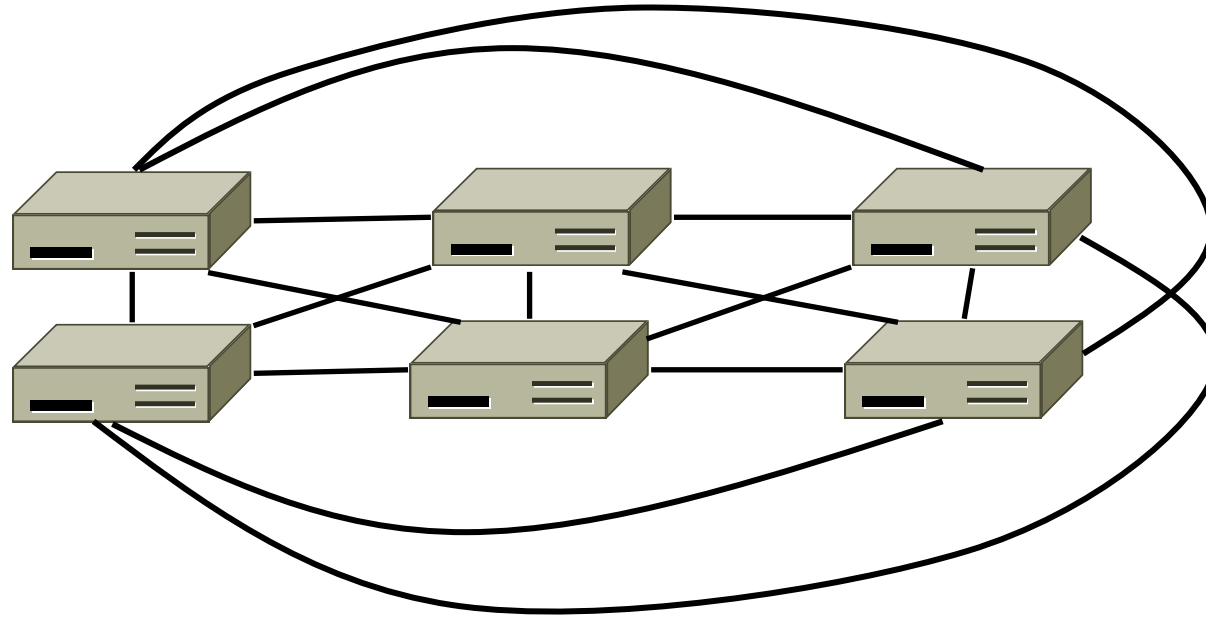


# Network Layer

Addressing, forwarding, routing

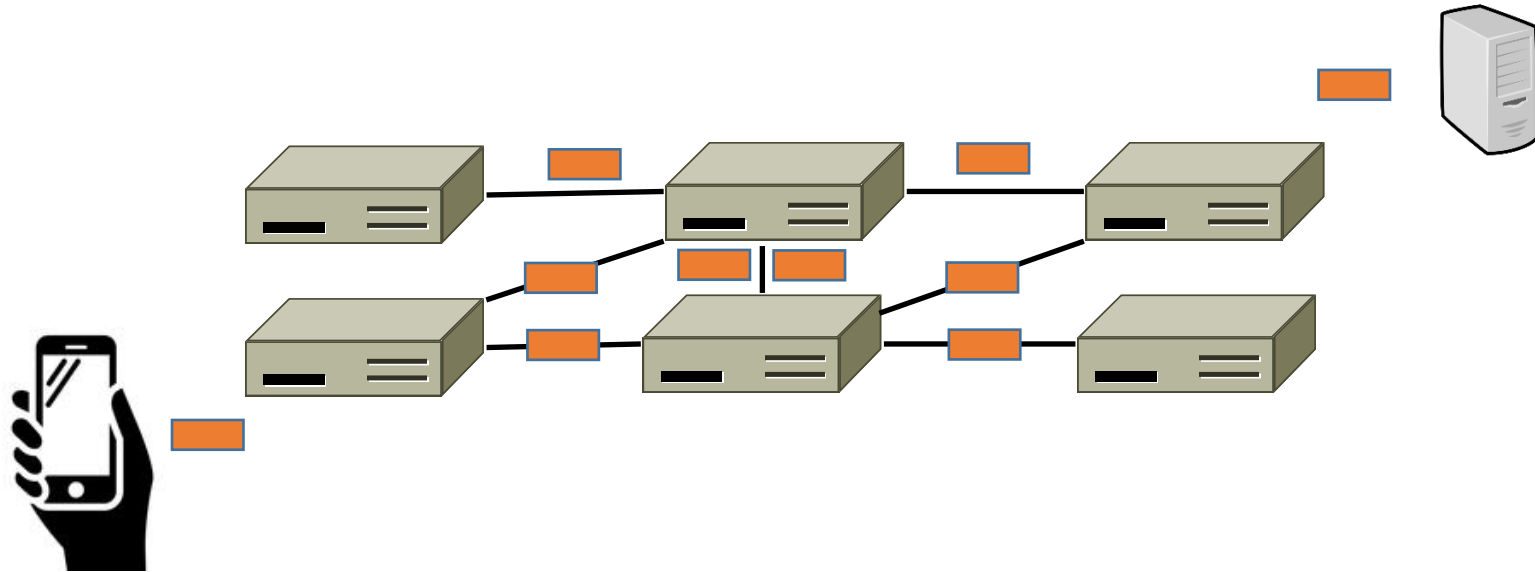
# Why do we need a Network layer?

- Cannot afford to directly connect everyone



# Why do we need a Network layer?

- Cannot broadcast all packets globally



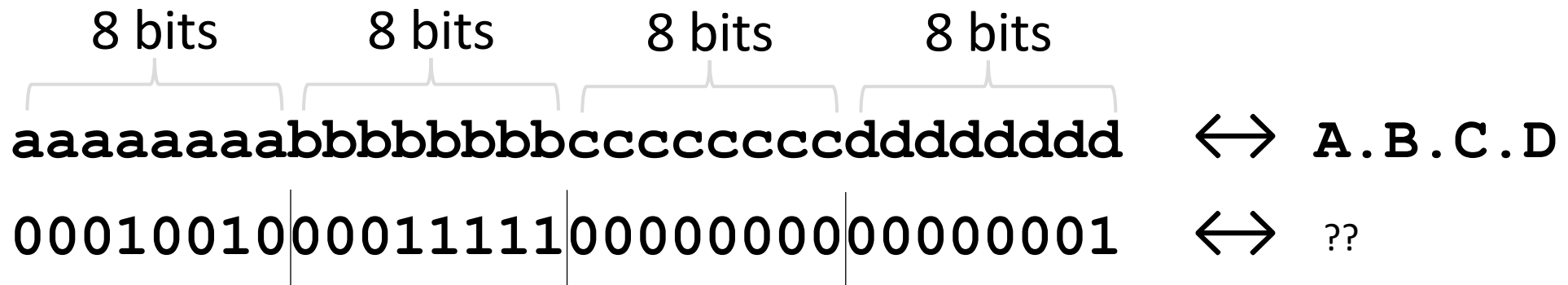
# Network layer functions

- Addressing
  - A globally unique way to “address” hosts
- Routing and forwarding
  - Finding paths and forwarding packets between hosts

