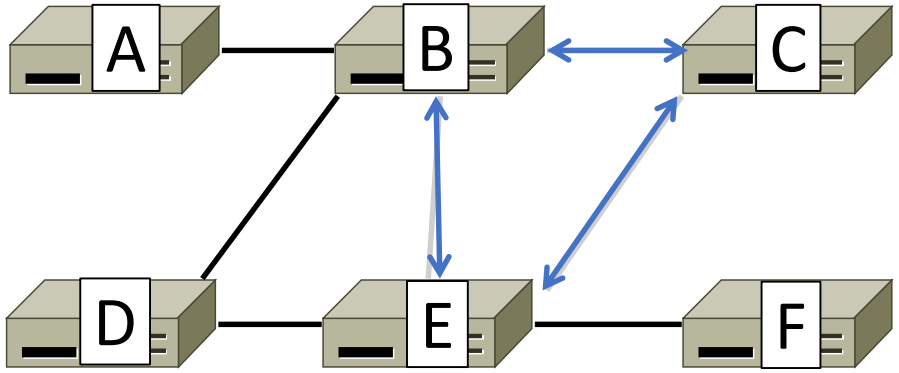


Improving on the Spanning Tree

- 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

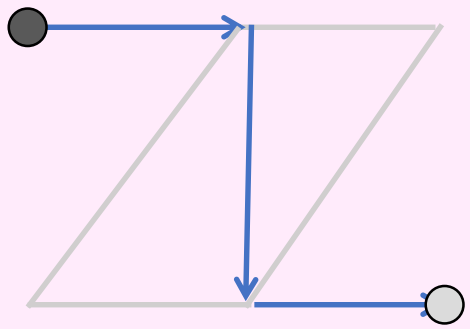
- Routing allocates network bandwidth adapting to failures; other mechanisms used at other timescales

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

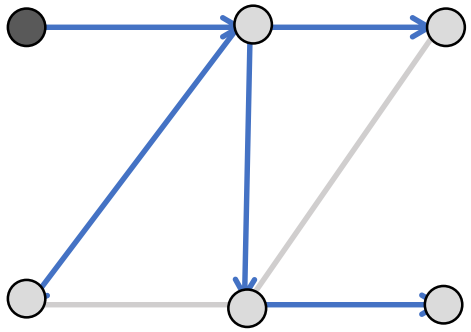
Delivery Models

- Different routing used for different delivery models

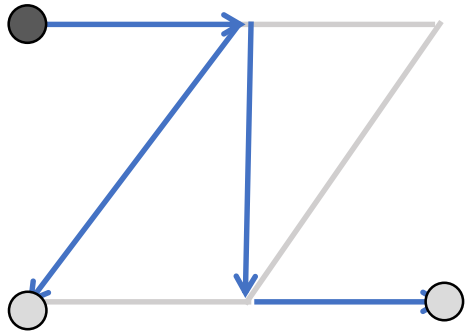
Unicast (§5.2)



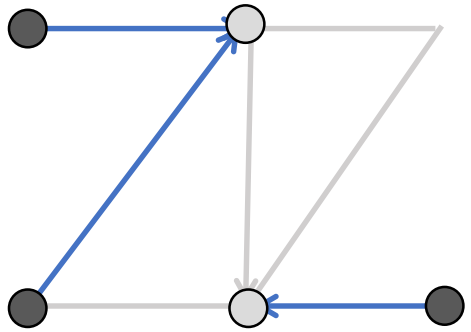
Broadcast (§5.2.7)



Multicast (§5.2.8)



Anycast (§5.2.9)



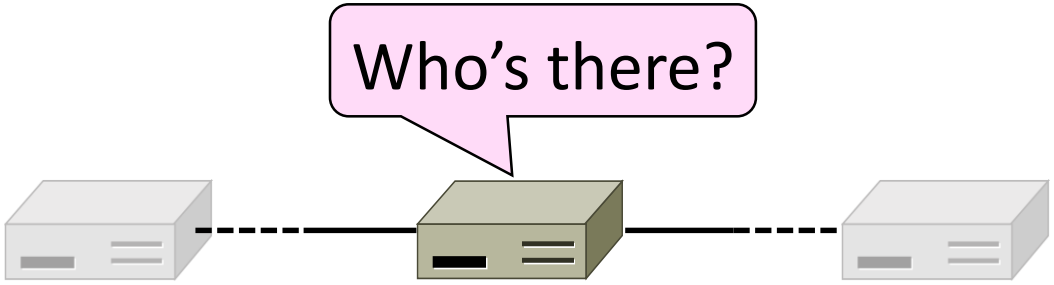
Goals of Routing Algorithms

- We want several properties of any routing scheme:

Property	Meaning
Correctness	Finds paths that work
Efficient paths	Uses network bandwidth well
Fair paths	Doesn’t starve any nodes
Fast convergence	Recovers quickly after changes
Scalability	Works well as network grows large

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

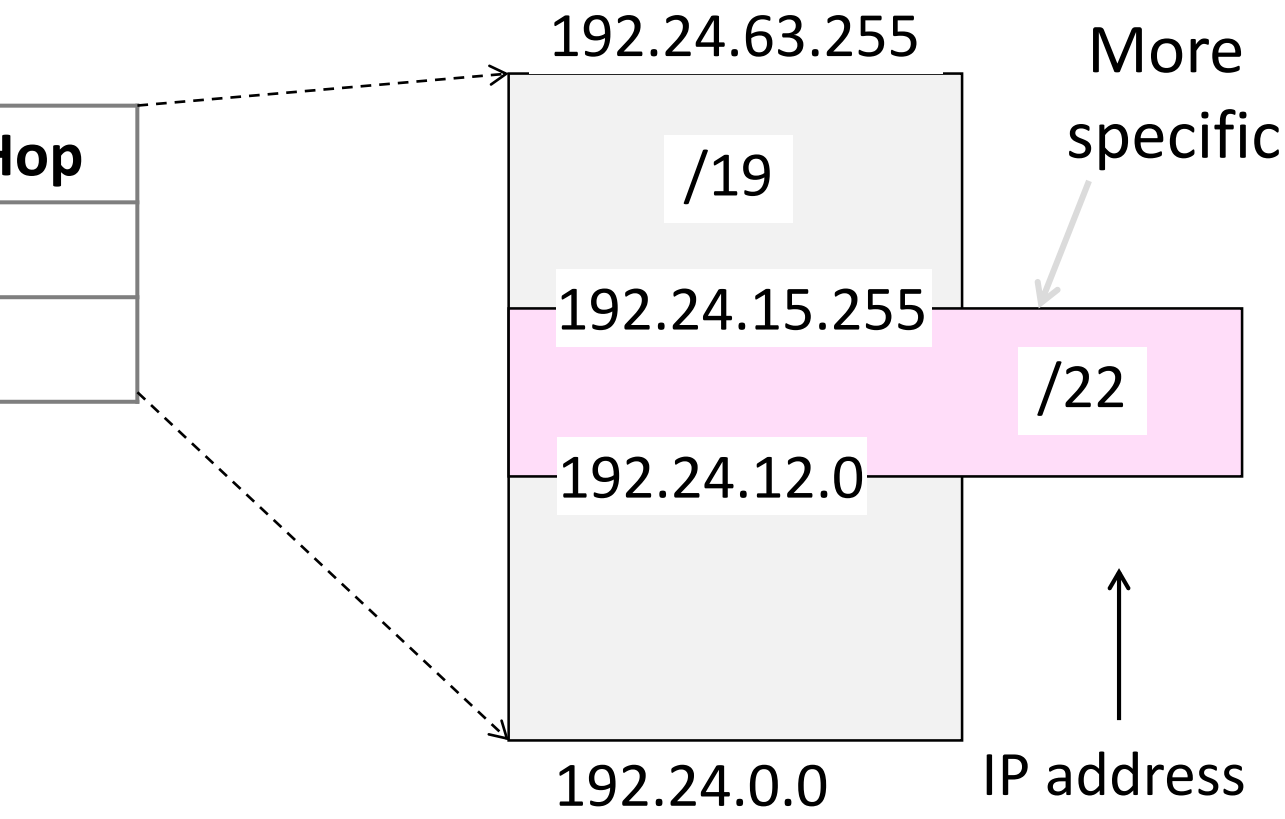


Recap: Classless Inter-Domain Routing (CIDR)

- In the Internet:
  - Hosts on same network have IPs in the same IP prefix
  - Hosts send off-network traffic to nearest router to handle
  - Routers discover the routes to use
  - Routers use longest prefix matching to send packets to the right next hop

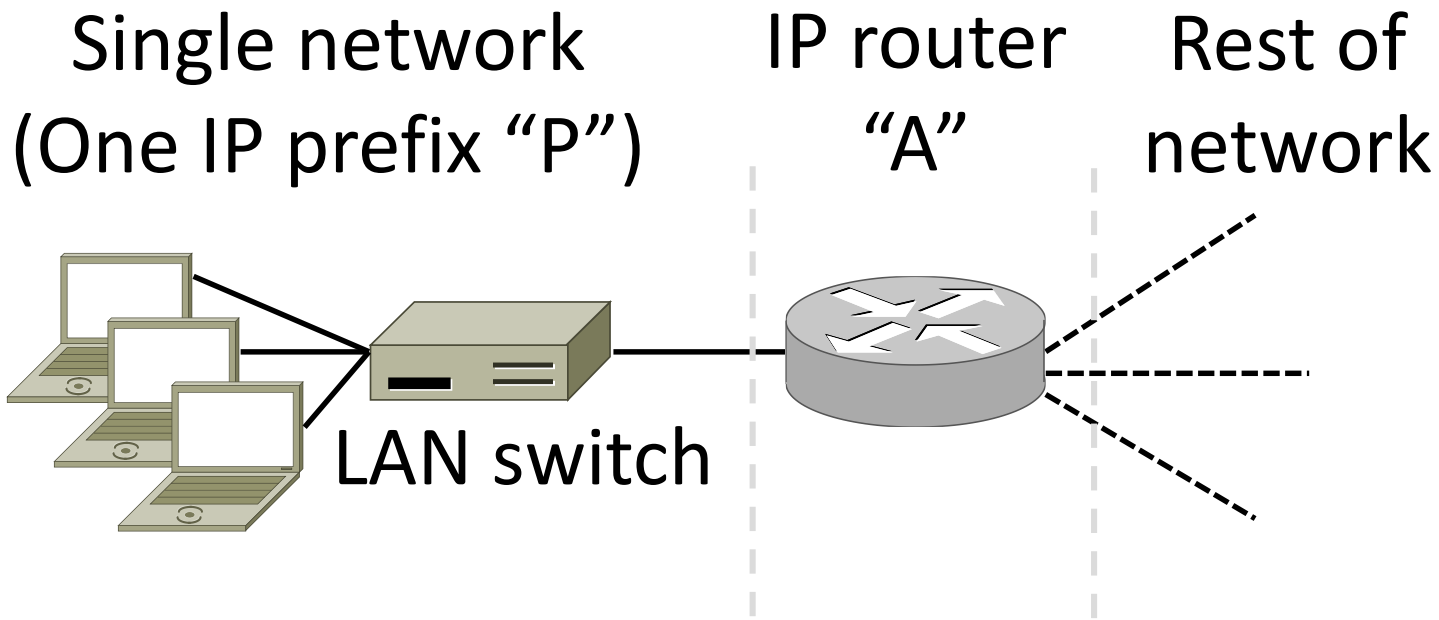
Longest Matching Prefix

Prefix	Next Hop
192.24.0.0/19	D
192.24.12.0/22	B



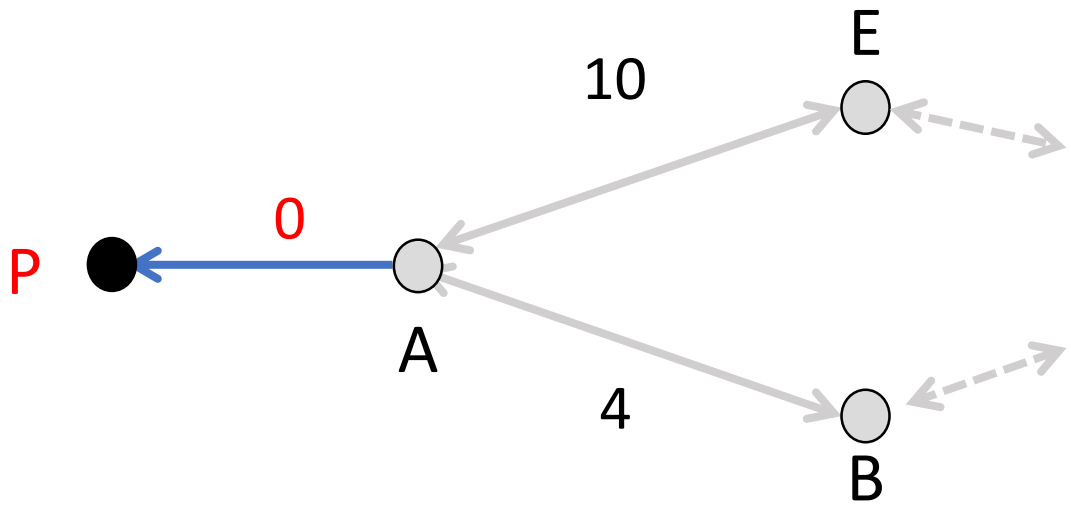
Host/Router Combination

- Hosts attach to routers as IP prefixes
  - Router needs table to reach all hosts



Network Topology for Routing

- Group hosts under IP prefix connected to router
  - One entry for all hosts



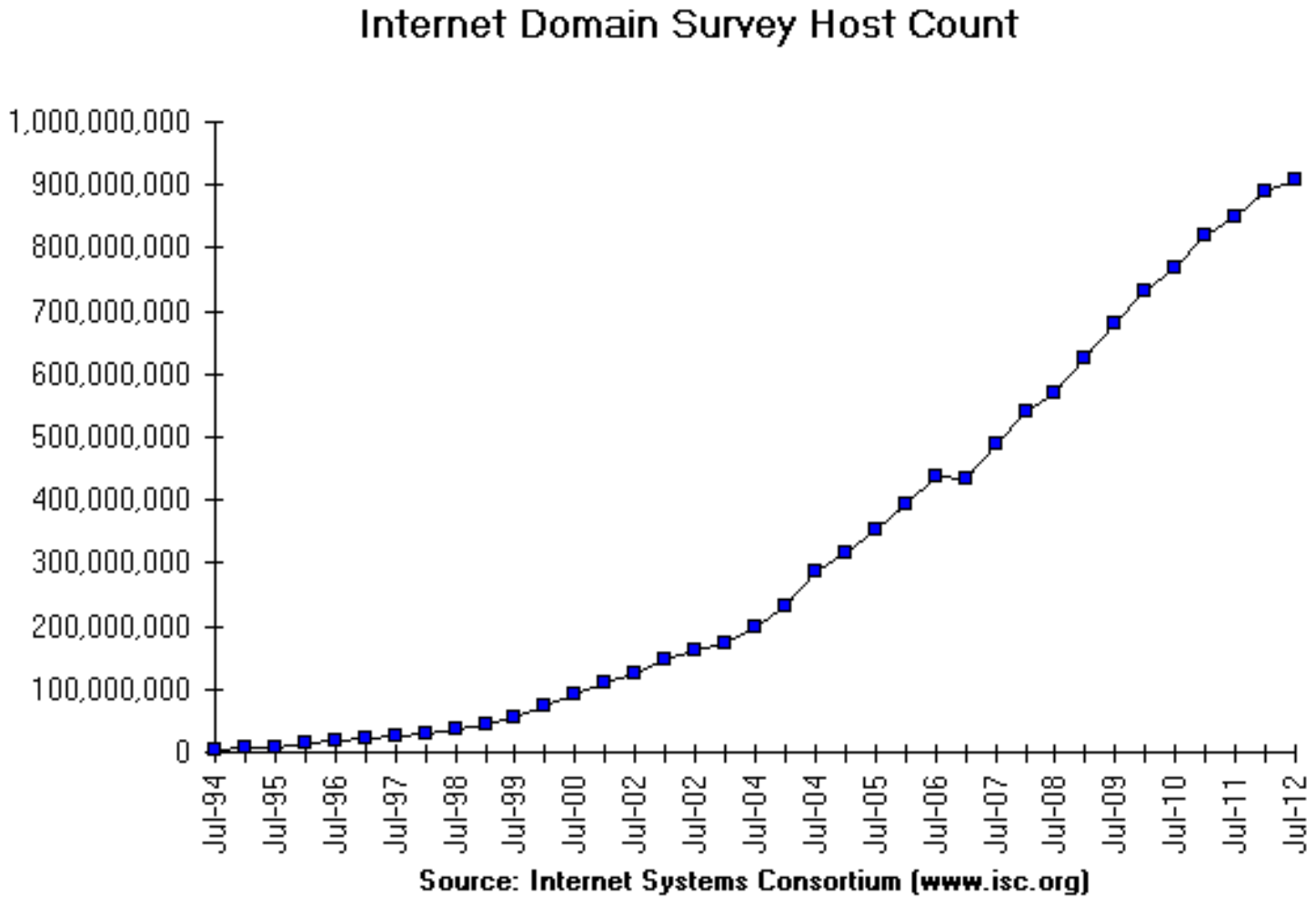
Network Topology for Routing (2)

- Routing now works!
  - Routers advertise IP prefixes for hosts
  - Router addresses are “/32” prefixes
  - Lets all routers find a path to hosts
  - Hosts find by sending to their router

Hierarchical Routing

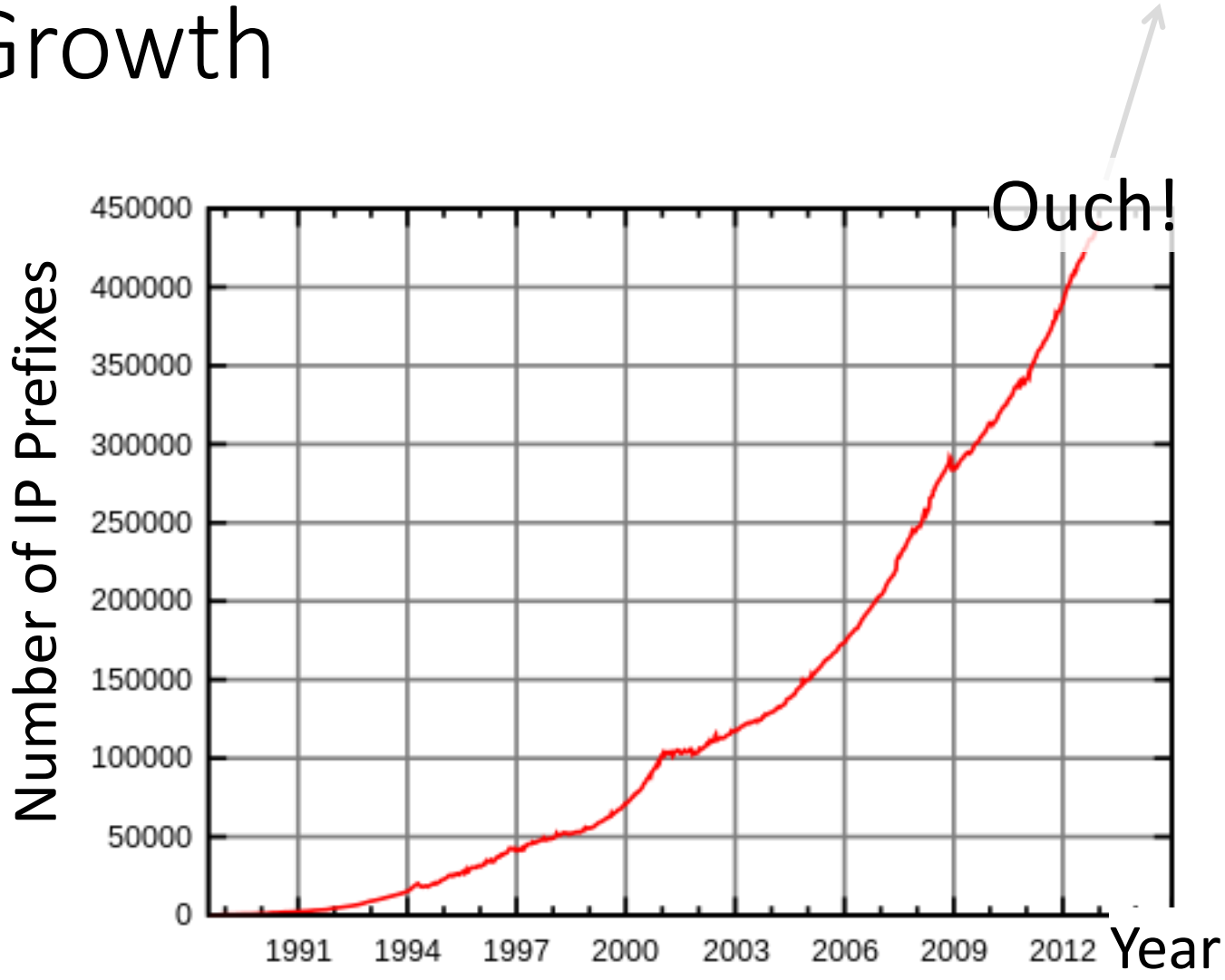
Internet Growth

- At least a billion Internet hosts and growing …



Internet Routing Growth

- Internet growth translates into routing table growth (even using prefixes) …



Source: By Mro (Own work), CC-BY-SA-3.0 , via Wikimedia Commons

Impact of Routing Growth

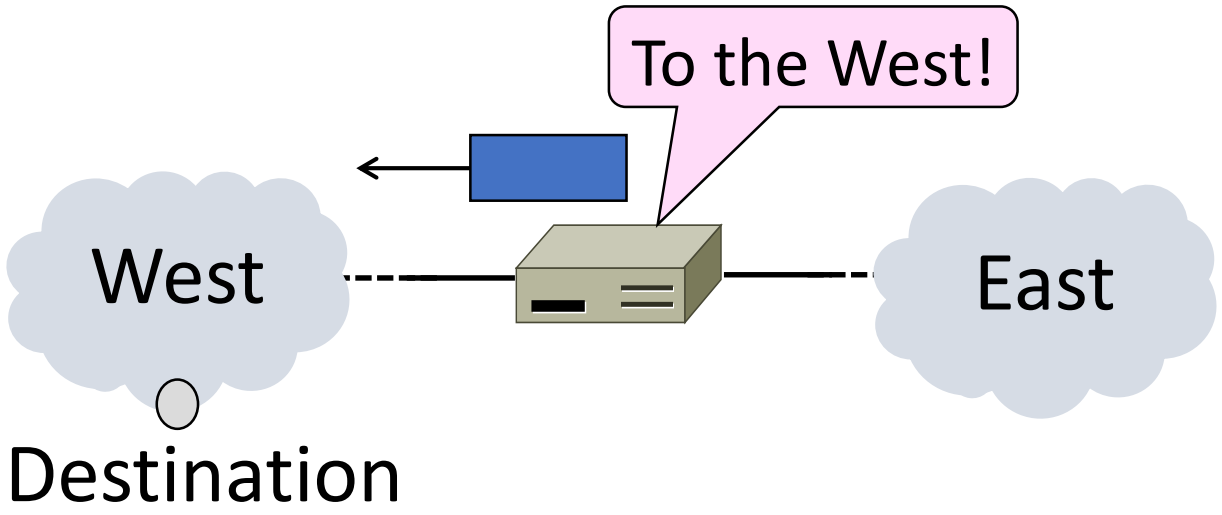
1. Forwarding tables grow
   - Larger router memories, may increase lookup time
2. Routing messages grow
   - Need to keeps all nodes informed of larger topology
3. Routing computation grows
   - Shortest path calculations grow faster than the network

Techniques to Scale Routing

- First: Network hierarchy
  - Route to network regions
- Next: IP prefix aggregation
  - Combine, and split, prefixes

