CS 138: Communication I

Topics

- Network Metrics
- Layering
- Reliability
- Congestion Control
- Routing

The Fallacies of Distributed Computing

1. *The network is reliable*
2. *Latency is zero*
3. *Bandwidth is infinite*
4. The network is secure
5. *Topology doesn't change*
6. There is one administrator
7. Transport cost is zero
8. The network is homogeneous

Performance Metrics

- **Bandwidth – Number of bits/unit of time the medium can transmit**
- **Latency – How long for message to cross network**
  - **Process + Queue + Transmit + Propagation**
- **Throughput – Effective number of bits received/unit of time**
  - ***e.g.* 10Mbps**
- **Goodput - *Useful* bits received per unit of time**
  - **Discounts protocol overhead**
- **Jitter – Variation in latency**

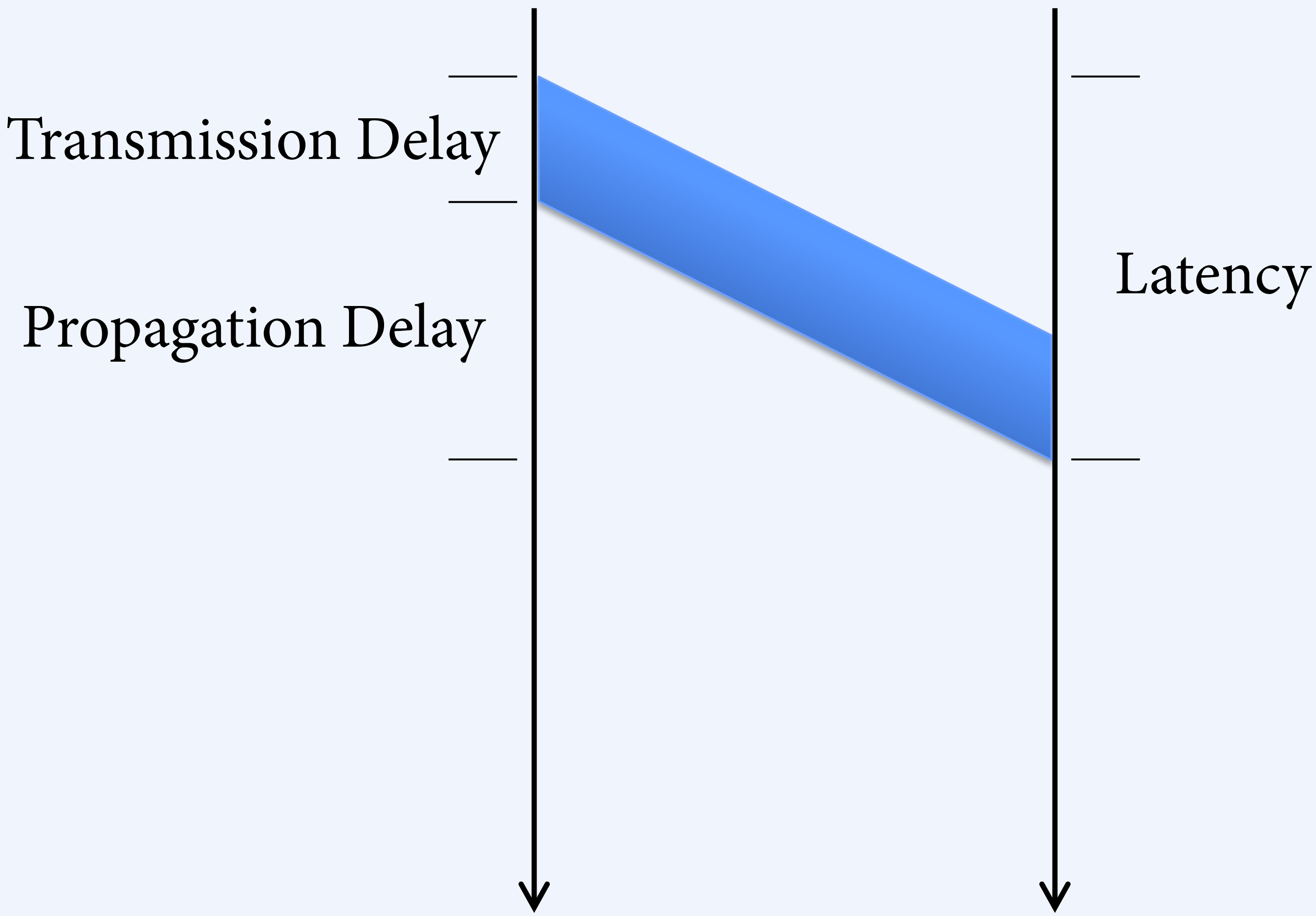
Latency

- **Processing Delay**
  - **Per message, small, limits throughput**
  - ***e.g. to achieve full rate at a 100Mbps link, you have a budget of* 120μs/pkt:**

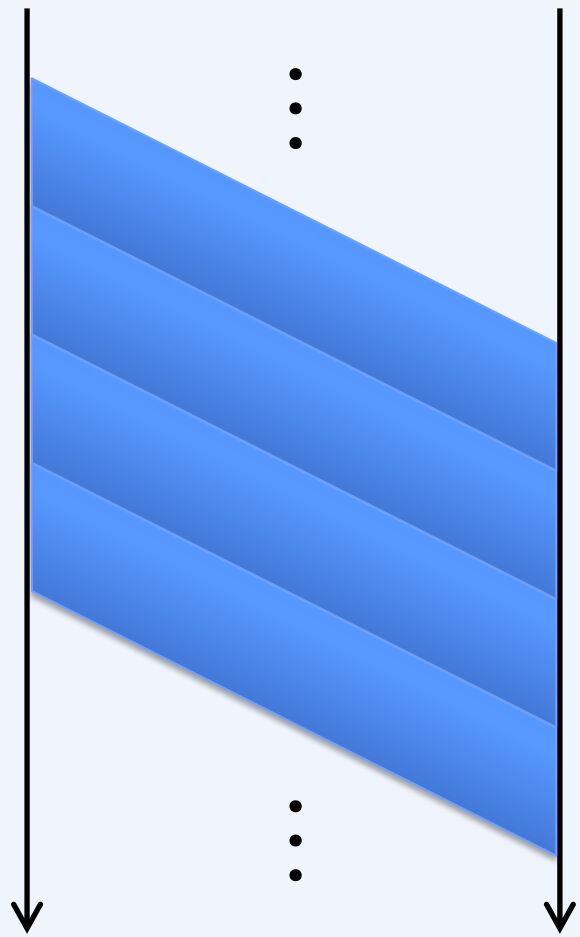
$$\frac{100Mb}{s} \times \frac{pkt}{1500B} \times \frac{B}{8b} \approx 8{,}333\,pkt/s$$

- **Queueing Delay**
  - **Highly variable, offered load vs outgoing b/w**
- **Transmission Delay – depends on medium**
  - **Size/Bandwidth**
- **Propagation Delay – depends on medium**
  - **Distance/Speed of Light**

Sending Packets Across



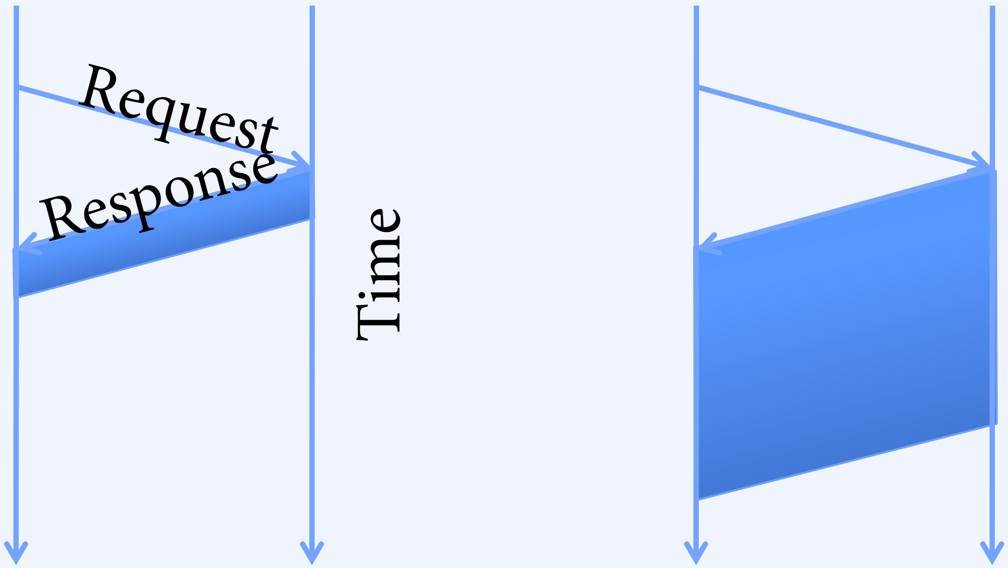
Sending Packets Across



Throughput: bits / s

Which matters most, bandwidth or delay?

- How much data can we send during one RTT?
- *E.g.*, send request, receive file



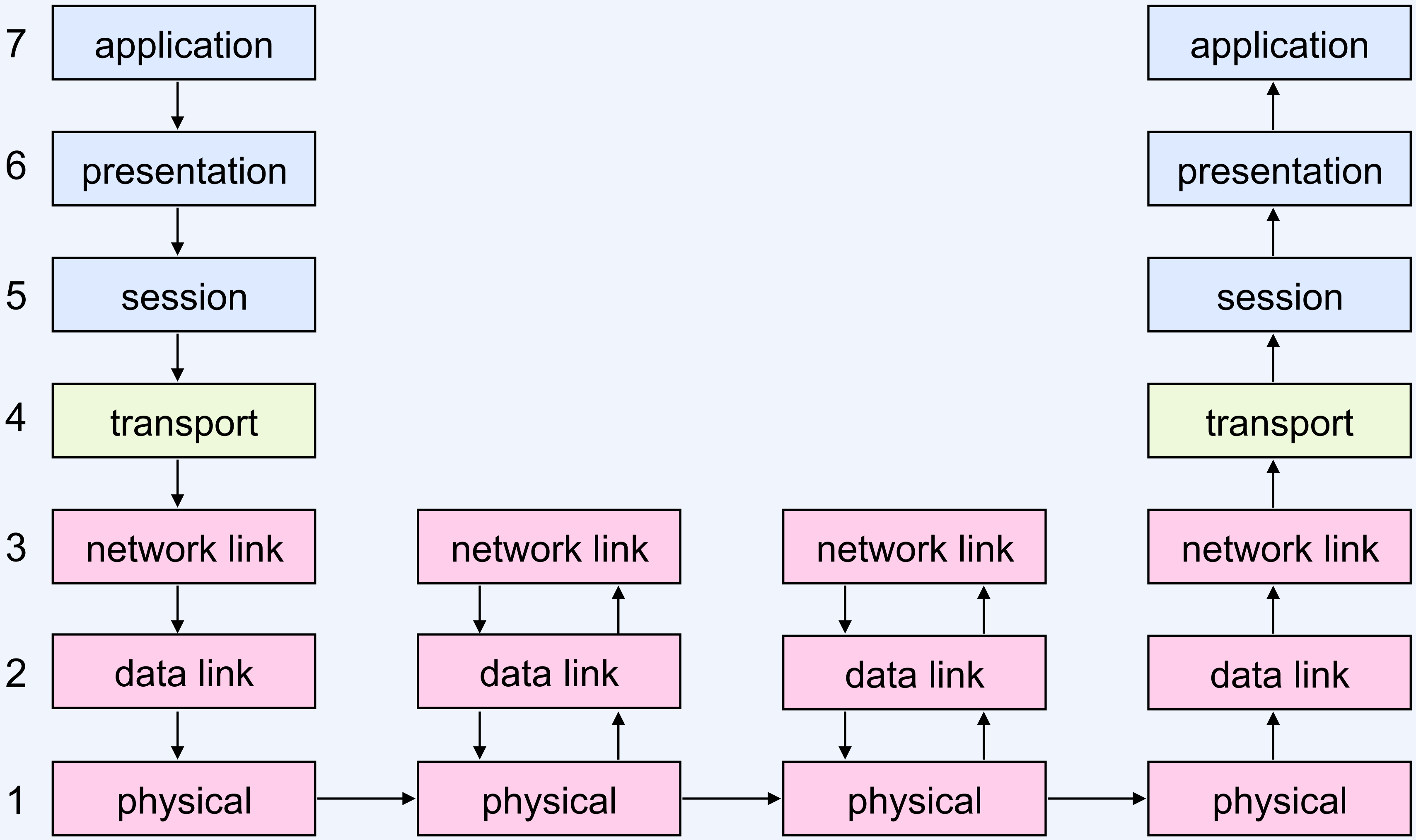
- For small transfers, latency more important, for bulk, throughput more important

Many Requirements

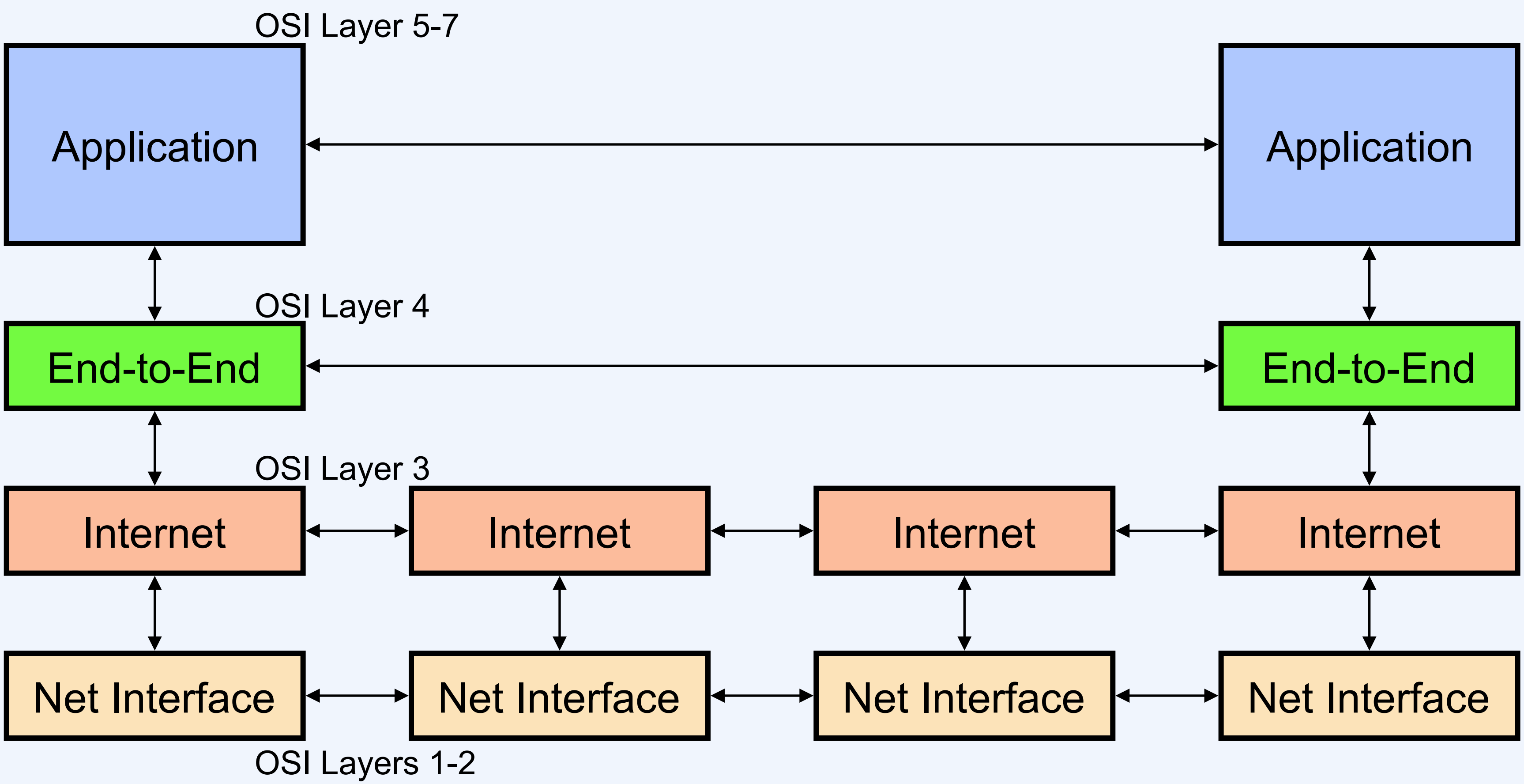
- Modulation, encoding, framing
- Routing
- Reliability
- Flow control
- Congestion control
- Security
- …

- How to organize all of these?

ISO OSI Reference Model



Internet Architecture



Layers

- Application – what the users sees, *e.g.*, HTTP
- Presentation – crypto, conversion between representations
- Session – can tie together multiple streams (*e.g.*, audio & video)
- Transport – demultiplexes, provides reliability, flow and congestion control
- Network – sends *packets*, using *routing*
- Data Link – sends *frames*, handles media access
- Physical – sends individual bits

How to Place Functionality

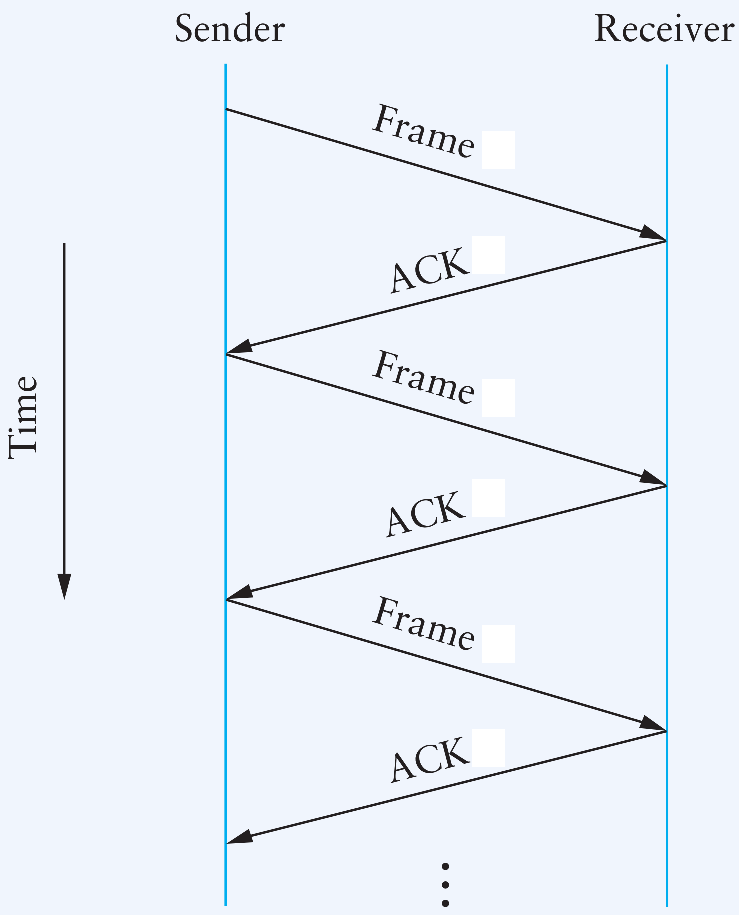
- Don't provide functionality on a layer that some of the users won't need
  - E.g. security, in-order-delivery conflicts with timeliness
- Don't provide functionality on a layer when it is insufficient
  - E.g. error correction, reliability
  - This is the "End-to-end" Principle
  - Can violate when there is a performance gain
- Do provide a functionality that can be reused
  - E.g., IP routing and forwarding is useful to many layers above

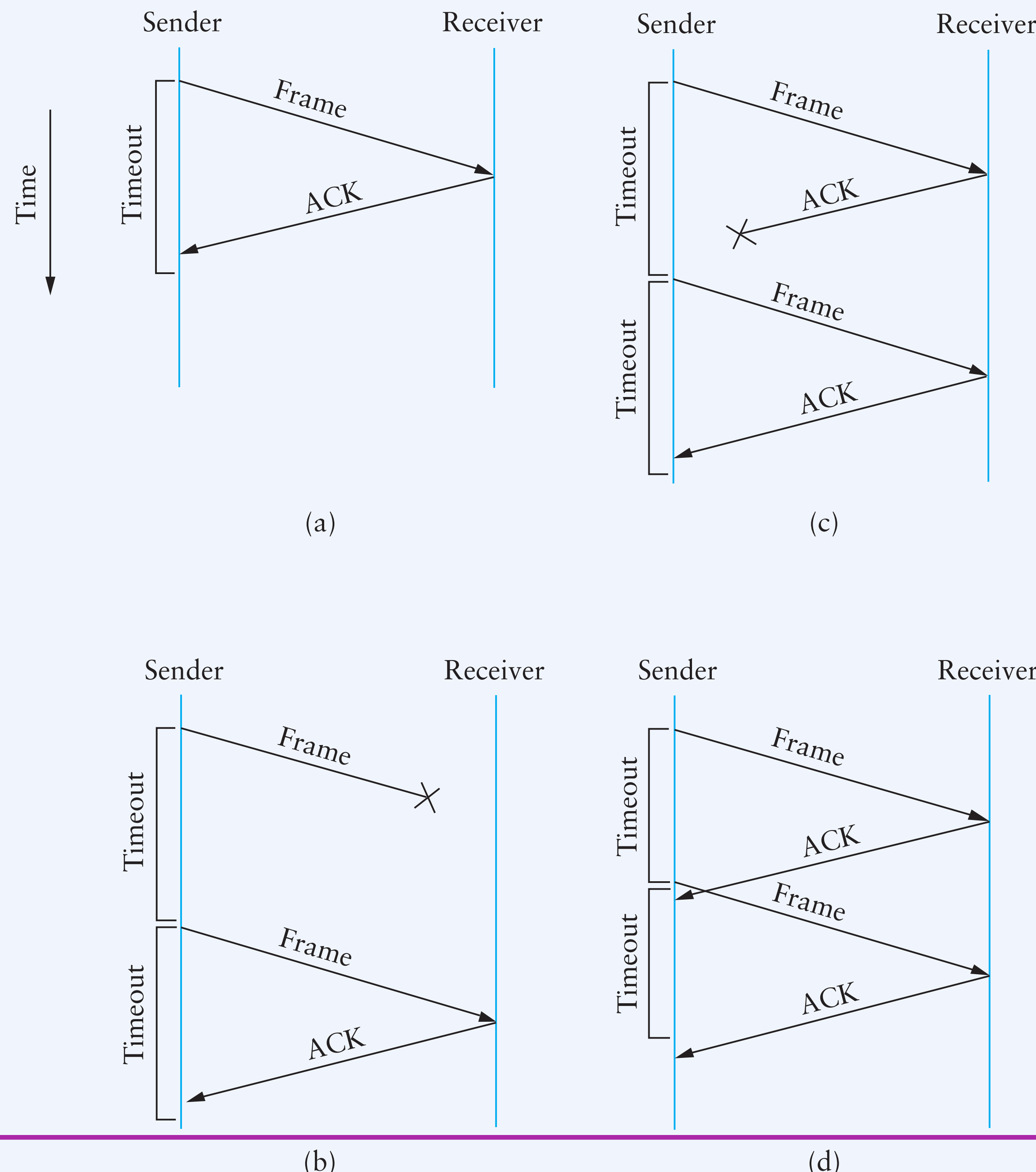
Reliable Delivery

- Several sources of errors in transmission
- Error detection can discard bad frames
- Problem: if bad packets are lost, how can we ensure reliable delivery?
  - Exactly-once semantics = at least once + at most once

At Least Once Semantics

- **How can the sender know packet arrived *at least once*?**
  - **Acknowledgments + Timeout**
- **Stop and Wait Protocol**
  - **S: Send packet, wait**
  - **R: Receive packet, send ACK**
  - **S: Receive ACK, send next packet**
  - **S: No ACK, timeout and retransmit**

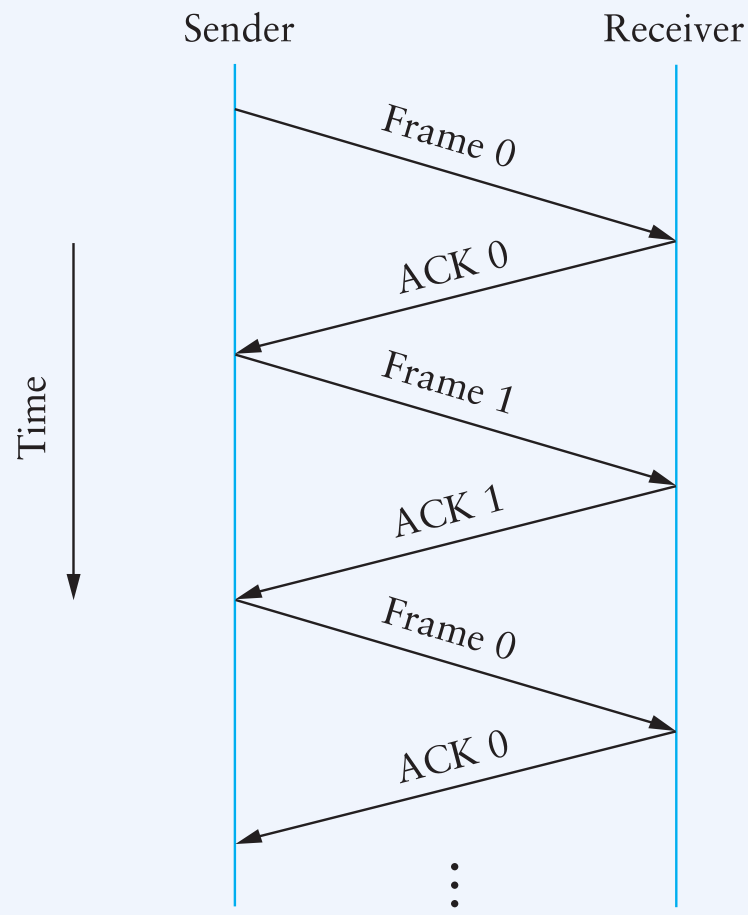

Sender
Receiver
Time
Frame
ACK
Frame
ACK
Frame
ACK


Sender
Receiver
Time
Timeout
Frame
ACK
(a)
(b)
(c)
(d)

Stop and Wait Problems

- Duplicate data
- Duplicate acks
- Slow (channel idle most of the time!)
- May be difficult to set the timeout value

Duplicate data: adding sequence numbers

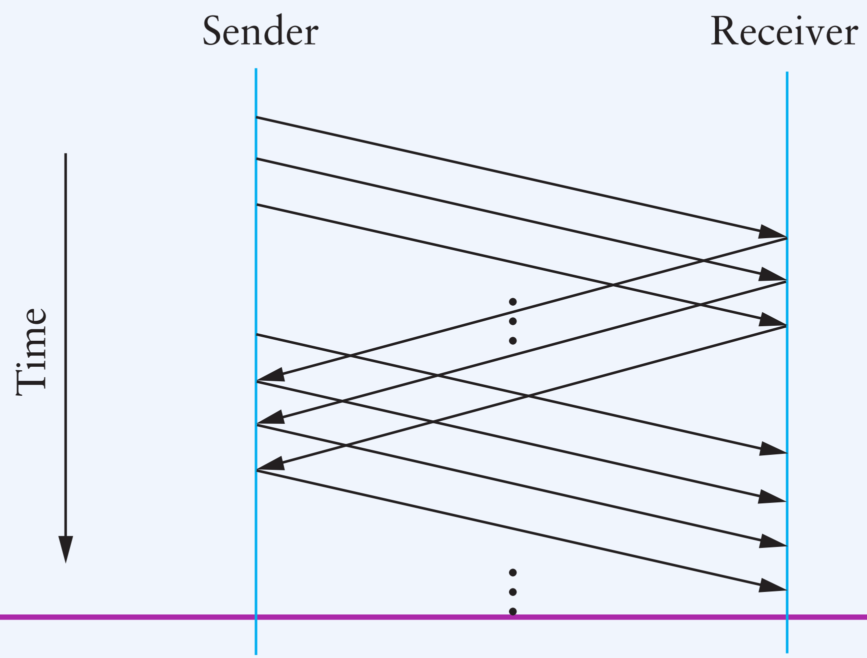


At Most Once Semantics

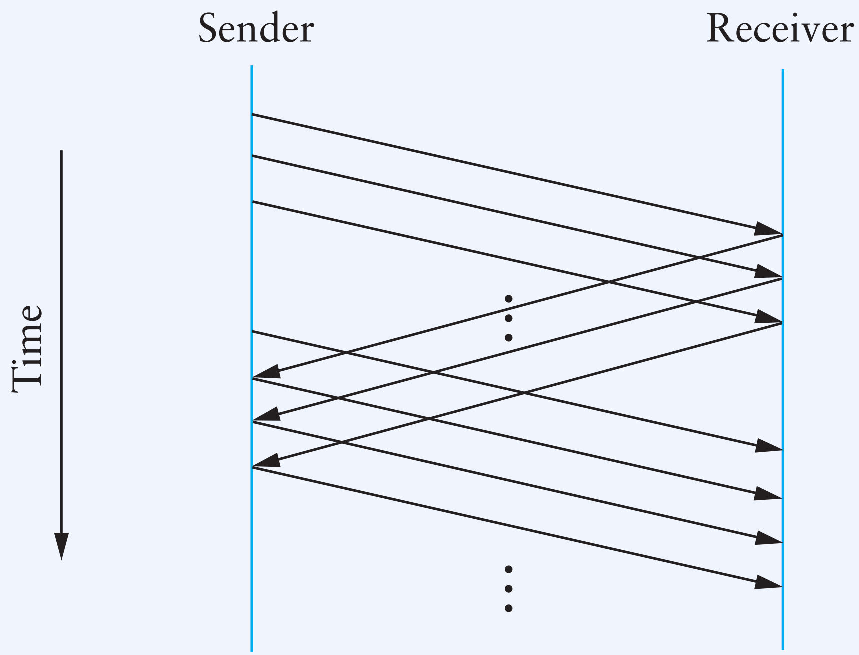
- How to avoid duplicates?
  - Uniquely identify each packet
  - Have receiver and sender remember
- Stop and Wait: add 1 bit to the header
  - Why is it enough?