Addressing

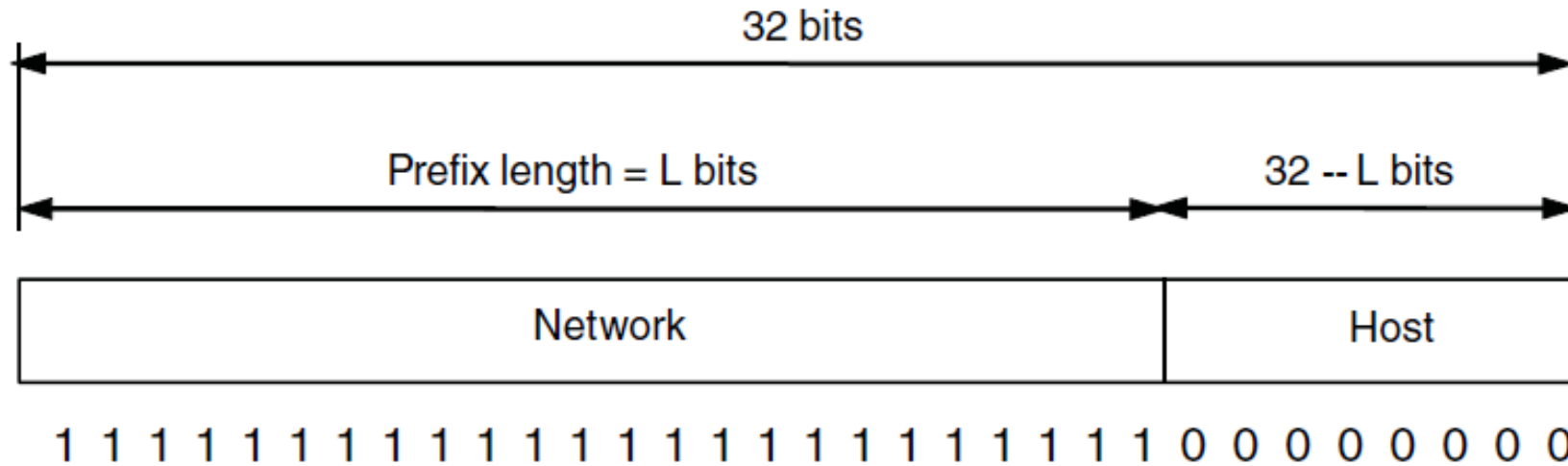
# IP Addresses

- IPv4 uses 32-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 L MSBs
  - There are  $2^{32-L}$  addresses aligned on  $2^{32-L}$  boundary



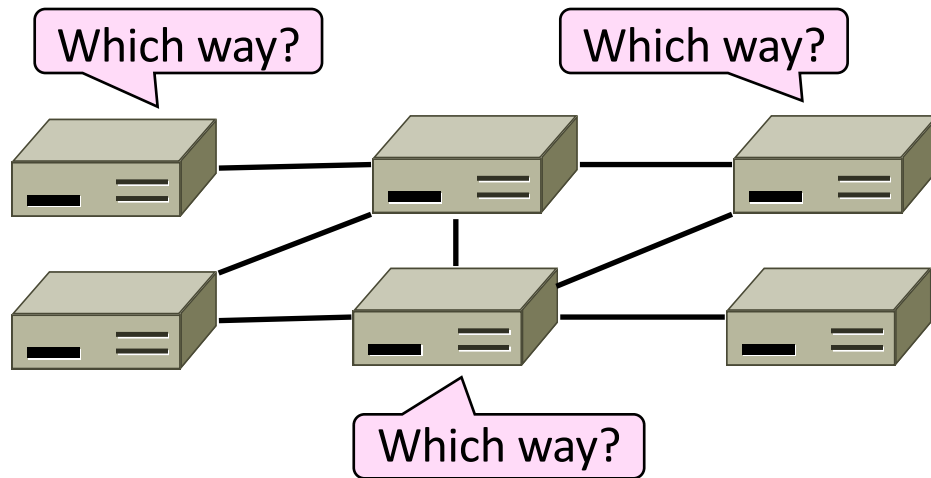
## IP Prefixes (2)

- Written in “IP 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 /32 is 1 address
- Generally (not always!), hosts with the same prefix are close to each other

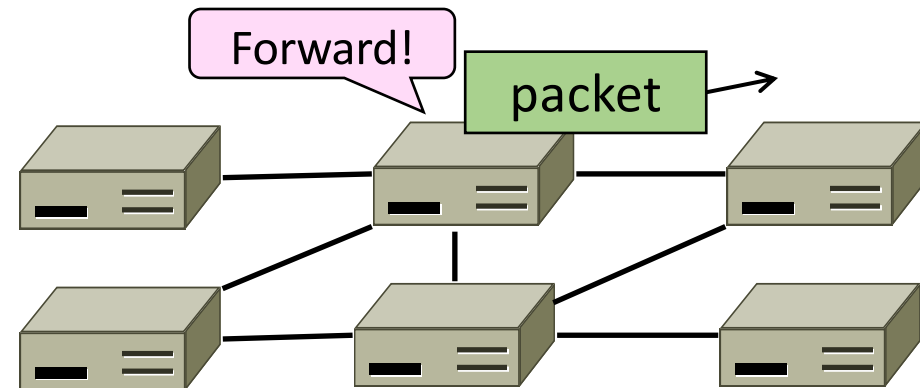


# Routing versus Forwarding

- Routing: deciding where to send traffic



- Forwarding: sending a packet on its way

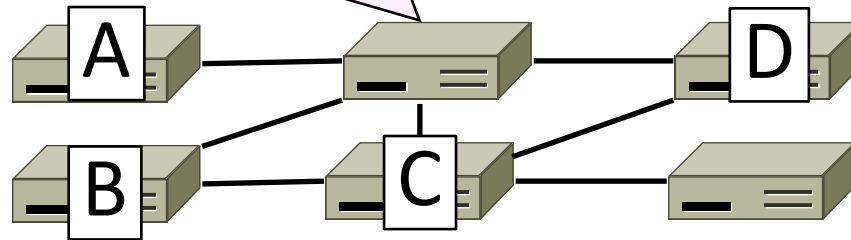


Forwarding

# IP Forwarding

- Nodes use a table that lists the next hop for prefixes
- Lookup the destination address's prefix in the table

Prefix	Next Hop
102.24.0.0/19	D
192.24.12.0/22	B



# Longest Prefix Matching

- Prefixes in the forwarding table can overlap

Prefix	Next Hop
0.0.0.0/0	A
192.24.0.0/19	B
192.24.12.0/22	C

- Longest prefix matching 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

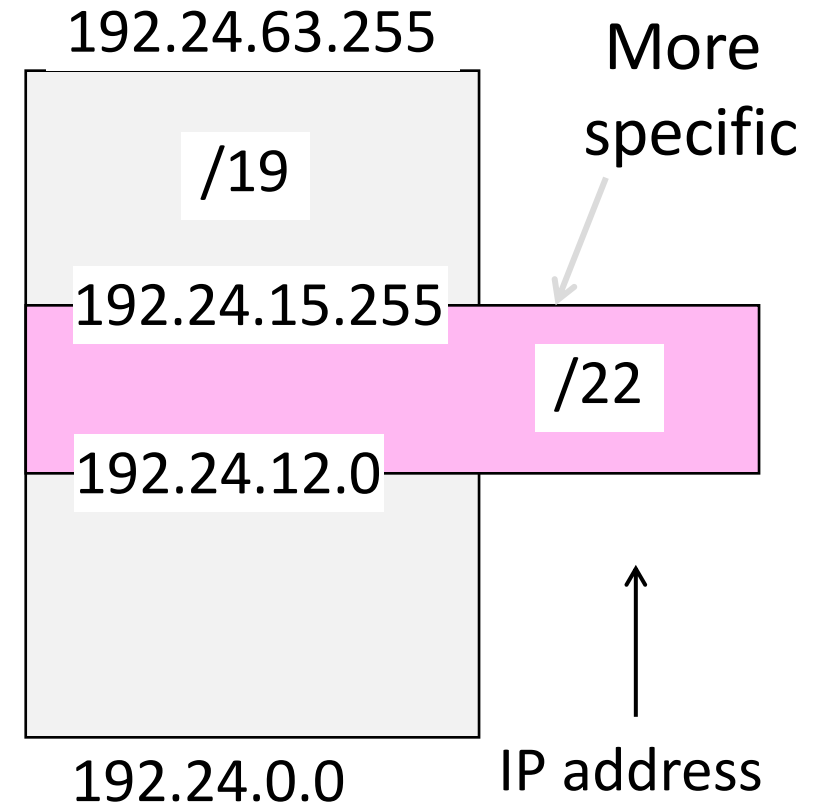
# Longest Prefix Matching (2)

Prefix	Next Hop
192.24.0.0/19	D
192.24.12.0/22	B

192.24.6.0 → ?

