Georgia Tech Computer Science

CS2200
Systems and Networks
Spring 2024

Lecture 26: Networking

Alexandros (Alex) Daglis
School of Computer Science
Georgia Institute of Technology
adaglis@gatech.edu

Lecture slides adapted from Bill Leahy and Charles Lively of Georgia Tech

Networking

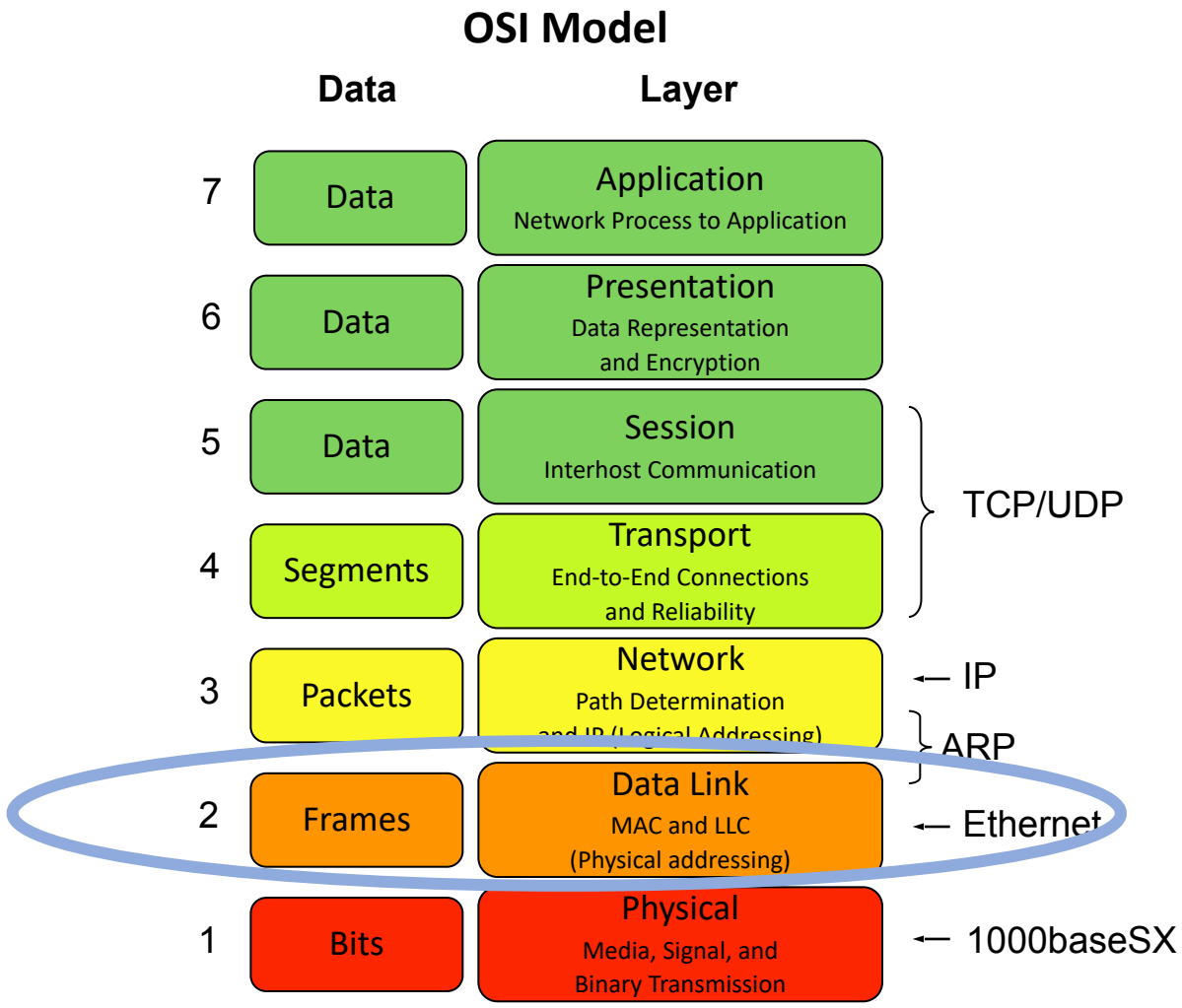
- Network stack overview – layer encapsulation
- Three transport protocol types
- Basic distinctions between TCP and UDP
- Network layer
- Ports and connections

Today:

- Data Link layer (Ethernet)
- Performance metrics
- Routing and IP table construction

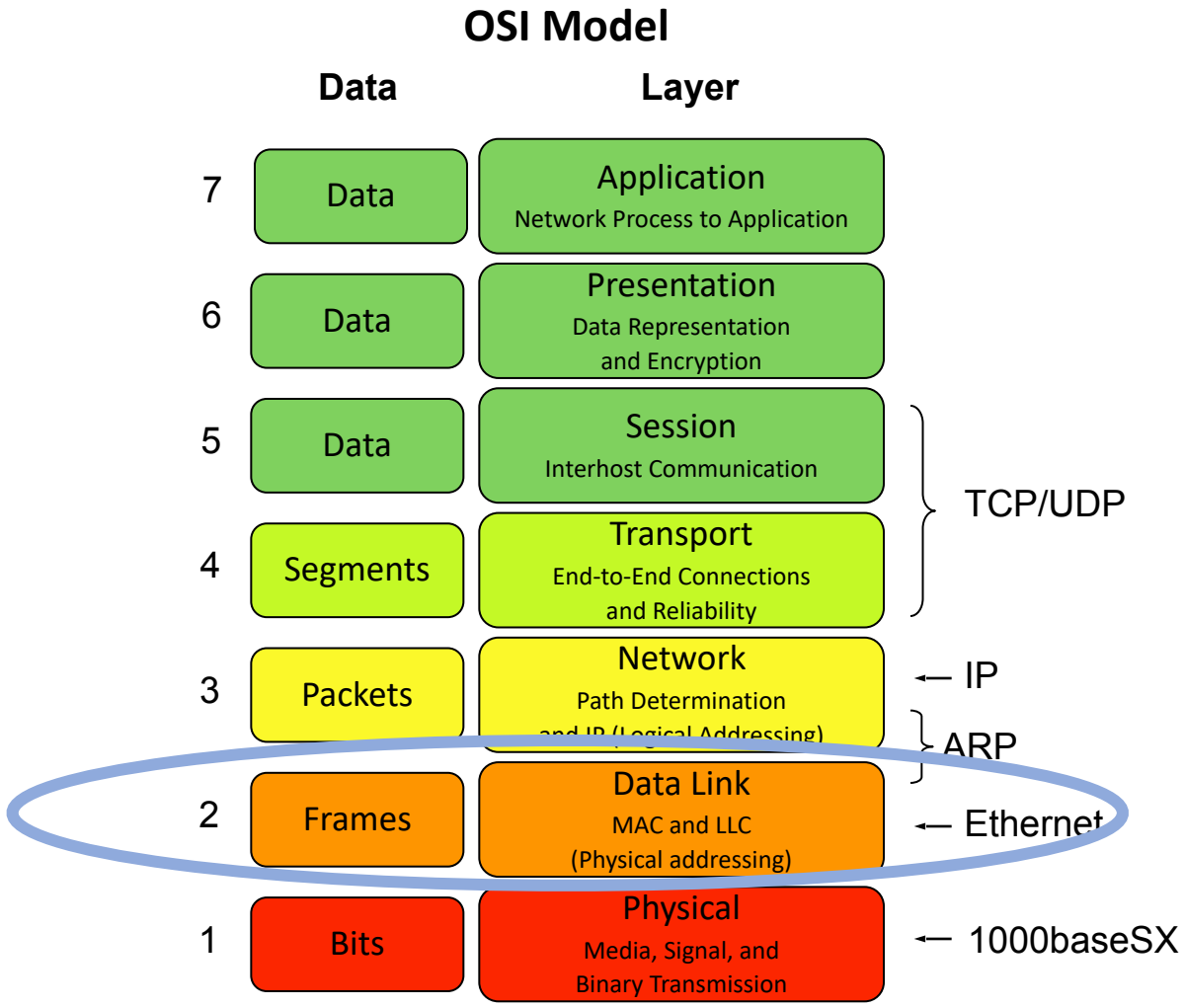
Last 3 lectures: Disk scheduling & filesystem

Now let's look at the data link layer



Now let's look at the data link layer

- Ethernet is a very common data link network



Let's talk about Ethernet

- Ethernet is a particular type of data link network
- It was designed for relatively small areas, like a building, hence the term Local Area Network (LAN)
- It has its very own standard (IEEE 802.3)
- It's evolved a great deal since the 1980s

Terminology

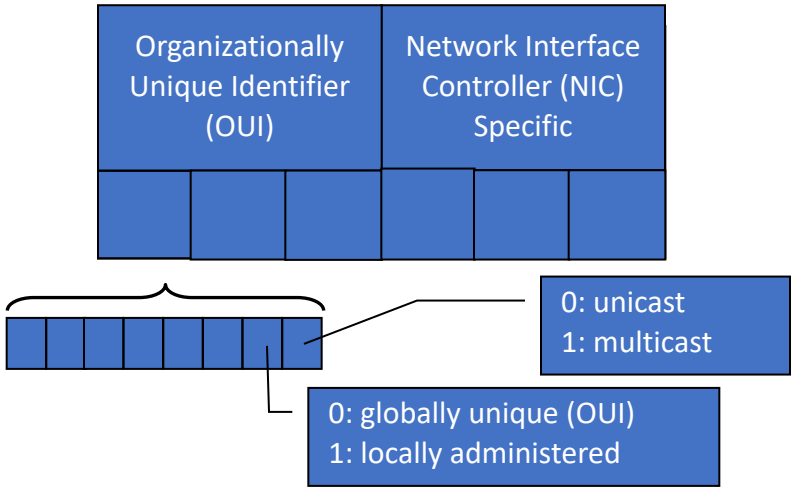
- Collision domain - All the NICs that share a physical ethernet; only one can transmit at a time
- Broadcast domain - All the NICs that can hear an ethernet broadcast frame

---

- Repeater/hub - layer 1 replication/amplification of bits (both sides stay in one collision domain)
- Bridge/switch - layer 2 re-transmission of ethernet frames (both sides stay in one broadcast domain)
- Router - layer 3 forwarding of packets based on a layer-3 (IP) routing table
- Host – a computer with one or more NICs; if it forwards traffic using multiple NICs, we're going to call it a router, not a host.

Ethernet Addressing

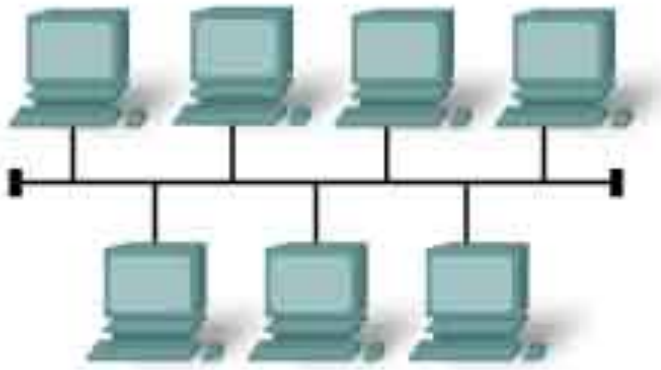
Every ethernet device gets a unique burned-in 48-bit MAC address



- 6 bytes/48 bits
  - 3 bytes OUI
  - 3 bytes NIC specific
- Examples
  - Apple: 00:25:00:f4:6d:c2
  - Oracle: 00-03-ba-a1-02-df
- To identify the OUI
  - https://www.wireshark.org/tools/oui-lookup.html

Ethernet Frames

Preamble	Destination MAC	Source MAC	802.1Q tag	Ether type/ Length	Payload	CRC	Interframe Gap
8	6	6	0-4	2	46-1500	4	12

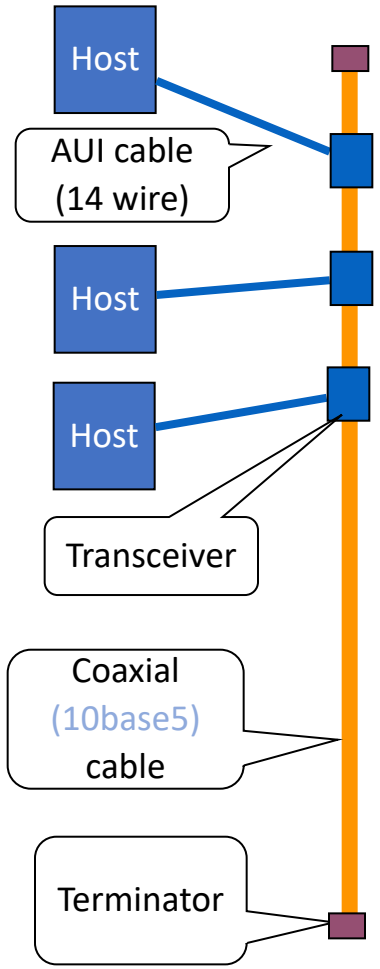


- To send a frame
  - Wait until no one using the cable (carrier sense)
  - Assert carrier signal on the cable
  - Broadcast the frame to all stations on the cable
  - If you detect a **collision** while transmitting, stop and requeue the frame for retransmission (known as CSMA/CD)
- If you hear a frame on the cable
  - Accept it if the destination MAC belongs to us (or if it has a broadcast or multicast destination MAC)
  - Drop it if you see a collision during its receipt or its CRC doesn't match
- Ethernet works like a logical bus!

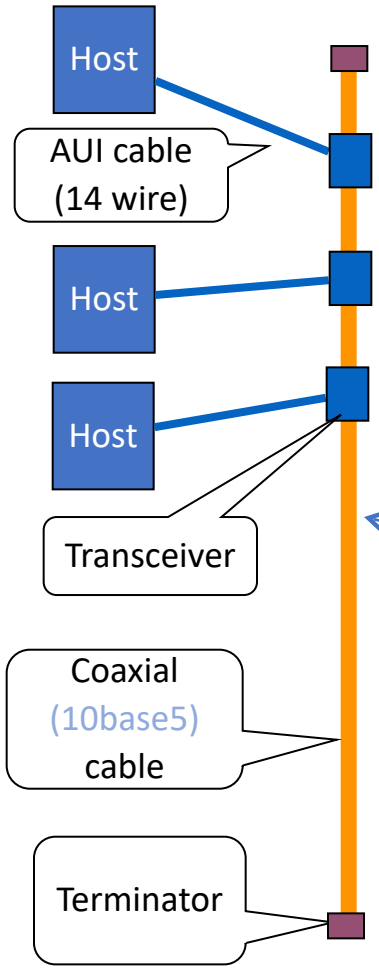
Something is Missing…

- Only one station can send at a time?
- Where are the IP addresses?
- We'll get back to this…

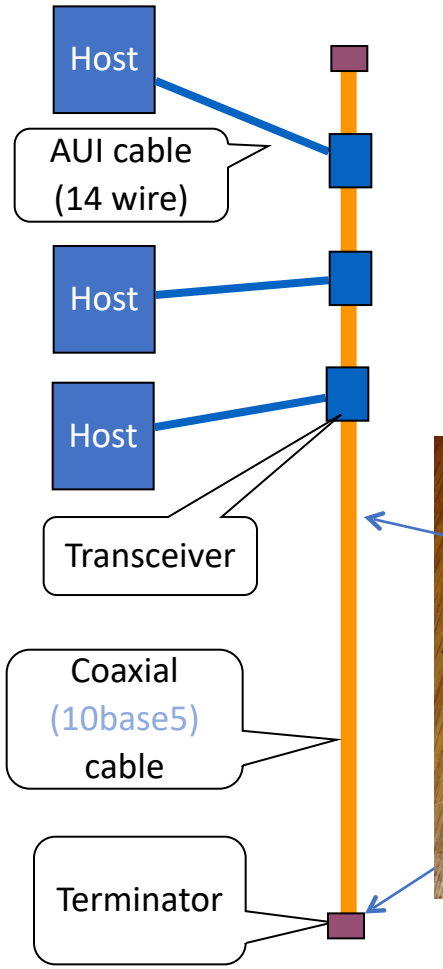
Ethernet Topology (some history)

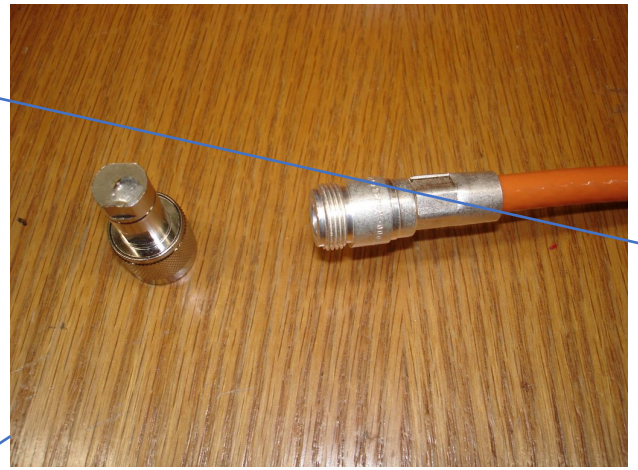


Ethernet Topology (some history)

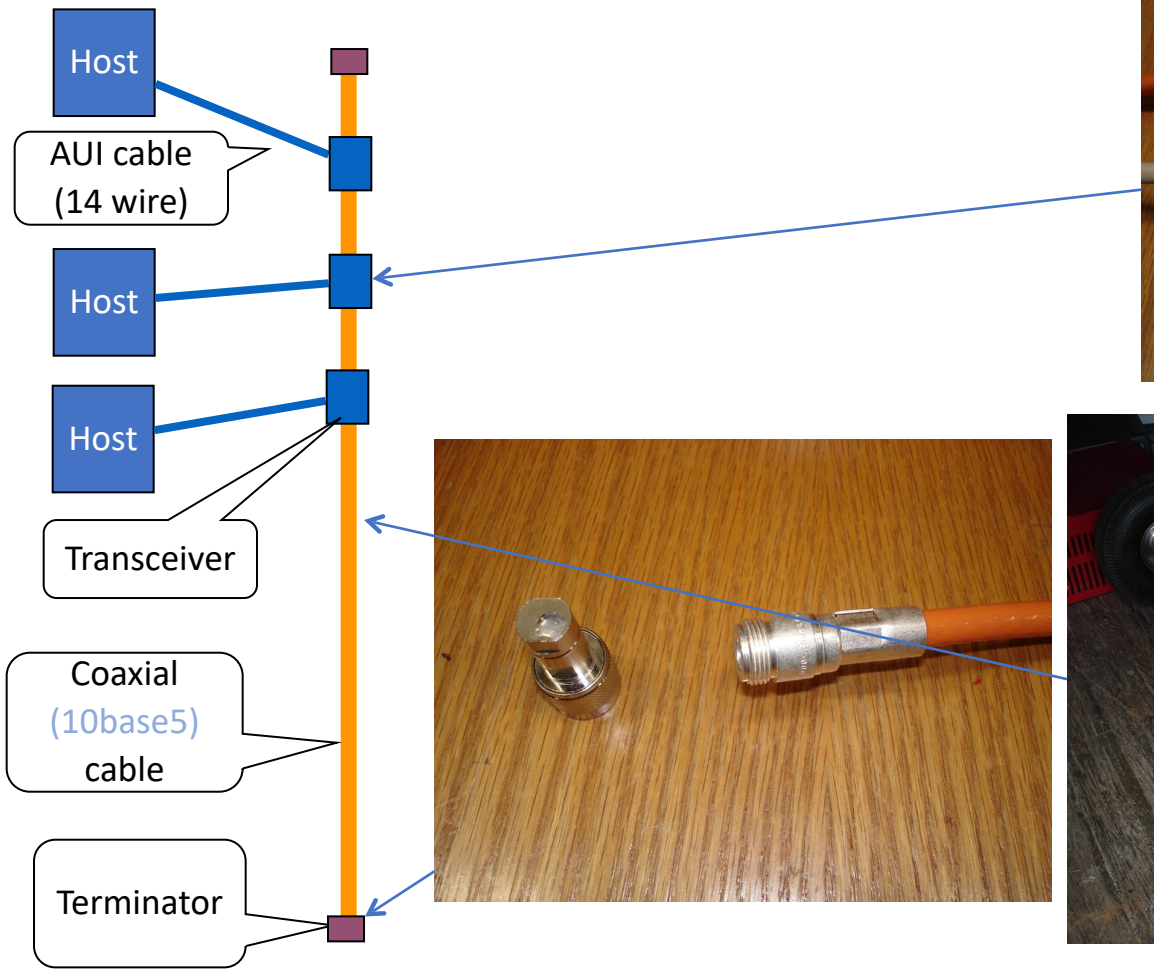


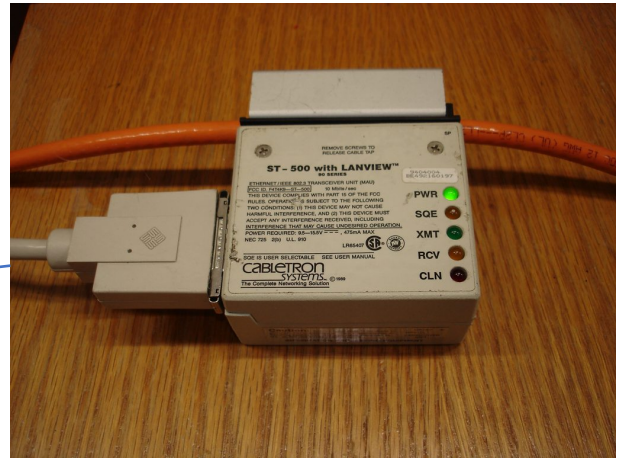
Ethernet Topology (some history)



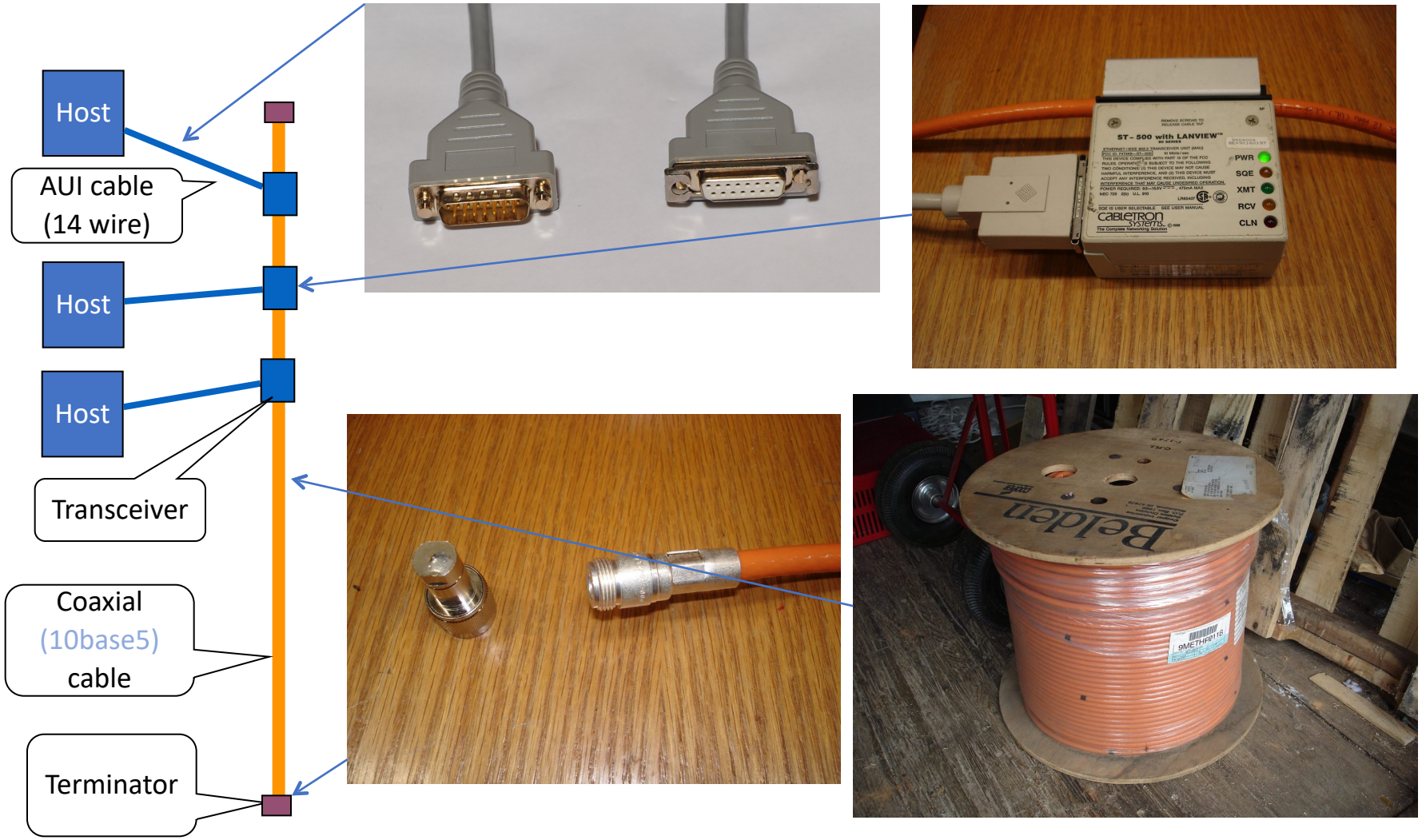


Ethernet Topology (some history)



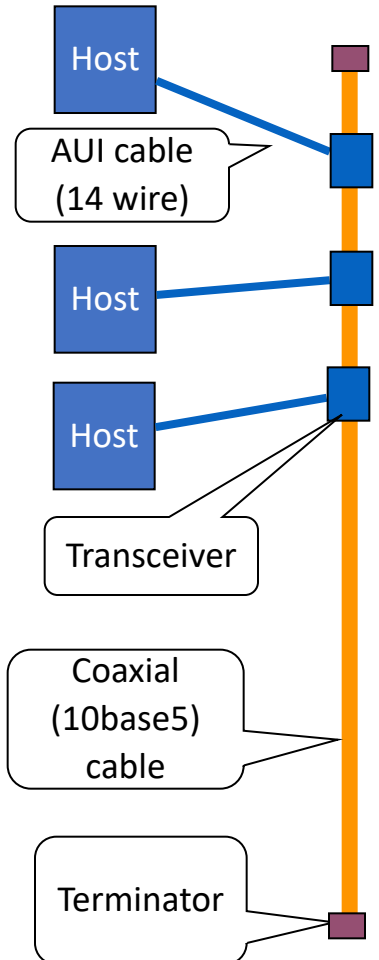


Ethernet Topology (some history)

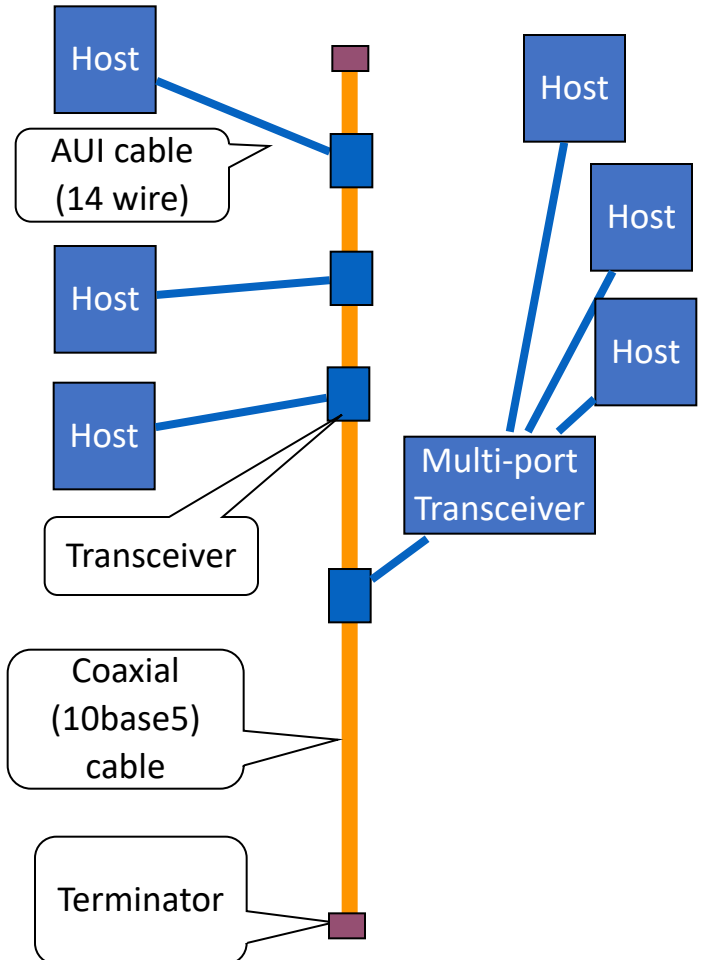


Ethernet Topology (some history)