Going faster: sliding window protocol

- Still have the problem of keeping pipe full
  - Generalize approach with > 1-bit counter
  - Allow multiple outstanding (unACKed) frames
  - Upper bound on unACKed frames, called *window*

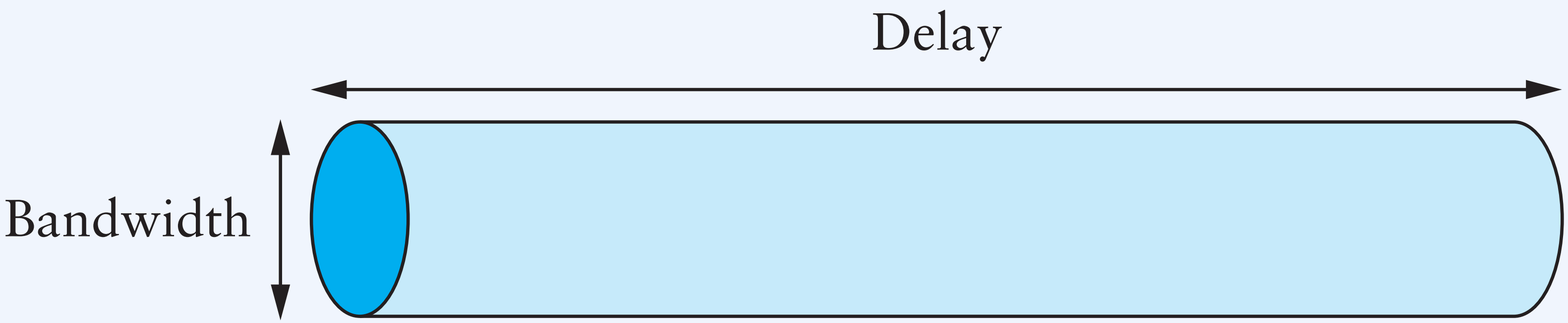


How big should the window be?



- How many bytes can we transmit in one RTT?
  - BW B/s x RTT s => “Bandwidth-Delay Product”

Maximizing Throughput



- Can view network as a pipe
  - For full utilization want bytes in flight ≥ bandwidth × delay
  - But don't want to overload the network
- What if protocol doesn't involve bulk transfer?
  - Get throughput through concurrency – service multiple clients simultaneously

Problem: on the Internet, the pipe varies

- Different paths
- Queues along the way
- Other flows

- Q: How to set the window size then?

Efficiency

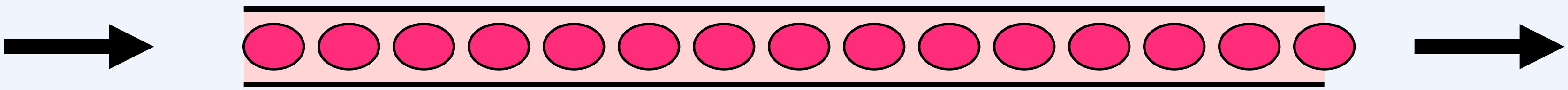
- 3 goals:
  - Utilize the network
  - Don't overwhelm the receiver: flow control
  - Don't overwhelm the network: congestion control

Congestion Control

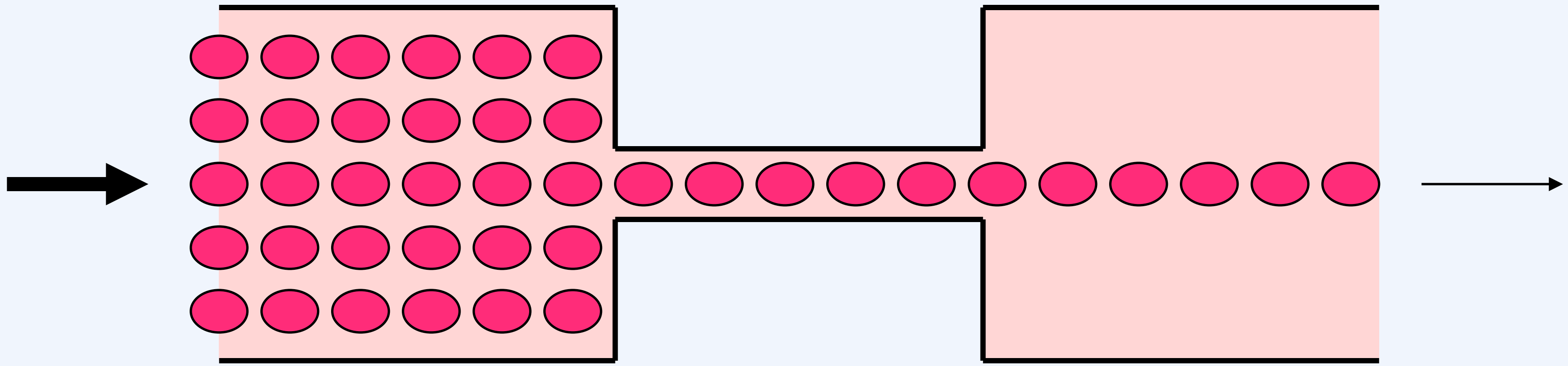
- 3 Key Challenges
  – Determining the available capacity in the first place
  – Adjusting to changes in the available capacity
  – Sharing capacity between flows

- Idea
  – Each source determines network capacity for itself
  – Rate is determined by window size
  – Uses implicit feedback (drops, delay)
  – ACKs pace transmission (self-clocking)

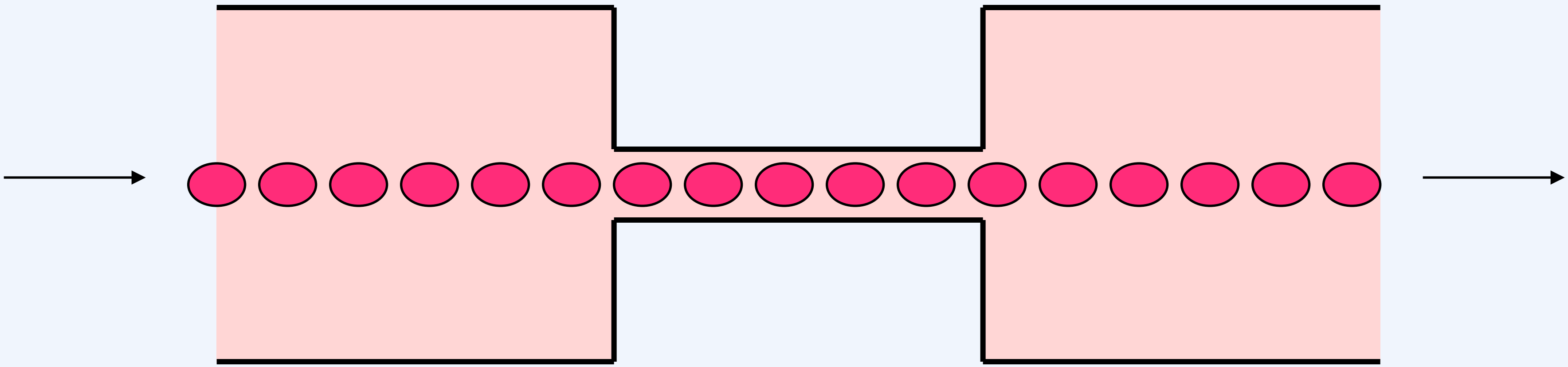
Ack Clocking



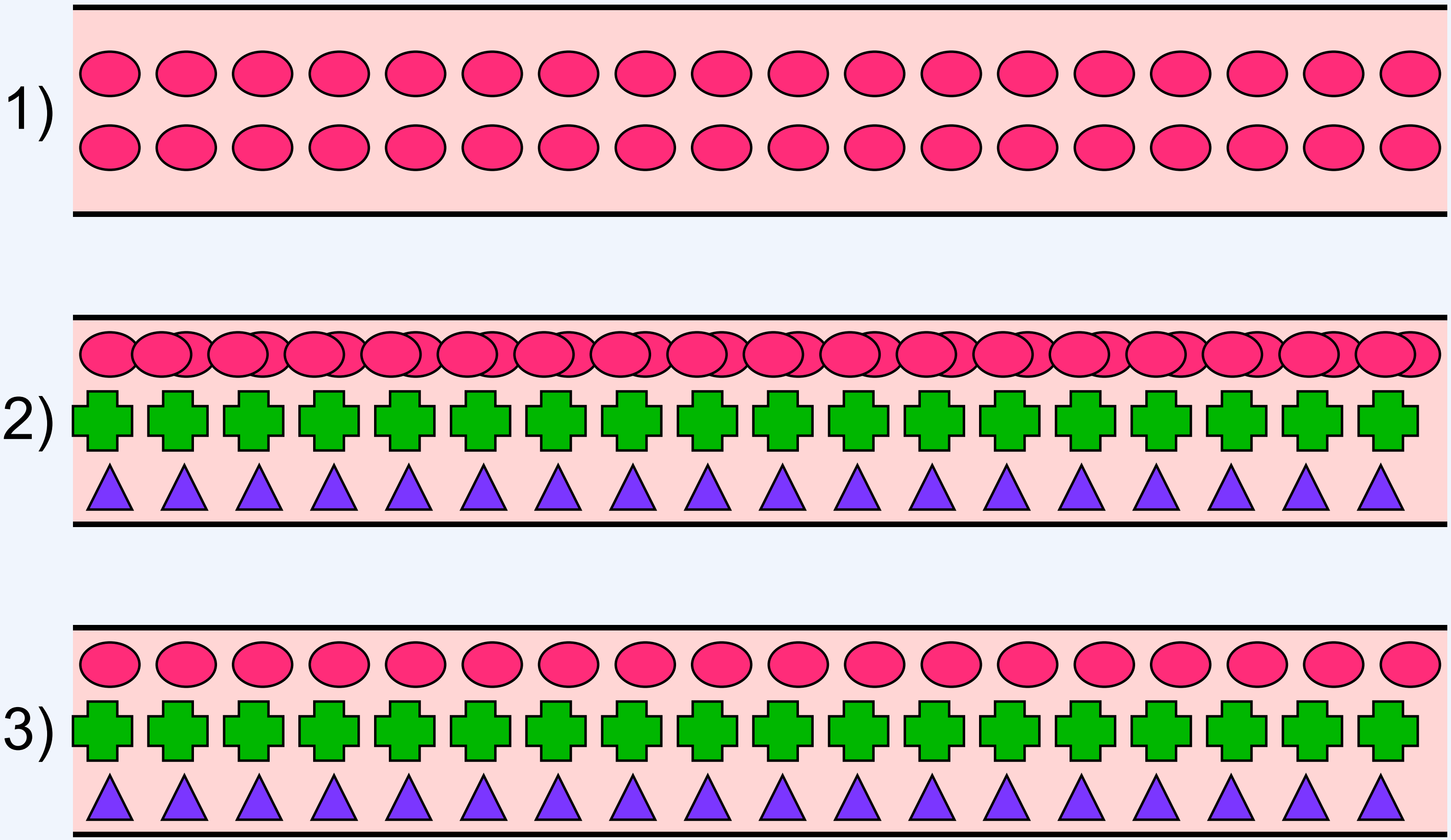
Fast Start



Slow Start



Congestion Control (1)



Dealing with Congestion

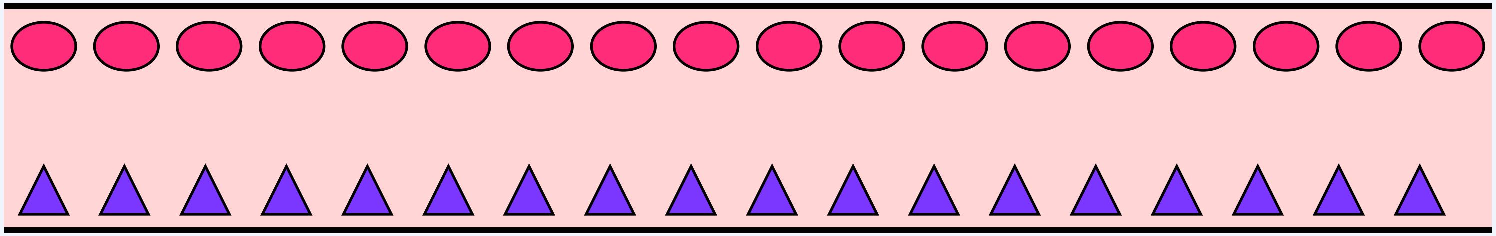
- Assume losses are due to congestion
- After a loss, reduce congestion window
  - How much to reduce?

How much to reduce window?

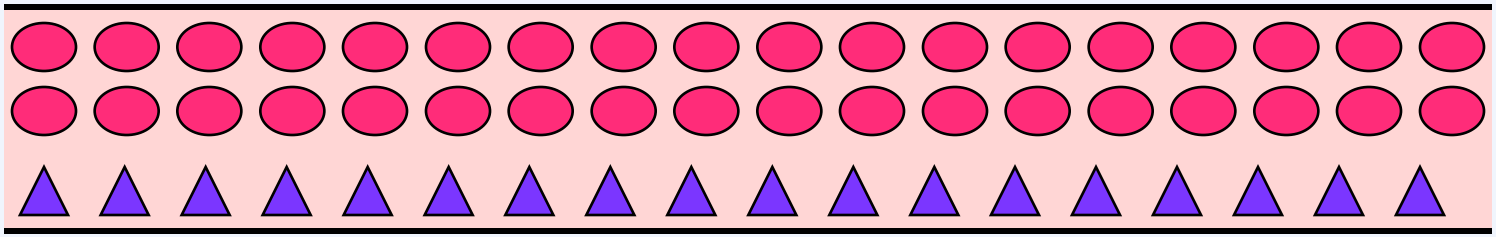
- Crude model of the network
  - Let L_i be the load (# pkts) in the network at time I
  - If network uncongested, roughly constant $L_i = N$
- What happens under congestion?
  - Some fraction γ of packets can't exit the network
  - Now $L_i = N + \gamma L_{i-1}$, or $L_i \approx g^i L_0$
  - Exponential increase in congestion
- Sources must decrease offered rate exponentially
  - i.e, multiplicative decrease in window size
  - TCP chooses to cut window in half

Congestion Control (2)

4)



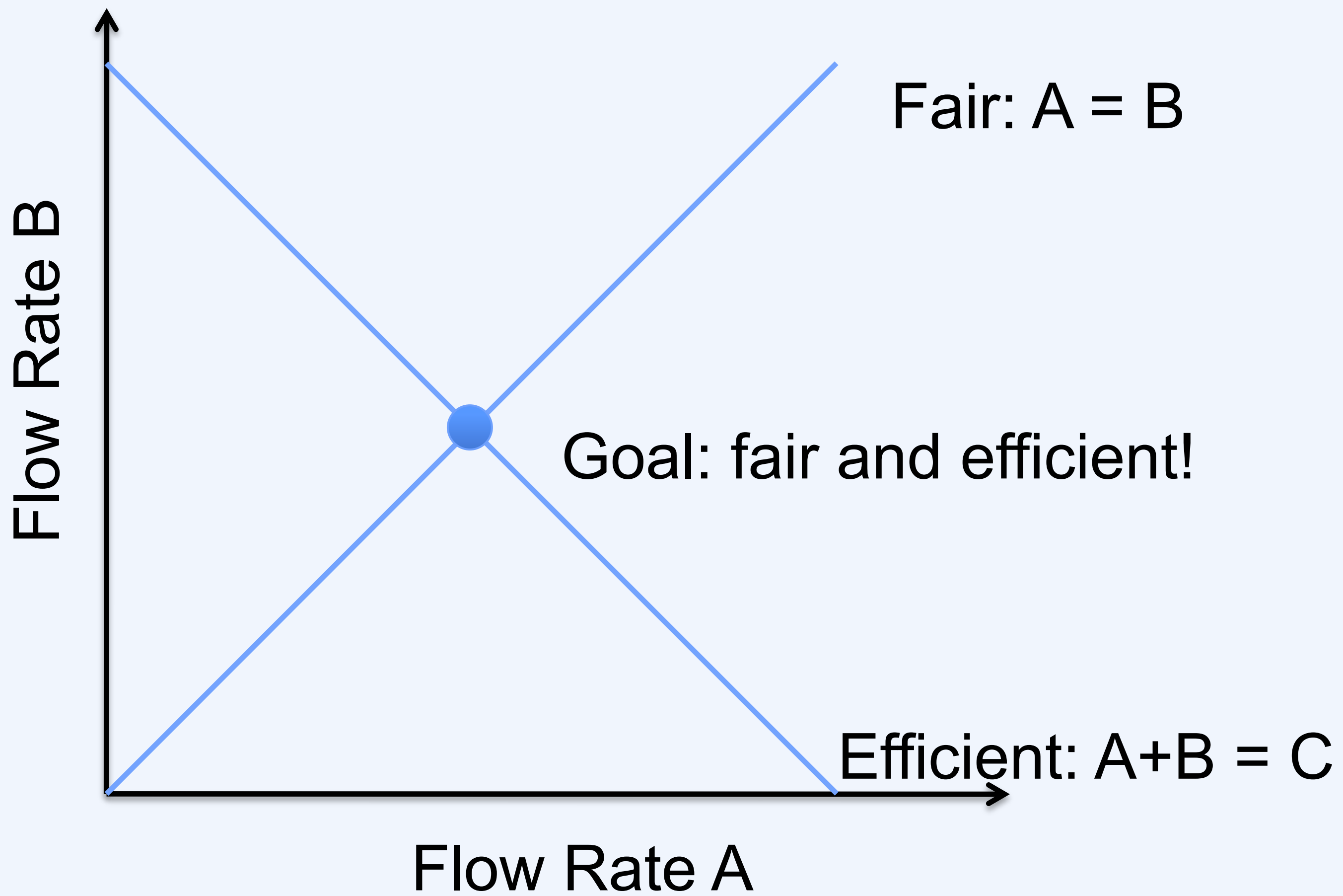
5)



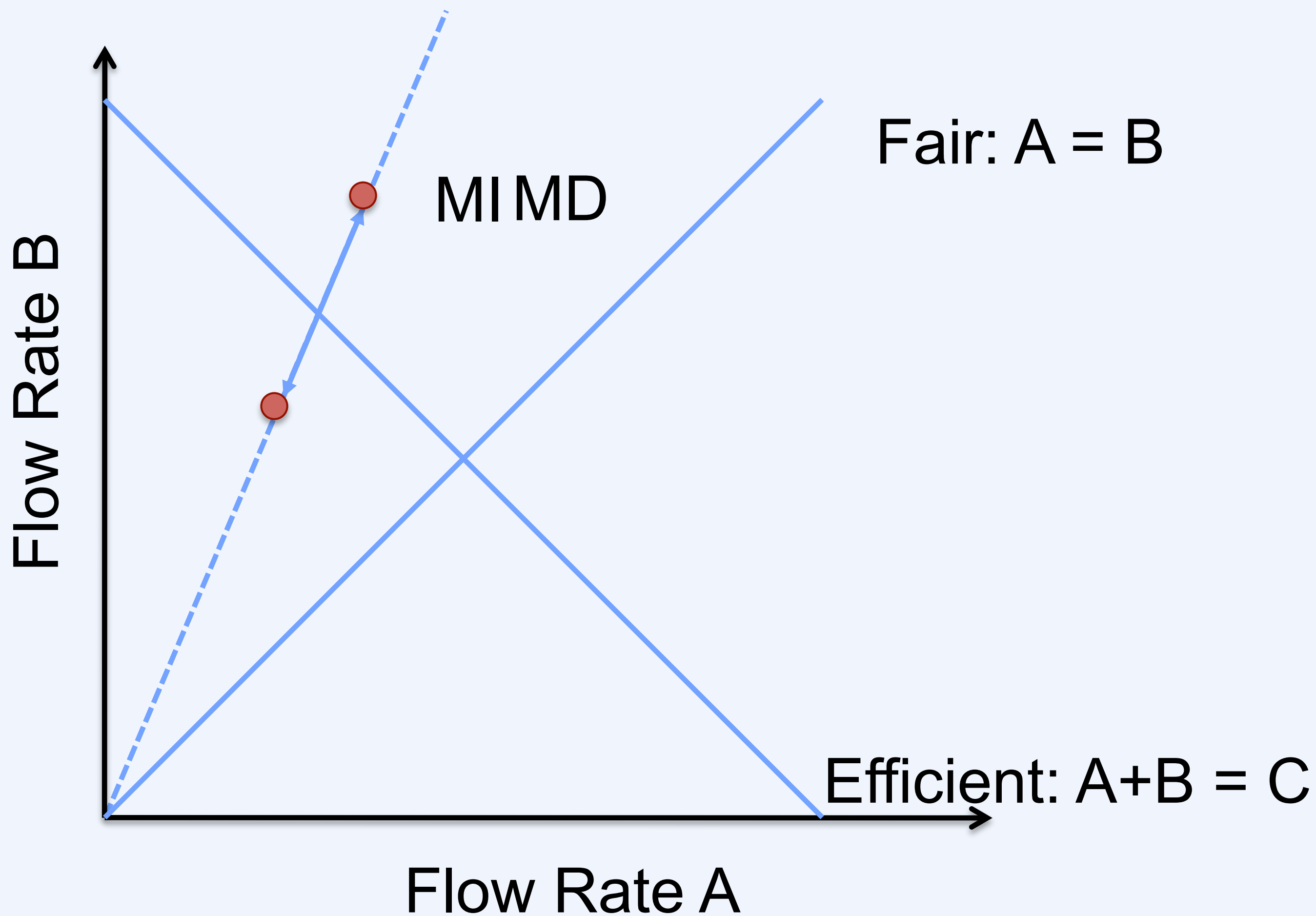
How to use extra capacity?

- Network signals congestion, but says nothing of underutilization
  - Senders constantly try to send faster, see if it works
  - So, increase window if no losses… By how much?
- Multiplicative increase?
  - Easier to saturate the network than to recover
  - Too fast, will lead to saturation, wild fluctuations
- Additive increase?
  - Won’t saturate the network
  - Remember fairness?

