Operating Systems and Networks
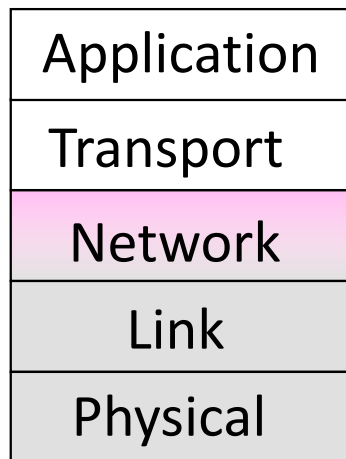
Network Lecture 7: Network Layer 2

Adrian Perrig
Network Security Group
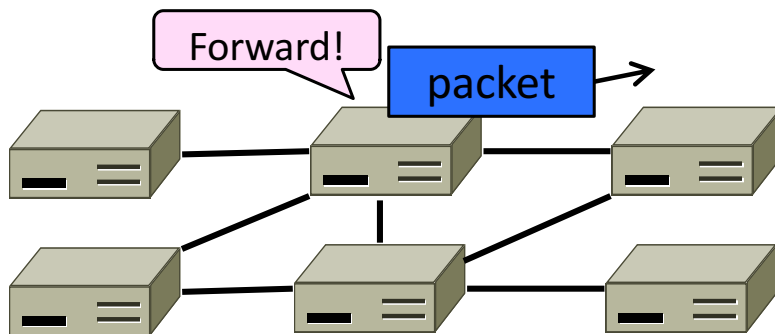ETH Zürich

Where we are in the Course

- More fun in the Network Layer!
  - We've covered packet forwarding
  - Now we'll learn about <u>routing</u>

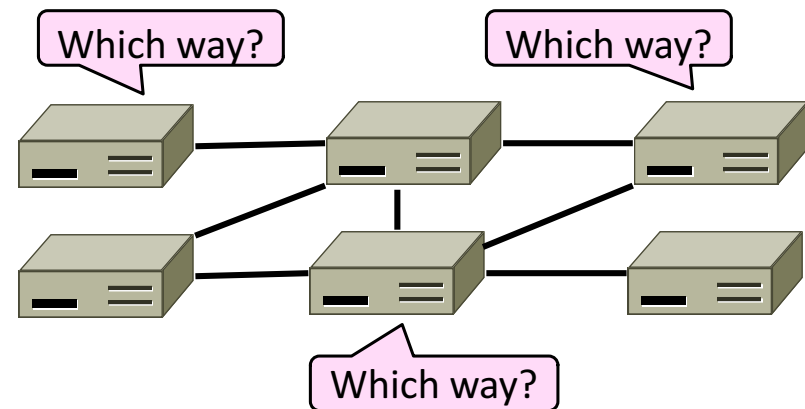| Application |
| :---: |
| Transport |
| Network |
| Link |
| Physical |

Routing versus Forwarding

- <u>Forwarding</u> is the process of sending a packet on its way
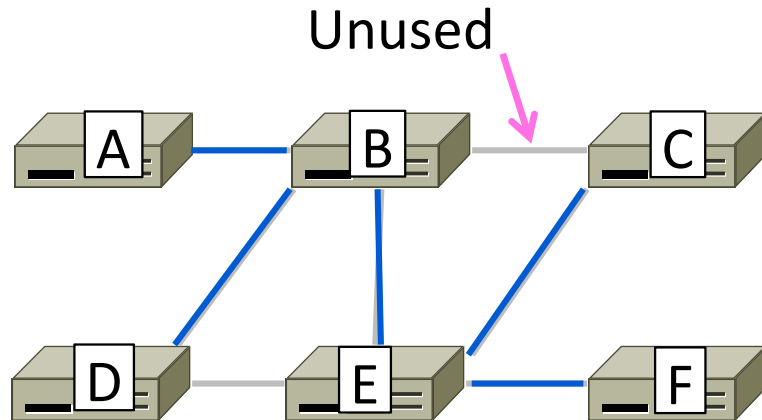
- <u>Routing</u> is the process of deciding in which direction to send traffic
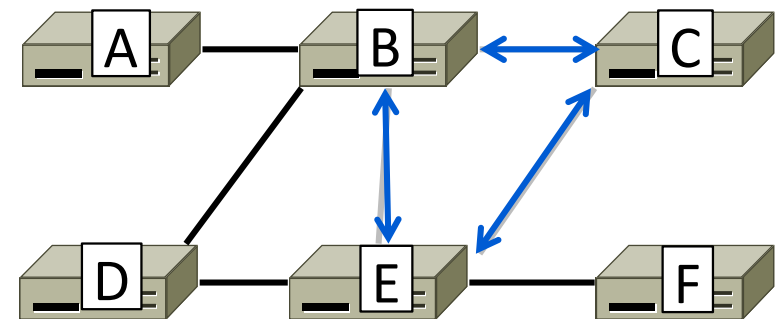
Improving on the Spanning Tree

- 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

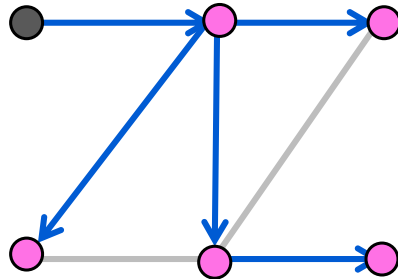| Mechanism	Timescale / Adaptation
Load-sensitive routing	Seconds / Traffic hotspots
Routing	Minutes / Equipment failures
Traffic Engineering	Hours / Network load
Provisioning	Months / Network customers

5

Delivery Models

- Different routing used for different delivery models

Unicast
(§5.2)
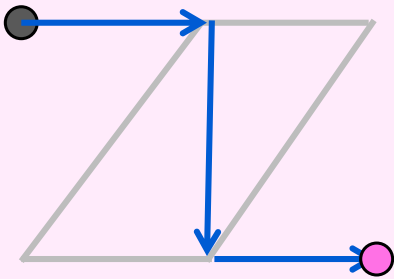
Broadcast
(§5.2.7)
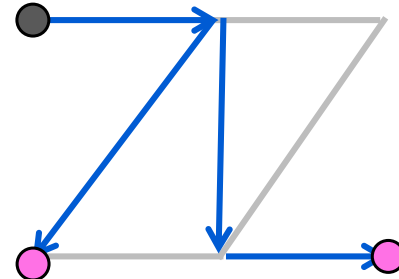
Multicast
(§5.2.8)

Anycast
(§5.2.9)
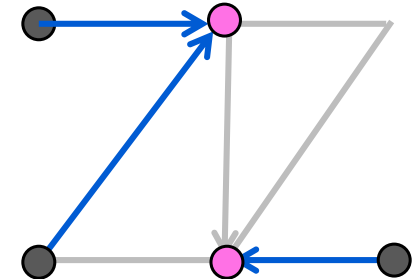
Goals of Routing Algorithms

- 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
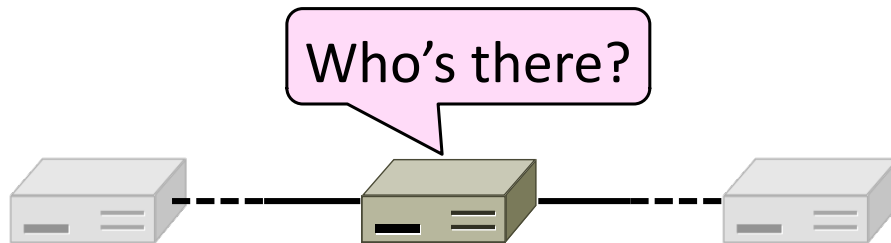
Topics

- IPv4, IPv6, NATs and all that   } Last time

- Shortest path routing
- Distance Vector routing
- Flooding
- Link-state routing
- Equal-cost multi-path
- Inter-domain routing (BGP)

This time

Shortest Path Routing (§5.2.1-5.2.2)

- Defining "best" paths with link costs
  - These are <u>shortest path</u> routes
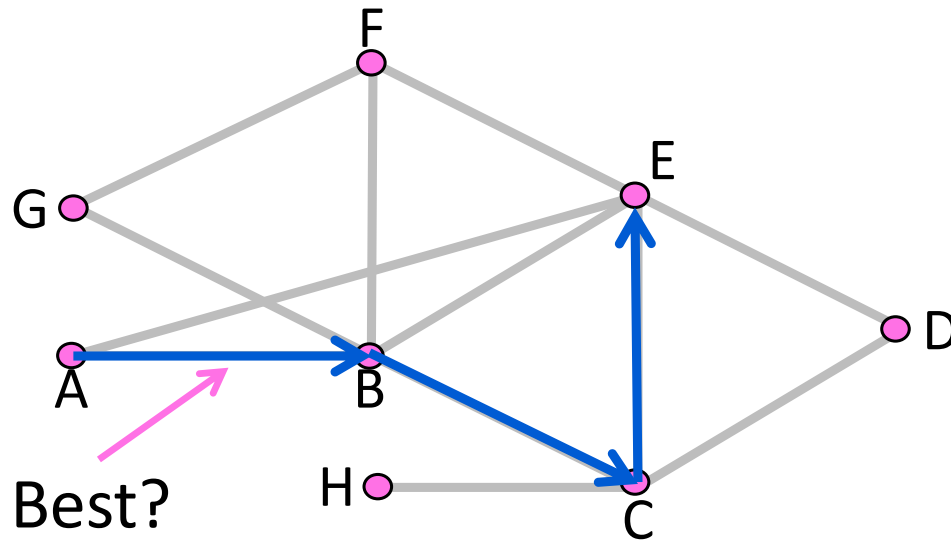
Best?

What are "Best" paths anyhow?

- Many possibilities:
  - Latency, avoid circuitous paths
  - Bandwidth, avoid small pipes
  - Money, avoid expensive links
  - Hops, to reduce switching

- But only consider topology
  - Ignore workload, e.g., hotspots
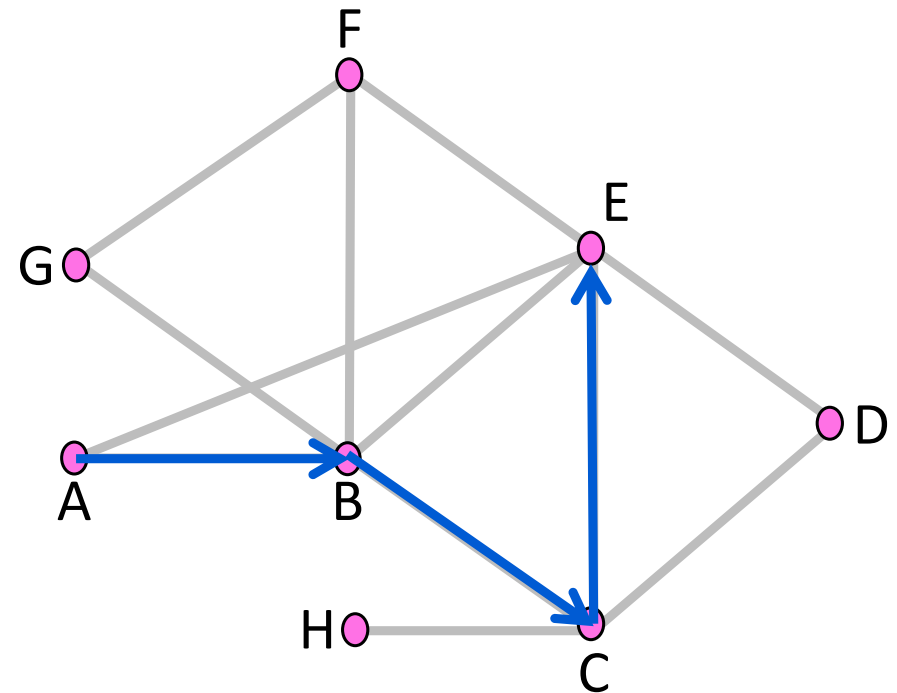
Shortest Paths

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 any 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
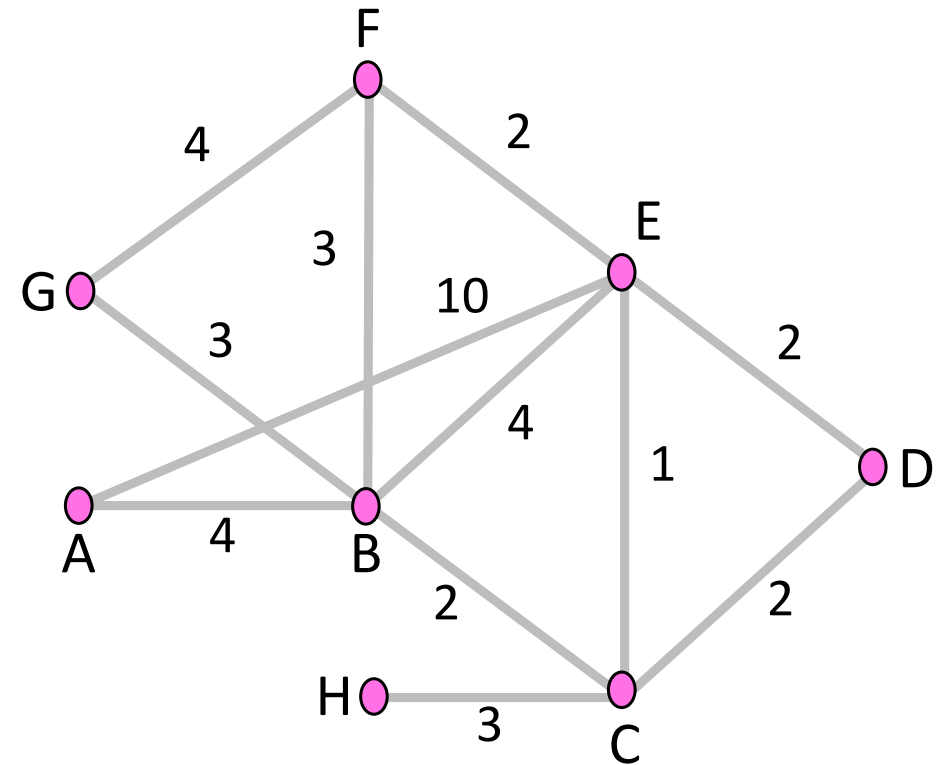
Shortest Paths (3)

- 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
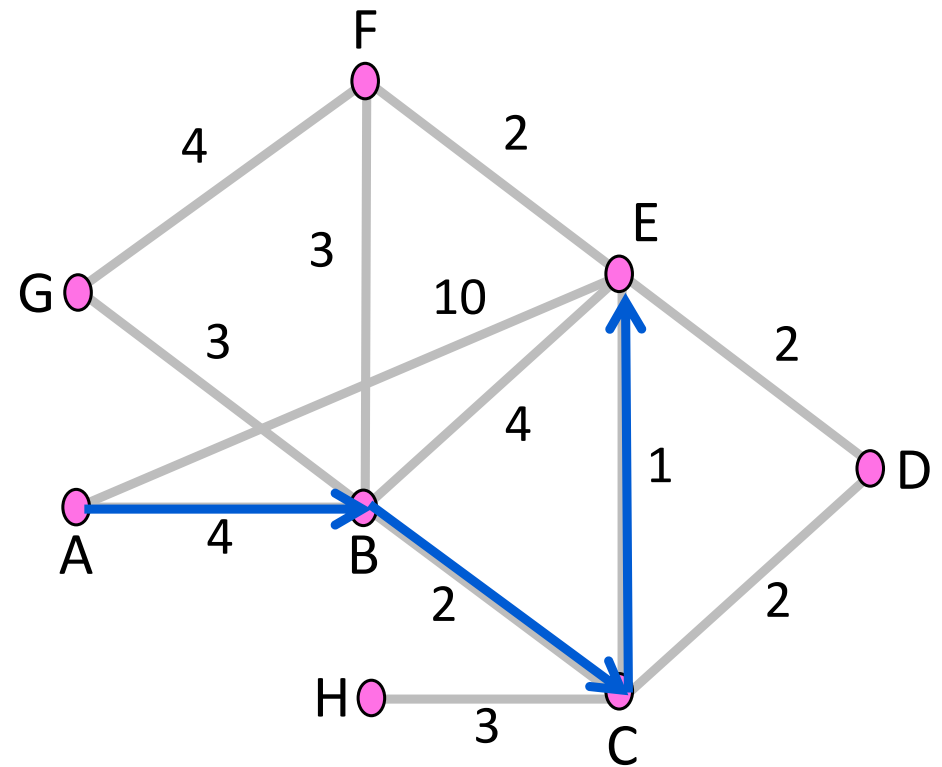
Shortest Paths (4)

- Optimality property:
  - Subpaths of shortest paths are also shortest paths

- ABCE is a shortest path
  → So are ABC, AB, BCE, BC, CE
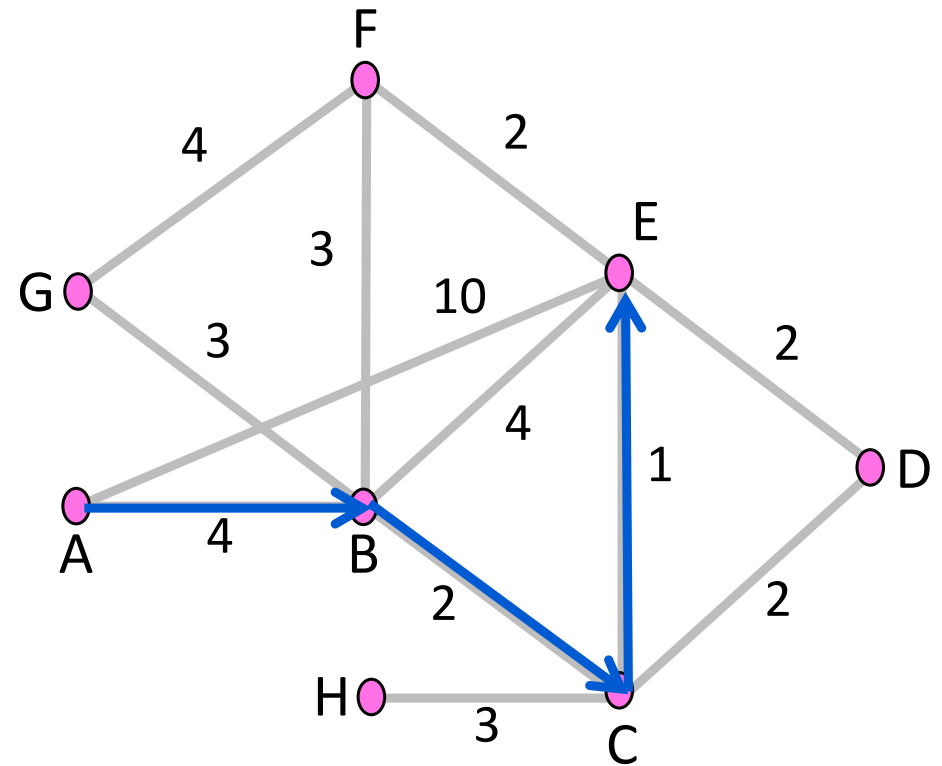
Sink Trees

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

- Find the sink tree for E
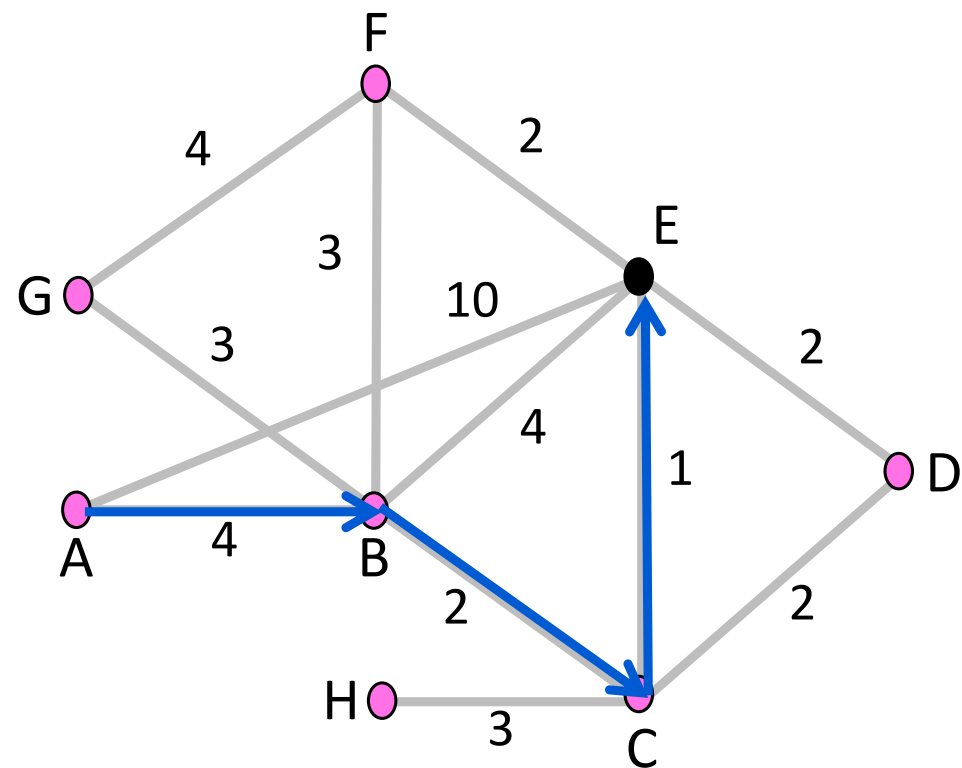
Sink Trees (2)