192.24.14.32 → ?

192.24.54.0 → ?



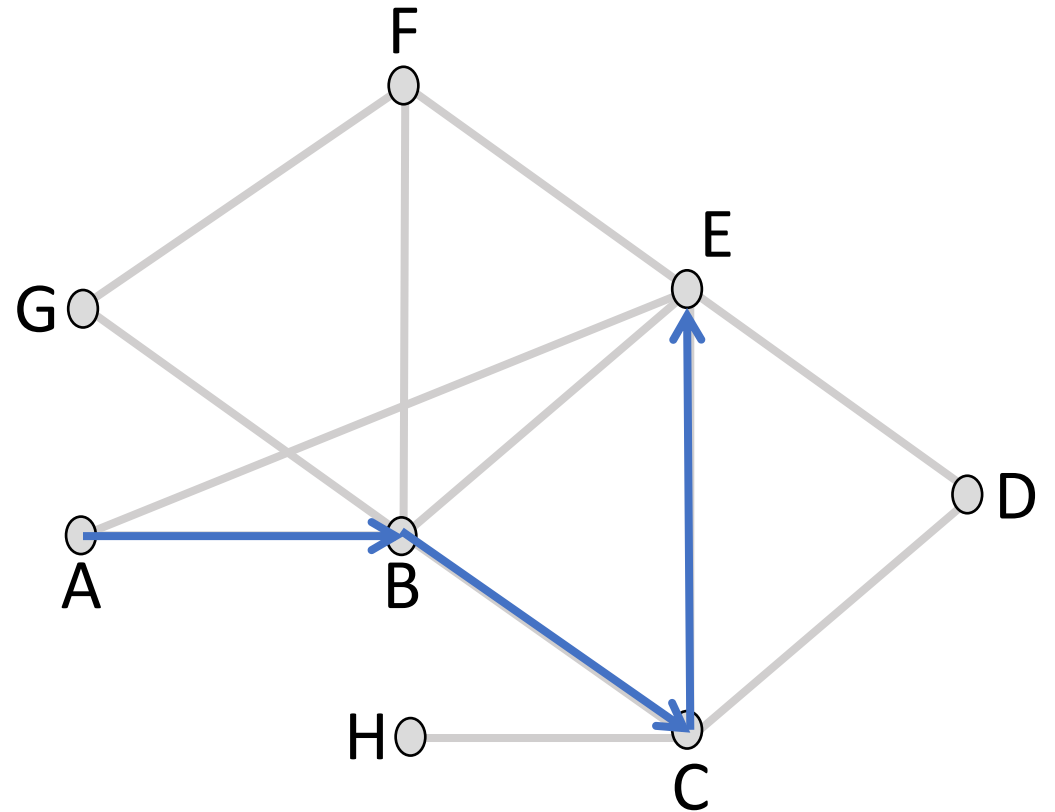
# Flexibility of Longest Prefix Matching

- Can provide default behavior, with less specifics
  - Send traffic going outside an organization to a border router (gateway)
- Can special case behavior, with more specifics
  - For performance, economics, security, ...

Routing

# What are “Best” paths?

- 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





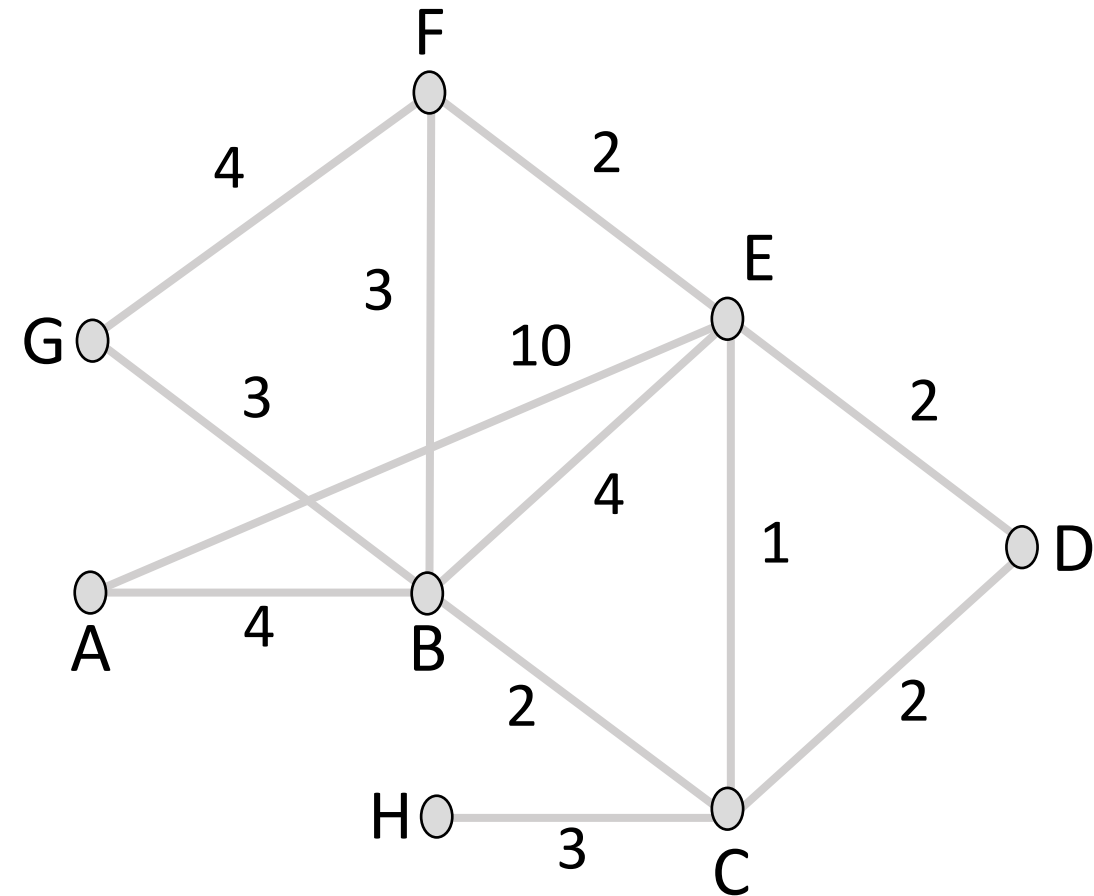
# Shortest paths or least cost paths

Approximate “best” with a cost function that captures the factors

1. Assign each link a cost (distance)
2. Define best paths between each pair of nodes as paths with the least cost
3. Break ties among best paths or use all of them