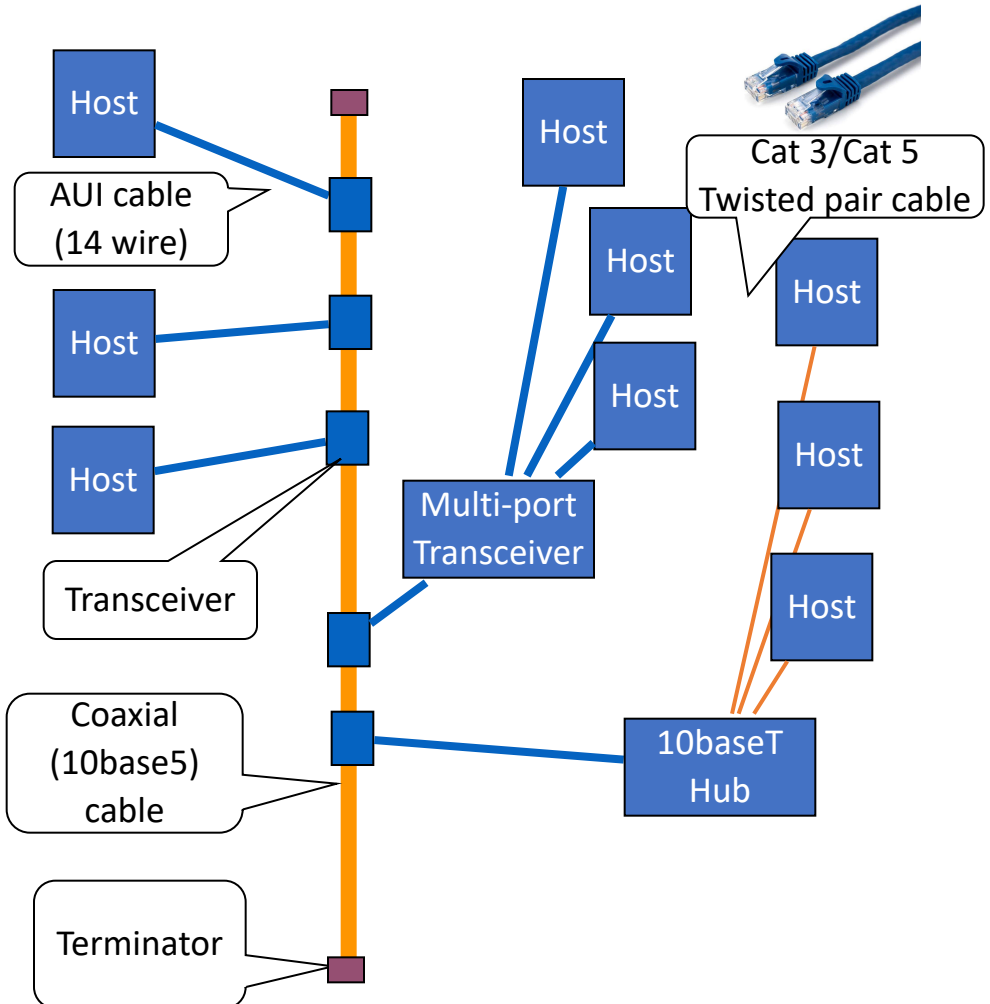
Ethernet Topology (some history)



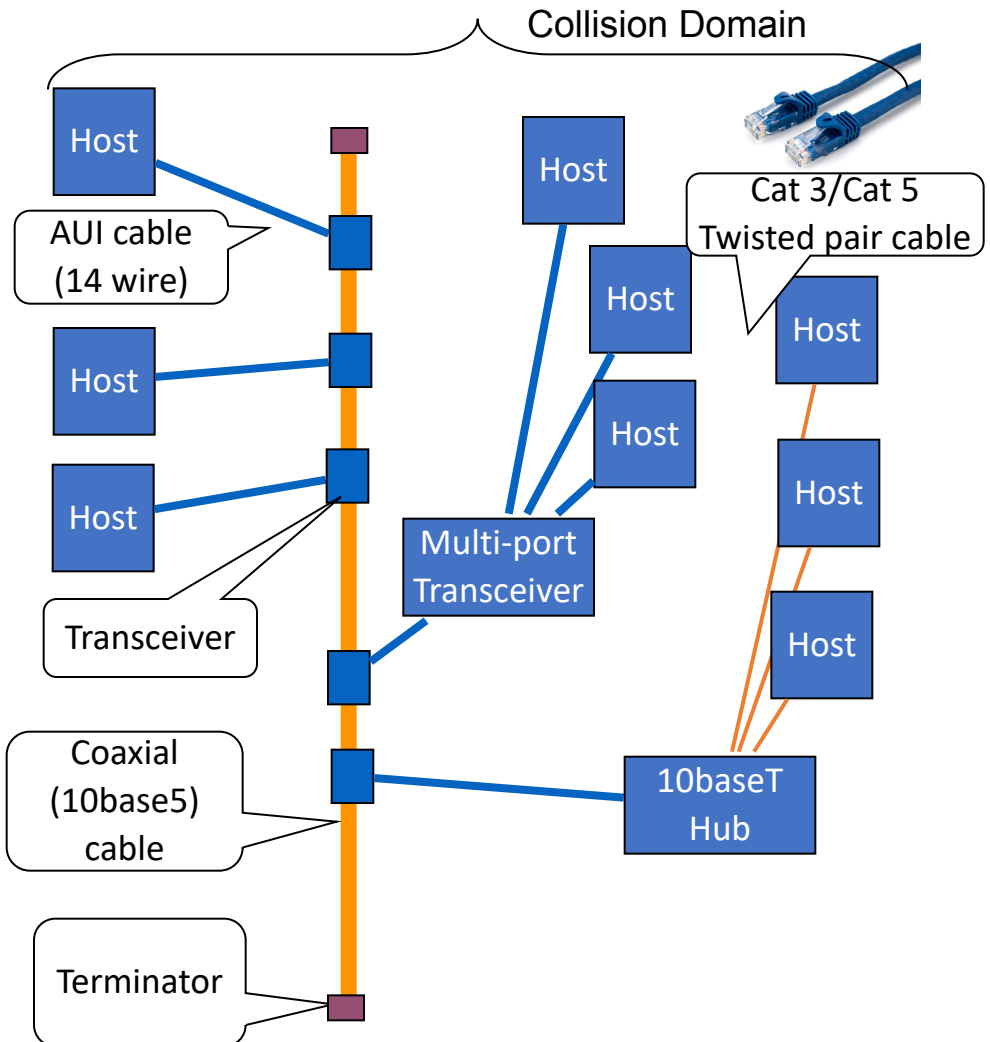
Ethernet Topology (some history)



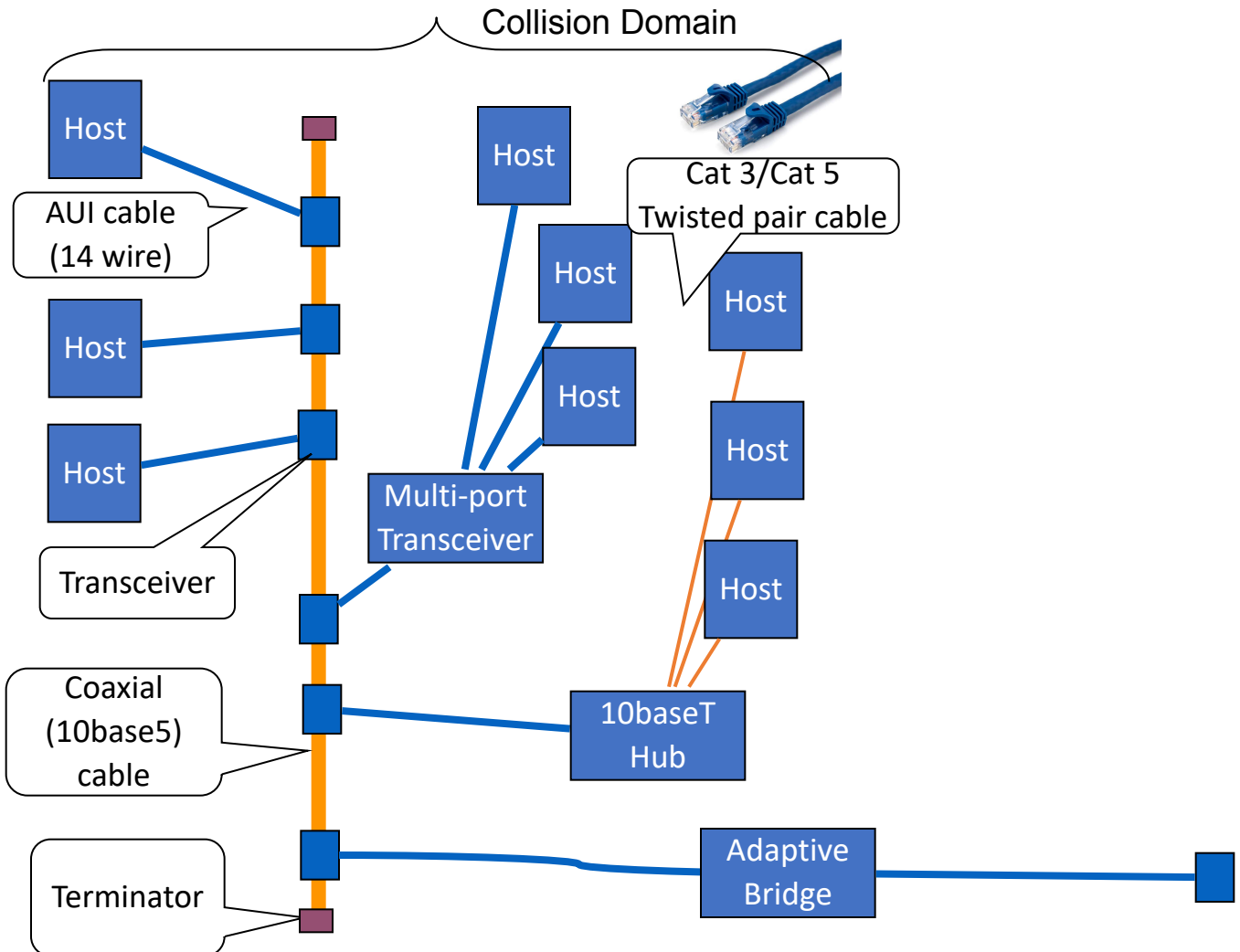
Ethernet Topology (some history)



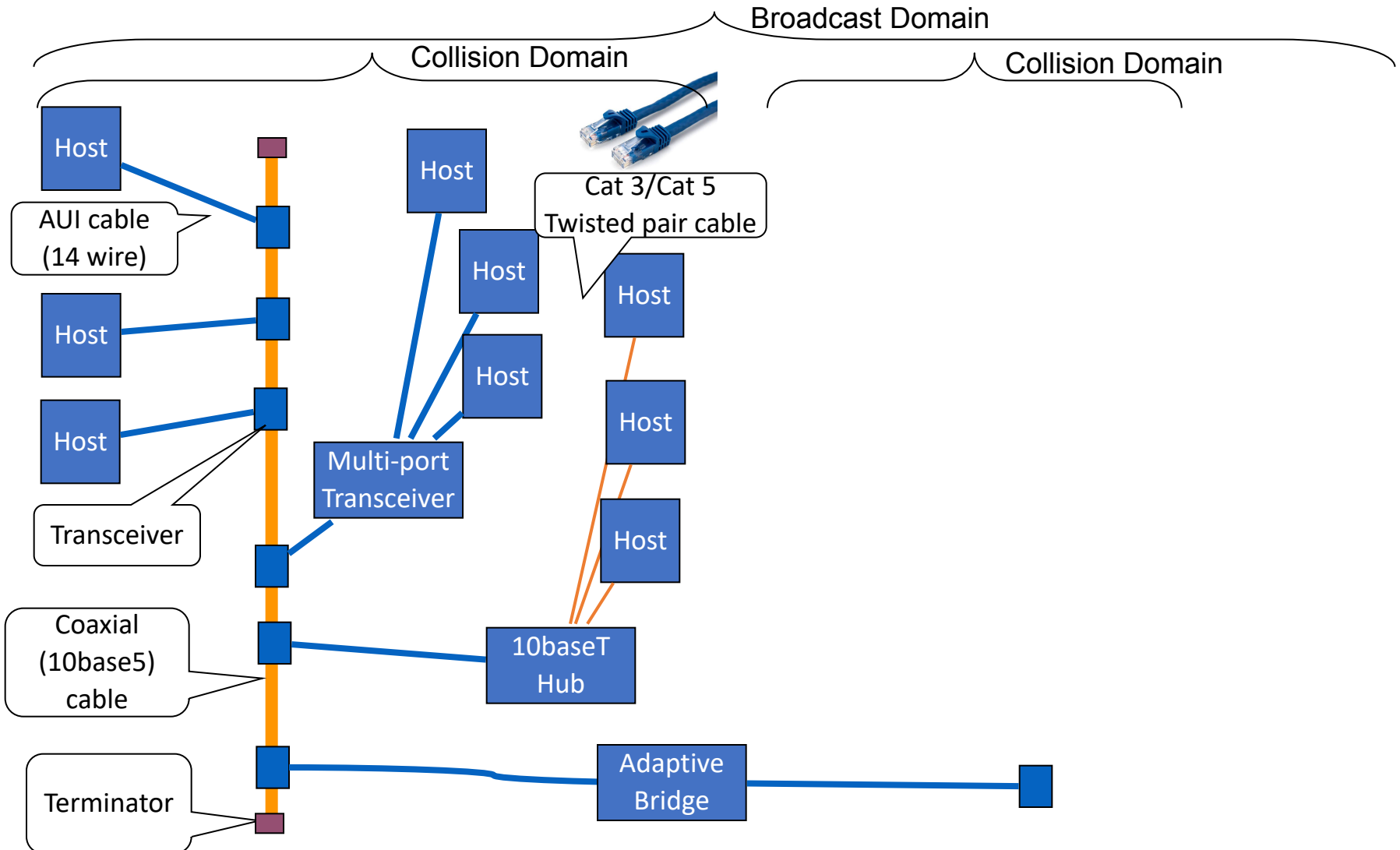
Ethernet Topology (some history)



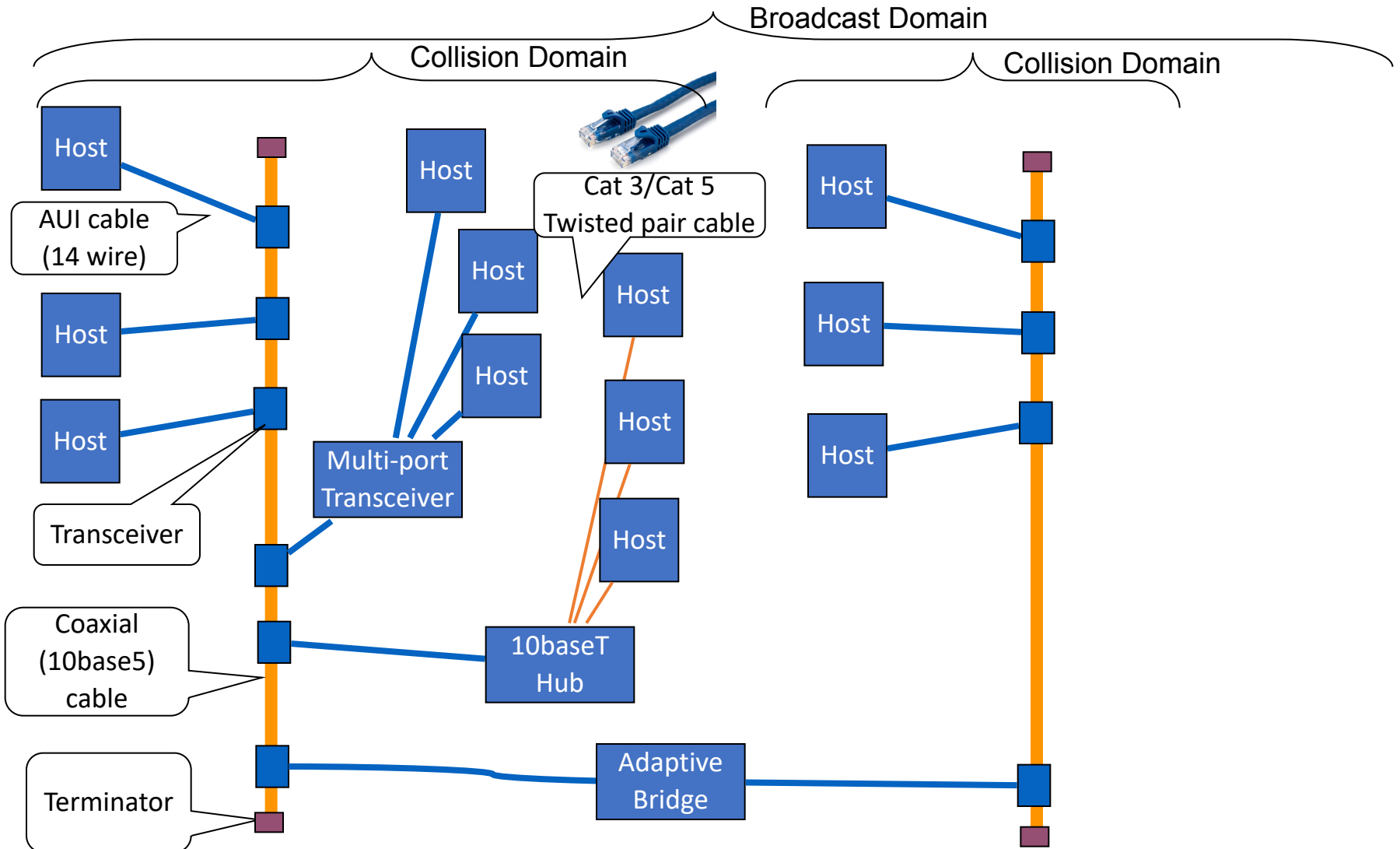
Ethernet Topology (some history)



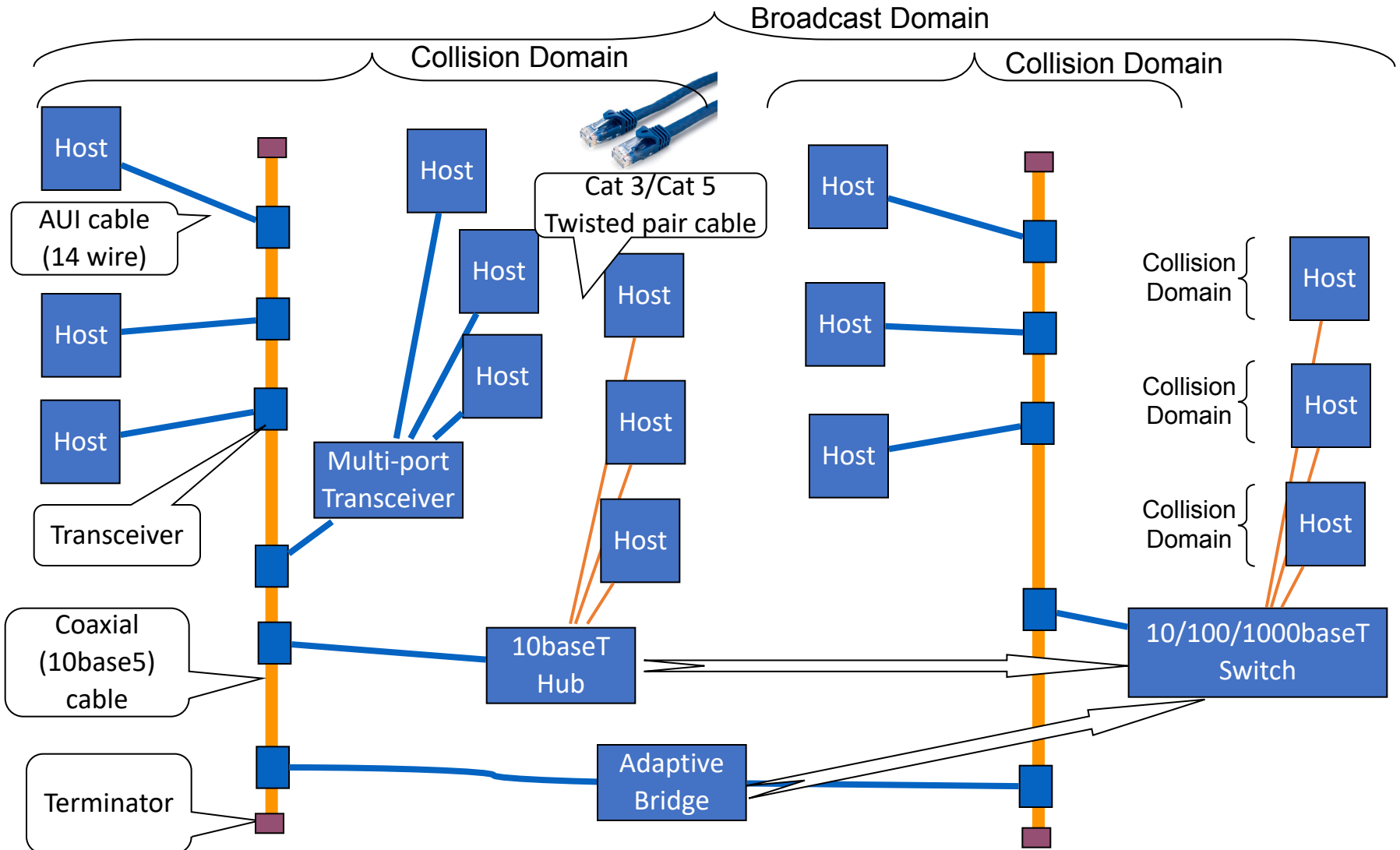
Ethernet Topology (some history)



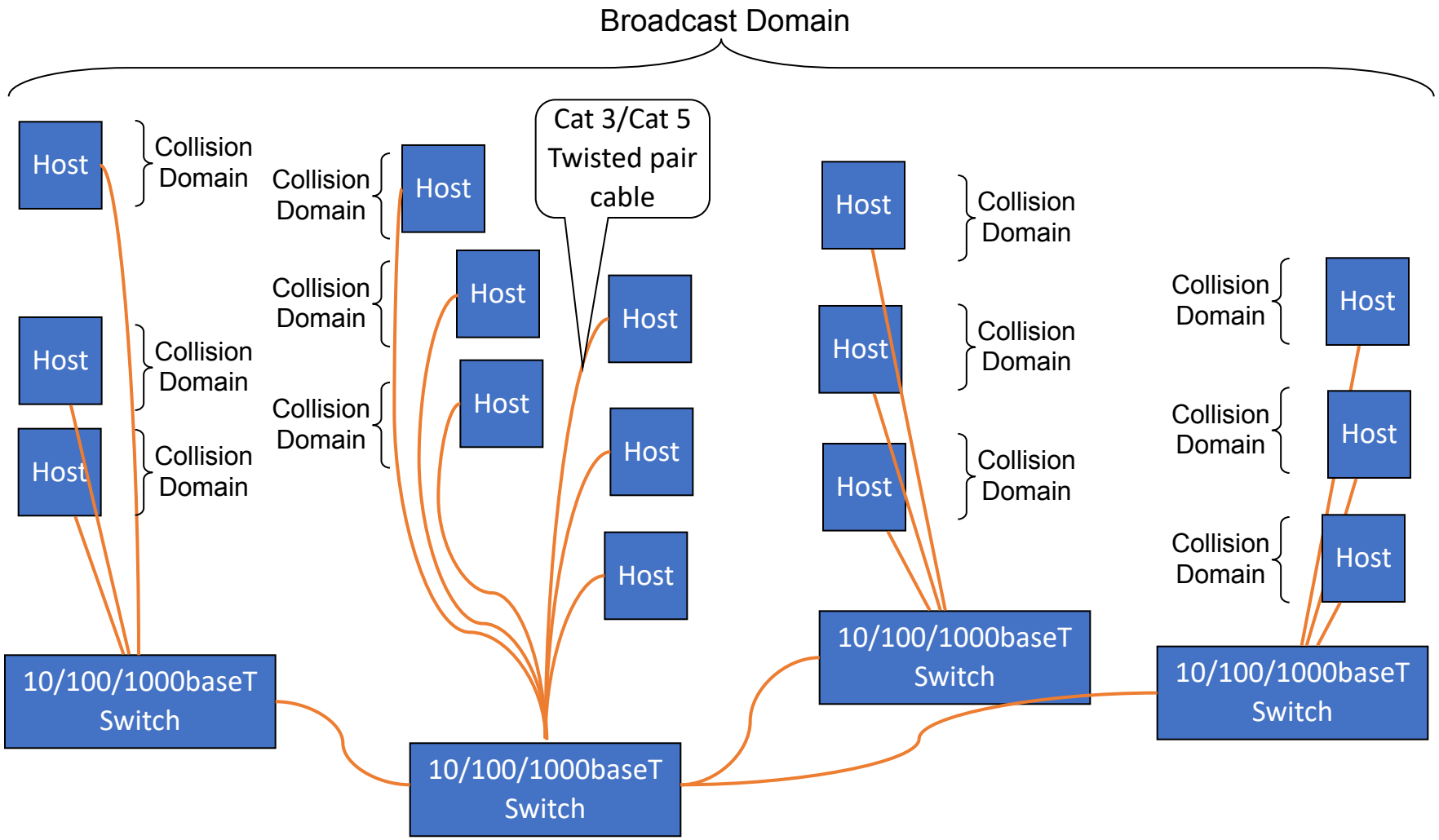
Ethernet Topology (some history)



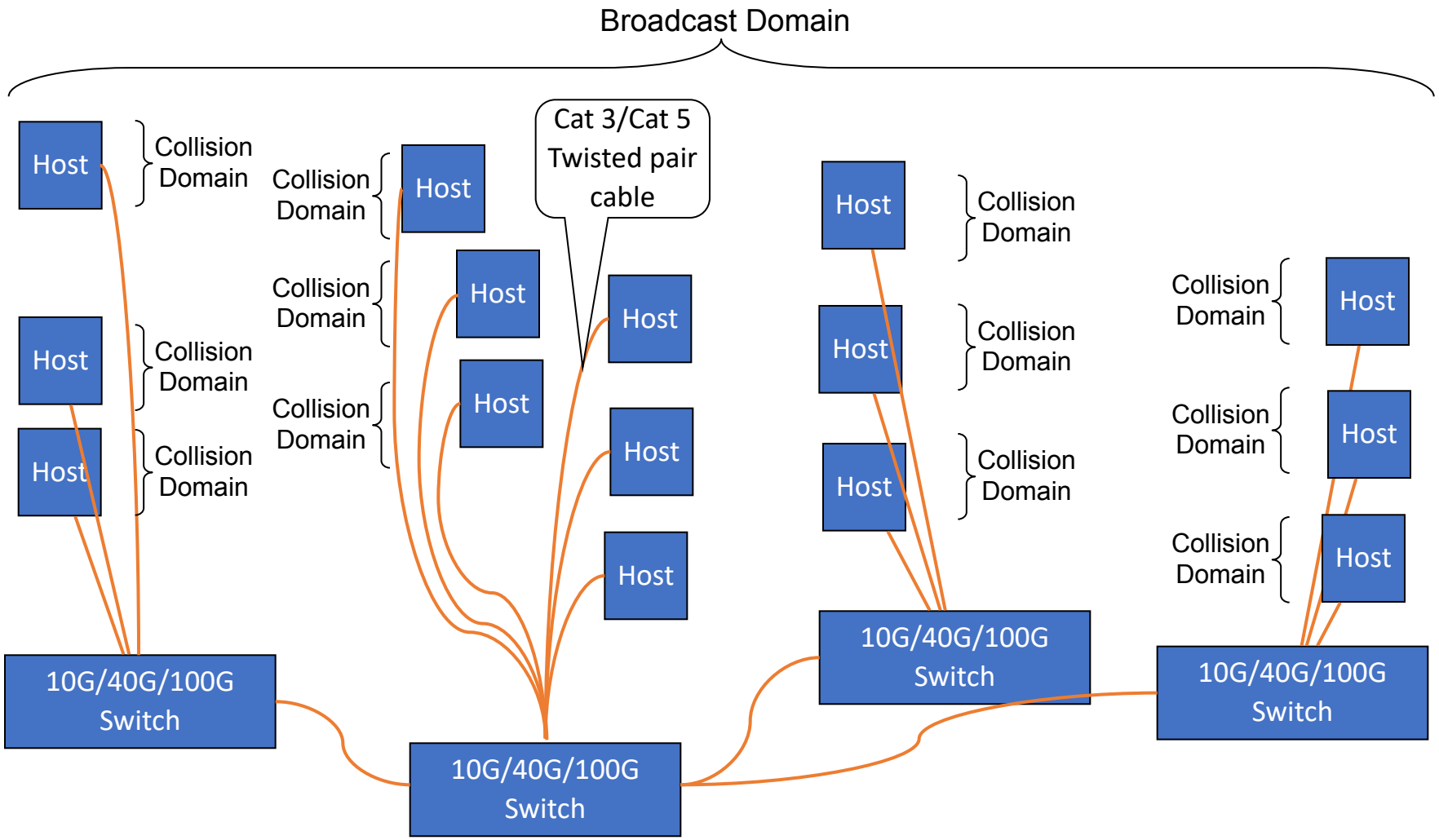
Ethernet Topology (some history)



Ethernet Topology (some history)



Ethernet Topology (modern)



How to deal with collisions during transmission?

CSMA/CD (Carrier Sense Multiple Access/Collision Detection)

How to deal with collisions during transmission?

CSMA/CD (Carrier Sense Multiple Access/Collision Detection)

- Check physical medium while transmitting to detect possible interference
- If interference, retry after exponential backoff
- Performance problem in large collision domains, alleviated by modern switches

How to deal with collisions during transmission?