Idea

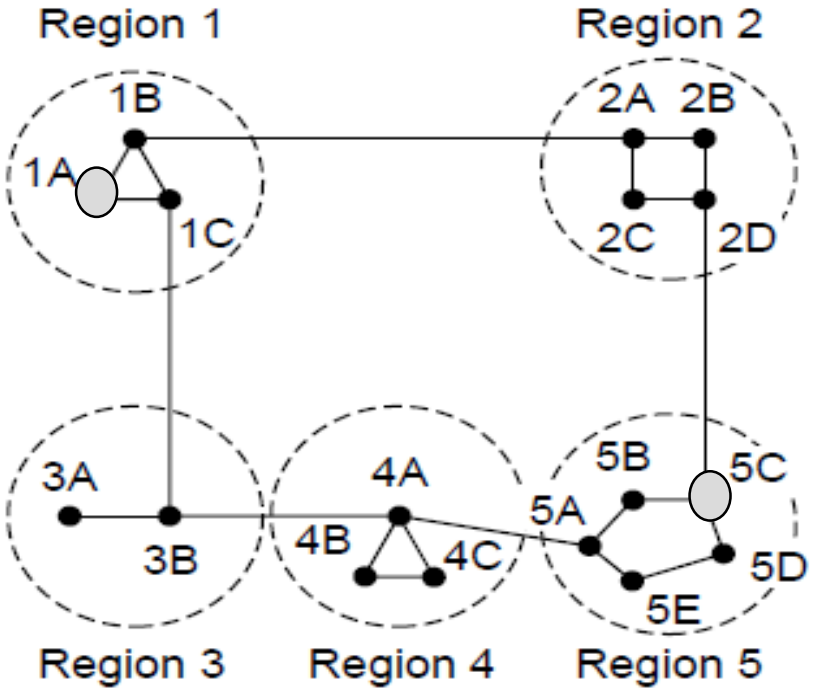
- Scale routing using hierarchy with regions
  - Route to regions, not individual nodes



Hierarchical Routing

- Introduce a larger routing unit
  - IP prefix (hosts) ← from one host
  - Region, e.g., ISP network
- Route first to the region, then to the IP prefix within the region
  - Hide details within a region from outside of the region

Hierarchical Routing (2)



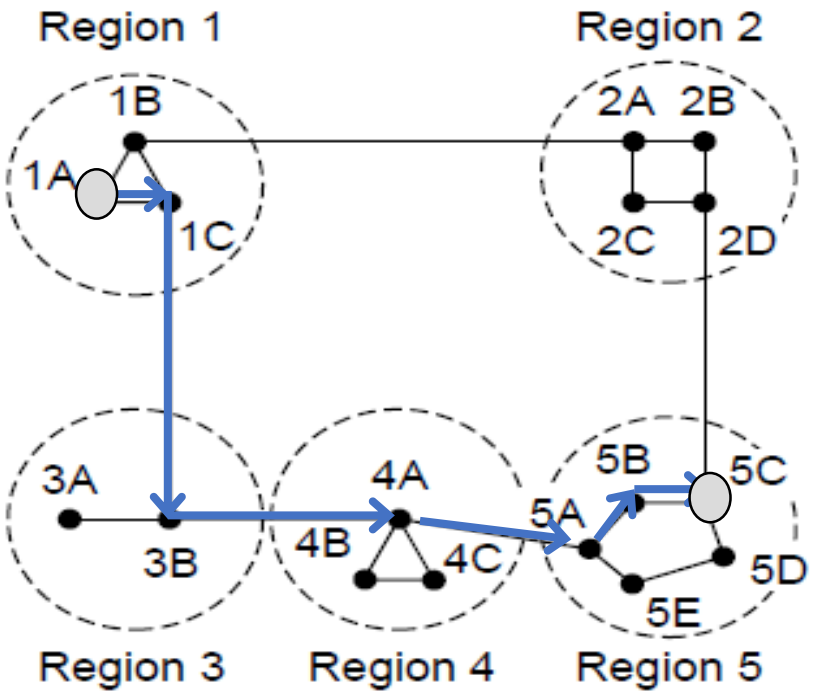
Full table for 1A

Dest.	Line	Hops
1A	–	–
1B	1B	1
1C	1C	1
2A	1B	2
2B	1B	3
2C	1B	3
2D	1B	4
3A	1C	3
3B	1C	2
4A	1C	3
4B	1C	4
4C	1C	4
5A	1C	4
5B	1C	5
5C	1B	5
5D	1C	6
5E	1C	5

Hierarchical table for 1A

Dest.	Line	Hops
1A	–	–
1B	1B	1
1C	1C	1
2	1B	2
3	1C	2
4	1C	3
5	1C	4

Hierarchical Routing (3)



Full table for 1A

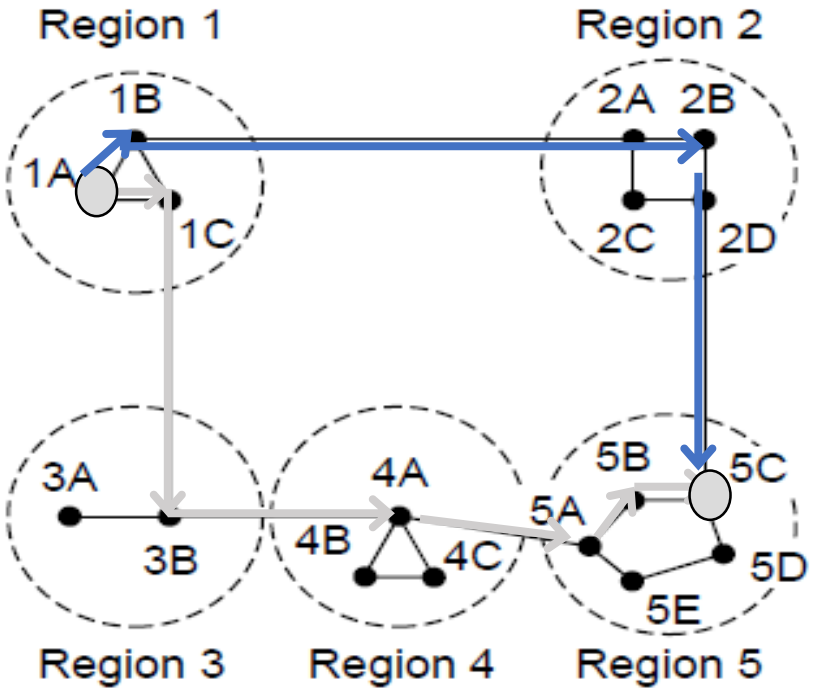
Dest.	Line	Hops
1A	–	–
1B	1B	1
1C	1C	1
2A	1B	2
2B	1B	3
2C	1B	3
2D	1B	4
3A	1C	3
3B	1C	2
4A	1C	3
4B	1C	4
4C	1C	4
5A	1C	4
5B	1C	5
5C	1B	5
5D	1C	6
5E	1C	5

Hierarchical table for 1A

Dest.	Line	Hops
1A	–	–
1B	1B	1
1C	1C	1
2	1B	2
3	1C	2
4	1C	3
5	1C	4

Hierarchical Routing (4)

- Penalty is longer paths



Full table for 1A

Dest.	Line	Hops
1A	–	–
1B	1B	1
1C	1C	1
2A	1B	2
2B	1B	3
2C	1B	3
2D	1B	4
3A	1C	3
3B	1C	2
4A	1C	3
4B	1C	4
4C	1C	4
5A	1C	4
5B	1C	5
5C	1B	5
5D	1C	6
5E	1C	5

Hierarchical table for 1A

Dest.	Line	Hops
1A	–	–
1B	1B	1
1C	1C	1
2	1B	2
3	1C	2
4	1C	3
5	1C	4

1C is best route to region 5, except for destination 5C

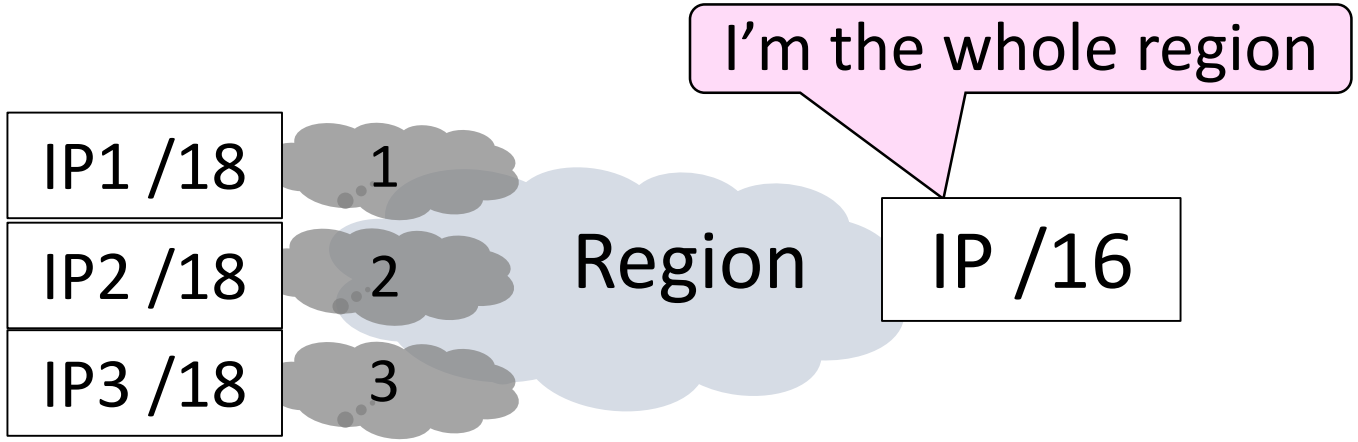
Observations

- Outside a region, nodes have one route to all hosts within the region
  - This gives savings in table size, messages and computation
- However, each node may have a different route to an outside region
  - Routing decisions are still made by individual nodes; there is no single decision made by a region

IP Prefix Aggregation and Subnets

Idea

- Scale routing by adjusting the size of IP prefixes
  - Split (subnets) and join (aggregation)



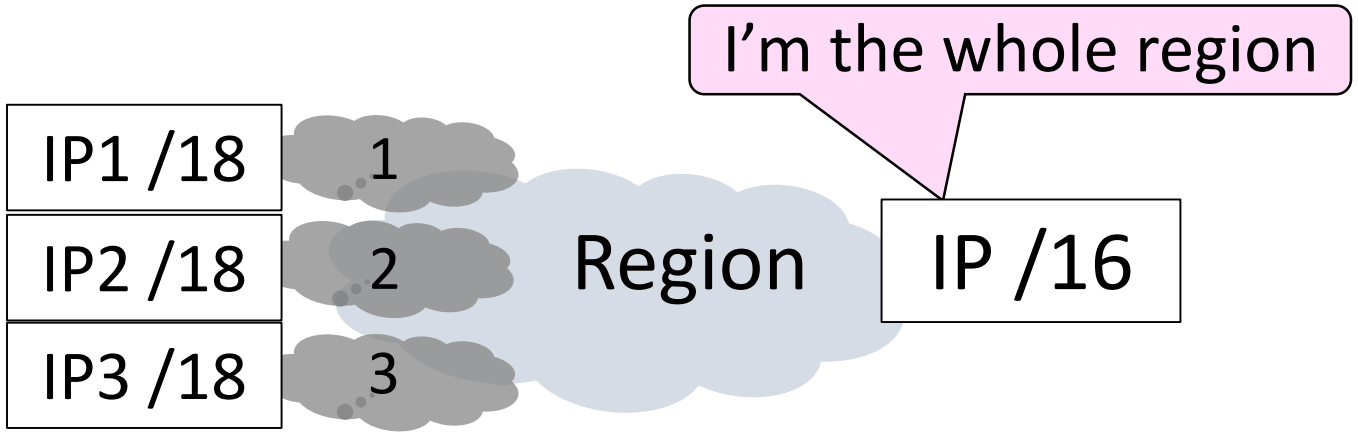
Recall

- IP addresses are allocated in blocks called IP prefixes, e.g., 18.31.0.0/16
  - Hosts on one network in same prefix
- "/N" prefix has the first N bits fixed and contains 2^{32-N} addresses
  - E.g., a "/24" has 256 addresses
- Routers keep track of prefix lengths
  - Use it as part of longest prefix matching

Routers can change prefix lengths without affecting hosts

Prefixes and Hierarchy

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



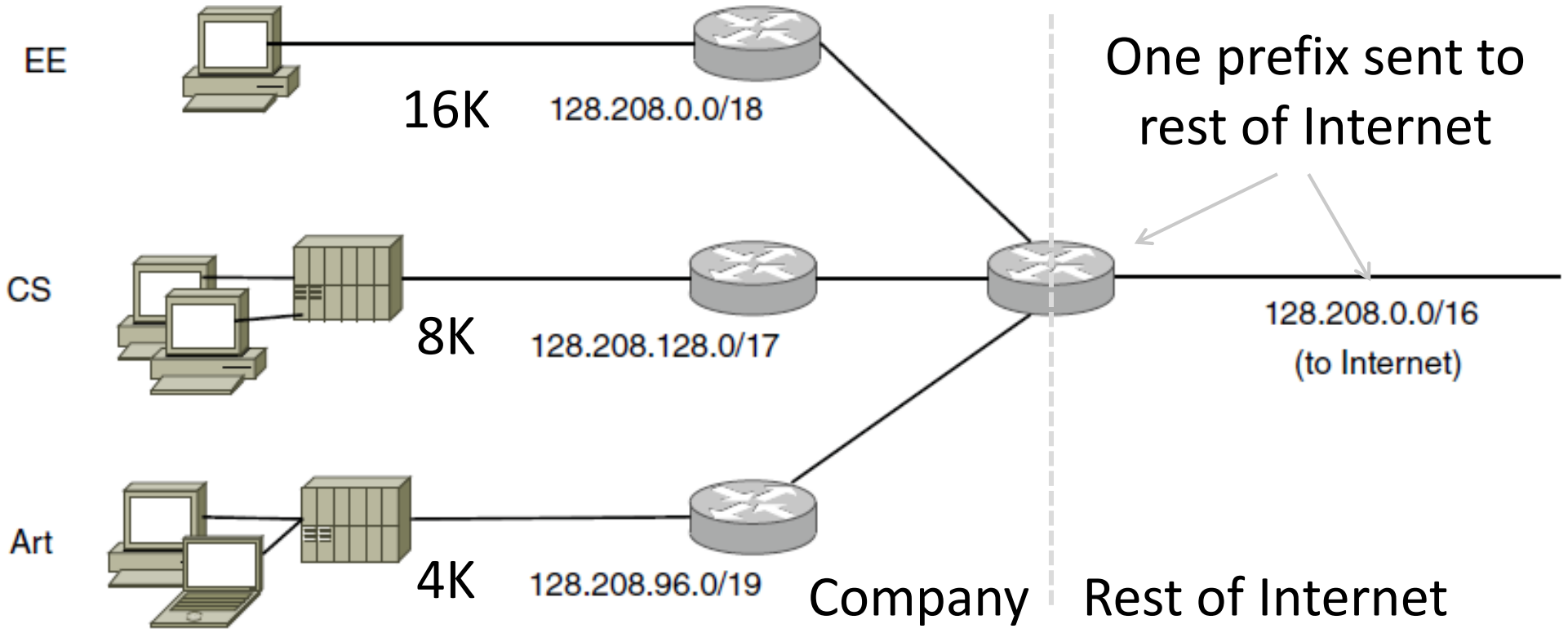
Subnets and Aggregation

- Two use cases for adjusting the size of IP prefixes; both reduce routing table

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

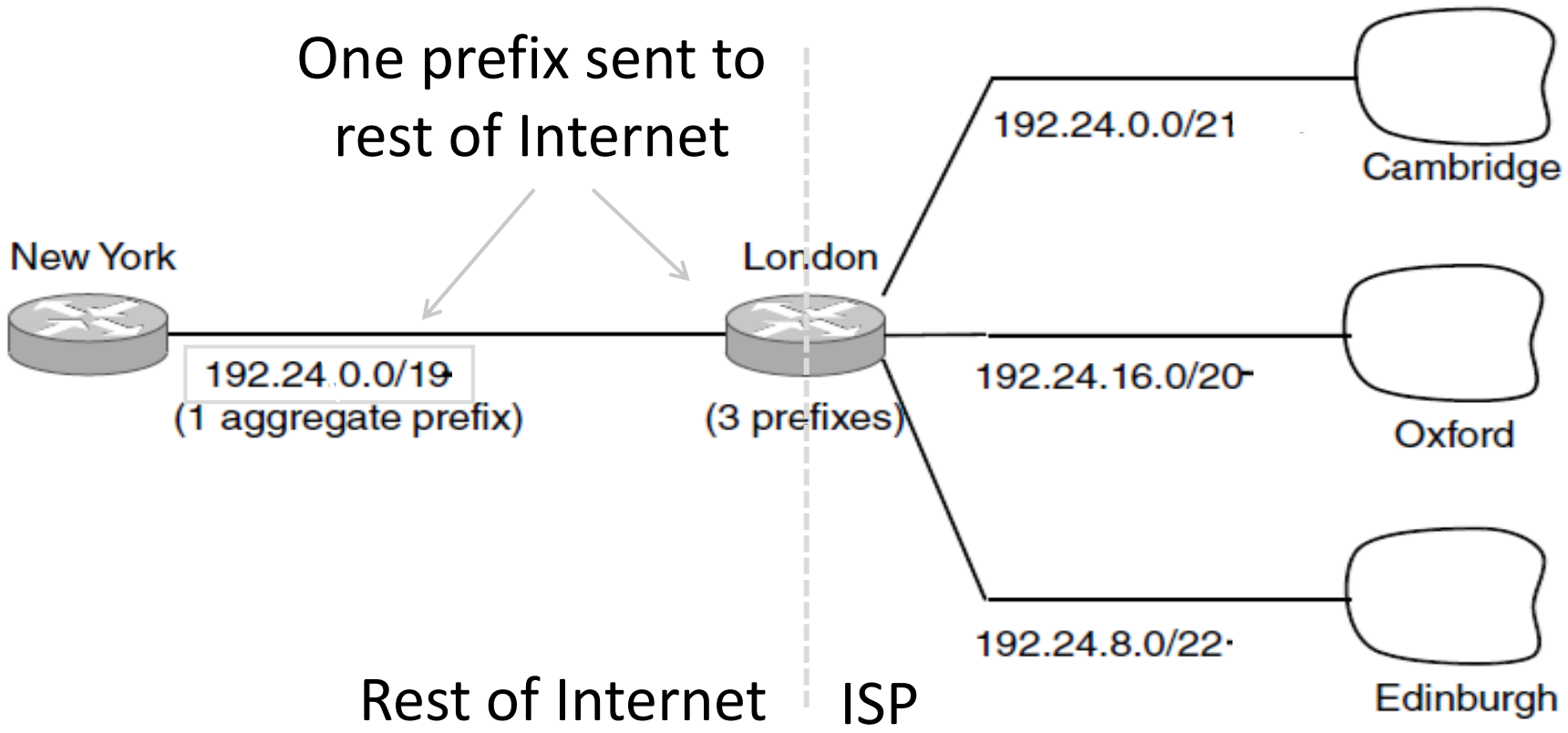
Subnets

- Internally split up one IP prefix



Aggregation

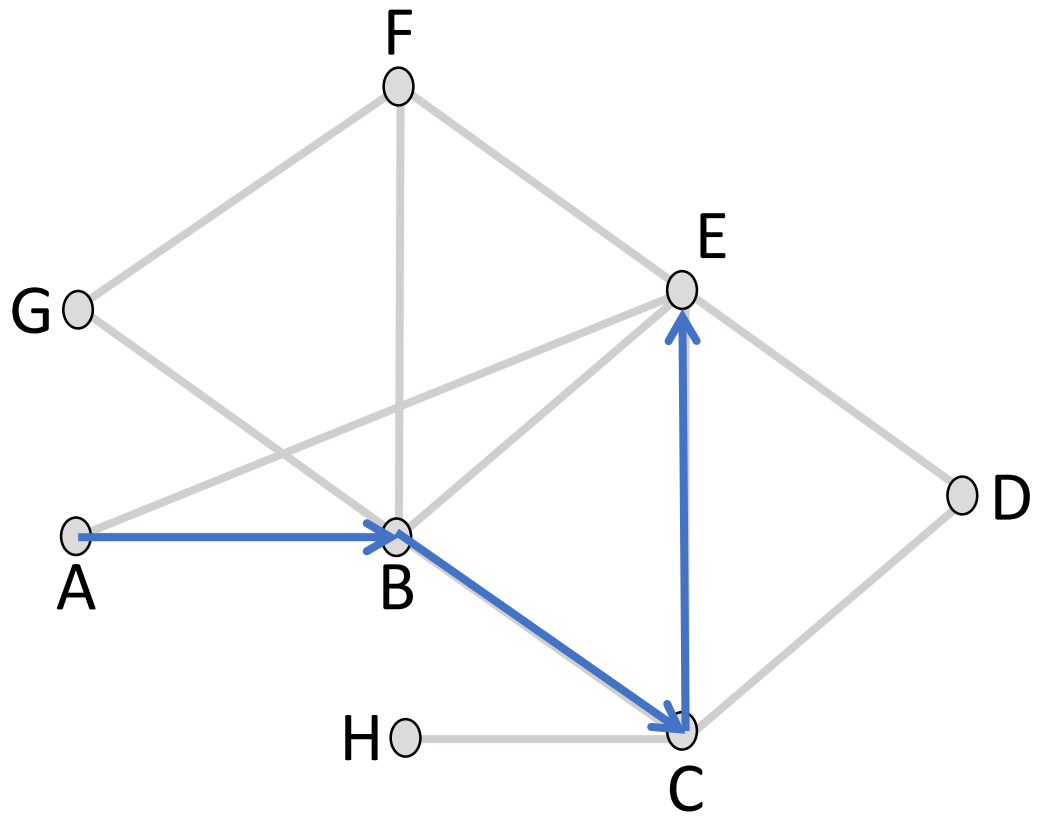
- Externally join multiple separate IP prefixes



Best 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

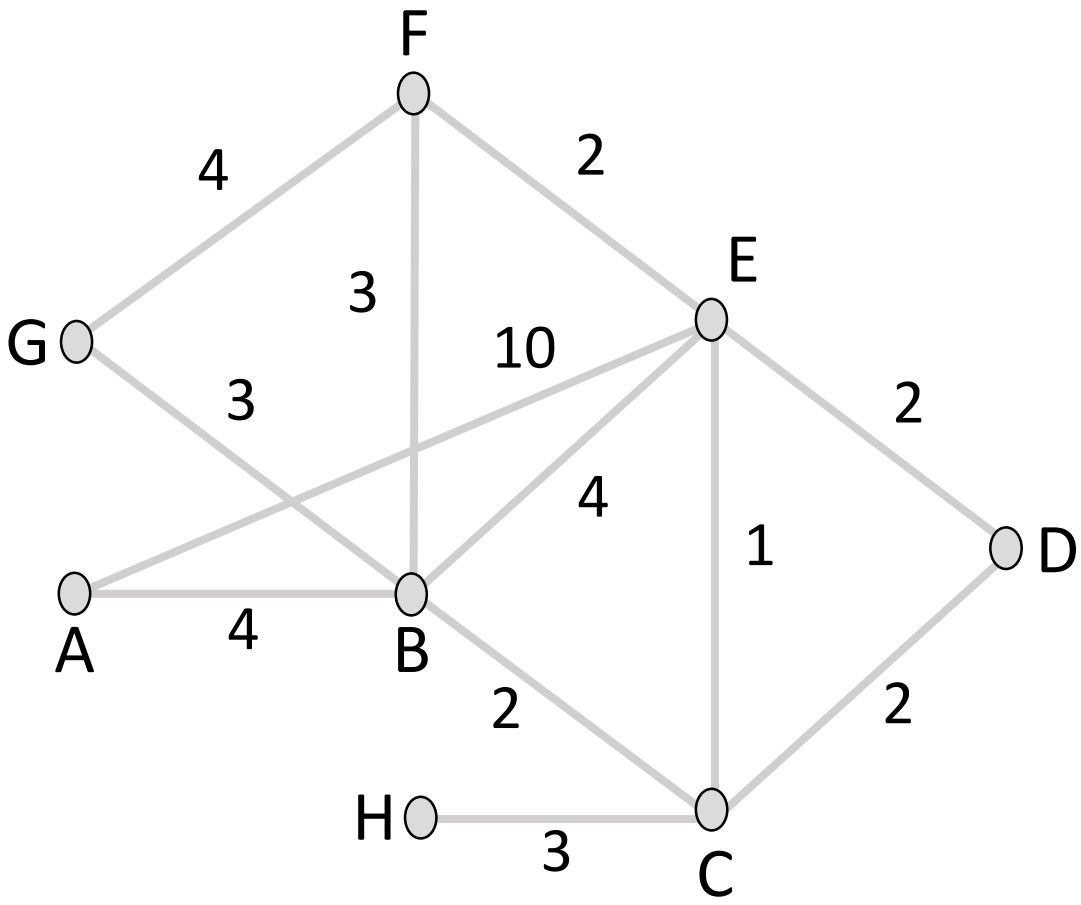
We'll approximate "best" by a cost function that captures the factors

- 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 any break ties

Shortest Paths (2)