- 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
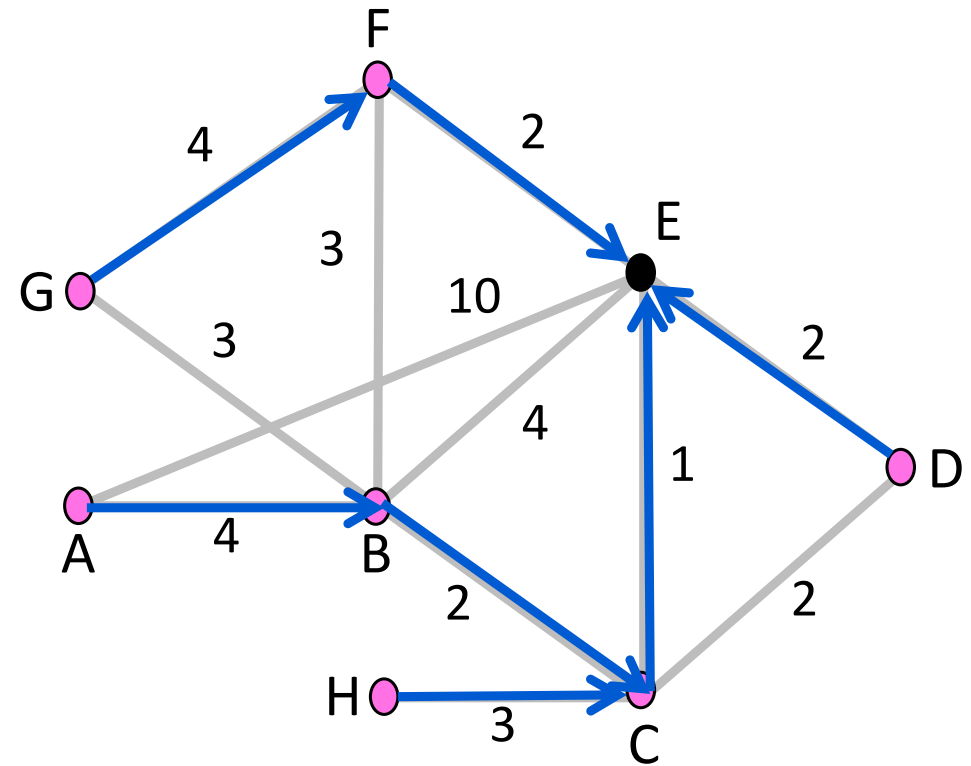
Computing Shortest Paths with Dijkstra (§5.2.2)

- How to compute shortest path given the network topology
  - With Dijkstra's algorithm

Source tree for E
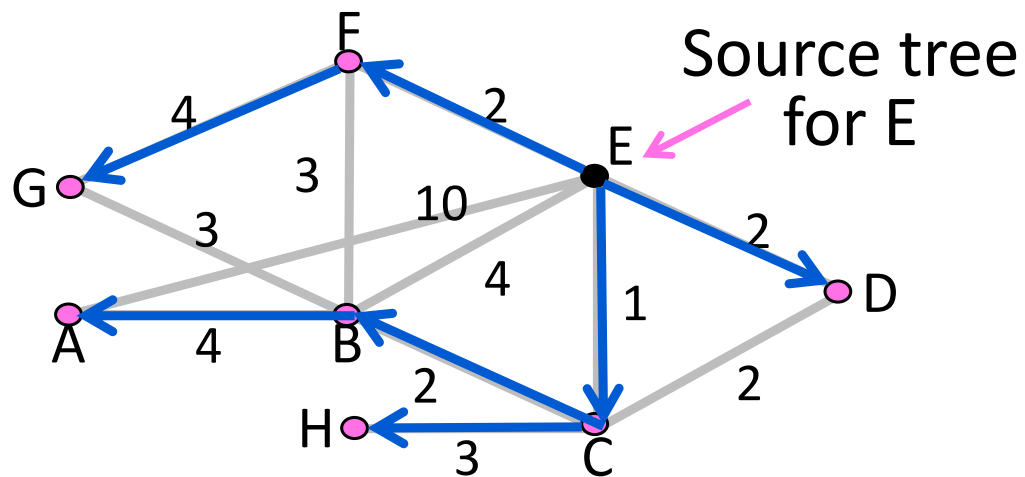
Edsger W. Dijkstra (1930-2002)

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

- Famous computer scientist
  - Programming languages
  - Distributed algorithms
  - Program verification

- Dijkstra's algorithm, 1959
  - Single-source shortest paths, given network with non-negative link costs

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
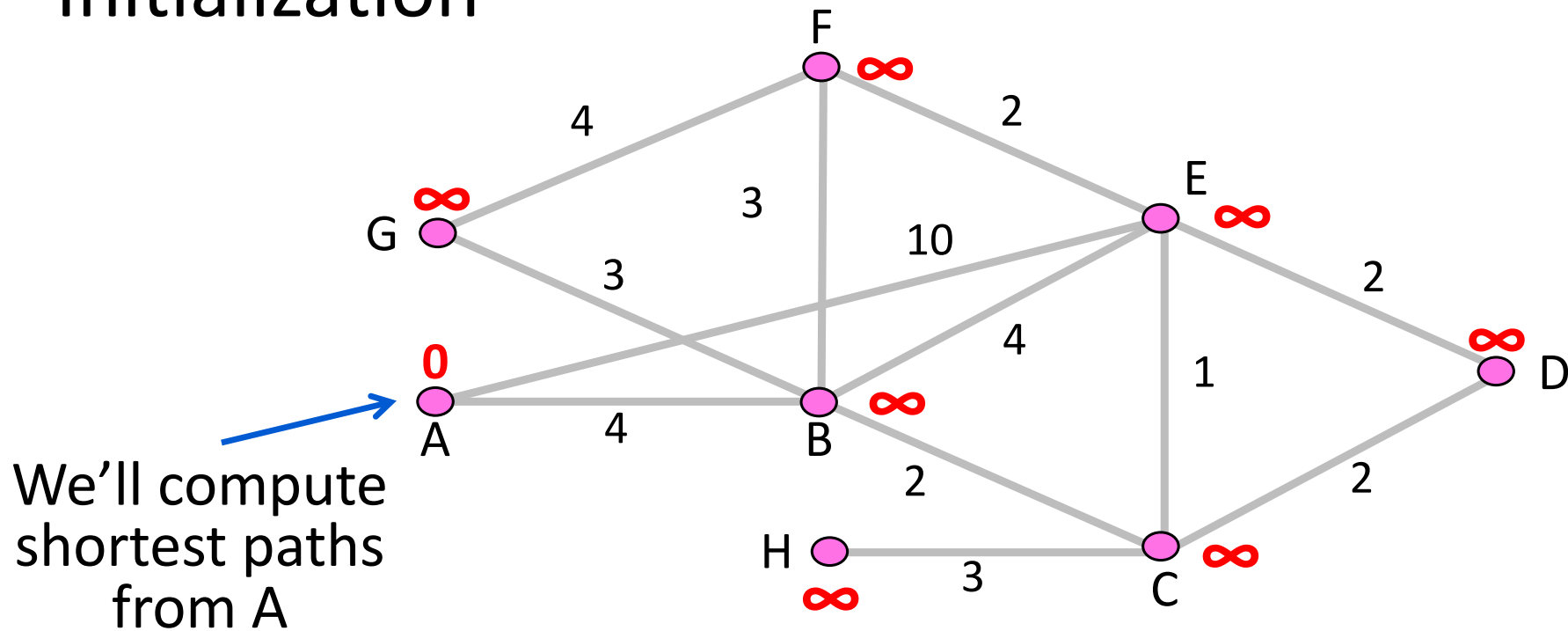
Dijkstra's Algorithm (3)

- Relax around A
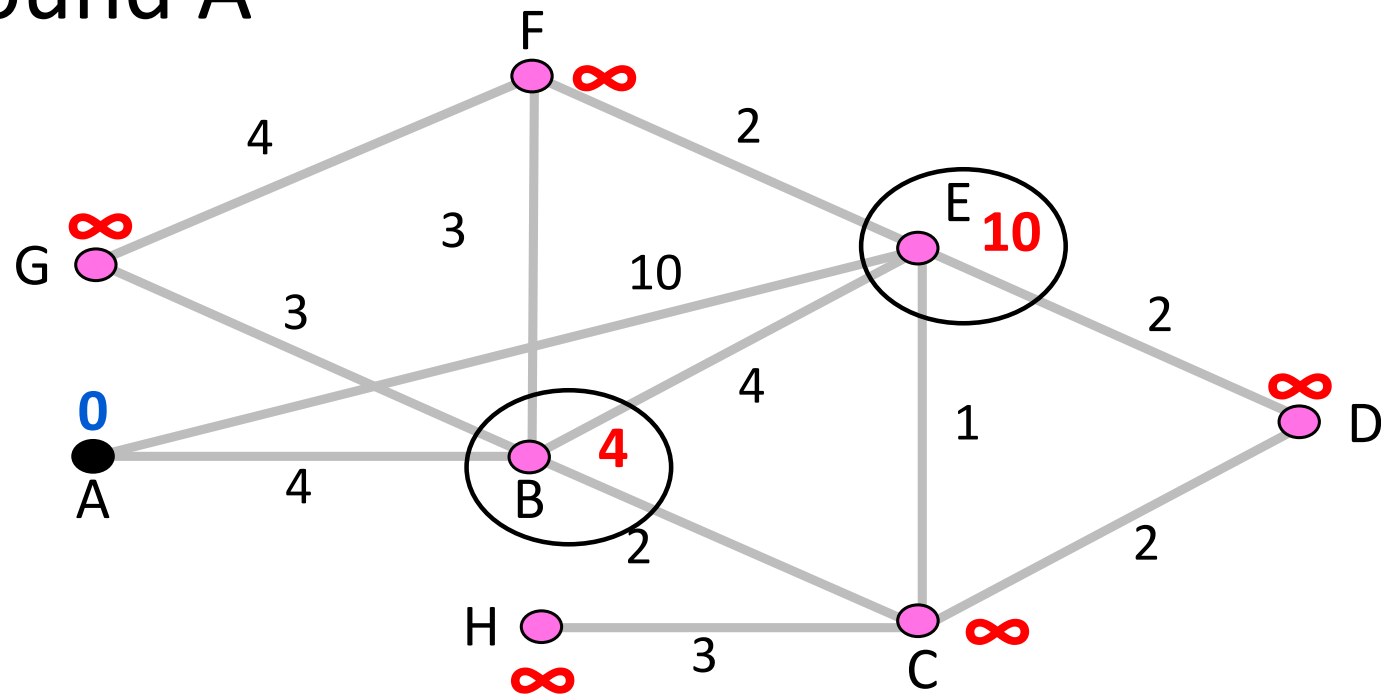
Dijkstra's Algorithm (4)

- Relax around B

Distance fell!
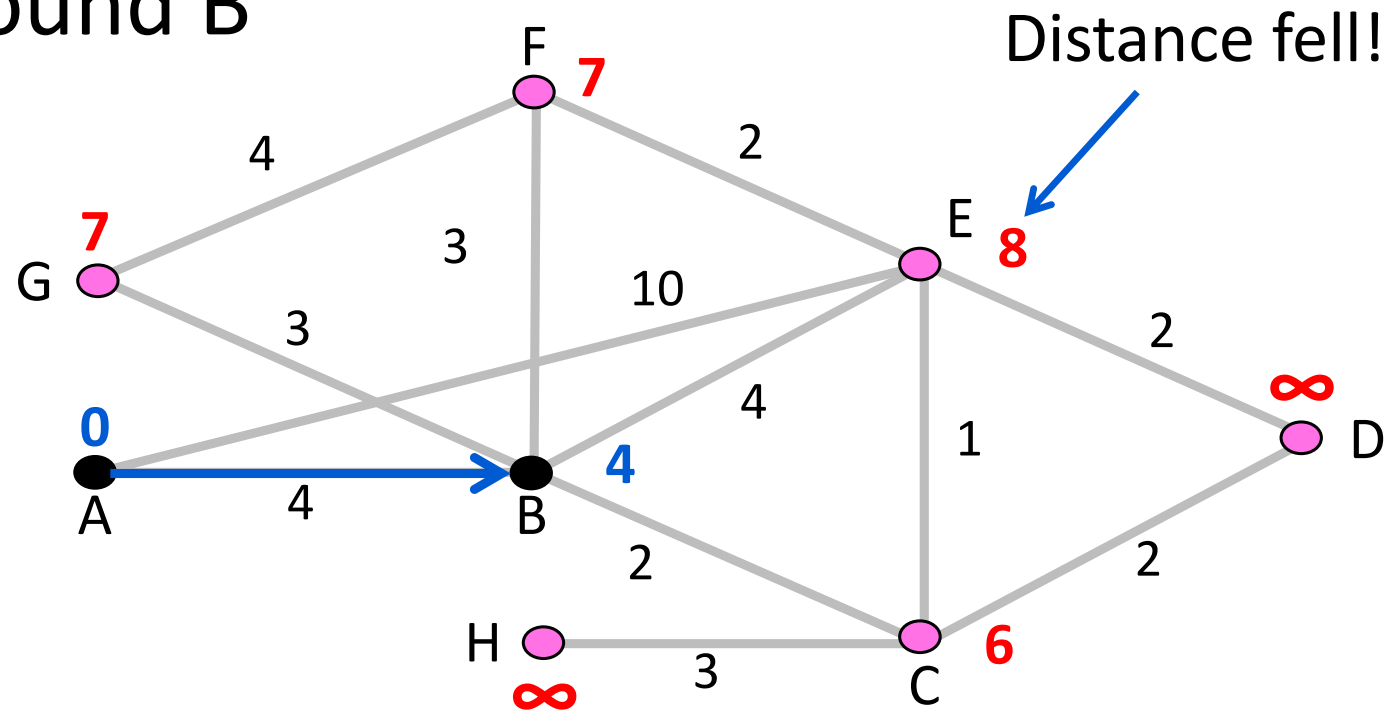
Dijkstra's Algorithm (5)

- Relax around C

Distance fell again!
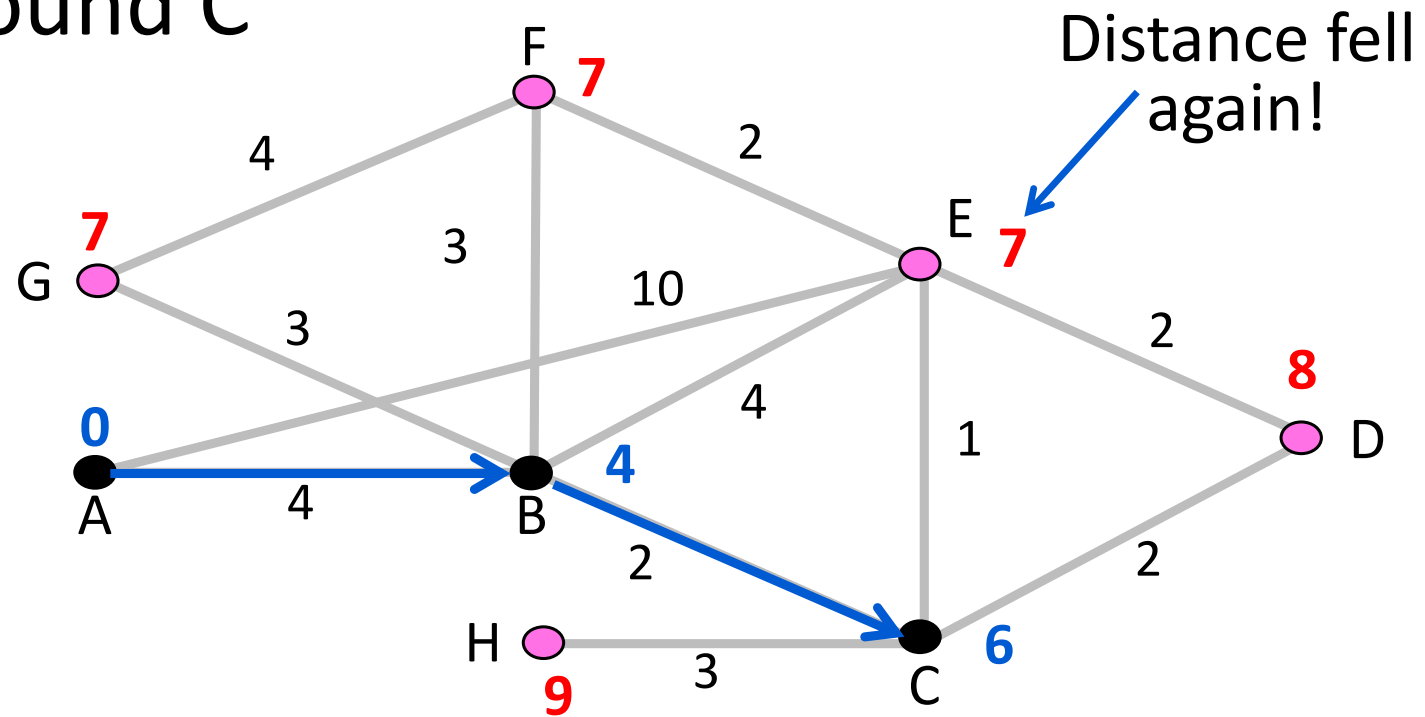
Dijkstra's Algorithm (6)

- Relax around G (say)

Didn't fall …

25
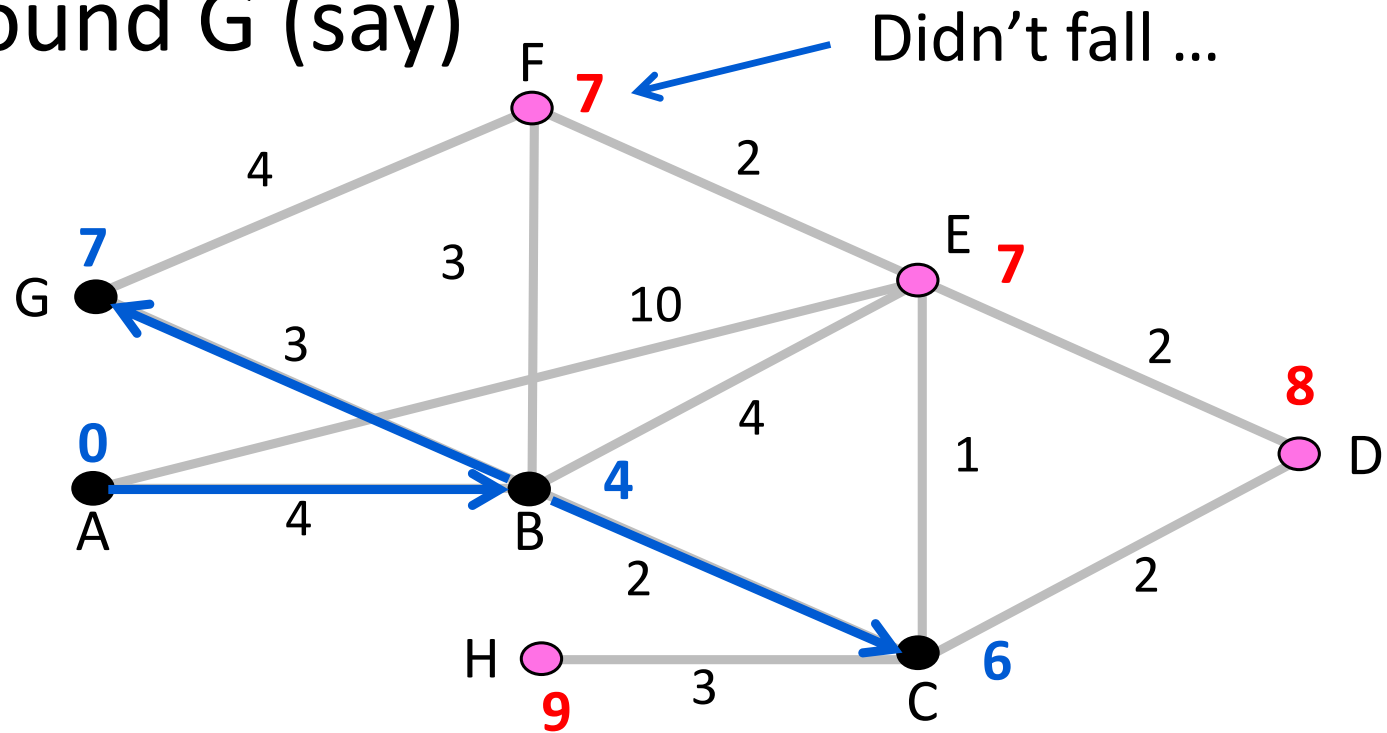
Dijkstra's Algorithm (7)