CSMA/CD (Carrier Sense Multiple Access/Collision Detection)

- Check physical medium while transmitting to detect possible interference
- If interference, retry after exponential backoff
- Performance problem in large collision domains, alleviated by modern switches

CSMA/CA (Collision Avoidance)

- First handshake with destination to ensure they are ready to receive transmission
- Still needed in wireless networks (WiFi)

Ethernet Bridging Algorithm (simplified)

Bridge Table		
MAC Addr	Last Seen on Port	Last Used
002080 00324a	*FastEthernet 1/4*	*12 sec ago*
0003ba 51733a	*FastEthernet 2/9*	*72 sec ago*
. . .	. . .	. . .

- When a frame enters the bridge (switch):

Ethernet Bridging Algorithm (simplified)

Bridge Table		
MAC Addr	Last Seen on Port	Last Used
002080 00324a	*FastEthernet 1/4*	*12 sec ago*
0003ba 51733a	*FastEthernet 2/9*	*72 sec ago*
. . .	. . .	. . .

- When a frame enters the bridge (switch):
  - If the **destination** MAC address is in the bridge table
    - Send the frame out the port listed in the bridge table entry
  - Otherwise
    - Flood the frame out all ports except the ingress port
  - Put the **source** MAC address in the bridge table, if necessary, along with the frame ingress port and mark that entry “recently used”

Ethernet Bridging Algorithm (simplified)

Bridge Table		
MAC Addr	Last Seen on Port	Last Used
002080 00324a	*FastEthernet 1/4*	*12 sec ago*
0003ba 51733a	*FastEthernet 2/9*	*72 sec ago*
. . .	. . .	. . .

- When a frame enters the bridge (switch):
  - If the **destination** MAC address is in the bridge table
    - Send the frame out the port listed in the bridge table entry
  - Otherwise
    - Flood the frame out all ports except the ingress port
  - Put the **source** MAC address in the bridge table, if necessary, along with the frame ingress port and mark that entry “recently used”
  - Remove any “stale” entries in the bridge table

Something Is Still Missing

Something Is Still Missing

- Now more than one message can be in transit at a time!

Something Is Still Missing

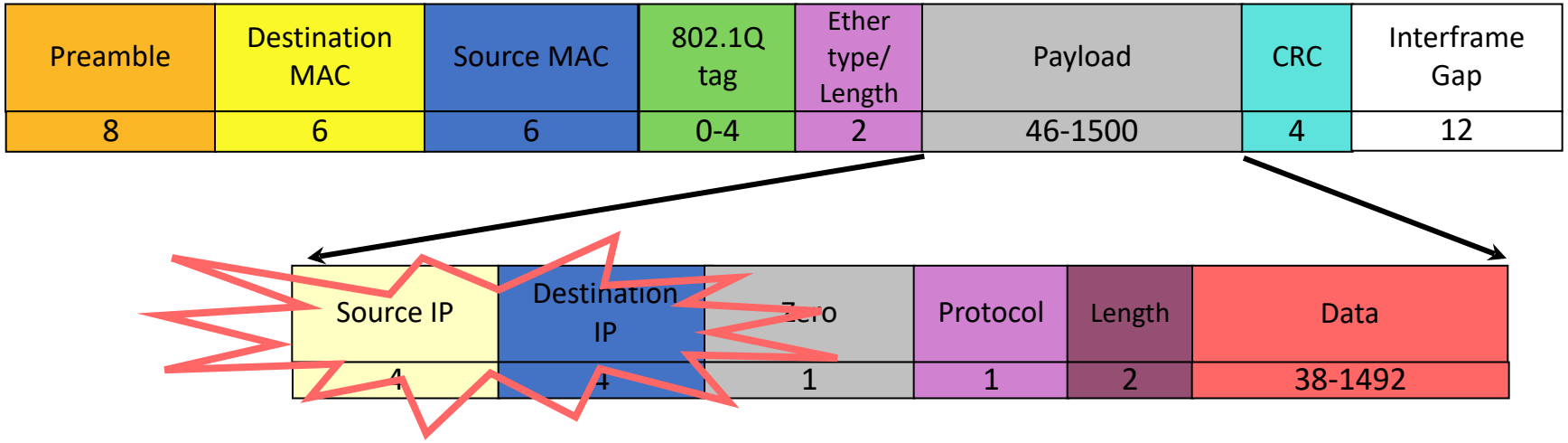
- Now more than one message can be in transit at a time!
- We've only talked about MAC addresses.

Something Is Still Missing

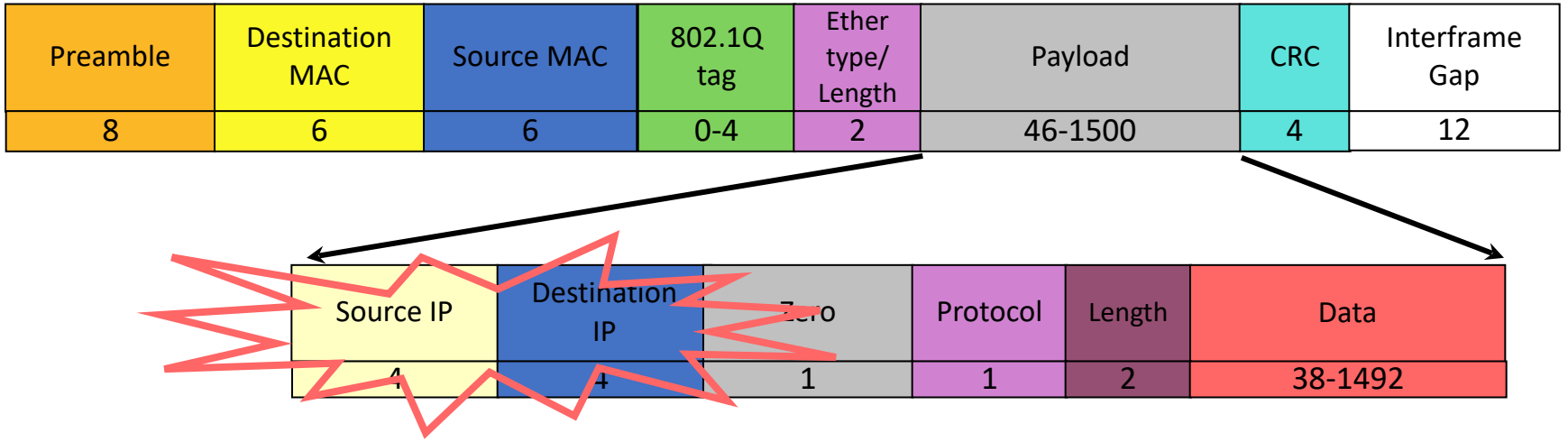
- Now more than one message can be in transit at a time!

- We've only talked about MAC addresses.
- Where are the IP addresses?

Layer 3 (IP) packets

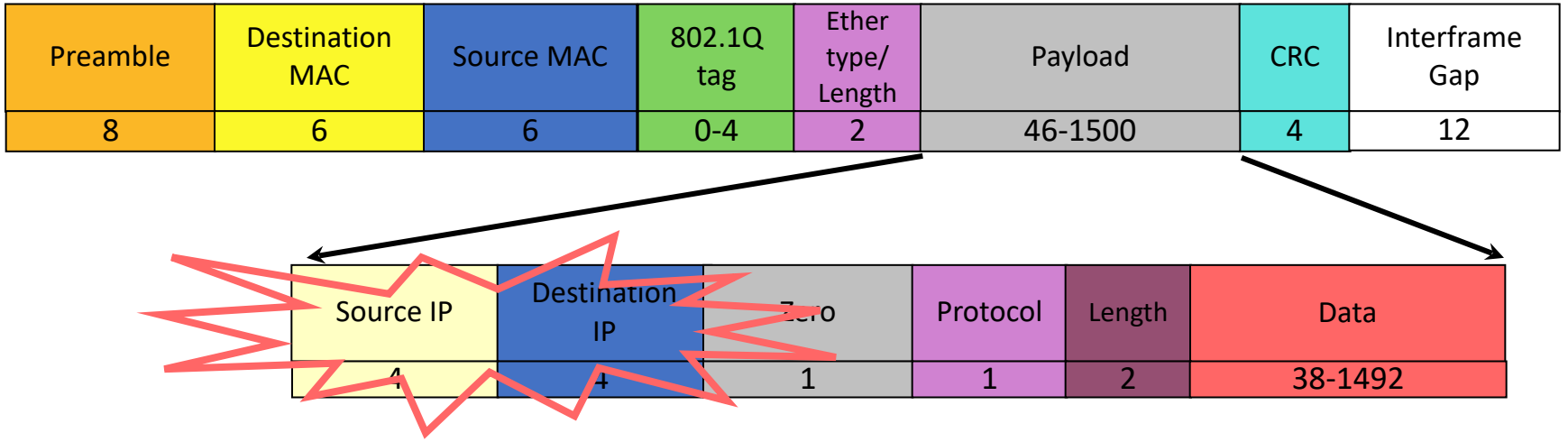


Layer 3 (IP) packets



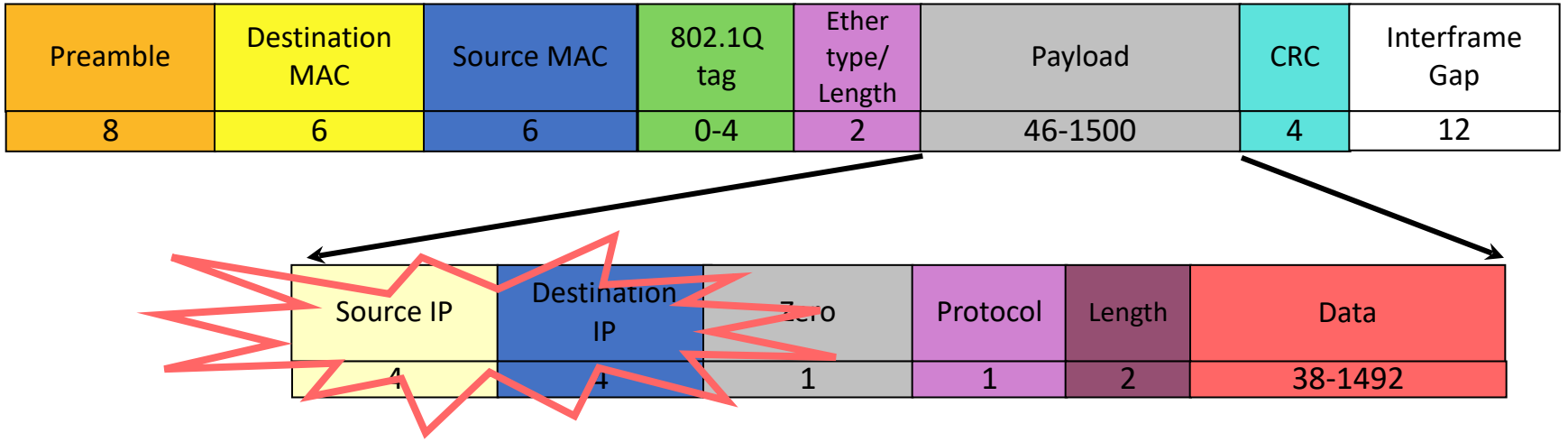
- IP addresses appear in the Ethernet frame payload, not the header

Layer 3 (IP) packets



- IP addresses appear in the Ethernet frame payload, not the header
- But Ethernet hardware only examines the MAC addresses in the Ethernet frame(!)

Layer 3 (IP) packets



- IP addresses appear in the Ethernet frame payload, not the header
- But Ethernet hardware only examines the MAC addresses in the Ethernet frame(!)
- How do we connect these two layers?

ARP

- Address Resolution Protocol

ARP

- Address Resolution Protocol
- Used with Ethernet to discover IP addresses

ARP

- Address Resolution Protocol
- Used with Ethernet to discover IP addresses
- How?

ARP

- Address Resolution Protocol
- Used with Ethernet to discover IP addresses
- How?
  - IP stack needs to send to IP address *a*, but doesn't know what destination MAC address to use.

ARP

- Address Resolution Protocol
- Used with Ethernet to discover IP addresses
- How?
  - IP stack needs to send to IP address *a*, but doesn't know what destination MAC address to use.
  - Broadcast an ARP-request message: "Who has IP address *a*?"

ARP

- Address Resolution Protocol
- Used with Ethernet to discover IP addresses
- How?
  - IP stack needs to send to IP address *a*, but doesn't know what destination MAC address to use.
  - Broadcast an ARP-request message: "Who has IP address *a*?"
  - Wait on unicast ARP-response: "I have IP address *a*."

ARP

- Address Resolution Protocol
- Used with Ethernet to discover IP addresses
- How?
  - IP stack needs to send to IP address *a*, but doesn't know what destination MAC address to use.
  - Broadcast an ARP-request message: "Who has IP address *a*?"
  - Wait on unicast ARP-response: "I have IP address *a*."
  - Store the response in a local *ARP Table*, discarding entries when they become stale.

ARP

- Address Resolution Protocol
- Used with Ethernet to discover IP addresses
- How?
  - IP stack needs to send to IP address *a*, but doesn't know what destination MAC address to use.
  - Broadcast an ARP-request message: "Who has IP address *a*?"
  - Wait on unicast ARP-response: "I have IP address *a*."
  - Store the response in a local *ARP Table*, discarding entries when they become stale.
  - Use the corresponding entry in the ARP table to set the destination MAC address of the frame.

ARP Example

```
bash-3.00# arp -an
Net to Media Table: IPv4
Device   IP Address               Mask      Flags      Phys Addr
------ -------------------- --------------- -------- ---------------
bge0   130.207.165.229      255.255.255.255 o        00:50:56:91:4a:2b
bge0   130.207.165.248      255.255.255.255 o        00:14:4f:f2:8c:ce
bge0   130.207.165.240      255.255.255.255 o        00:50:56:91:62:be
bge0   130.207.165.242      255.255.255.255 o        00:14:4f:21:14:f6
bge0   130.207.165.245      255.255.255.255 o        00:14:4f:94:d7:0c
bge0   130.207.165.194      255.255.255.255 o        00:50:56:91:59:25
bge0   130.207.165.197      255.255.255.255 o        00:50:56:91:46:a3
bge0   130.207.165.221      255.255.255.255 o        00:50:56:91:4c:af
bge0   130.207.165.163      255.255.255.255 o        00:14:4f:7d:7f:58
bge0   130.207.165.137      255.255.255.255          00:03:ba:1a:28:a7
bge0   130.207.165.138      255.255.255.255          00:03:ba:1a:2a:ff
bge0   130.207.165.158      255.255.255.255 o        00:14:4f:1e:26:ec
bge0   130.207.165.96       255.255.255.255 o        00:1b:24:1d:eb:7c
bge0   130.207.165.103      255.255.255.255 o        00:50:56:a4:7e:df
bge0   130.207.165.122      255.255.255.255 o        00:50:56:91:43:f8
```

VLANs

VLANs

- Virtual LAN, a.k.a. VLAN
  - A group of hosts that can communicate as if they were attached to the same **broadcast domain**, regardless of their physical location.
  - A VLAN has the same attributes as a physical LAN, but it allows for hosts to be grouped together even if they are not connected to the same network switch.
  - Network reconfiguration can be done through software instead of physically relocating devices and wires.

Geographic Addressing

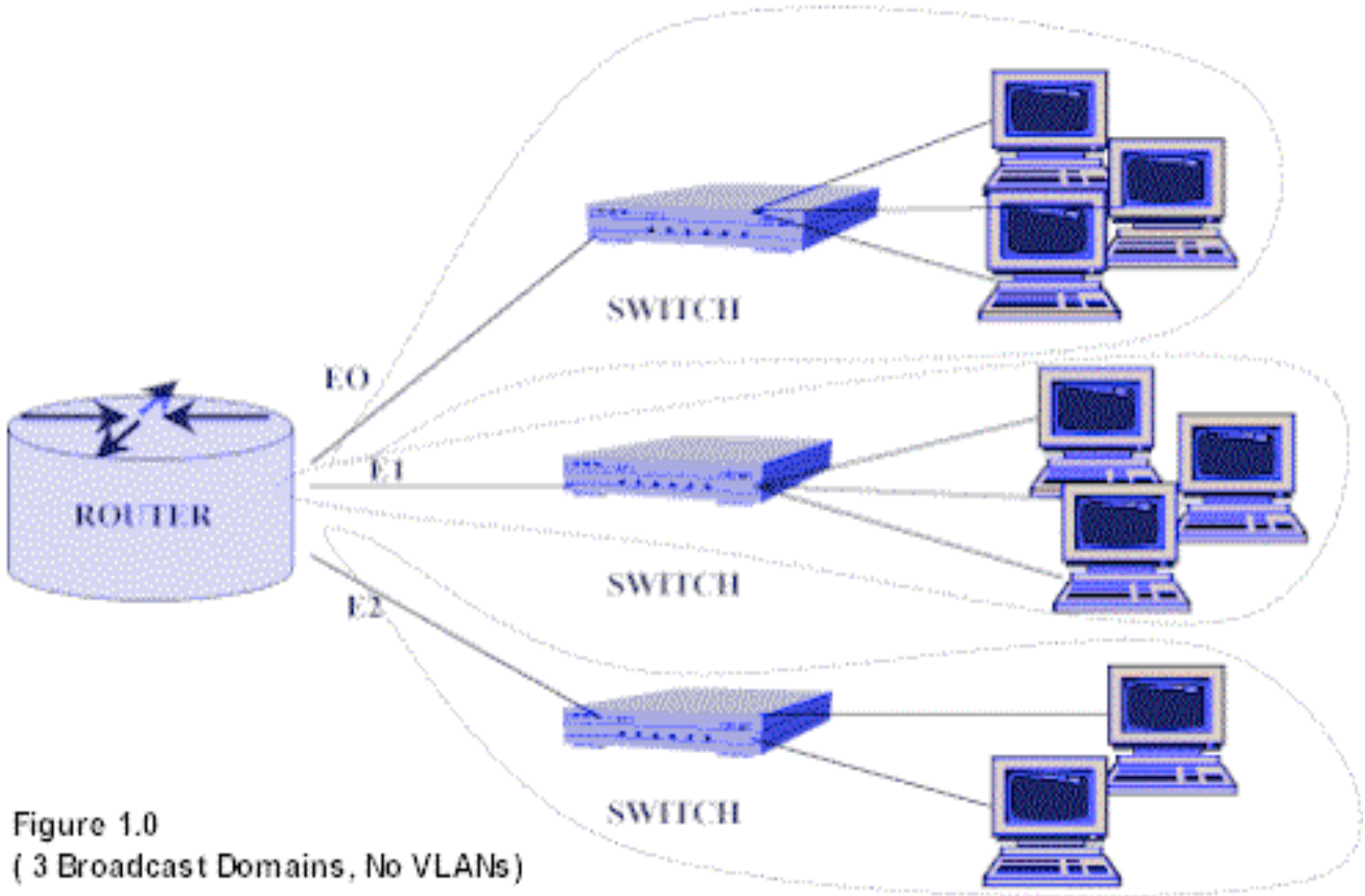
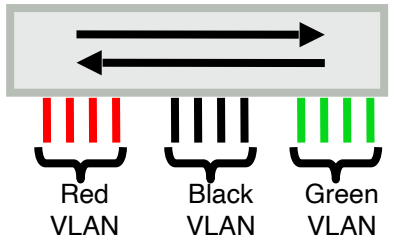


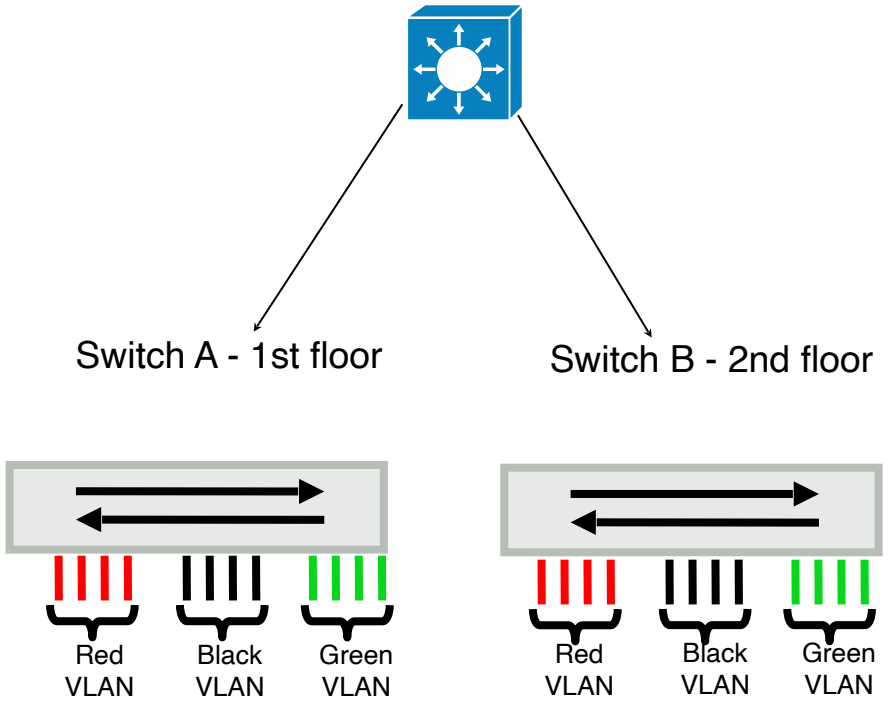
Figure 1.0
(3 Broadcast Domains, No VLANs)

After VLANs

Switch A



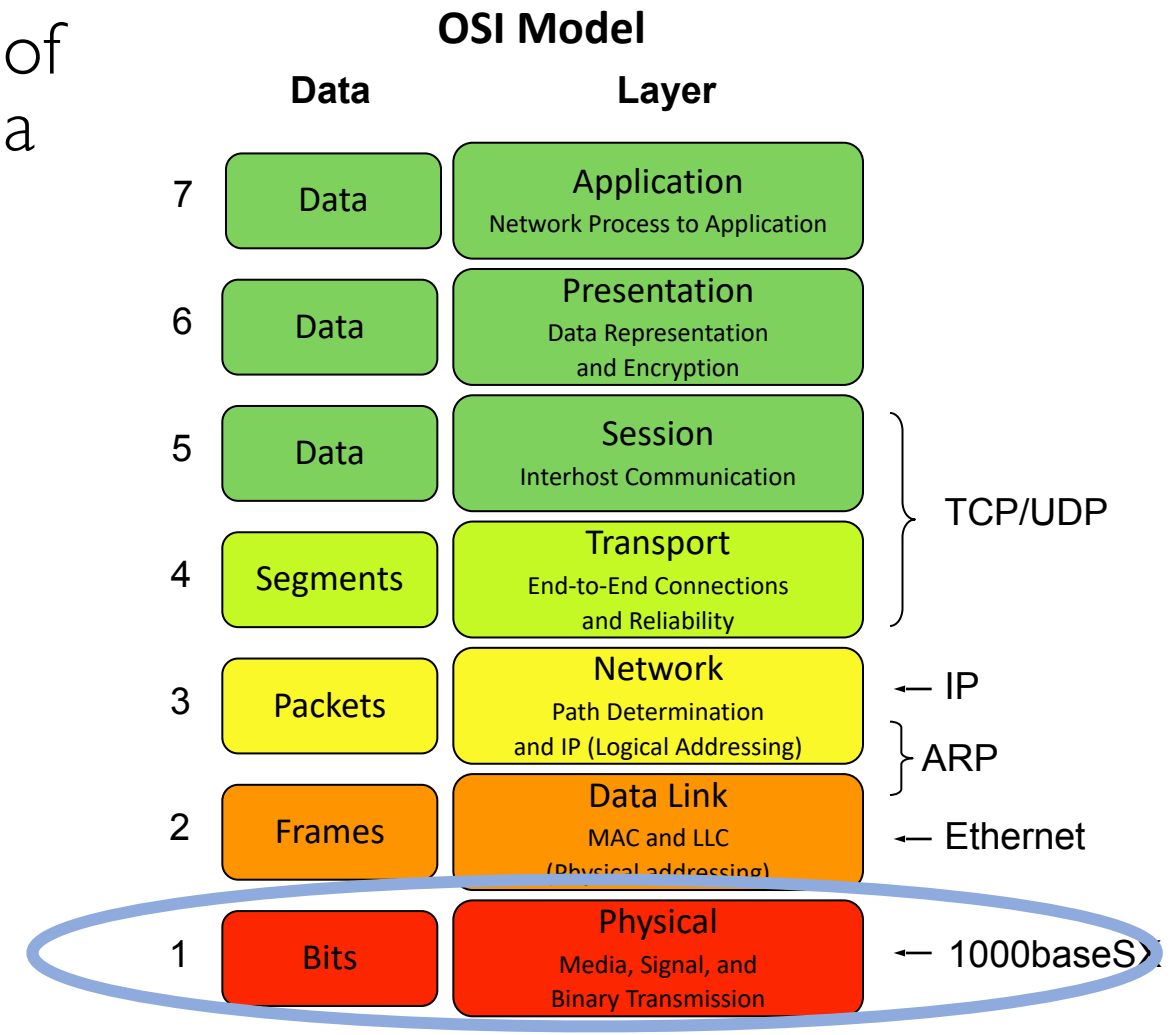
- Each logical VLAN is like a separate physical bridge
  - College of Engineering (red)
  - College of Computing (black)
  - Classroom network (green)



- Each logical VLAN is like a separate physical bridge
- VLANs can span across multiple switches

Now let's look at the last layer

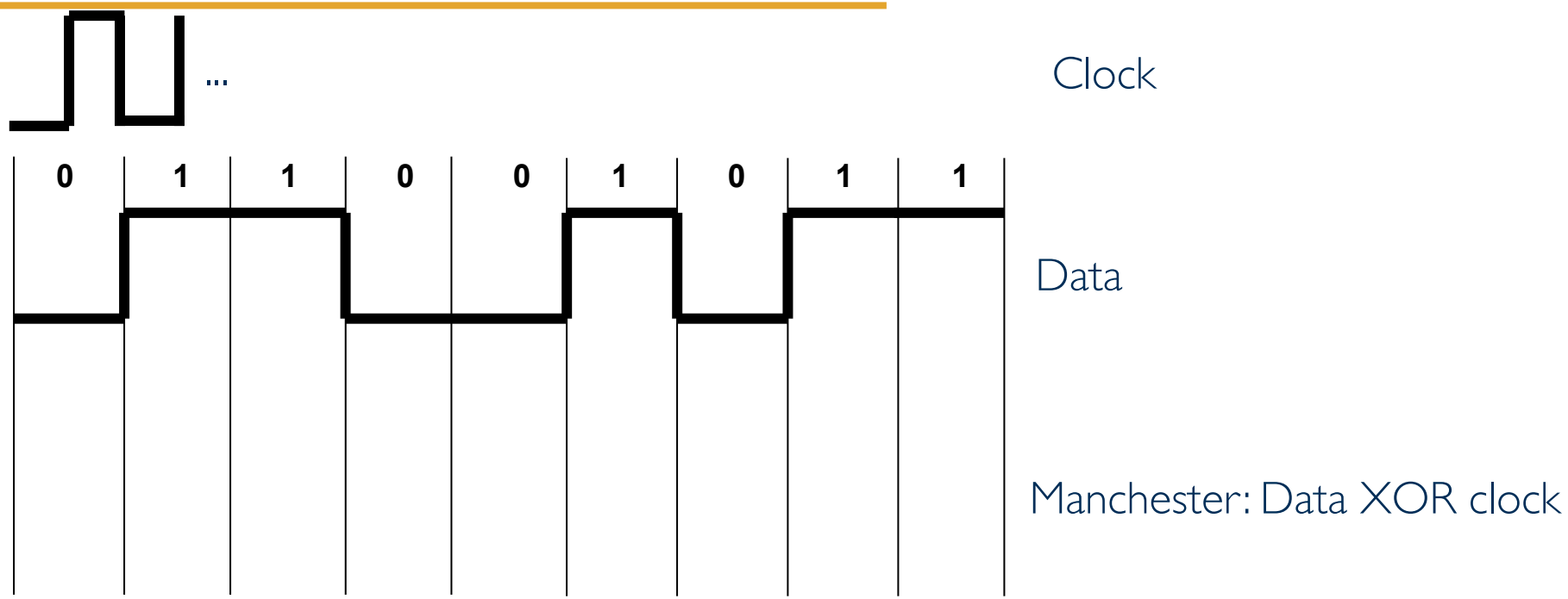
- Ethernet standards allow use of several types of physical media