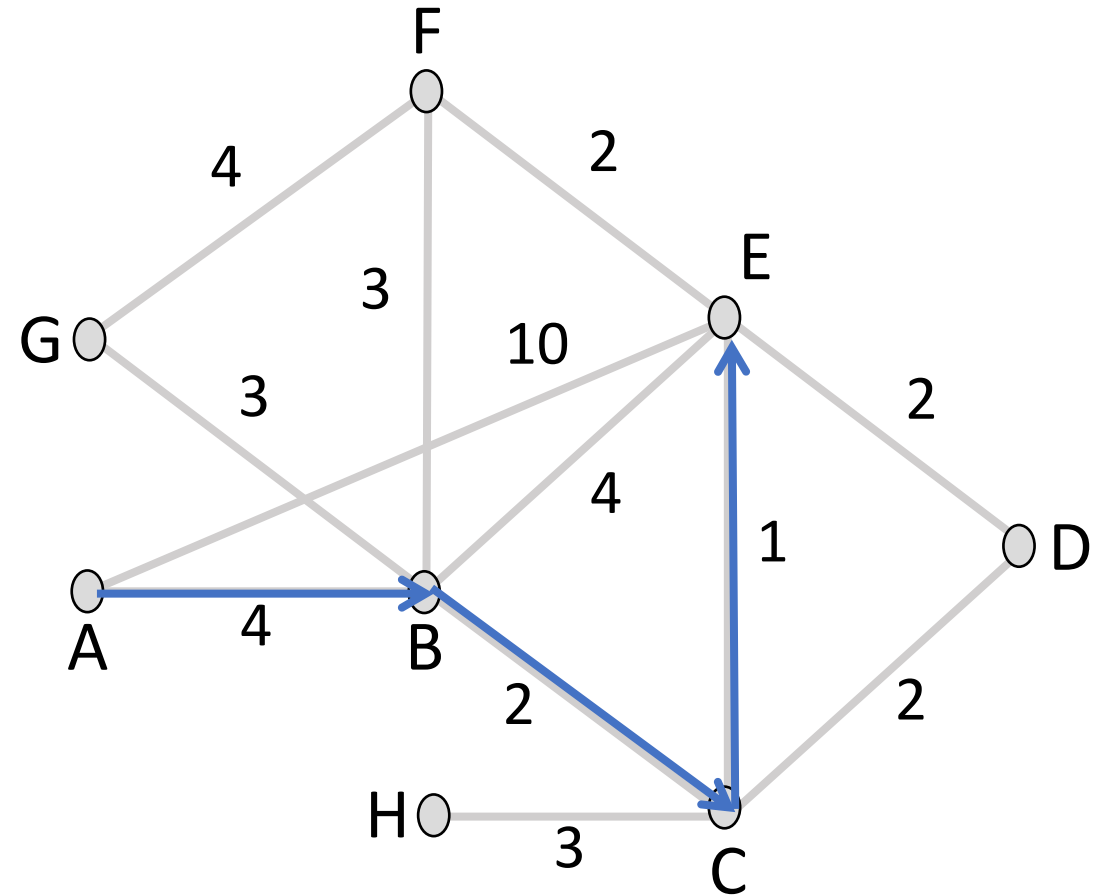
## Shortest Paths (2)

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



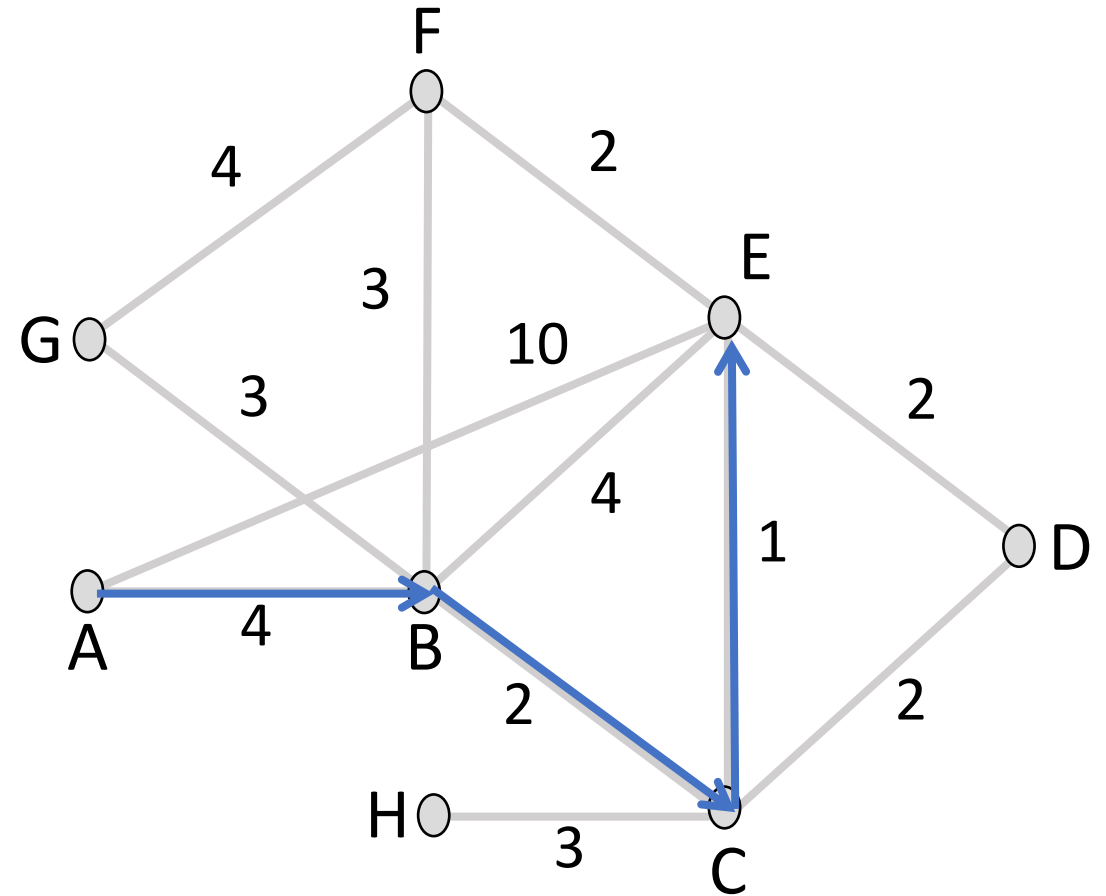
# Shortest Paths (3)

- ABCE is a shortest path
  - $\text{cost}(\text{ABCE}) = 4 + 2 + 1 = 7$
- It is shorter than:
  - $\text{cost}(\text{ABE}) = 8$
  - $\text{cost}(\text{ABFE}) = 9$
  - $\text{cost}(\text{AE}) = 10$
  - $\text{cost}(\text{ABCDE}) = 10$