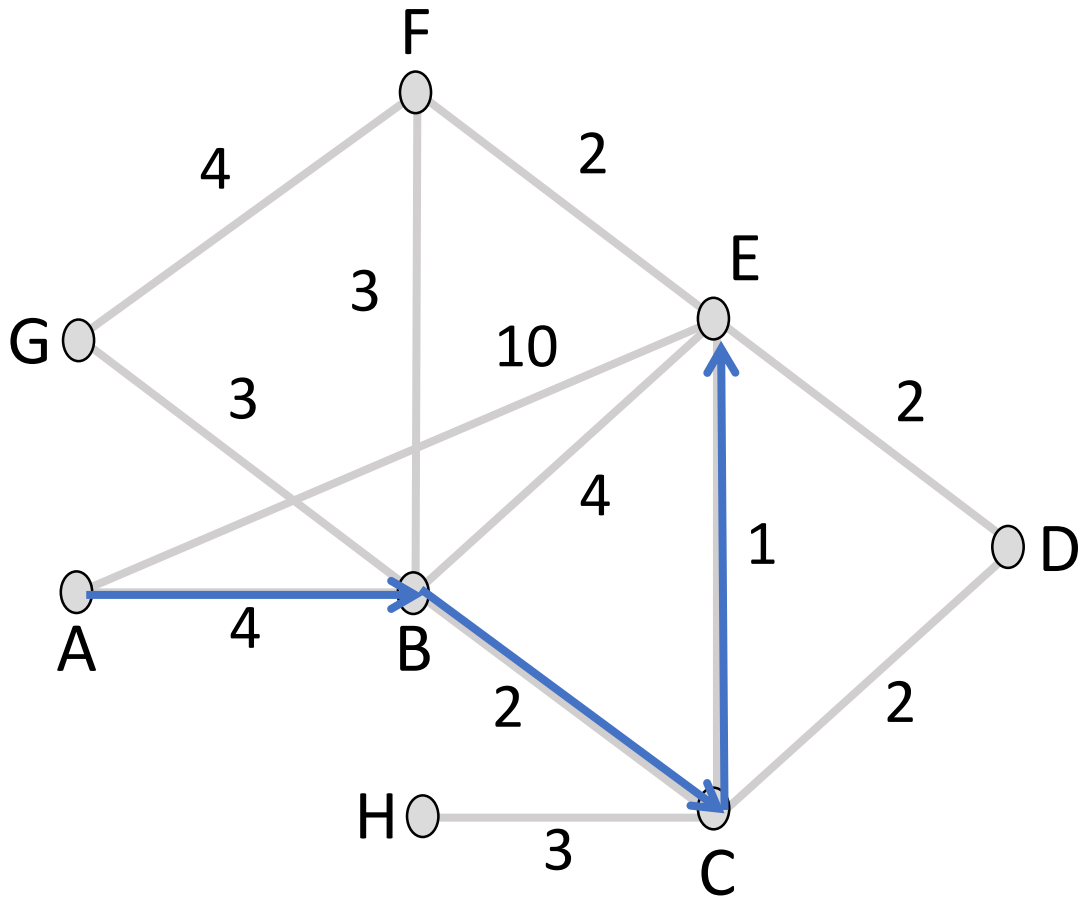
- Find the shortest path A → E
- All links are bidirectional, with equal costs in each direction
  - Can extend model to unequal costs if needed



Shortest Paths (3)

- ABCE is a shortest path
- dist(ABCE) = 4 + 2 + 1 = 7

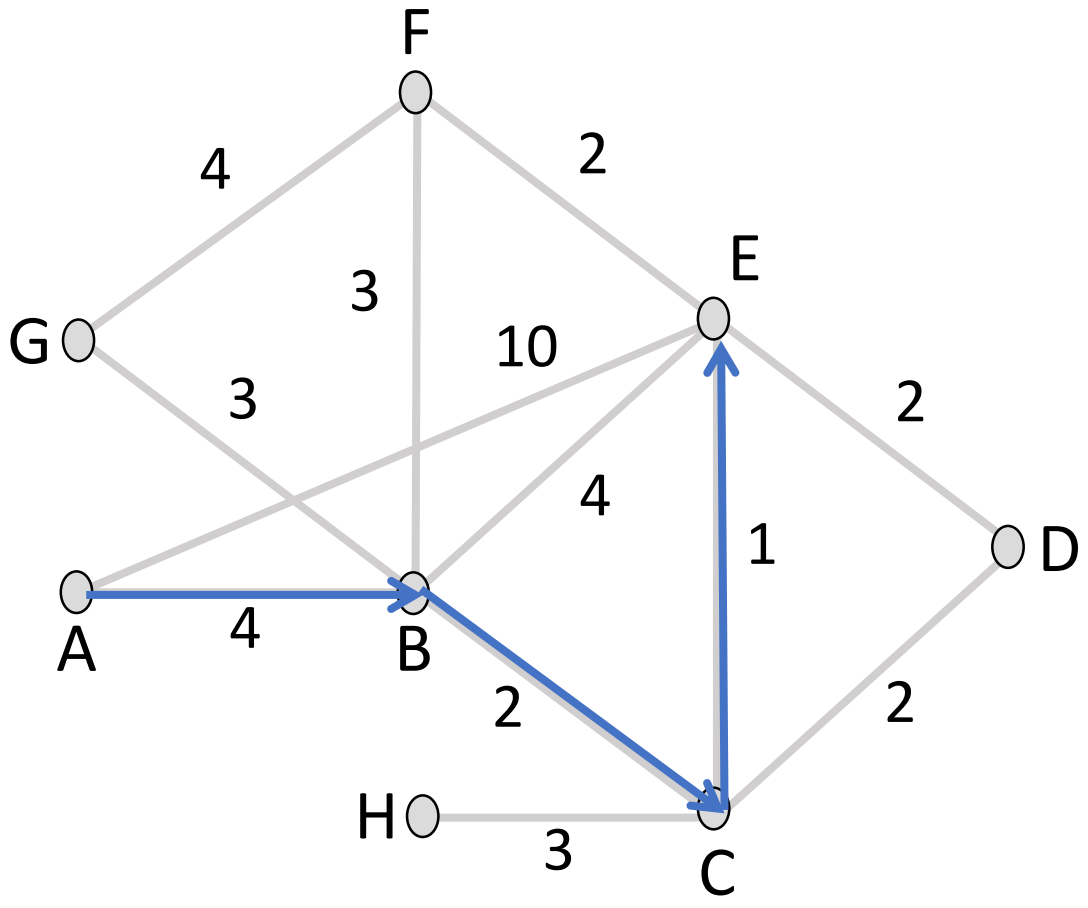
- This is less than:
  - dist(ABE) = 8
  - dist(ABFE) = 9
  - dist(AE) = 10
  - dist(ABCDE) = 10



Shortest Paths (4)

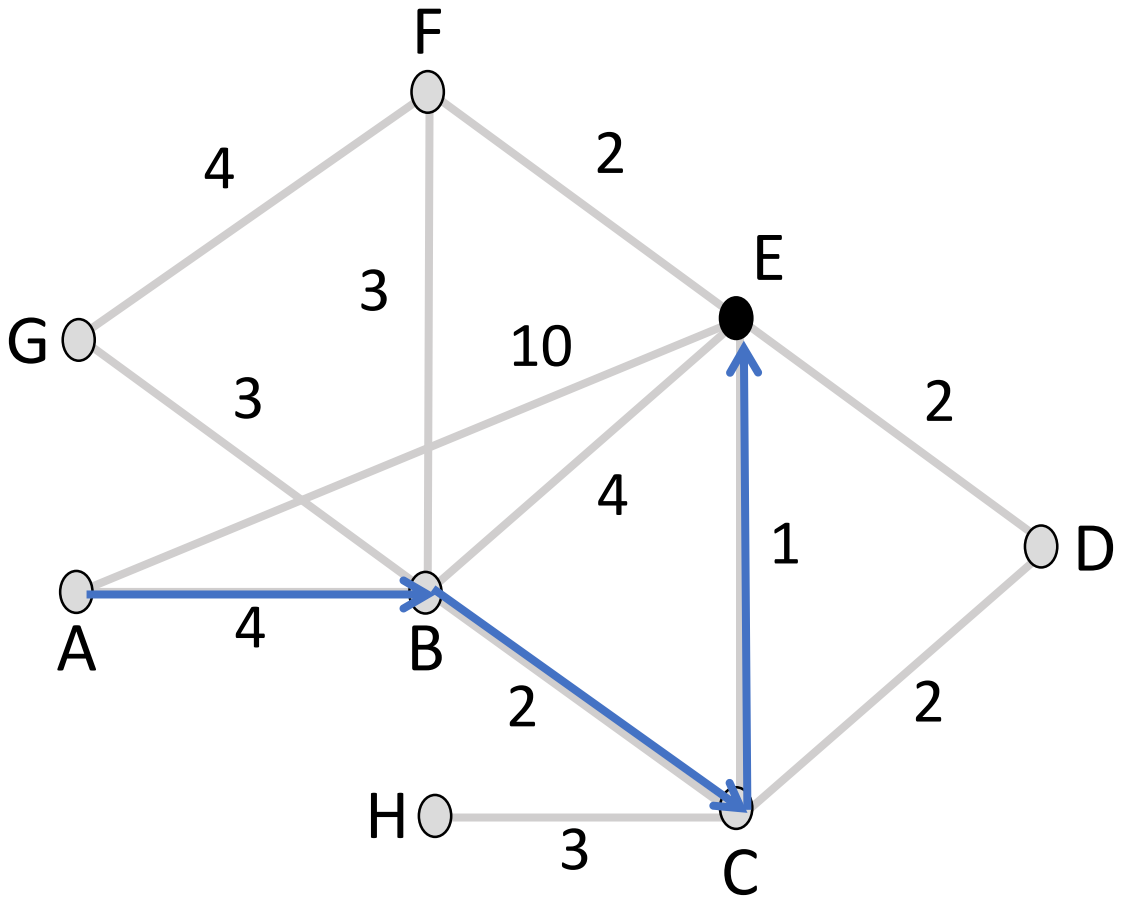
- Optimality property:
  - Subpaths of shortest paths are also shortest paths
- ABCE is a shortest path

  →So are ABC, AB, BCE, BC, CE



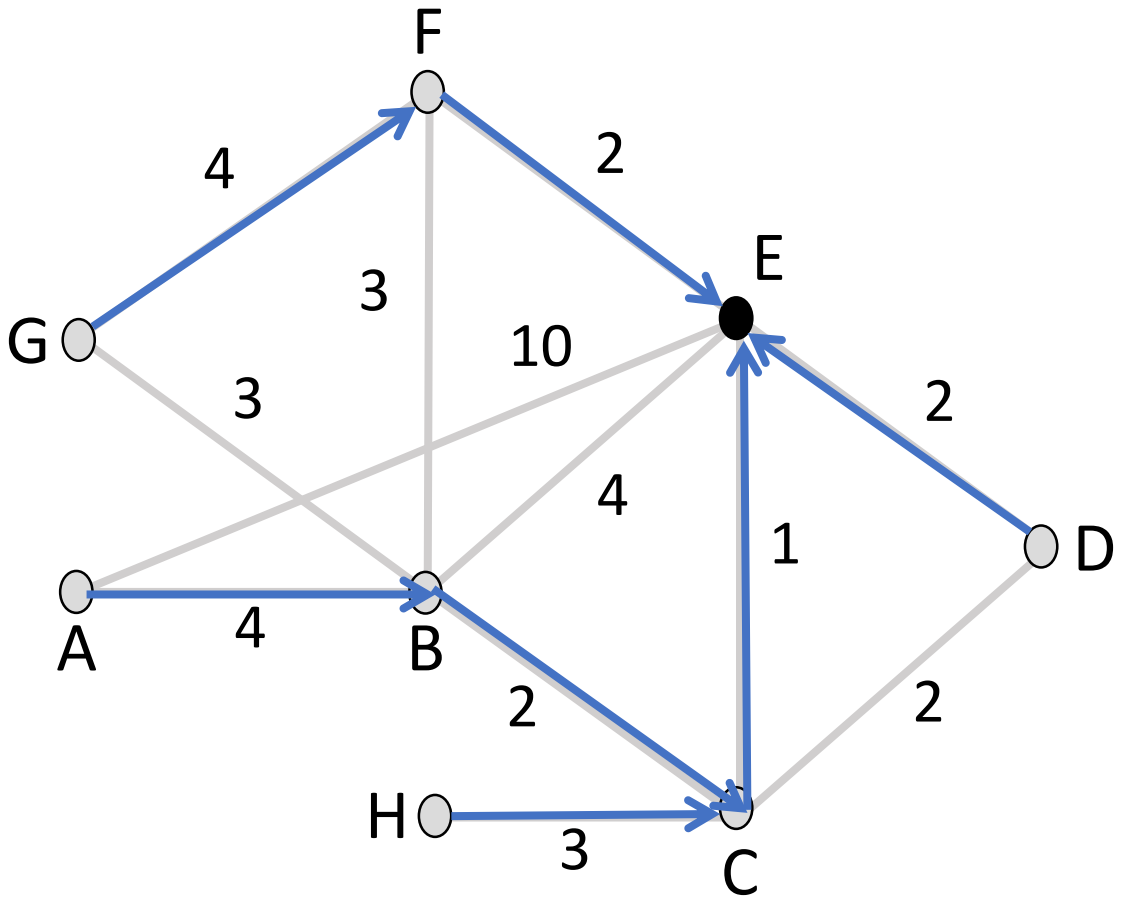
Sink Trees

- Sink tree for a destination is the union of all shortest paths towards the destination
  - Similarly source tree
- Find the sink tree for E



Sink Trees (2)

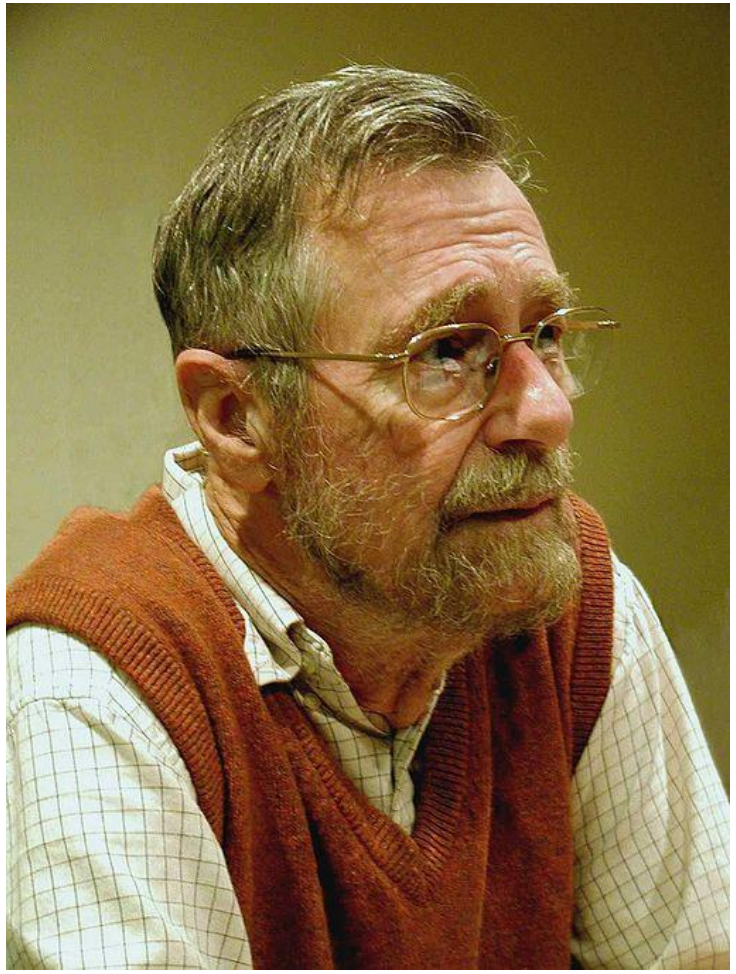
- Implications:
  - Only need to use destination to follow shortest paths
  - Each node only need to send to the next hop
- Forwarding table at a node
  - Lists next hop for each destination
  - Routing table may know more



Dijkstra’s Algorithm

Edsger W. Dijkstra (1930-2002)

- Famous computer scientist
  - Programming languages
  - Distributed algorithms
  - Program verification
- Dijkstra's algorithm, 1969
  - Single-source shortest paths, given network with non-negative link costs



By Hamilton Richards, CC-BY-SA-3.0, via Wikimedia Commons

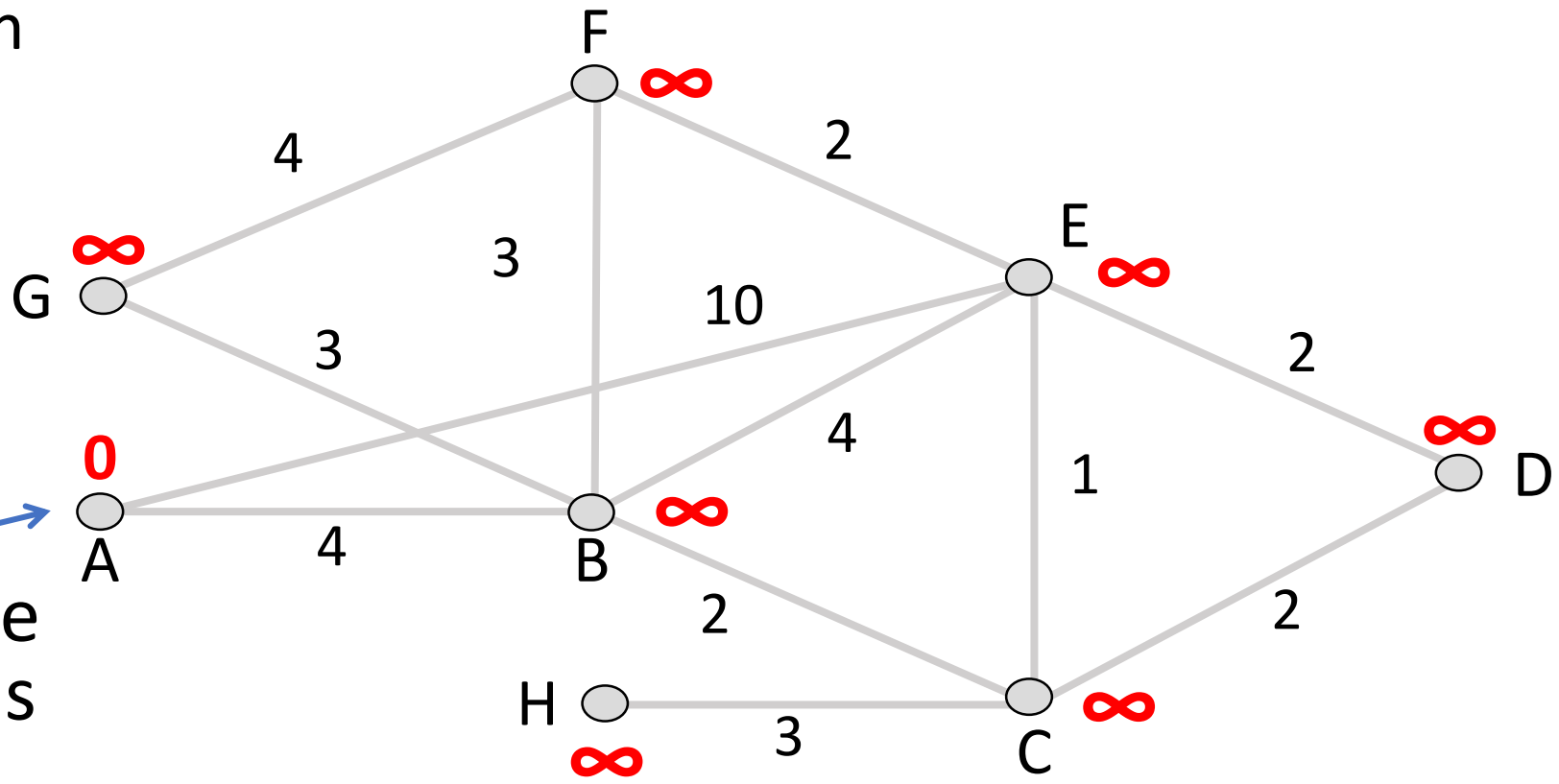
Dijkstra's Algorithm

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, a node 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's Algorithm (2)

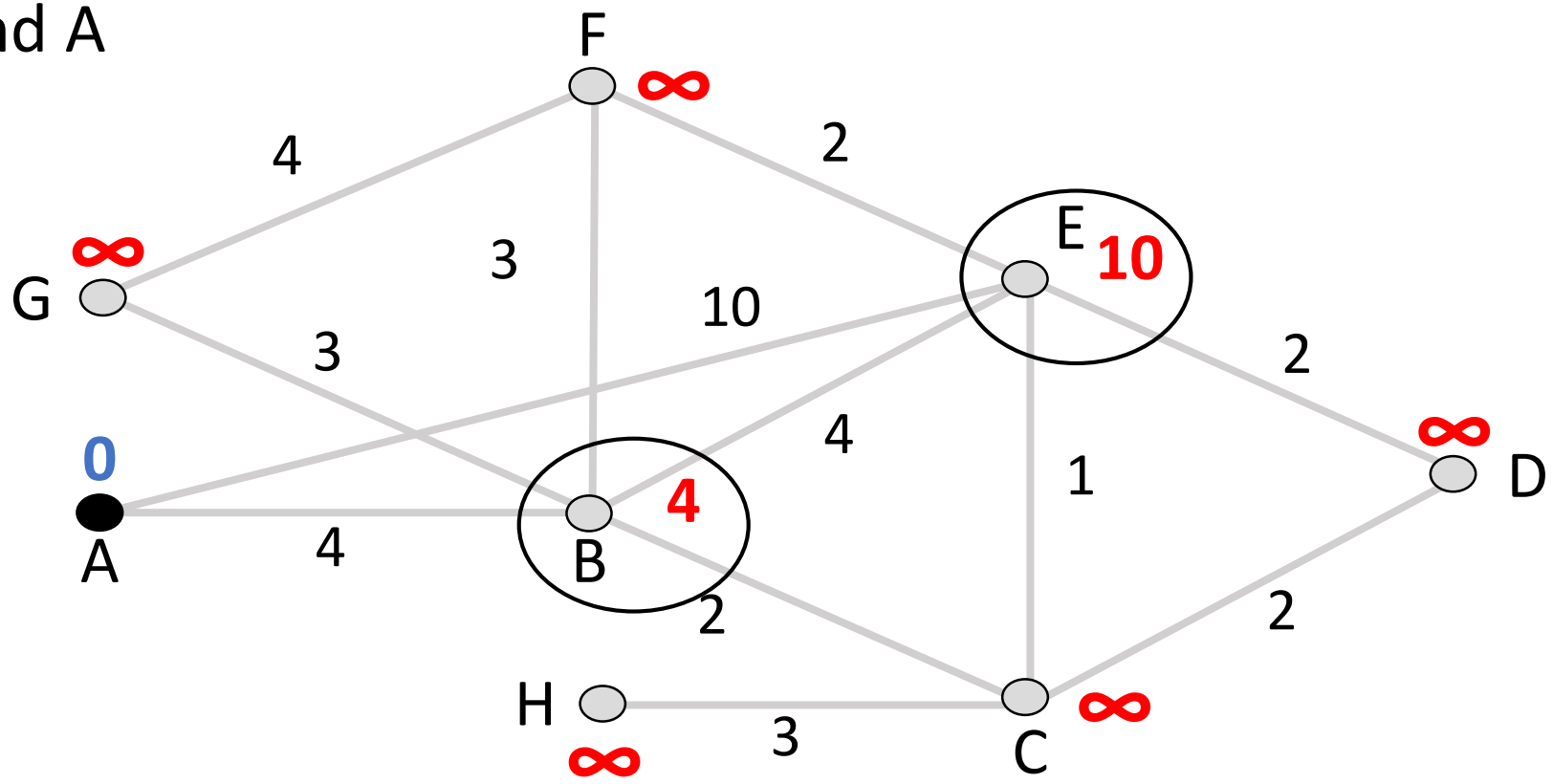
- Initialization



We'll compute shortest paths from A

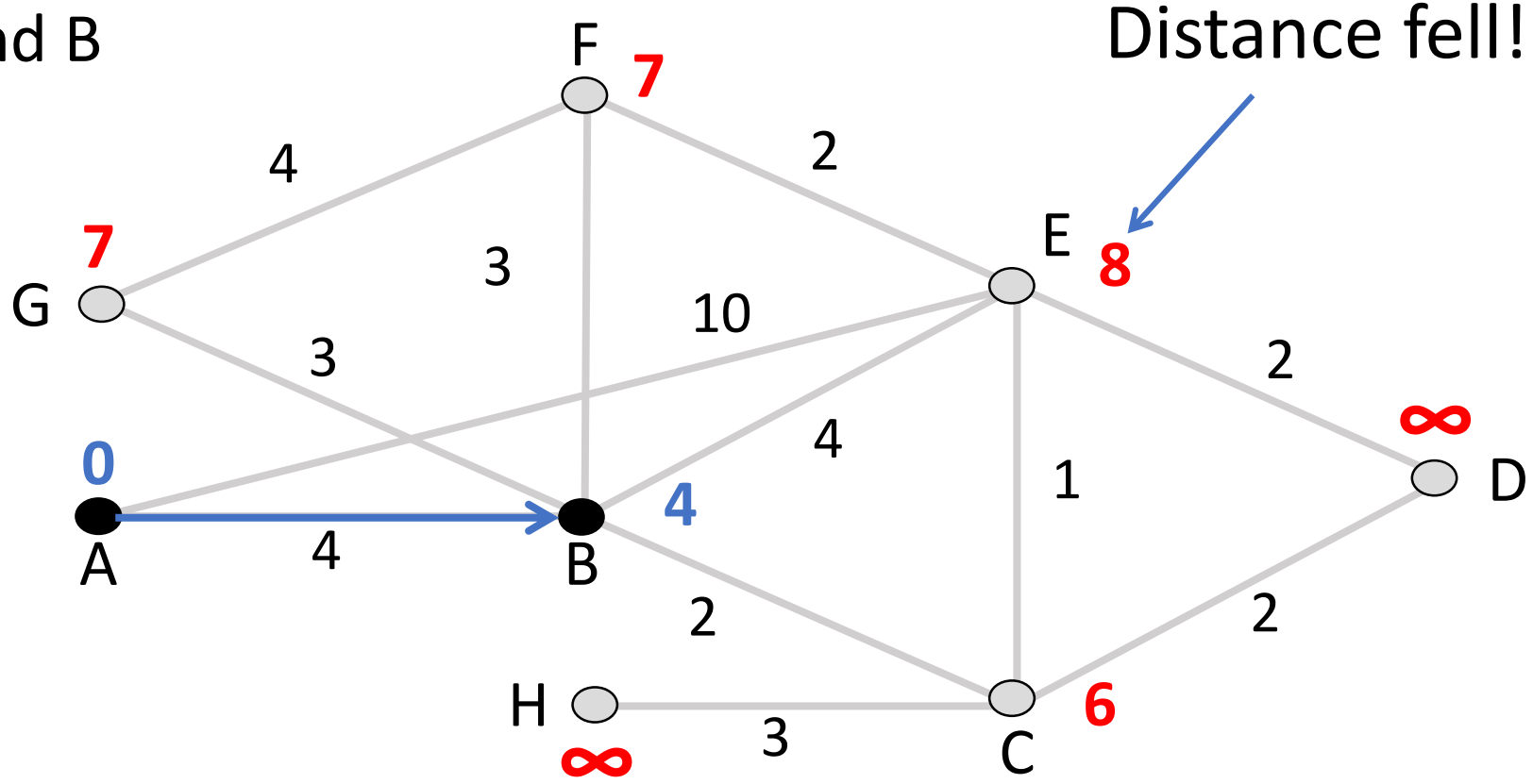
Dijkstra's Algorithm (3)