Ethernet physical media standards

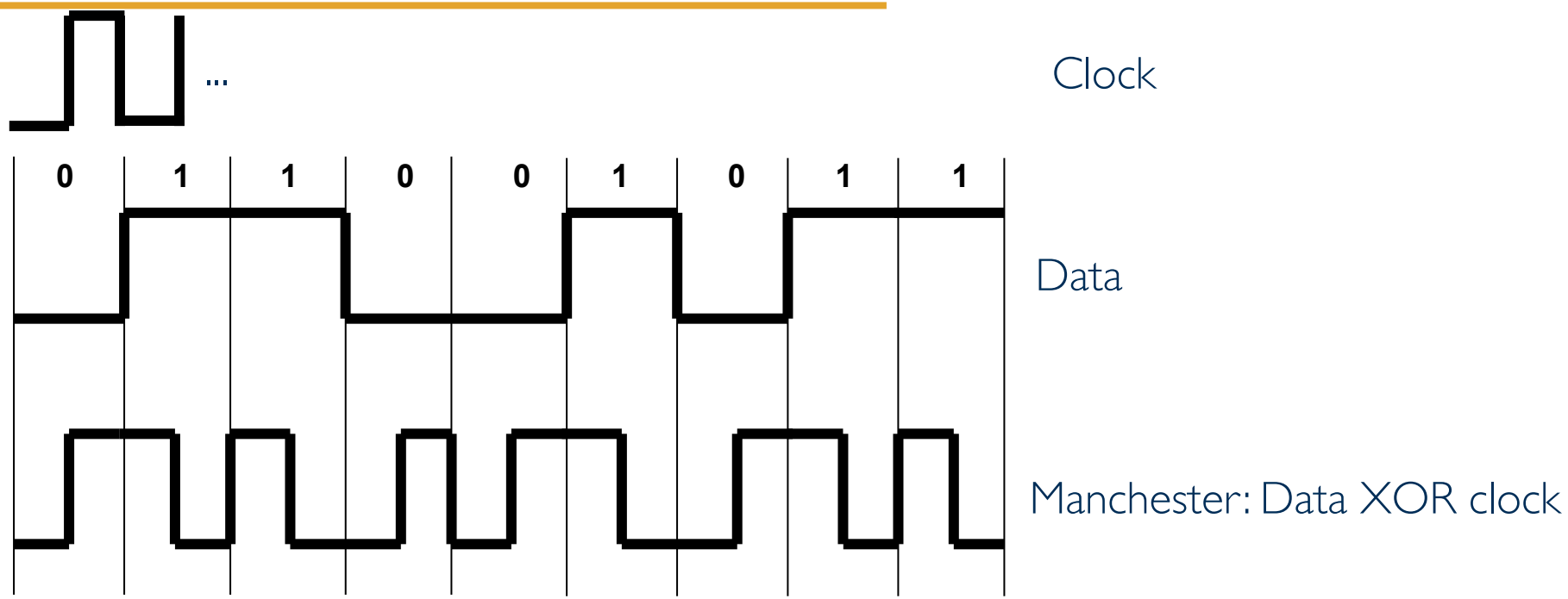
- Twisted pair copper (1m – 100m reach)
  - 10base-T
    100base-T
    1000base-TX
    10Gbase-T
- Fiber optic (100m – 40km reach)
  - 1000base-SX, 1000base-LX
    10Gbase-SR, 10Gbase-LR
    40Gbase-SR4, 40Gbase-LR4
    100Gbase-SR4, 100Gbase-SR10, 100Gbase-LR4
    and many more

Manchester encoding

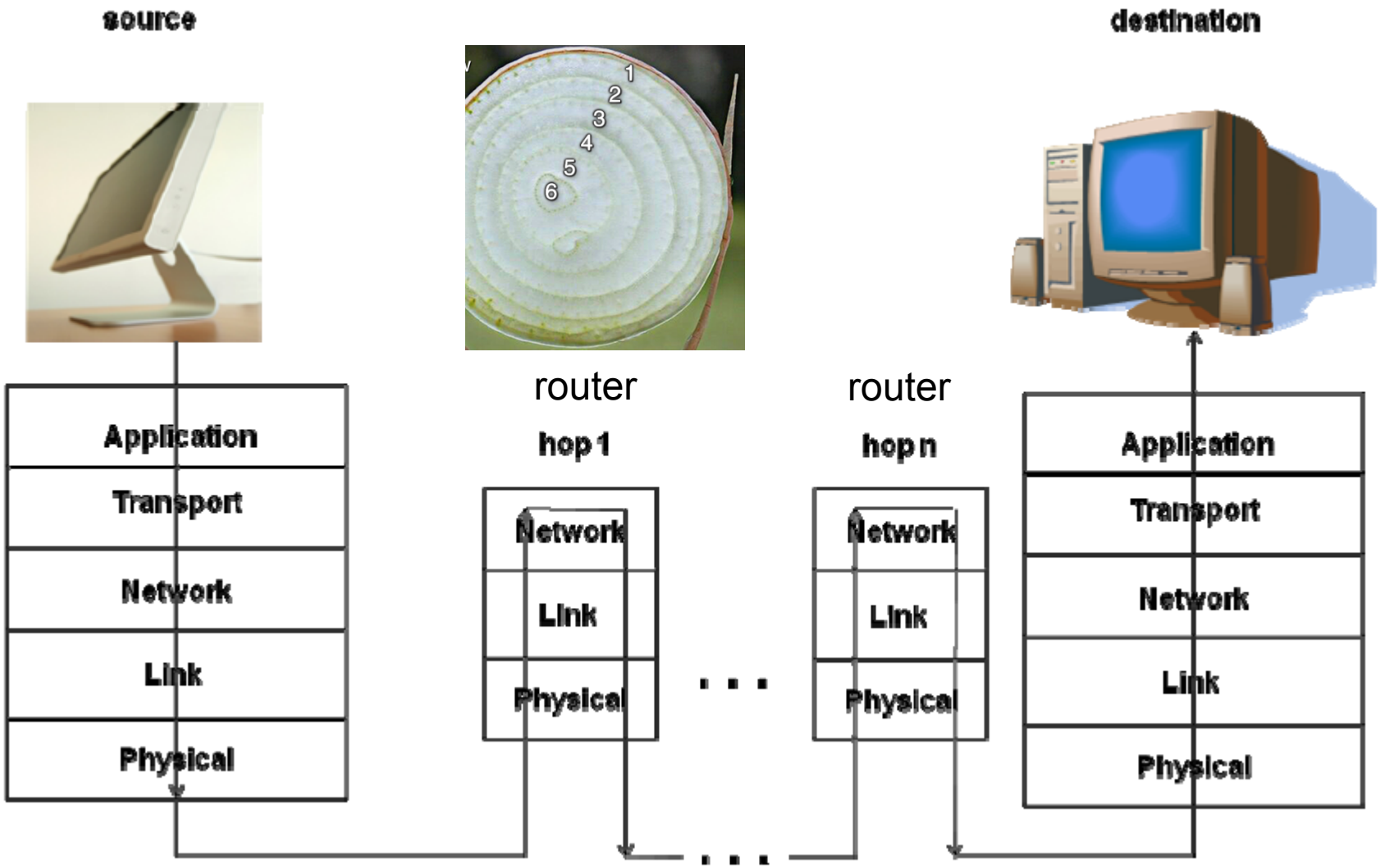


- Ethernet coax signaling uses Manchester encoding which is a scheme that guarantees at least one voltage transition during each clock tick

Manchester encoding



- Ethernet coax signaling uses Manchester encoding which is a scheme that guarantees at least one voltage transition during each clock tick



PHY (Layer 1)
Header

Ethernet
(Layer 2)
Header

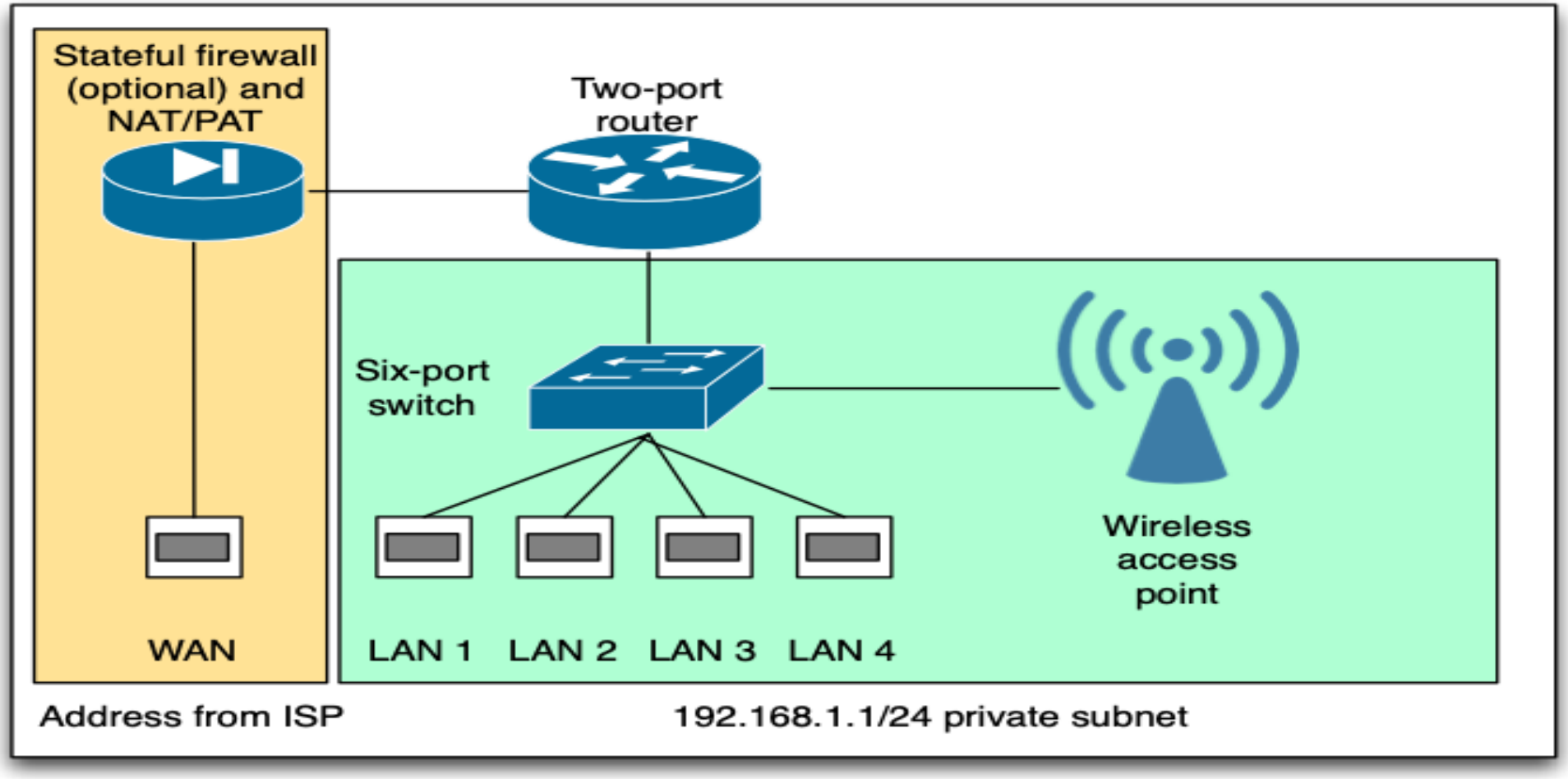
IP (Layer 3)
Header

TCP (Layer 4-5)
Header

TCP Payload

Ethernet
Trailer

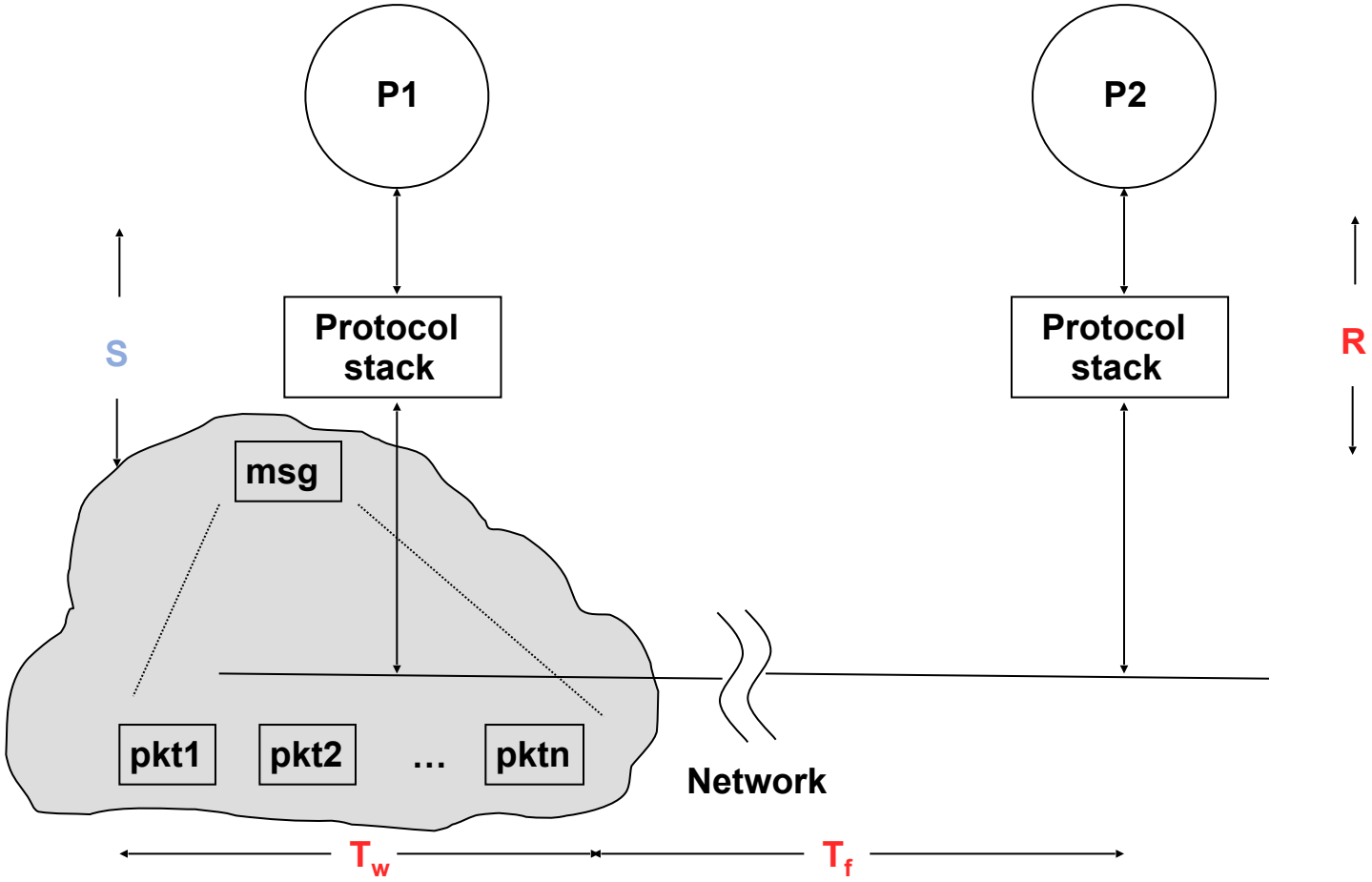
What's in a Home Router?



Performance metrics

Transmission time or end-to-end latency = $S+T_W+T_f+R$

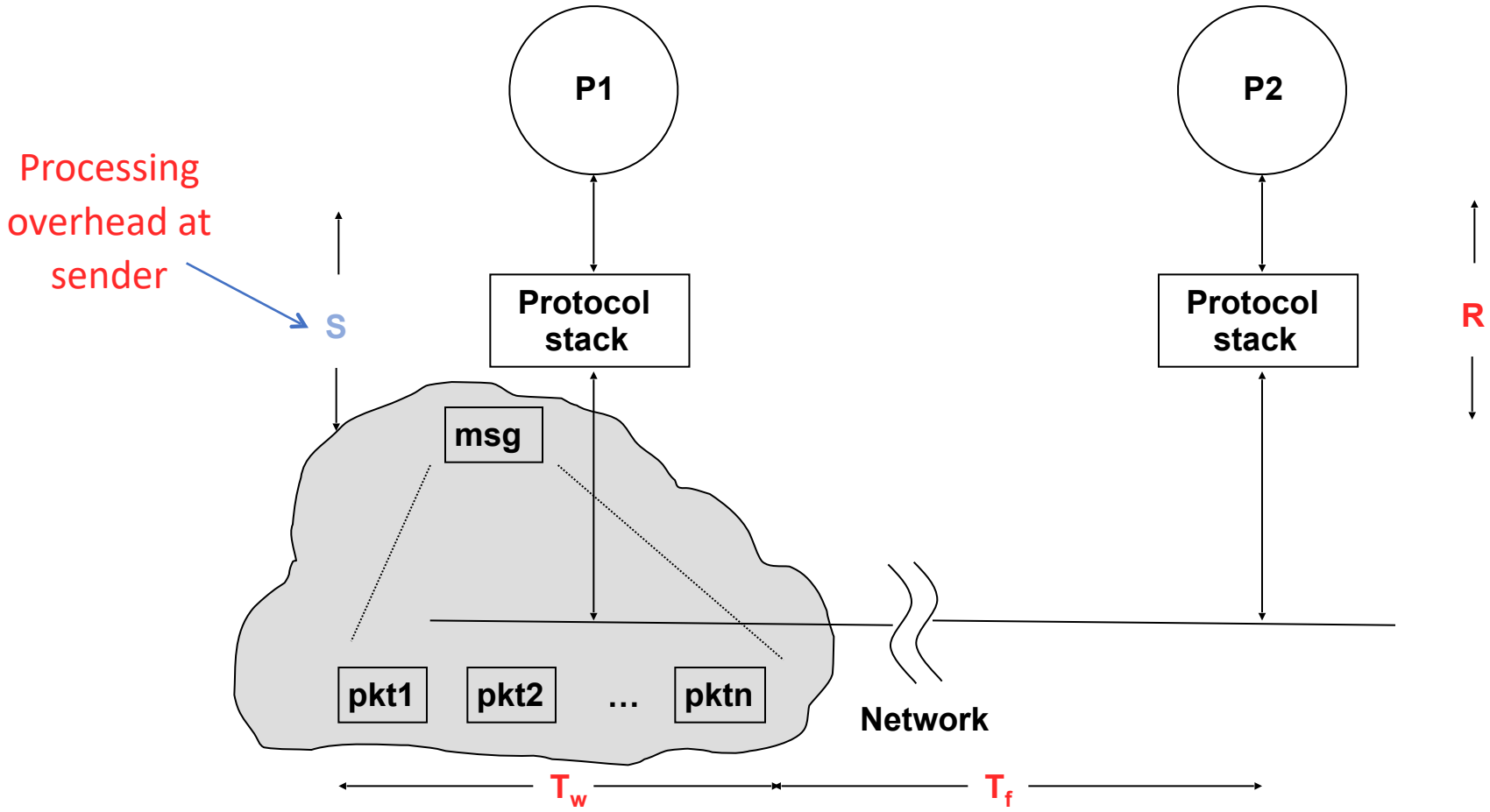
Message throughput = message-size/end-to-end-latency



Performance metrics

Transmission time or end-to-end latency = $S+T_W+T_f+R$

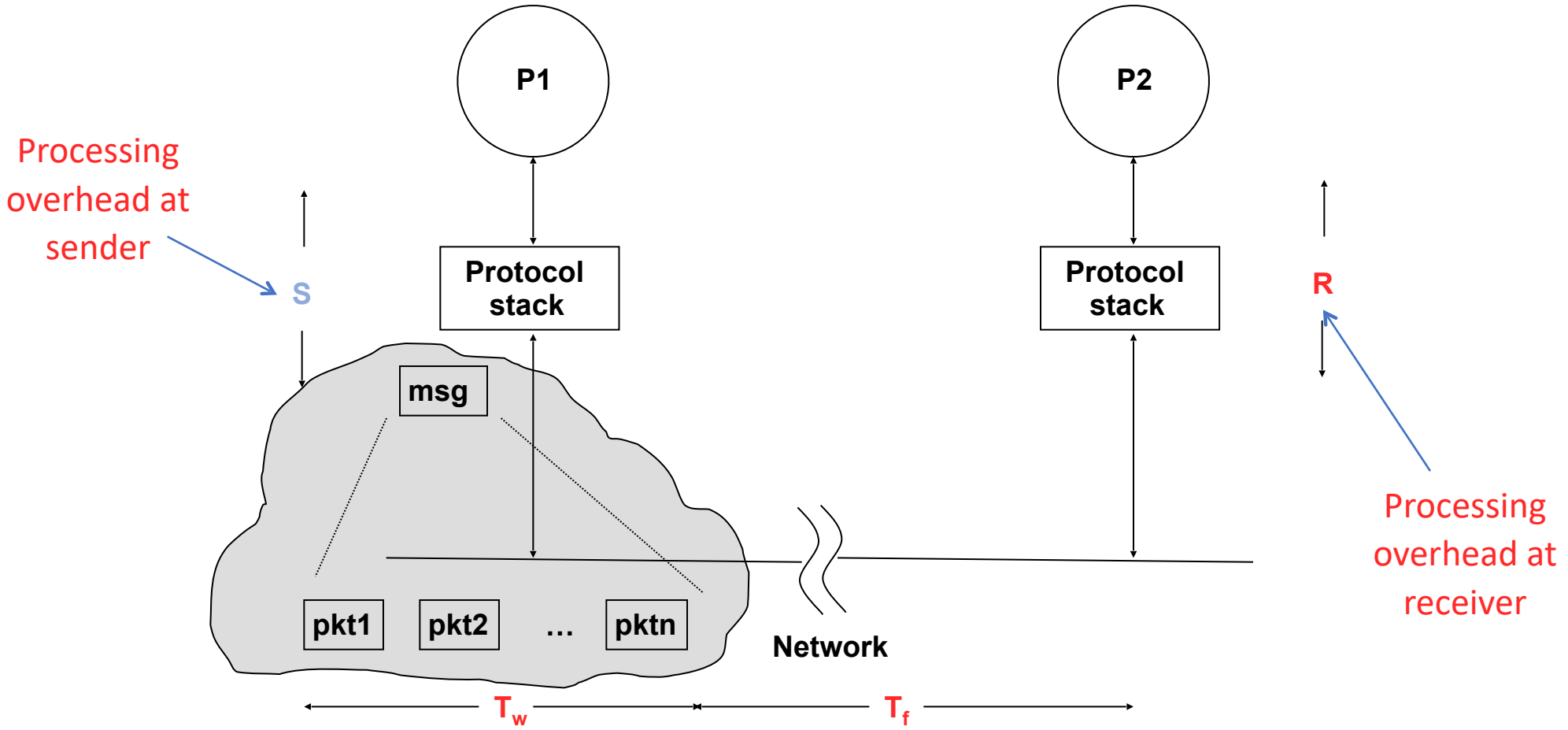
Message throughput = message-size/end-to-end-latency



Performance metrics

Transmission time or end-to-end latency = $S+T_W+T_f+R$

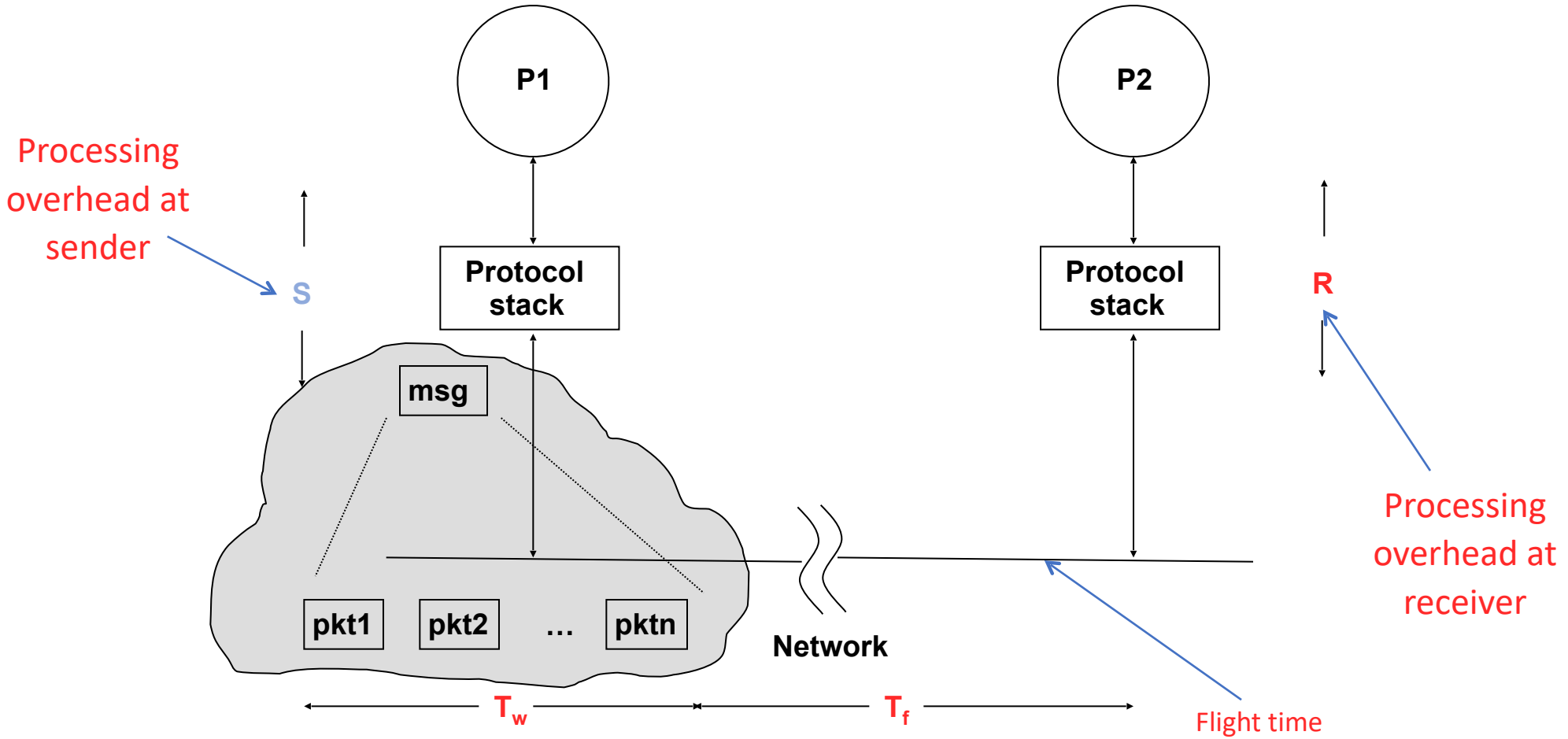
Message throughput = message-size/end-to-end-latency



Performance metrics

Transmission time or end-to-end latency = $S+T_W+T_f+R$

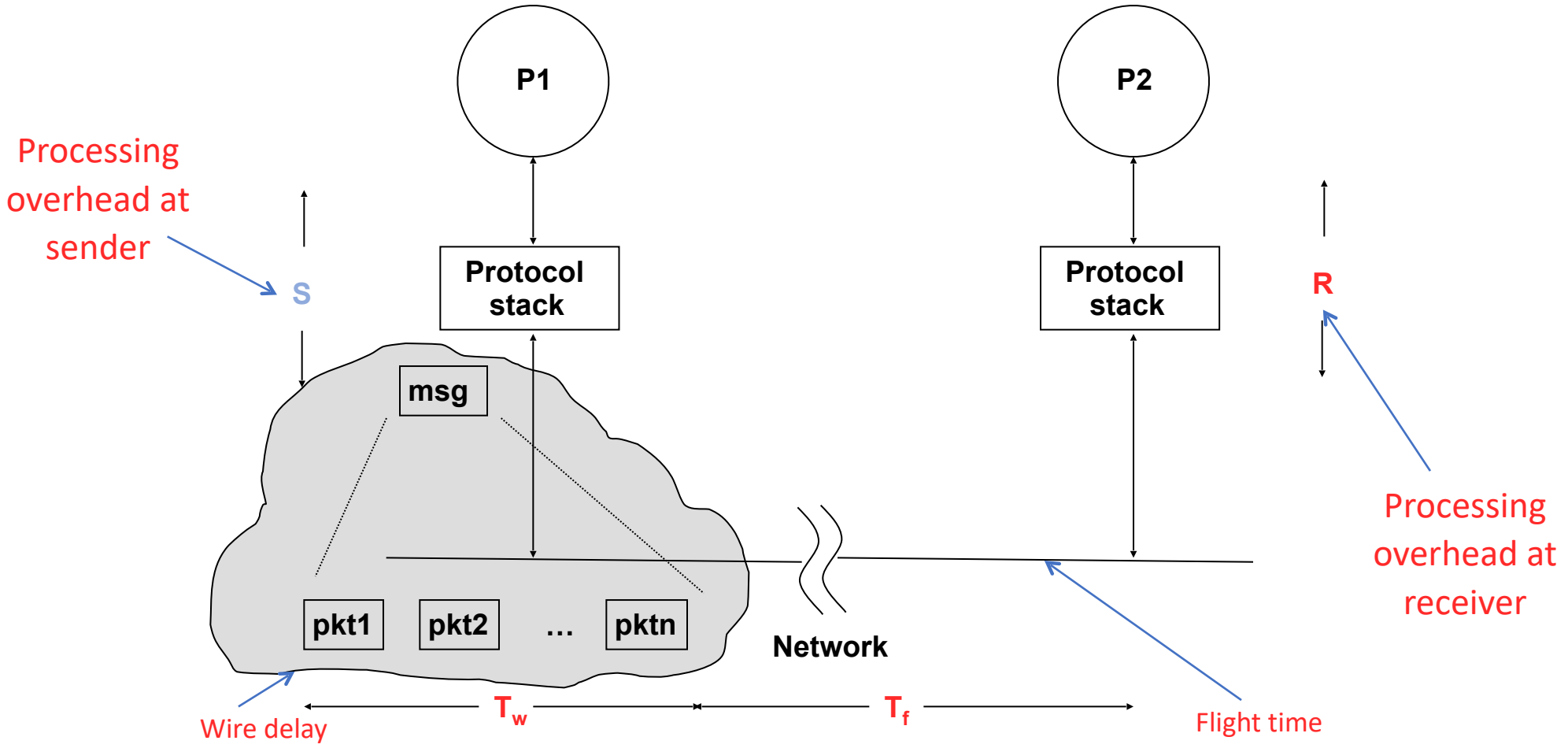
Message throughput = message-size/end-to-end-latency