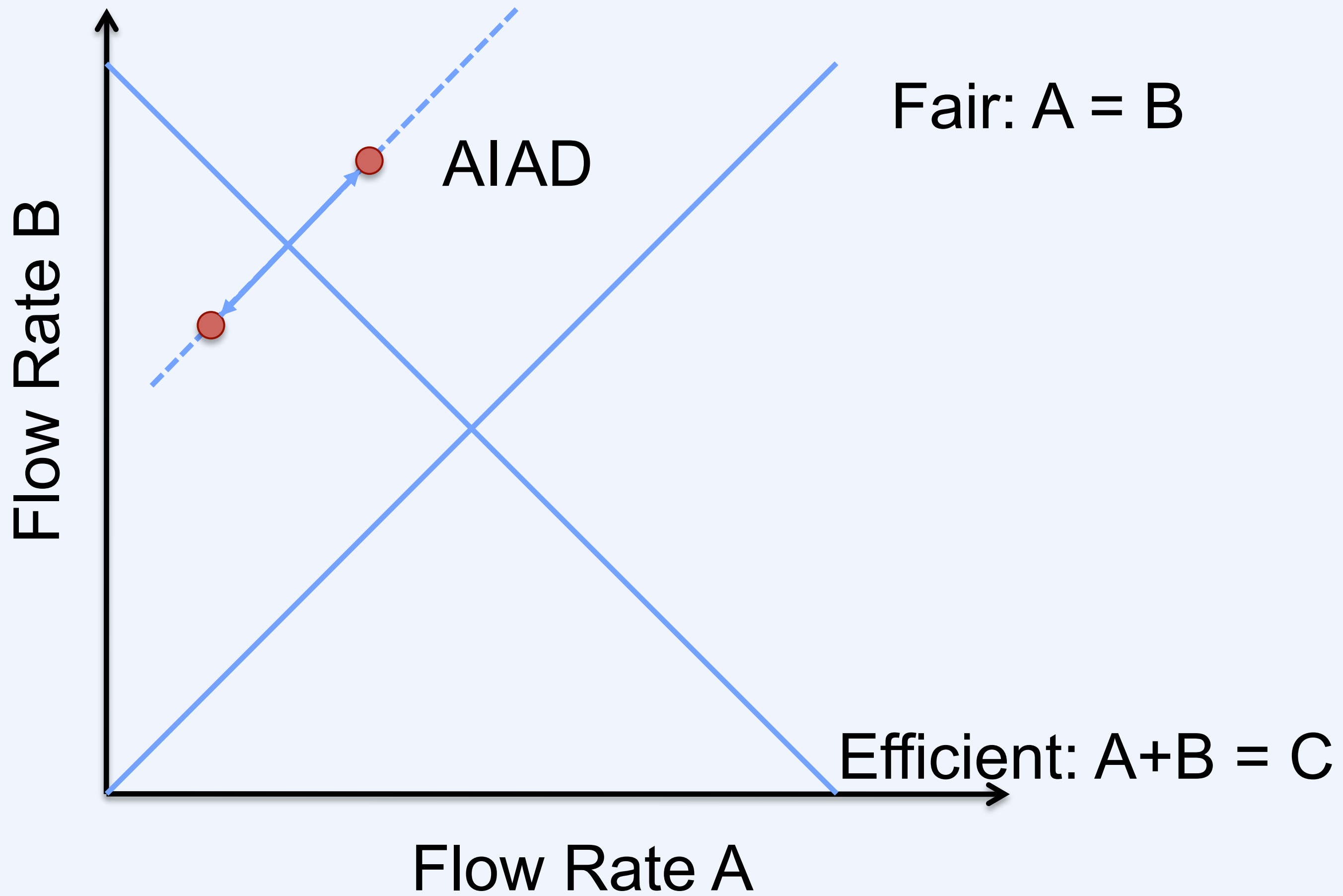
Chiu Jain Phase Plots



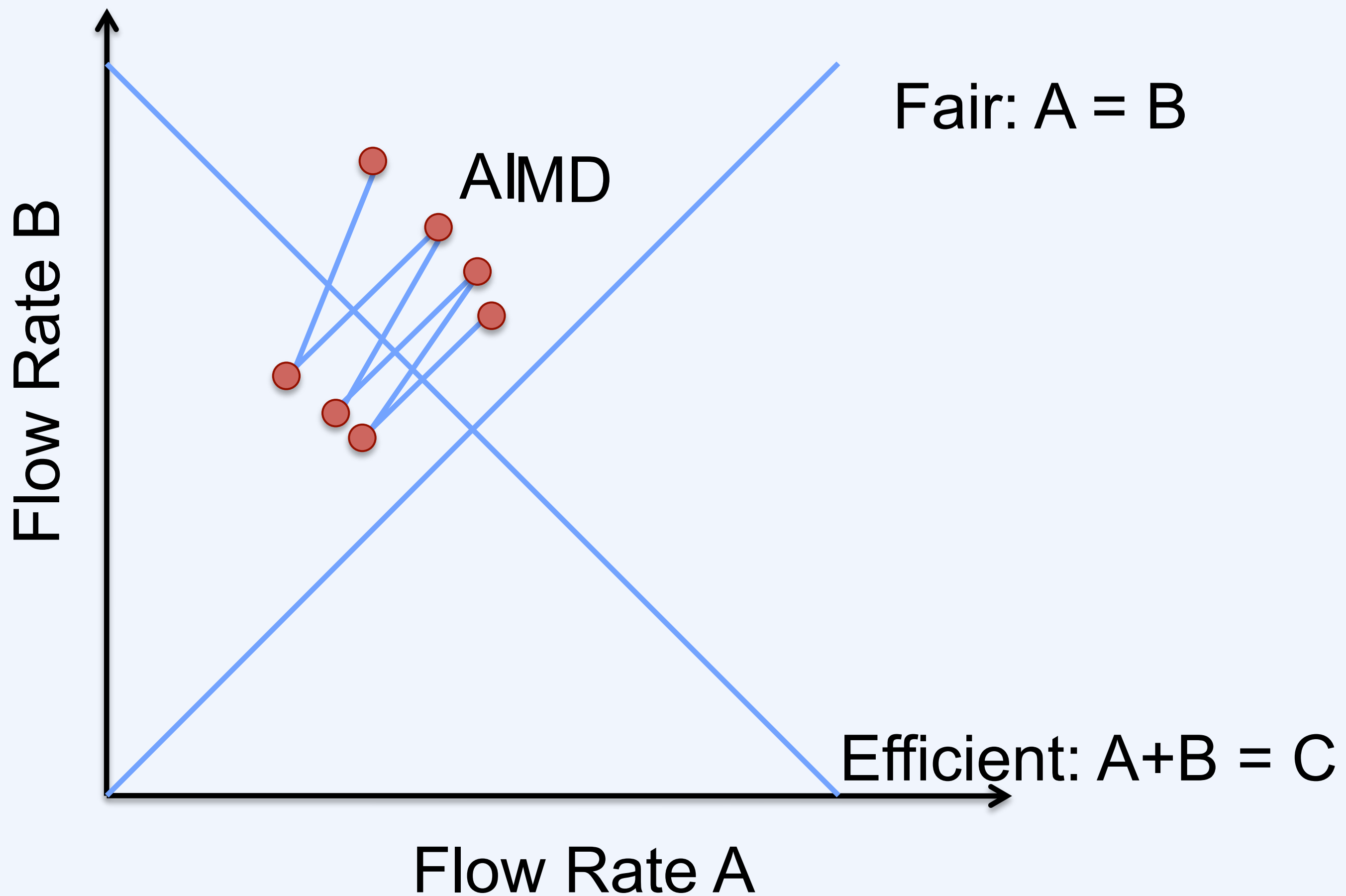
Chiu Jain Phase Plots



Chiu Jain Phase Plots

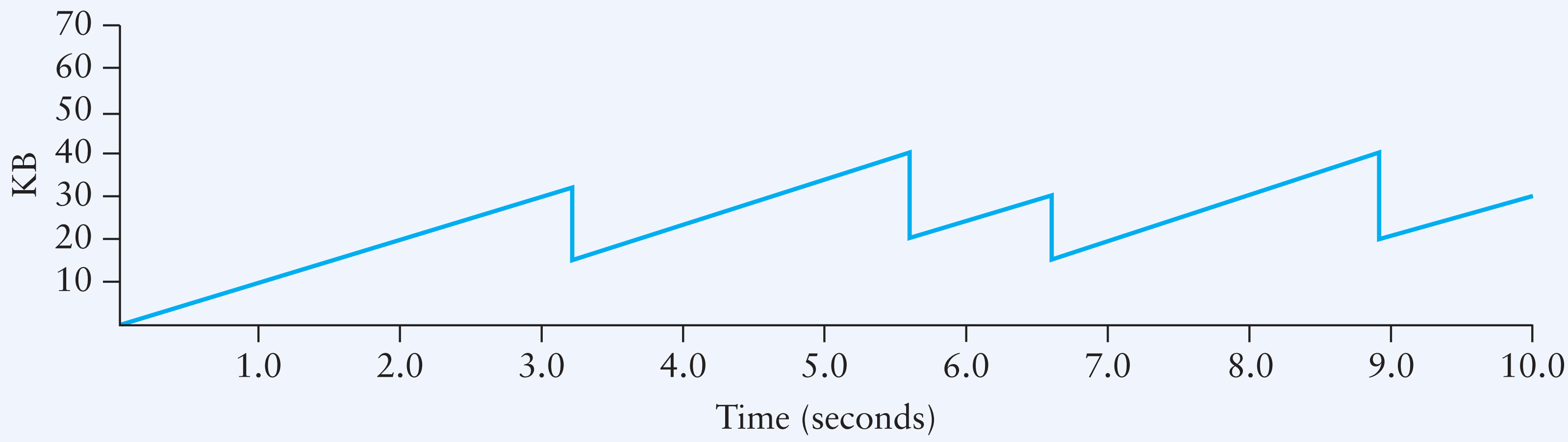


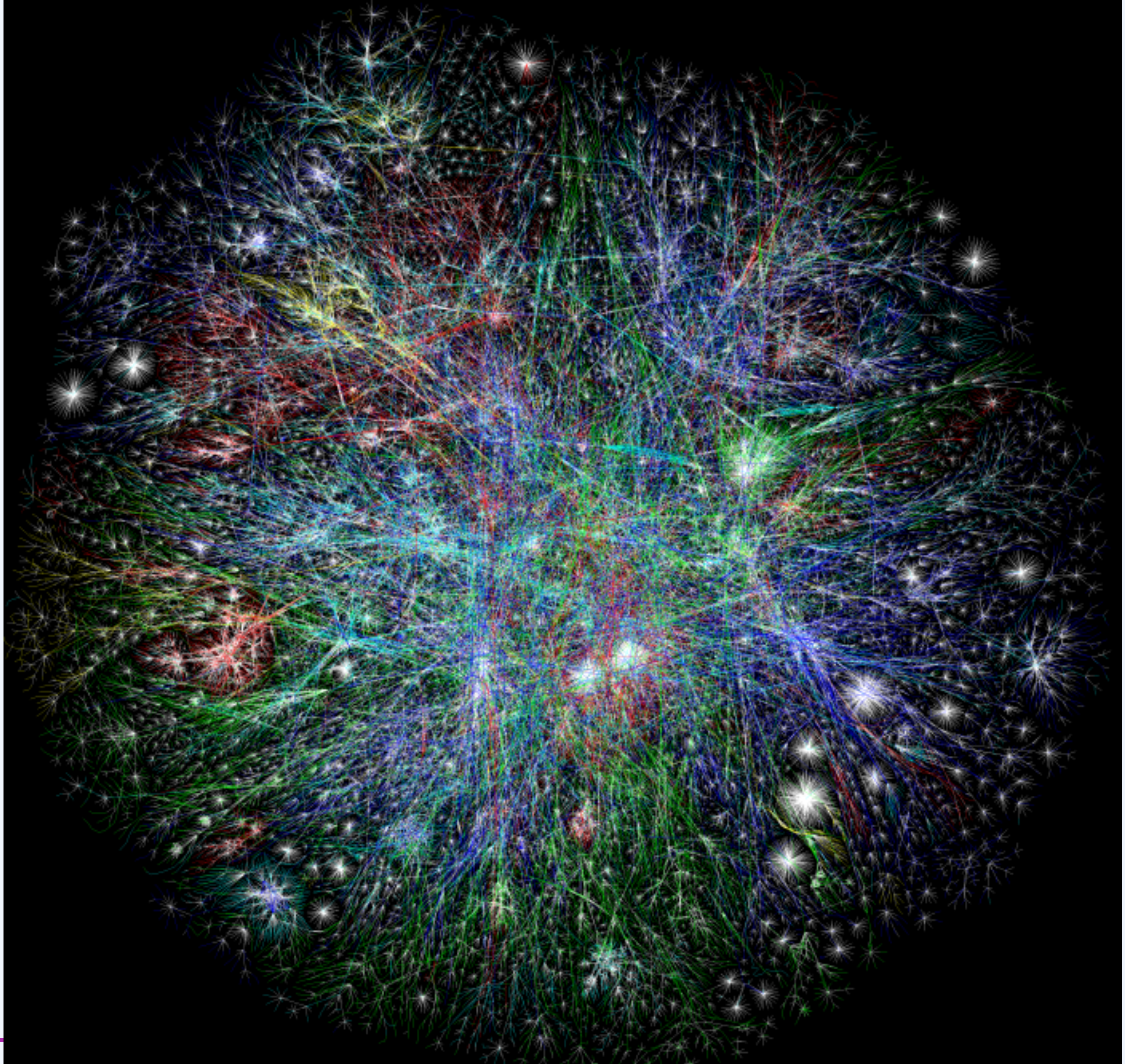
Chiu Jain Phase Plots



AIMD Trace

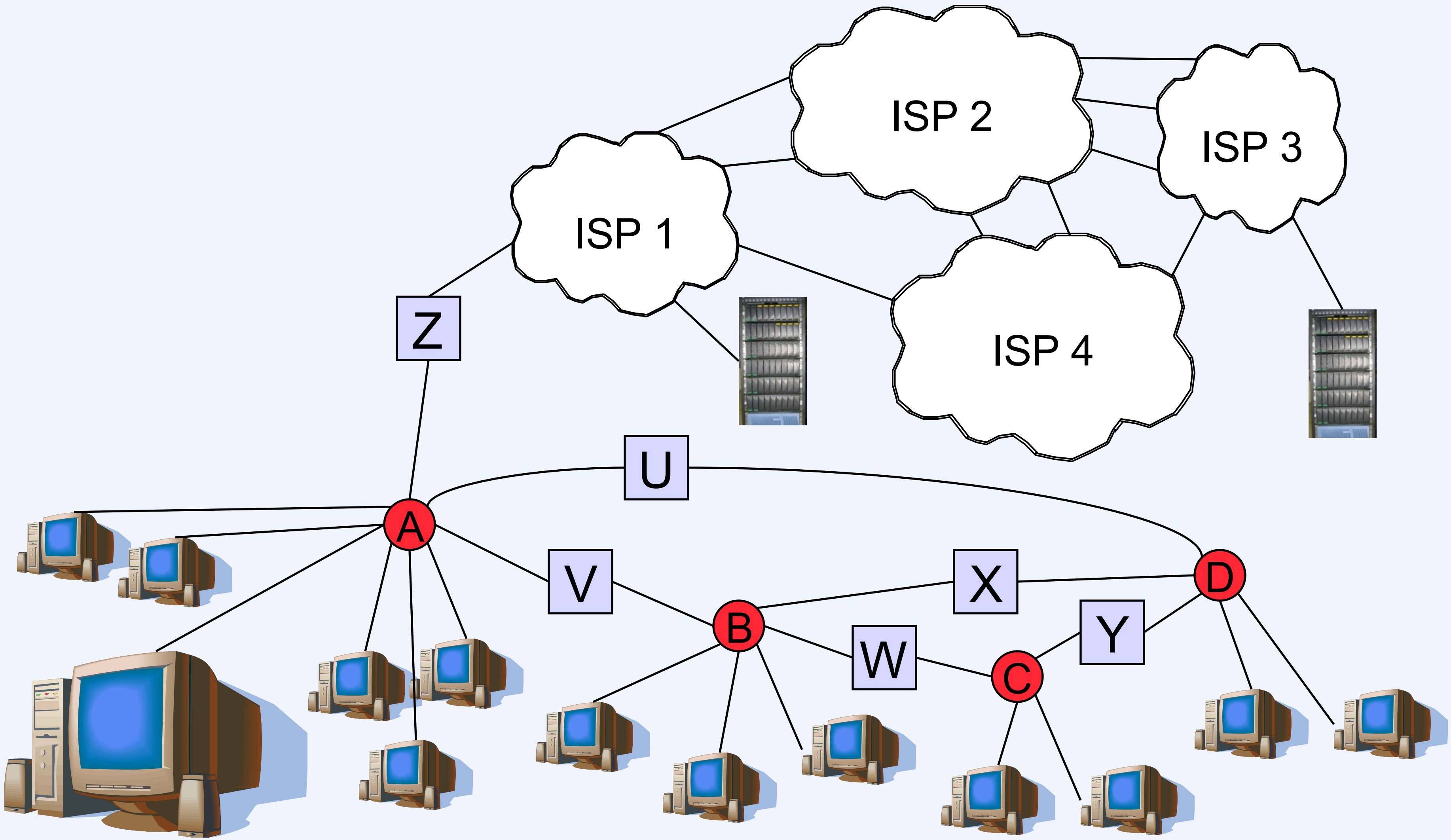
- **AIMD produces sawtooth pattern of window size**
  - **Always probing available bandwidth**





Traceroute map of the Internet, ~5 million edges, circa 2003. opte.org

Finding a Route



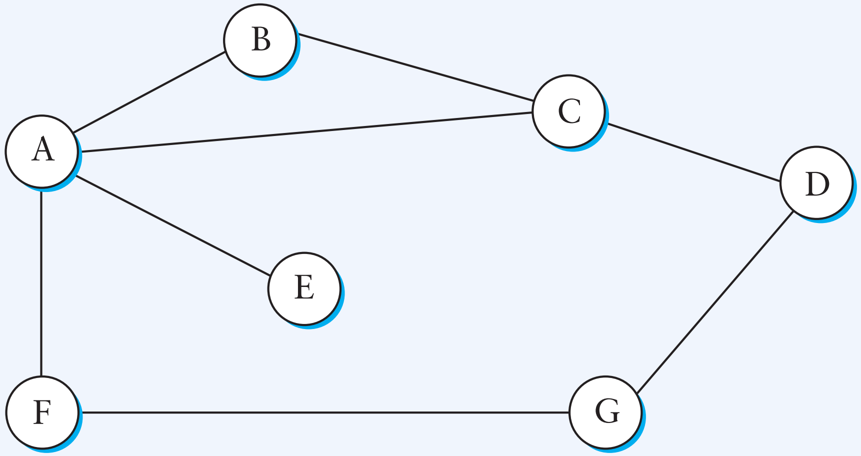
Types of Routing Protocols

- Interior
  - RIP
  - OSPF
- Exterior Routing
  - Exterior Gateway Protocol (EGP)
    - obsolete
  - Border Gateway Protocol (BGP)
    - RFC 1771

Dynamic Routing

- Distance vector
  - each router maintains table containing best known distance to each destination and the first hop of the associated path
  - routers periodically exchange tables with their neighbors
- Link state
  - each router maintains table containing description of entire network
  - each router computes best route based on global information
  - routers periodically send all others their local network-description information (i.e., the state of their connections with neighbors)

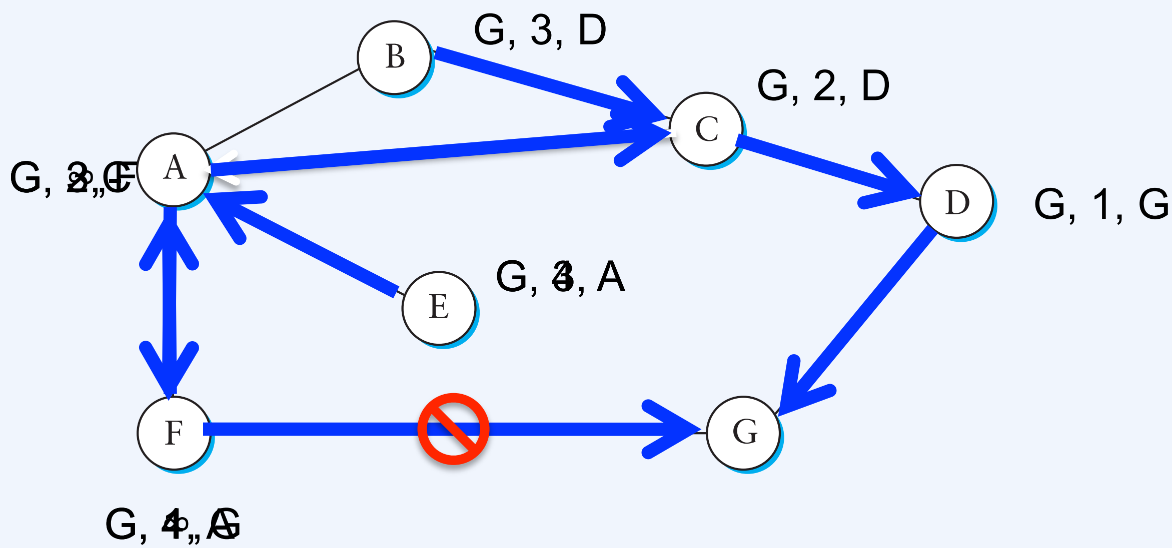
DV Example



B's routing table

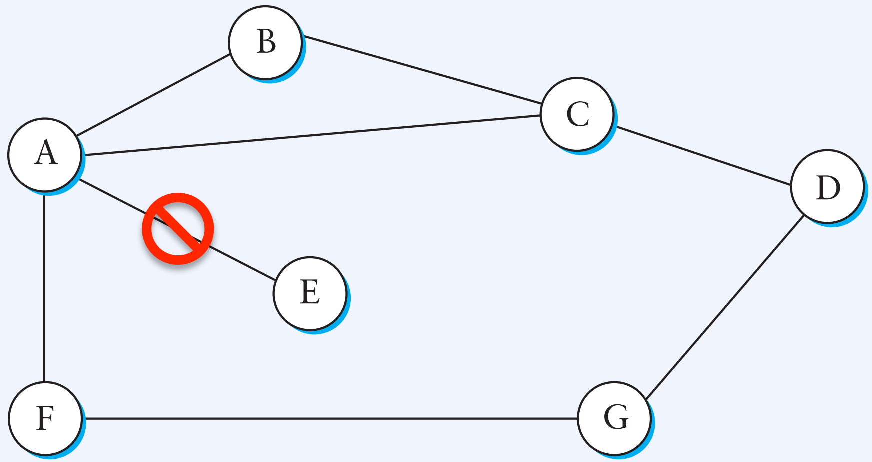
Destination	Cost	Next Hop
A	1	A
C	1	C
D	2	C
E	2	A
F	2	A
G	3	A

Adapting to Failures



- F-G fails
- F sets distance to G to infinity, propagates
- A sets distance to G to infinity
- A receives periodic update from C with 2-hop path to G
- A sets distance to G to 3 and propagates
- F sets distance to G to 4, through A

Count-to-Infinity



- Link from A to E fails
- A advertises distance of infinity to E
- B and C advertise a distance of 2 to E
- B decides it can reach E in 3 hops through C
- A decides it can reach E in 4 hops through B
- C decides it can reach E in 5 hops through A, …
- When does this stop?