- Relax around A



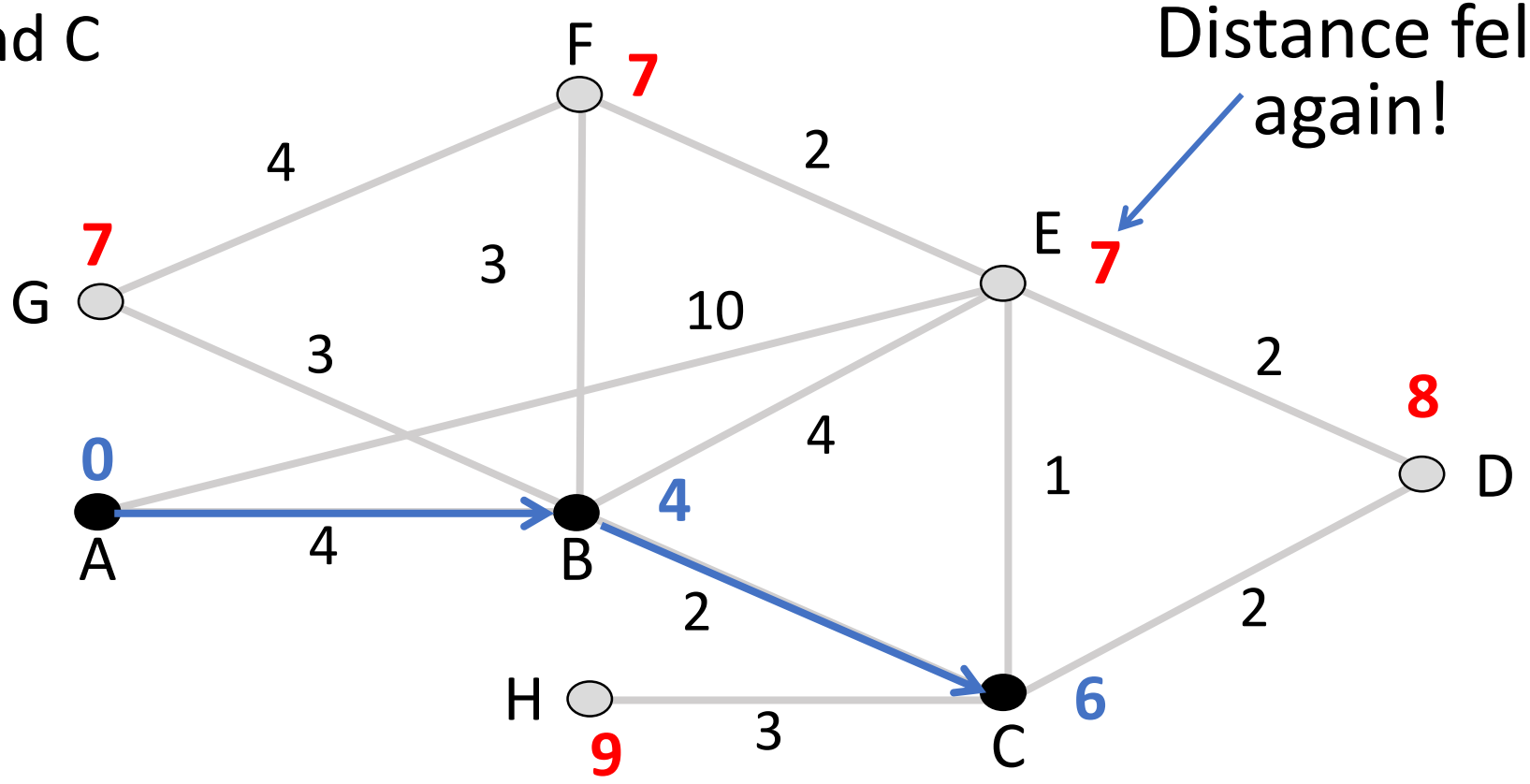
Dijkstra's Algorithm (4)

- Relax around B



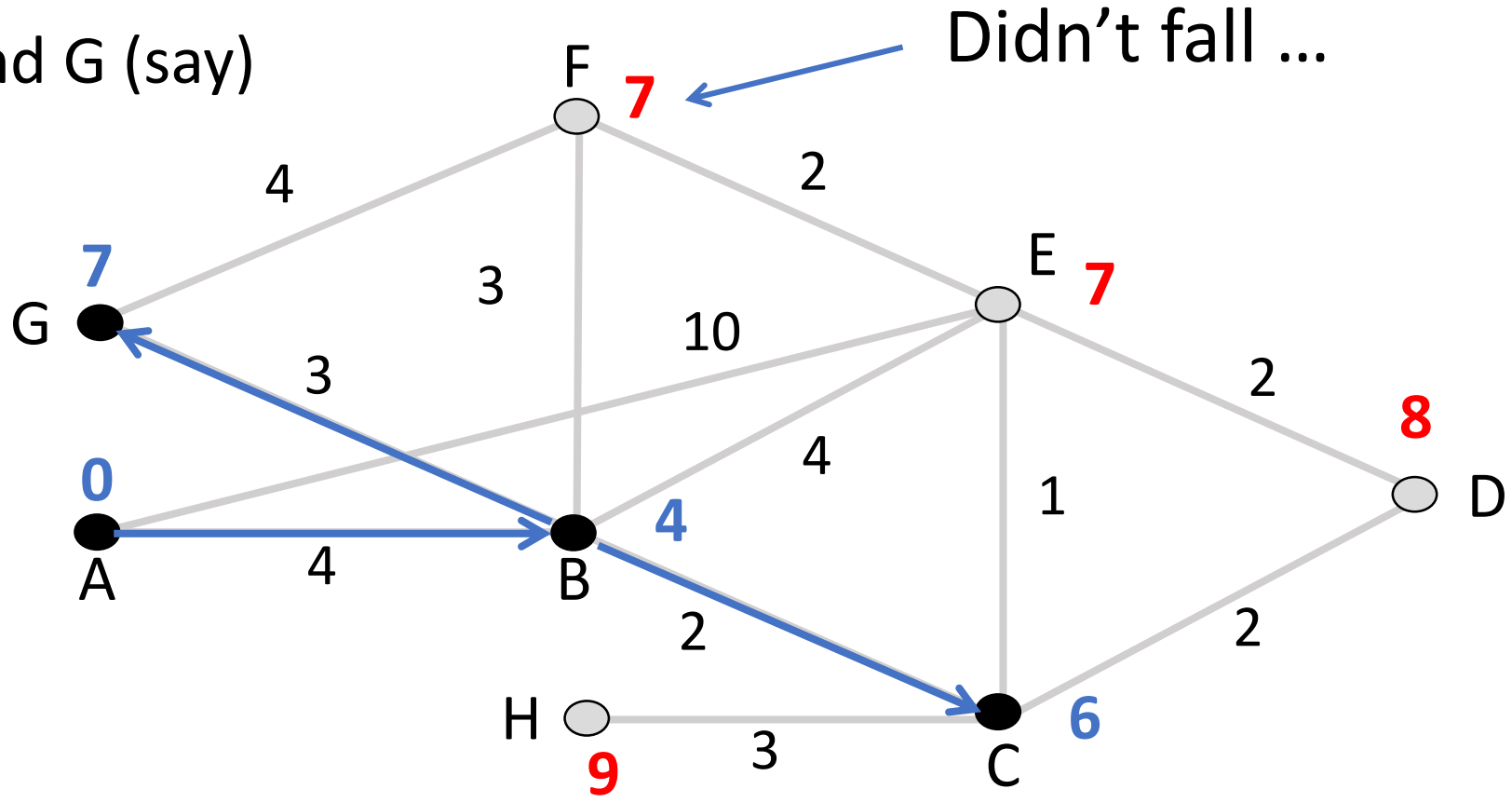
Dijkstra's Algorithm (5)

- Relax around C



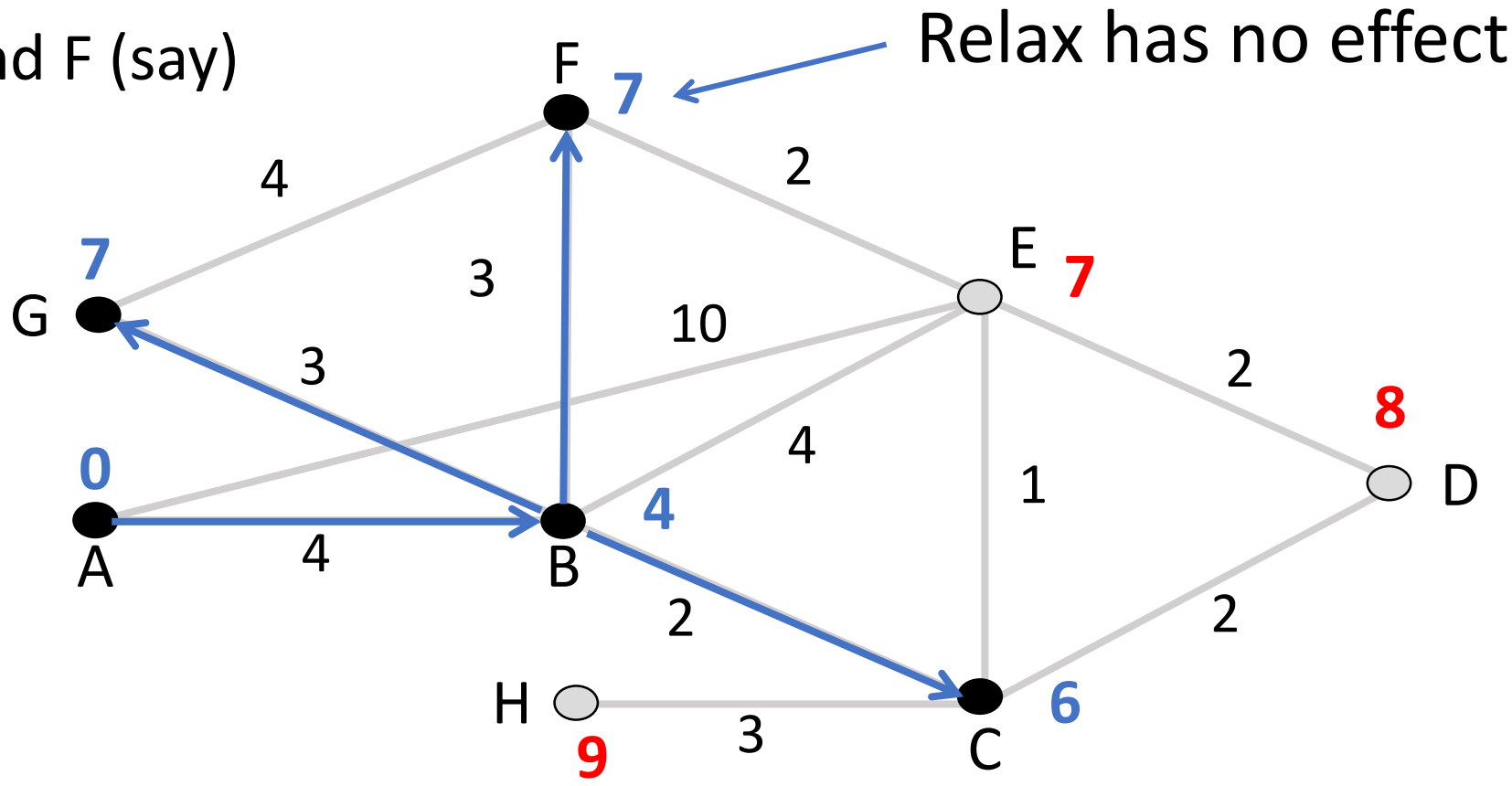
Dijkstra's Algorithm (6)

- Relax around G (say)



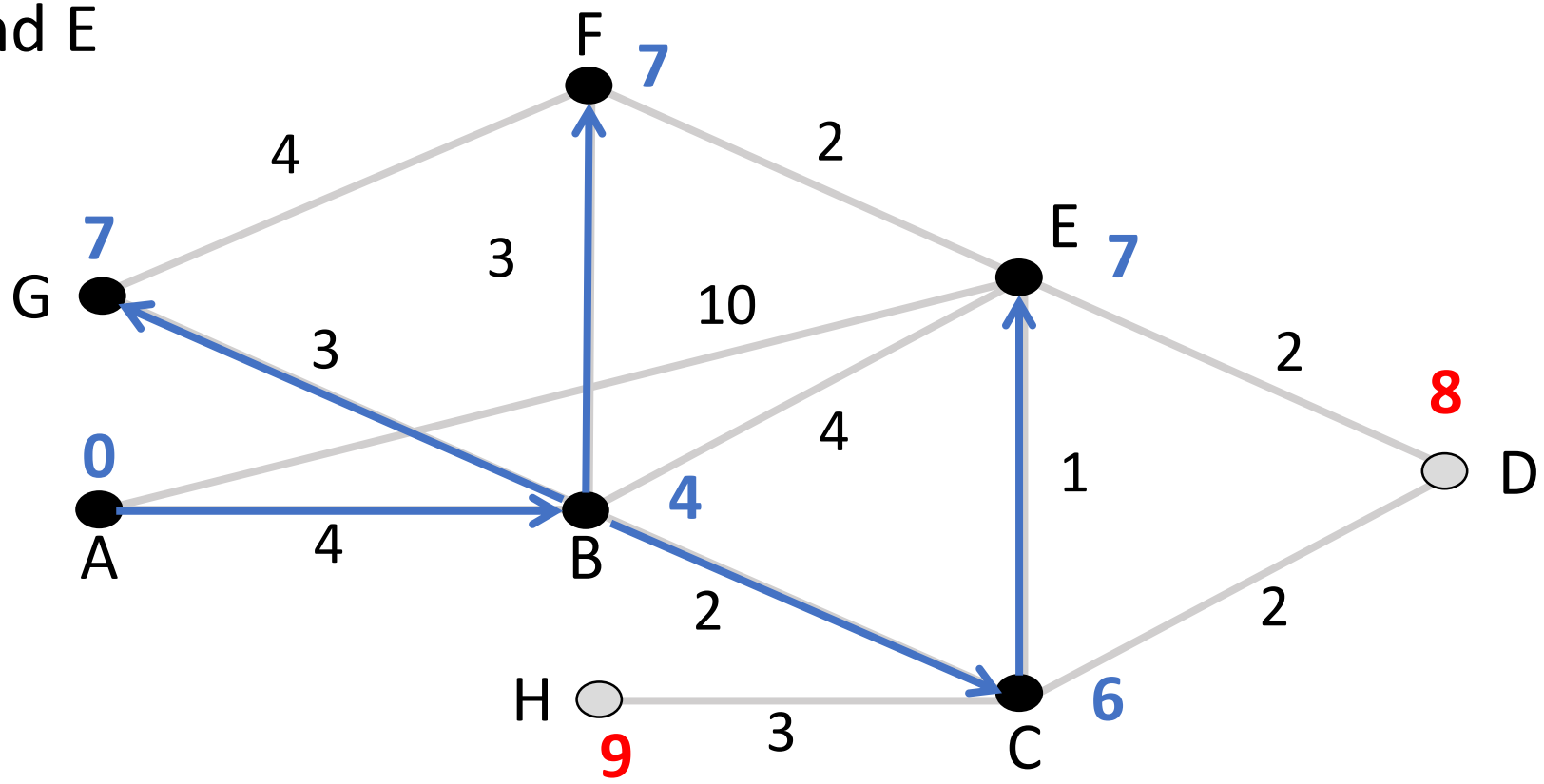
Dijkstra's Algorithm (7)

- Relax around F (say)



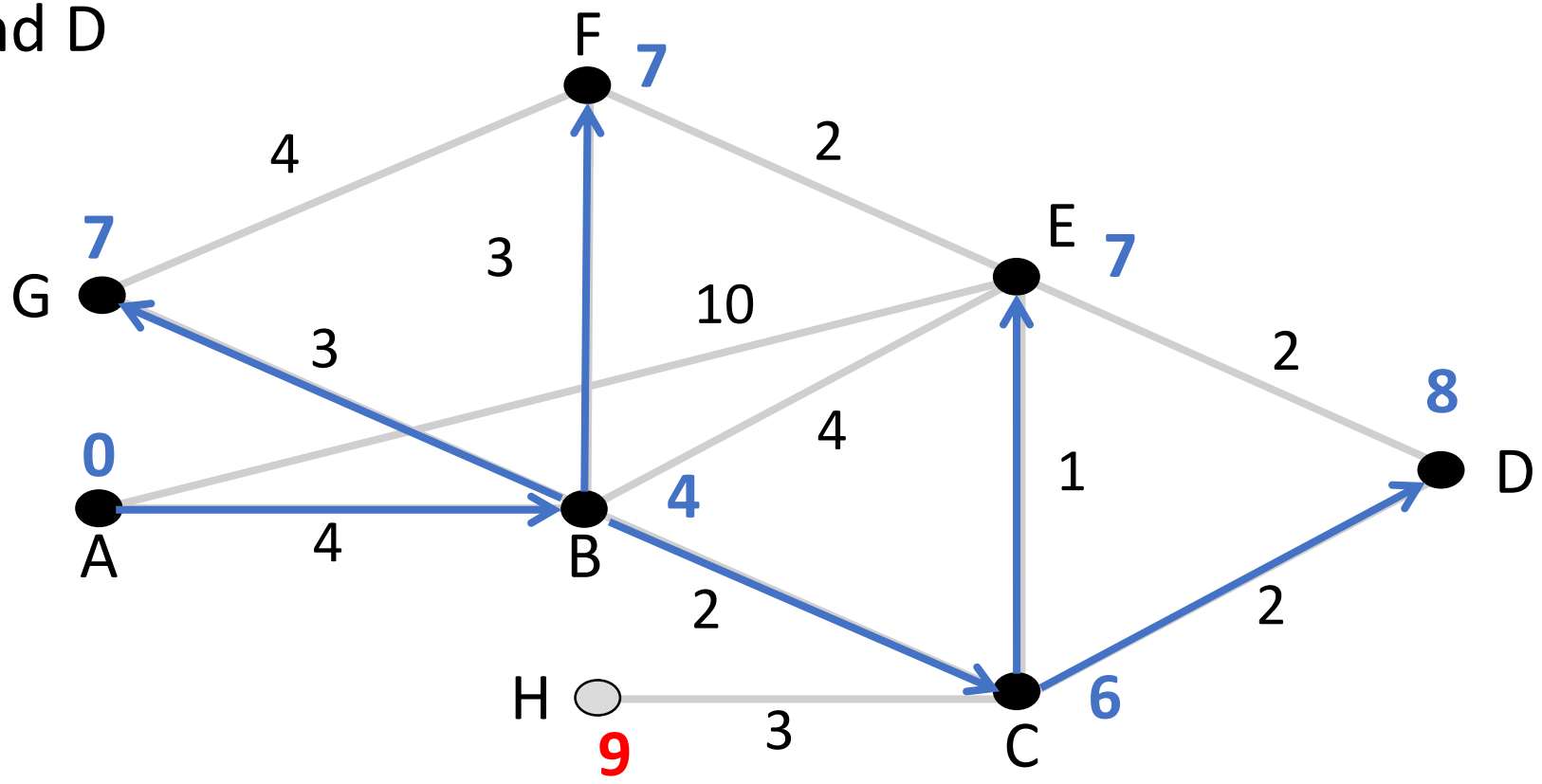
Dijkstra's Algorithm (8)

- Relax around E



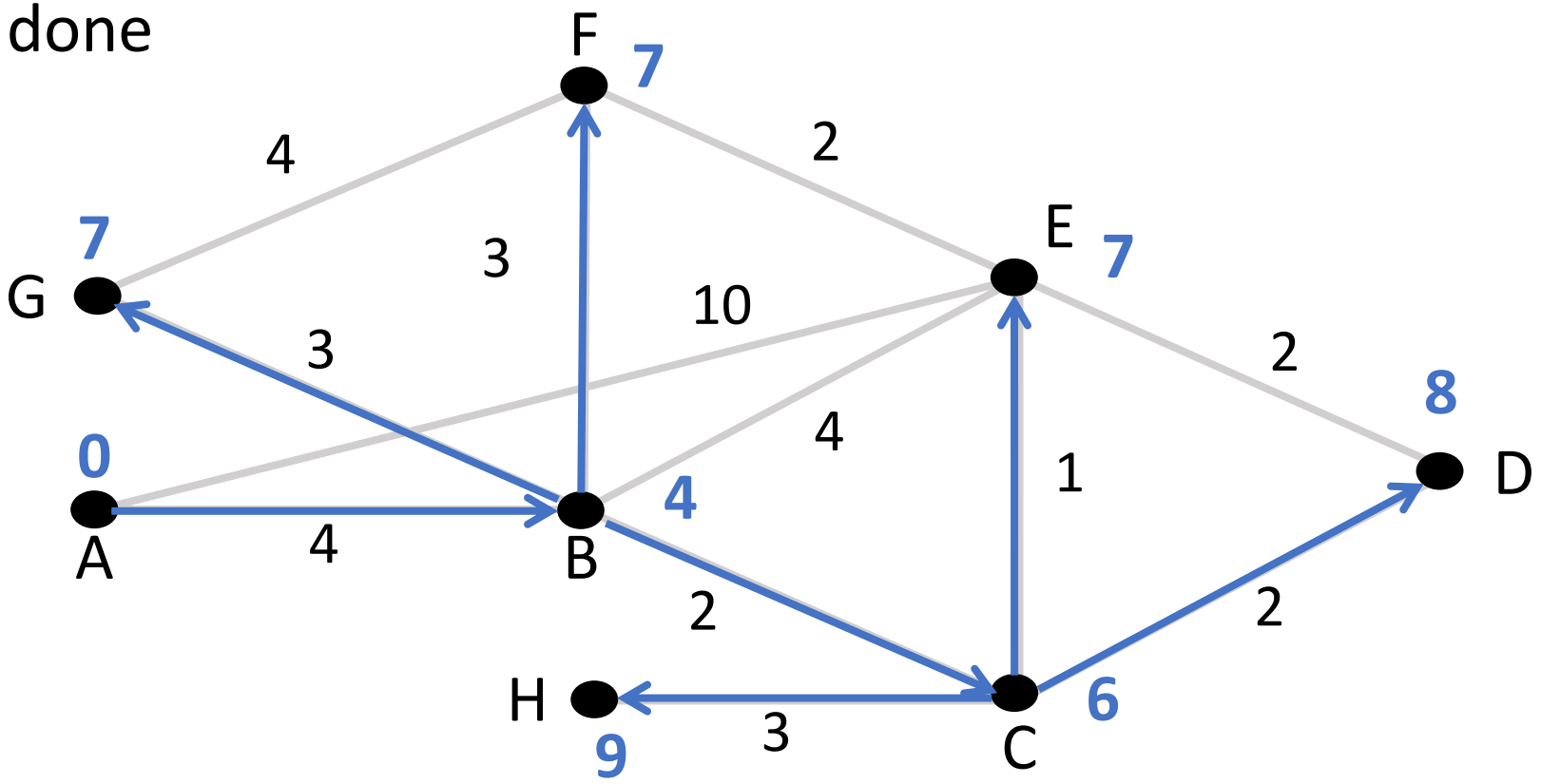
Dijkstra's Algorithm (9)