Performance metrics

Transmission time or end-to-end latency = $S+T_W+T_f+R$

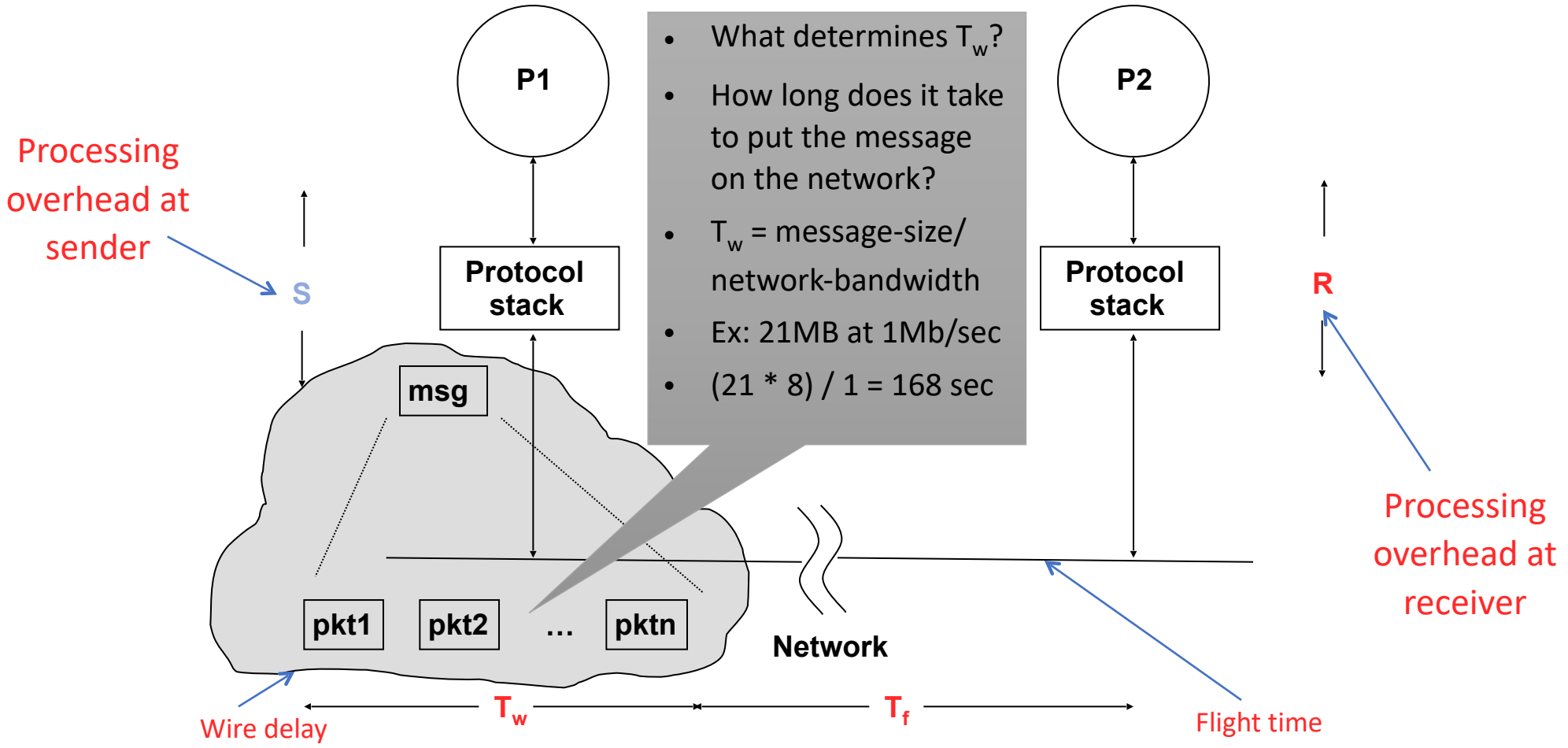
Message throughput = message-size/end-to-end-latency



Performance metrics

Transmission time or end-to-end latency = $S+T_W+T_f+R$

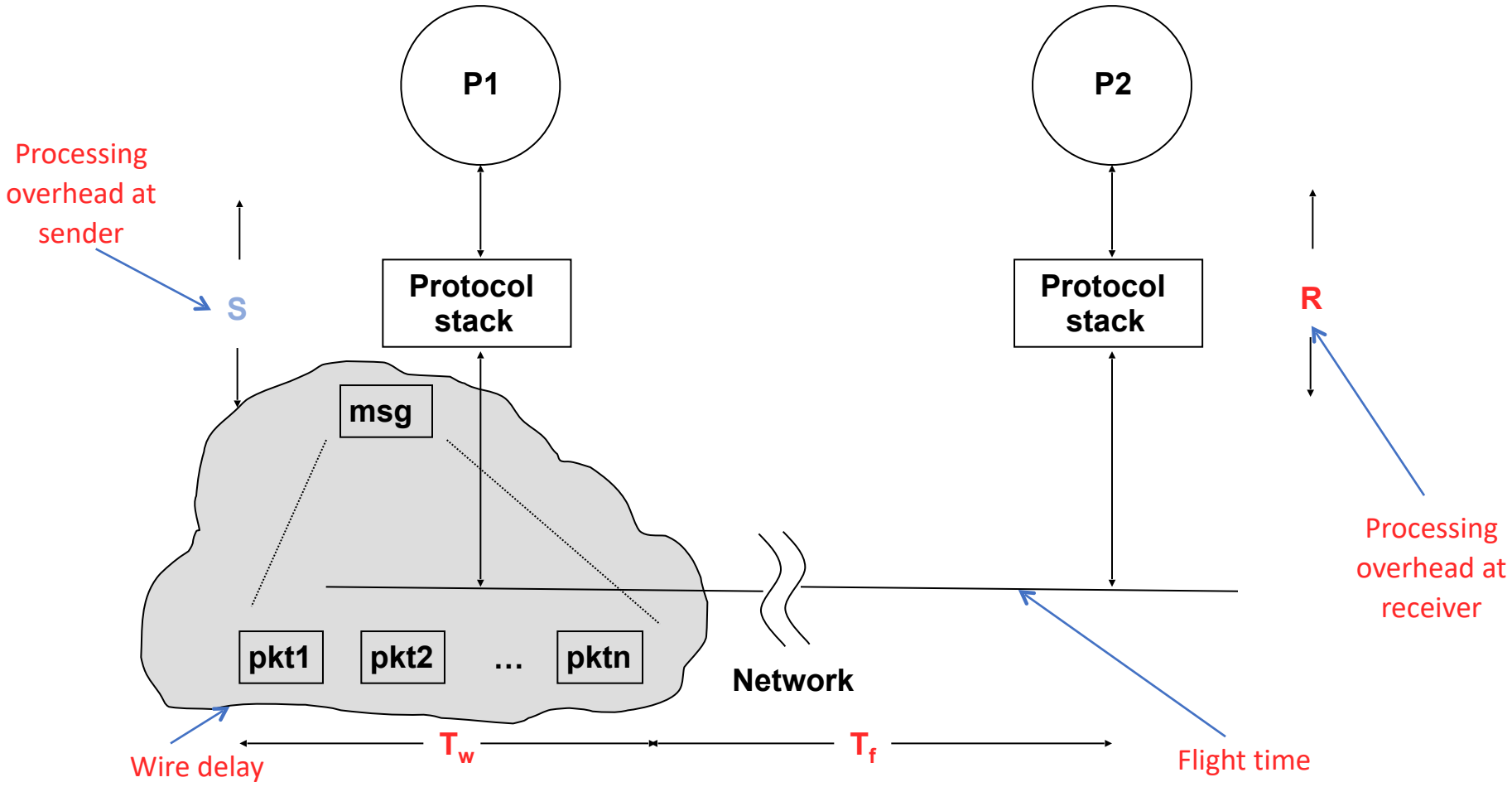
Message throughput = message-size/end-to-end-latency



Performance metrics

Transmission time or end-to-end latency = $S+T_W+T_f+R$

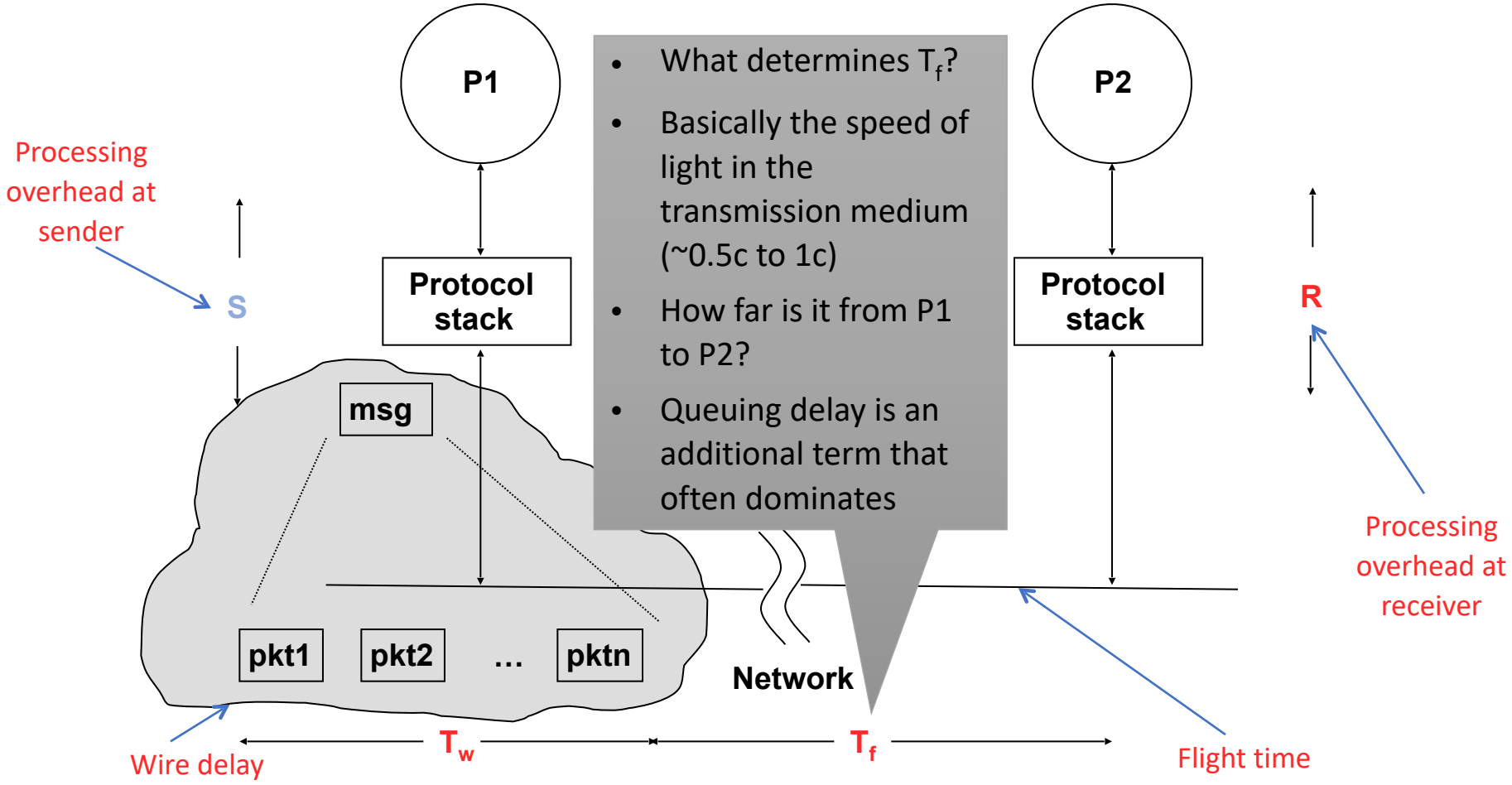
Message throughput = message-size/end-to-end-latency



Performance metrics

Transmission time or end-to-end latency = $S+T_W+T_f+R$

Message throughput = message-size/end-to-end-latency

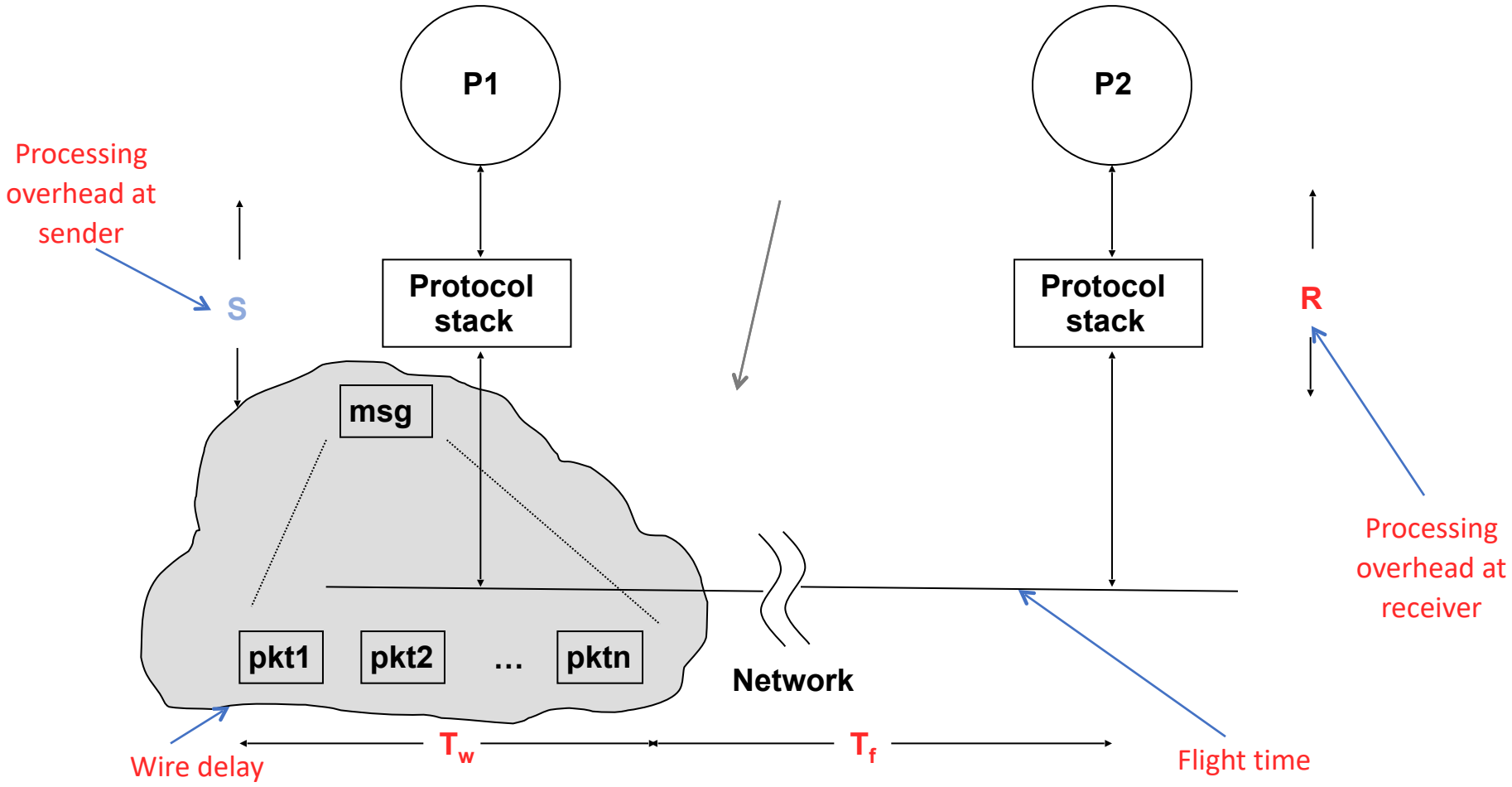


- What determines T_f?
- Basically the speed of light in the transmission medium (~0.5c to 1c)
- How far is it from P1 to P2?
- Queuing delay is an additional term that often dominates

Performance metrics

Transmission time or end-to-end latency = $S+T_W+T_f+R$

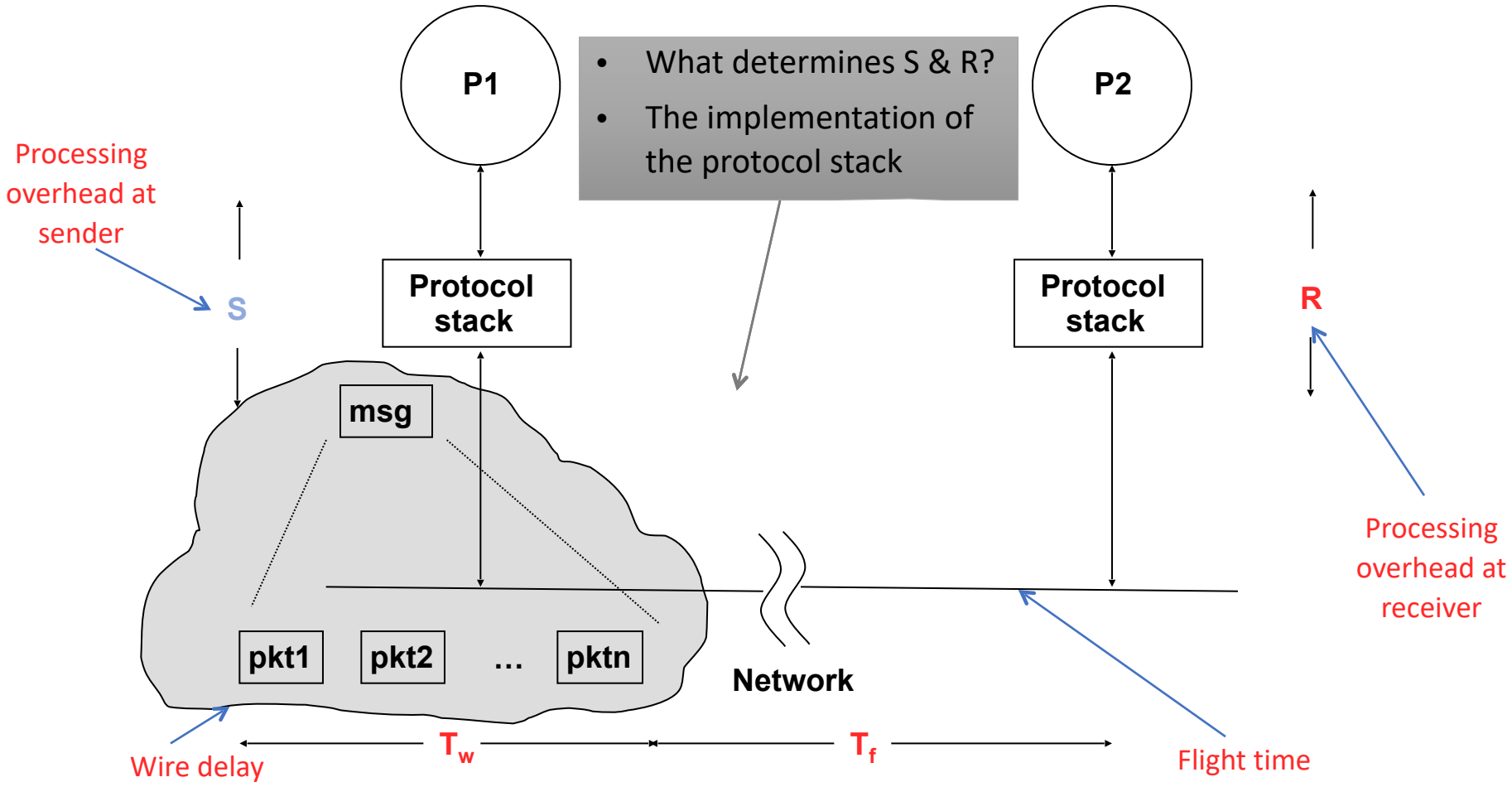
Message throughput = message-size/end-to-end-latency



Performance metrics

Transmission time or end-to-end latency = $S+T_W+T_f+R$

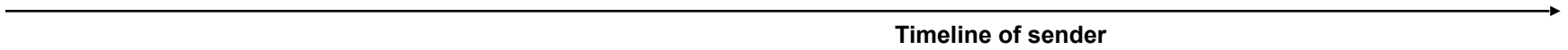
Message throughput = message-size/end-to-end-latency



Timeline example

100ms latency (flight time)
10 packets
Window size = 5

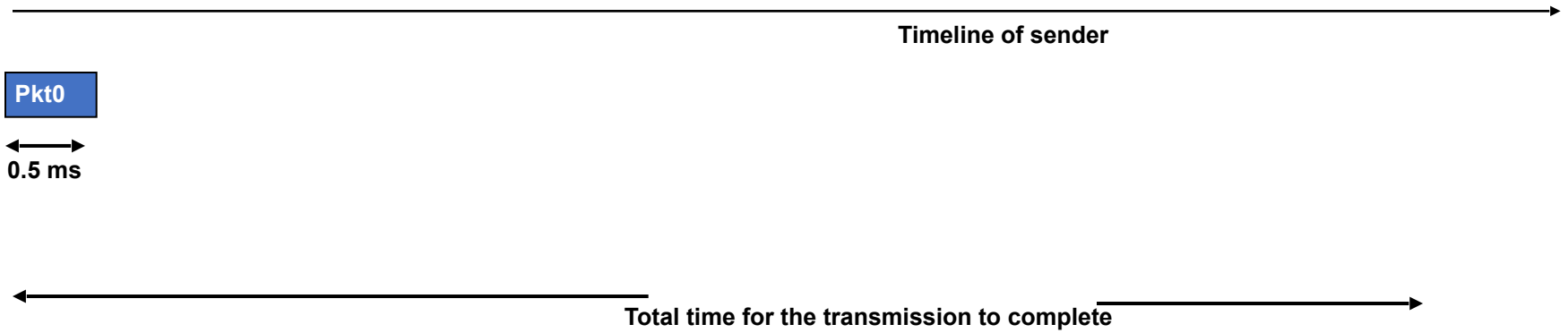
Wire delay (data) = 0.5ms
Wire delay (ack) = ~0
Receiver & Sender overhead = ~0



Timeline example

100ms latency (flight time)
10 packets
Window size = 5

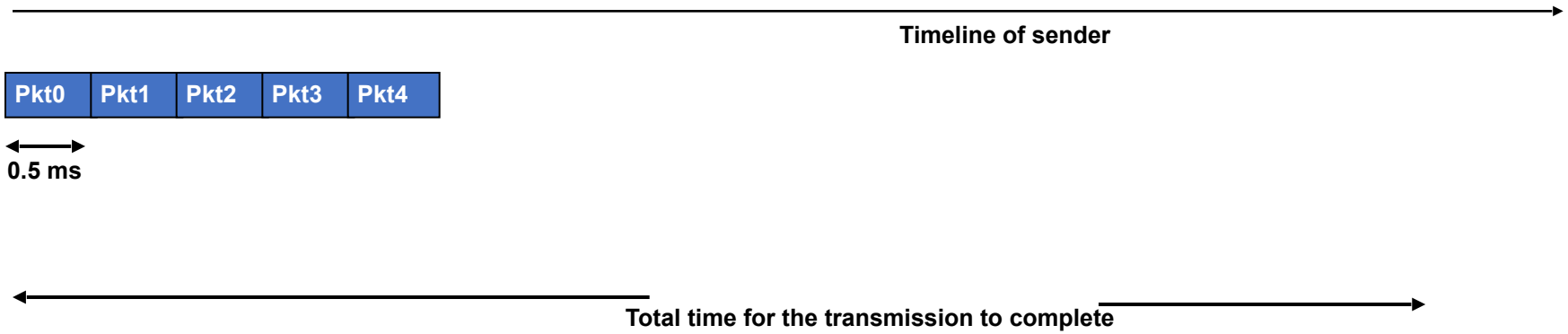
Wire delay (data) = 0.5ms
Wire delay (ack) = ~0
Receiver & Sender overhead = ~0



Timeline example

100ms latency (flight time)
10 packets
Window size = 5

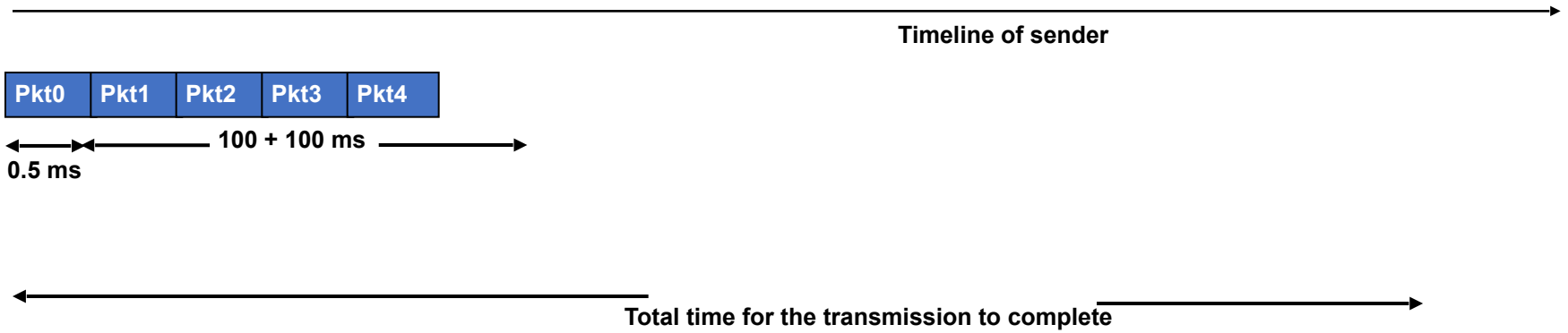
Wire delay (data) = 0.5ms
Wire delay (ack) = ~0
Receiver & Sender overhead = ~0



Timeline example

100ms latency (flight time)
10 packets
Window size = 5

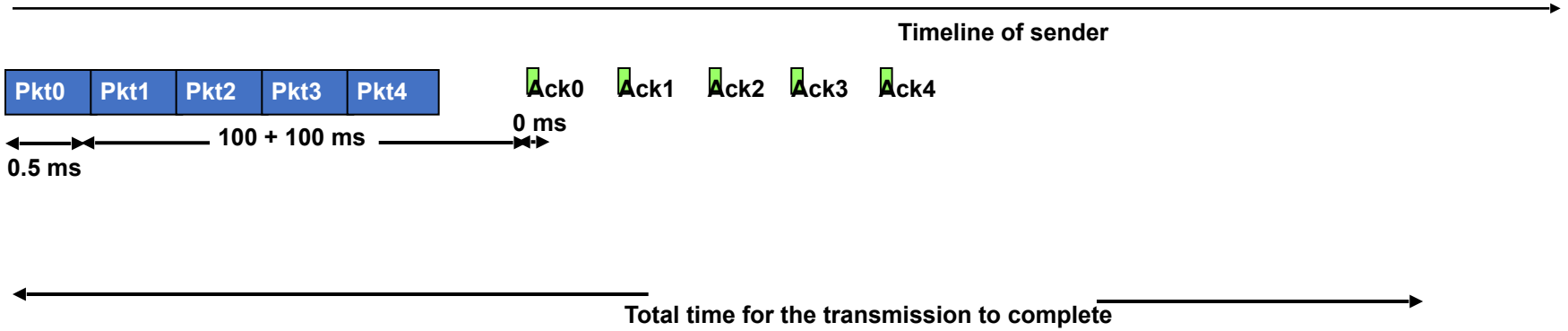
Wire delay (data) = 0.5ms
Wire delay (ack) = ~0
Receiver & Sender overhead = ~0



Timeline example

100ms latency (flight time)
10 packets
Window size = 5

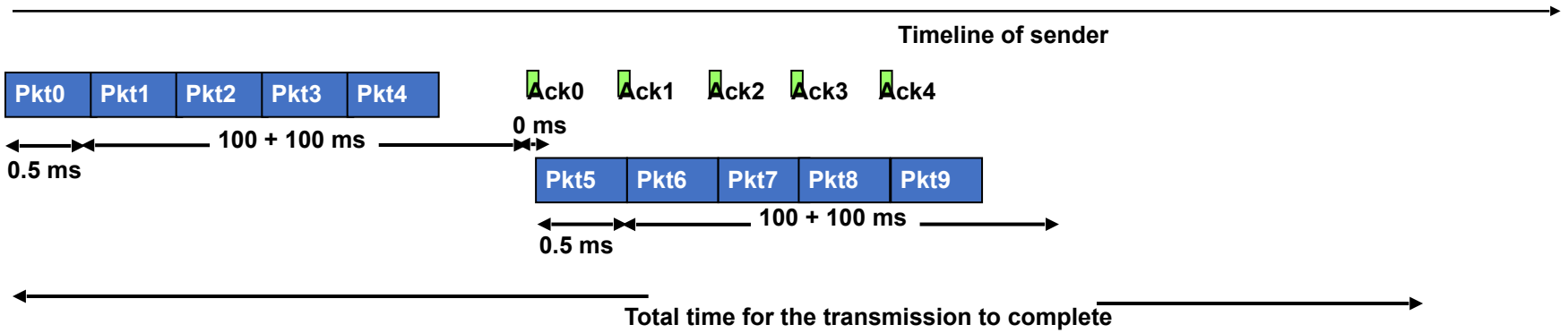
Wire delay (data) = 0.5ms
Wire delay (ack) = ~0
Receiver & Sender overhead = ~0