- Relax around F (say)

Relax has no effect
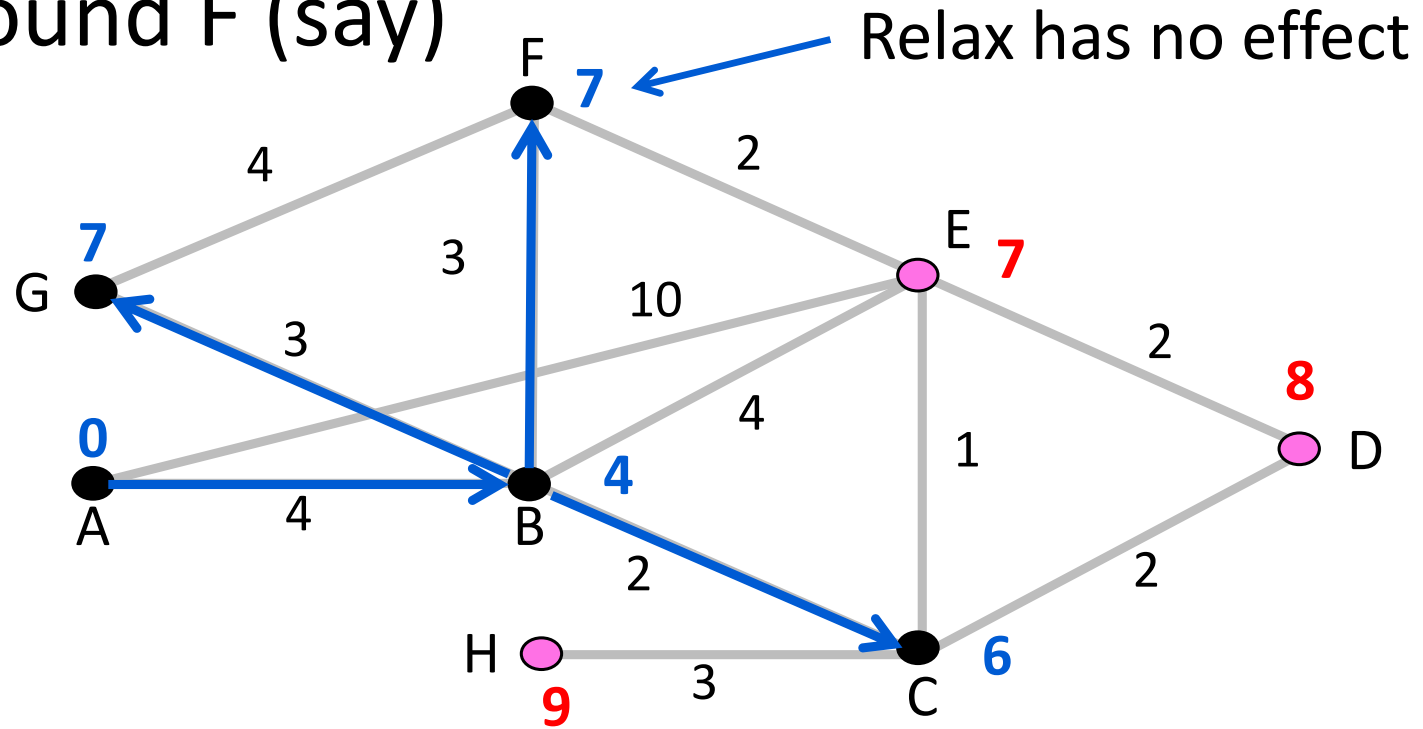
Dijkstra's Algorithm (8)

- Relax around E
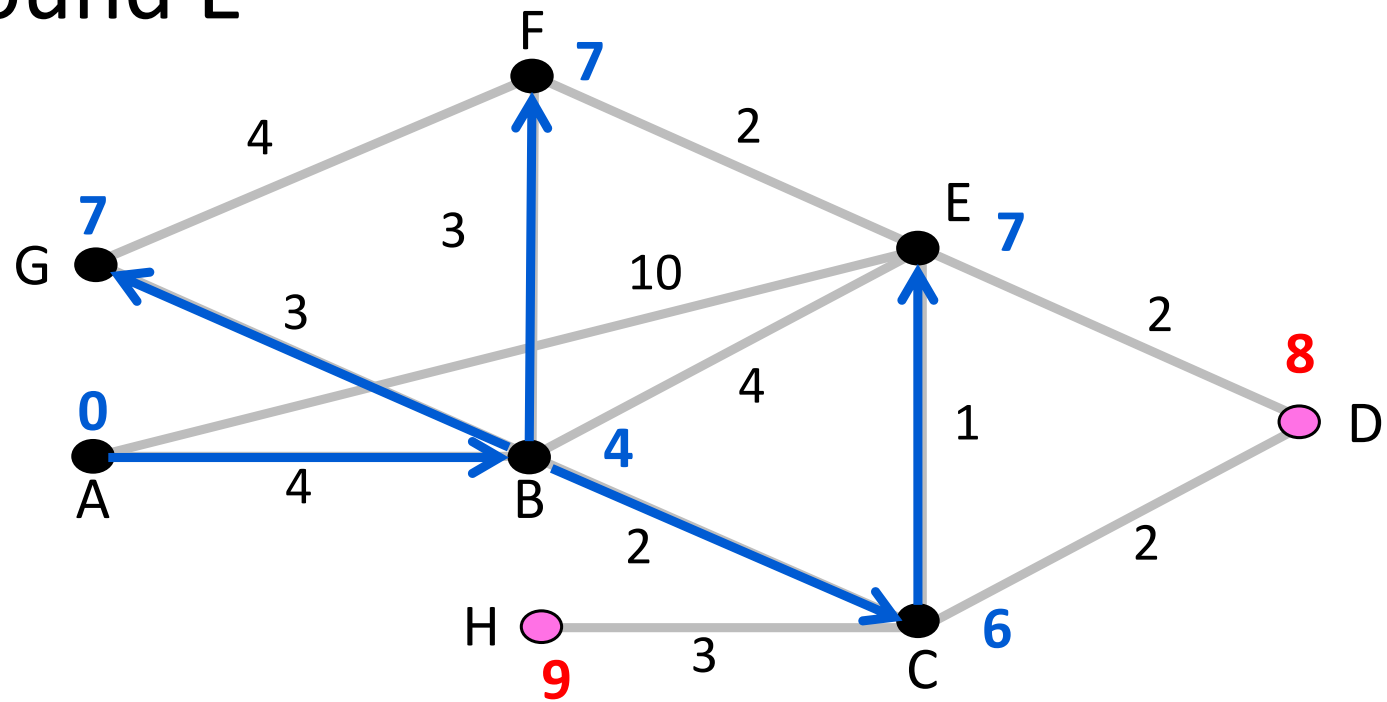
Dijkstra's Algorithm (9)

- Relax around D
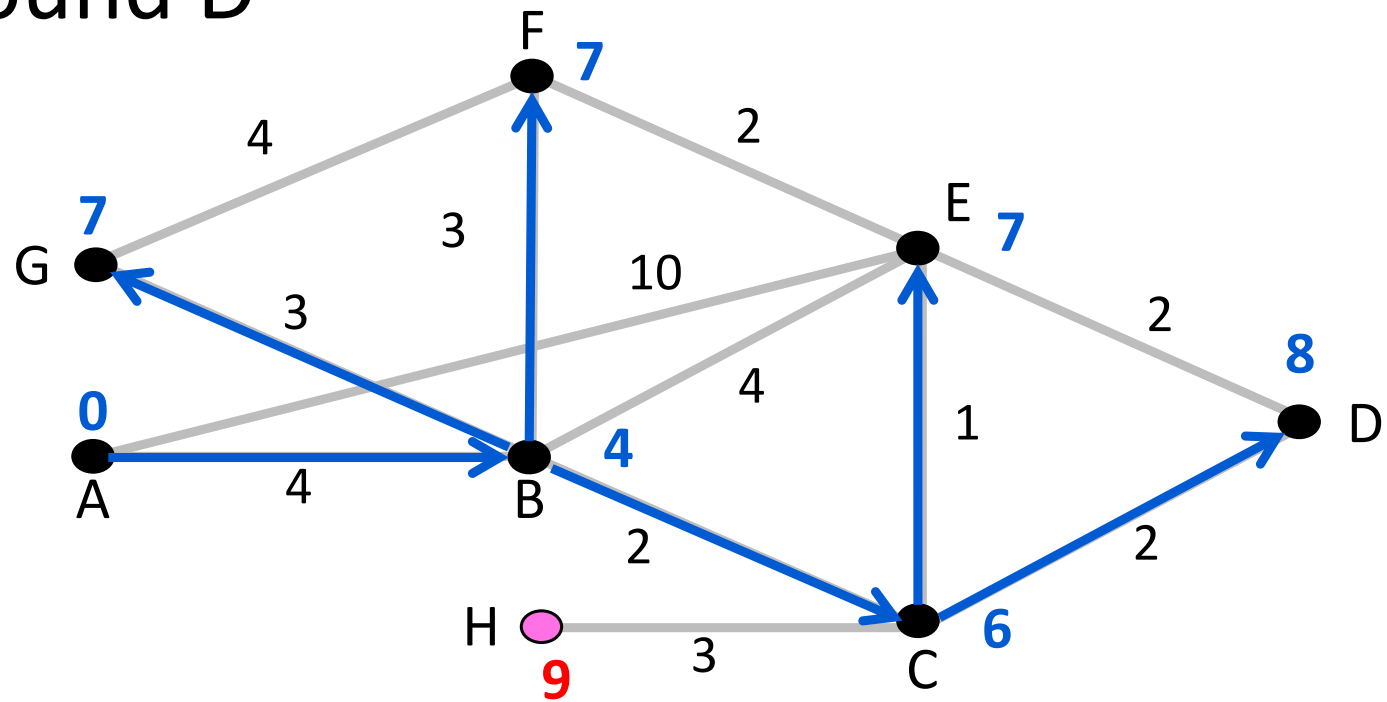
Dijkstra's Algorithm (10)

- Finally, H … done
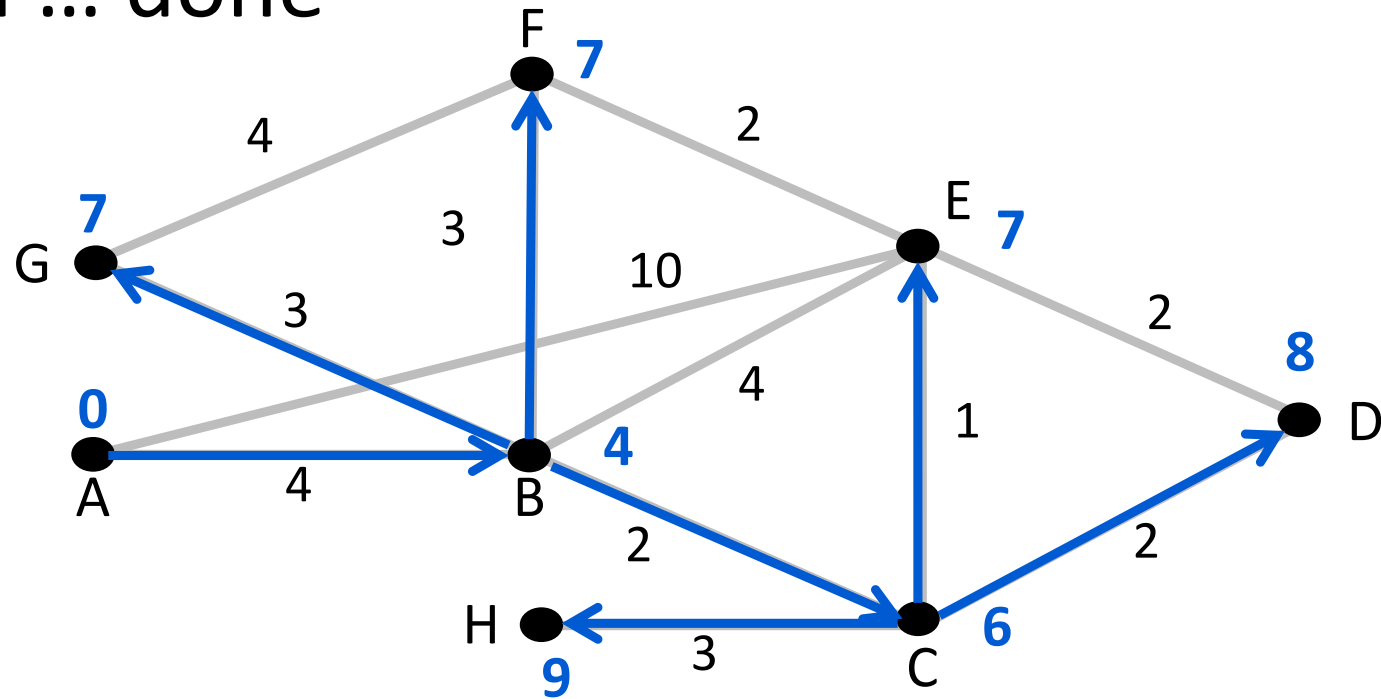
Dijkstra Comments

- Finds shortest paths in order of increasing distance from source
  - Leverages optimality property

- Runtime depends on efficiency 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 (§5.2.4)

- How to compute shortest paths in a distributed network
  - The Distance Vector (DV) approach
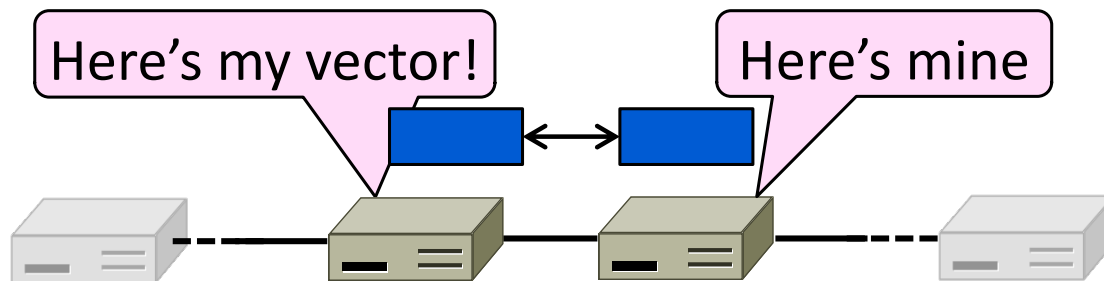
Distance Vector Routing

- Simple, early routing approach
  - Used in ARPANET, and RIP (Routing Information Protocol)

- 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 communicate only with 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
   – Use the best neighbor for forwarding

Distance Vector Example

- Consider a simple network. Each node runs on its own
  - E.g., node A can only talk to nodes B and D

A's initial vector →

To	Cost
A	0
B	∞
C	∞
D	∞
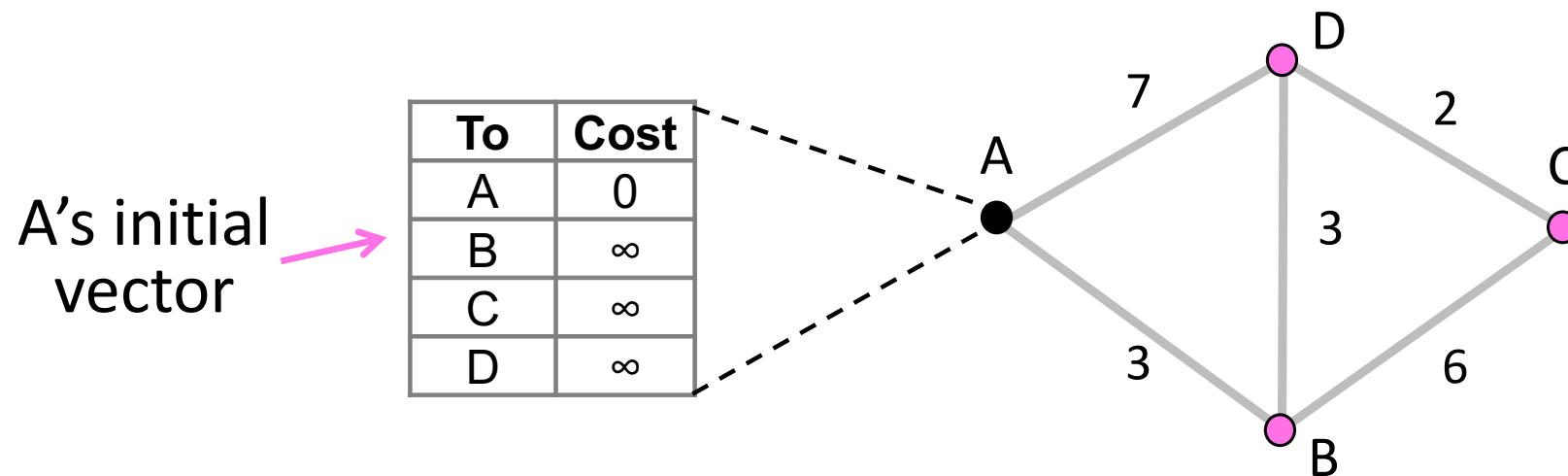
DV Example (2)

- First exchange, A hears from B, D and finds 1-hop routes
  - A always learns min(B+3, D+7)

To	B says	D says
A	∞	∞
B	0	∞
C	∞	∞
D	∞	0

→

B +3	D +7
∞	∞
3	∞
∞	∞
∞	7

→

A learns Cost	Next
0	--
3	B
∞	--
7	D

▮ = learned better route
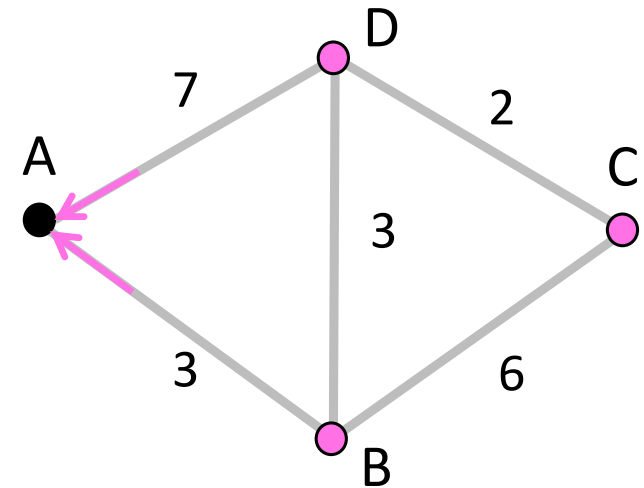
DV Example (3)

- First exchange for all nodes to find best 1-hop routes
  - E.g., B learns min(A+3, C+6, D+3)

To	A says	B says	C says	D says
A	0	∞	∞	∞
B	∞	0	∞	∞
C	∞	∞	0	∞
D	∞	∞	∞	0

→

A learns Cost	Next	B learns Cost	Next	C learns Cost	Next	D learns Cost	Next
0	--	3	A	∞	--	7	A
3	B	0	--	6	B	3	B
∞	--	6	C	0	--	2	C
7	D	3	D	2	D	0	--

▇ = learned better route
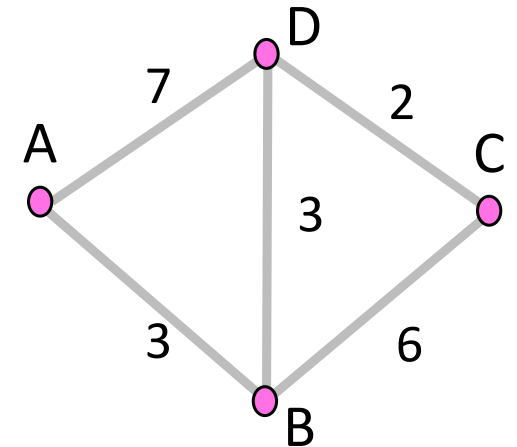
DV Example (4)