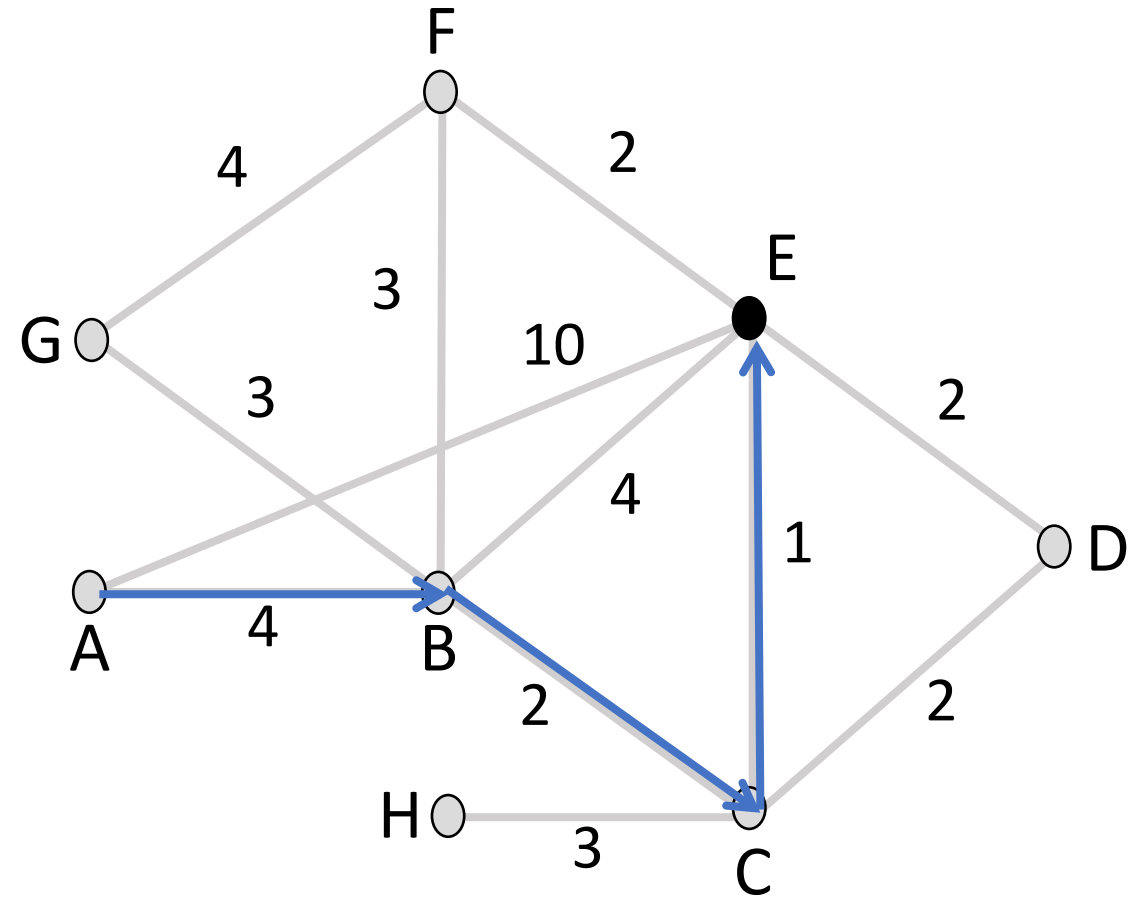
# Shortest Paths (4)

- Optimality property:
  - Subpaths of shortest paths are also shortest paths
- ABCE is a shortest path
  - So are ABC, AB, BCE, BC, CE



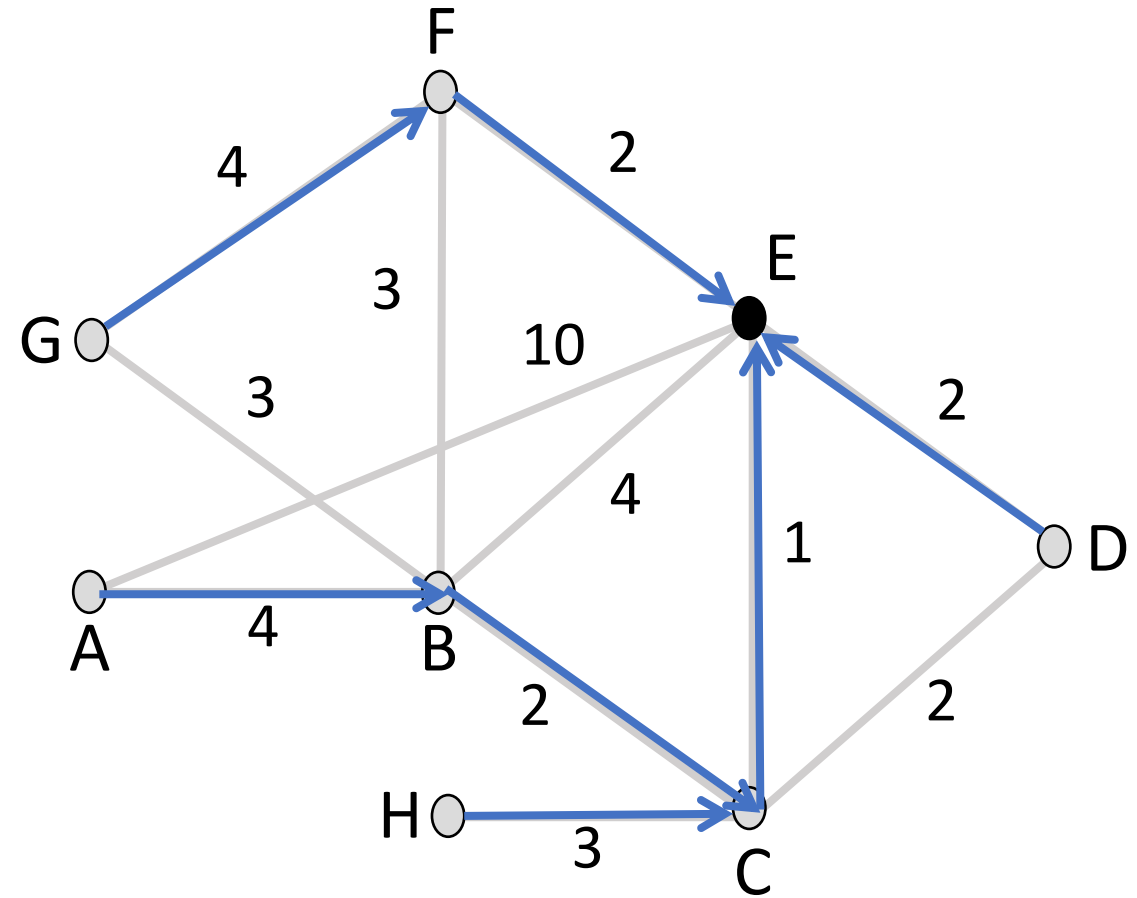
# Sink Trees

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



# Implications of Sink Trees

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



# How to find shortest paths?

- We'll illustrate distance vector routing
  - Distributed version of Bellman-Ford
  - One of the main approaches to routing
    - Another approach is link-state (ignore for now)
    - Another approach is path-vector (ignore for now)