How to avoid loops

- **IP TTL field prevents a packet from living forever**
  - **Does not *repair* a loop**
- **Simple approach: consider a small cost *n* (e.g., 16) to be infinity**
  - **After *n* rounds decide node is unavailable**
  - **But rounds can be long, this takes time**
- **Problem: distance vector based only on local information`**

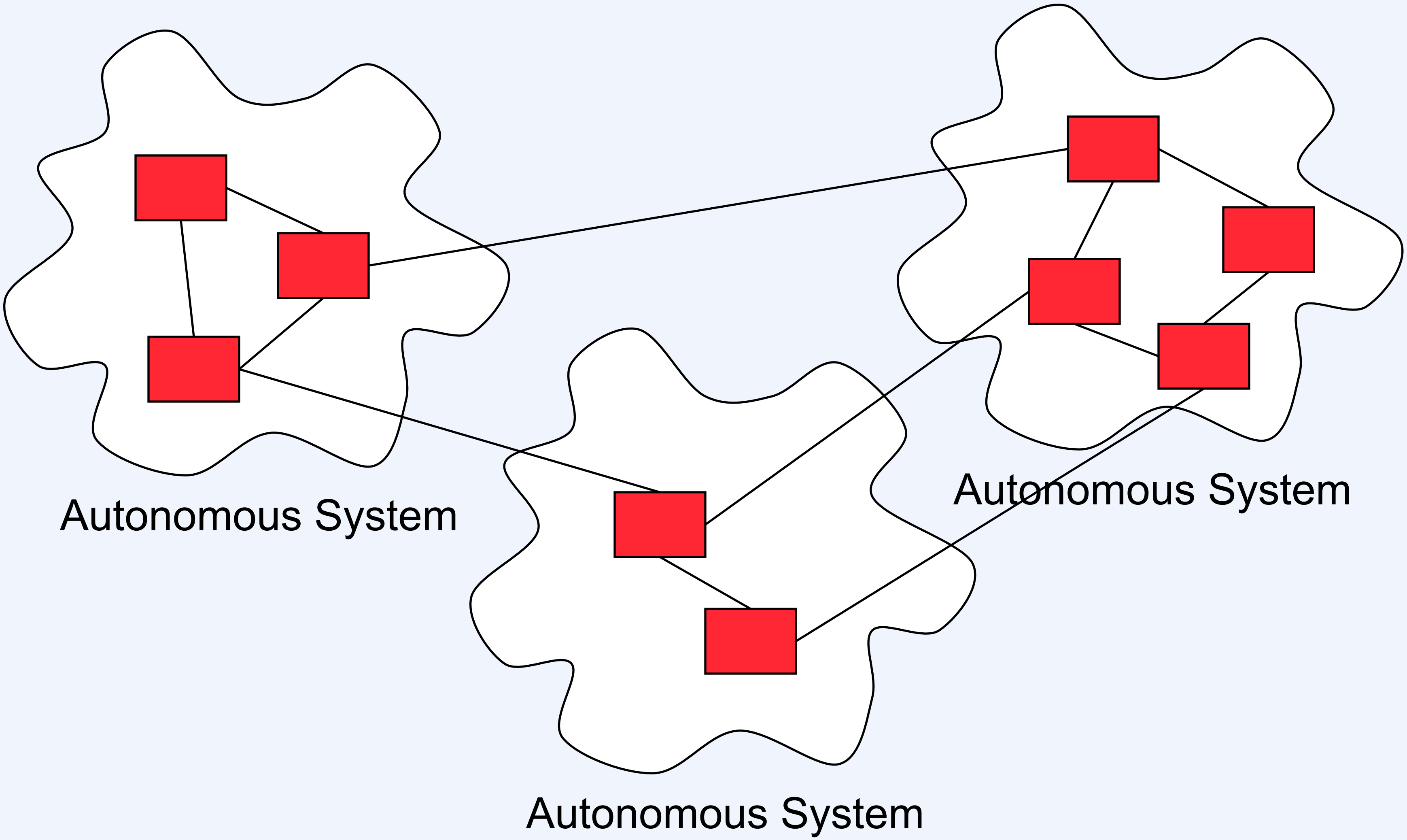
Better loop avoidance

- Split Horizon
  - When sending updates to node A, don't include routes you learned from A
  - Prevents B and C from sending cost 2 to A
- Split Horizon with Poison Reverse
  - Rather than not advertising routes learned from A, explicitly include cost of ∞.
  - Faster to break out of loops, but increases advertisement sizes

Warning

- Split horizon/split horizon with poison reverse only help between two nodes
  - Can still get loop with three nodes involved
  - Might need to delay advertising routes after changes, but affects convergence time

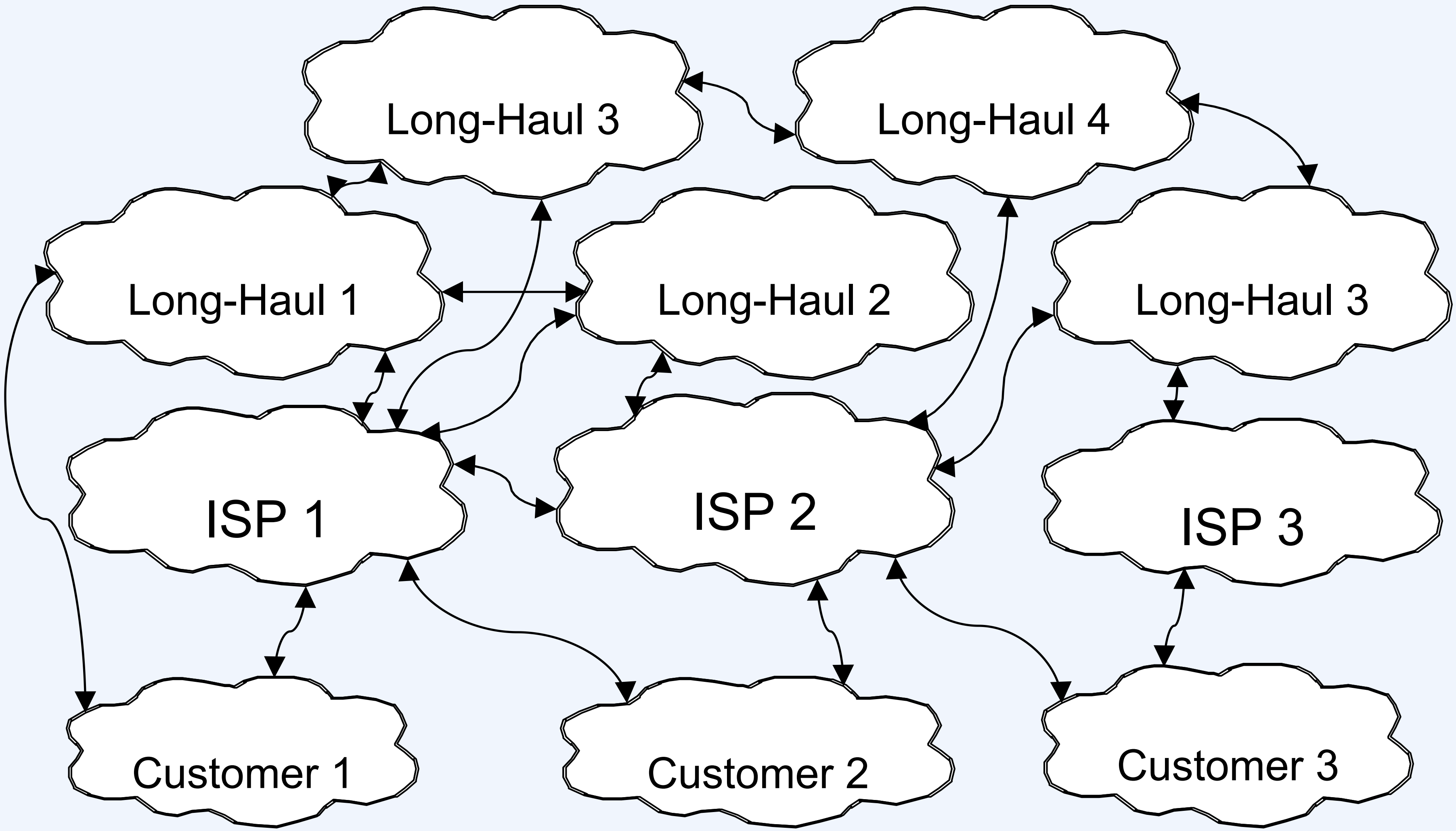
Autonomous Systems



Types of Routing Protocols

- Interior
  - RIP
  - OSPF
- Exterior Routing
  - Exterior Gateway Protocol (EGP)
    - obsolete
  - Border Gateway Protocol (BGP)
    - RFC 1771

Exterior Routing



Addressing

- Each AS has a 16-bit AS number
  - in short supply … later increased to 32 bits
- Ideally
  - easy mapping from IP address to AS number
- But no …
  - people move machines
  - highly decentralized (hierarchical) process
  - change ISPs
- Result
  - routers with no default entries have *lots* of entries
    - Over 500,000 entries as of 2015

BGP

- References: RFCs 1771, 1772, 1773, 1774
- A path-vector protocol
  - unit of routing is the AS
  - avoids count-to-infinity problems by using path vectors (of ASs)
  - routers must enforce policy constraints
  - no notion of determining optimal routes: reachability is the goal

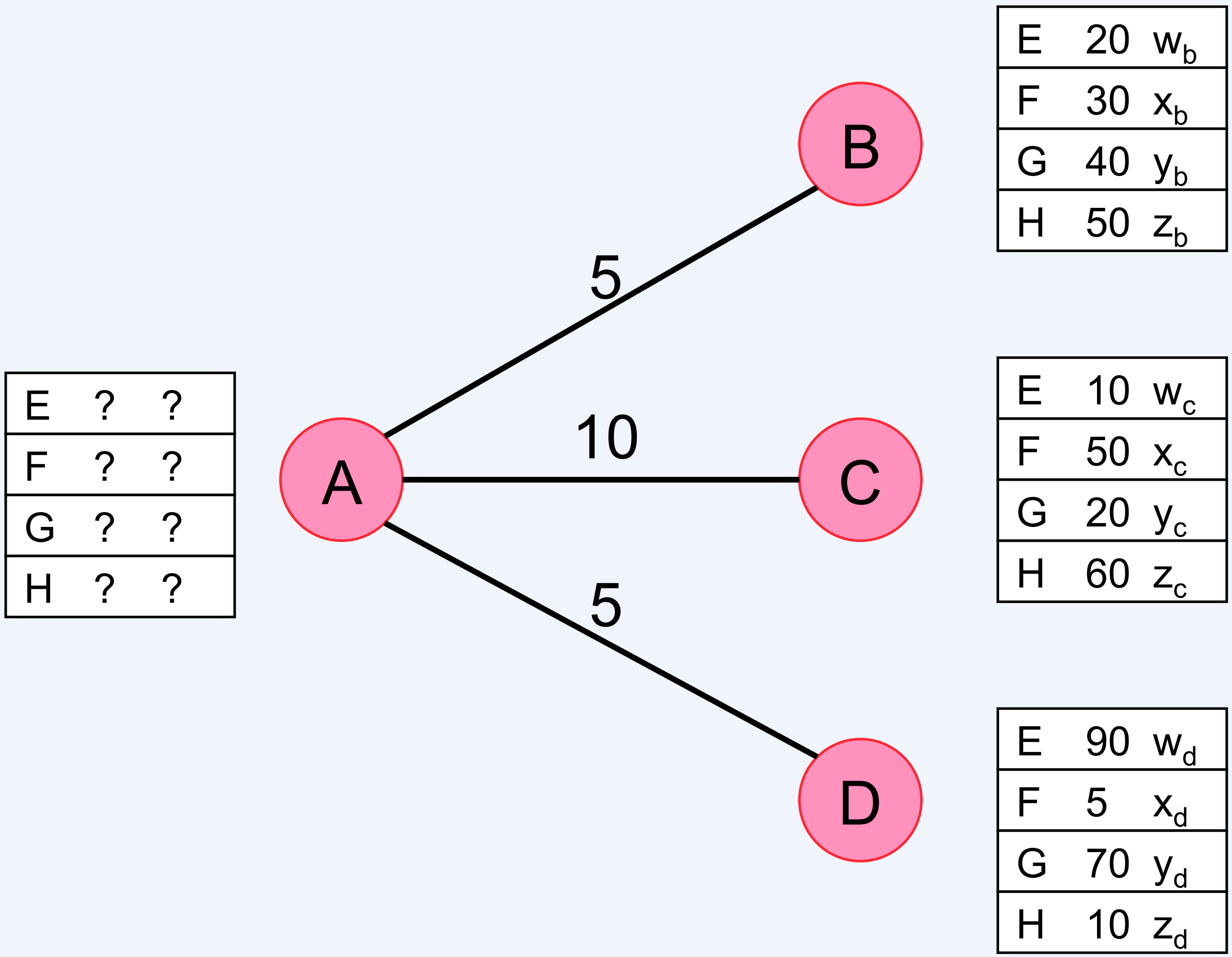
Extra Slides

- These were not covered in class, but are really good if you want to understand classic routing protocols better.

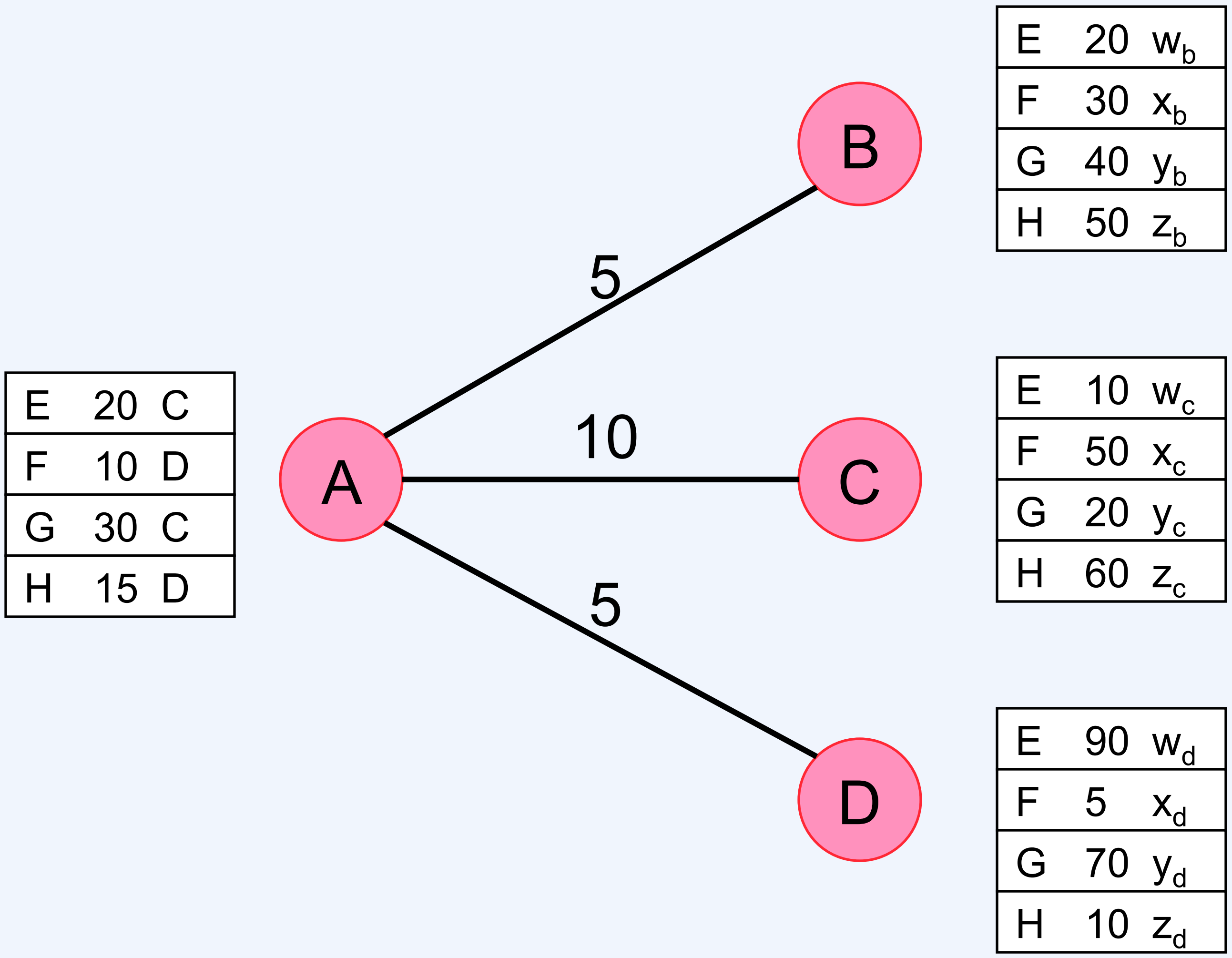
Routing Information Protocol (RIP)

- Distance-vector protocol
- In use since the ’70s
- Formerly in widespread use
  - implemented on Unix as *routed*
- Described in RFCs 1058, 1723

RIP Theory (1)

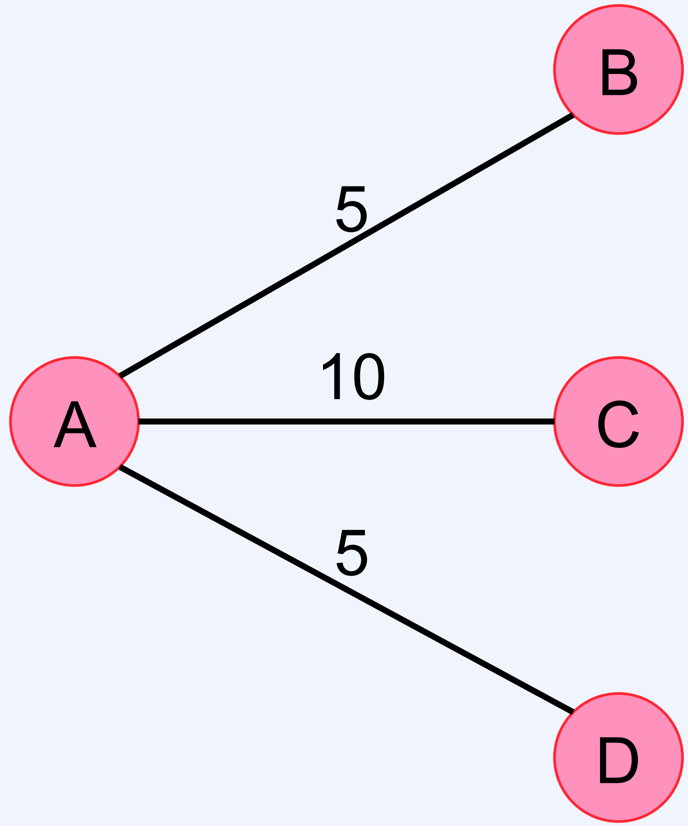

E ? ?
F ? ?
G ? ?
H ? ?
A
B
C
D
5
10
5
E 20 w_b
F 30 x_b
G 40 y_b
H 50 z_b
E 10 w_c
F 50 x_c
G 20 y_c
H 60 z_c
E 90 w_d
F 5 x_d
G 70 y_d
H 10 z_d

RIP Theory (2)



E 20 C
F 10 D
G 30 C
H 15 D
A
5
10
5
B
C
D
E 20 w_b
F 30 x_b
G 40 y_b
H 50 z_b
E 10 w_c
F 50 x_c
G 20 y_c
H 60 z_c
E 90 w_d
F 5 x_d
G 70 y_d
H 10 z_d

RIP Theory (3)



A:

E	20	C
F	10	D
G	30	C
H	15	D

B:

E	20	w_b
F	*15*	*A*
G	*35*	*A*
H	*20*	*A*

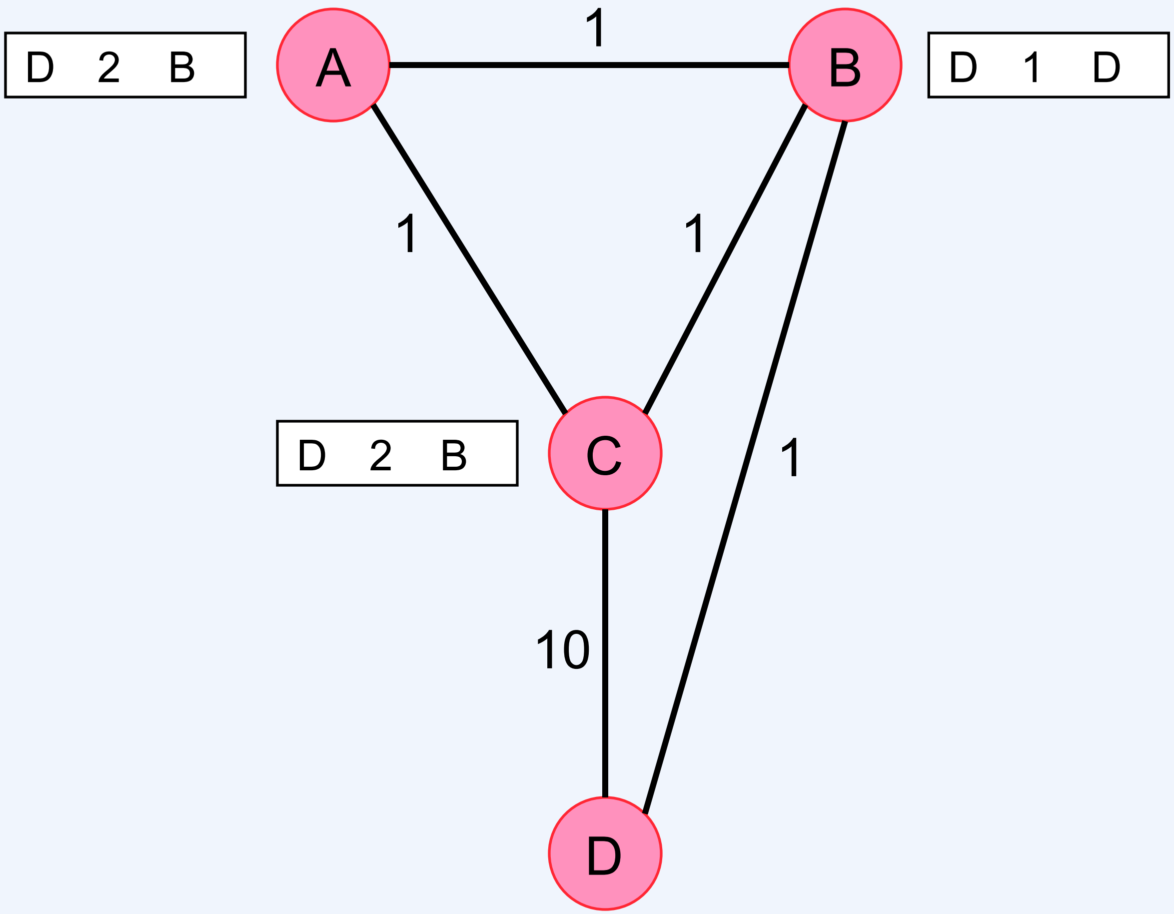
C:

E	10	w_c
F	*20*	*A*
G	20	y_c
H	*25*	*A*

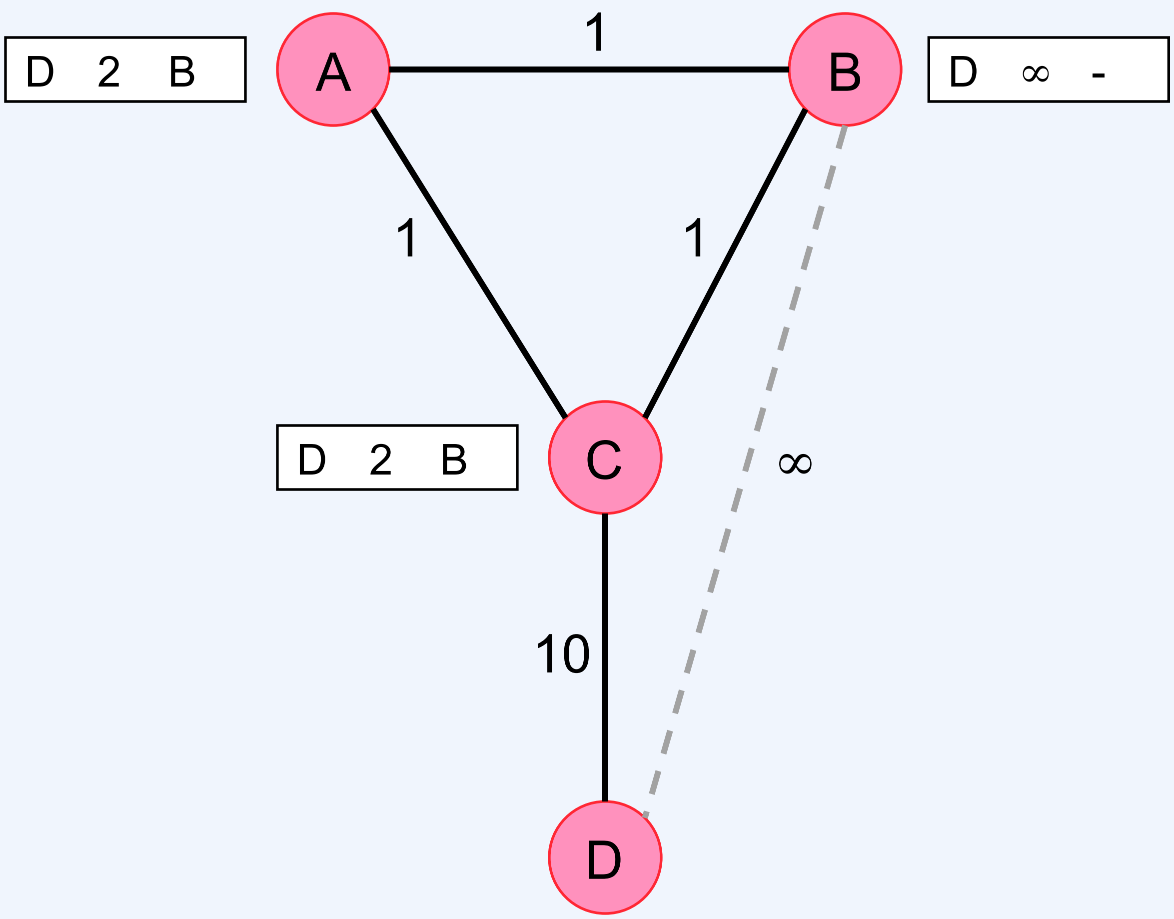
D:

E	*25*	*A*
F	5	x_d
G	*35*	*A*
H	10	z_d

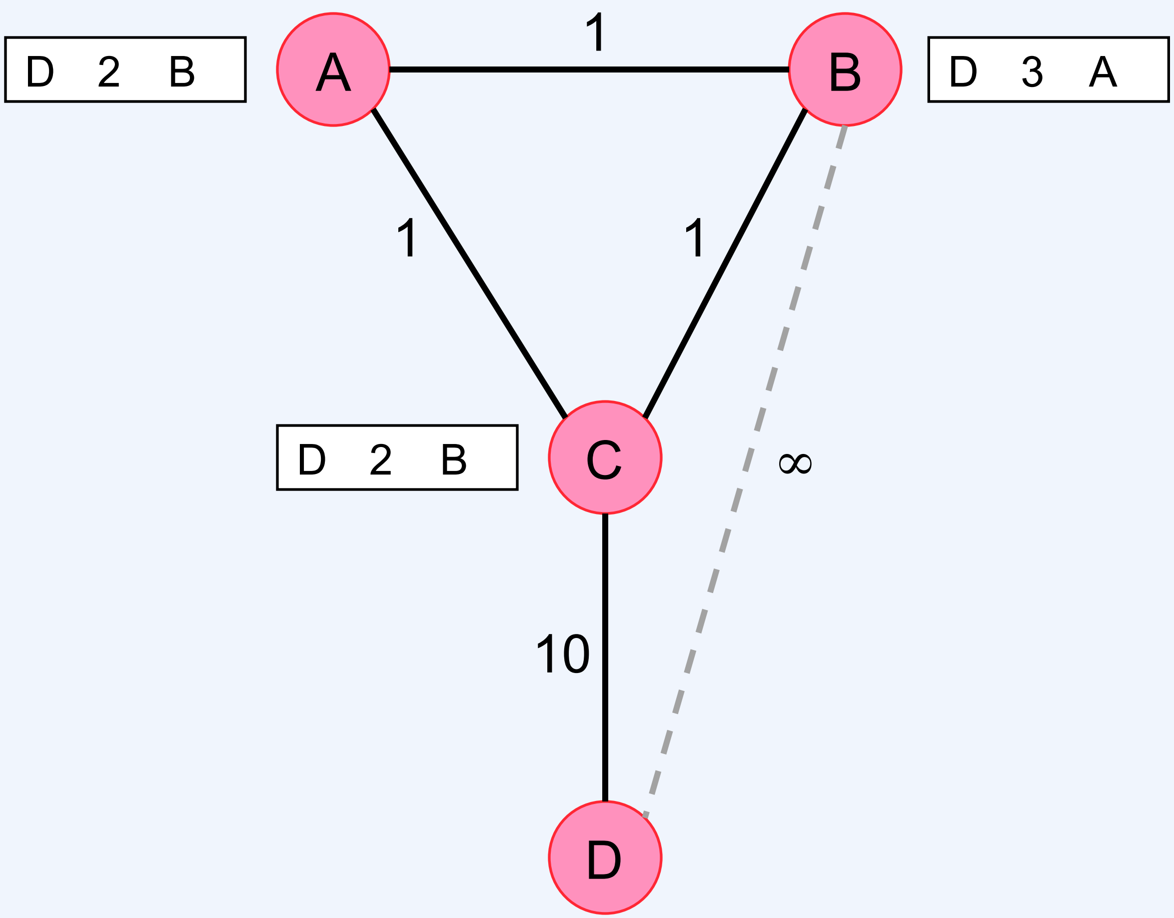
Loss of Route (1)