- Second exchange for all nodes to find best 2-hop routes

To	A says	B says	C says	D says
A	0	3	∞	7
B	3	0	6	3
C	∞	6	0	2
D	7	3	2	0

→

A learns		B learns		C learns		D learns	
Cost	Next	Cost	Next	Cost	Next	Cost	Next
0	--	3	A	9	B	6	B
3	B	0	--	5	D	3	B
9	D	5	D	0	--	2	C
6	B	3	D	2	D	0	--

■ = learned better route

38
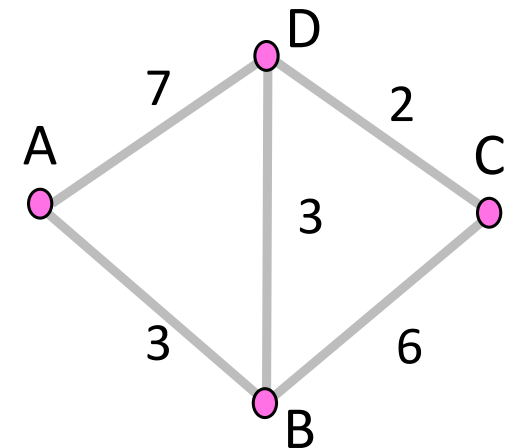
DV Example (5)

- Third exchange for all nodes to find best 3-hop routes

To	A says	B says	C says	D says
A	0	3	9	6
B	3	0	5	3
C	9	5	0	2
D	6	3	2	0

→

A learns		B learns		C learns		D learns	
Cost	Next	Cost	Next	Cost	Next	Cost	Next
0	--	3	A	8	D	6	B
3	B	0	--	5	D	3	B
8	B	5	D	0	--	2	C
6	B	3	D	2	D	0	--

▮ = learned better route

39
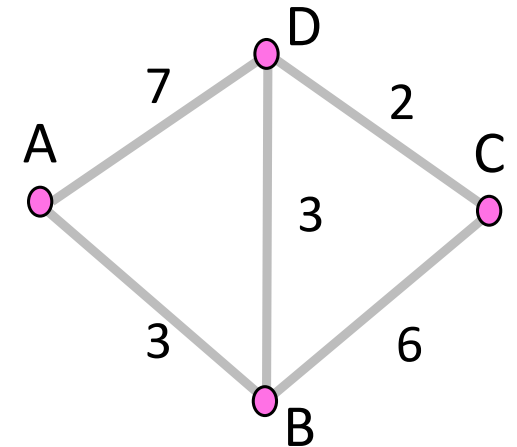
DV Example (5)

- Fourth and subsequent exchanges; converged

To	A says	B says	C says	D says
A	0	3	8	6
B	3	0	5	3
C	8	5	0	2
D	6	3	2	0

→

A learns Cost	Next	B learns Cost	Next	C learns Cost	Next	D learns Cost	Next
0	--	3	A	8	D	6	B
3	B	0	--	5	D	3	B
8	B	5	D	0	--	2	C
6	B	3	D	2	D	0	--

= learned better route
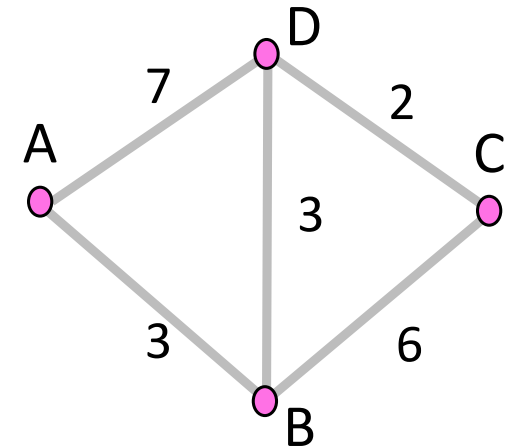
Distance Vector Dynamics

- Adding routes:
  - News travels one hop per exchange

- Removing routes
  - When a node fails, no more exchanges, other nodes forget

- But <u>partitions</u> (unreachable nodes in divided network) are a problem
  - "Count to infinity" scenario

DV Dynamics (2)

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

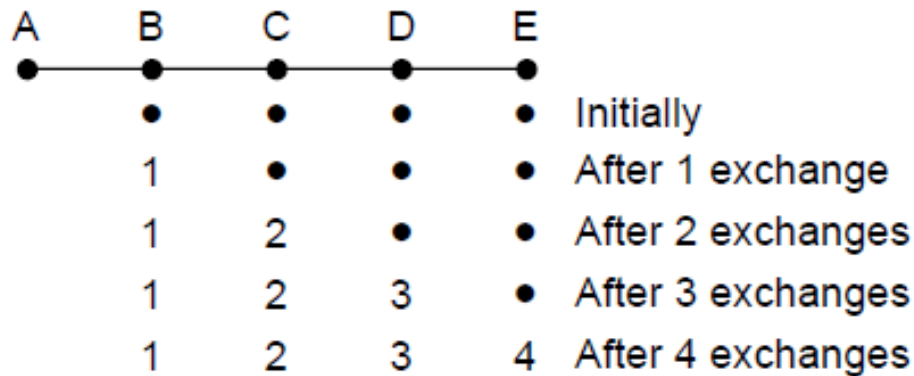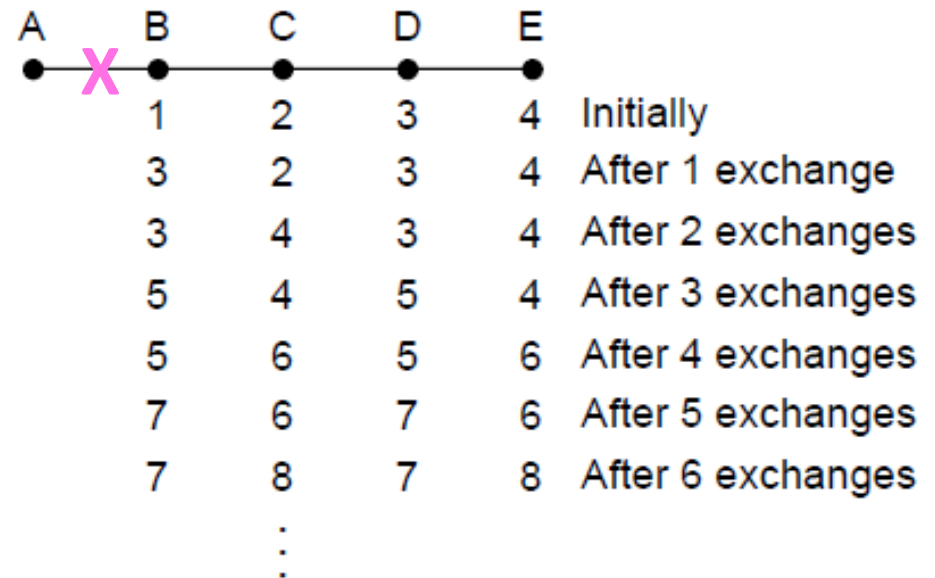| A	B	C	D	E	
●	●	●	●	●	
	●	●	●	●	Initially
1	●	●	●	●	After 1 exchange
1	2	●	●	●	After 2 exchanges
1	2	3	●	●	After 3 exchanges
1	2	3	4		After 4 exchanges

Desired convergence

A		B	C	D	E	
●	X	●	●	●	●	
		1	2	3	4	Initially
		3	2	3	4	After 1 exchange
		3	4	3	4	After 2 exchanges
		5	4	5	4	After 3 exchanges
		5	6	5	6	After 4 exchanges
		7	6	7	6	After 5 exchanges
		7	8	7	8	After 6 exchanges

$\vdots$

"Count to infinity" scenario

DV Dynamics (3)

- Various heuristics to address
  - e.g., "Split horizon, poisoned reverse"
    - Split horizon: omit routes learned from neighbor in updates sent to neighbor
    - Poisoned reverse: set metric = ∞ for routes learned from neighbor sent back to that neighbor
- But none are very effective
  - Can you come up with topology where split horizon / poisoned reverse does not work well?
  - Link state now favored in practice in intra-domain (LAN) settings
  - 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 secs
  - Runs on top of UDP
  - Timeout in 180 secs to detect failures

- RIPv1 specified in RFC1058 (1988)

Flooding (§5.2.3)

- How to broadcast a message to all nodes in the network with <u>flooding</u>
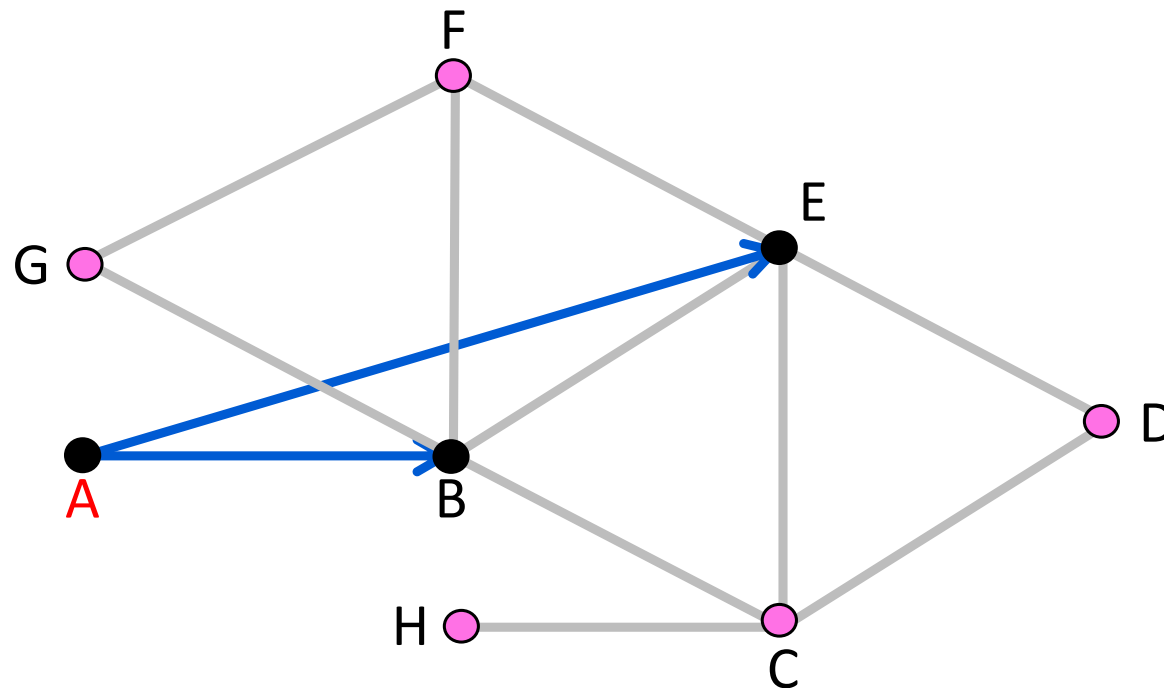 - Simple mechanism, but inefficient
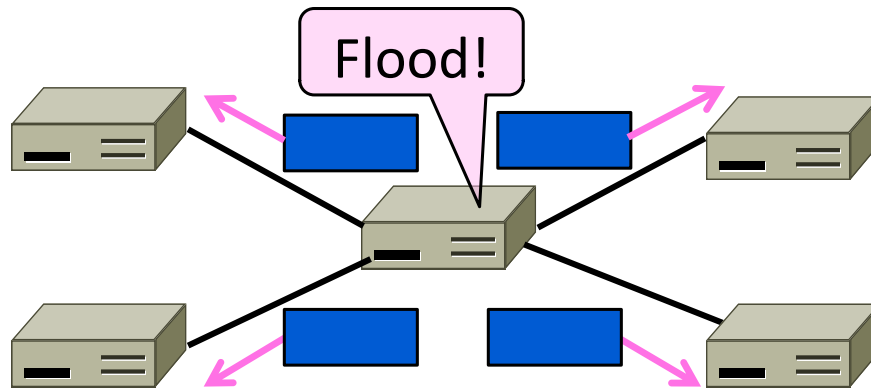
Flooding

- Rule used at each node:
  - Sends an incoming message on to all other neighbors
  - Remember the message so that it is only sent once over each link (called duplicate suppression)

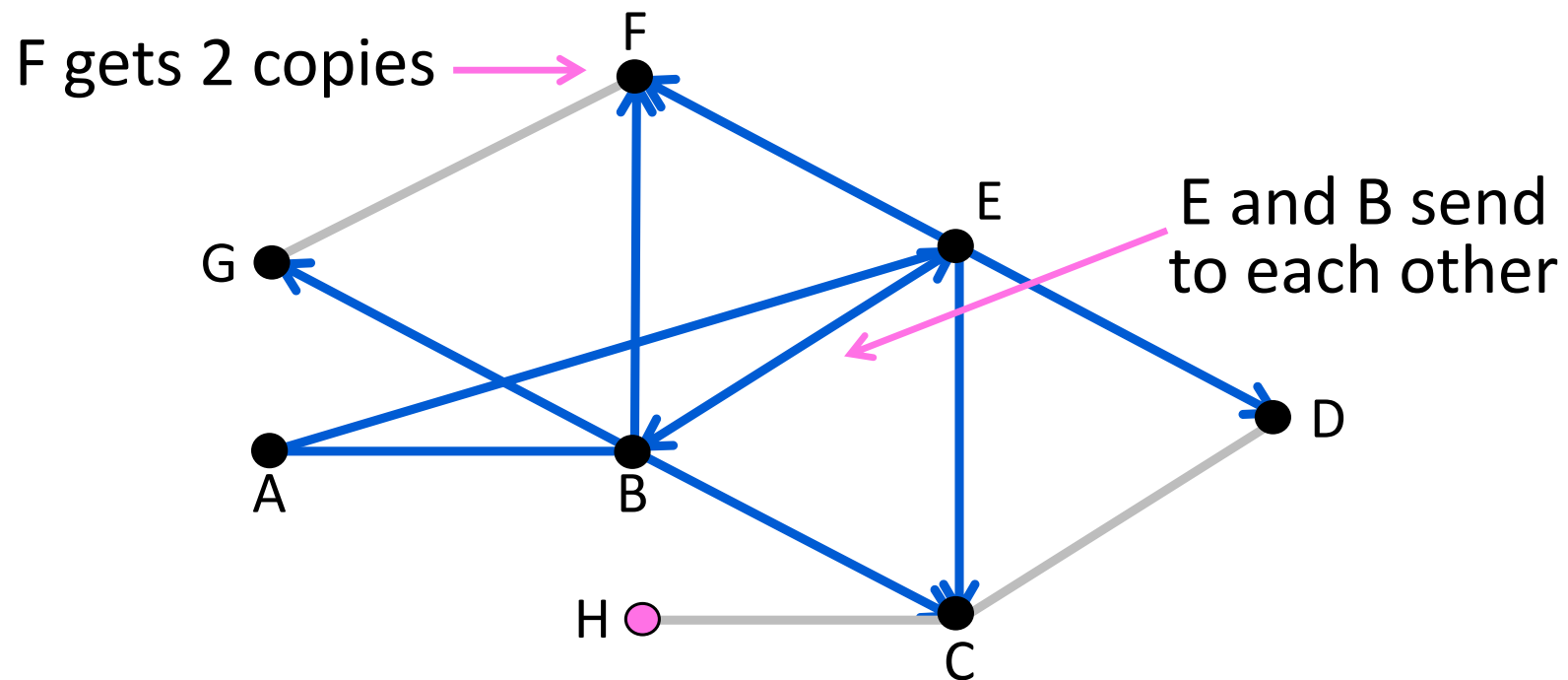- Inefficient because one node may receive multiple copies of message

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

E and B send to each other
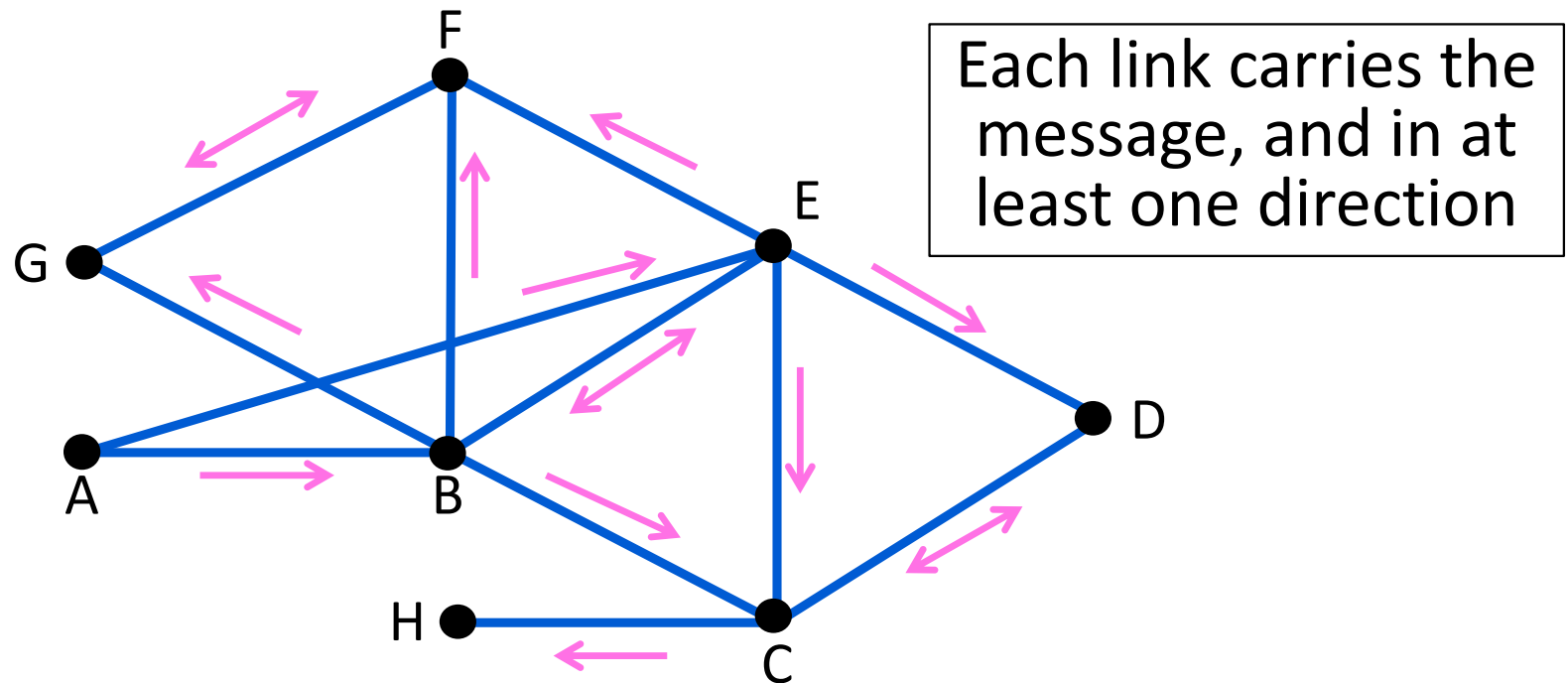
F
E
G
A
B
H
C
D

Flooding (4)