- Relax around D



Dijkstra's Algorithm (10)

- Finally, H … done



Dijkstra Comments

- Finds shortest paths in order of increasing distance from source
  - Leverages optimality property
- Runtime depends on cost of extracting min-cost node
  - Superlinear in network size (grows fast)
- Gives complete source/sink tree
  - More than needed for forwarding!
  - 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 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 (and next hops) to all destinations

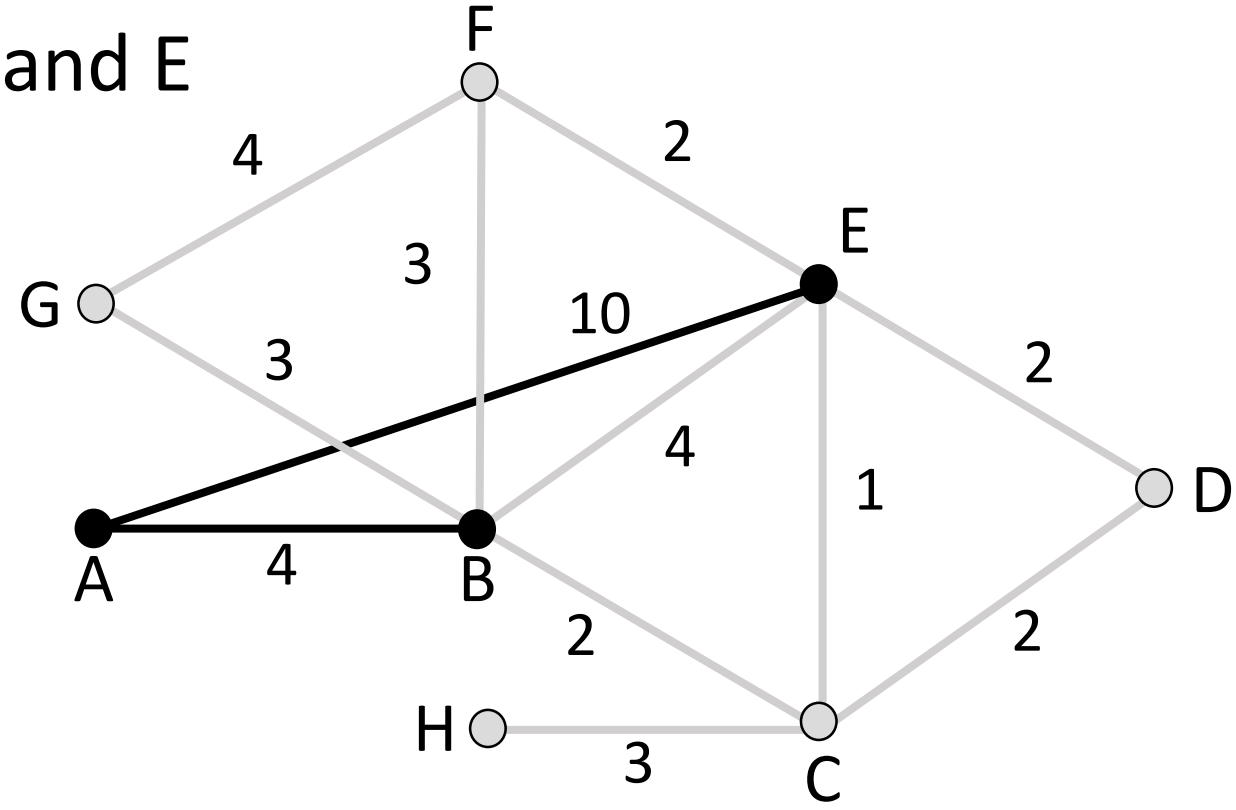
1. Initialize vector with 0 (zero) cost to self, ∞ (infinity) to other destinations
2. Periodically send vector to neighbors
3. Update vector for each destination by selecting the shortest distance heard, after adding cost of neighbor link
4. Use the best neighbor for forwarding

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	∞	∞

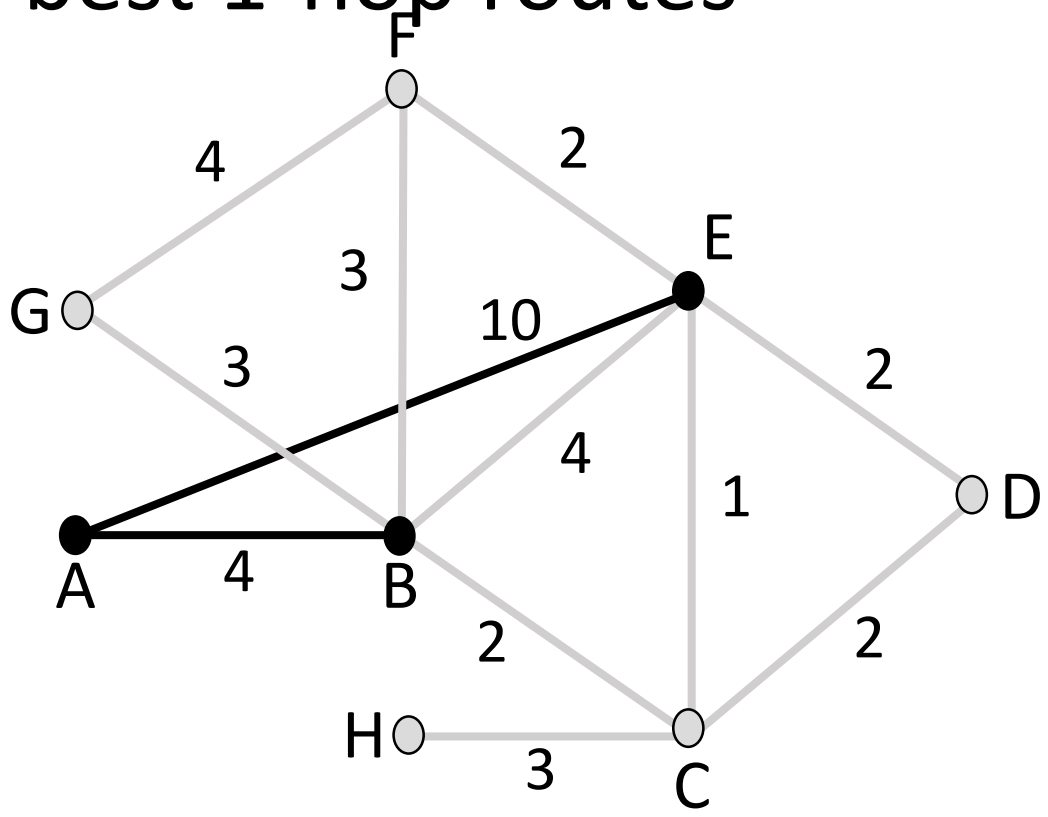
→

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

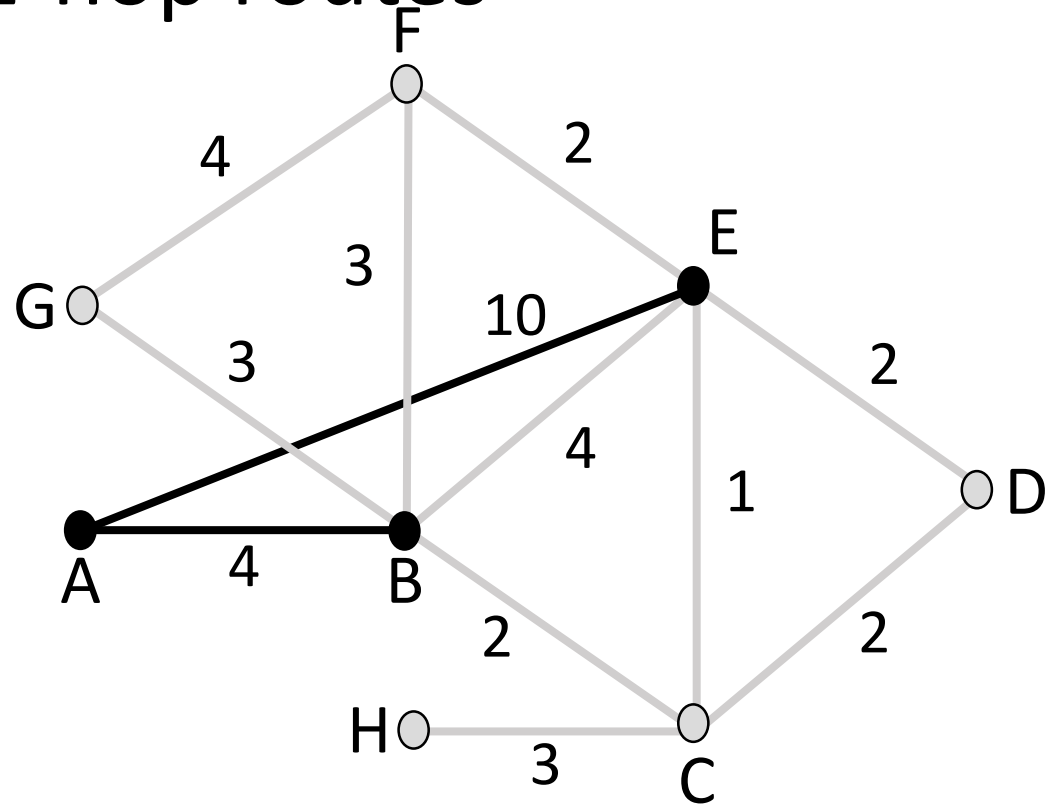
To	B says	E says
A	4	10
B	0	4
C	2	1
D	∞	2
E	4	0
F	3	2
G	3	∞
H	∞	∞

→

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

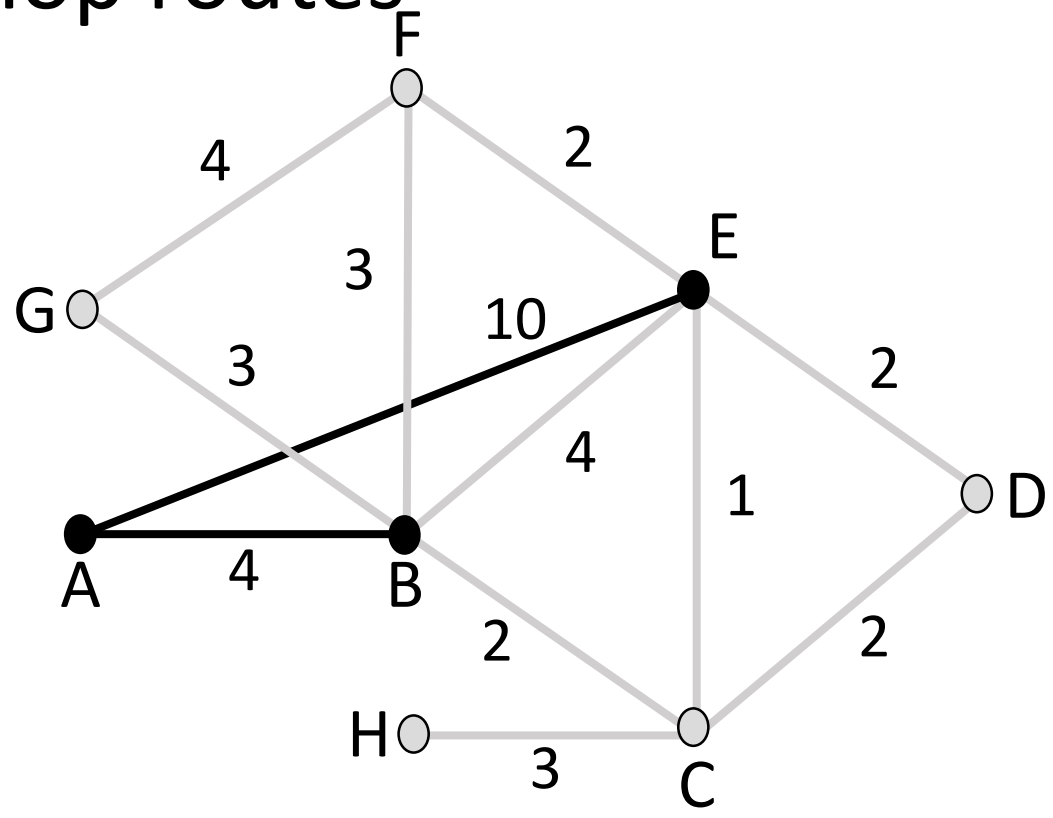
To	B says	E says
A	4	8
B	0	3
C	2	1
D	4	2
E	3	0
F	3	2
G	3	6
H	5	4

→

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

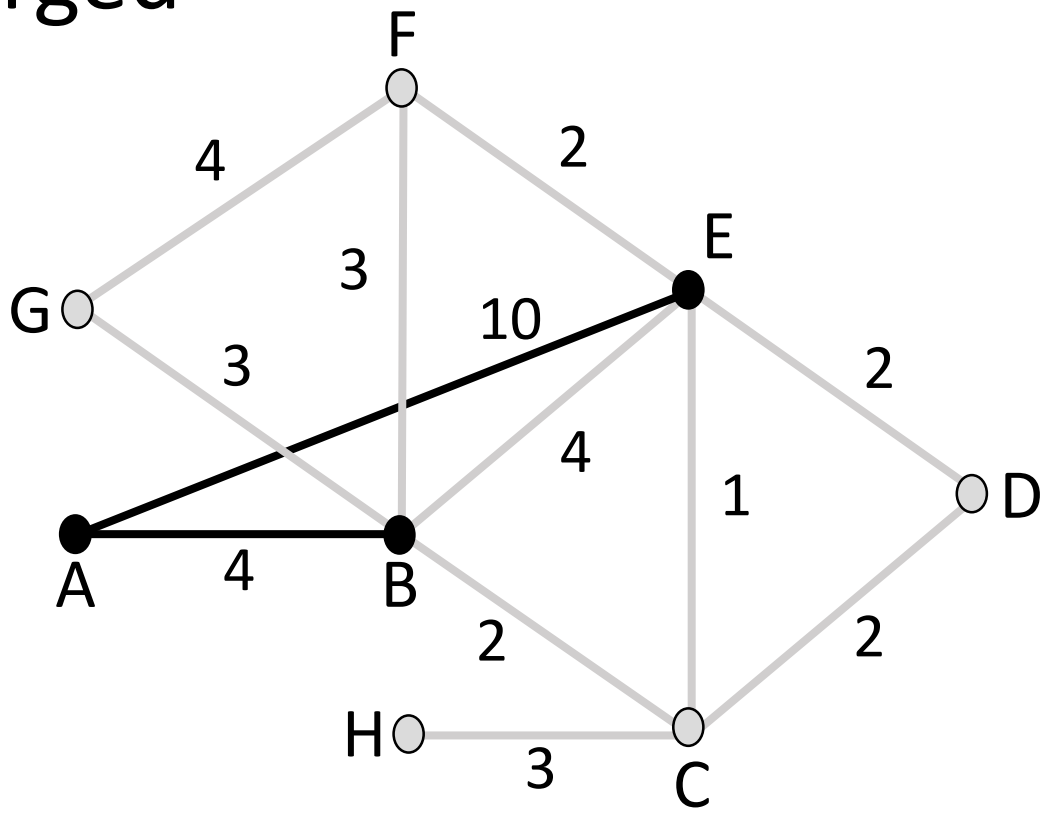
To	B says	E says
A	4	7
B	0	3
C	2	1
D	4	2
E	3	0
F	3	2
G	3	6
H	5	4

→

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

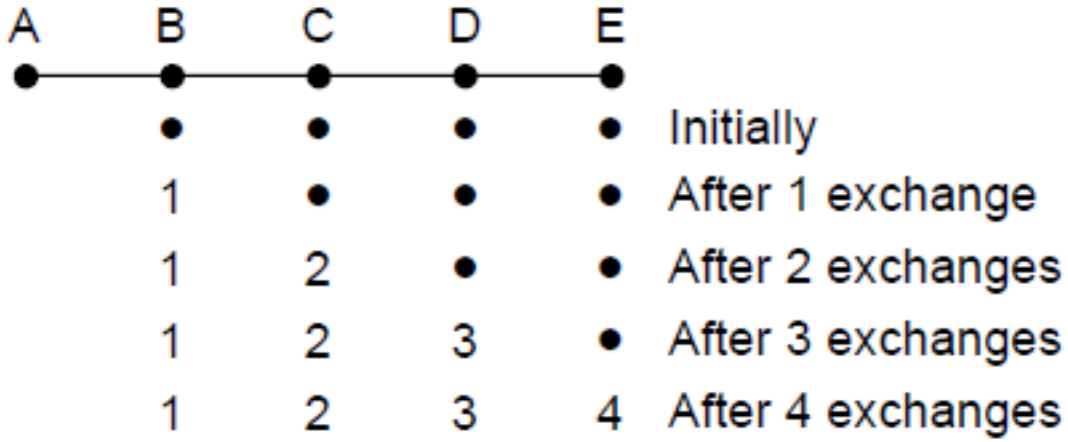


Distance Vector Dynamics

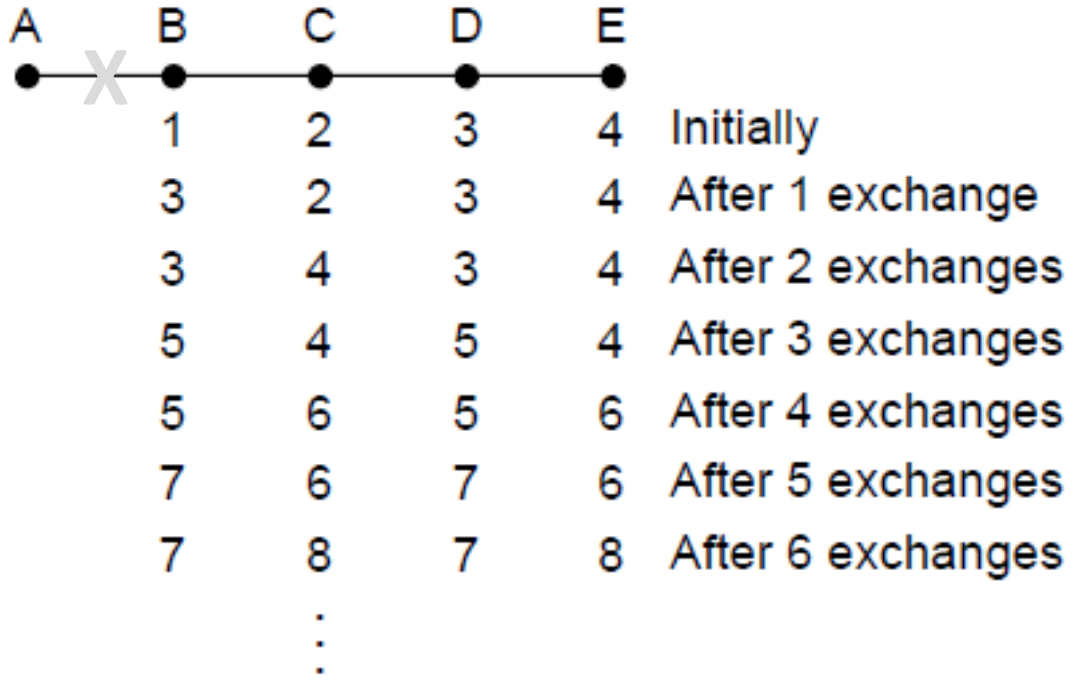
- Adding routes:
  - News travels one hop per exchange
- Removing routes:
  - When a node fails, no more exchanges, other nodes forget
- But partitions (unreachable nodes in divided network) are a problem
  - "Count to infinity" scenario

DV Dynamics (2)

- Good news travels quickly, bad news slowly



Desired convergence



"Count to infinity" scenario

DV 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

RIP (Routing Information Protocol)

- DV protocol with hop count as metric
  - Infinity is 16 hops; limits network size
  - Includes split horizon, poison reverse
- Routers send vectors every 30 seconds
  - Runs on top of UDP
  - Time-out in 180 secs to detect failures
- RIPv1 specified in RFC1058 (1988)

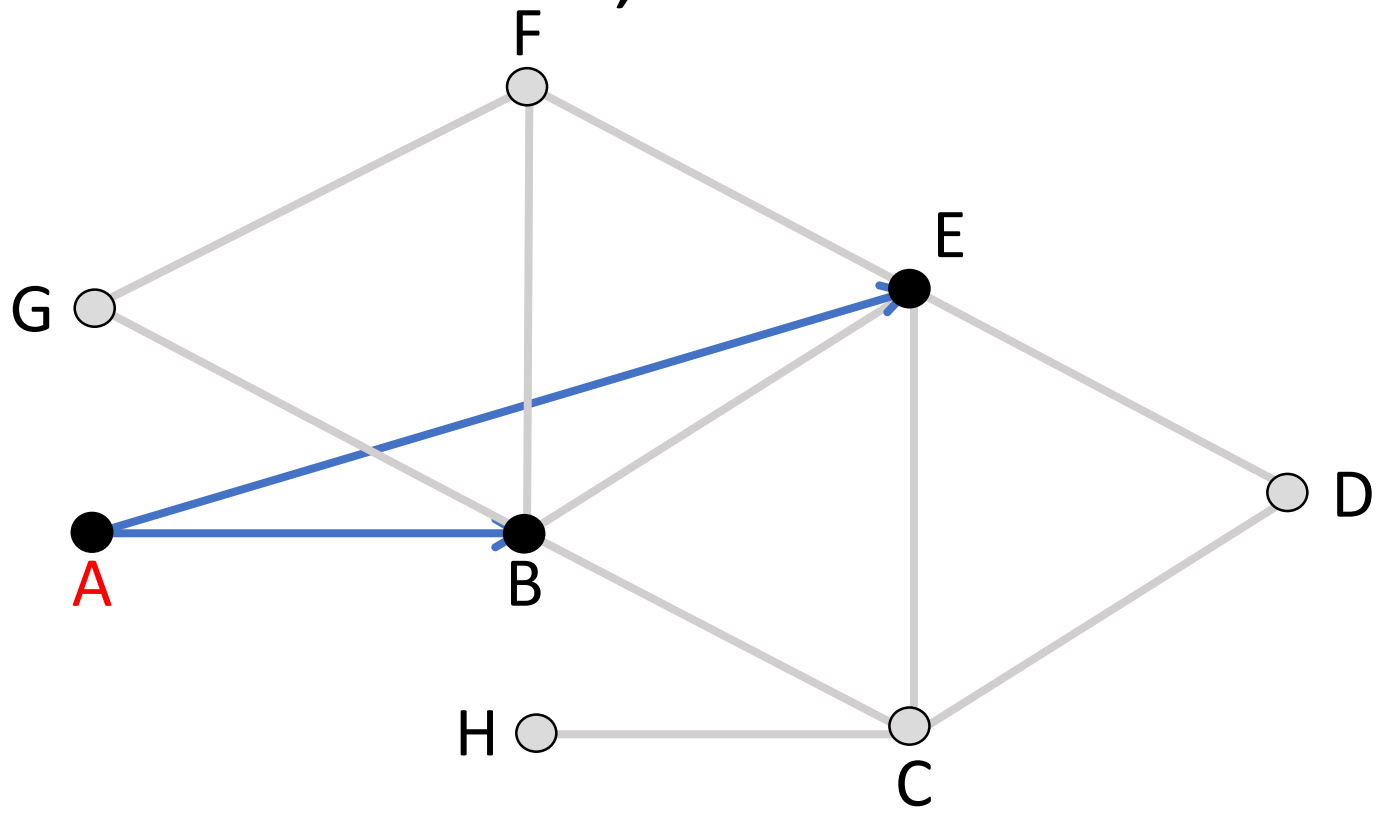
Flood Routing

Flooding

- Rule used at each node:
  - Sends an incoming message on to all other neighbors
  - Remember the message so that it is only flood once
- Inefficient because one node may receive multiple copies of message