Timeline example

100ms latency (flight time)
10 packets
Window size = 5

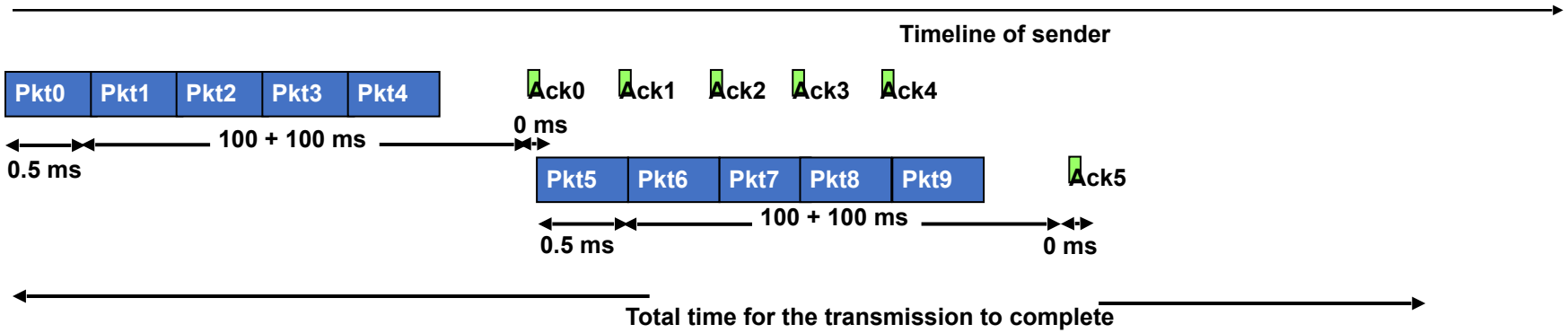
Wire delay (data) = 0.5ms
Wire delay (ack) = ~0
Receiver & Sender overhead = ~0



Timeline example

100ms latency (flight time)
10 packets
Window size = 5

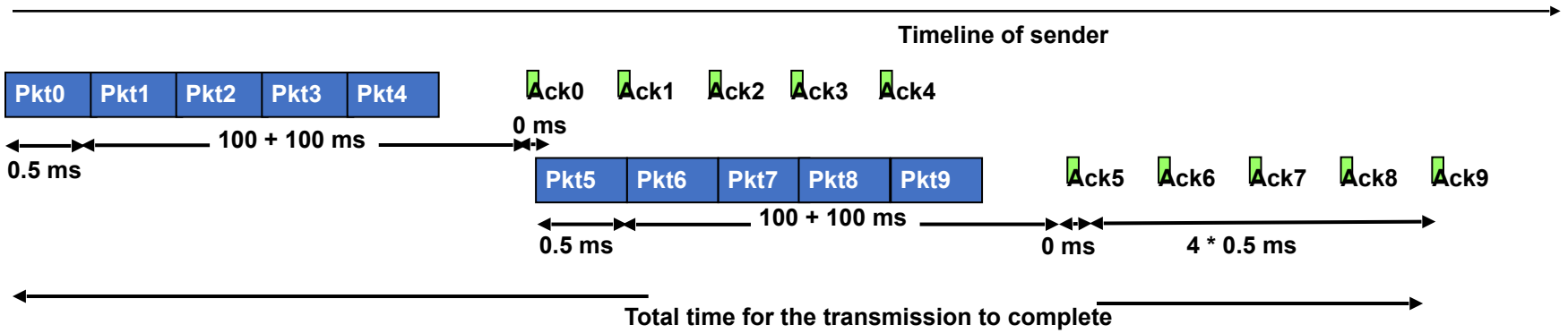
Wire delay (data) = 0.5ms
Wire delay (ack) = ~0
Receiver & Sender overhead = ~0



Timeline example

100ms latency (flight time)
10 packets
Window size = 5

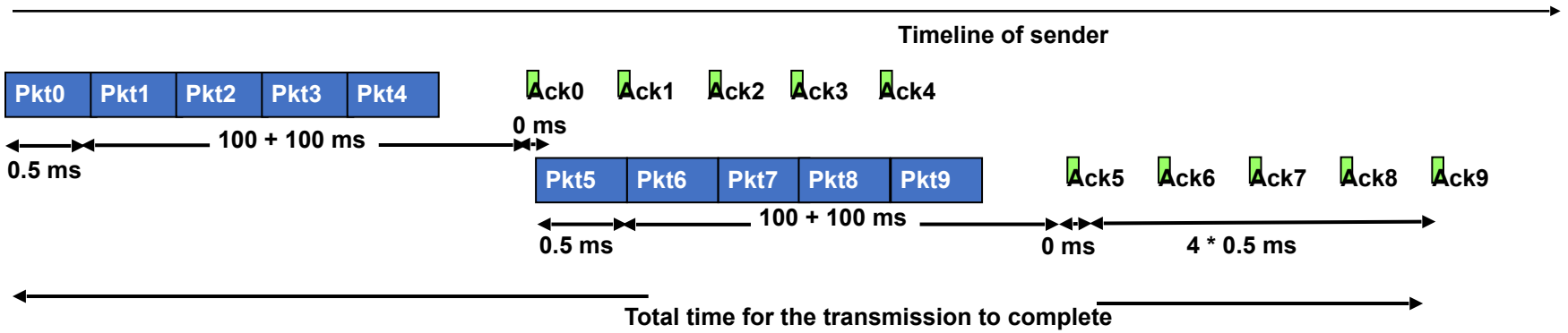
Wire delay (data) = 0.5ms
Wire delay (ack) = ~0
Receiver & Sender overhead = ~0



Timeline example

100ms latency (flight time)
10 packets
Window size = 5

Wire delay (data) = 0.5ms
Wire delay (ack) = ~0
Receiver & Sender overhead = ~0

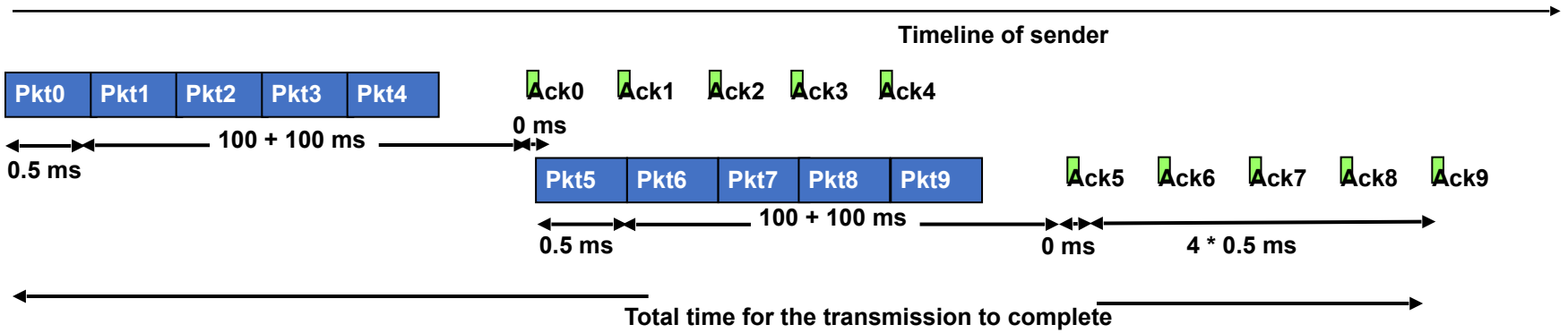


End-to-end latency for Pkt0
$= S + T_w + T_f + R$

Timeline example

100ms latency (flight time)
10 packets
Window size = 5

Wire delay (data) = 0.5ms
Wire delay (ack) = ~0
Receiver & Sender overhead = ~0



End-to-end latency for Pkt0

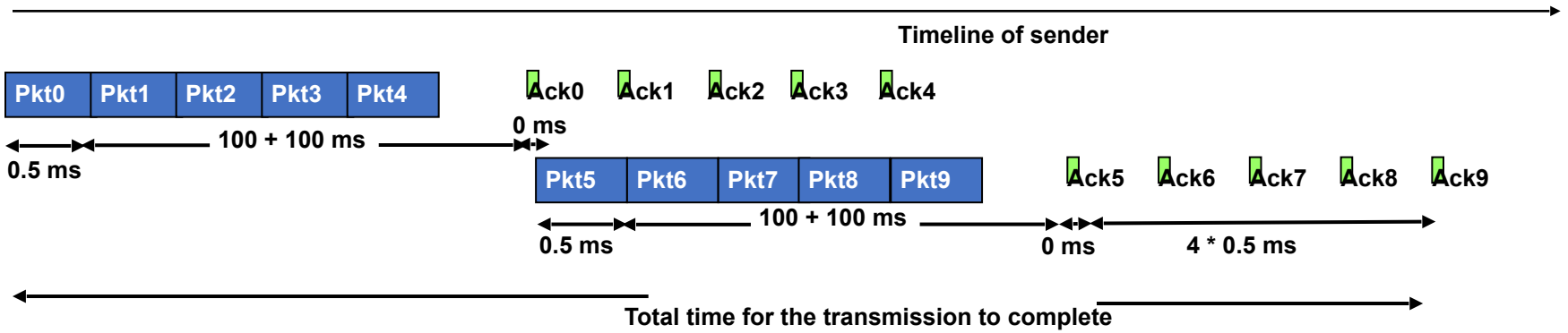
$= S + T_w + T_f + R$

$0 + 0.5 + 100 + 0$

Timeline example

100ms latency (flight time)
10 packets
Window size = 5

Wire delay (data) = 0.5ms
Wire delay (ack) = ~0
Receiver & Sender overhead = ~0



End-to-end latency for Pkt0

$= S + T_w + T_f + R$

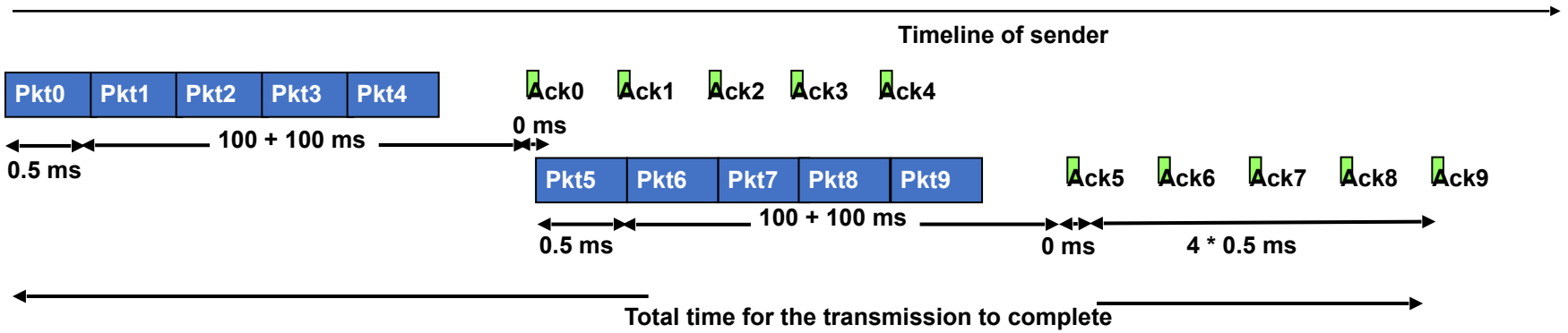
$0 + 0.5 + 100 + 0$

100.5 ms

Timeline example

100ms latency (flight time)
10 packets
Window size = 5

Wire delay (data) = 0.5ms
Wire delay (ack) = ~0
Receiver & Sender overhead = ~0



End-to-end latency for Pkt0

$= S + T_w + T_f + R$

$0 + 0.5 + 100 + 0$

100.5 ms

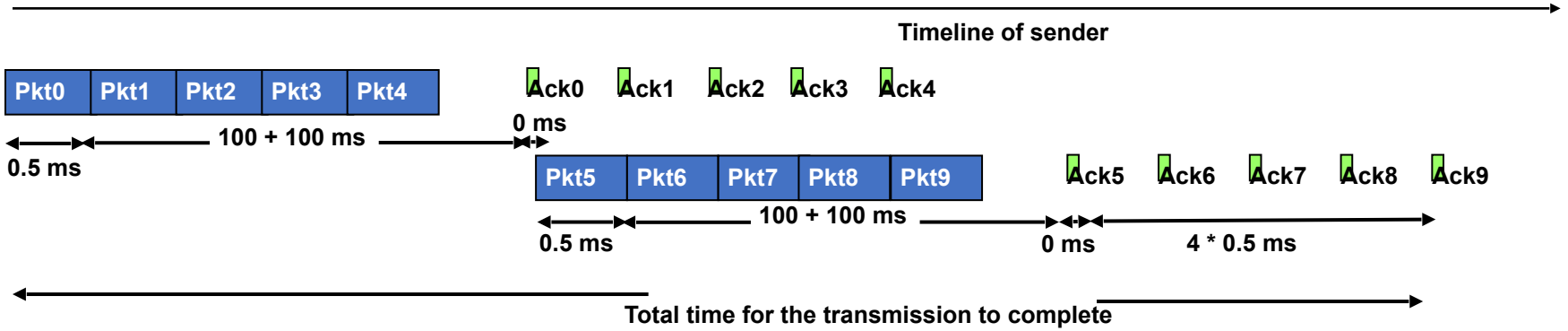
End-to-end latency for Ack0

$= S + T_w + T_f + R$

Timeline example

100ms latency (flight time)
10 packets
Window size = 5

Wire delay (data) = 0.5ms
Wire delay (ack) = ~0
Receiver & Sender overhead = ~0



End-to-end latency for Pkt0

$= S + T_w + T_f + R$

$0 + 0.5 + 100 + 0$

100.5 ms

End-to-end latency for Ack0

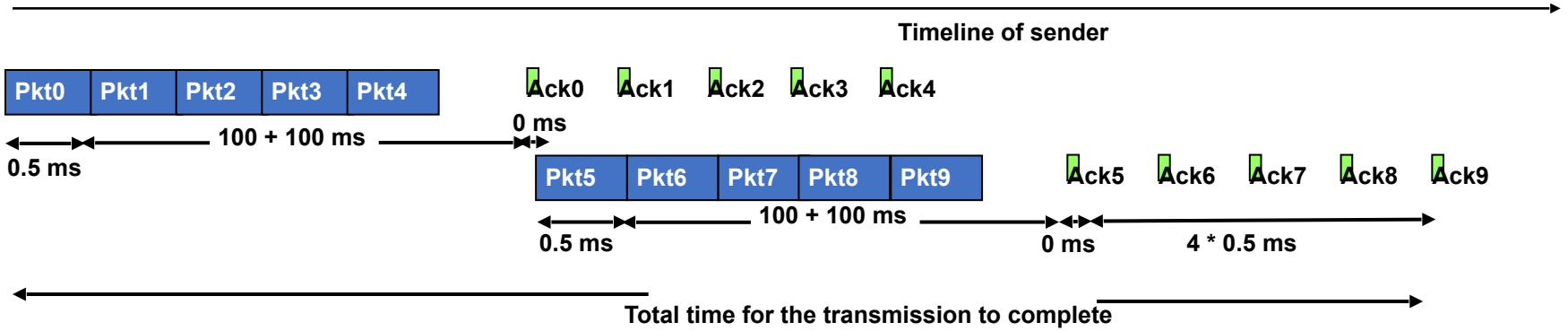
$= S + T_w + T_f + R$

$0 + 0.0 + 100 + 0$

Timeline example

100ms latency (flight time)
10 packets
Window size = 5

Wire delay (data) = 0.5ms
Wire delay (ack) = ~0
Receiver & Sender overhead = ~0



End-to-end latency for Pkt0

$= S + T_w + T_f + R$

$0 + 0.5 + 100 + 0$

100.5 ms

End-to-end latency for Ack0

$= S + T_w + T_f + R$

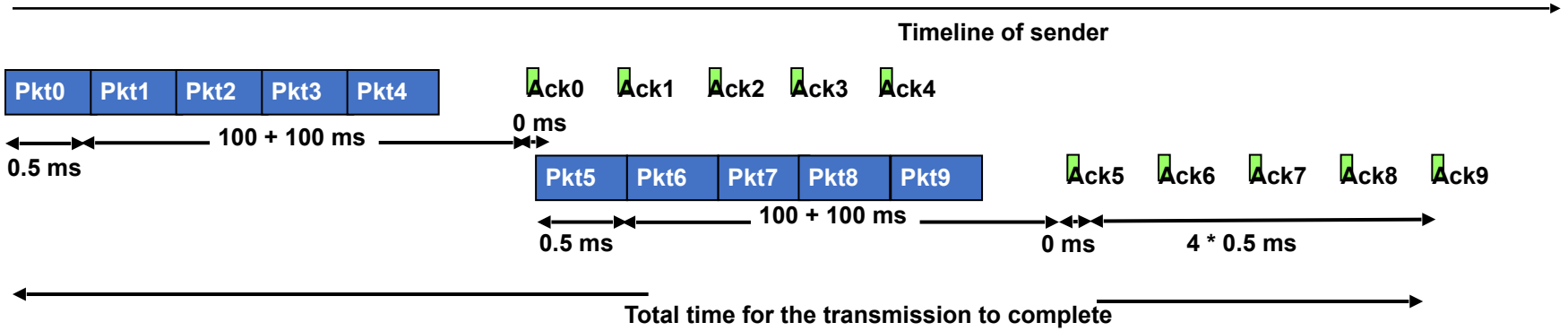
$0 + 0.0 + 100 + 0$

100.0 ms

Timeline example

100ms latency (flight time)
10 packets
Window size = 5

Wire delay (data) = 0.5ms
Wire delay (ack) = ~0
Receiver & Sender overhead = ~0



End-to-end latency for Pkt0
= $S + T_w + T_f + R$
0 + 0.5 + 100 + 0
100.5 ms

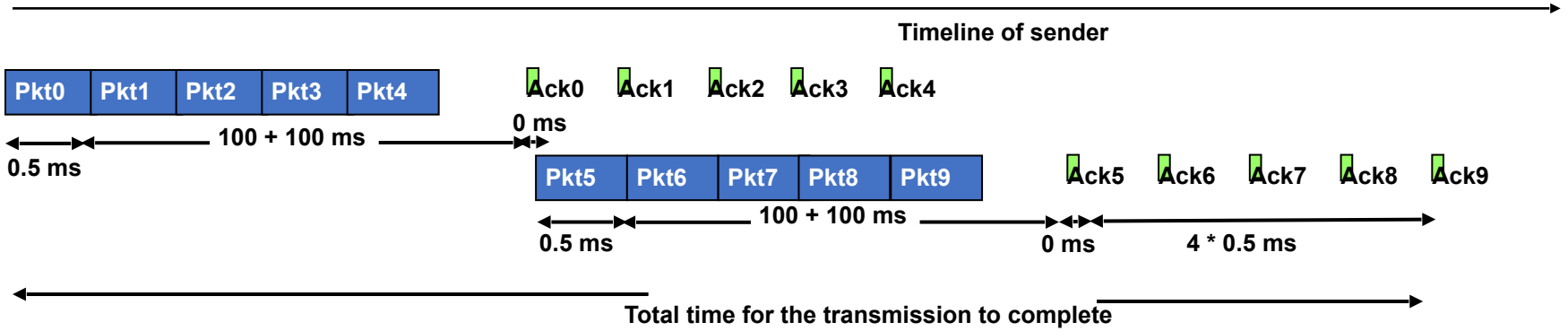
End-to-end latency for Ack0
= $S + T_w + T_f + R$
0 + 0.0 + 100 + 0
100.0 ms

Total transmission time

Timeline example

100ms latency (flight time)
10 packets
Window size = 5

Wire delay (data) = 0.5ms
Wire delay (ack) = ~0
Receiver & Sender overhead = ~0



End-to-end latency for Pkt0
= $S + T_w + T_f + R$
0 + 0.5 + 100 + 0
100.5 ms

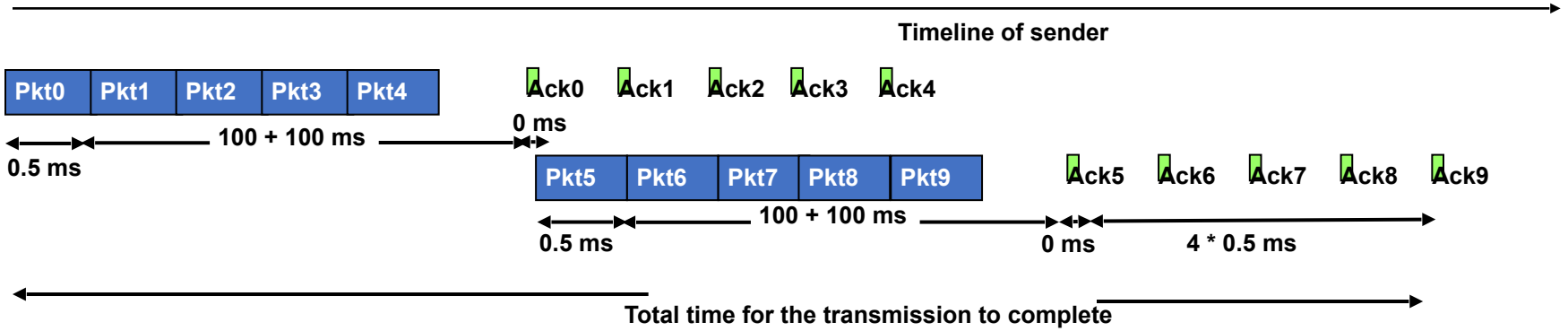
End-to-end latency for Ack0
= $S + T_w + T_f + R$
0 + 0.0 + 100 + 0
100.0 ms

Total transmission time
= 0.5 + 100 + 100 + 0 + 0.5 + 100 + 100 + 0.5 + 0.5 + 0.5 + 0.5

Timeline example

100ms latency (flight time)
10 packets
Window size = 5

Wire delay (data) = 0.5ms
Wire delay (ack) = ~0
Receiver & Sender overhead = ~0



End-to-end latency for Pkt0
= $S + T_w + T_f + R$
0 + 0.5 + 100 + 0
100.5 ms

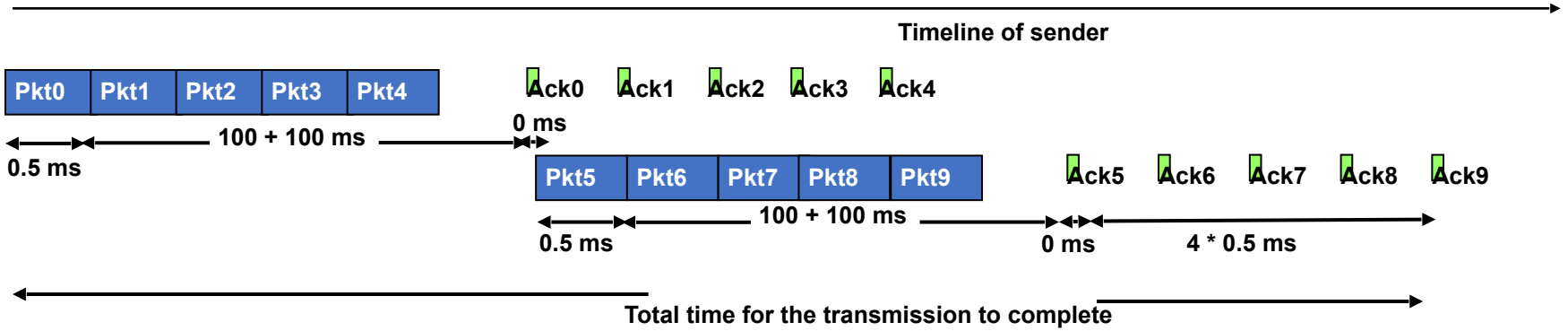
End-to-end latency for Ack0
= $S + T_w + T_f + R$
0 + 0.0 + 100 + 0
100.0 ms

Total transmission time
= 0.5 + 100 + 100 + 0 + 0.5 + 100 + 100 + 0.5 + 0.5 + 0.5 + 0.5
0.5 + 200 + 0.5 + 200 + 4 * 0.5

Timeline example

100ms latency (flight time)
10 packets
Window size = 5

Wire delay (data) = 0.5ms
Wire delay (ack) = ~0
Receiver & Sender overhead = ~0



End-to-end latency for Pkt0
= $S + T_w + T_f + R$
0 + 0.5 + 100 + 0
100.5 ms

End-to-end latency for Ack0
= $S + T_w + T_f + R$
0 + 0.0 + 100 + 0
100.0 ms

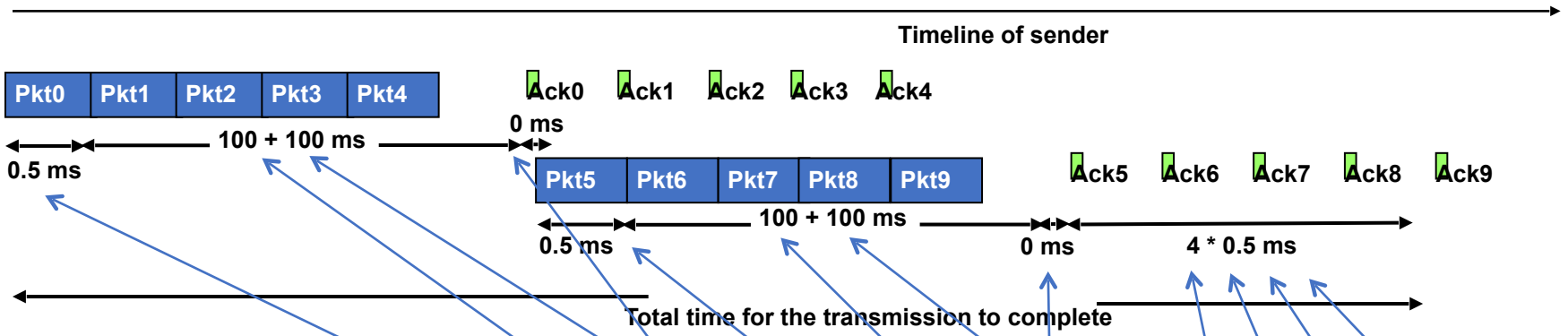
Total transmission time
= 0.5 + 100 + 100 + 0 + 0.5 + 100 + 100 + 0.5 + 0.5 + 0.5 + 0.5
0.5 + 200 + 0.5 + 200 + 4 * 0.5
403 ms

Timeline example

Upcoming: dag2200

100ms latency
10 packets
Window size = 5

Wire delay (data) = 0.5ms
Wire delay (ack) = ~0
Receiver & Sender overhead = ~0



End-to-end latency for Pkt0
= $S + T_w + T_f + R$
0 + 0.5 + 100 + 0
100.5 ms

End-to-end latency for Ack0
= $S + T_w + T_f + R$
0 + 0.0 + 100 + 0
100.0 ms

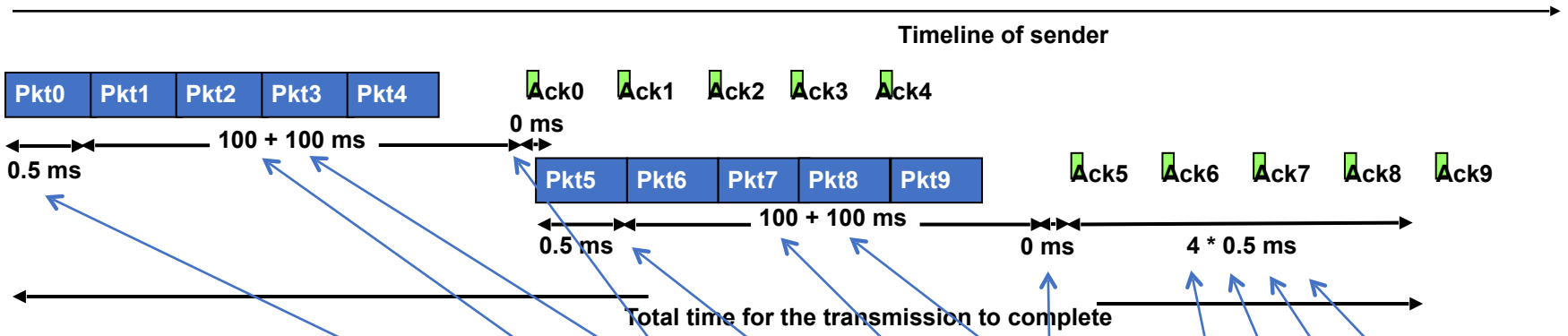
Total transmission time
= 0.5 + 100 + 0 + 100 + 0.5 + 100 + 0 + 100 + 0.5 + 0.5 + 0.5 + 0.5
0.5 + 200 + 0.5 + 200 + 4 * 0.5
403 ms

Timeline example

Upcoming: dag2200

100ms latency
10 packets
Window size = 5

Wire delay (data) = 0.5ms
Wire delay (ack) = ~0
Receiver & Sender overhead = ~0



End-to-end latency for Pkt0
= $S + T_w + T_f + R$
0 + 0.5 + 100 + 0
100.5 ms

End-to-end latency for Ack0
= $S + T_w + T_f + R$
0 + 0.0 + 100 + 0
100.0 ms

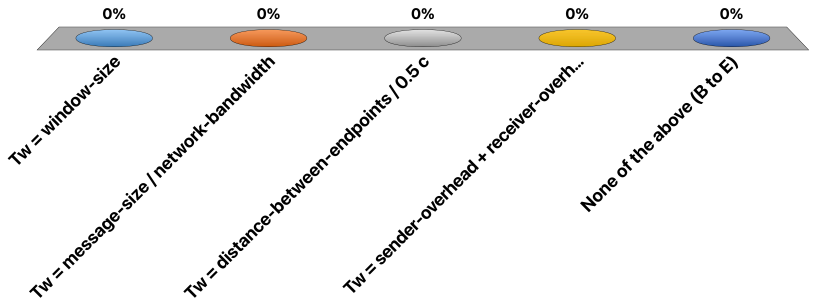
Total transmission time
= 0.5 + 100 + 0 + 100 + 0.5 + 100 + 0 + 100 + 0.5 + 0.5 + 0.5 + 0.5
0.5 + 200 + 0.5 + 200 + 4 * 0.5
403 ms

Repeats

Transmission tail

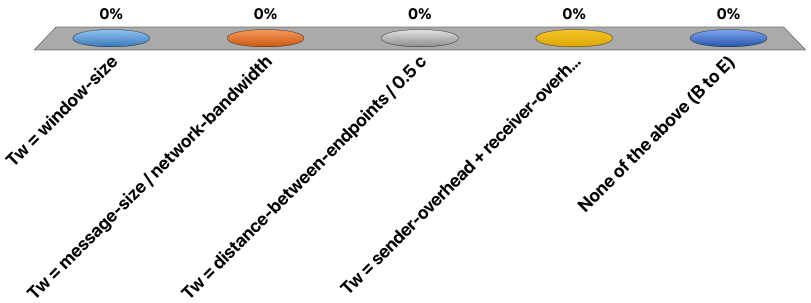
In the expression $S + T_w + T_f + R$

A. T_w = window-size
B. T_w = message-size / network-bandwidth
C. T_w = distance-between-endpoints / 0.5 *c*
D. T_w = sender-overhead + receiver-overhead
E. None of the above (B to E)



In the expression $S + T_w + T_f + R$

A. T_w = window-size
B. T_w = message-size / network-bandwidth
C. T_w = distance-between-endpoints / 0.5 *c* T_f
D. T_w = sender-overhead + receiver-overhead S+R
E. None of the above (B to E)



Examples

Assuming no loss, how many packets to transmit the message given?