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

F gets another copy
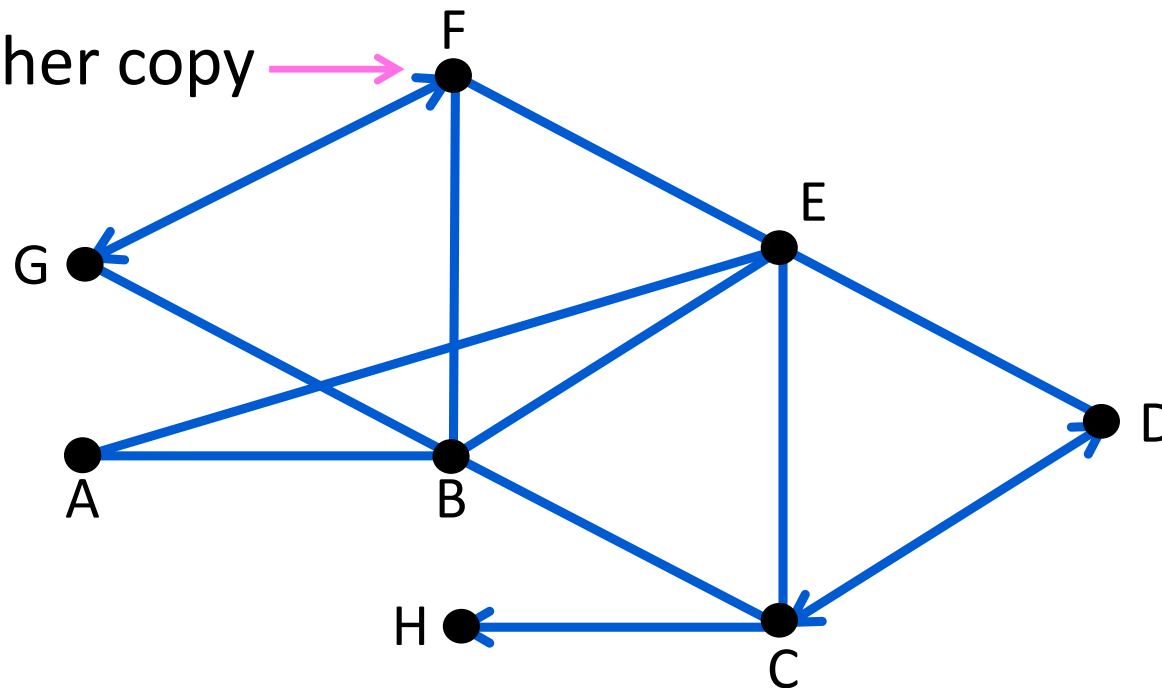
Flooding (5)

- 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
  - Used for duplicate suppression, so same message is only sent once to neighbors
  - So subsequent message (with higher sequence number) will again be flooded

- To make flooding reliable, use ARQ
  - So receiver acknowledges, and sender resends if needed

Link State Routing (§5.2.5, 5.6.6)

- How to compute shortest paths in a distributed network
  - The Link-State (LS) approach
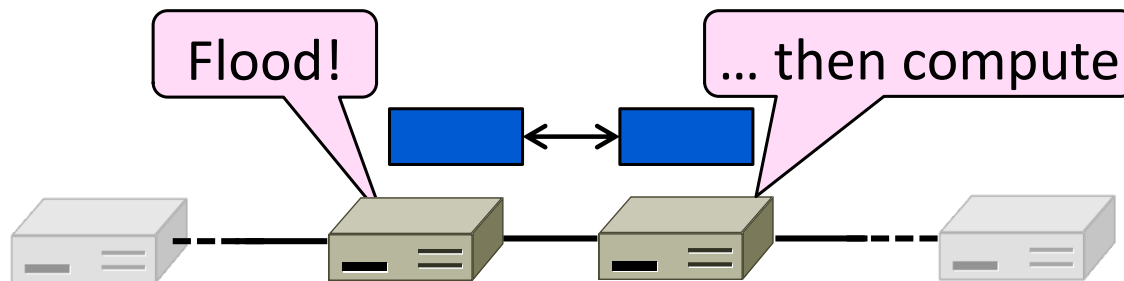
Link-State Routing

- 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 for intra-domain routing

Link-State Setting

Each node computes their forwarding table in the same distributed setting as distance vector:

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

Link-State Algorithm
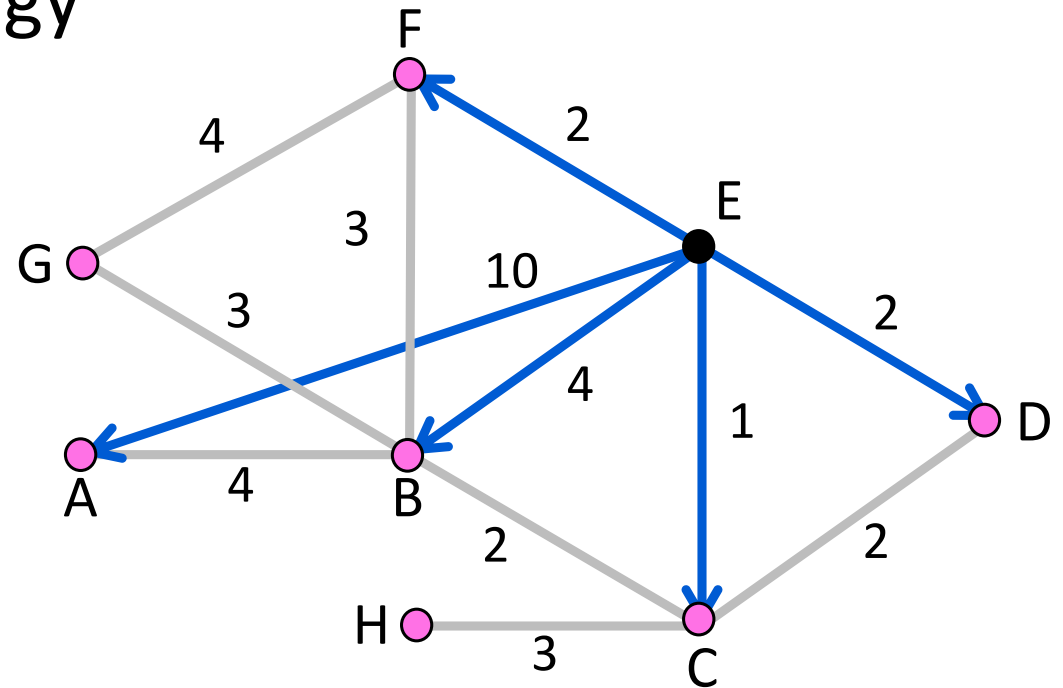
Proceeds in two phases:

1. Nodes <u>flood</u> 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)

Phase 1: Topology Dissemination

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

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

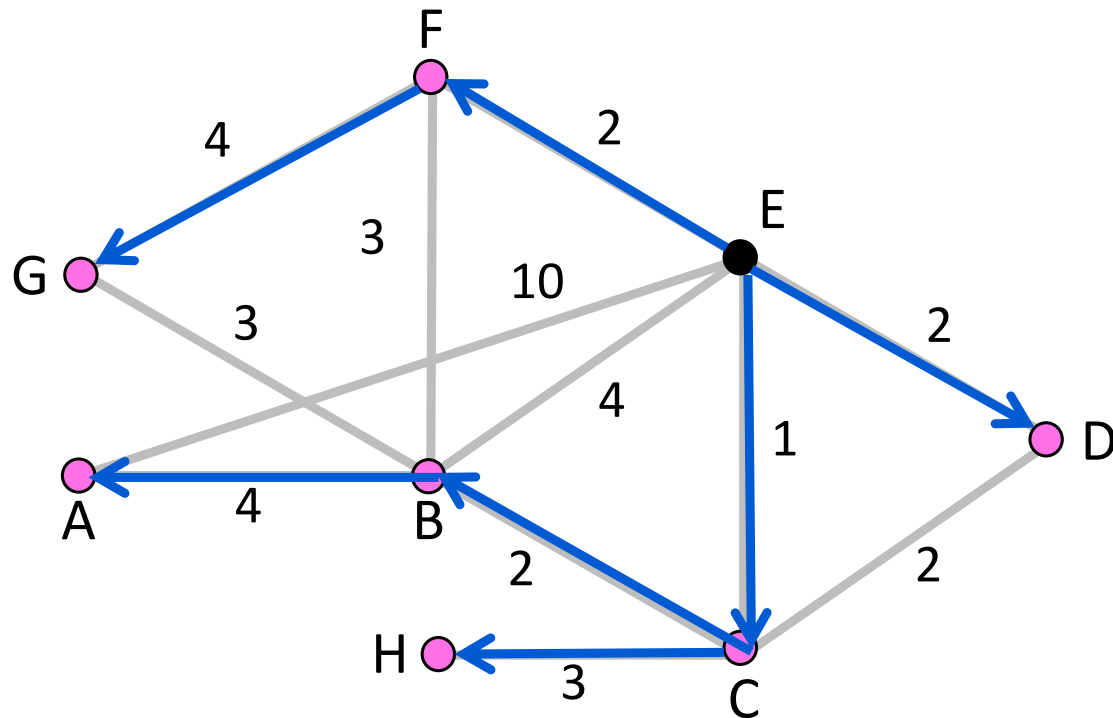| Seq. #	
A	10
B	4
C	1
D	2
F	2

Phase 2: Route Computation

- Each node has full topology
  - By combining all LSPs

- Each node simply runs Dijkstra
  - Some 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

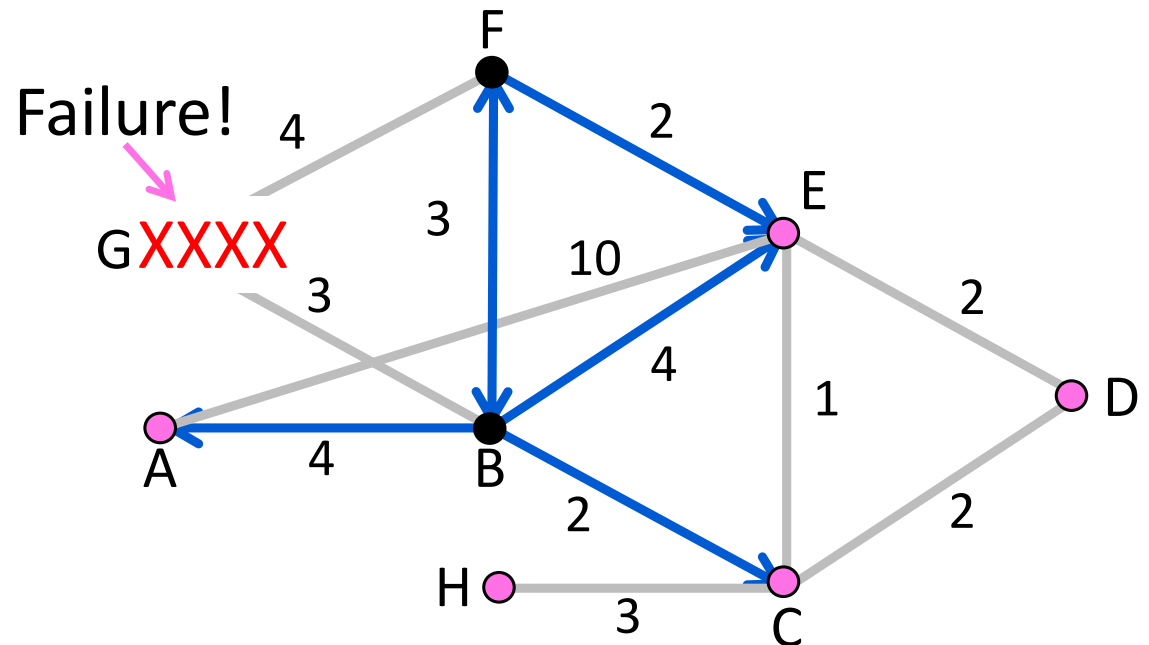| To	Next
A	C
B	C
C	C
D	D
E	--
F	F
G	F
H	C

Handling Changes

- On change, flood updated LSPs, and 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	∞

Failure!

G XXXX

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 (as usual!)

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 (§5.2.1, 5.6.6)

- More on shortest path routes
  - Allow multiple shortest paths

Use ABE as well as ABCE from A→E
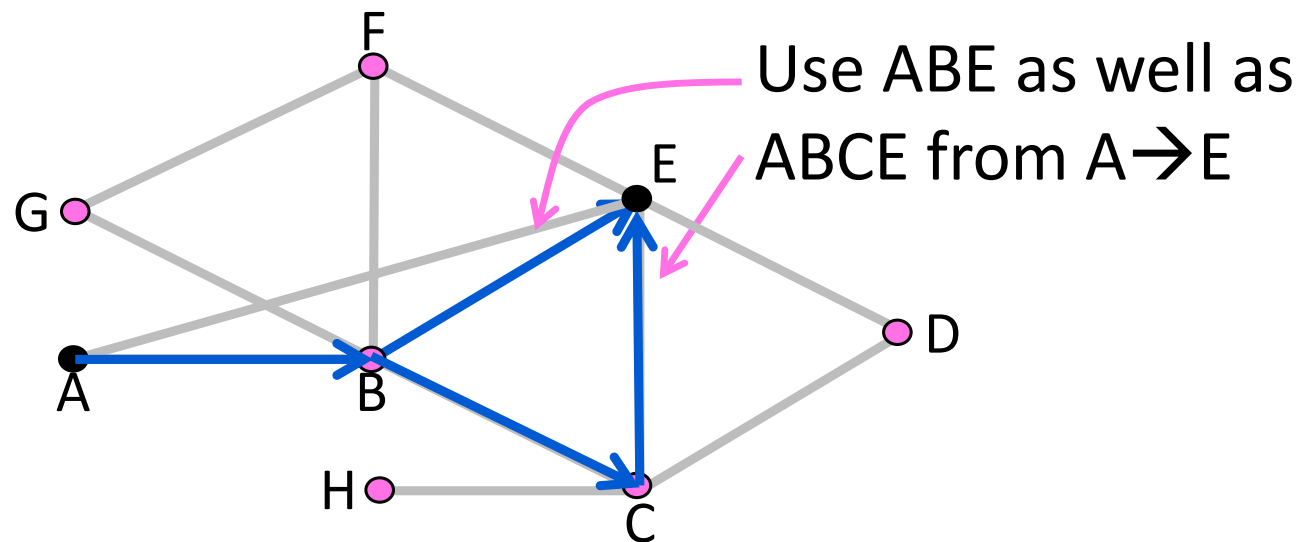
Multipath Routing

- Allow multiple routing paths from node to destination be used at once
  - Topology has them for redundancy
  - Using them can improve performance and reliability

- Questions:
  - How do we find multiple paths?
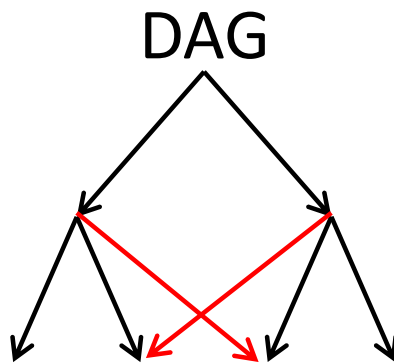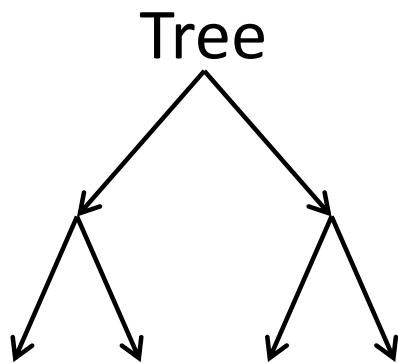 - How do we send traffic along them?

Equal-Cost Multipath Routes

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

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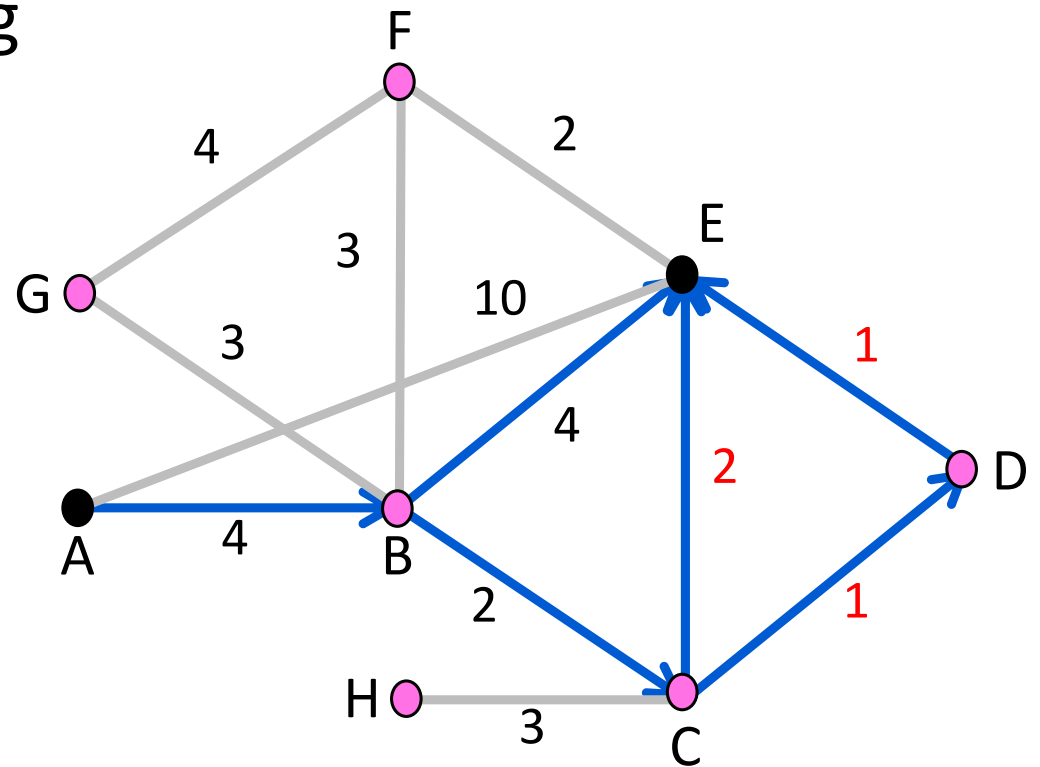
Source "Trees"

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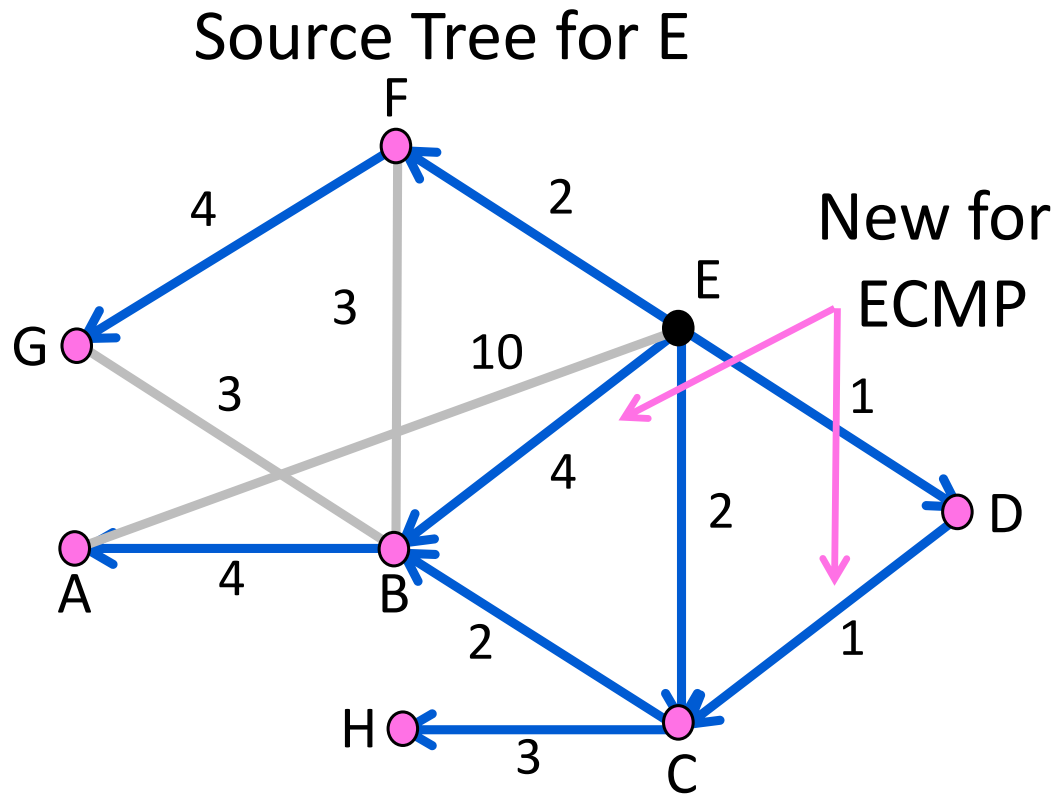
Tree