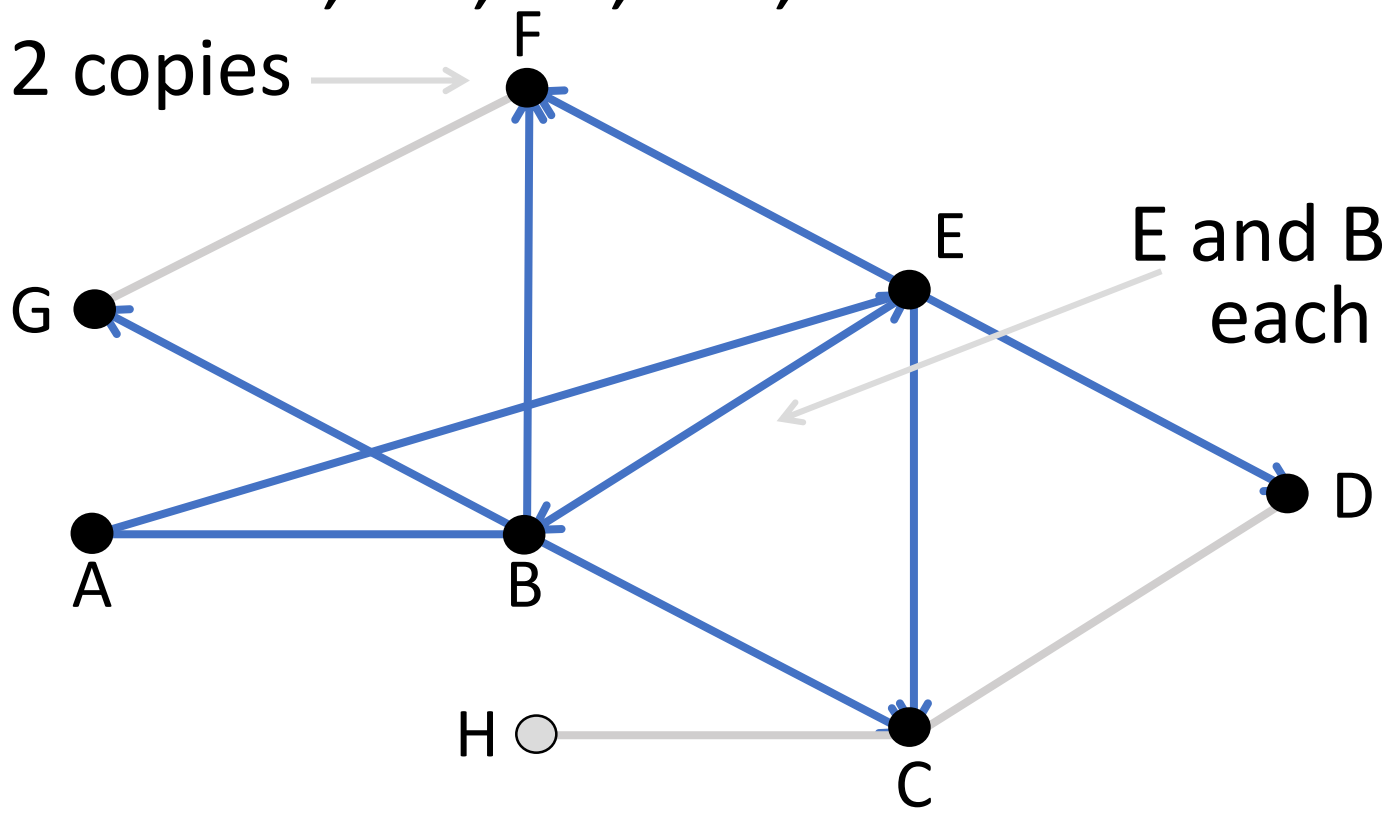
Flooding (2)

- Consider a flood from A; first reaches B via AB, E via AE



Flooding (3)

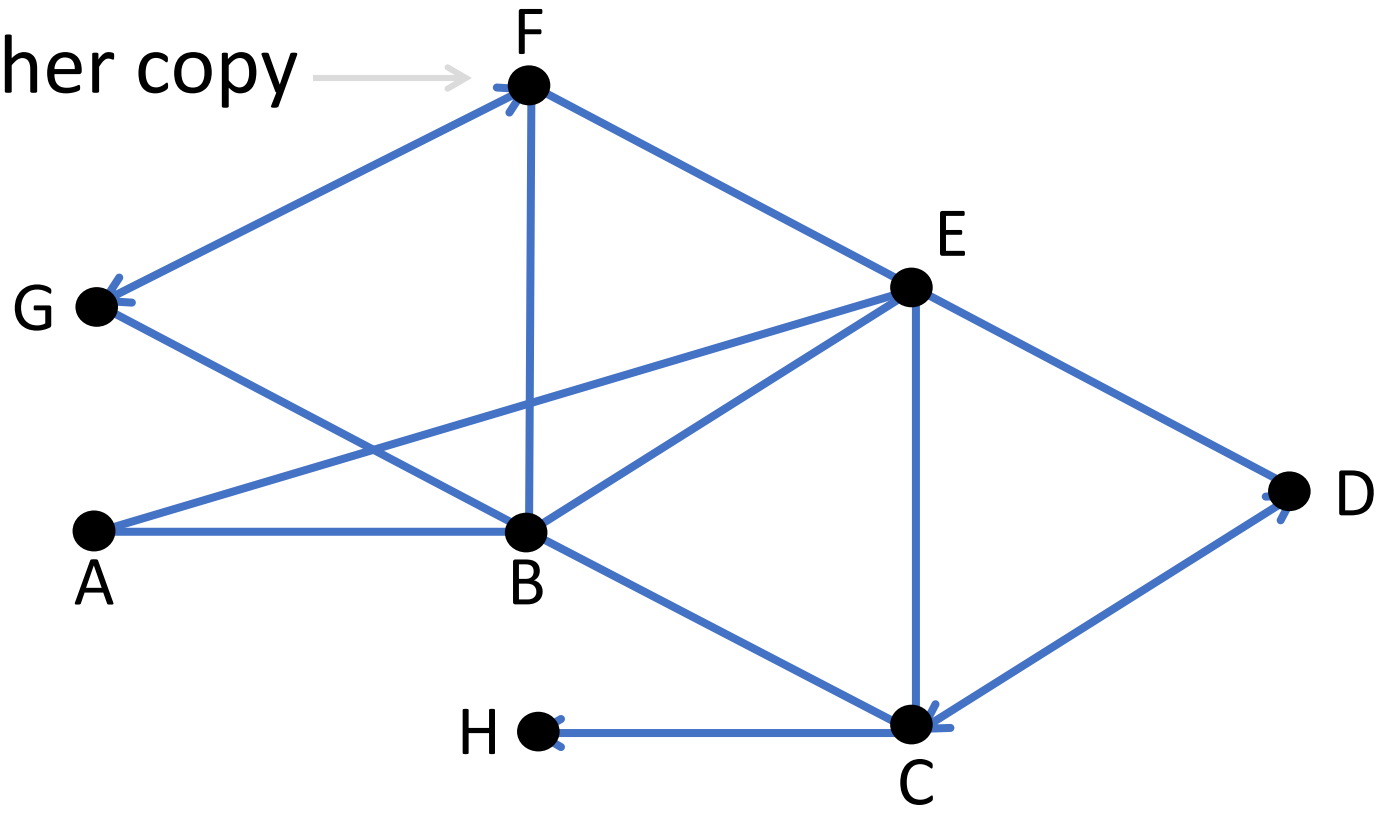
- Next B floods BC, BE, BF, BG, and E floods EB, EC, ED, EF F gets 2 copies



Flooding (4)

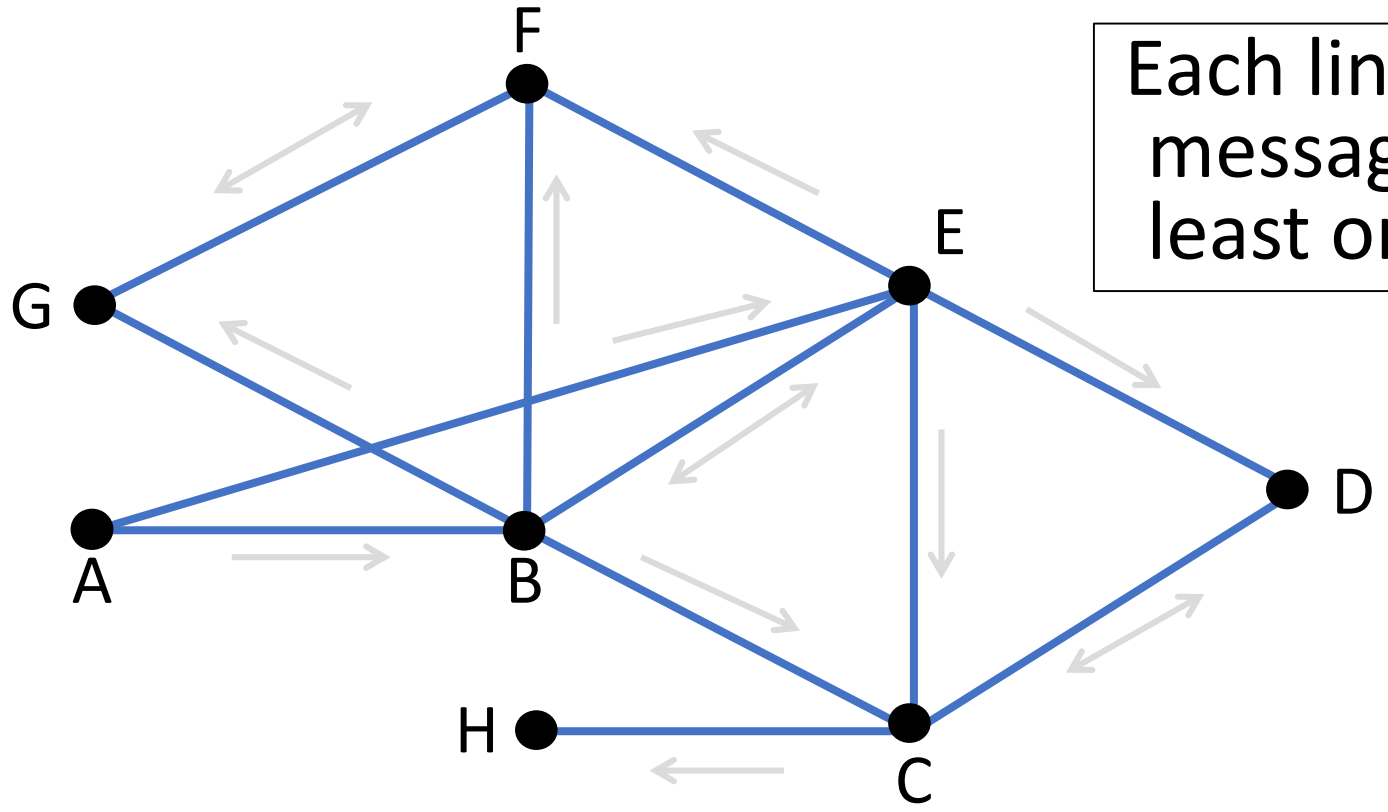
- C floods CD, CH; D floods DC; F floods FG; G floods GF

F gets another copy



Flooding (5)

- H has no-one to flood … and we're done



Each link carries the message, and in at least one direction

Flooding Details

- Remember message (to stop flood) using source and sequence number
  - So next message (with higher sequence) will go through
- To make flooding reliable, use ARQ
  - So receiver acknowledges, and sender resends if needed

Link-State Routing

Link-State Routing

- One of two approaches to routing
  - Trades more computation than distance vector for better dynamics
- Widely used in practice
  - Used in Internet/ARPANET from 1979
  - Modern networks use OSPF and IS-IS

Link-State Setting

Nodes compute their forwarding table in the same distributed setting as for distance vector:

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

Link-State Algorithm

Proceeds in two phases:

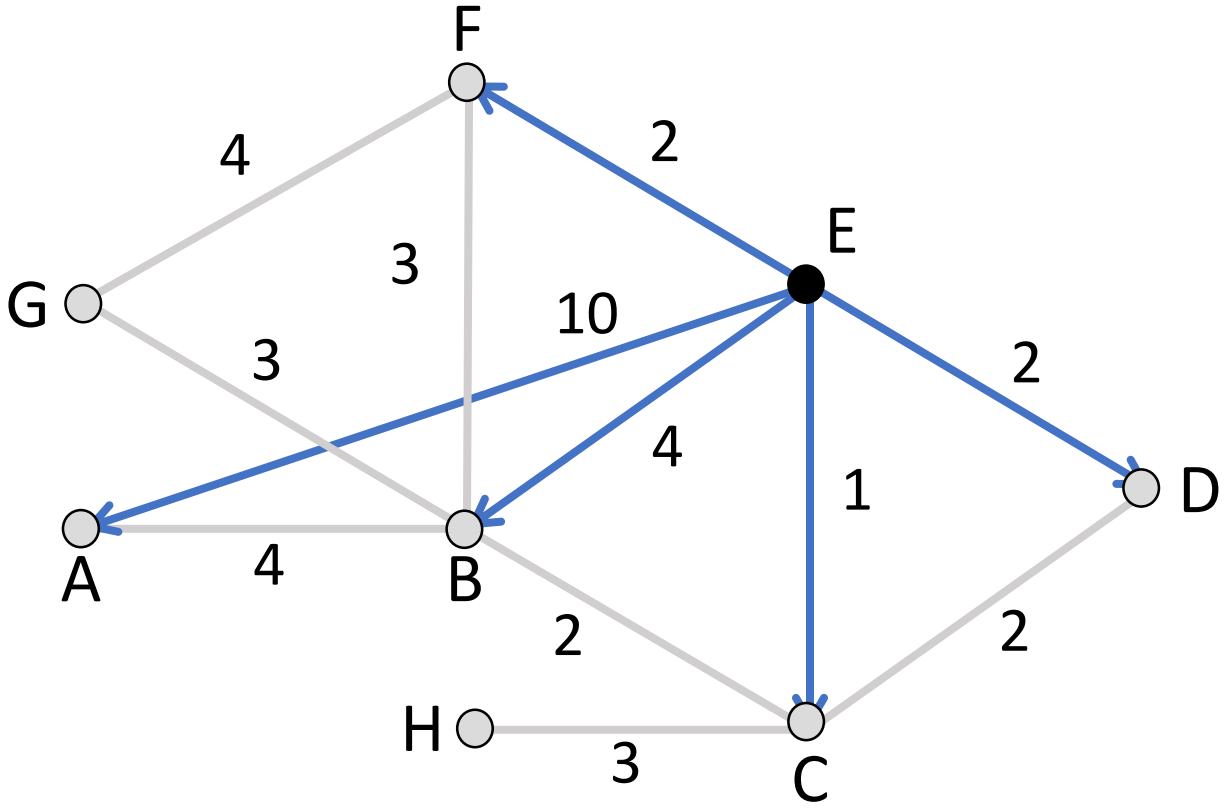
1. Nodes flood topology with link state packets
   - Each node learns full topology
2. Each node computes its own forwarding table
   - By running Dijkstra (or equivalent)

Phase 1: 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

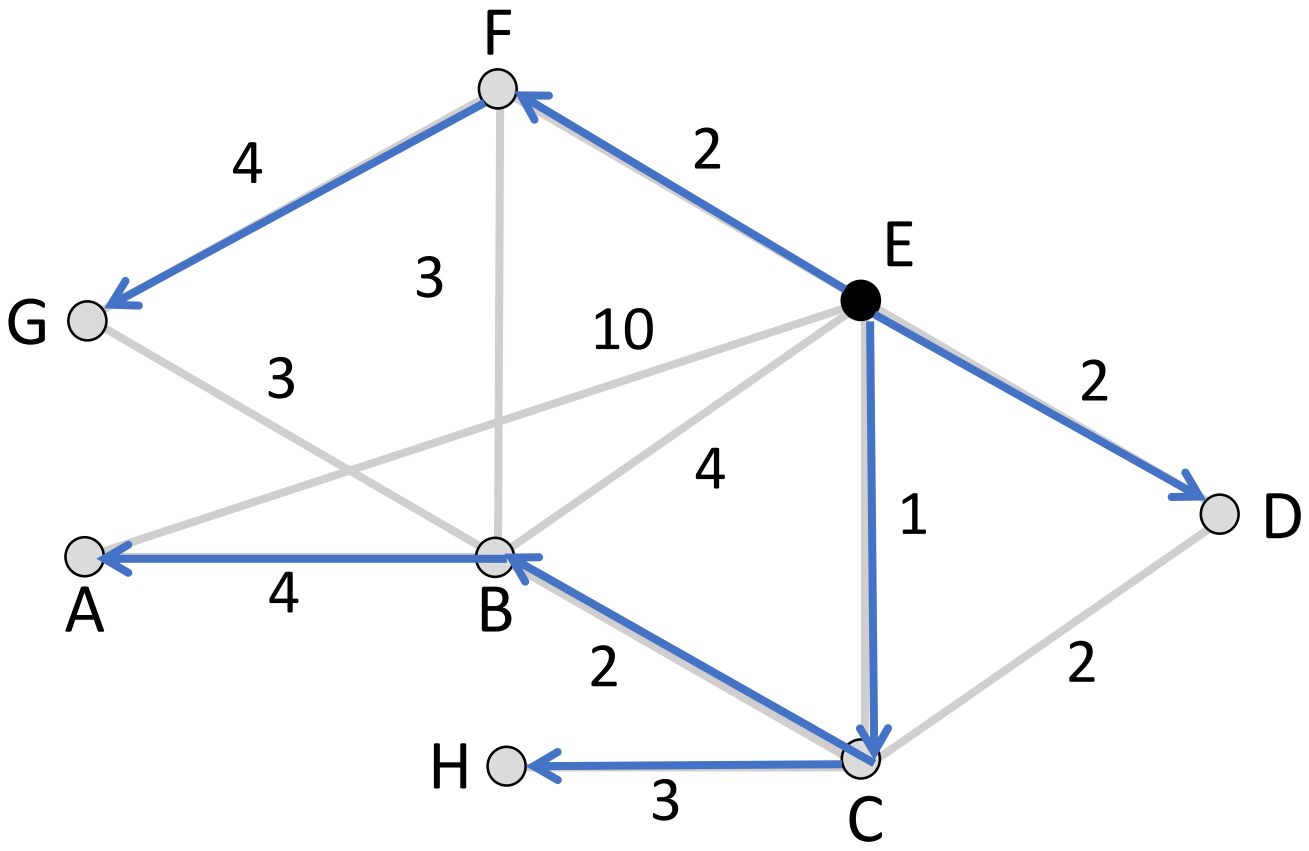


Phase 2: Route Computation

- Each node has full topology
  - By combining all LSPs
- Each node simply runs Dijkstra
  - Replicated computation, but finds required routes directly
  - Compile forwarding table from sink/source tree
  - That's it folks!

Forwarding Table

Source Tree for E (from Dijkstra)



E's Forwarding Table

To	Next
A	C
B	C
C	C
D	D
E	--
F	F
G	F
H	C

Handling Changes

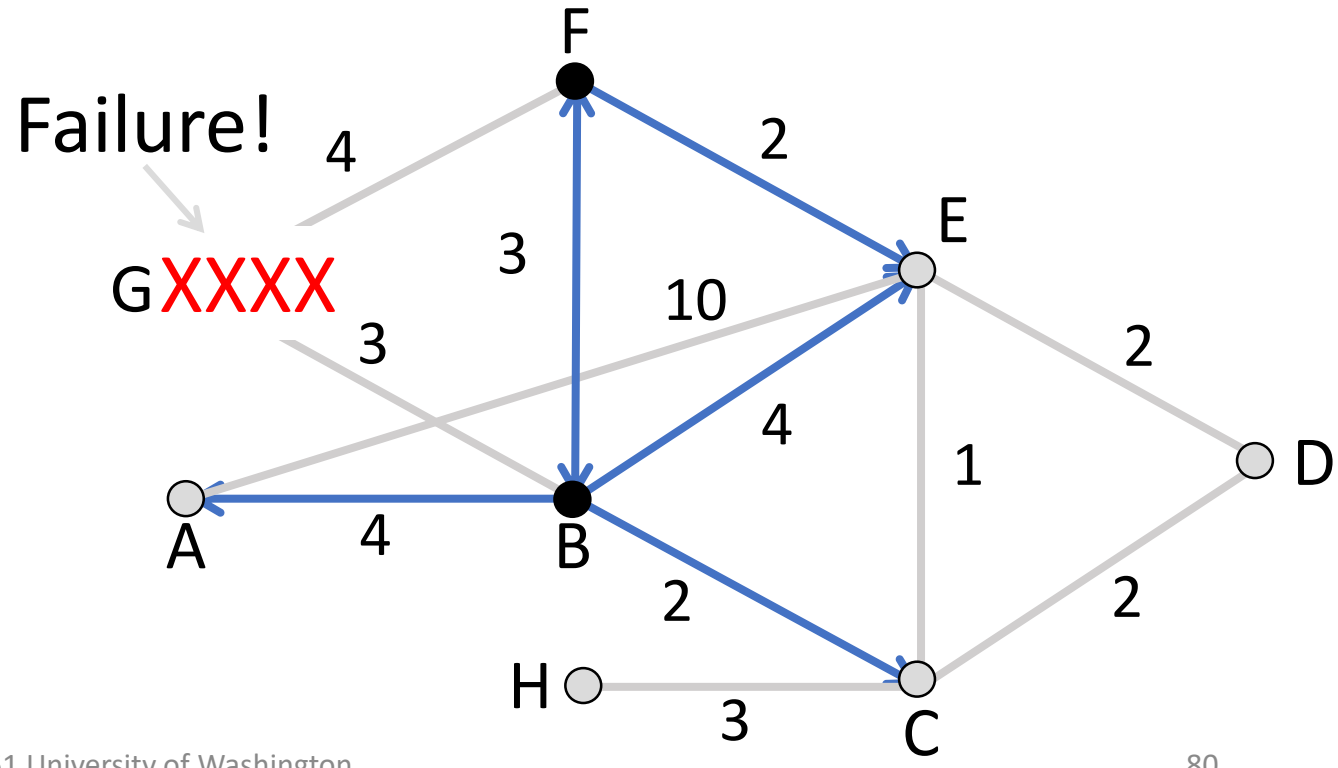
- On change, flood updated LSPs, re-compute routes
  - E.g., nodes adjacent to failed link or node initiate

B's LSP

Seq. #	
A	4
C	2
E	4
F	3
G	∞

F's LSP

Seq. #	
B	3
E	2
G	∞



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 …

Link-State Complications

- Things that can go wrong:
  - Seq. number reaches max, or is corrupted
  - 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

DV/LS Comparison

Goal	Distance Vector	Link-State
Correctness	Distributed Bellman-Ford	Replicated Dijkstra
Efficient paths	Approx. with shortest paths	Approx. with shortest paths
Fair paths	Approx. with shortest paths	Approx. with shortest paths
Fast convergence	Slow – many exchanges	Fast – flood and compute
Scalability	Excellent – storage/compute	Moderate – storage/compute

IS-IS and OSPF Protocols

- Widely used in large enterprise and ISP networks
  - IS-IS = Intermediate System to Intermediate System
  - OSPF = Open Shortest Path First
- Link-state protocol with many added features
  - E.g., “Areas” for scalability