# Distance Vector Routing

Each node maintains a vector of (distance, next hop) to all destinations

1. Initialize vector with 0 (zero) cost to self,  $\infty$  (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

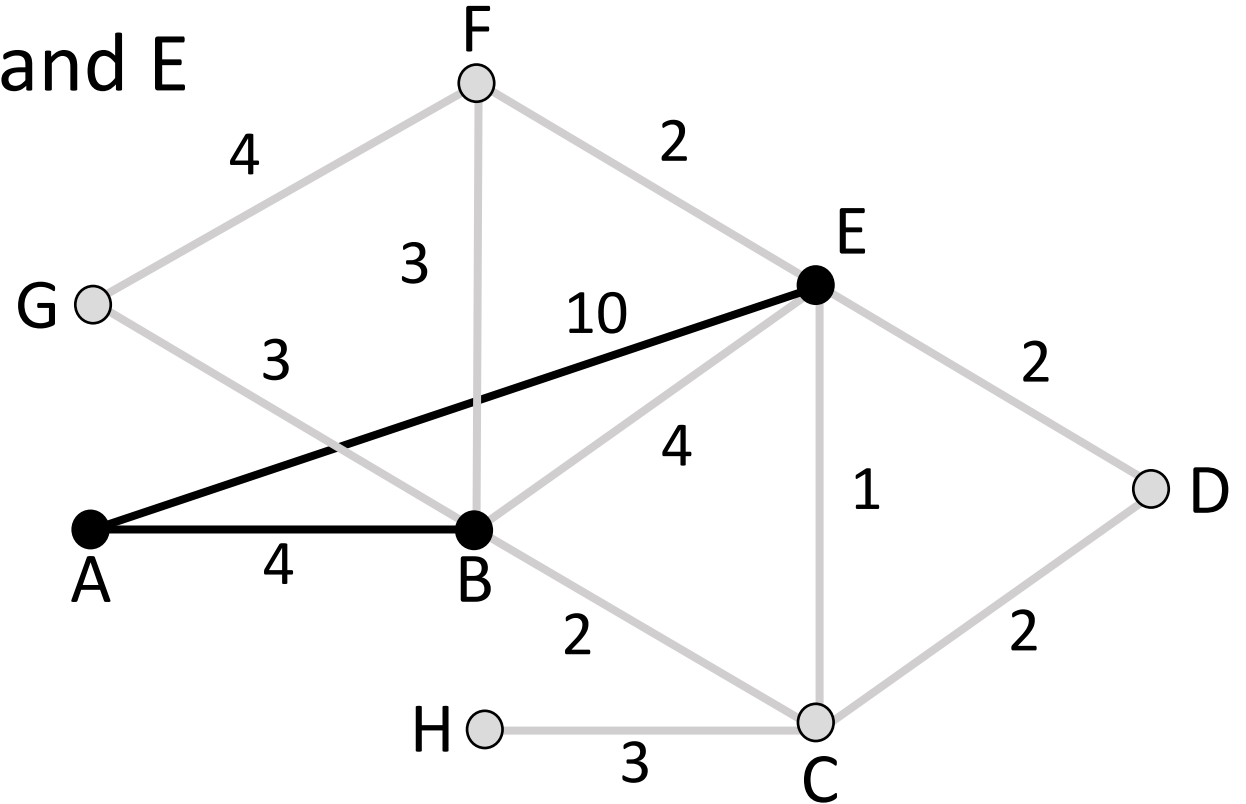


# Distance Vector (2)

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

Initial vector

To	Cost
A	0
B	$\infty$
C	$\infty$
D	$\infty$
E	$\infty$
F	$\infty$
G	$\infty$
H	$\infty$



# Distance Vector (3)

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

To	B says	E says
A	$\infty$	$\infty$
B	0	$\infty$
C	$\infty$	$\infty$
D	$\infty$	$\infty$
E	$\infty$	0
F	$\infty$	$\infty$
G	$\infty$	$\infty$
H	$\infty$	$\infty$

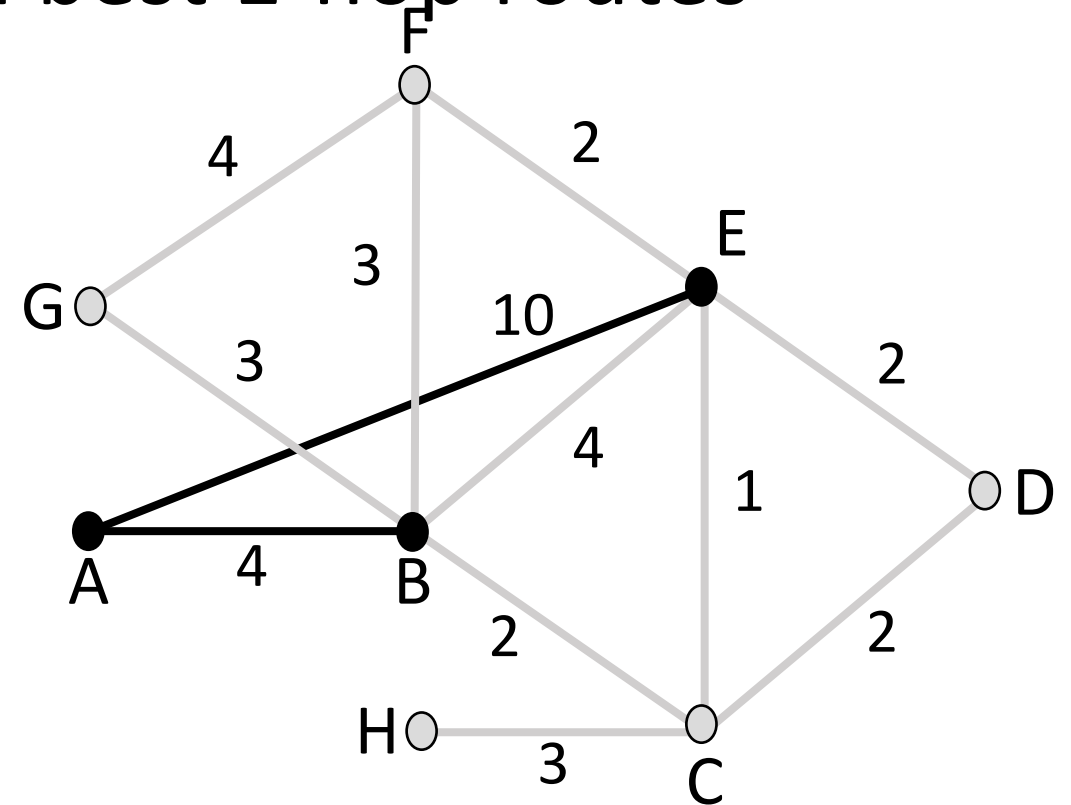
→

B +4	E +10
$\infty$	$\infty$
4	$\infty$
$\infty$	$\infty$
$\infty$	$\infty$
$\infty$	10
$\infty$	$\infty$
$\infty$	$\infty$
$\infty$	$\infty$

→

A's Cost	A's Next
0	--
4	B
$\infty$	--
$\infty$	--
10	E
$\infty$	--
$\infty$	--
$\infty$	--

Learned better route



# Distance Vector (4)

- Second exchange; learn best 2-hop routes

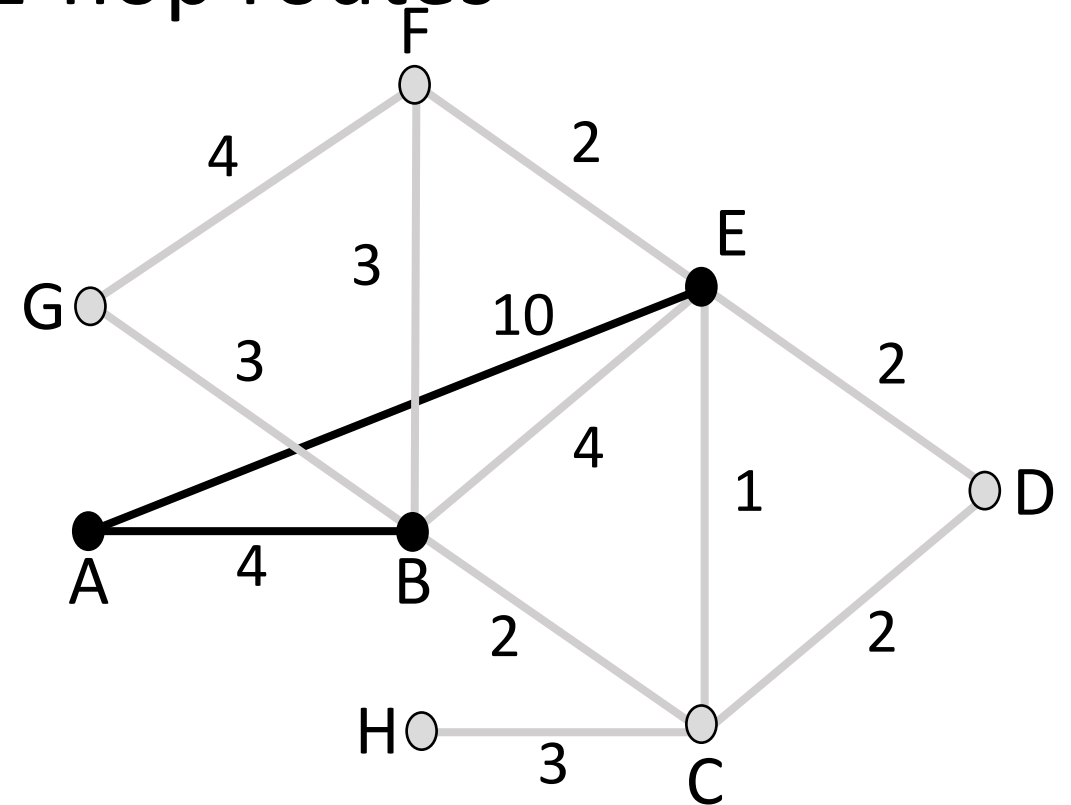
To	B says	E says
A	4	10
B	0	4
C	2	1
D	$\infty$	2
E	4	0
F	3	2
G	3	$\infty$
H	$\infty$	$\infty$