DAG

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

New for ECMP
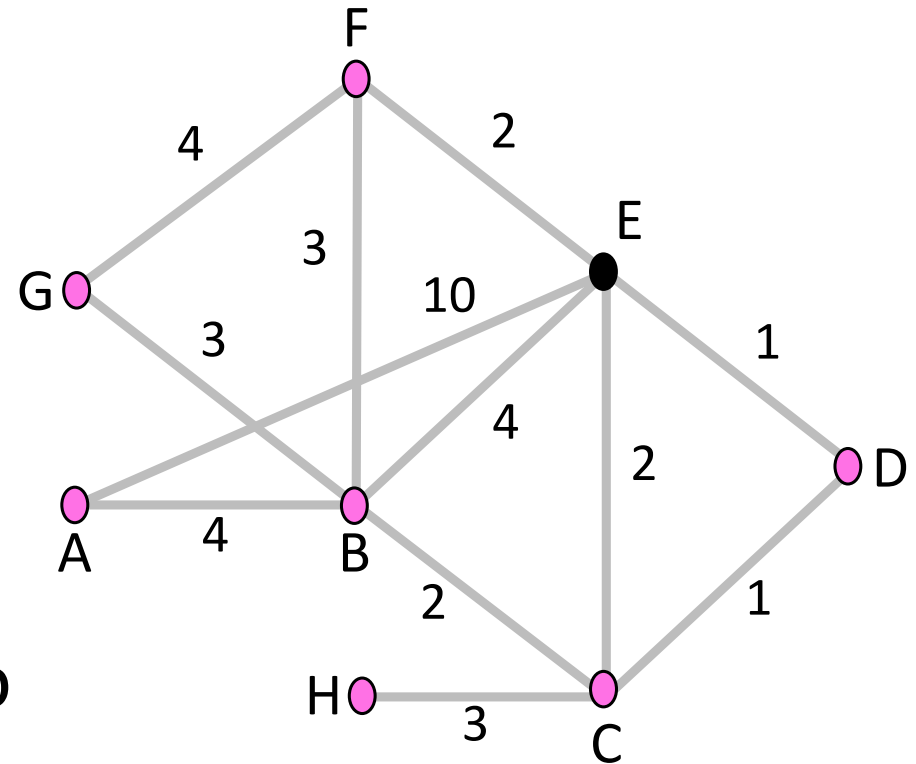
E's Forwarding Table

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 <u>flow</u>

  – 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
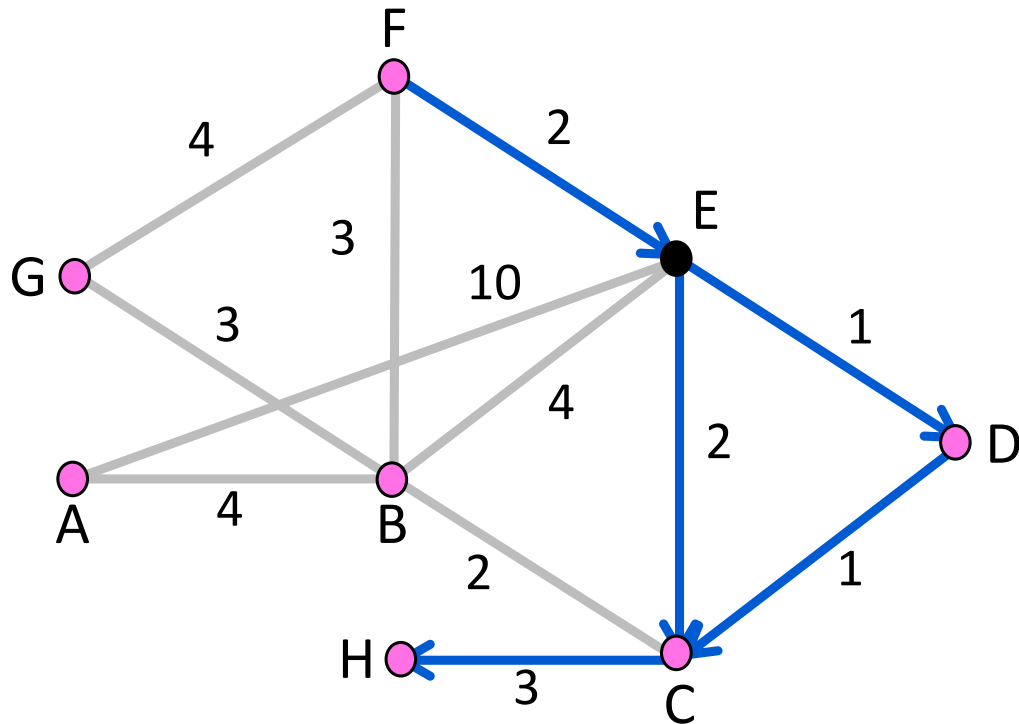
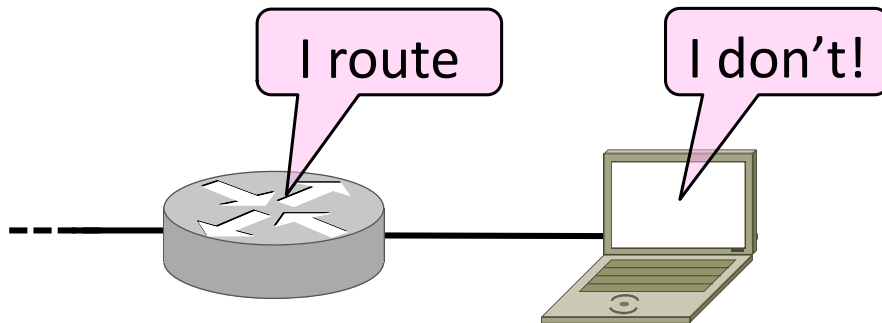## E's Forwarding Choices

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

72

Combining Hosts and Routers

- How routing protocols work with IP
 - The Host/Router distinction

Recap

- In the Internet:
  - Hosts on same network have IP addresses in the same <u>IP prefix</u>
  - Hosts just send off-network traffic to the nearest router to handle

  - Routers discover the routes to use
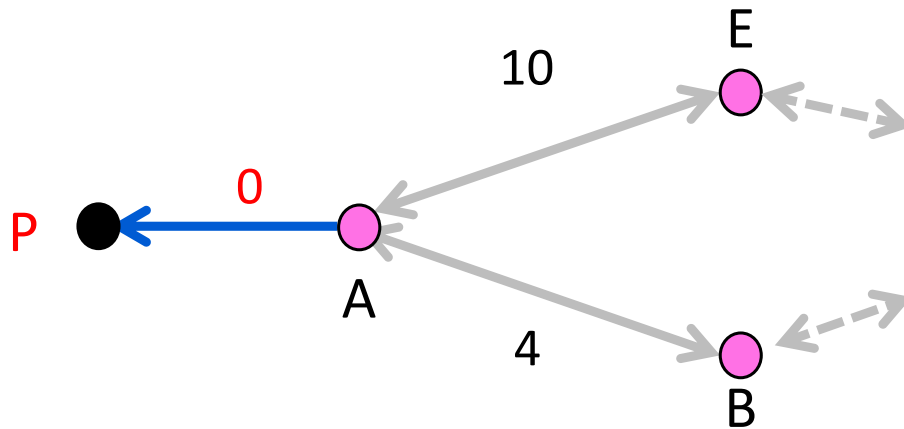 - Routers use <u>longest prefix matching</u> to send packets to the right next hop

Host/Router Combination

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

Single network
(One IP prefix "P")
IP router "A"
Rest of network

LAN switch
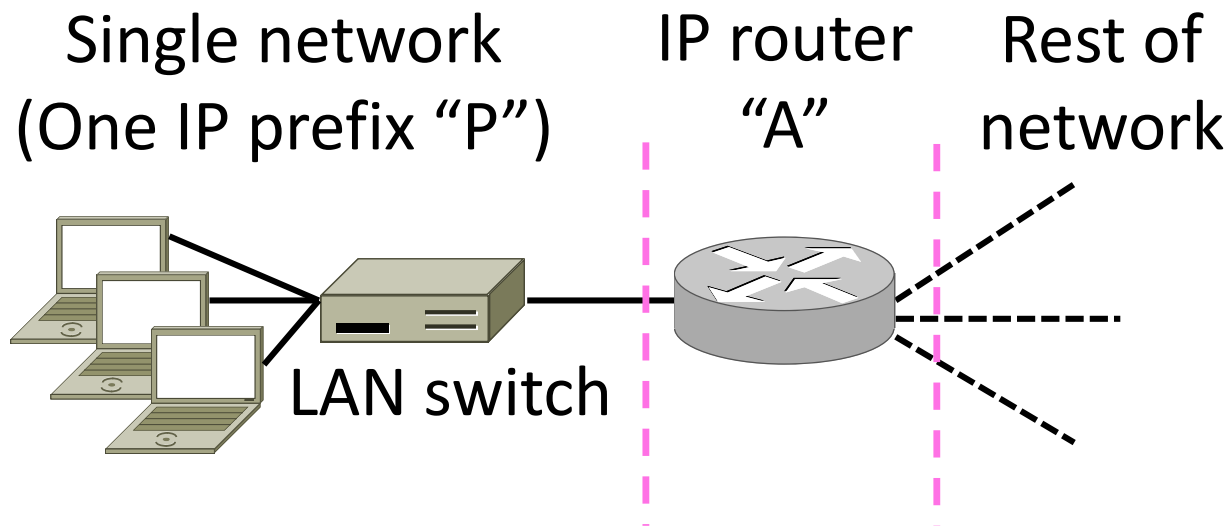
Network Topology for Routing

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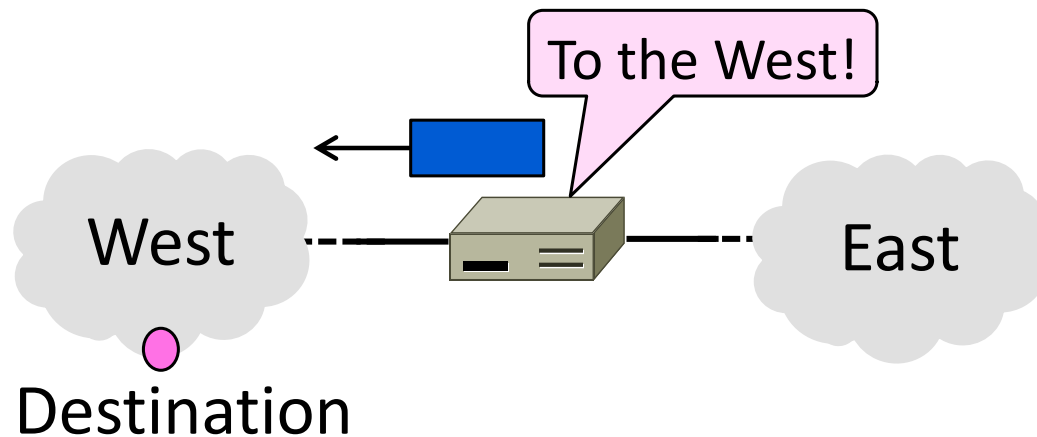 - One entry for all hosts

Network Topology for Routing (2)

- Routing now works as before!
  - 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

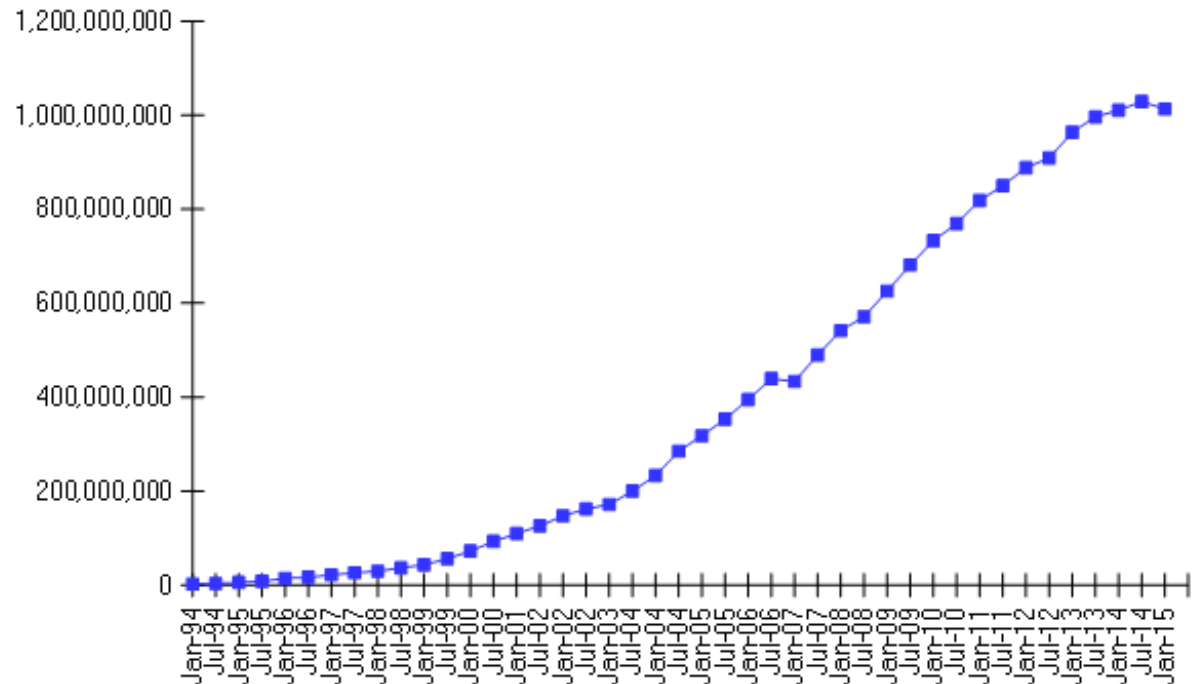# Hierarchical Routing (§5.2.6)

- How to scale routing with hierarchy in the form of regions
  - Route to regions, not individual nodes

Internet Growth

- 1.1 billion Internet hosts ...

- Likely to continue growth with mobile and IoT devices

Internet Domain Survey Host Count

Source: Internet Systems Consortium (www.isc.org)

Internet Routing Growth

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

Ouch!
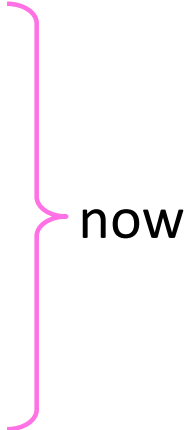
Number of IP Prefixes — Year

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 size of the network
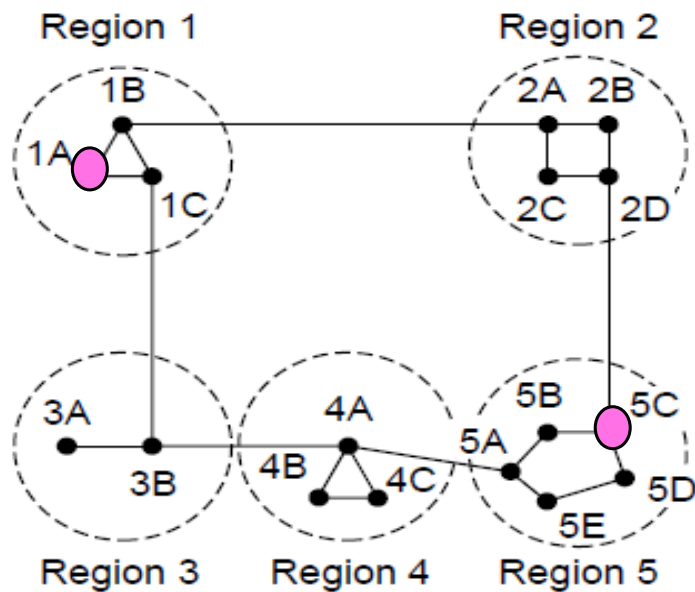
Techniques to Scale Routing

1. IP prefixes
   - Route to blocks of hosts

2. Network hierarchy
   - Route to network regions

3. IP prefix aggregation
   - Combine, and split, prefixes

now

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
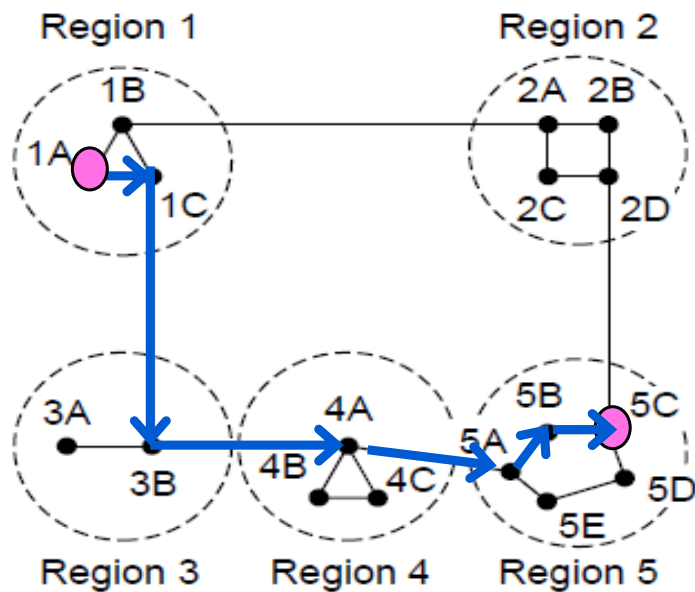
Hierarchical Routing (2)

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

84

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

86

Observations

- Outside a region, nodes have <u>one route</u> to all hosts within the region
  - This gives savings in table size, messages and computation

- However, each node may have a <u>different route</u> 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 (§5.6.2)

- How to help scale routing by adjusting the size of IP prefixes
  - Split (subnets) and join (aggregation)
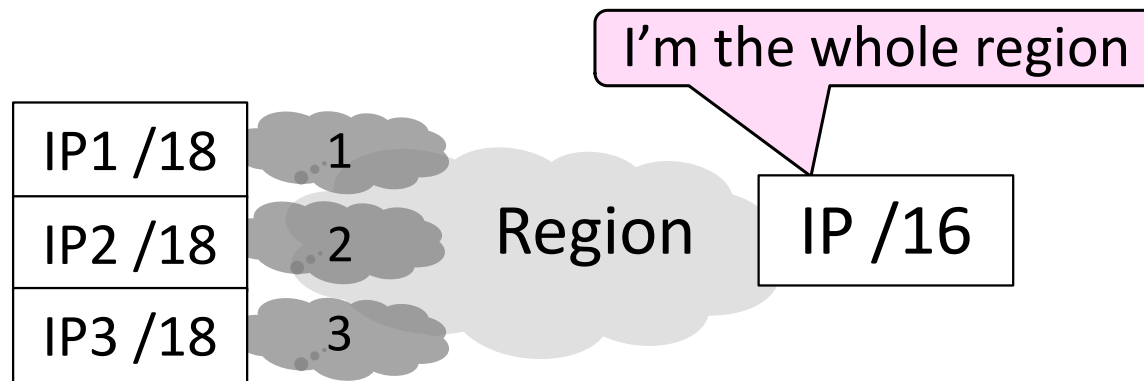
Recall

- IP addresses are allocated in blocks called <u>IP prefixes</u>, e.g., 18.31.0.0/16
  - Hosts on one network in same prefix

- A "/N" prefix has the first N bits fixed and contains 2^{32-N} addresses
  - E.g., "/24"
 - E.g., "/16"

Key Flexibility

- Routers keep track of prefix lengths
  - Use it for longest prefix matching

  Routers can change prefix
  lengths without affecting hosts

- <u>More specific</u> IP prefix
  - Longer prefix, fewer IP addresses

- <u>Less specific</u> IP prefix
  - Shorter prefix, more IP addresses

Prefixes and Hierarchy

- IP prefixes already help to scale routing, but we can go further
  - Can use a less specific prefix to name a region made up of several prefixes