- Message size = 100,000 bytes, header size per packet = 100 bytes, packet size = 1100 bytes

Examples

Assuming no loss, how many packets to transmit the message given?

- Message size = 100,000 bytes, header size per packet = 100 bytes, packet size = 1100 bytes
  - 100 packets

Examples

Assuming no loss, how many packets to transmit the message given?

- Message size = 100,000 bytes, header size per packet = 100 bytes, packet size = 1100 bytes
  - 100 packets

Assuming 10% loss, what is the number of packets to transmit the message?

Examples

Assuming no loss, how many packets to transmit the message given?

- Message size = 100,000 bytes, header size per packet = 100 bytes, packet size = 1100 bytes
  - 100 packets

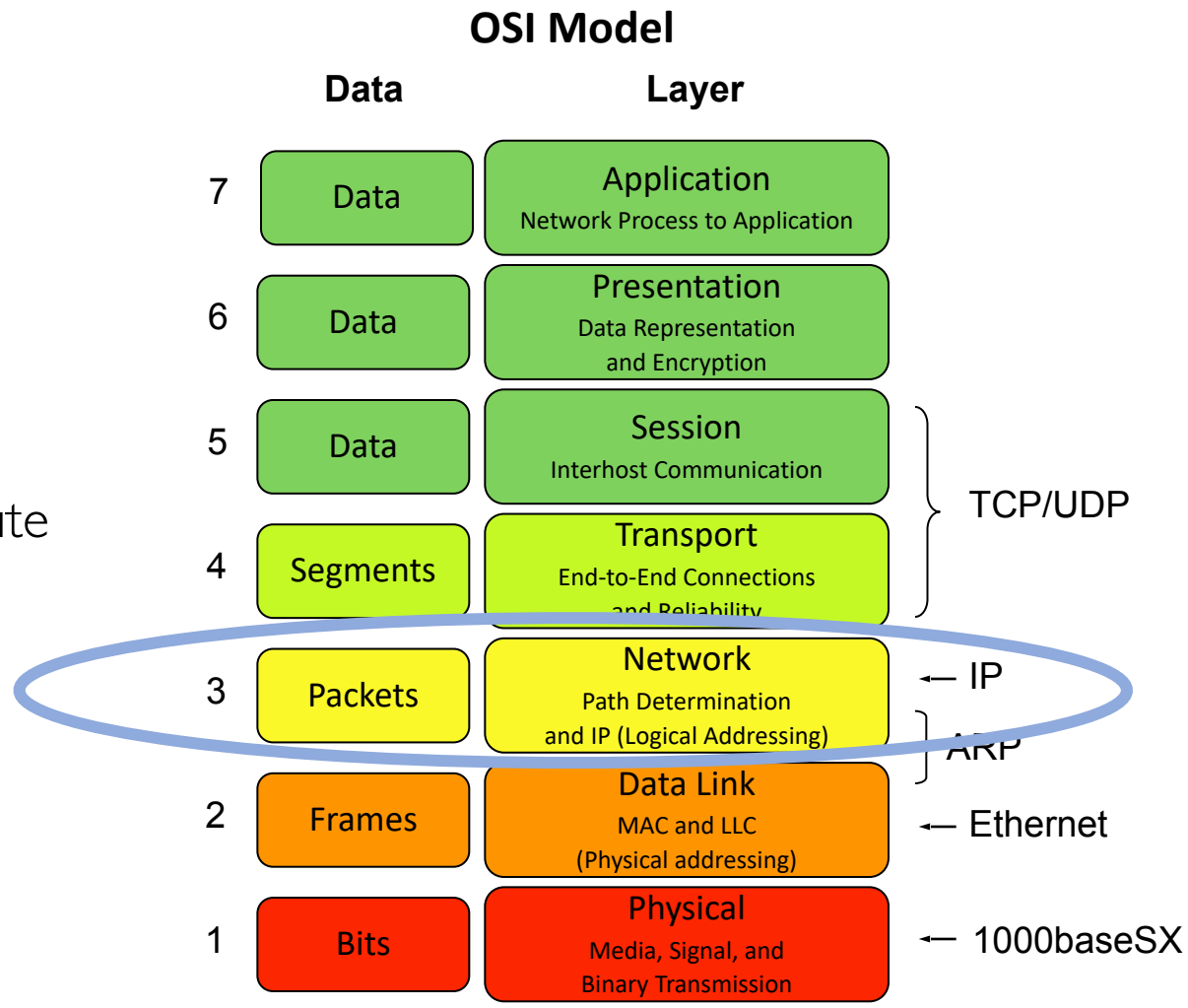
Assuming 10% loss, what is the number of packets to transmit the message?

- 111
  - 100 + 10 + 1

More advanced performance estimation examples in the book!

Back to the network layer

- Why are we back here?
- Recall, IP routing happens here
- We didn't answer a question:
  How does a router know where to route (i.e. who sets up the routing table)?

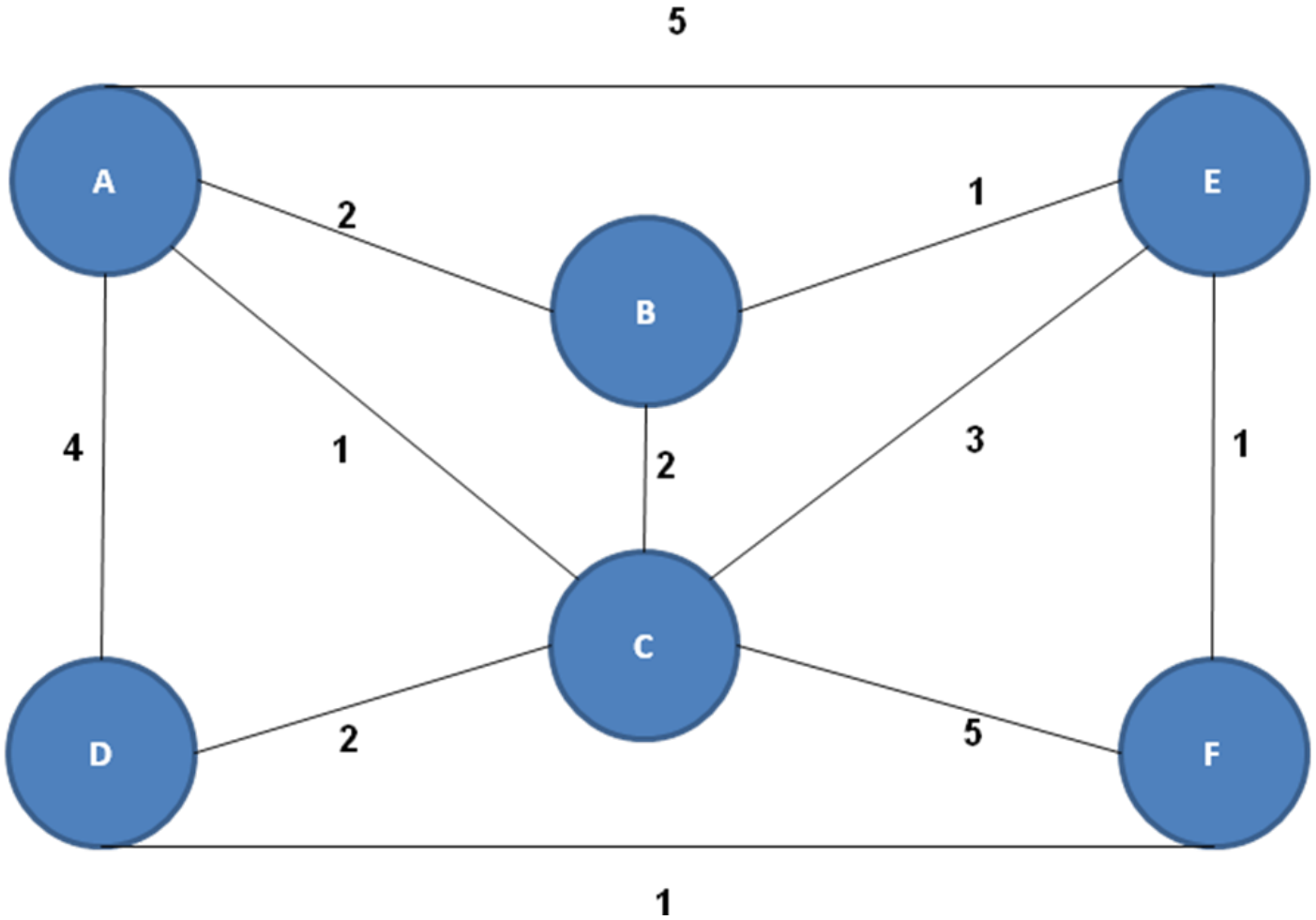


Network layer

- Routing information protocols
  - Link state protocols
  - Distance vector protocols
  - Hierarchical routing
- Addressing

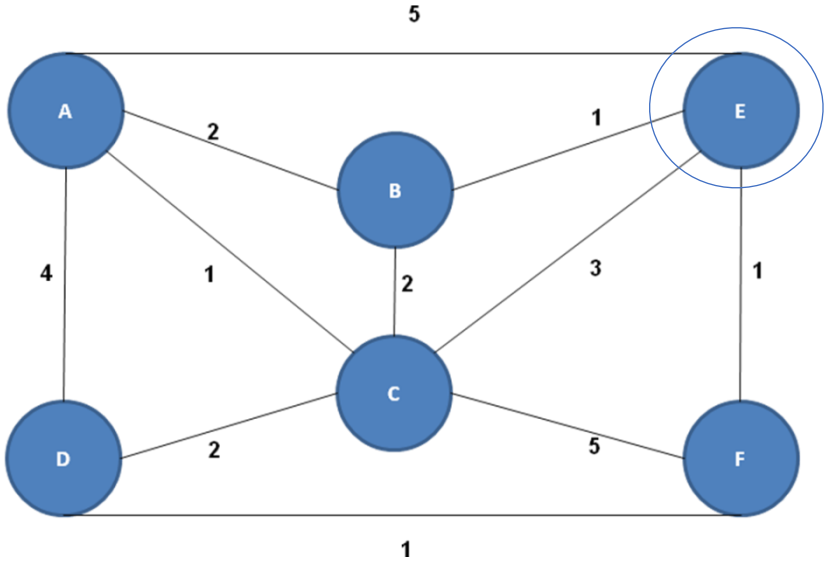
Intuition behind routing algorithms (link state, distance vector)

What's the latency to get to your neighbor?



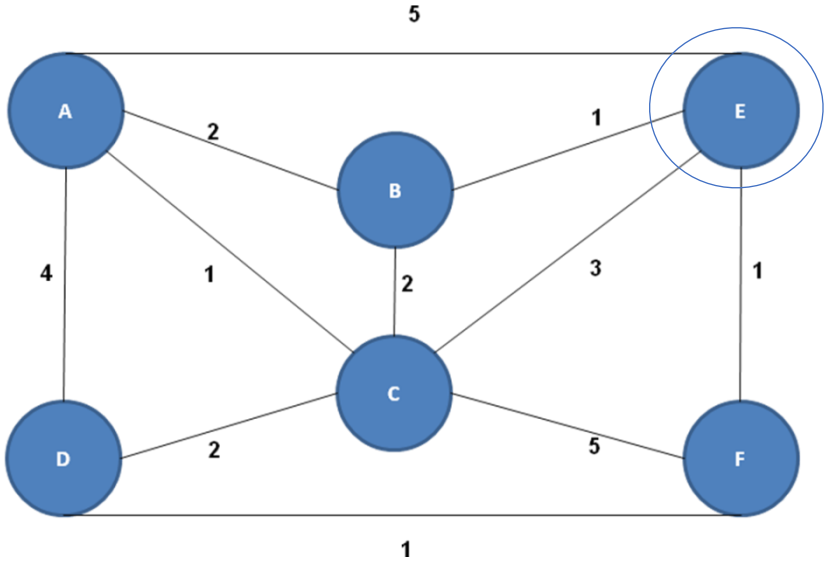
Distance vector in action

- Uses neighbor info, local algorithm (i.e. every node only sees its neighbors)
- Each node knows the cost to each of its neighbors
- Each node sends its idea of routing to each of its neighbors
- Known colloquially as "routing by rumor"
- For every other node, each node determines the lowest cost next-hop
- Algorithm runs simultaneously on every node
- How does it work?



Distance vector in action

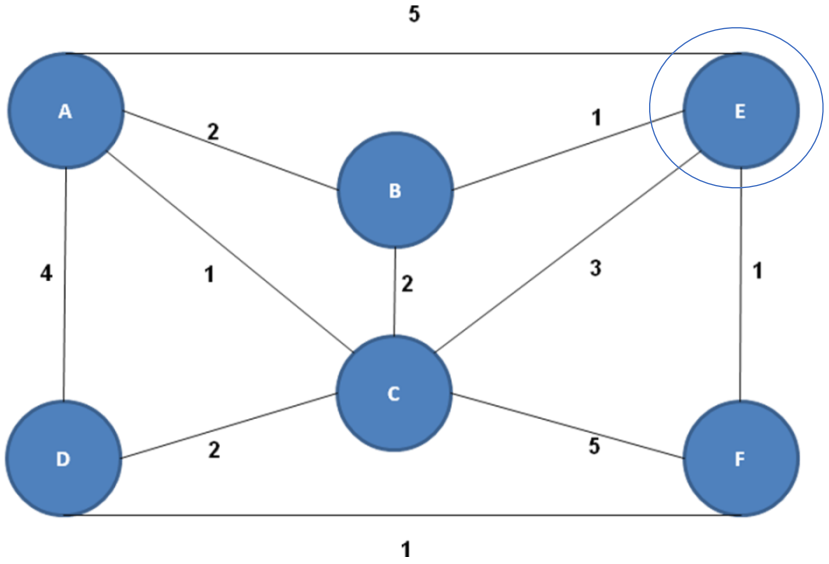
- Uses neighbor info, local algorithm (i.e. every node only sees its neighbors)
- Each node knows the cost to each of its neighbors
- Each node sends its idea of routing to each of its neighbors
- Known colloquially as "routing by rumor"
- For every other node, each node determines the lowest cost next-hop
- Algorithm runs simultaneously on every node
- How does it work?
- Node E knows its costs to its neighbors:



Routes from E	A	B	C	D	F
Cost/next hop	5/A	1/B	3/C	∞/	1/F

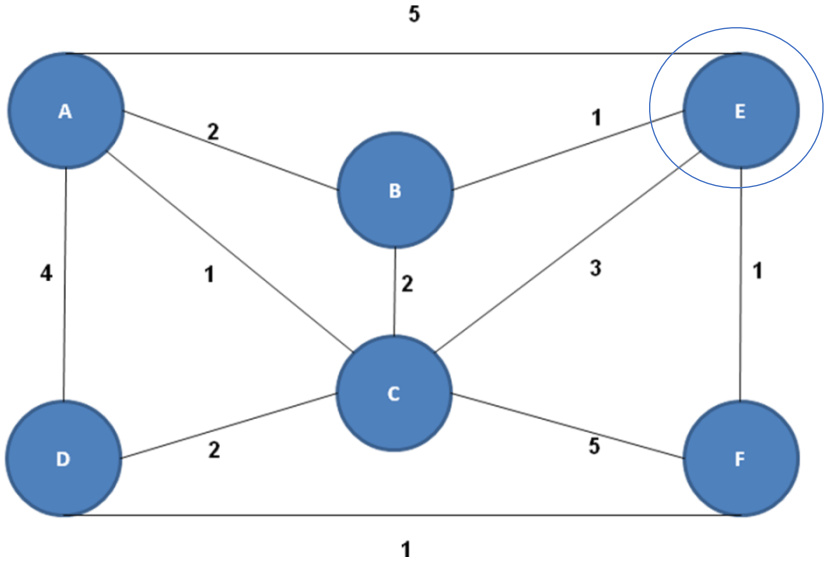
Distance vector in action

Routes from E	A	B	C	D	F
Cost/next hop	5/A	1/B	3/C	∞/	1/F



Distance vector in action

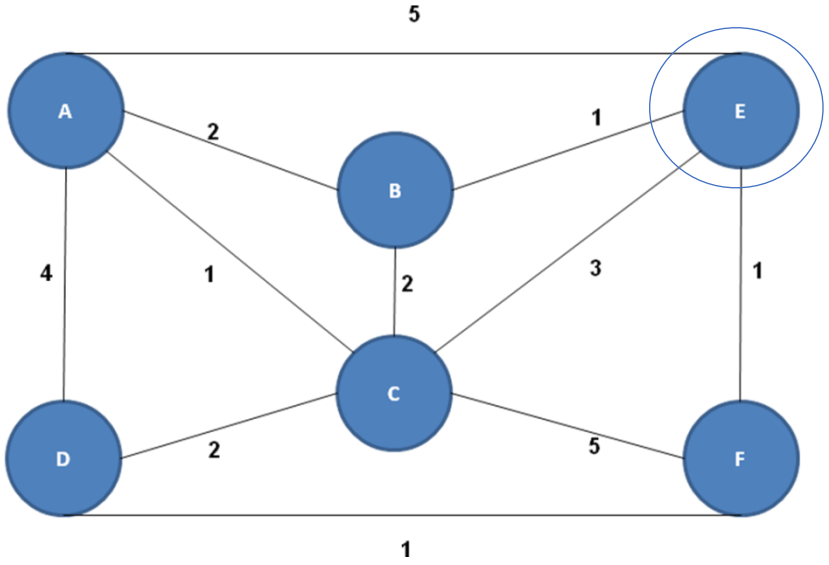
Routes from E	A	B	C	D	F
Cost/next hop	5/A	1/B	3/C	∞/	1/F



- Node E listens to messages from its neighbors:
  - A says its routes are 0/A, 5/E, 2/B, 1/C, 4/D, 5/F
  - B says its routes are 0/B, 1/E, 2/C, 2/A, 5/F, 4/D
  - C says its routes are 0/C, 1/A, 2/D, 5/F, 3/E, 2/B
  - F says its routes are 0/F, 5/C, 1/D, 1/E, 2/B, 4/A
  - D isn't heard because it isn't connected to E

Distance vector in action

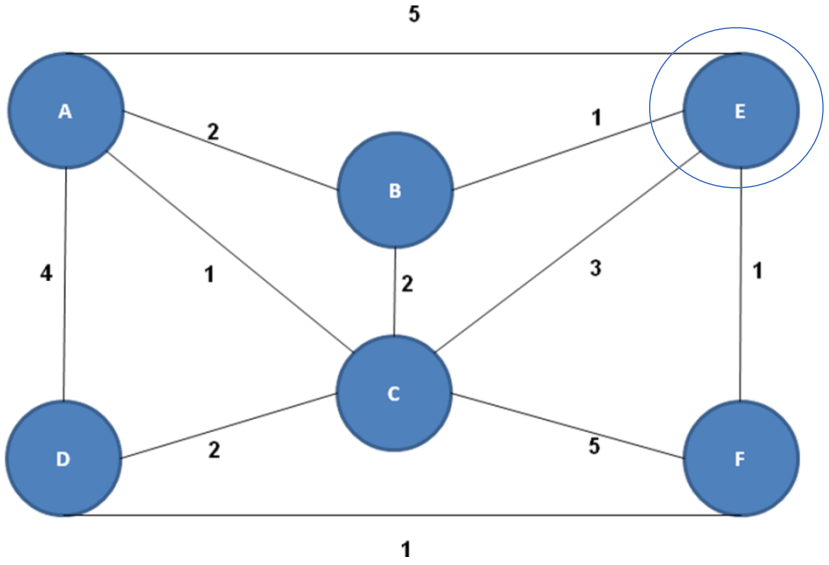
Routes from E	A	B	C	D	F
Cost/next hop	5/A	1/B	3/C	∞/	1/F



- Node E listens to messages from its neighbors:
  - A says its routes are 0/A, 5/E, 2/B, 1/C, 4/D, 5/F
  - B says its routes are 0/B, 1/E, 2/C, 2/A, 5/F, 4/D
  - C says its routes are 0/C, 1/A, 2/D, 5/F, 3/E, 2/B
  - F says its routes are 0/F, 5/C, 1/D, 1/E, 2/B, 4/A
  - D isn't heard because it isn't connected to E
- Node E looks at each route from each neighbor N; it adds its cost to get to N to N's cost to get to each other node.

Distance vector in action

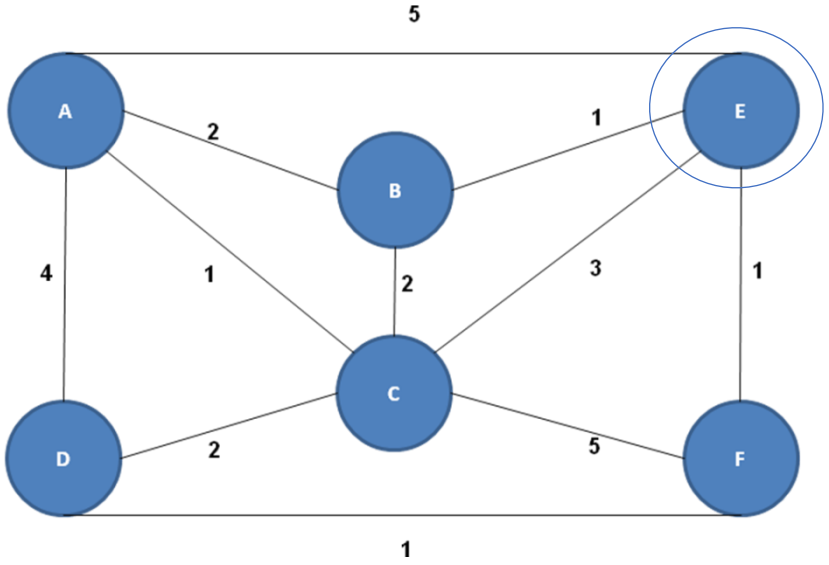
Routes from E	A	B	C	D	F
Cost/next hop	5/A	1/B	3/C	∞/	1/F



- Node E listens to messages from its neighbors:
  - A says its routes are 0/A, 5/E, 2/B, 1/C, 4/D, 5/F
  - B says its routes are 0/B, 1/E, 2/C, 2/A, 5/F, 4/D
  - C says its routes are 0/C, 1/A, 2/D, 5/F, 3/E, 2/B
  - F says its routes are 0/F, 5/C, 1/D, 1/E, 2/B, 4/A
  - D isn't heard because it isn't connected to E
- Node E looks at each route from each neighbor N; it adds its cost to get to N to N's cost to get to each other node.
- For each node, if the sum is lower cost than what E has in its route table, E changes its own route to run through the neighbor

Distance vector in action

Routes from E	A	B	C	D	F
Cost/next hop	5/A	1/B	3/C	∞/	1/F

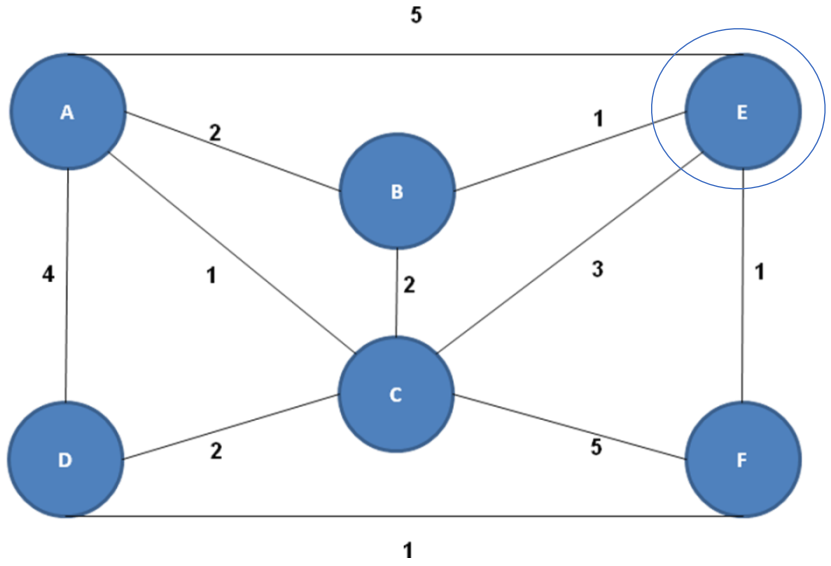


- Node E listens to messages from its neighbors:
  - A says its routes are 0/A, 5/E, 2/B, 1/C, 4/D, 5/F
  - B says its routes are 0/B, 1/E, 2/C, 2/A, 5/F, 4/D
  - C says its routes are 0/C, 1/A, 2/D, 5/F, 3/E, 2/B
  - F says its routes are 0/F, 5/C, 1/D, 1/E, 2/B, 4/A
  - D isn't heard because it isn't connected to E
- Node E looks at each route from each neighbor N; it adds its cost to get to N to N's cost to get to each other node.
- For each node, if the sum is lower cost than what E has in its route table, E changes its own route to run through the neighbor

- Reviewing node A's message
  - 5+2/B, 5+1/C, 5+4/D, 5+5/F
  - Only 9/D is less
  - We'll set the next hop to A

Distance vector in action

Routes from E	A	B	C	D	F
Cost/next hop	5/A	1/B	3/C	∞/	1/F

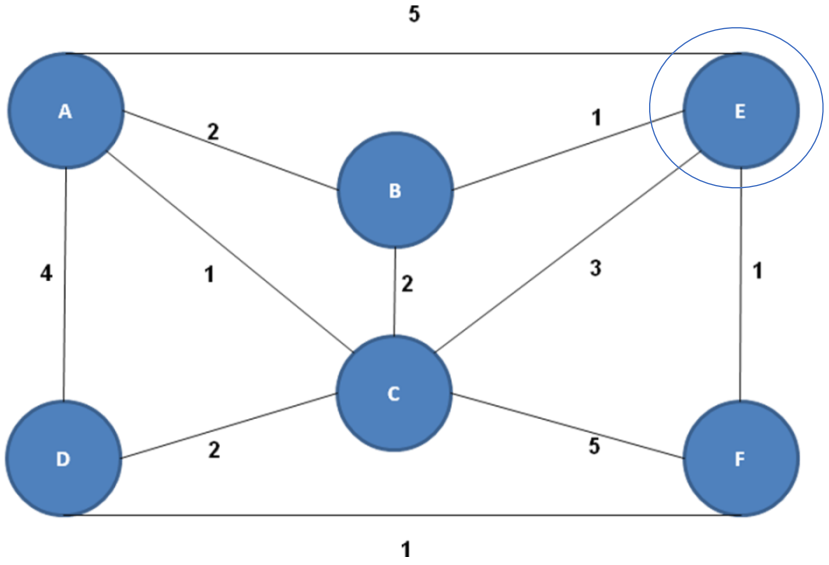


- Node E listens to messages from its neighbors:
  - A says its routes are 0/A, 5/E, 2/B, 1/C, 4/D, 5/F
  - B says its routes are 0/B, 1/E, 2/C, 2/A, 5/F, 4/D
  - C says its routes are 0/C, 1/A, 2/D, 5/F, 3/E, 2/B
  - F says its routes are 0/F, 5/C, 1/D, 1/E, 2/B, 4/A
  - D isn't heard because it isn't connected to E
- Node E looks at each route from each neighbor N; it adds its cost to get to N to N's cost to get to each other node.
- For each node, if the sum is lower cost than what E has in its route table, E changes its own route to run through the neighbor