Equal-Cost Multi-Path Routing

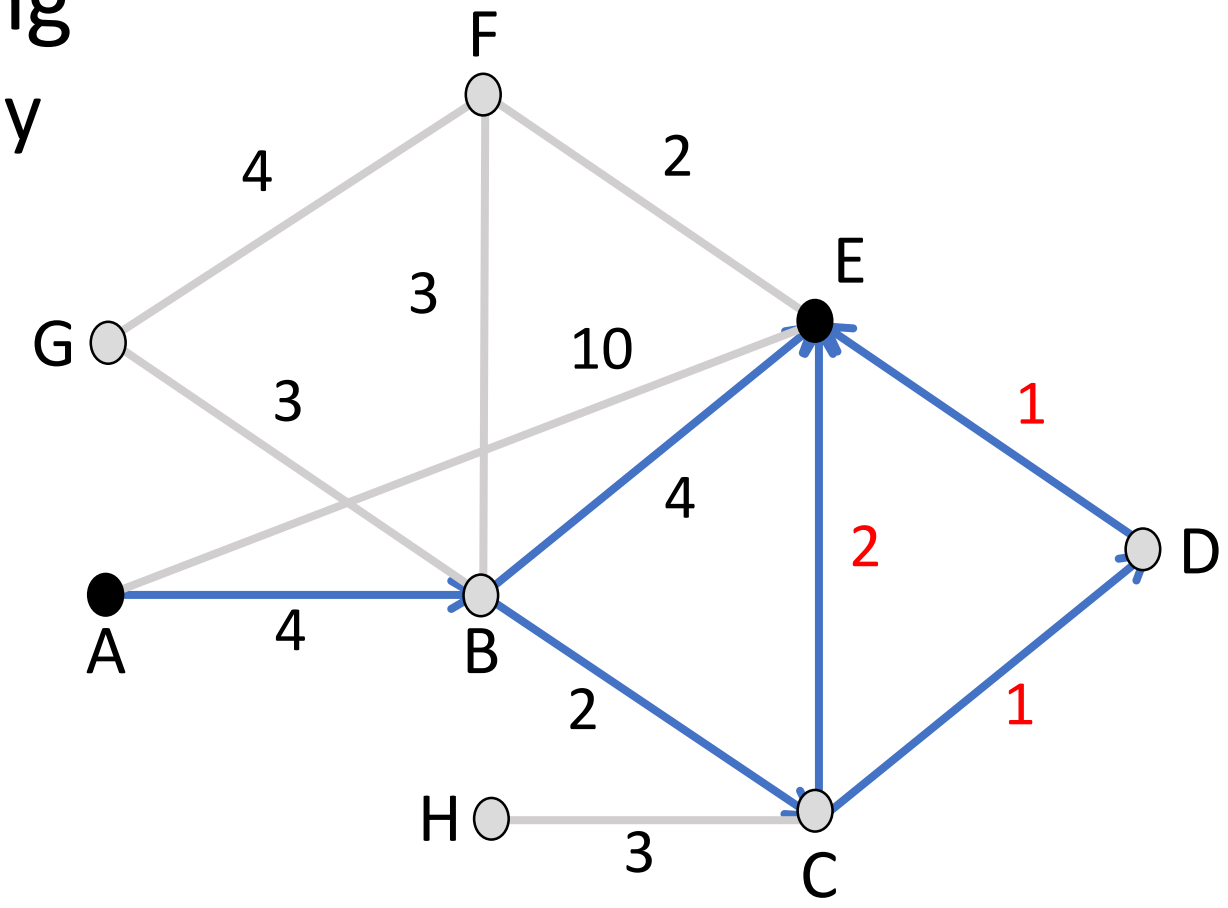
Multipath Routing

- 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

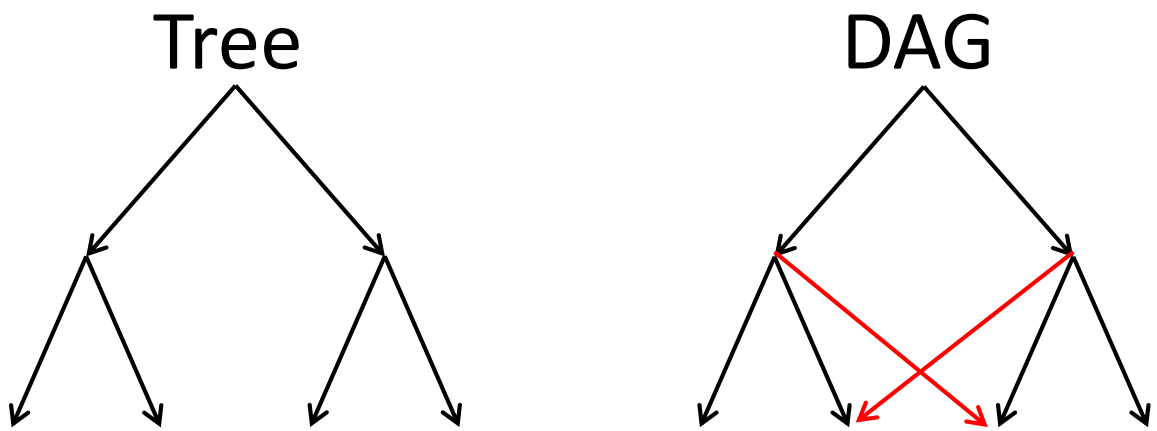
- One form of multipath routing
  - Extends shortest path model by keeping set if there are ties

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



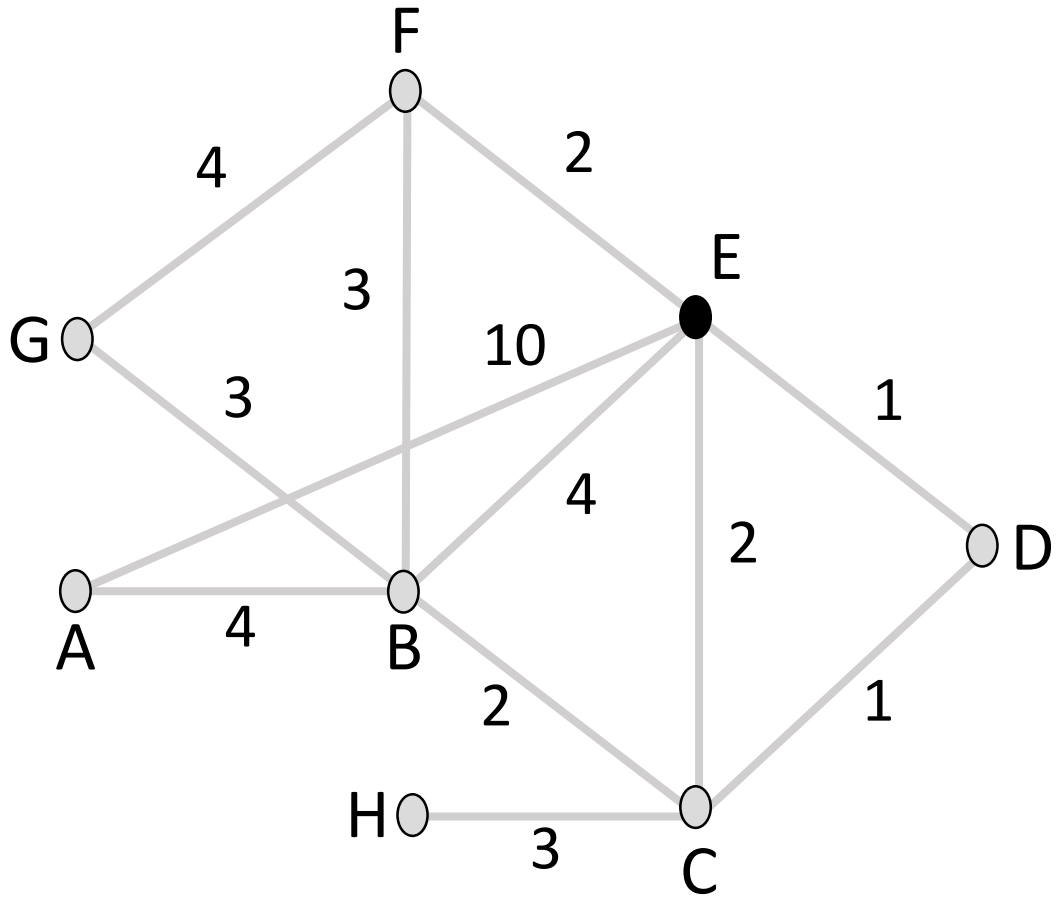
Source "Trees"

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



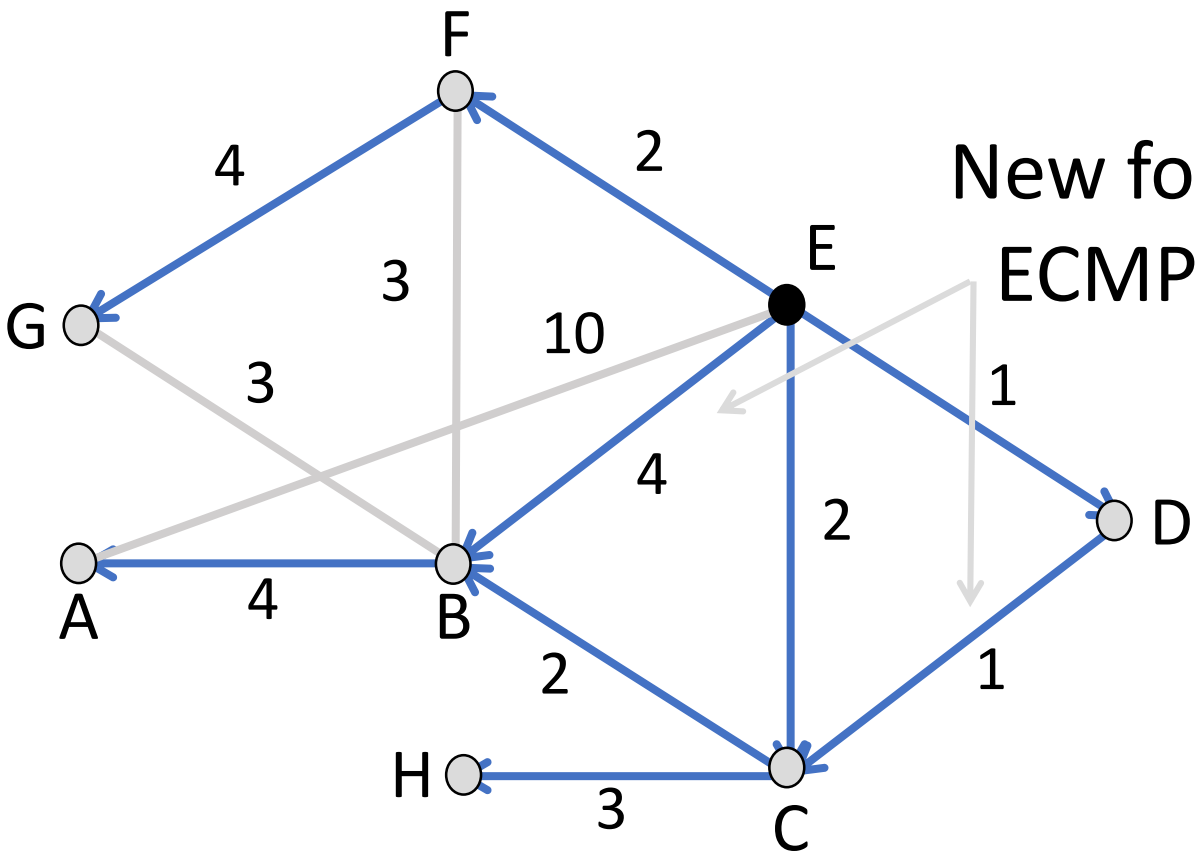
Source "Trees" (2)

- 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)

Source Tree for E



E's Forwarding Table

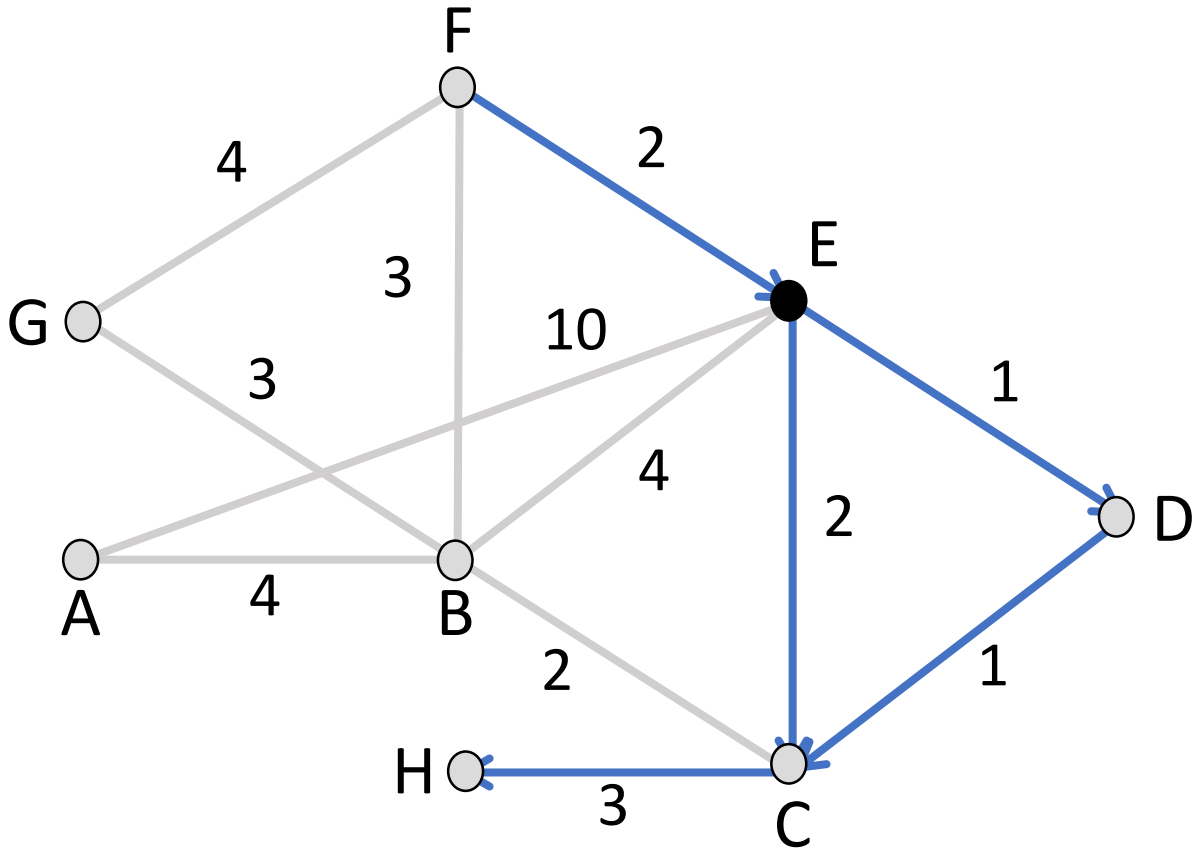
Node	Next hops
A	B, C, D
B	B, C, D
C	C, D
D	D
E	--
F	F
G	F
H	C, D

Forwarding with ECMP

- 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 pair on the same path
  - Source/destination pair is called a flow
  - Map flow identifier to single next hop
  - No jitter within flow, but less balanced

Forwarding with ECMP (2)

Multipath routes from F/E to C/H



E's Forwarding Choices

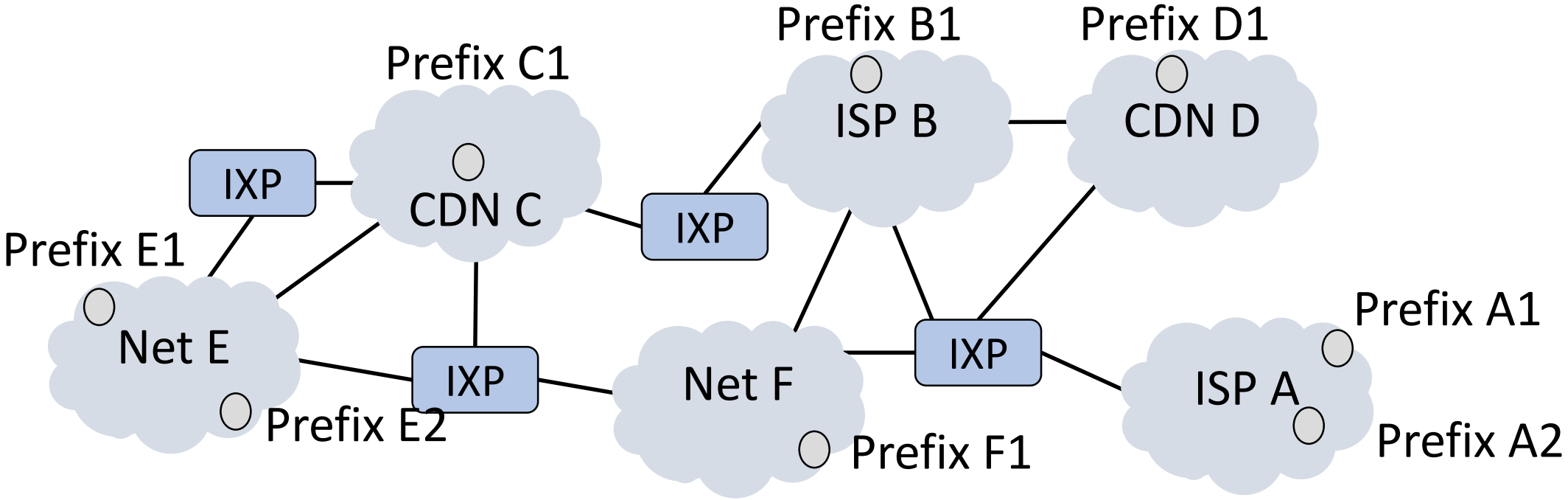
Flow	Possible next hops	Example choice
F → H	C, D	D
F → C	C, D	D
E → H	C, D	C
E → C	C, D	C

Use both paths to get to one destination

Border Gateway Protocol (BGP)

Structure of the Internet

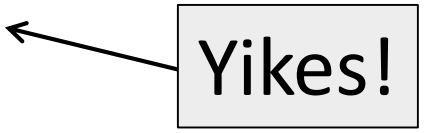
- Networks (ISPs, CDNs, etc.) group with IP prefixes
- Networks are richly interconnected, often using IXPs



Internet-wide Routing Issues

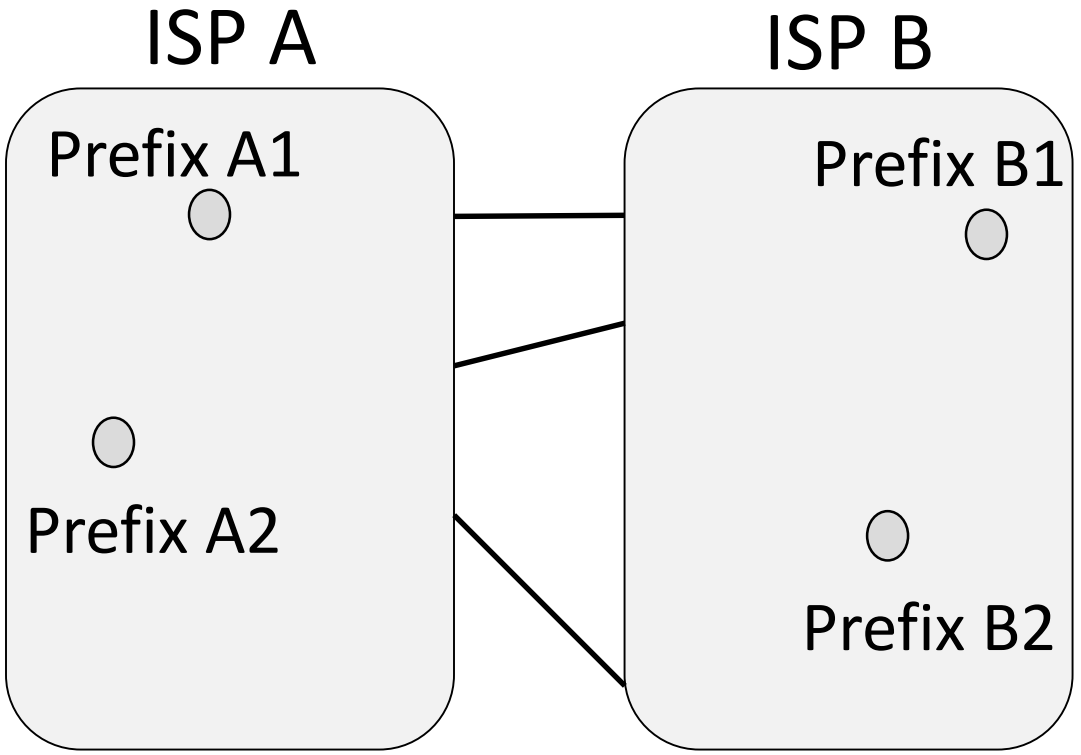
- Two problems beyond routing within a network

1. Scaling to very large networks
   - Techniques of IP prefixes, hierarchy, prefix aggregation
2. Incorporating policy decisions
   - Letting different parties choose their routes to suit their own needs



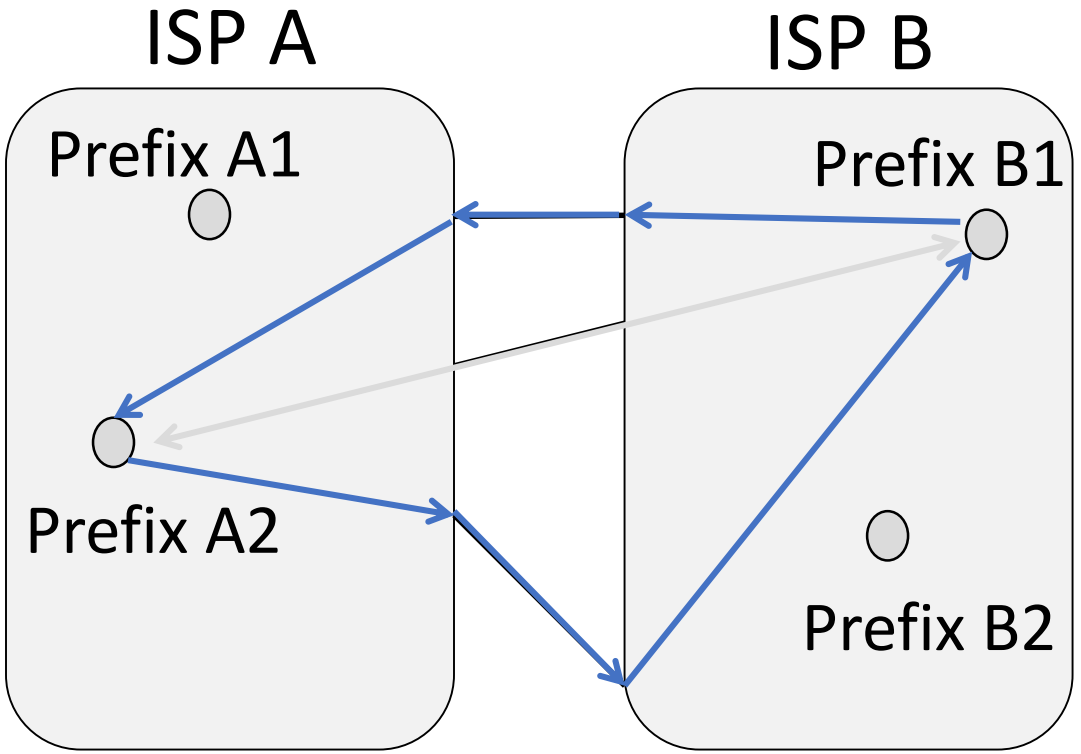
Effects of Independent Parties

- Each party selects routes to suit its own interests
  - e.g, shortest path in ISP
- What path will be chosen for A2→B1 and B1→A2?
  - What is the best path?