I'm the whole region

IP1 /18	1
IP2 /18	2
IP3 /18	3

Region   IP /16
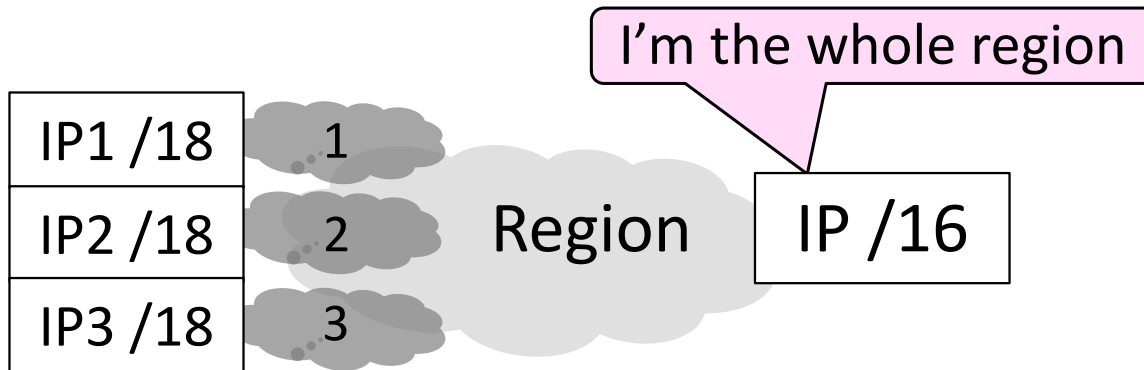
Subnets and Aggregation

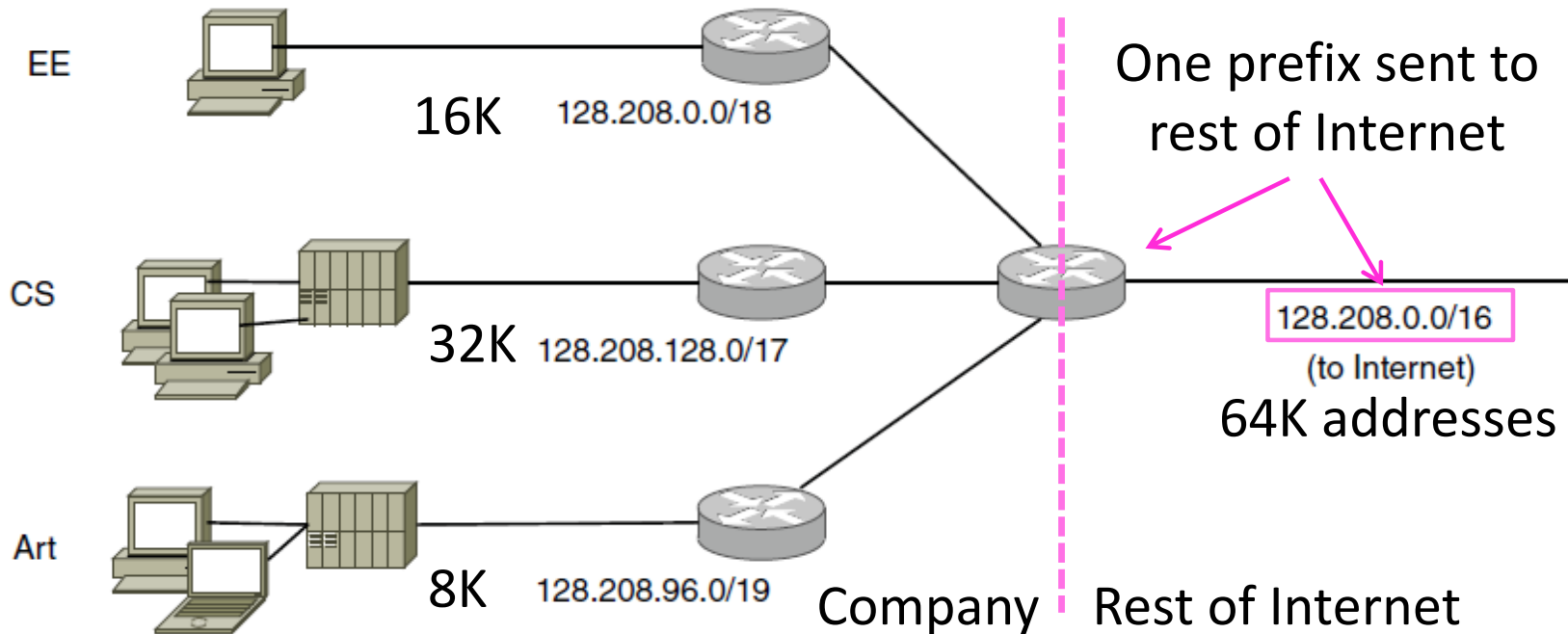Two use cases for adjusting the size of IP prefixes; both reduce routing table size

1. Subnets
   – Internally split one less specific prefix into multiple more specific prefixes
2. Aggregation
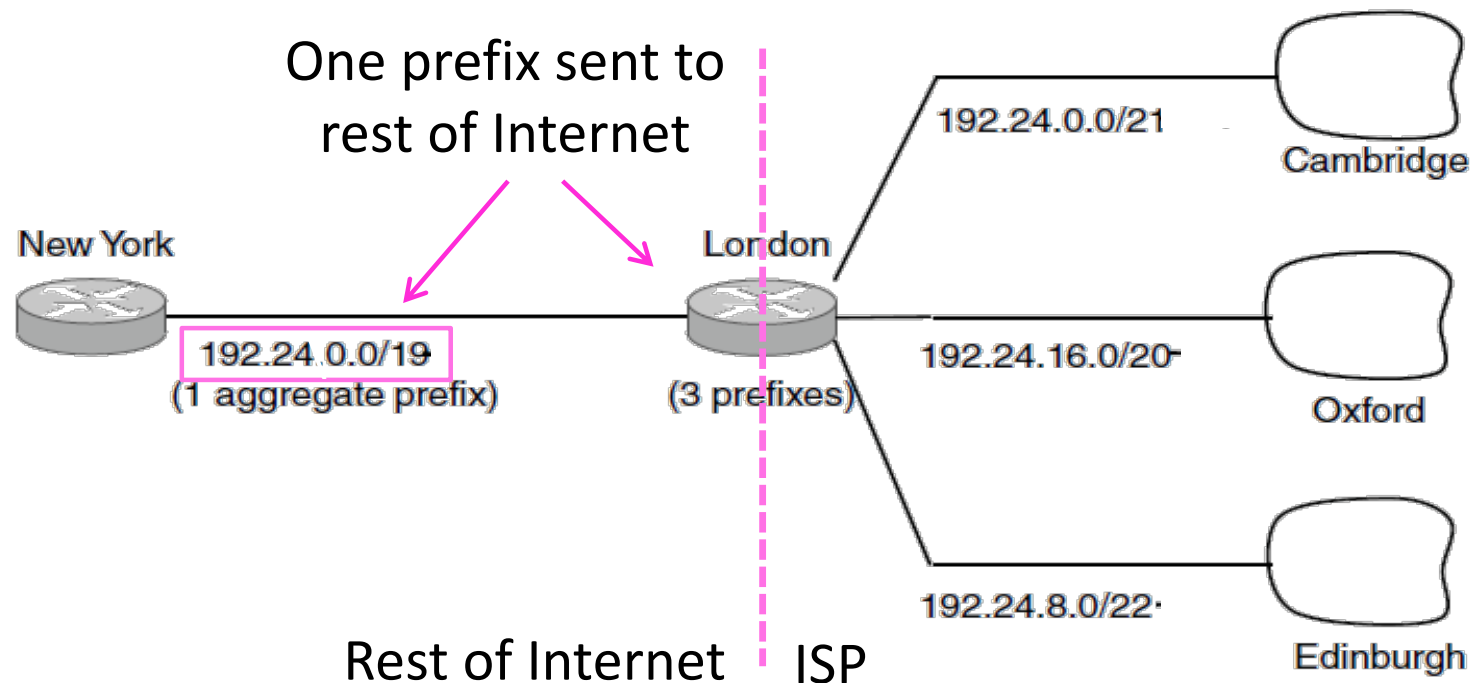   – Externally join multiple more specific prefixes into one large prefix

Subnets

- Internally split up one IP prefix

EE — 16K — 128.208.0.0/18

CS — 32K — 128.208.128.0/17

Art — 8K — 128.208.96.0/19

One prefix sent to rest of Internet

128.208.0.0/16
(to Internet)

64K addresses

Company | Rest of Internet

Aggregation

- Externally join multiple separate IP prefixes

One prefix sent to
rest of Internet

New York

192.24.0.0/19
(1 aggregate prefix)

London

(3 prefixes)

192.24.0.0/21
Cambridge

192.24.16.0/20
Oxford

192.24.8.0/22
Edinburgh

Rest of Internet | ISP

94

Routing with Multiple Parties (§5.6.7)

- Routing when there are multiple parties, each with their own goals
  - Like Internet routing across ISPs …
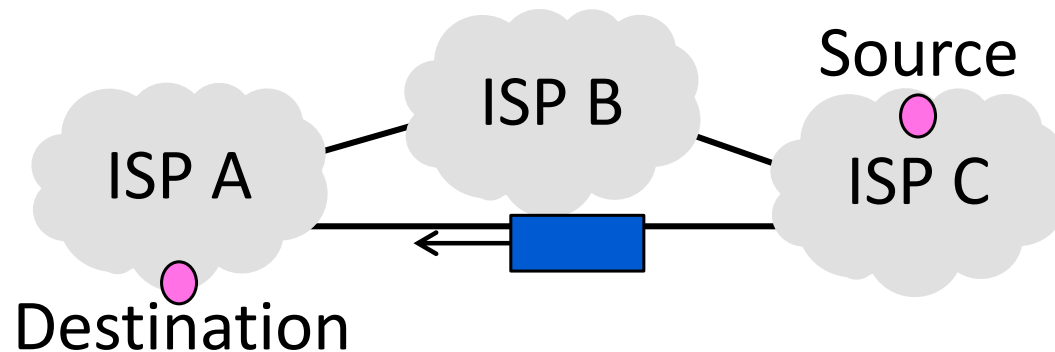
Structure of the Internet

- Networks (ISPs, CDNs, etc.) group hosts as IP prefixes
- Networks are richly interconnected, often using IXPs (Internet eXchange Points)
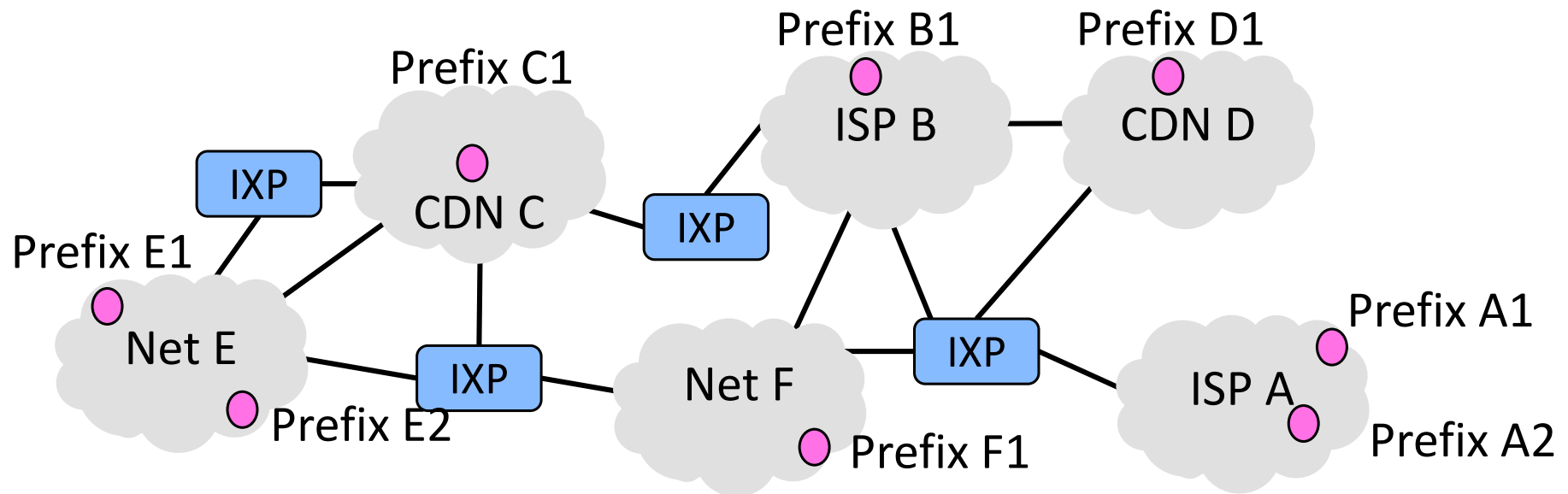
Internet-wide Routing Issues

- Two problems beyond routing within an individual 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

Yikes!

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?

ISP A

ISP B

Prefix A1

Prefix B1

Prefix A2

Prefix B2
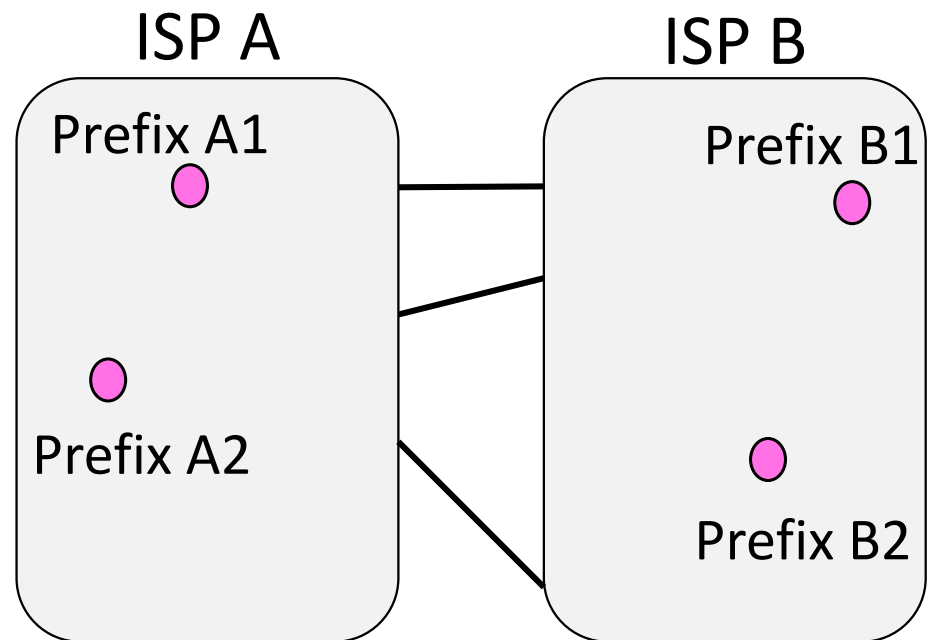
Effects of Independent Parties (2)

- Selected paths are longer than overall shortest path
  - And asymmetric too!
- This is a consequence of independent goals and decisions, not hierarchy
- Called "hot-potato" routing

ISP A　　　　　　　　　ISP B

Prefix A1

Prefix B1

Prefix A2

Prefix B2
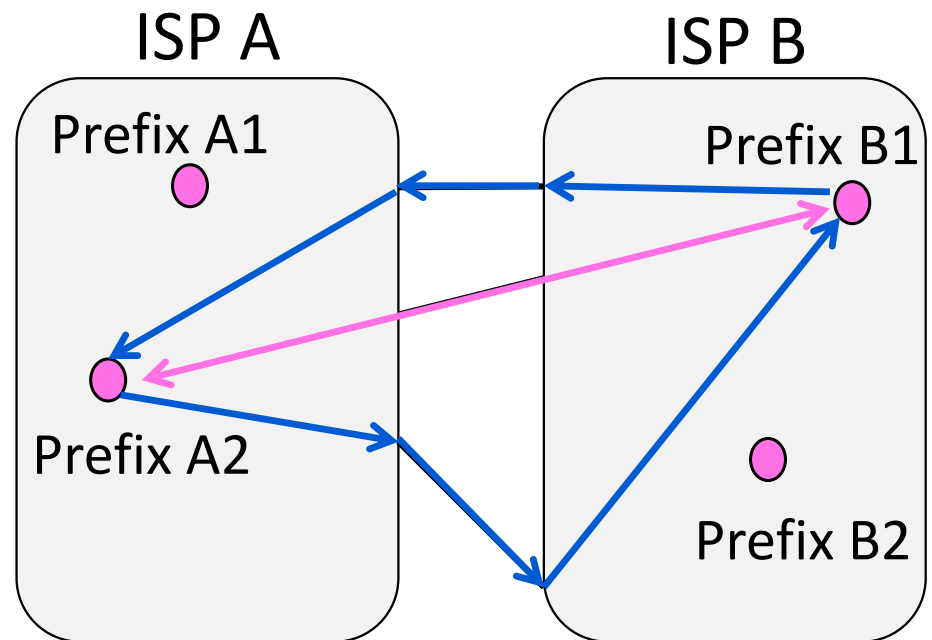
Routing Policies

- 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 from customer to deliver to the rest of Internet

  - ISP accepts traffic from the rest of the Internet to deliver to customer

  - Customer pays ISP for the privilege

ISP

Customer 1

Customer 2

Rest of Internet
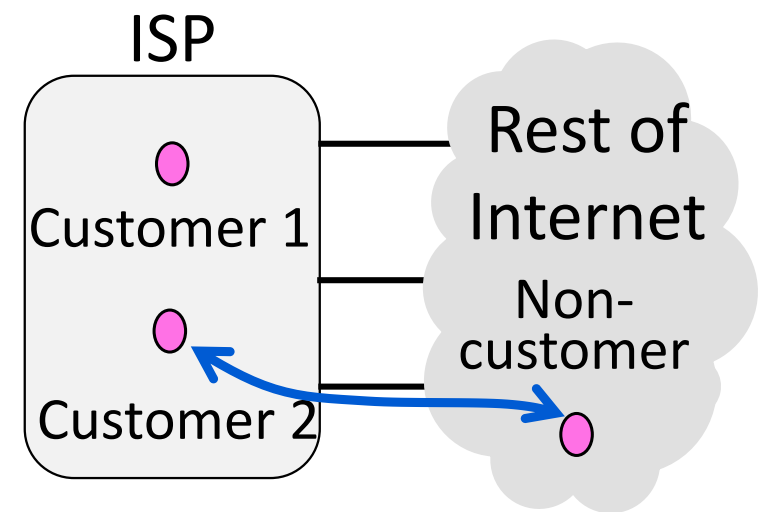
Non-customer

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
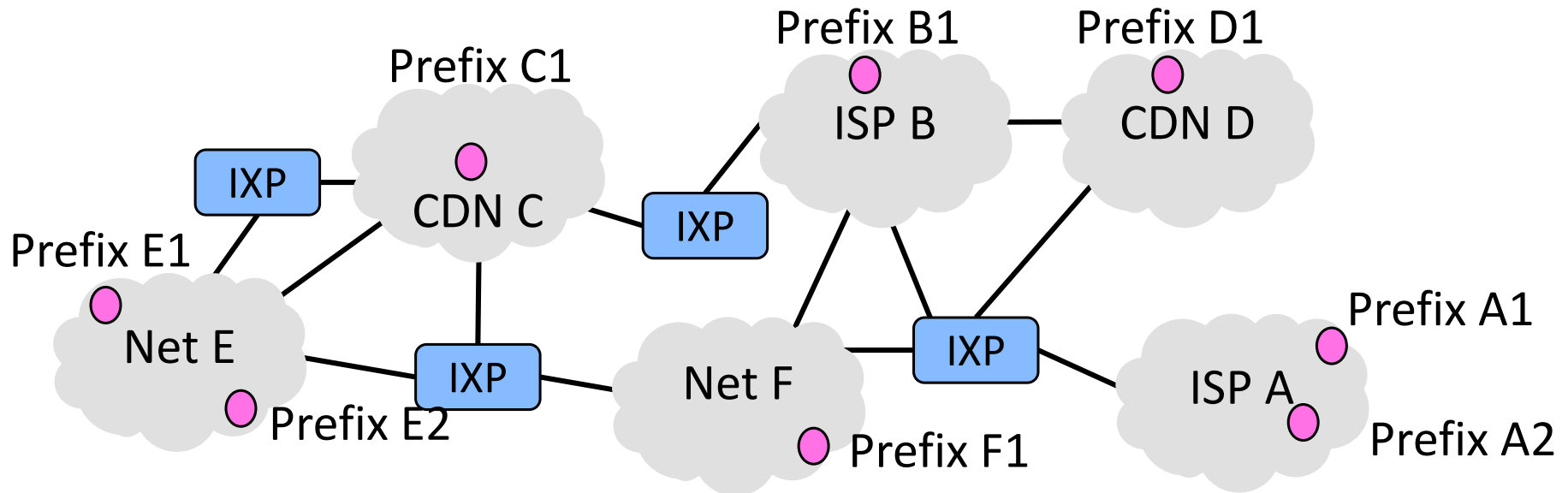 - ISPs don't pay each other
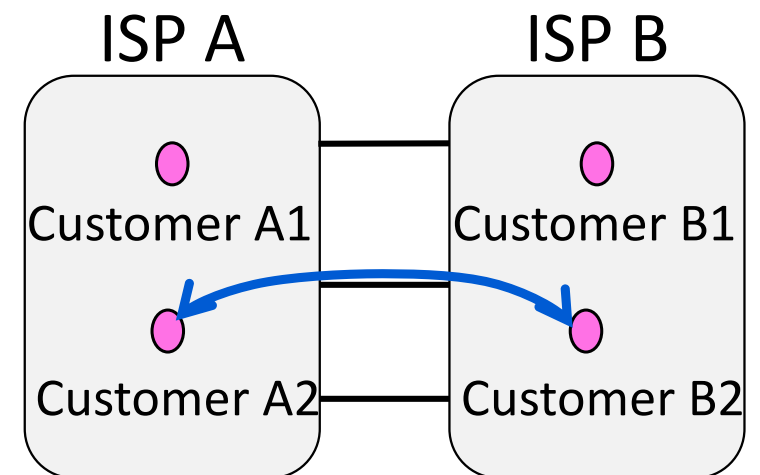
Border Gateway Protocol (§5.6.7)