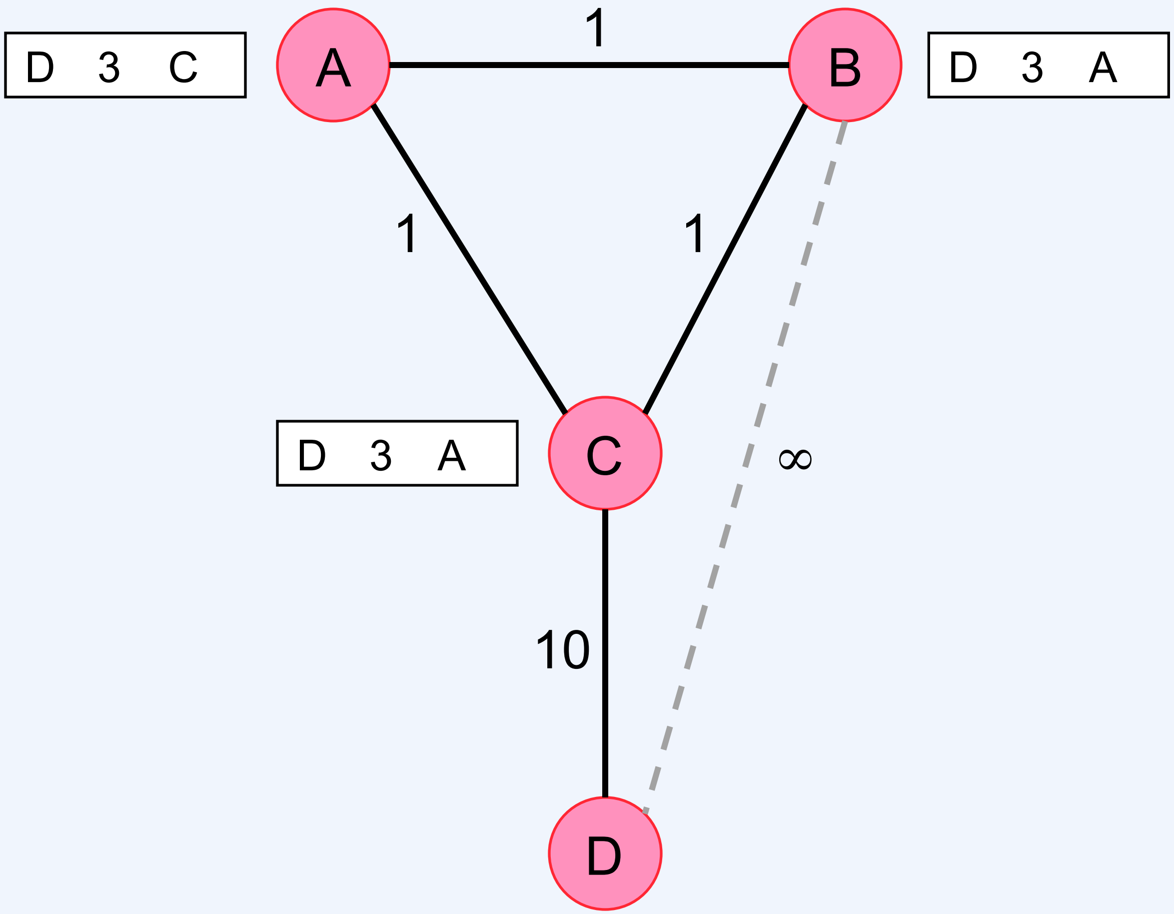
Loss of Route (2)



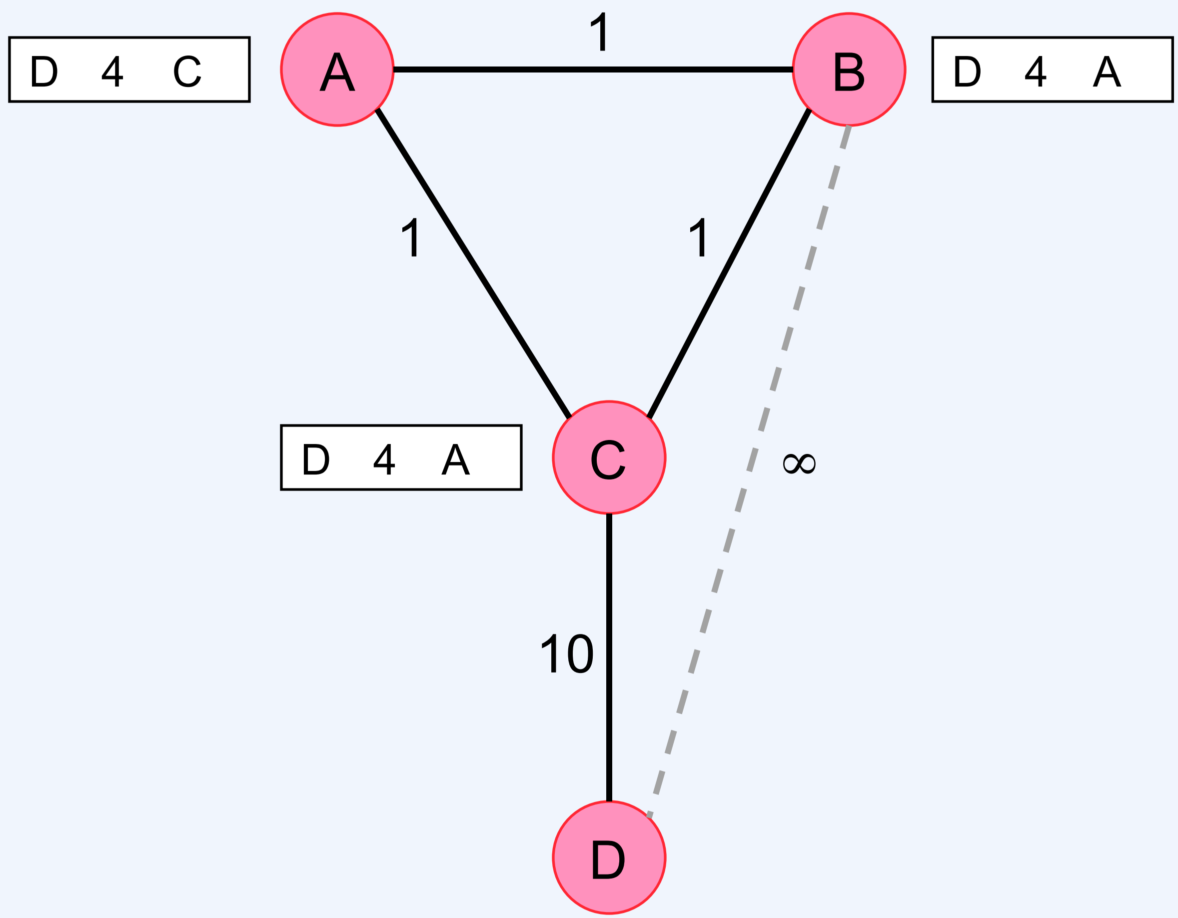
Loss of Route (3)



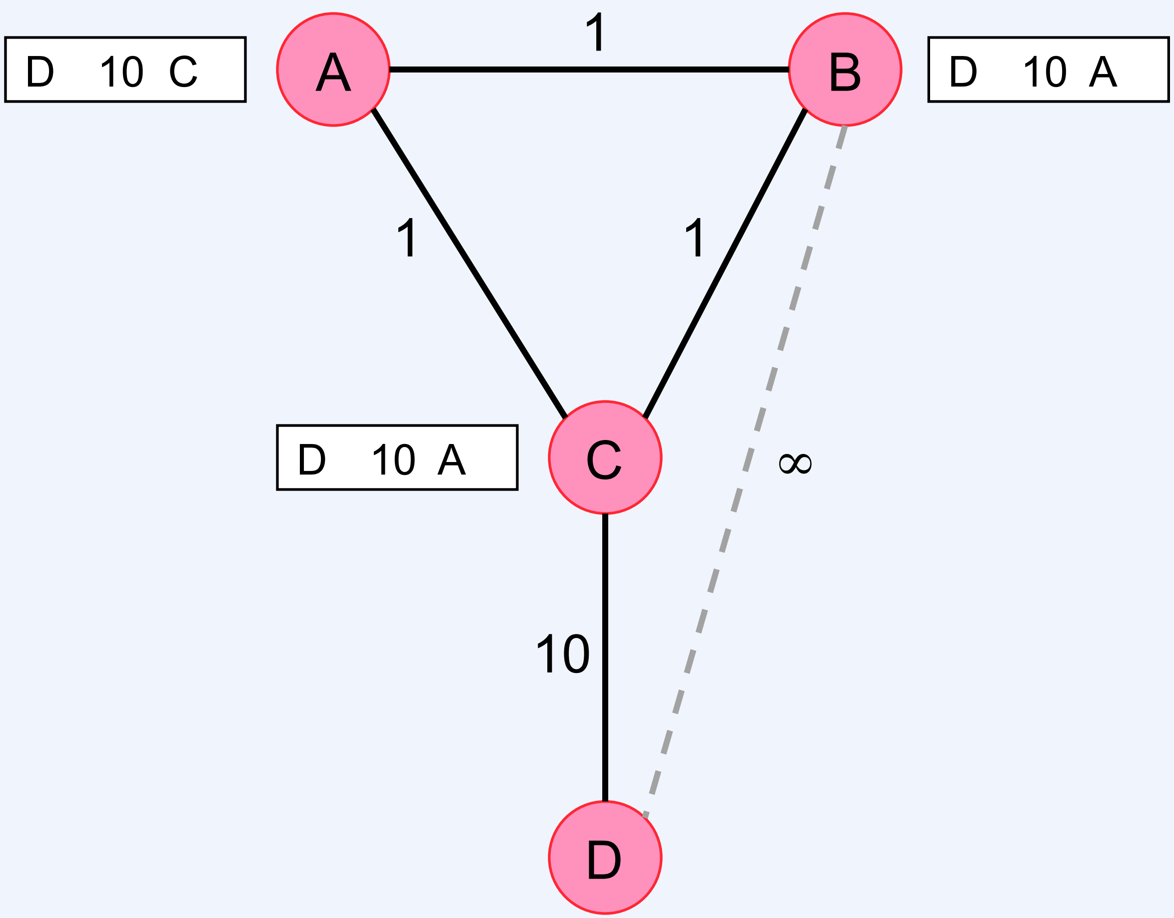
Loss of Route (4)



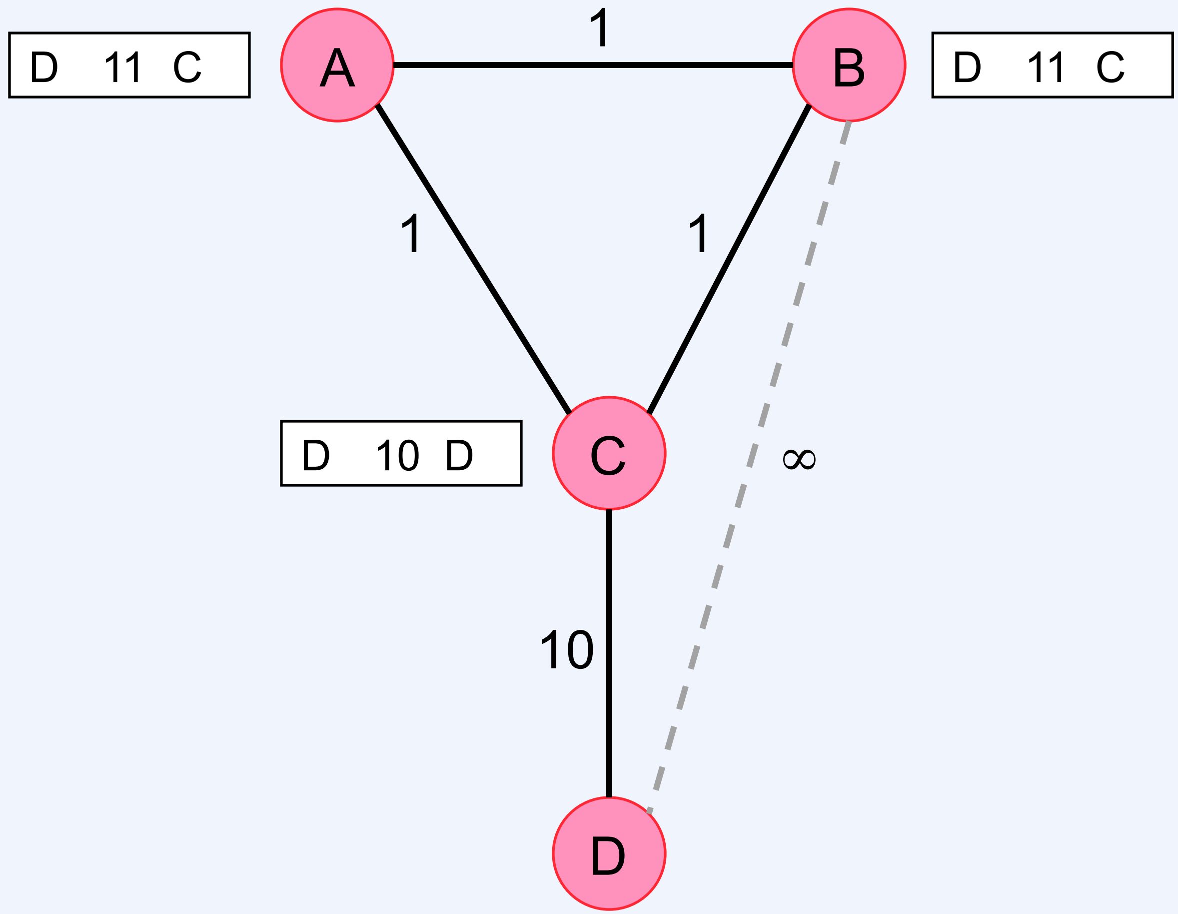
Loss of Route (5)



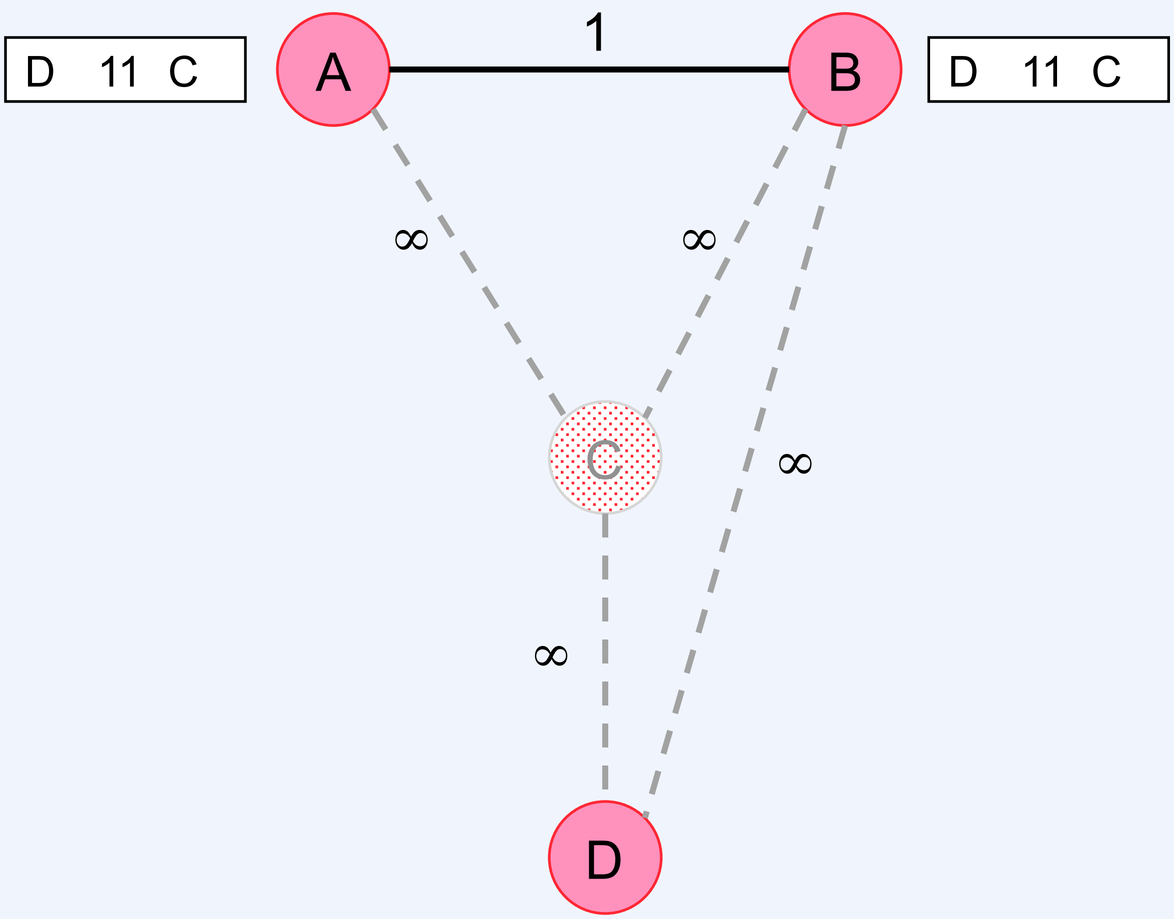
Loss of Route (6)



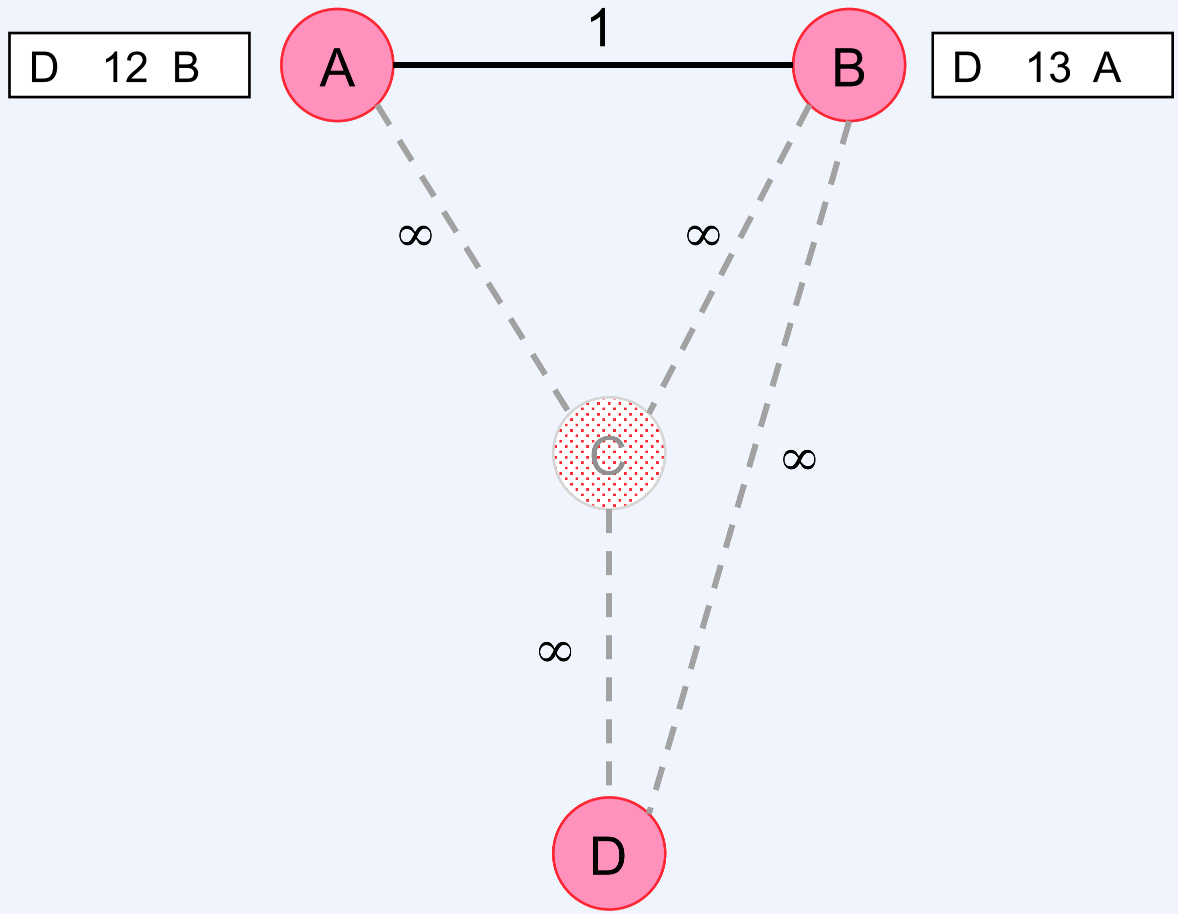
Loss of Route (7)



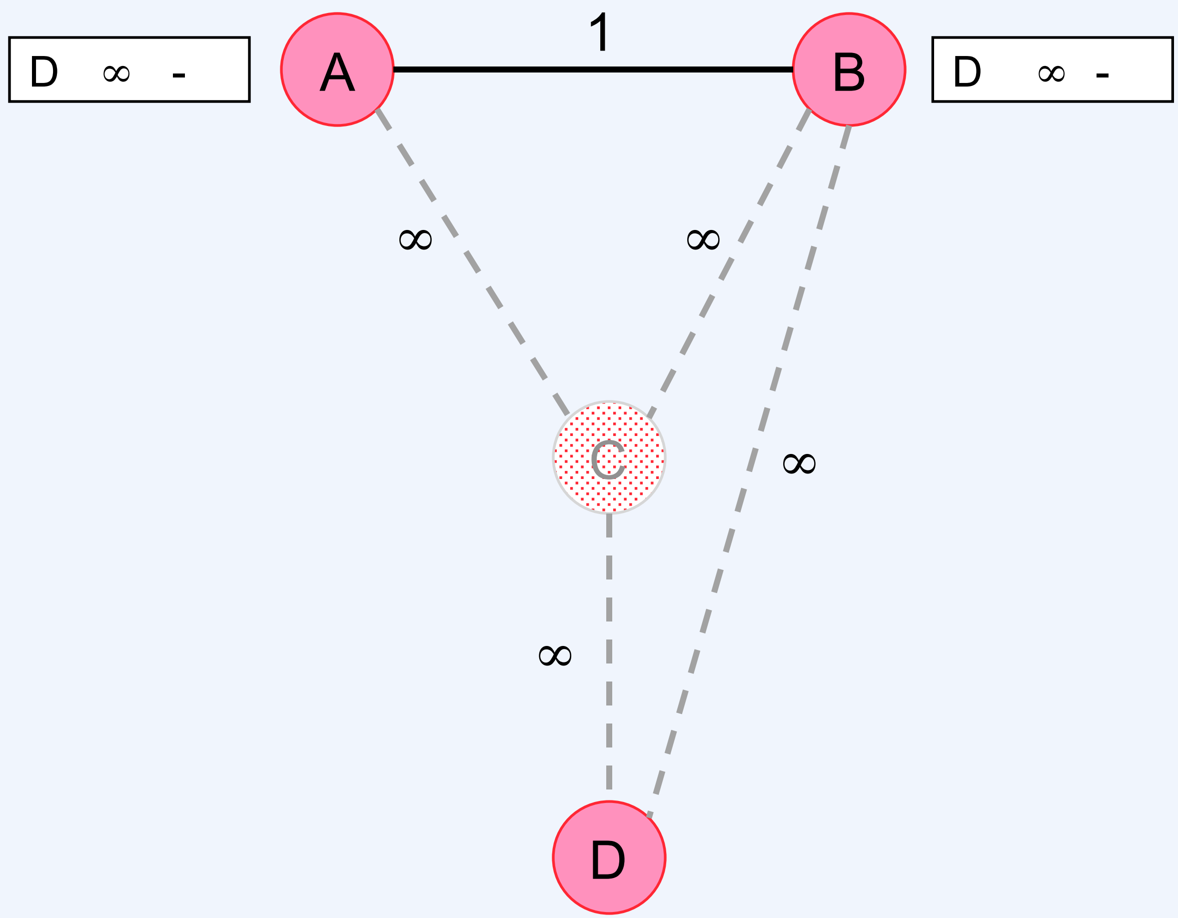
Loss of Route (8)



Loss of Route (9)



Loss of Route (∞)



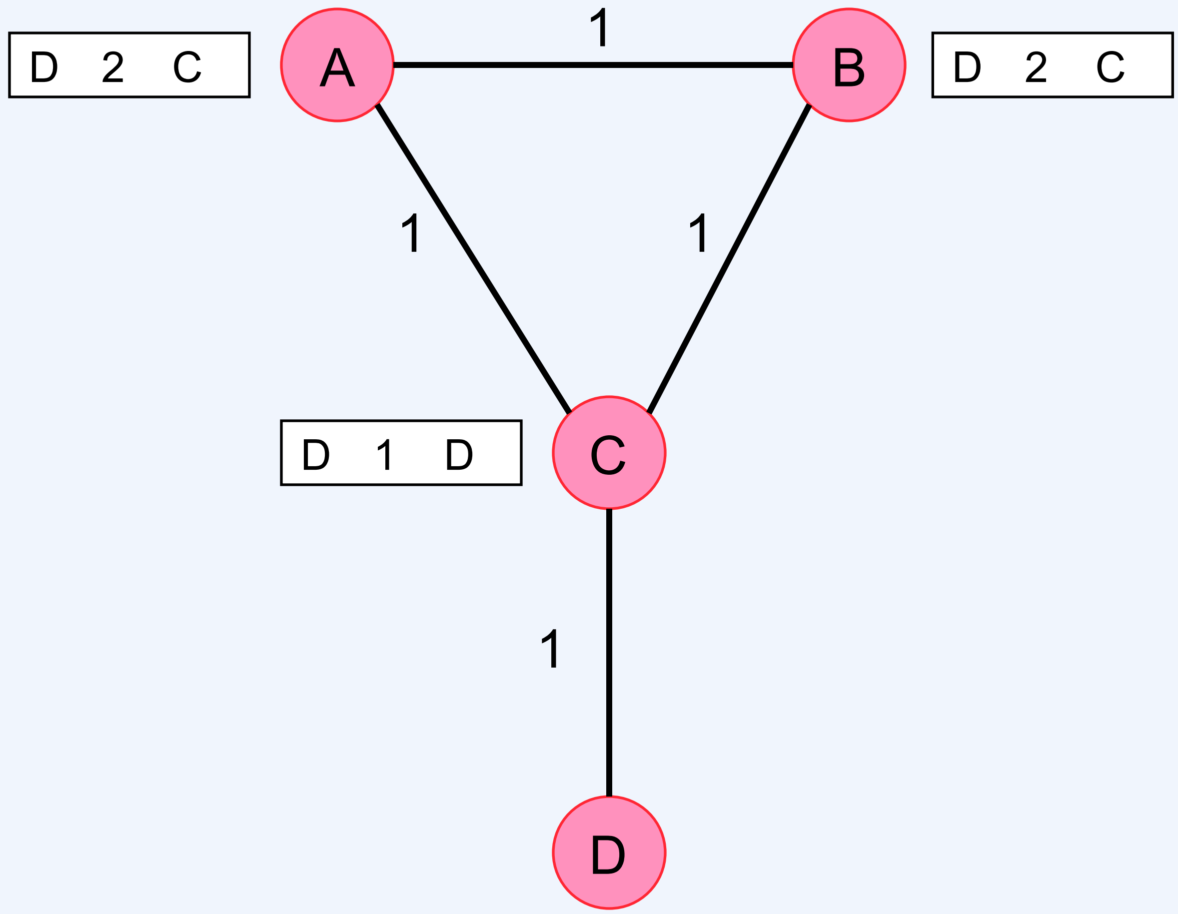
Counting to Infinity

- Make infinity small
  - 16 is chosen as infinity in RIP
  - longest path in network can be no longer than 15