→

B +4	E +10
8	20
4	14
6	11
$\infty$	12
8	10
7	12
7	$\infty$
$\infty$	$\infty$

→

A's Cost	A's Next
0	--
4	B
6	B
12	E
8	B
7	B
7	B
$\infty$	--



# Distance Vector (4)

- Third exchange; learn best 3-hop routes

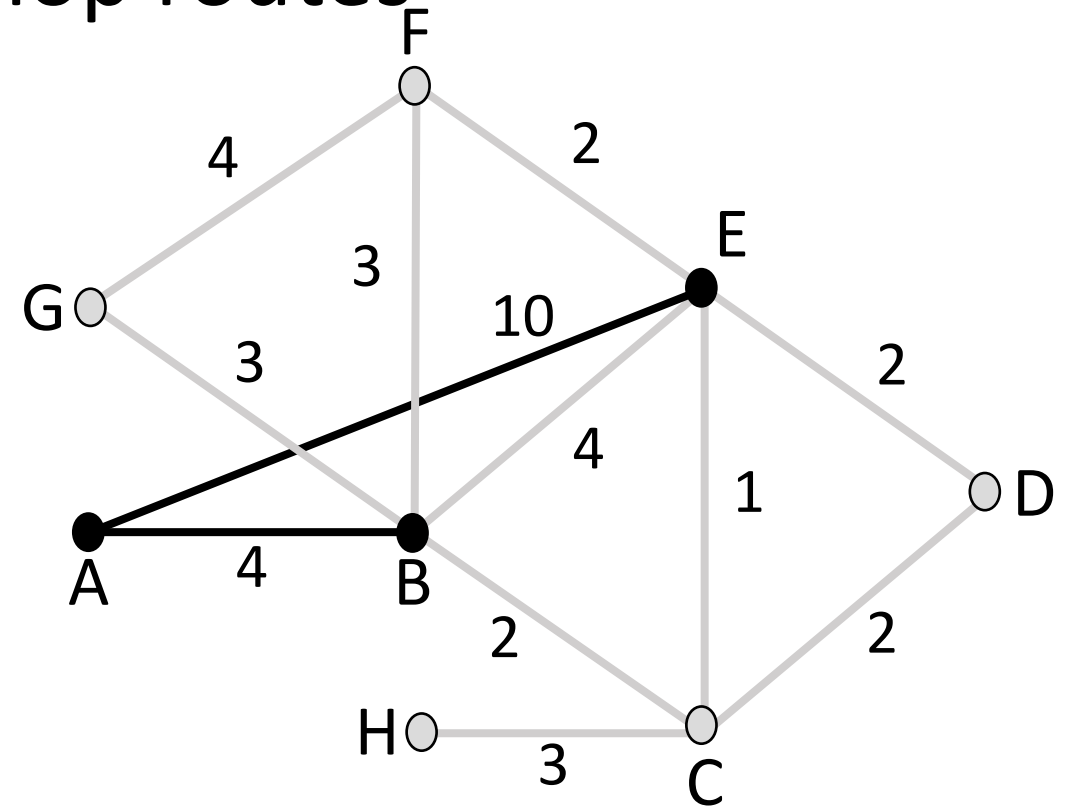
To	B says	E says
A	4	8
B	0	3
C	2	1
D	4	2
E	3	0
F	3	2
G	3	6
H	5	4



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

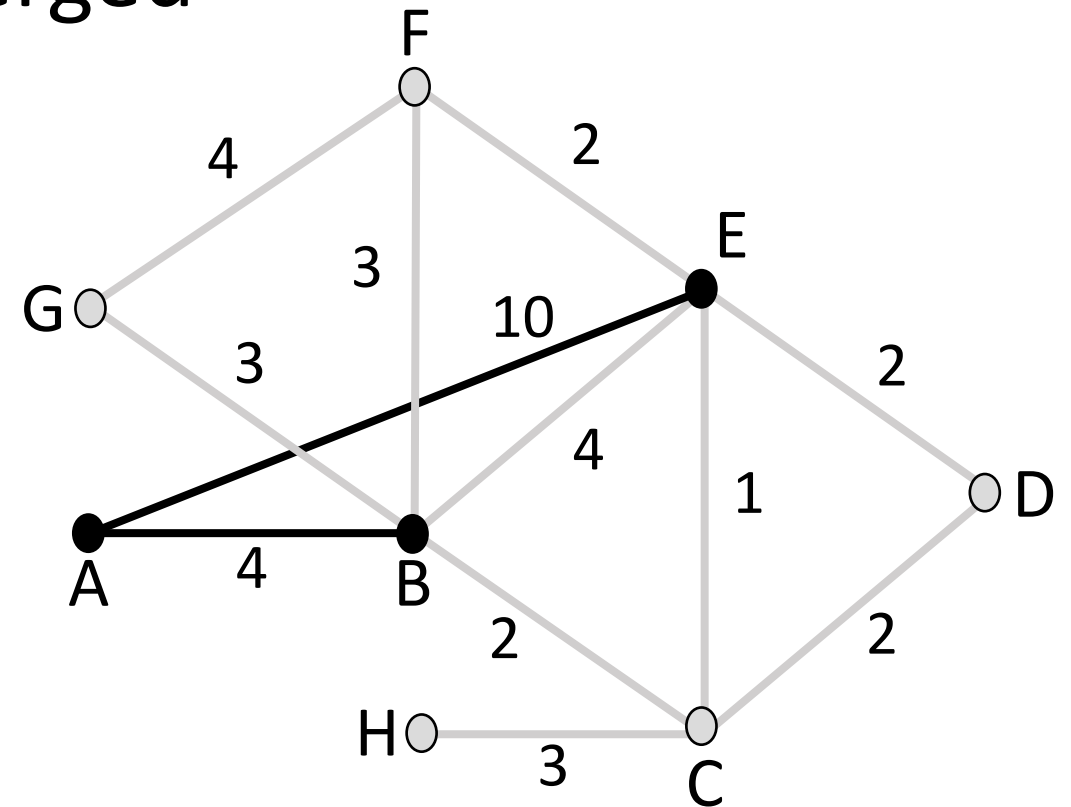
To	B says	E says
A	4	7
B	0	3
C	2	1
D	4	2
E	3	0
F	3	2
G	3	6
H	5	4