- How to route with multiple parties, each with their own routing policies
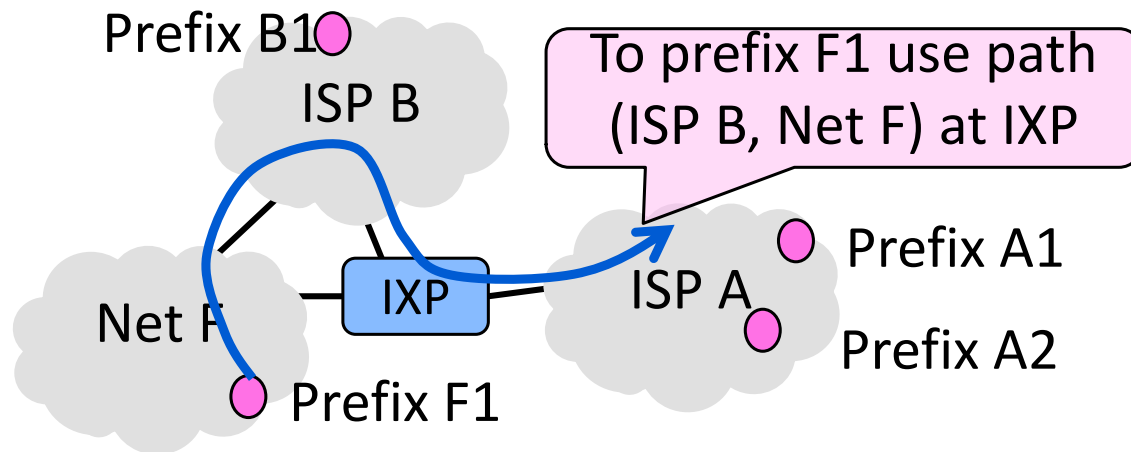 - BGP computes Internet-wide routes
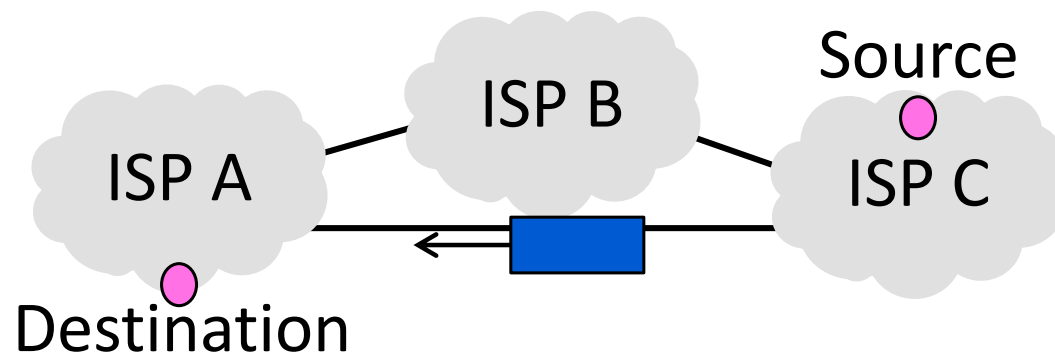
Recall

- Internet is made up of independently run networks
- Each network has its own route preferences (policies)

BGP (Border Gateway Protocol)

- BGP is the protocol that computes interdomain routes in the Internet
  - Path vector, a kind of distance vector

BGP (2)

- Different parties like ISPs are called AS (Autonomous Systems)
- Border routers of ASes announce BGP routes to each other

- Route announcements contain an 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

BGP (3)
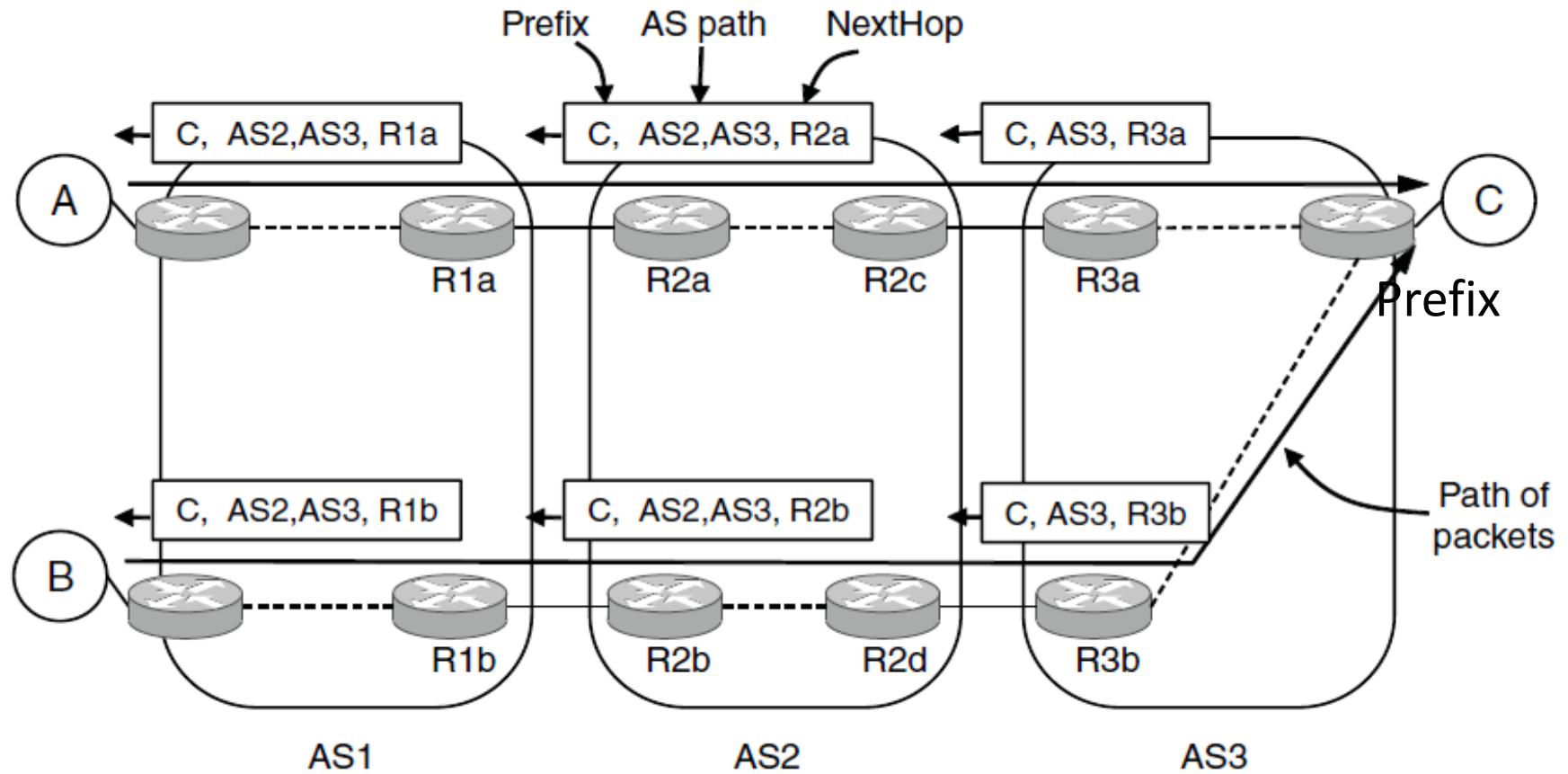
BGP (4)

Routing 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

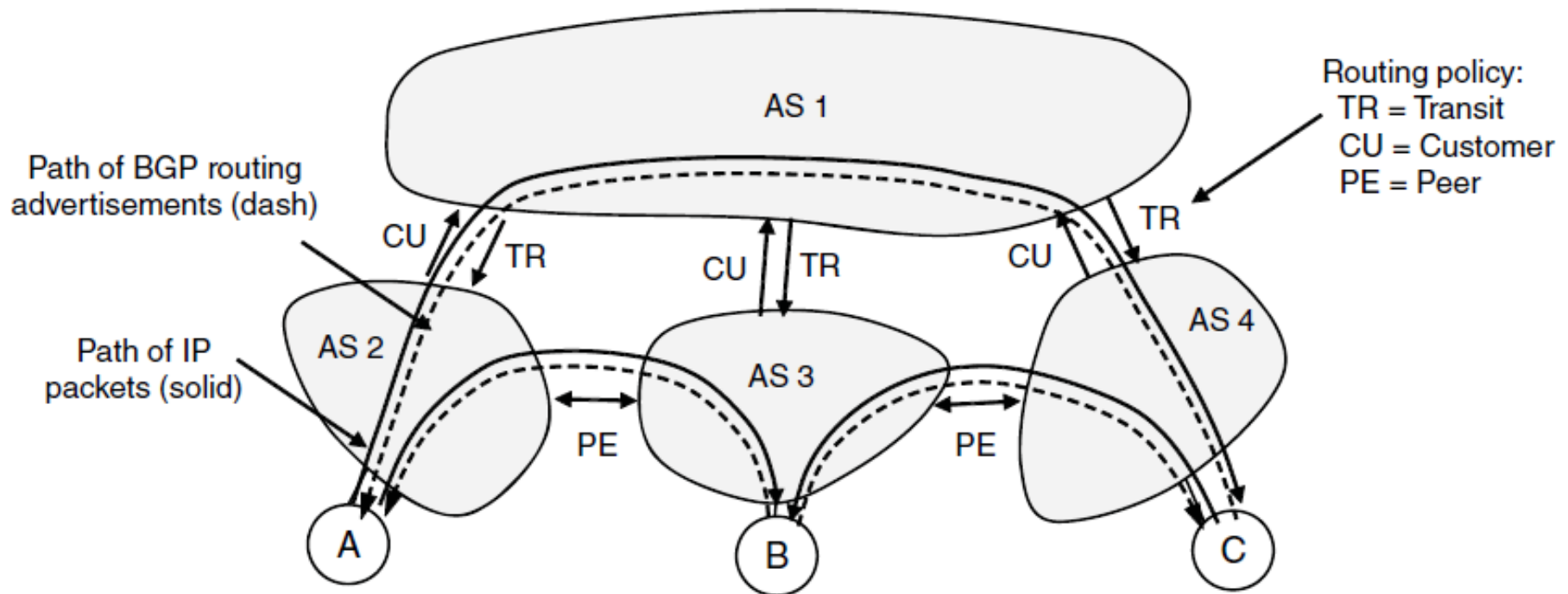2. Border routers of ISP select the best path amongst the paths they hear

BGP Example

- AS2 buys TRANSIT service from AS1 and has PEER service with AS3

BGP Example (2)

- CUSTOMER (other side of TRANSIT): AS2 says [A, (AS2)] to AS1
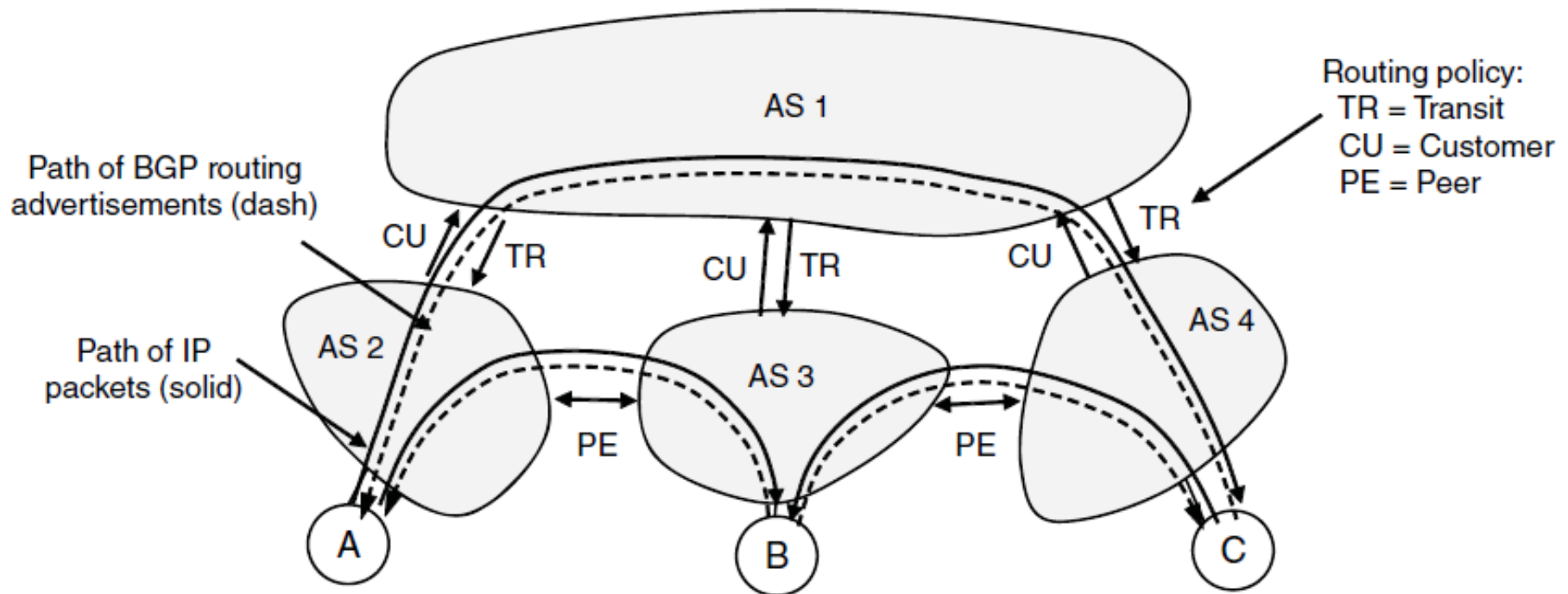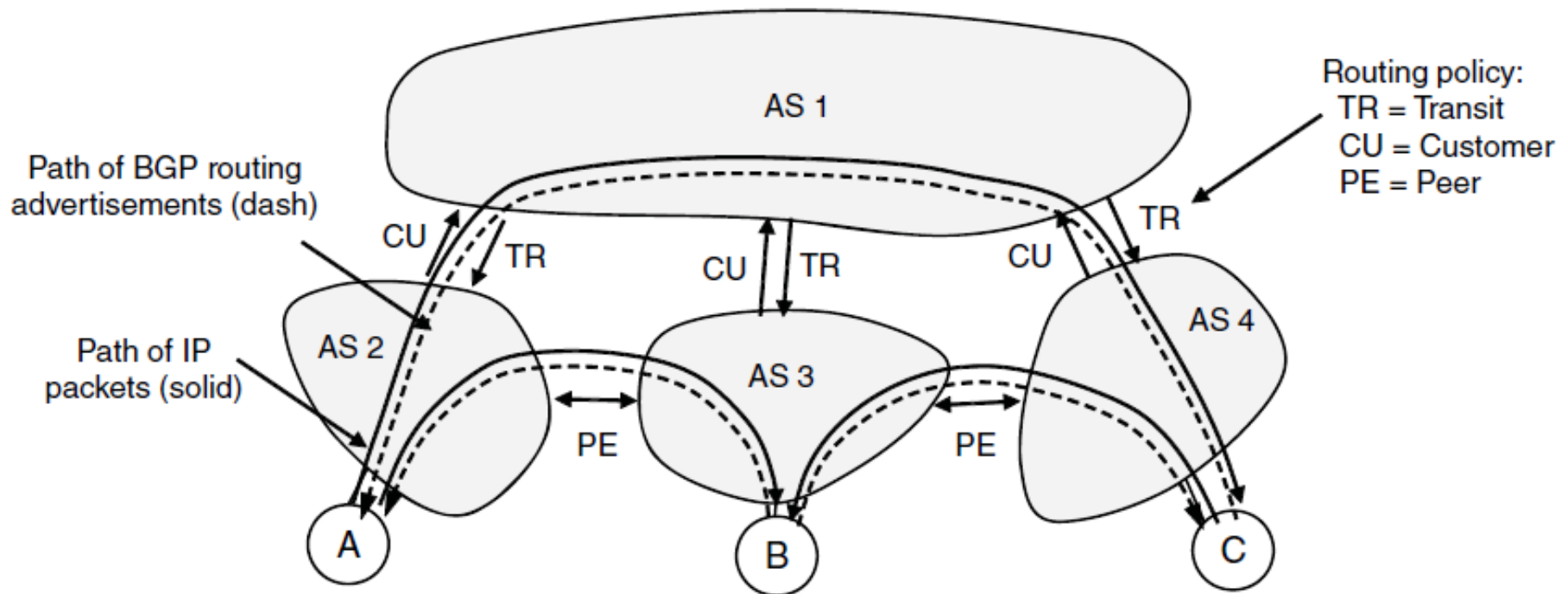
BGP Example (3)

- TRANSIT: AS1 says [B, (AS1, AS3)], [C, (AS1, AS4)] to AS2
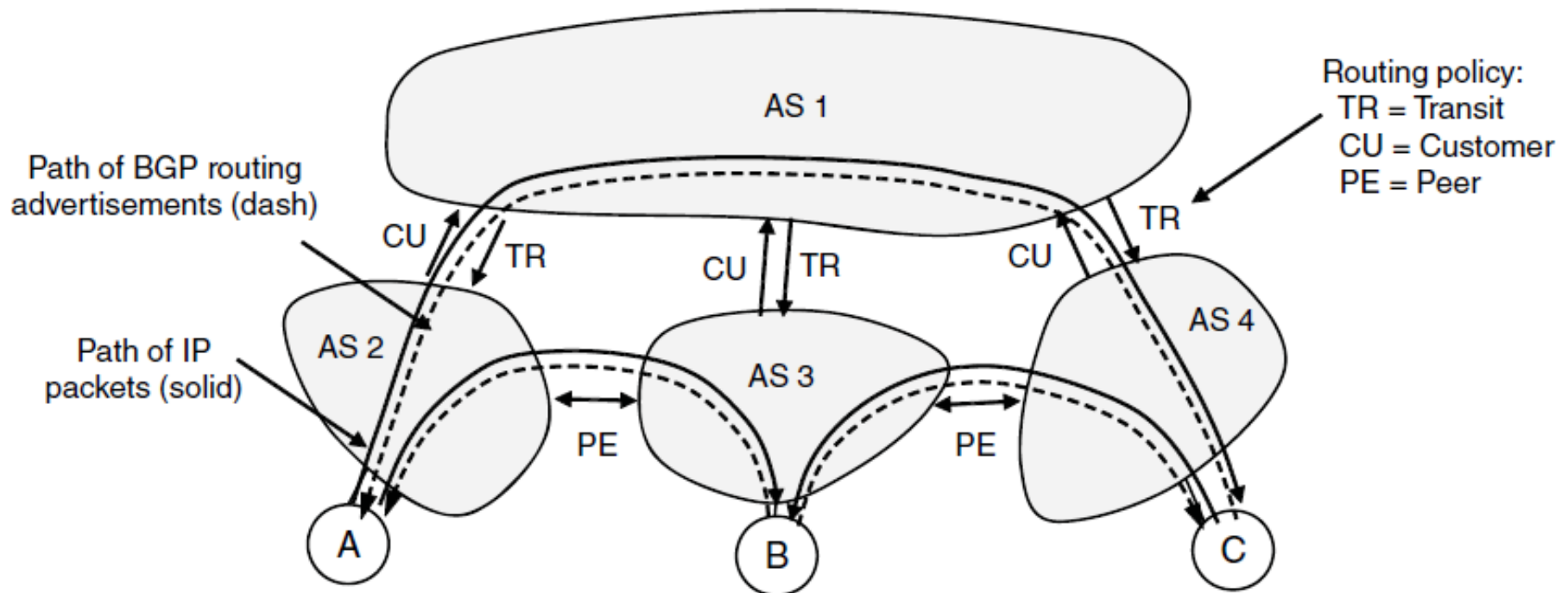
BGP Example (4)

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

BGP Example (5)

- AS2 hears one route to C, and two routes to B (chooses AS3!)

Closing Thoughts

- Much more beyond basics to explore!

- Routing policies of ISPs are a tricky issue …
  - Can we be sure independent decisions will yield sensible overall routes?
- Other important factors:
  - Convergence effects
  - How well it scales
  - Integration with routing within ISPs
  - And more …