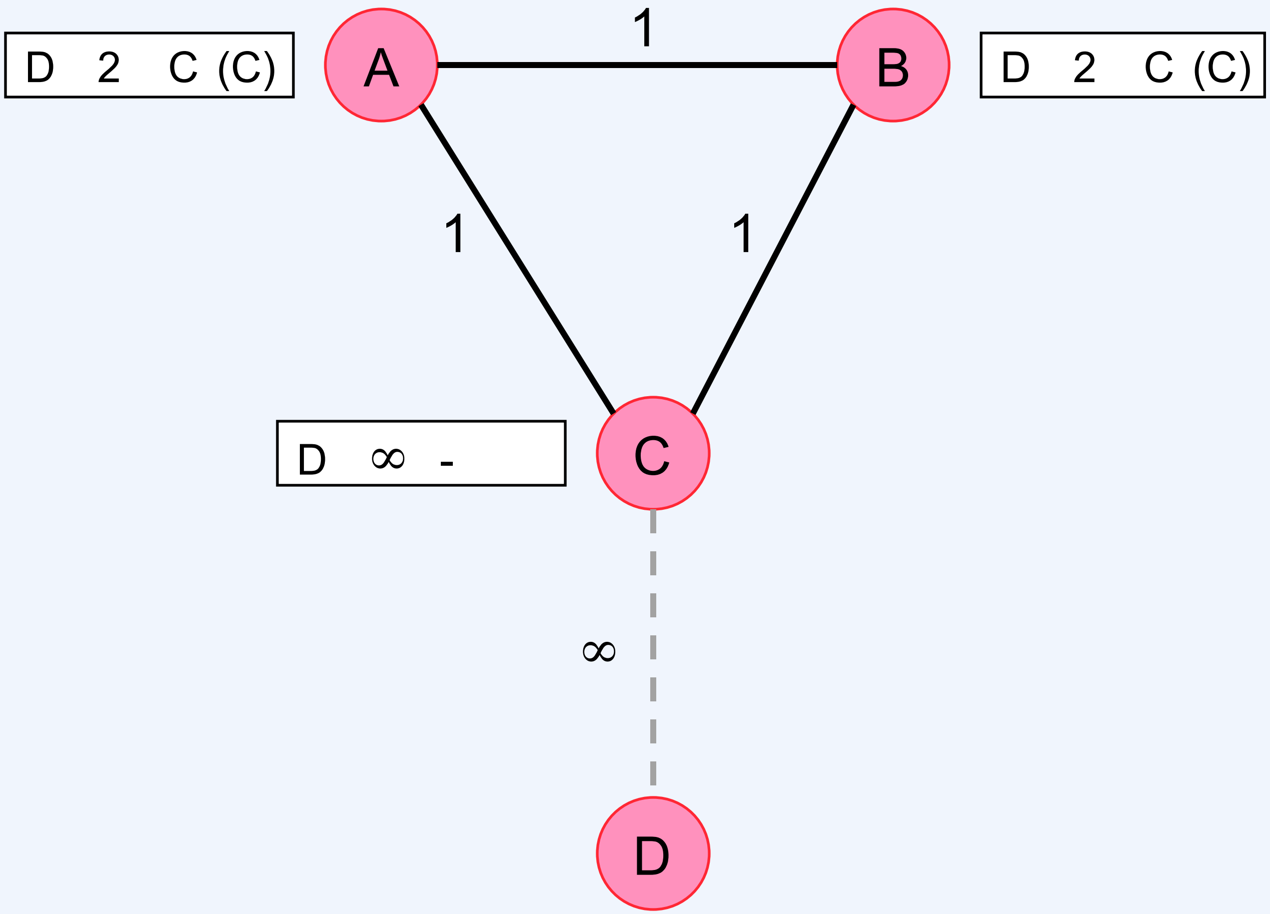
Split Horizon

- Don’t tell neighbor about routes obtained from it
  – quicker convergence in many cases

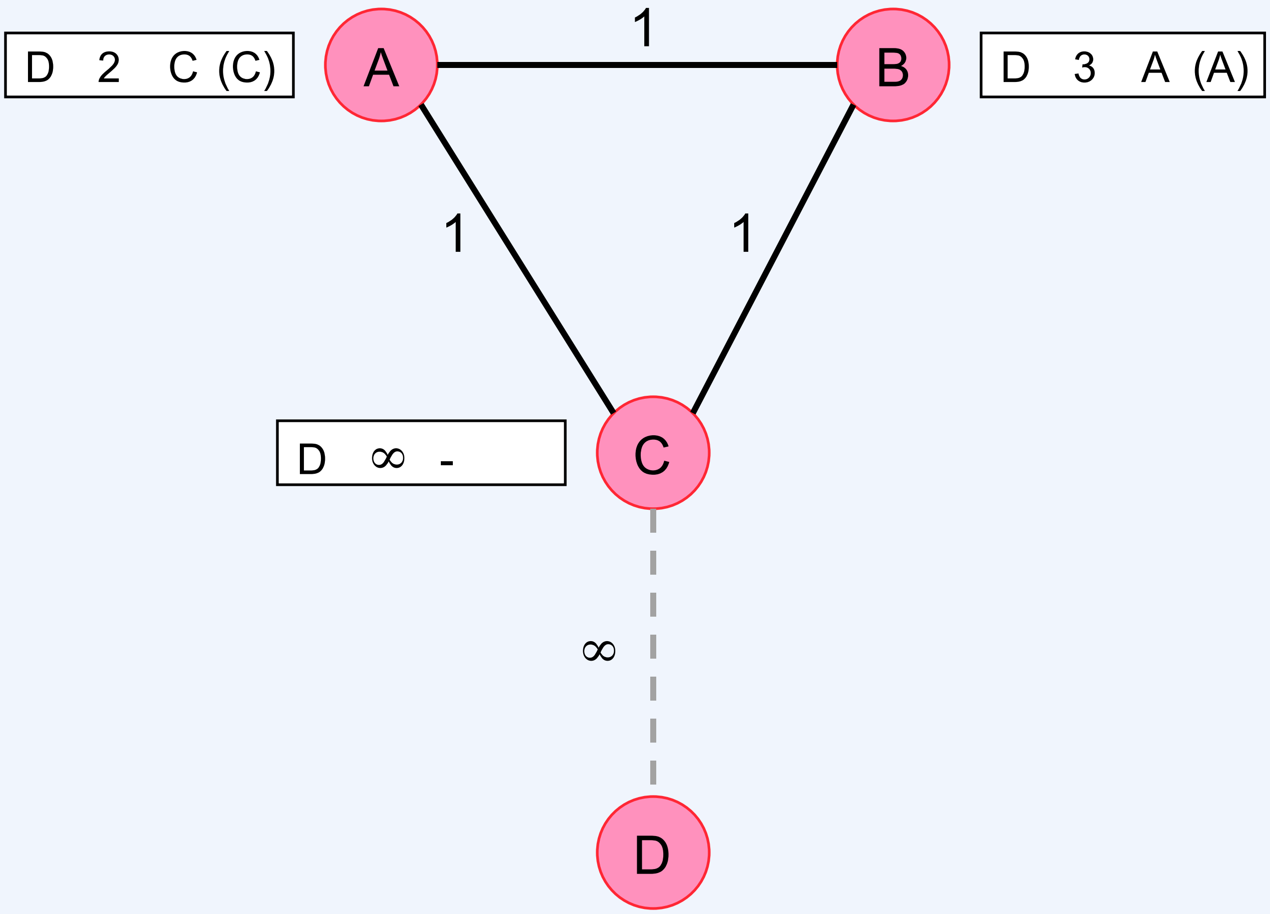
Split Horizon's Not Perfect (1)



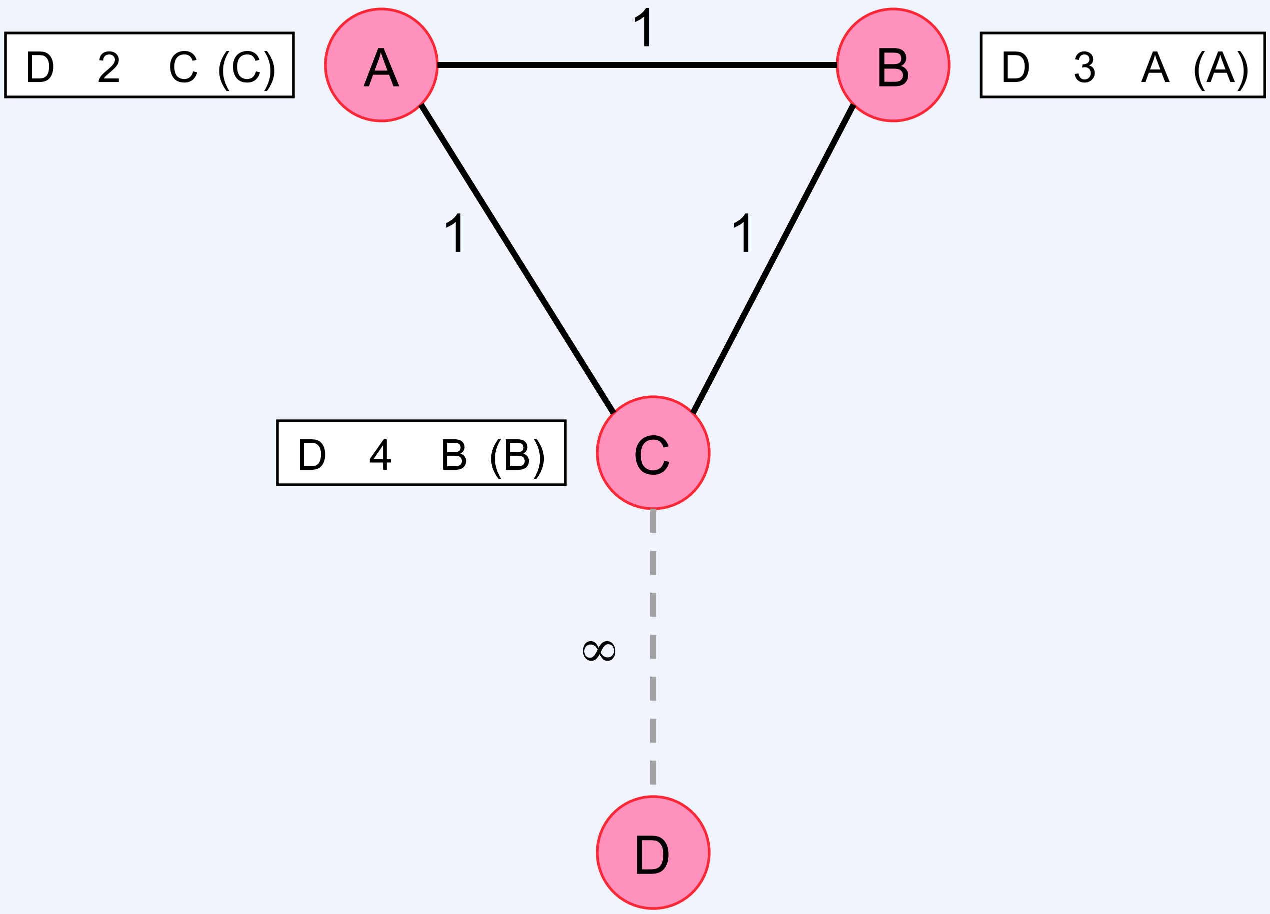
Split Horizon's Not Perfect (2)



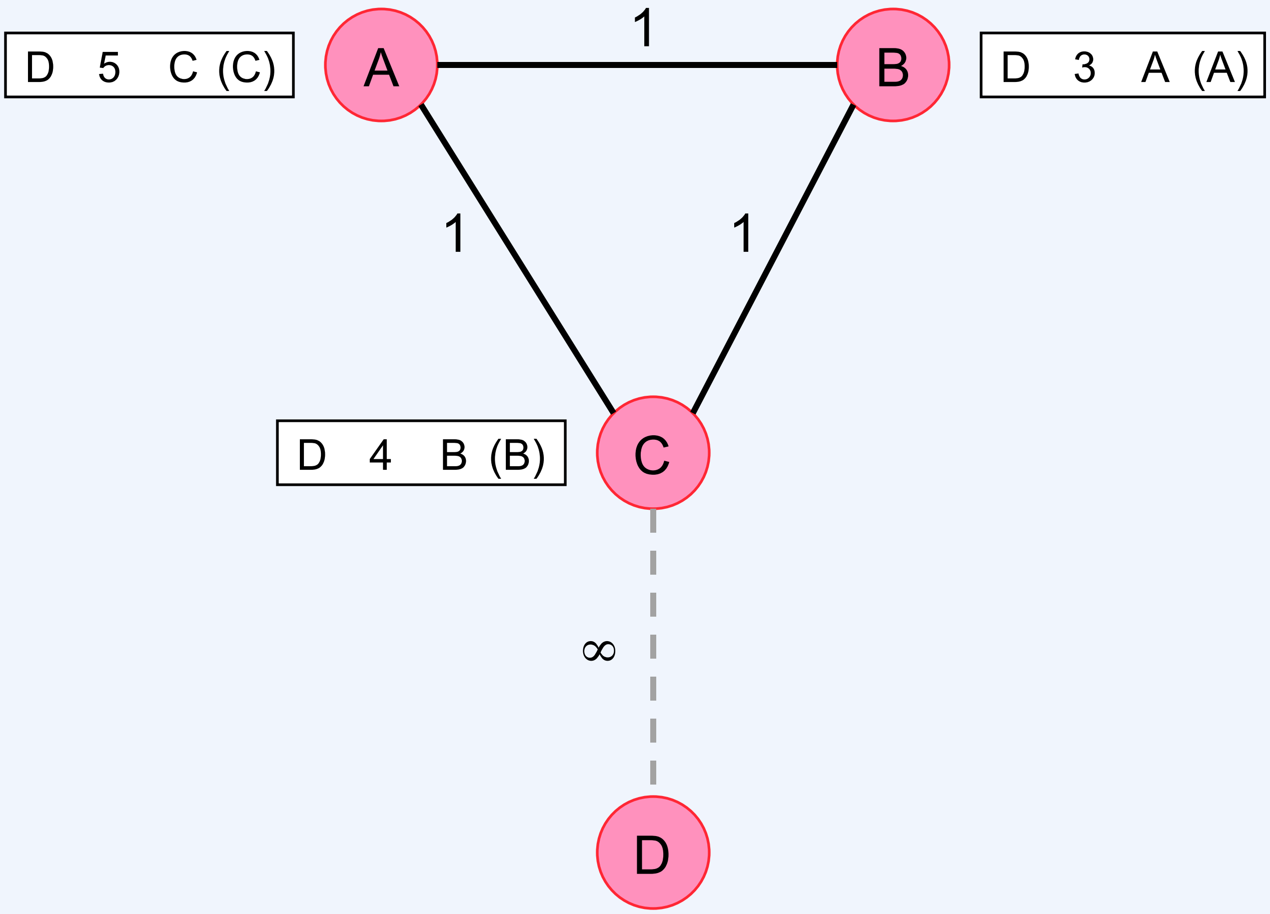
Split Horizon's Not Perfect (3)



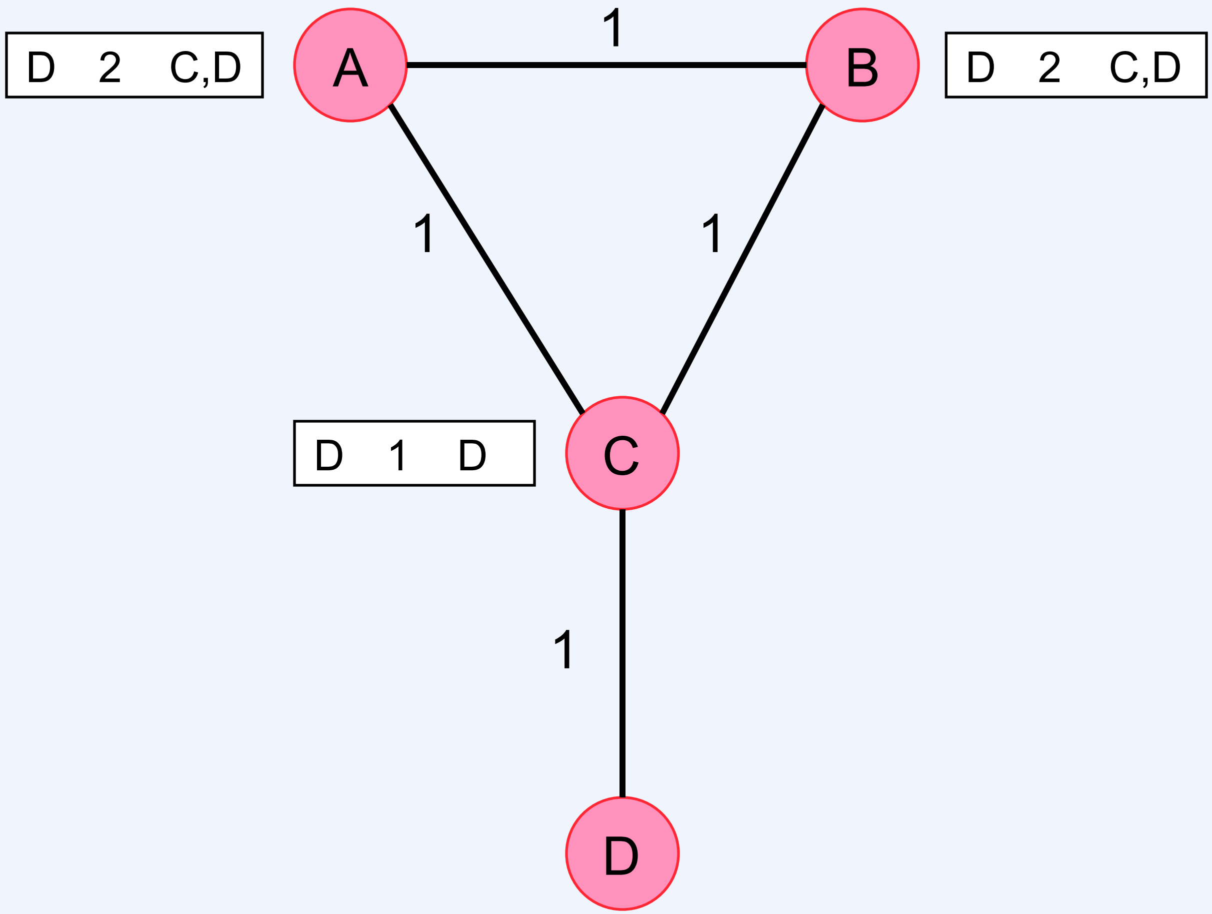
Split Horizon's Not Perfect (4)



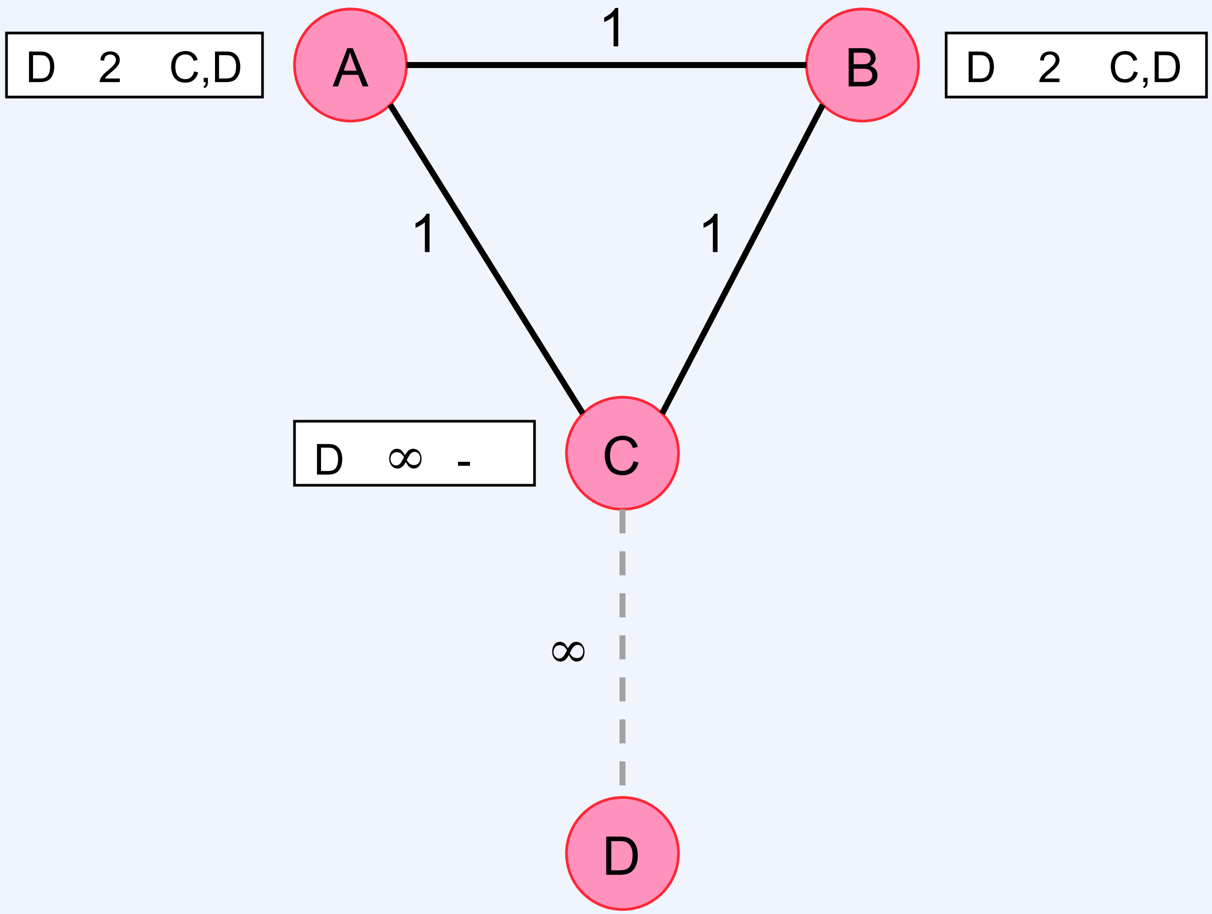
Split Horizon’s Not Perfect (5)



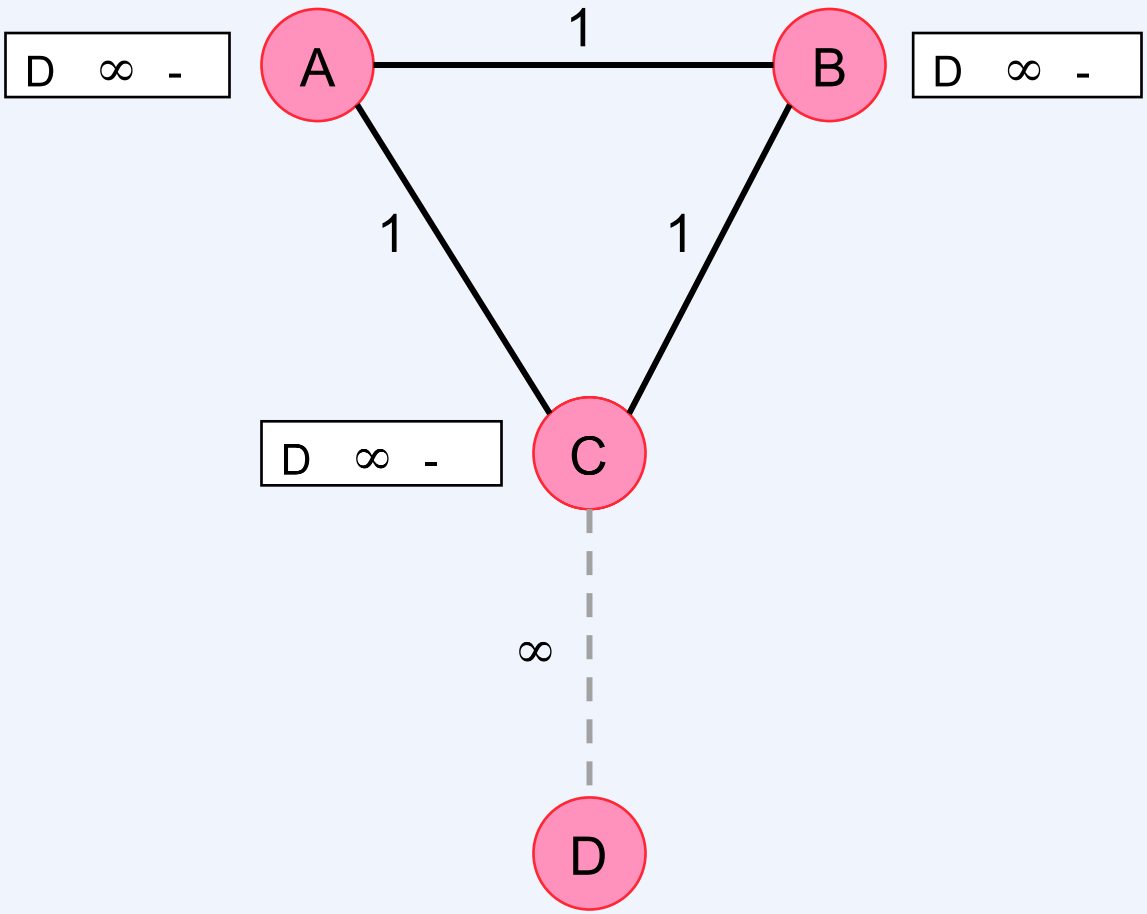
Path Vectors (1)