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



# Equal-Cost Multi-Path Routing

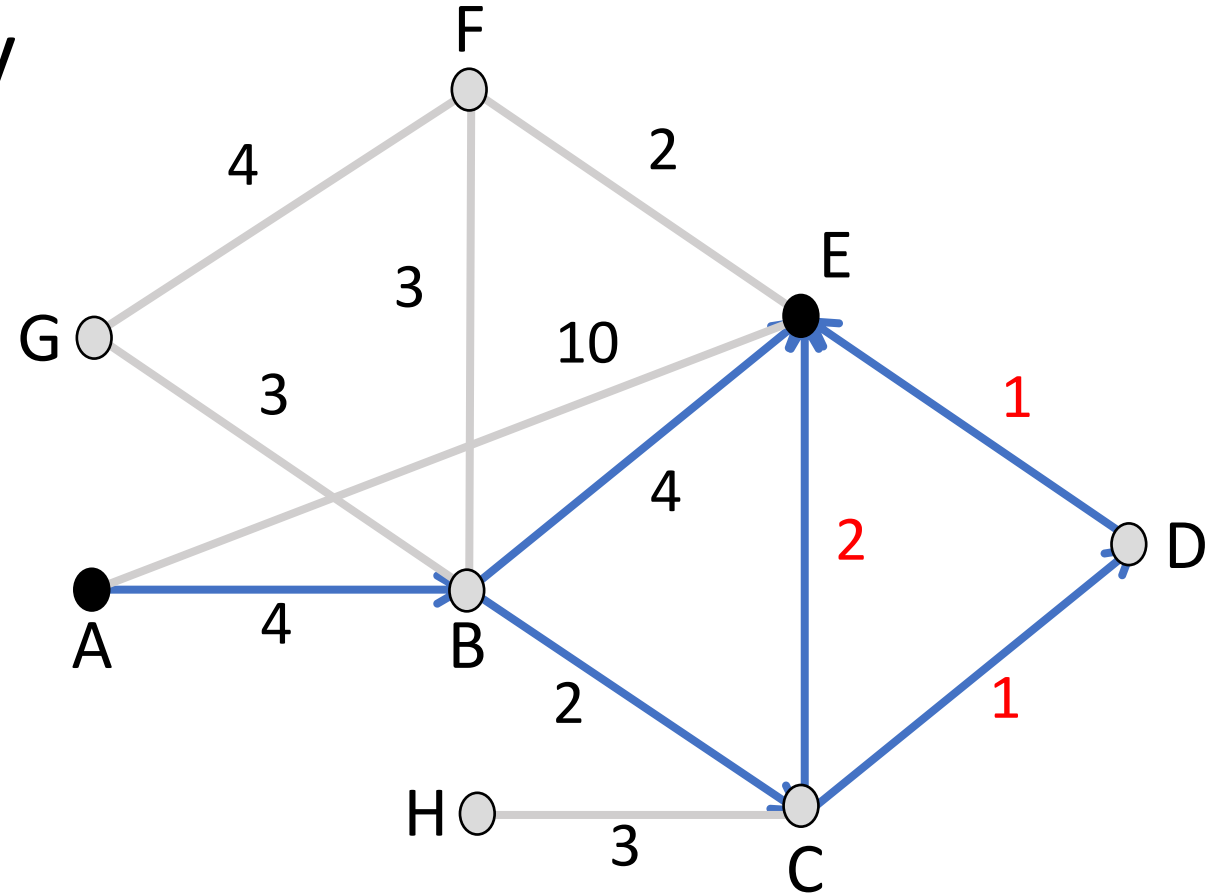
# Multipath Routing

- Use multiple best paths from node to destination
  - 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

Extends shortest path model by keeping set if there are ties

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







# Forwarding with ECMP

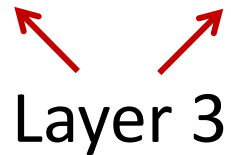
- Could randomly pick a next hop for each packet based on destination
  - Balances load, but adds jitter
- Instead, try to send packets from a given source/destination pair on the same path
  - Source/destination pair is called a flow
  - Map flow identifier to single next hop
  - No jitter within flow,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

Access control lists (ACLs)

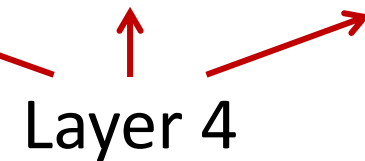
# ACLs

- Rules to permit or deny specific types of traffic
- Match 5 tuple (**source IP, dest IP, protocol, source port, dest port**)

Layer 3

Two red arrows point from the text 'Layer 3' to the source IP '10.1.1.2' and the destination IP '20.3.4.4' in the first two lines of the ACL rule.

Layer 4

Three red arrows point from the text 'Layer 4' to the protocol 'udp', the source port '53', and the destination port '20-21' in the first two lines of the ACL rule.

```
deny 10.1.1.2 20.3.4.4 udp any 53
permit 10.1.1.0/24 20.3.4.5 tcp any 20-21
deny all
```

- Rules are processed in order and the first rule that matches is applied
  - Unlike longest prefix matching

# ACLs (2)

- Match 5 tuple (**source IP, dest IP, protocol, source port, dest port**)
- Can have multiple TCP connections per host
  - Ports designate which application (process) to deliver the traffic to
  - Example: port 80 typically used for web server
- Real ACLs are often much more complex
  - Match TCP flags, ECN, DSCP, fragment offset...

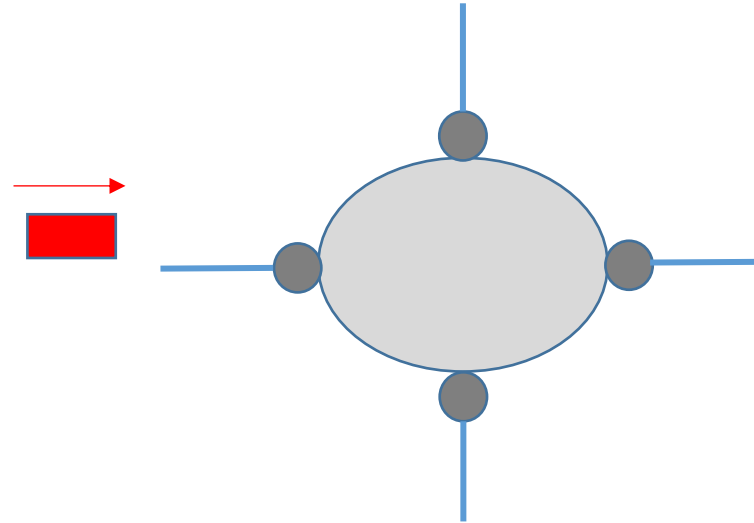
- Standard ACLs
- Extended ACLs
- Dynamic (lock and key) ACLs
- IP-named ACLs
- Reflexive ACLs
- Time-based ACLs that use time ranges
- Commented IP ACL entries
- Context-based ACLs
- Authentication proxy
- Turbo ACLs
- Distributed time-based ACLs

# ACLs (3)

- Can apply an ACL to a router interface
  - Router interface is a connector (often physical)
- Must specify direction of traffic (inbound, outbound)
- At most one ACL per interface and direction

```
interface ethernet0
  ip access-group 1 in
  ip access-group 2 out
```

# Forwarding pipeline with ACLs



1. Apply inbound ACLs of the incoming interface

2. Send to outgoing interface (lookup forwarding table)

3. Apply outbound ACLs of the outgoing interface

# Why network verification is hard

Say, you want to “simply” ensure that *no* packet can go from interface-A to interface-B on the same router

- Number of possible packets to consider = ???

Things get even more interesting when we consider network wide properties and routing



# Why network verification is hard