Effects of Independent Parties (2)

- Selected paths are longer than overall shortest path
  - And symmetric too!
- This is a consequence of independent goals and decisions, not hierarchy

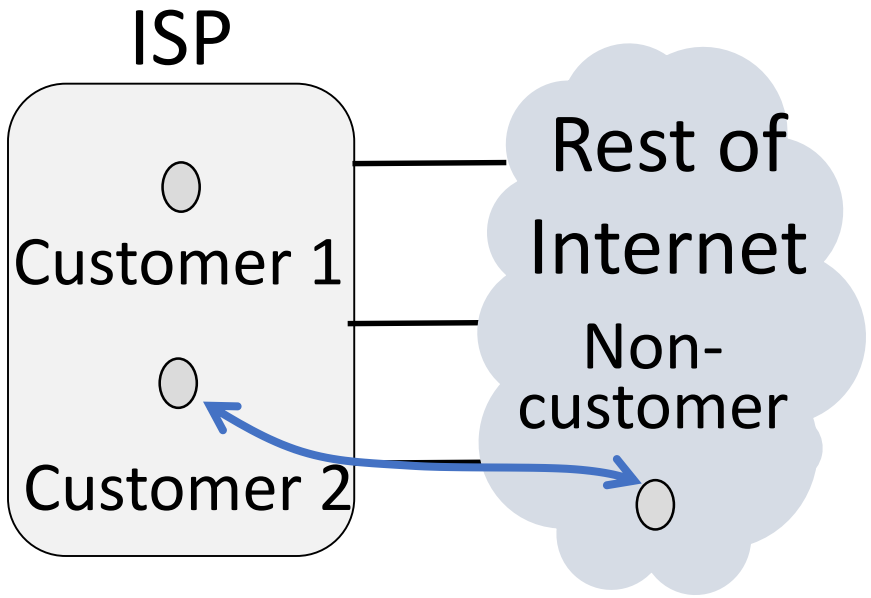


Routing Policies

- Capture the goals of different parties
  - Could be anything
  - E.g., Internet2 only carries non-commercial traffic
- Common policies we'll look at:
  - ISPs give TRANSIT service to customers
  - ISPs give PEER service to each other

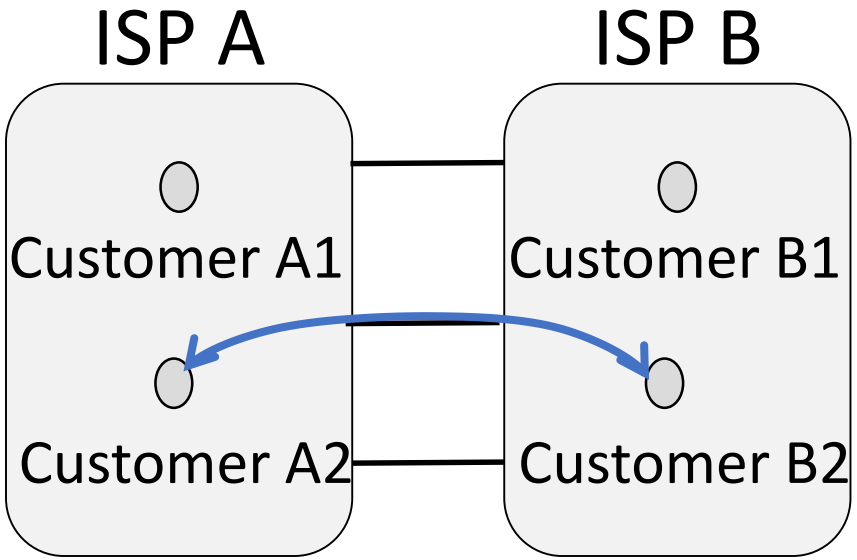
Routing Policies – Transit

- One party (customer) gets TRANSIT service from another party (ISP)
  - ISP accepts traffic for customer from the rest of Internet
  - ISP sends traffic from customer to the rest of Internet
  - Customer pays ISP for the privilege



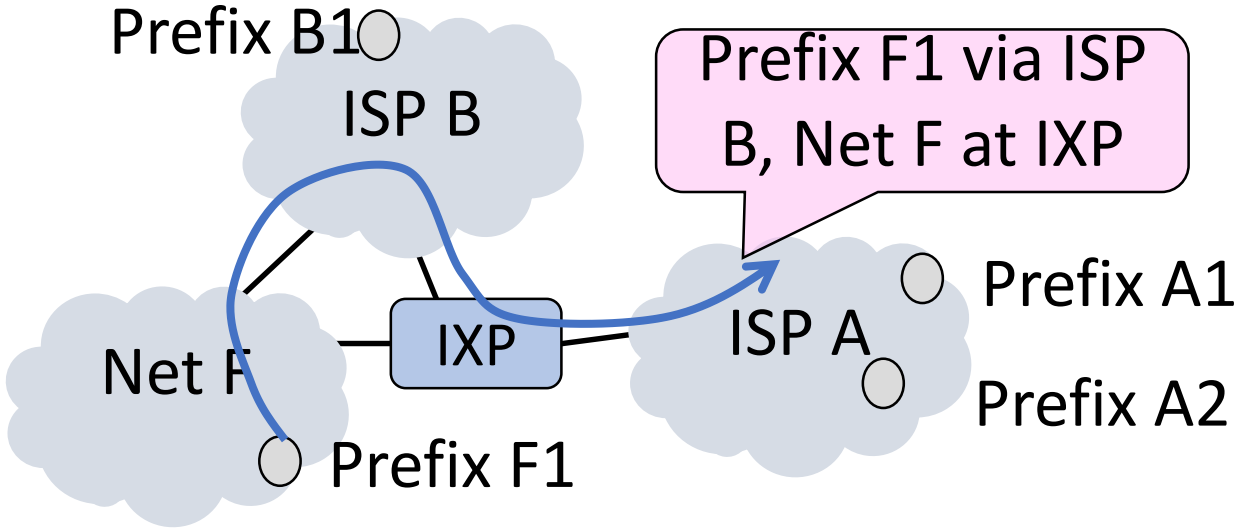
Routing Policies – Peer

- Both party (ISPs in example) get PEER service from each other
  - Each ISP accepts traffic from the other ISP only for their customers
  - ISPs do not carry traffic to the rest of the Internet for each other
  - ISPs don't pay each other



Routing with BGP (Border Gateway Protocol)

- iBGP is for internal routing
- eBGP is interdomain routing for the Internet
  - Path vector, a kind of distance vector



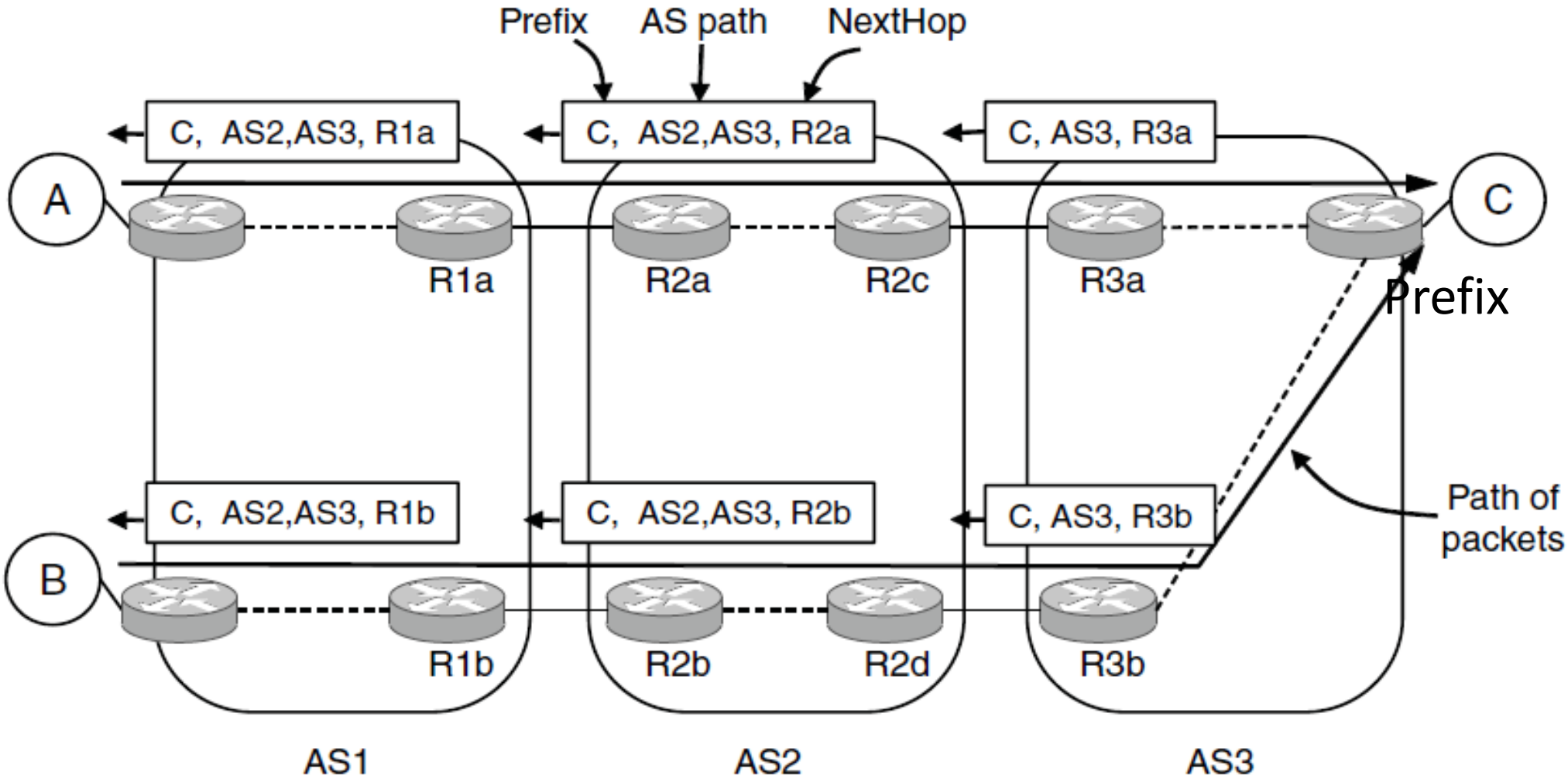
Routing with BGP (2)

- Parties like ISPs are called AS (Autonomous Systems)
- AS's **MANUALLY** configure their internal BGP routes/advertisements
- External routes go through complicated filters for forwarding/filtering
- AS BGP routers communicate with each other to keep consistent routing rules

Routing with BGP (2)

- Border routers of ASes announce BGP routes
- Route announcements have IP prefix, path vector, next hop
  - Path vector is list of ASes on the way to the prefix
  - List is to find loops
- Route announcements move in the opposite direction to traffic

Routing with BGP (3)



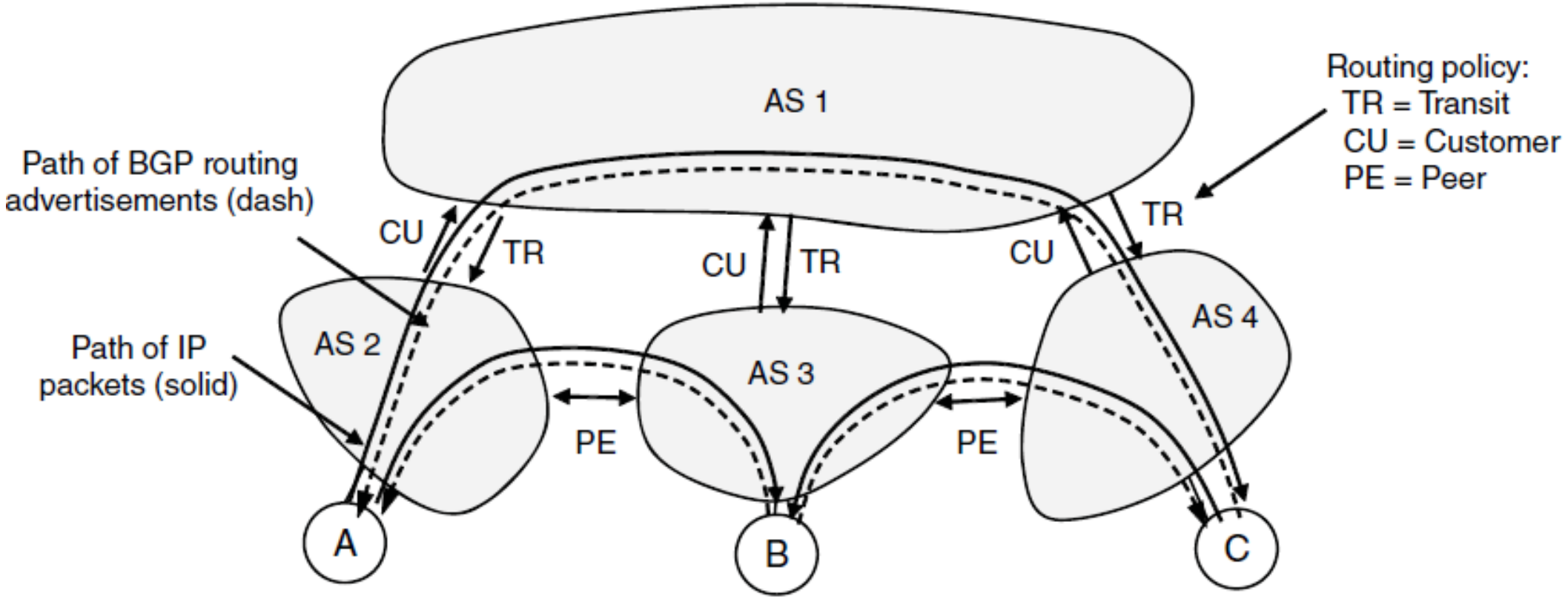
Routing with BGP (4)

Policy is implemented in two ways:

1. Border routers of ISP announce paths only to other parties who may use those paths
   - Filter out paths others can't use
2. Border routers of ISP select the best path of the ones they hear in any, non-shortest way

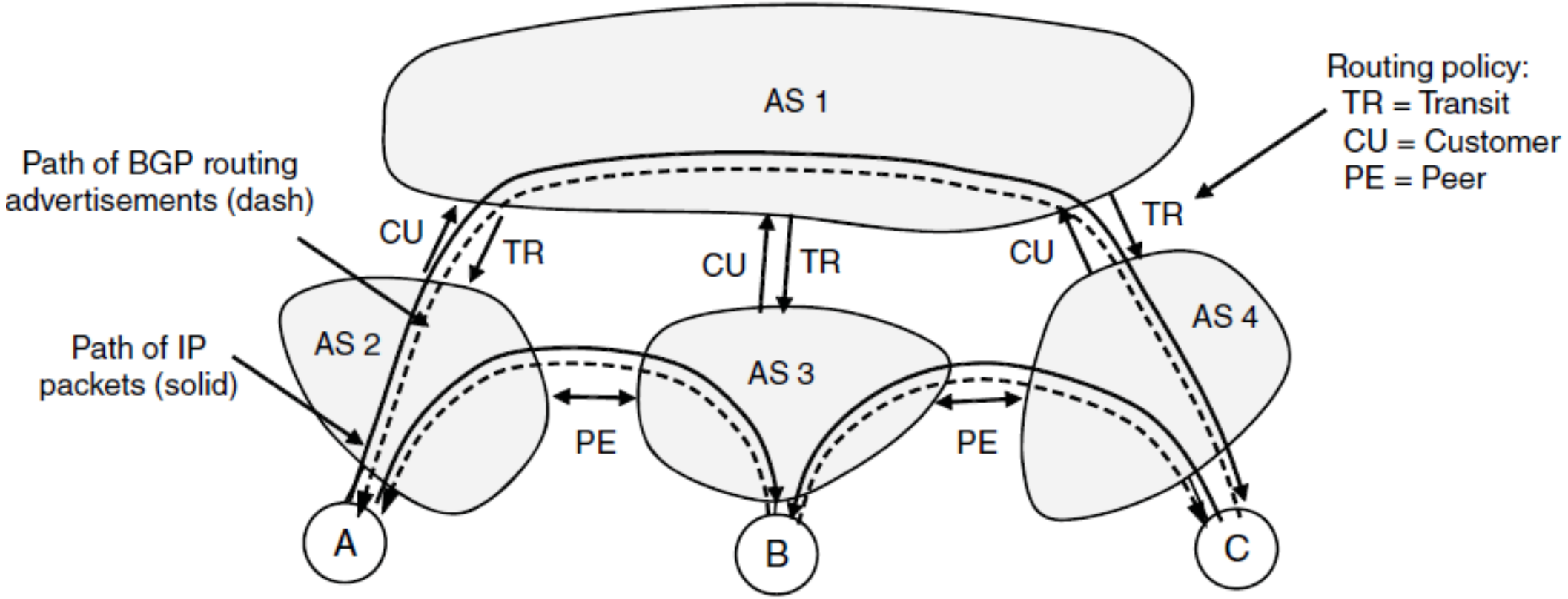
Routing with BGP (5)

- TRANSIT: AS1 says [B, (AS1, AS3)], [C, (AS1, AS4)] to AS2



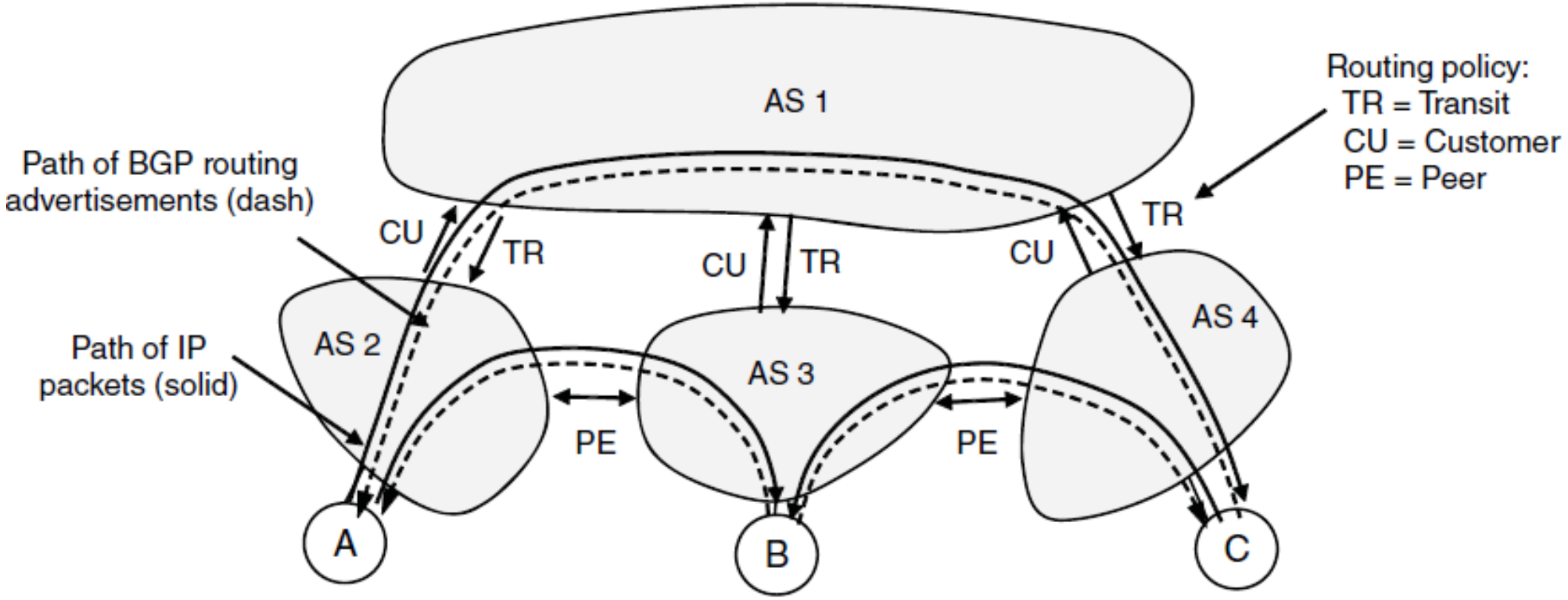
Routing with BGP (6)

- CUSTOMER (other side of TRANSIT): AS2 says [A, (AS2)] to AS1



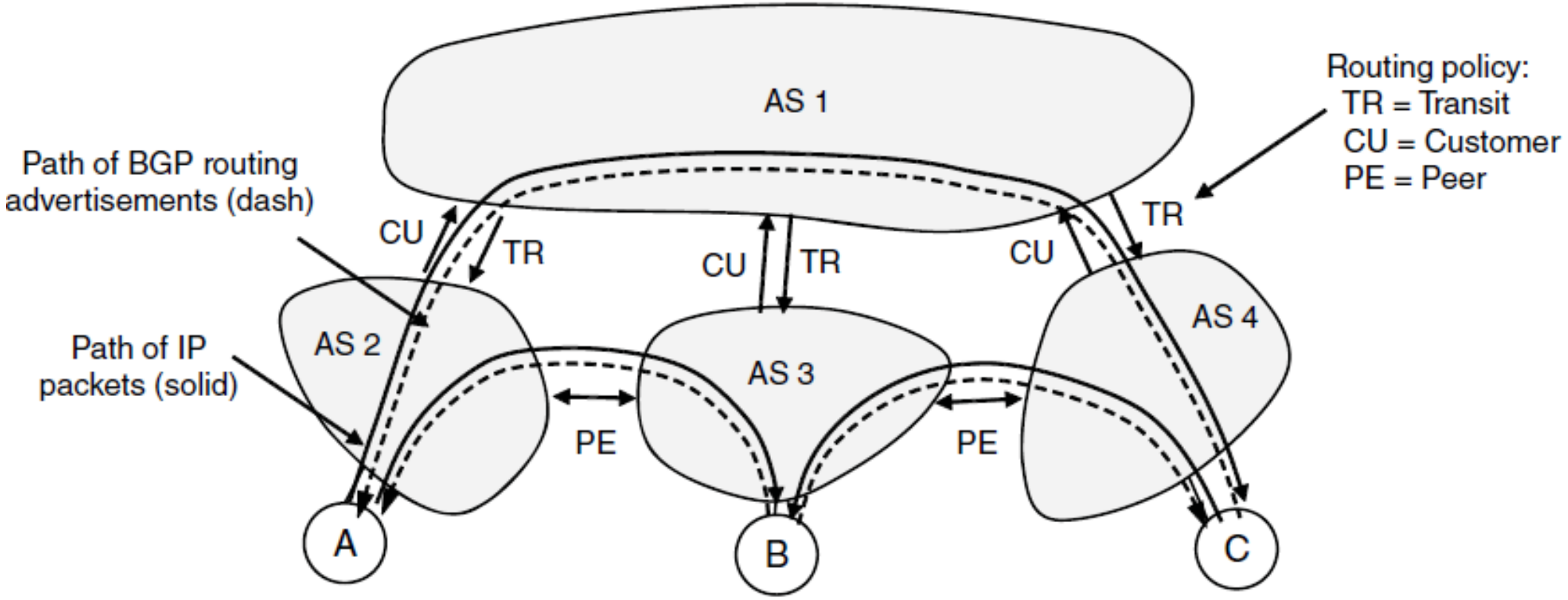
Routing with BGP (7)

- PEER: AS2 says [A, (AS2)] to AS3, AS3 says [B, (AS3)] to AS2



Routing with BGP (8)

- AS2 has two routes to B (AS1, AS3) and chooses AS3 (Free!)

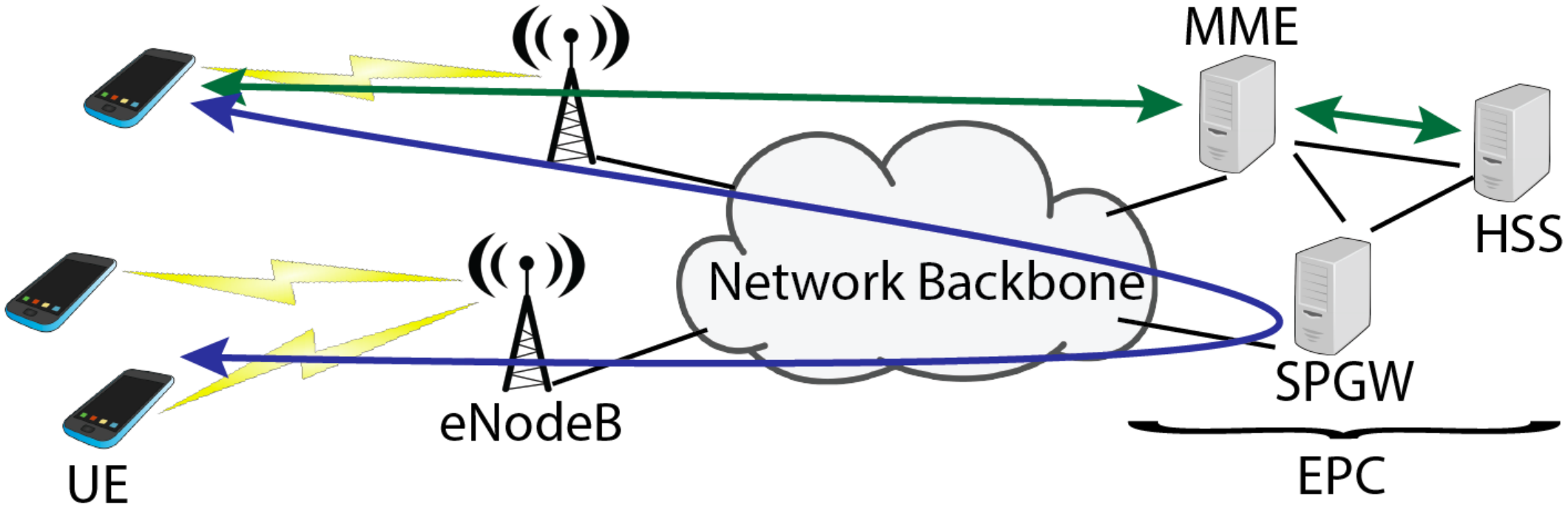


BGP Thoughts

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

Cellular Routing

Cellular Core Networks



Cellular Routing (old)

- Signaling System No. 5 (SS5)
  - Used inband signals to set up calls
  - Whistles and stuff that would indicate where the call is going and at what cost.

  Issues?

Cellular Routing

- Signaling System No. 7 (SS7)
  - Out of band signaling
  - Performs number translation, local number portability, prepaid billing, Short Message Service (SMS), roaming, and other stuff
  - Either directly connected or connected through aggregators such as Cybase
  - Business vs Protocols

Cellular Lookups

- An SSP telephone exchange receives a call to an 0800 number. This causes a trigger within the SSP that causes an SCP (Service Control Point) to be queried using SS7 protocols (INAP, TCAP). The SCP responds with a geographic number, e.g. 0121 XXX XXXX, and the call is actually routed to phone.