Path Vectors (2)



Path Vectors (3)



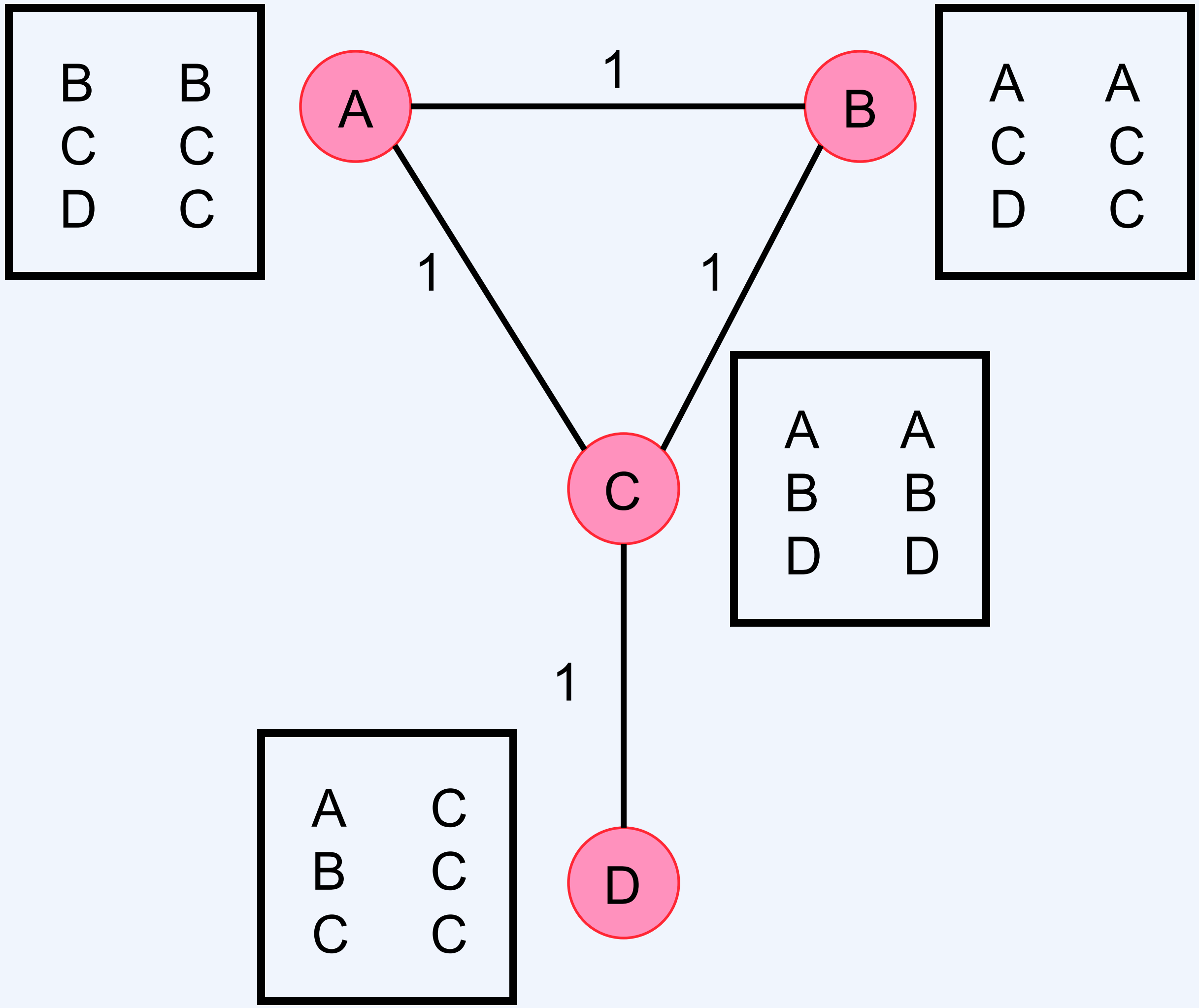
RIP Problems

- Slow convergence
- Infinity is too small
  - increasing infinity causes backwards compatibility problems and lengthens convergence
- Routing metric is number of hops
  - could easily be modified …
- Utilizes only a single path, even if multiple paths are possible

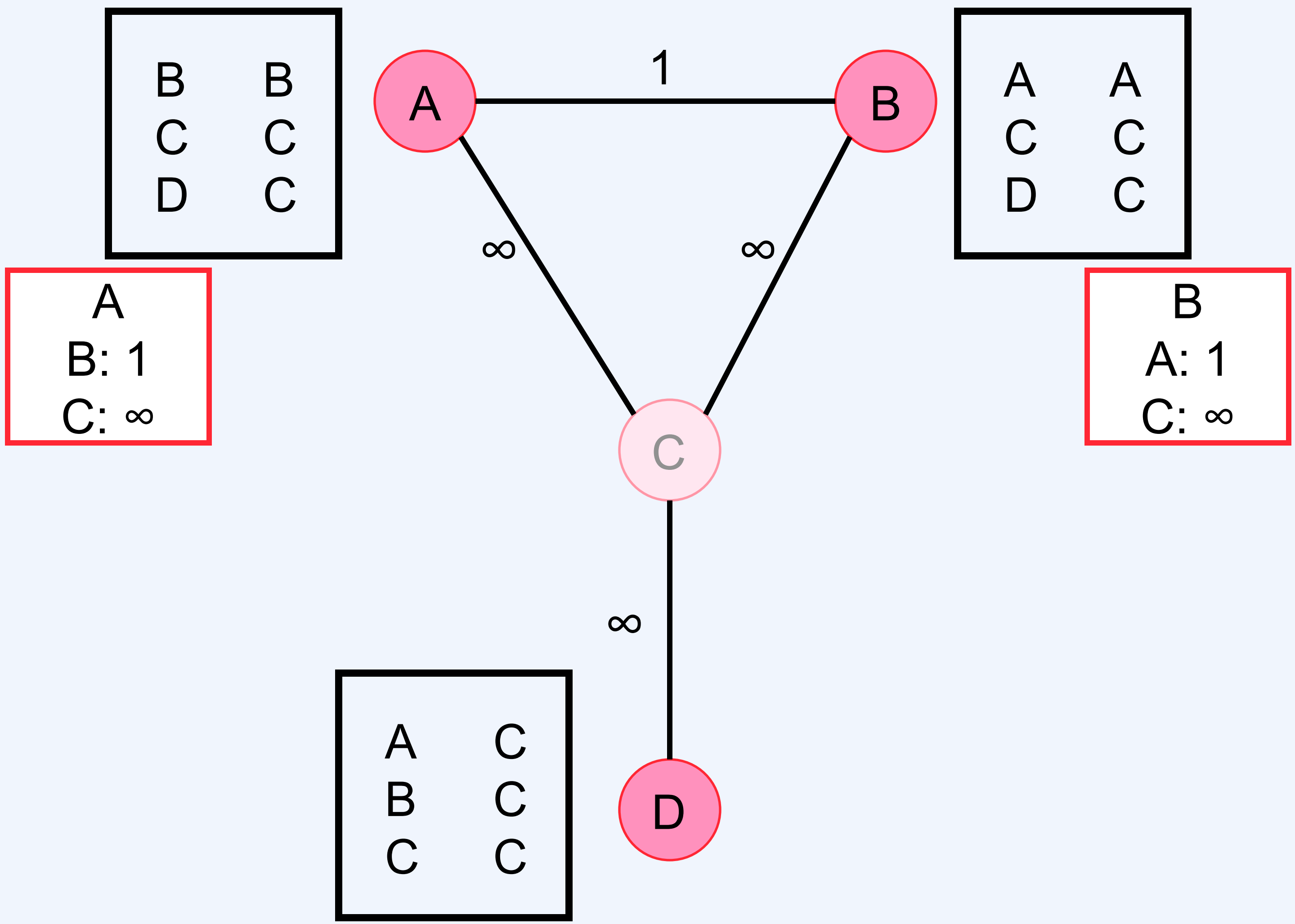
To the Rescue: Link-State Routing

- Routers flood internet with information about their direct links
- Each router computes shortest path independently, using global information

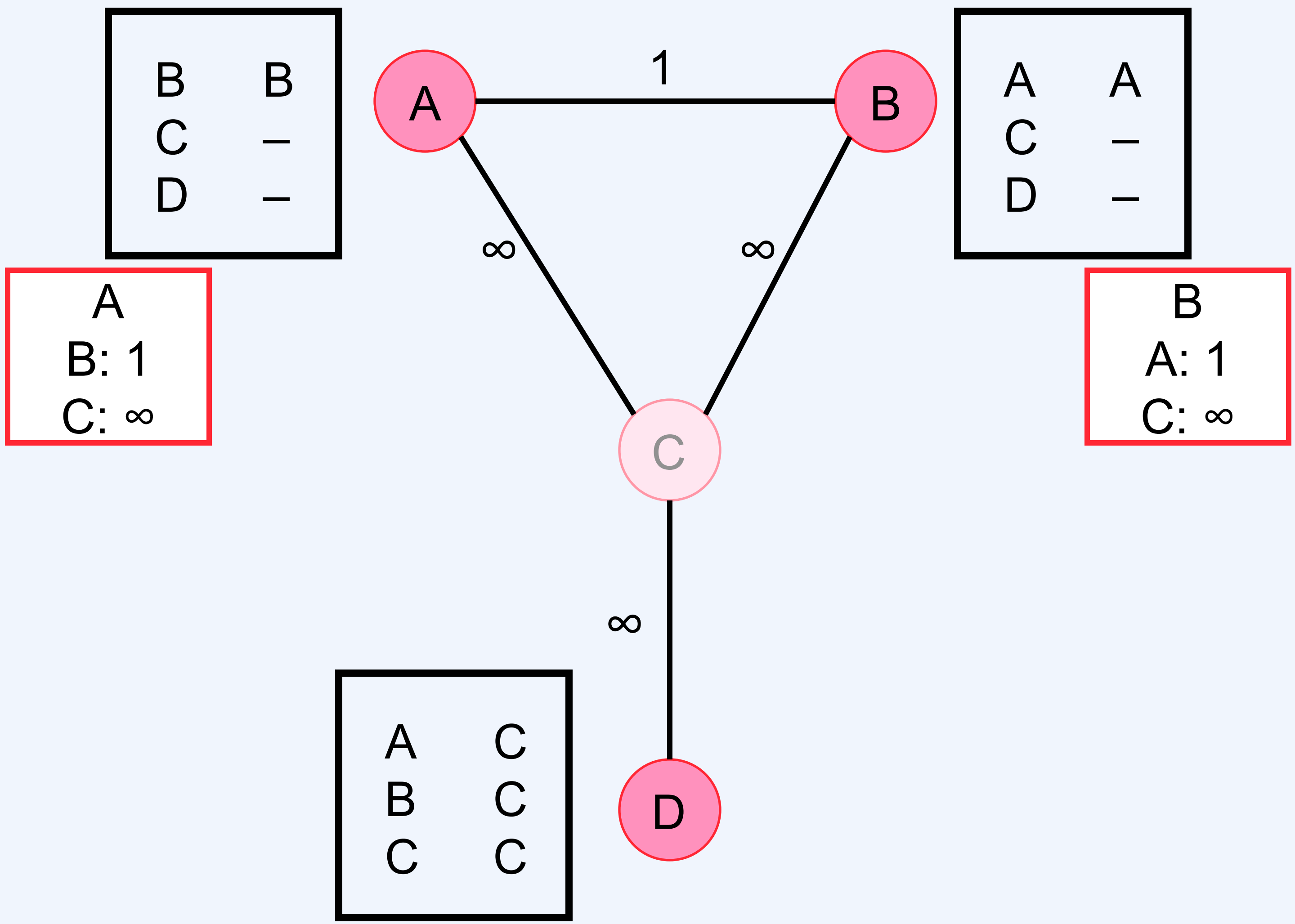
Quick Convergence



Quick Convergence (2)



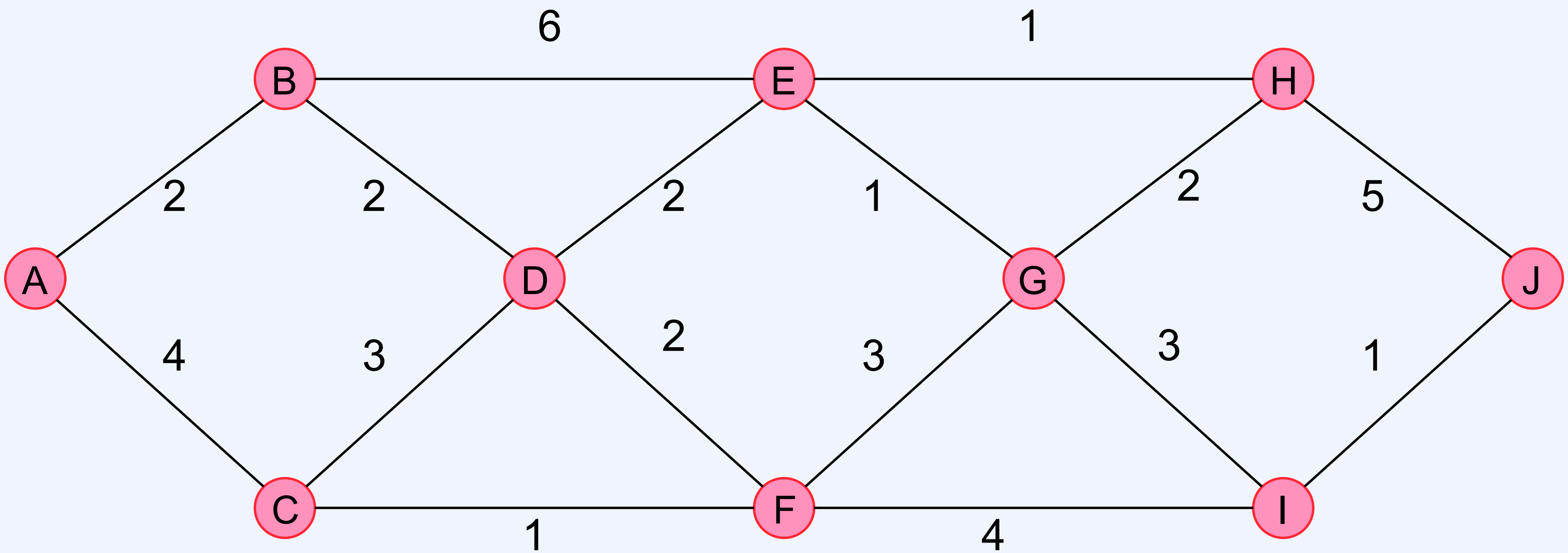
Quick Convergence (3)



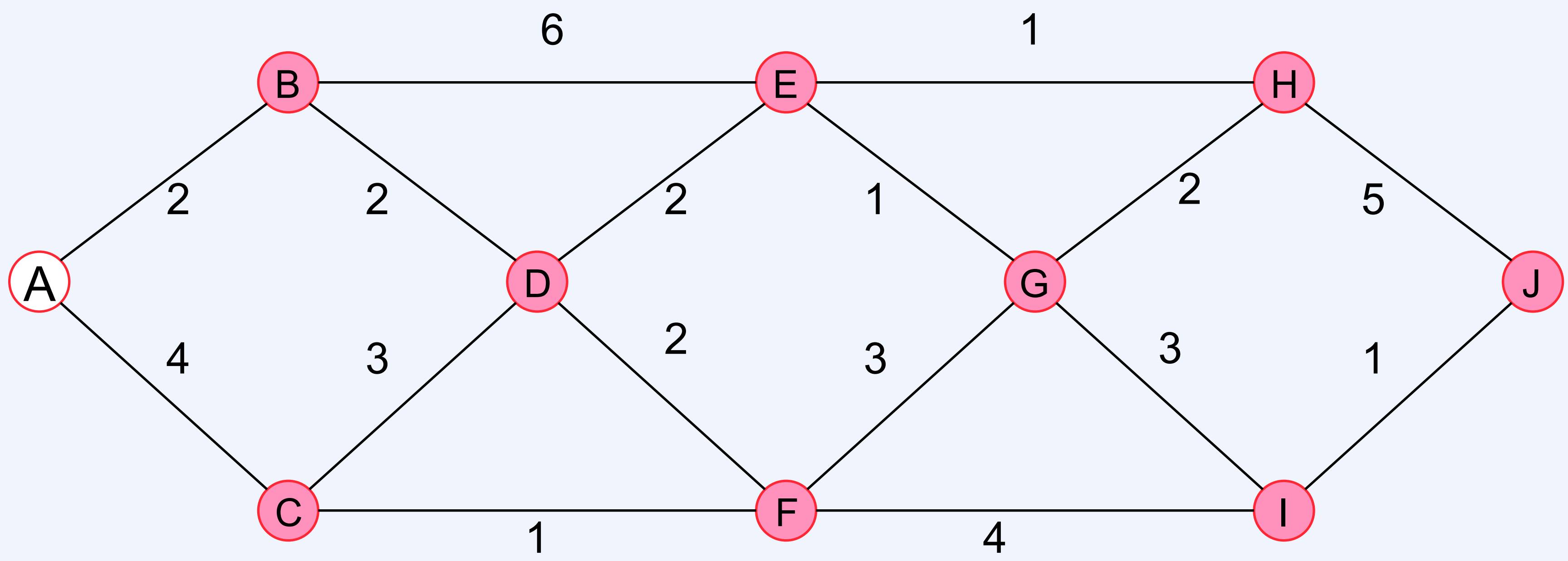
Link-State Advantages

- Quicker convergence
- Load sharing
- Better scaling

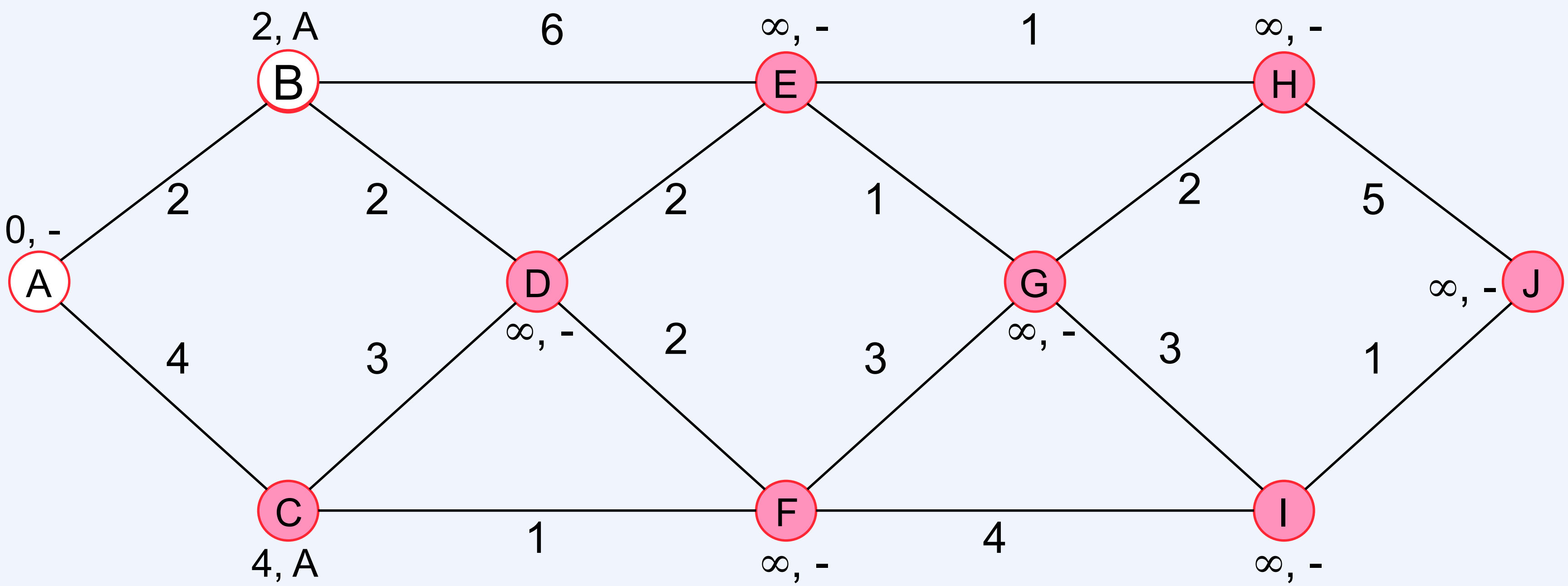
Finding the Shortest Path



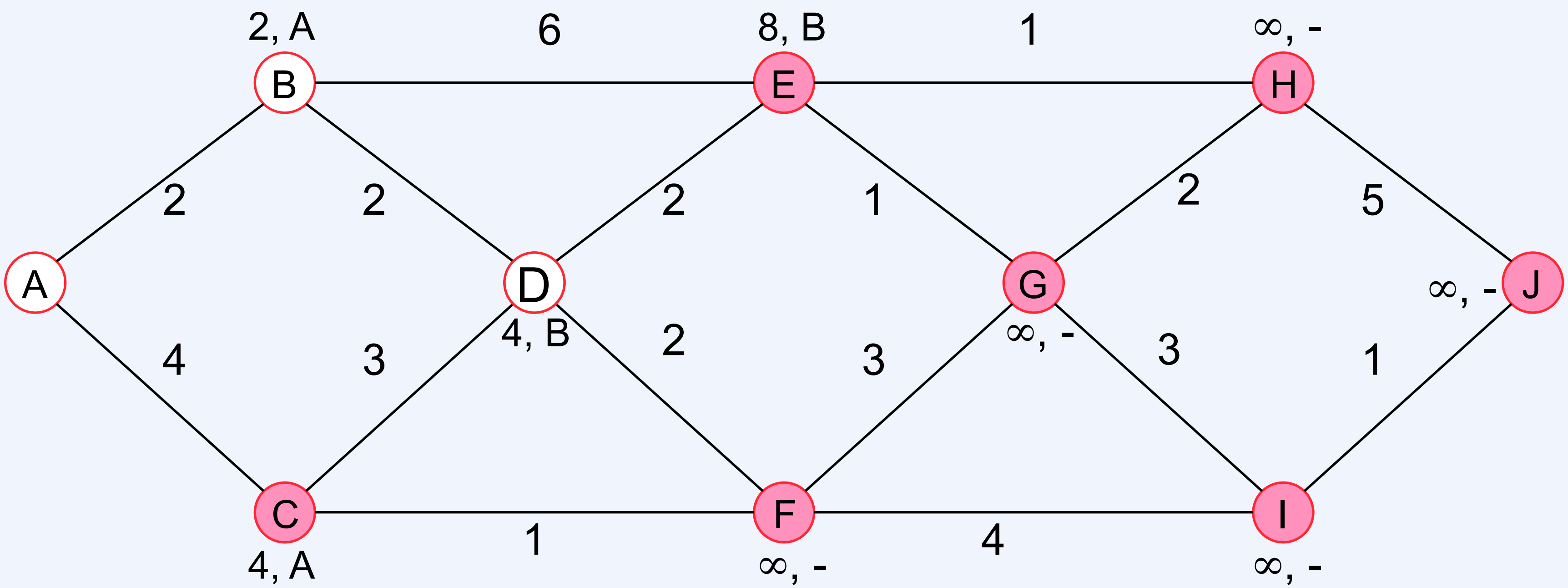
Shortest Path (1)



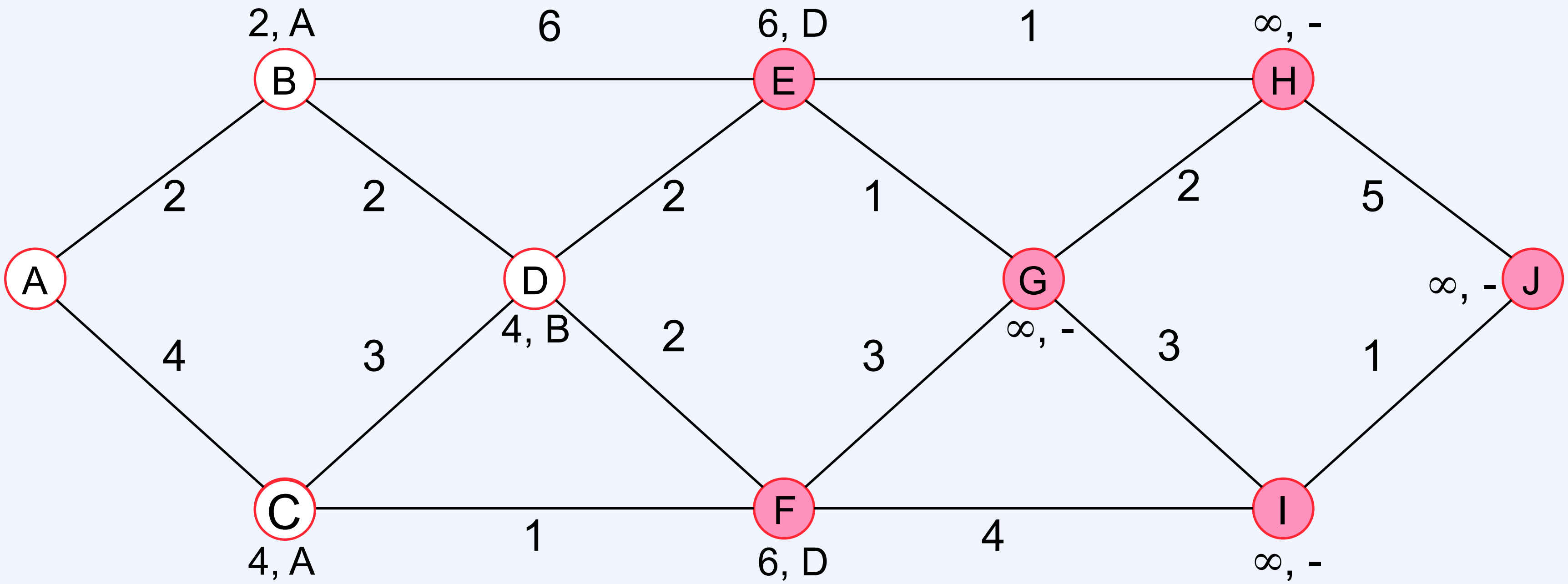
Shortest Path (2)



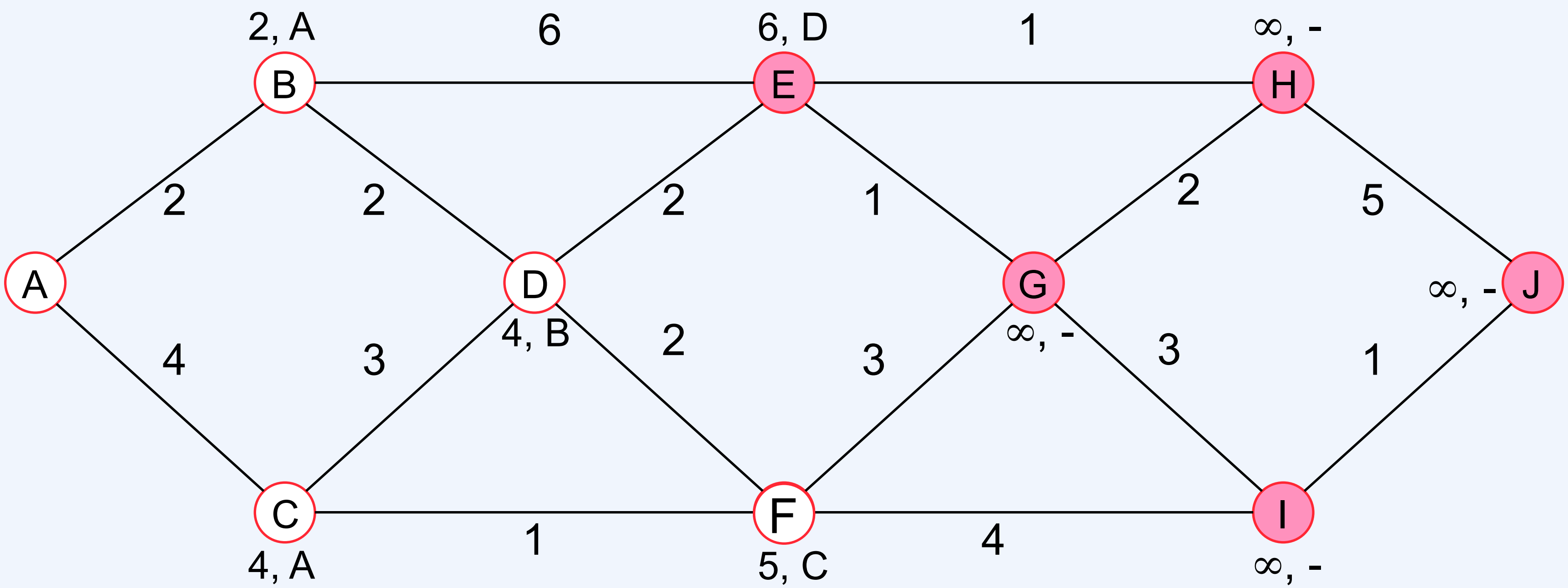
Shortest Path (3)



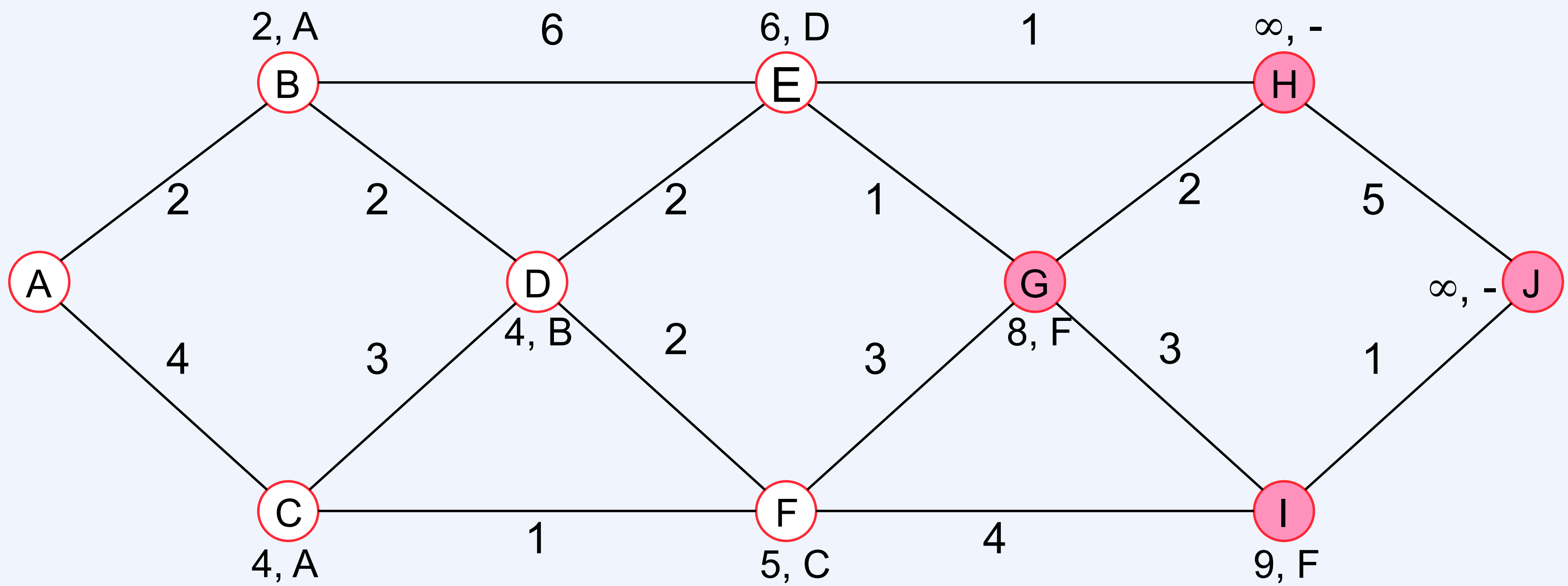
Shortest Path (4)



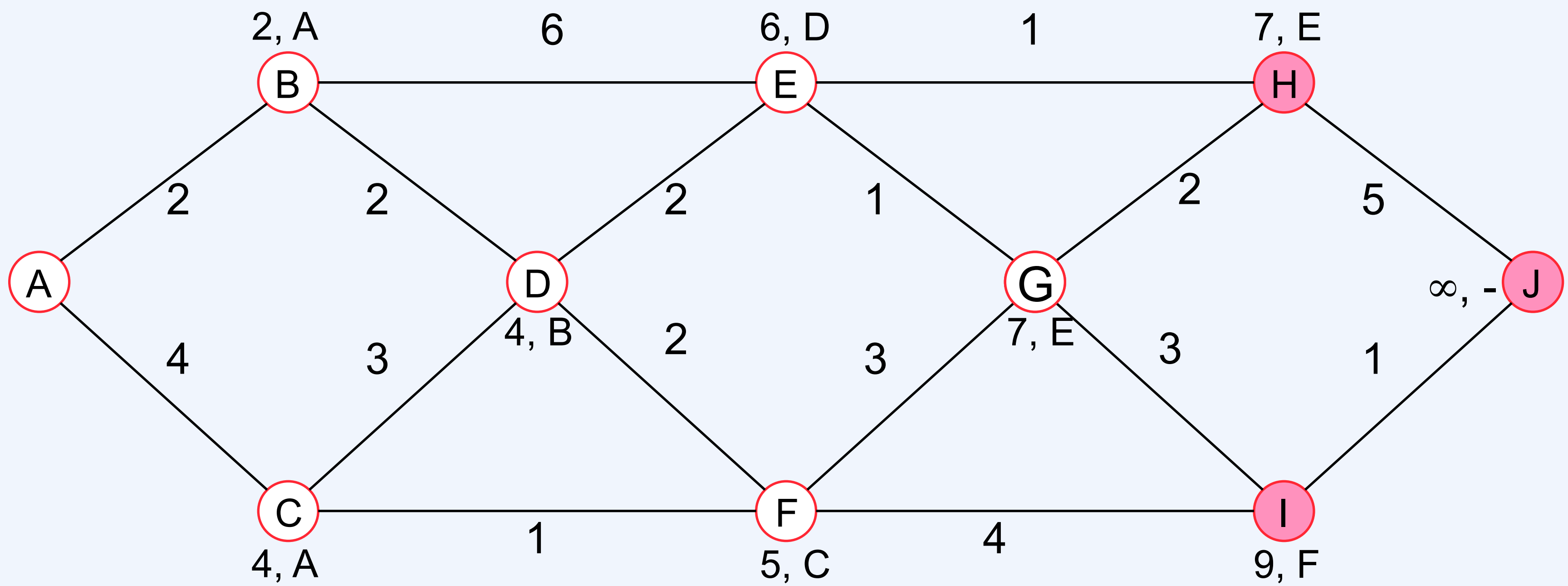
Shortest Path (5)



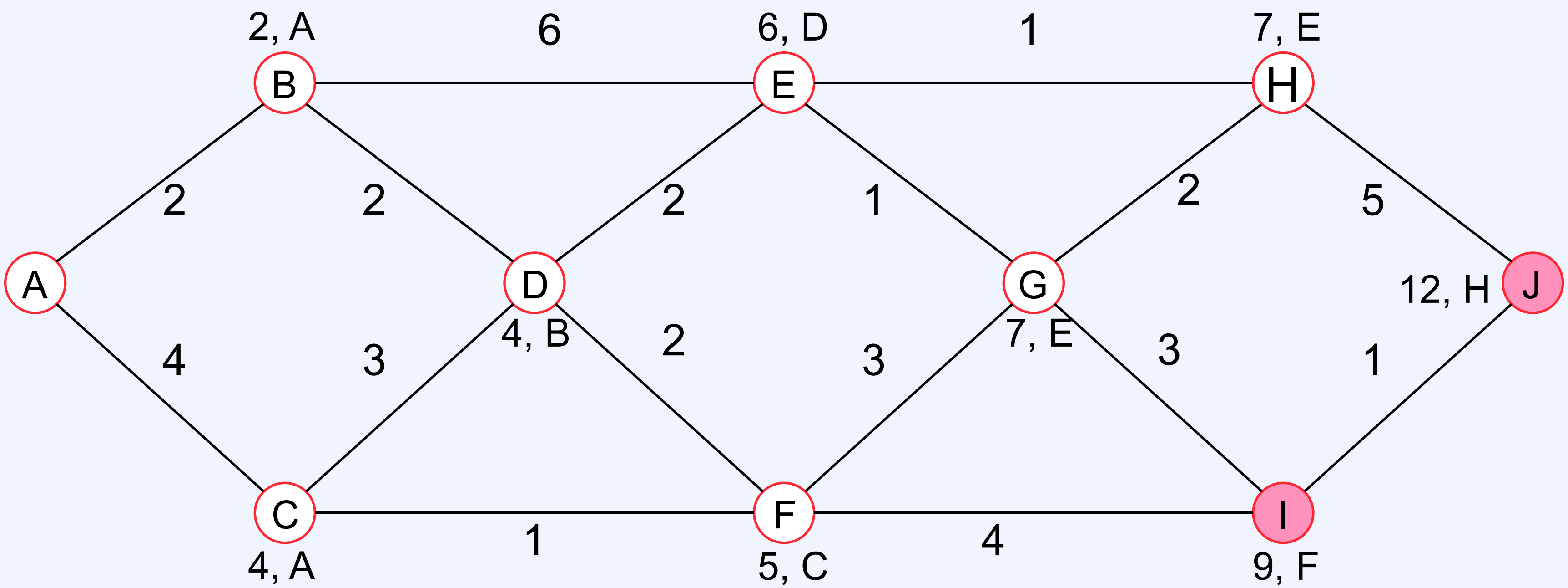
Shortest Path (6)



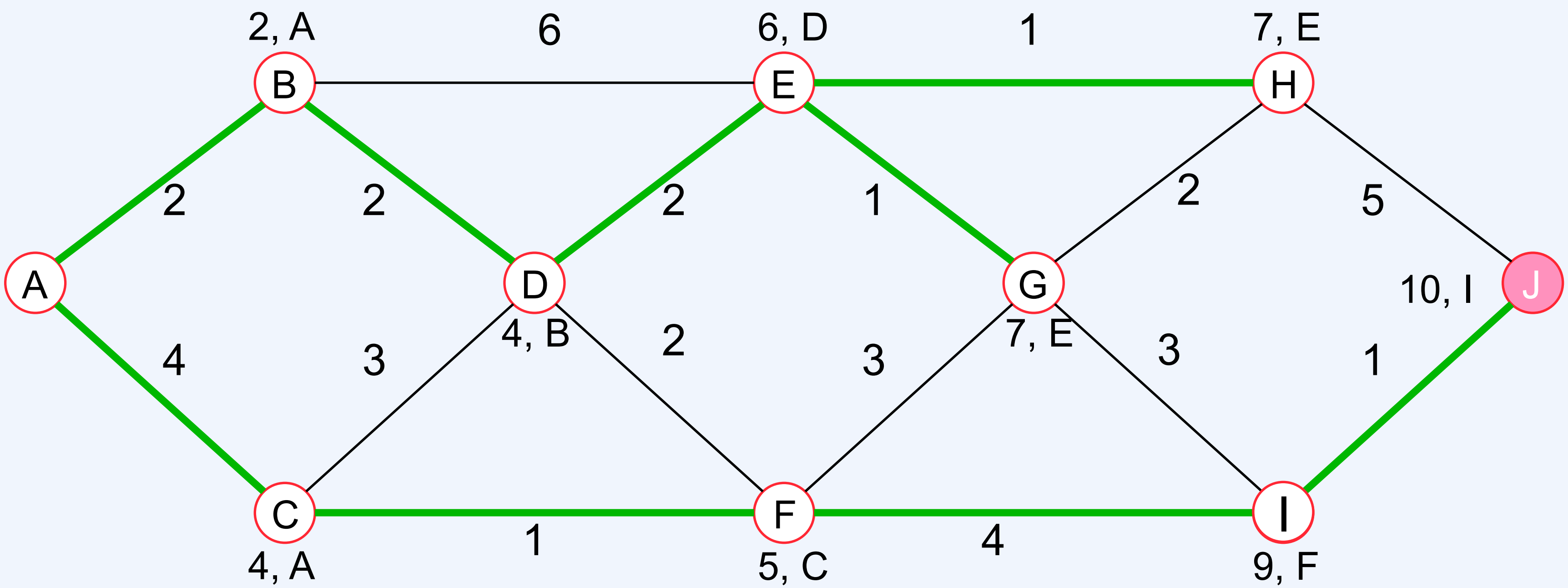
Shortest Path (7)



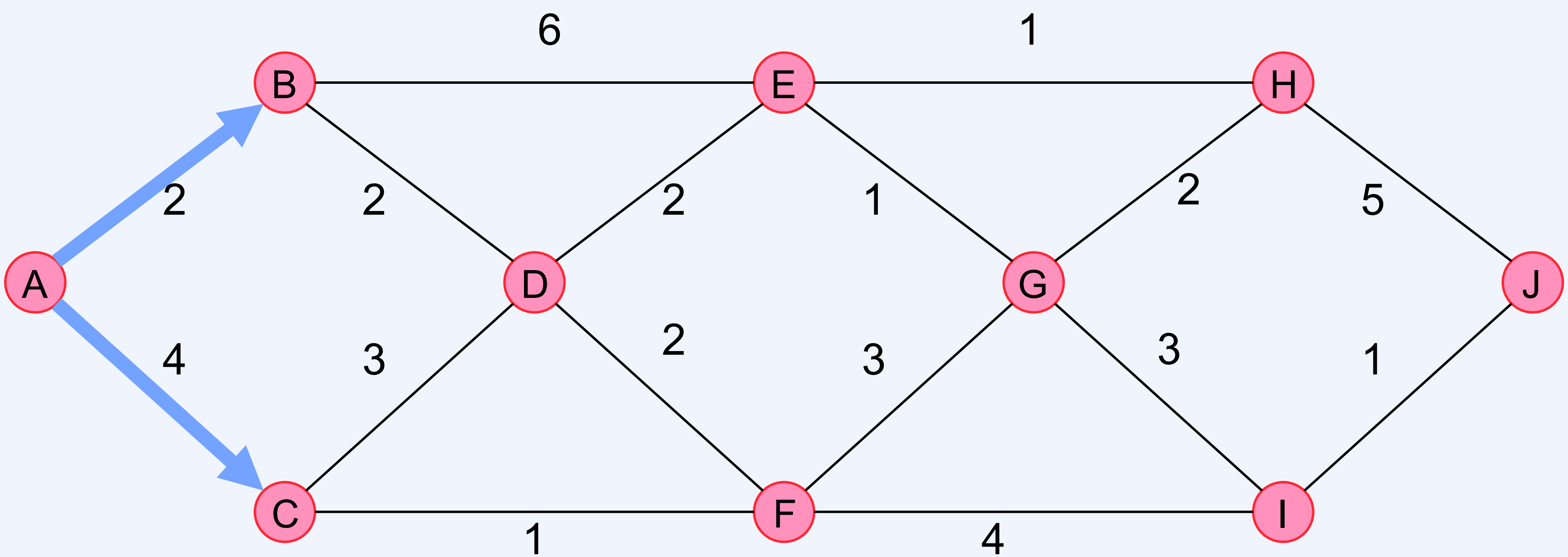
Shortest Path (8)



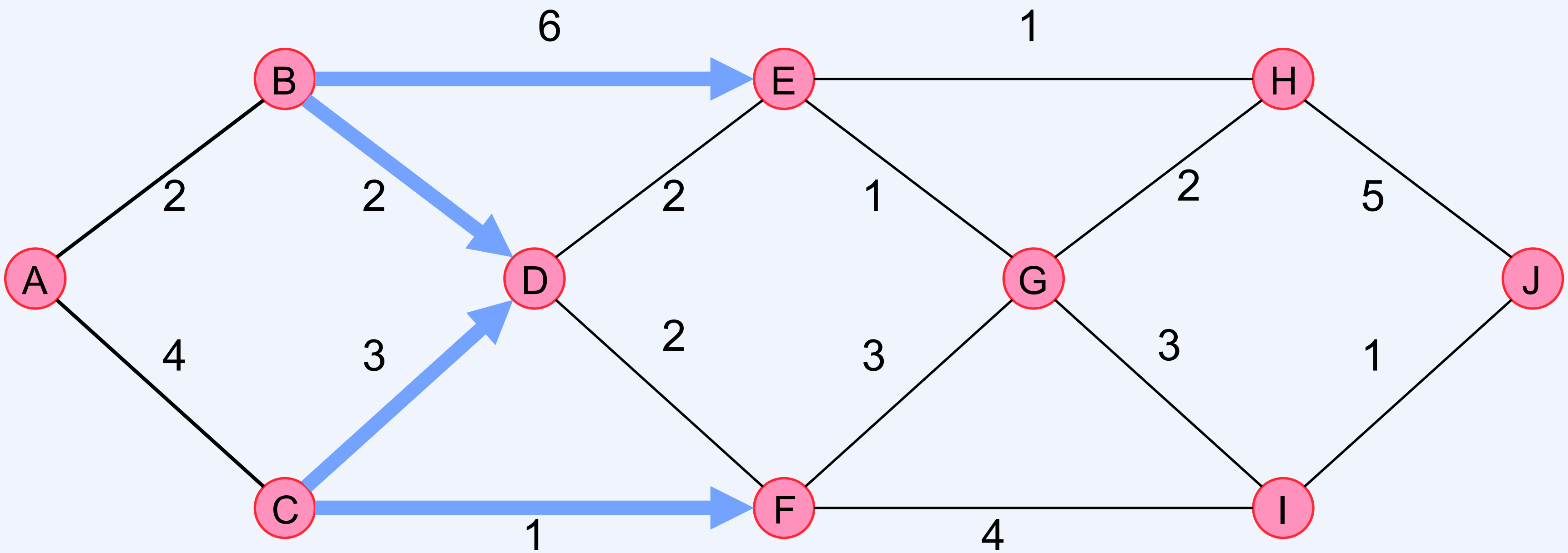
Shortest Path (9)



Flooding (1)



Flooding (2)



Flooding Problems

- Reliability
  - new information must replace old
- Robustness
  - misbehaving routers don't bring down everyone

Reliability

- New information must replace old
  – use sequence numbers and acks
  – but …
    - sequence-number wraparound
    - restarting a router
      - state info is lost

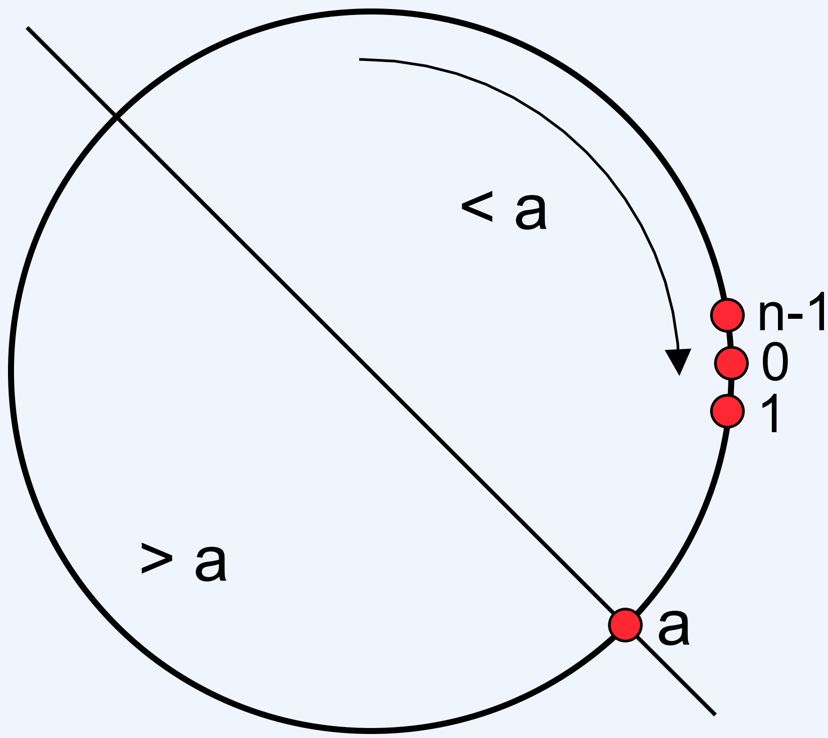
General Link-State Flooding Strategy

- Each router:
  - periodically and whenever there's a change:
    - flood internet with new link-state information
  - when receive link-state update:
    - if update is new, copy it into database
      - recompute routes

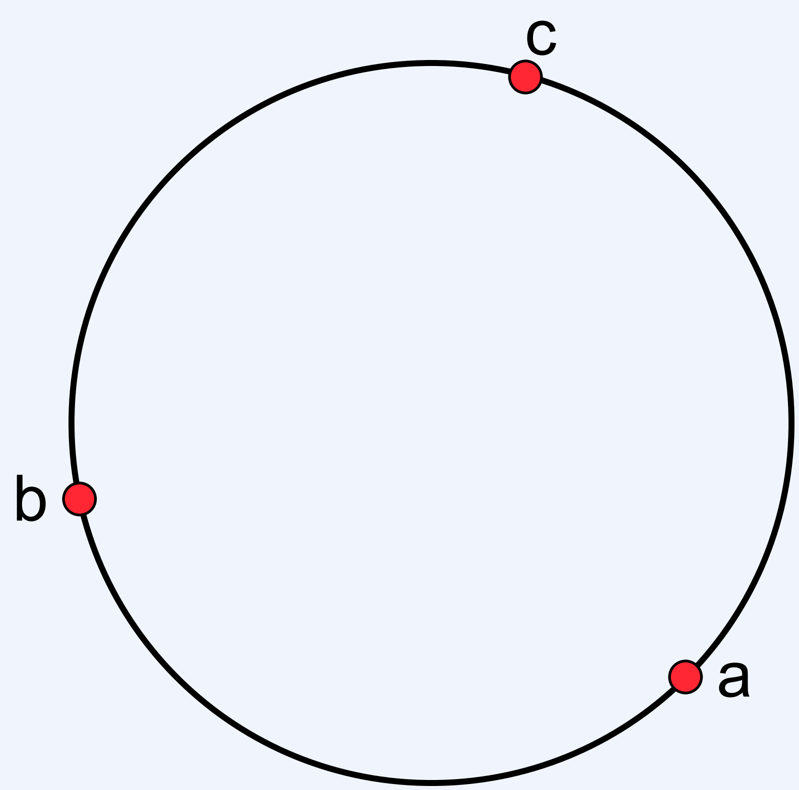
Approach 1

- Sequence numbers
  – each router assigns its next highest sequence number to its link-state updates (LSUs)
  – when receiving an LSU, ack it, but ignore its contents unless it has a higher sequence number than what's already received (in which case, forward it on)
- Restart
  – each router gives its LSUs an "age"
    - each router decrements the age of its copy of an LSU until it reaches zero or is replaced
  – if age is zero, then new LSU, regardless of sequence number, replaces old

Sequence-Number Wraparound



Whoops …



a < b < c < a

Solution

- **Sequence numbers not allowed to easily wraparound:**
  - **when max value is reached, no wraparound allowed until LSU "ages out"**
  - **thus after receiving LSUs with sequence numbers *a*, *b*, and *c*, by the time sequence numbers are allowed to wrap, a repeat occurrence of *a* will have "aged out"**
- **Sequence-number space made large enough so that wraparound occurs only on misbehaving routers**

Open Shortest Path First (OSPF)

- Link-state protocol
- Predecessors developed starting in late ’70s
- Implemented on Unix as *gated* (which also does RIP)
- In use since 1990
- Described in RFC 2328

The OSPF Approach

- All communication layered directly on top of IP
  - OSPF is a transport-layer protocol
- "Discover" neighbors
  - "hello" messages sent periodically on point-to-point links, multicast on LANs
- Measure or obtain cost of transmission to each neighbor
- Transmit routing information to all other routers
  - use reliable "flooding" algorithm
- Compute shortest path based upon current link-state information (use Dijkstra's algorithm)
  - modify Dijkstra's algorithm to obtain all shortest paths