- Reviewing node A's message
  - 5+2/B, 5+1/C, 5+4/D, 5+5/F
  - Only 9/D is less
  - We'll set the next hop to A

Distance vector in action

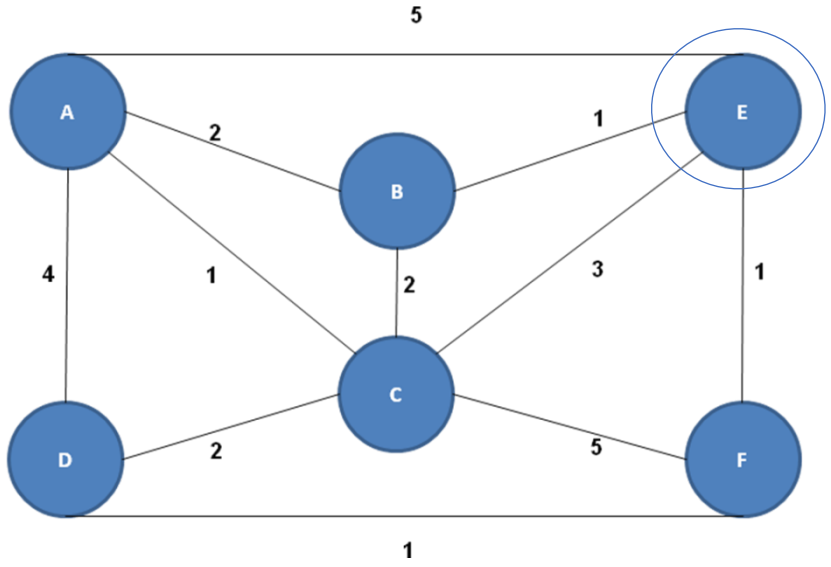
Routes from E	A	B	C	D	F
Cost/next hop	5/A	1/B	3/C	9/A	1/F



- Node E listens to messages from its neighbors:
  - A says its routes are 0/A, 5/E, 2/B, 1/C, 4/D, 5/F
  - B says its routes are 0/B, 1/E, 2/C, 2/A, 5/F, 4/D
  - C says its routes are 0/C, 1/A, 2/D, 5/F, 3/E, 2/B
  - F says its routes are 0/F, 5/C, 1/D, 1/E, 2/B, 4/A
  - D isn't heard because it isn't connected to E
- Node E looks at each route from each neighbor N; it adds its cost to get to N to N's cost to get to each other node.
- For each node, if the sum is lower cost than what E has in its route table, E changes its own route to run through the neighbor

Distance vector in action

Routes from E	A	B	C	D	F
Cost/next hop	5/A	1/B	3/C	9/A	1/F

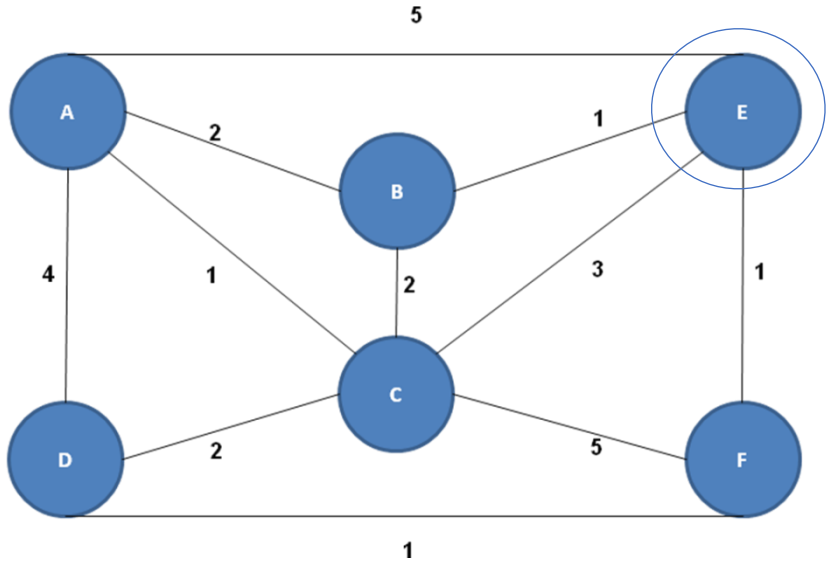


- Node E listens to messages from its neighbors:
  - A says its routes are 0/A, 5/E, 2/B, 1/C, 4/D, 5/F
  - B says its routes are 0/B, 1/E, 2/C, 2/A, 5/F, 4/D
  - C says its routes are 0/C, 1/A, 2/D, 5/F, 3/E, 2/B
  - F says its routes are 0/F, 5/C, 1/D, 1/E, 2/B, 4/A
  - D isn't heard because it isn't connected to E
- Node E looks at each route from each neighbor N; it adds its cost to get to N to N's cost to get to each other node.
- For each node, if the sum is lower cost than what E has in its route table, E changes its own route to run through the neighbor

- Reviewing Node B's message

Distance vector in action

Routes from E	A	B	C	D	F
Cost/next hop	5/A	1/B	3/C	9/A	1/F

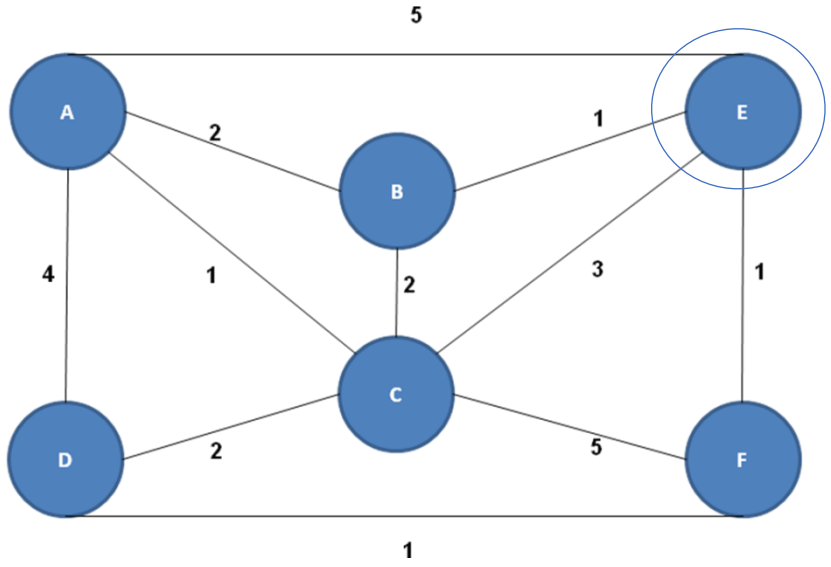


- Node E listens to messages from its neighbors:
  - A says its routes are 0/A, 5/E, 2/B, 1/C, 4/D, 5/F
  - B says its routes are 0/B, 1/E, 2/C, 2/A, 5/F, 4/D
  - C says its routes are 0/C, 1/A, 2/D, 5/F, 3/E, 2/B
  - F says its routes are 0/F, 5/C, 1/D, 1/E, 2/B, 4/A
  - D isn't heard because it isn't connected to E
- Node E looks at each route from each neighbor N; it adds its cost to get to N to N's cost to get to each other node.
- For each node, if the sum is lower cost than what E has in its route table, E changes its own route to run through the neighbor

- Reviewing Node B's message
  - 1+2/C, 1+2/A, 1+5/F, 1+4/D

Distance vector in action

Routes from E	A	B	C	D	F
Cost/next hop	5/A	1/B	3/C	9/A	1/F

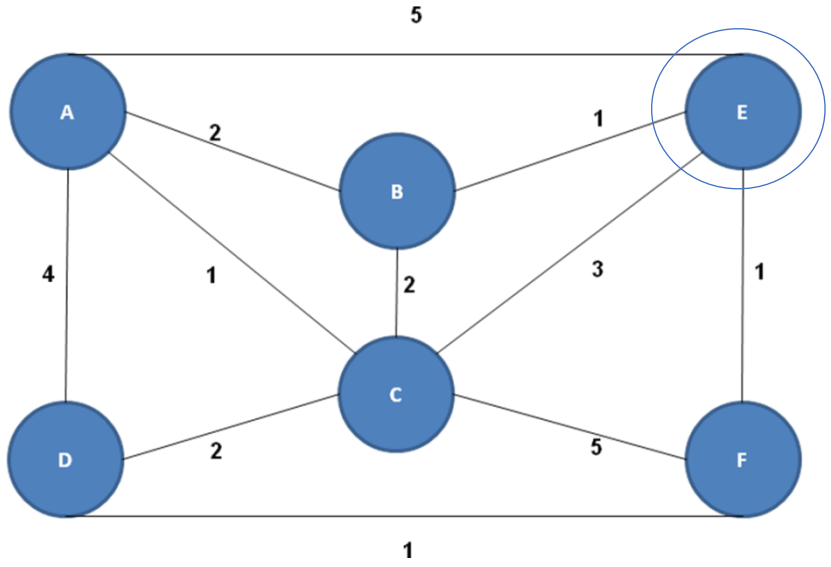


- Node E listens to messages from its neighbors:
  - A says its routes are 0/A, 5/E, 2/B, 1/C, 4/D, 5/F
  - B says its routes are 0/B, 1/E, 2/C, 2/A, 5/F, 4/D
  - C says its routes are 0/C, 1/A, 2/D, 5/F, 3/E, 2/B
  - F says its routes are 0/F, 5/C, 1/D, 1/E, 2/B, 4/A
  - D isn't heard because it isn't connected to E
- Node E looks at each route from each neighbor N; it adds its cost to get to N to N's cost to get to each other node.
- For each node, if the sum is lower cost than what E has in its route table, E changes its own route to run through the neighbor

- Reviewing Node B's message
  - 1+2/C, 1+2/A, 1+5/F, 1+4/D
  - 3/A and 5/D are lower cost

Distance vector in action

Routes from E	A	B	C	D	F
Cost/next hop	5/A	1/B	3/C	9/A	1/F

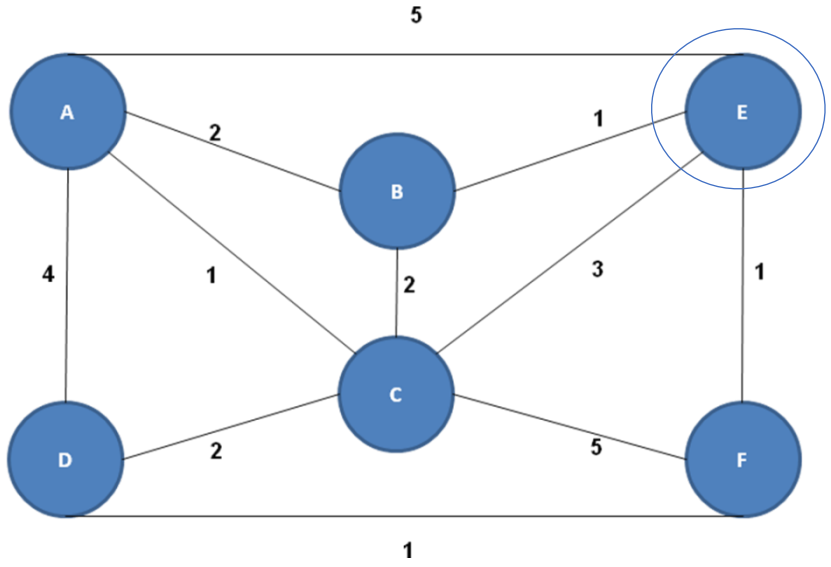


- Node E listens to messages from its neighbors:
  - A says its routes are 0/A, 5/E, 2/B, 1/C, 4/D, 5/F
  - B says its routes are 0/B, 1/E, 2/C, 2/A, 5/F, 4/D
  - C says its routes are 0/C, 1/A, 2/D, 5/F, 3/E, 2/B
  - F says its routes are 0/F, 5/C, 1/D, 1/E, 2/B, 4/A
  - D isn't heard because it isn't connected to E
- Node E looks at each route from each neighbor N; it adds its cost to get to N to N's cost to get to each other node.
- For each node, if the sum is lower cost than what E has in its route table, E changes its own route to run through the neighbor

- Reviewing Node B's message
  - 1+2/C, 1+2/A, 1+5/F, 1+4/D
  - 3/A and 5/D are lower cost
  - We set their next hop to B

Distance vector in action

Routes from E	A	B	C	D	F
Cost/next hop	5/A	1/B	3/C	9/A	1/F

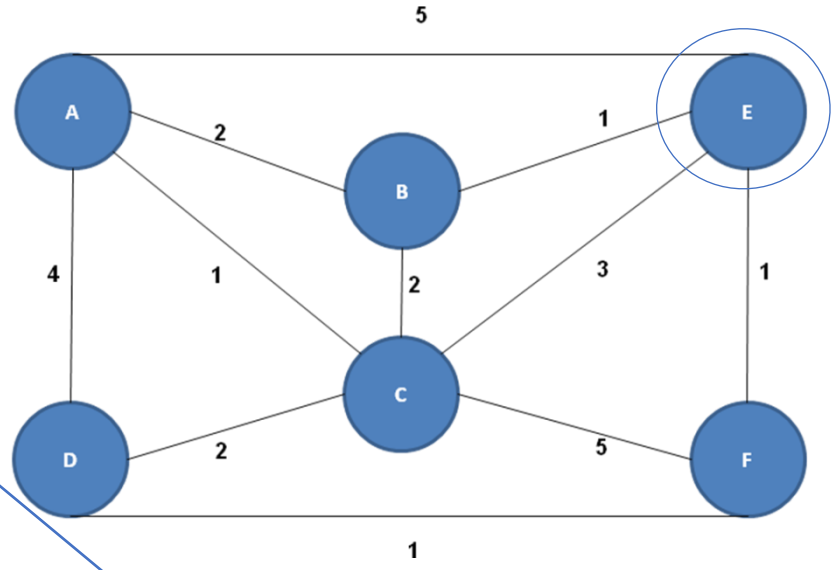


- Node E listens to messages from its neighbors:
  - A says its routes are 0/A, 5/E, 2/B, 1/C, 4/D, 5/F
  - B says its routes are 0/B, 1/E, 2/C, 2/A, 5/F, 4/D
  - C says its routes are 0/C, 1/A, 2/D, 5/F, 3/E, 2/B
  - F says its routes are 0/F, 5/C, 1/D, 1/E, 2/B, 4/A
  - D isn't heard because it isn't connected to E
- Node E looks at each route from each neighbor N; it adds its cost to get to N to N's cost to get to each other node.
- For each node, if the sum is lower cost than what E has in its route table, E changes its own route to run through the neighbor

- Reviewing Node B's message
  - 1+2/C, 1+2/A, 1+5/F, 1+4/D
  - 3/A and 5/D are lower cost
  - We set their next hop to B

Distance vector in action

Routes from E	A	B	C	D	F
Cost/next hop	5/A	1/B	3/C	9/A	1/F

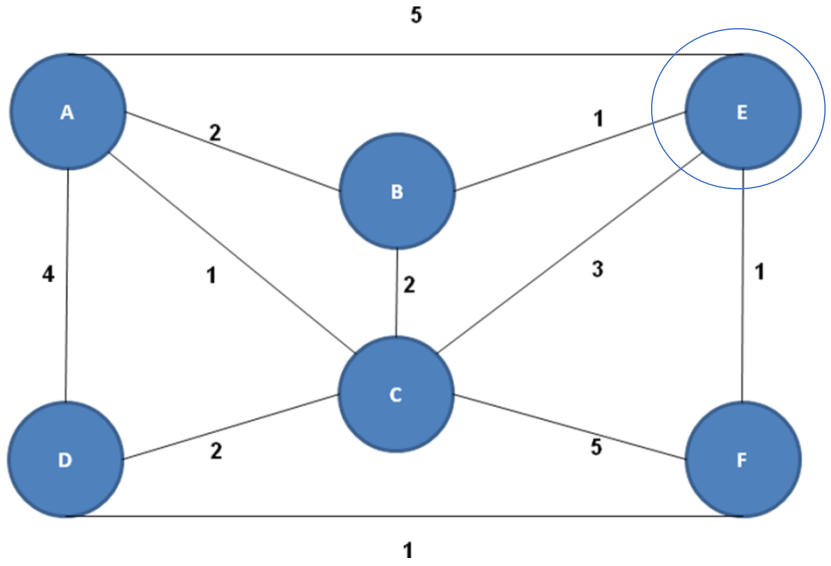


- Node E listens to messages from its neighbors:
  - A says its routes are 0/A, 5/E, 2/B, 1/C, 4/D, 5/F
  - B says its routes are 0/B, 1/E, 2/C, 2/A, 5/F, 4/D
  - C says its routes are 0/C, 1/A, 2/D, 5/F, 3/E, 2/B
  - F says its routes are 0/F, 5/C, 1/D, 1/E, 2/B, 4/A
  - D isn't heard because it isn't connected to E
- Node E looks at each route from each neighbor N; it adds its cost to get to N to N's cost to get to each other node.
- For each node, if the sum is lower cost than what E has in its route table, E changes its own route to run through the neighbor

- Reviewing Node B's message
  - 1+2/C, 1+2/A, 1+5/F, 1+4/D
  - 3/A and 5/D are lower cost
  - We set their next hop to B

Distance vector in action

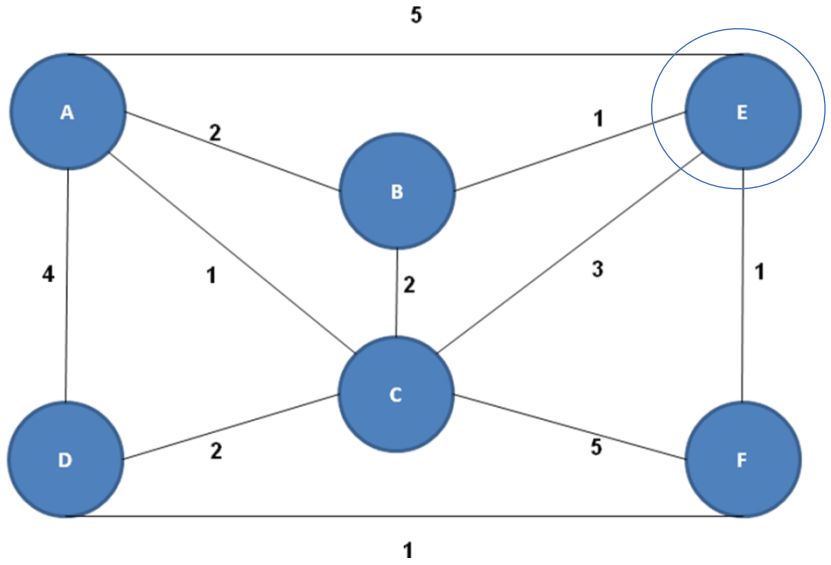
Routes from E	A	B	C	D	F
Cost/next hop	3/B	1/B	3/C	5/B	1/F



- Node E listens to messages from its neighbors:
  - A says its routes are 0/A, 5/E, 2/B, 1/C, 4/D, 5/F
  - B says its routes are 0/B, 1/E, 2/C, 2/A, 5/F, 4/D
  - C says its routes are 0/C, 1/A, 2/D, 5/F, 3/E, 2/B
  - F says its routes are 0/F, 5/C, 1/D, 1/E, 2/B, 4/A
  - D isn't heard because it isn't connected to E
- Node E looks at each route from each neighbor N; it adds its cost to get to N to N's cost to get to each other node.
- For each node, if the sum is lower cost than what E has in its route table, E changes its own route to run through the neighbor

Distance vector in action

Routes from E	A	B	C	D	F
Cost/next hop	3/B	1/B	3/C	5/B	1/F

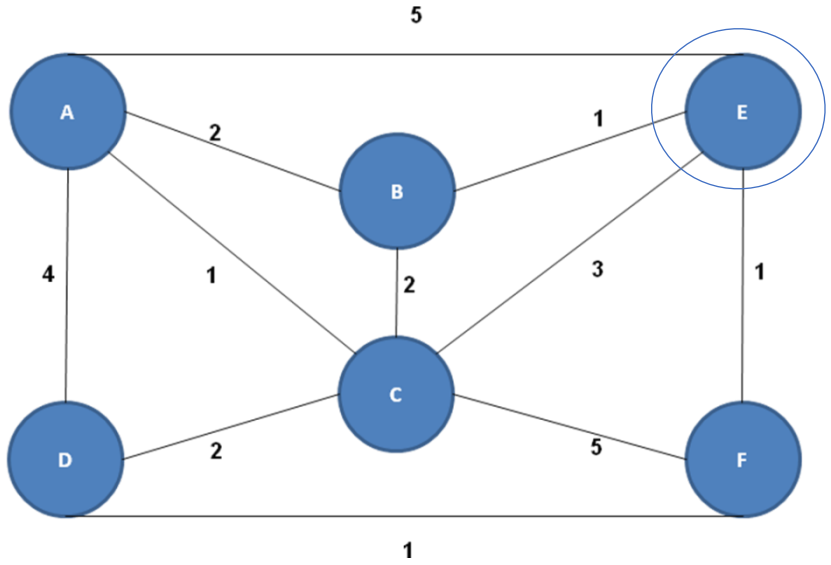


- Reviewing Node C's message

- Node E listens to messages from its neighbors:
  - A says its routes are 0/A, 5/E, 2/B, 1/C, 4/D, 5/F
  - B says its routes are 0/B, 1/E, 2/C, 2/A, 5/F, 4/D
  - C says its routes are 0/C, 1/A, 2/D, 5/F, 3/E, 2/B
  - F says its routes are 0/F, 5/C, 1/D, 1/E, 2/B, 4/A
  - D isn't heard because it isn't connected to E
- Node E looks at each route from each neighbor N; it adds its cost to get to N to N's cost to get to each other node.
- For each node, if the sum is lower cost than what E has in its route table, E changes its own route to run through the neighbor

Distance vector in action

Routes from E	A	B	C	D	F
Cost/next hop	3/B	1/B	3/C	5/B	1/F

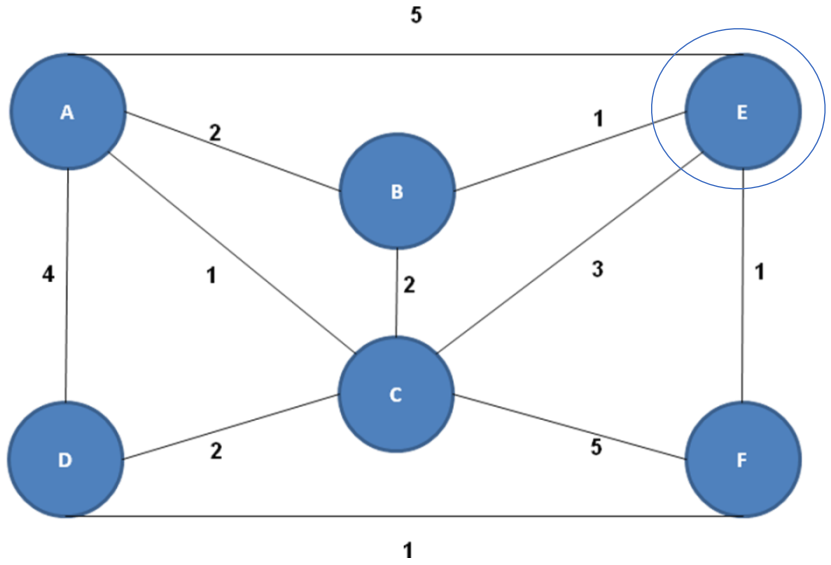


- Node E listens to messages from its neighbors:
  - A says its routes are 0/A, 5/E, 2/B, 1/C, 4/D, 5/F
  - B says its routes are 0/B, 1/E, 2/C, 2/A, 5/F, 4/D
  - C says its routes are 0/C, 1/A, 2/D, 5/F, 3/E, 2/B
  - F says its routes are 0/F, 5/C, 1/D, 1/E, 2/B, 4/A
  - D isn't heard because it isn't connected to E
- Node E looks at each route from each neighbor N; it adds its cost to get to N to N's cost to get to each other node.
- For each node, if the sum is lower cost than what E has in its route table, E changes its own route to run through the neighbor

- Reviewing Node C's message
  - 3+1/A, 3+2/D, 3+5/F, 3+2/B

Distance vector in action

Routes from E	A	B	C	D	F
Cost/next hop	3/B	1/B	3/C	5/B	1/F

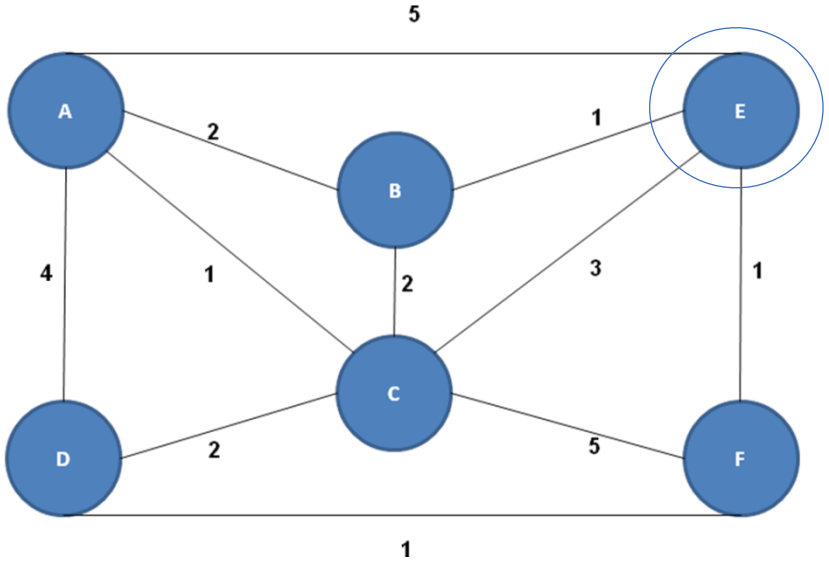


- Node E listens to messages from its neighbors:
  - A says its routes are 0/A, 5/E, 2/B, 1/C, 4/D, 5/F
  - B says its routes are 0/B, 1/E, 2/C, 2/A, 5/F, 4/D
  - C says its routes are 0/C, 1/A, 2/D, 5/F, 3/E, 2/B
  - F says its routes are 0/F, 5/C, 1/D, 1/E, 2/B, 4/A
  - D isn't heard because it isn't connected to E
- Node E looks at each route from each neighbor N; it adds its cost to get to N to N's cost to get to each other node.
- For each node, if the sum is lower cost than what E has in its route table, E changes its own route to run through the neighbor

- Reviewing Node C's message
  - 3+1/A, 3+2/D, 3+5/F, 3+2/B
  - None are lower cost

Distance vector in action

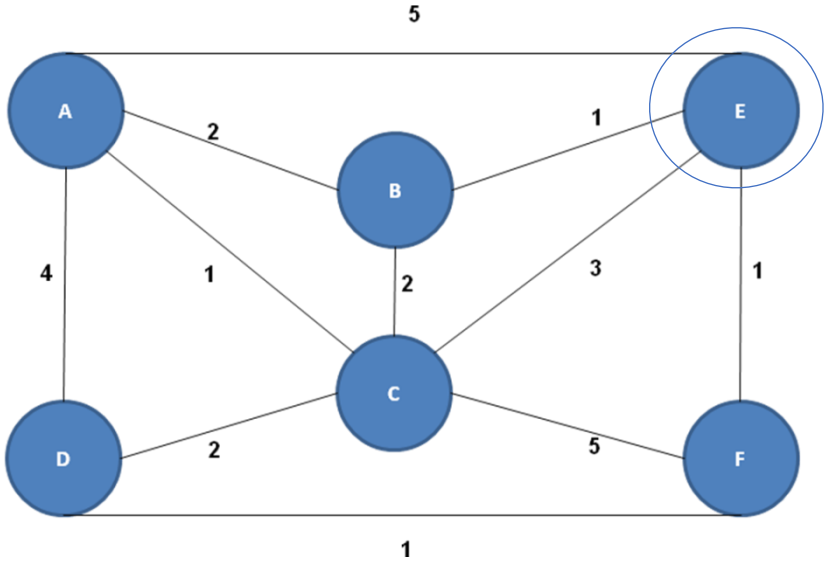
Routes from E	A	B	C	D	F
Cost/next hop	3/B	1/B	3/C	5/B	1/F



- Node E listens to messages from its neighbors:
  - A says its routes are 0/A, 5/E, 2/B, 1/C, 4/D, 5/F
  - B says its routes are 0/B, 1/E, 2/C, 2/A, 5/F, 4/D
  - C says its routes are 0/C, 1/A, 2/D, 5/F, 3/E, 2/B
  - F says its routes are 0/F, 5/C, 1/D, 1/E, 2/B, 4/A
  - D isn't heard because it isn't connected to E
- Node E looks at each route from each neighbor N; it adds its cost to get to N to N's cost to get to each other node.
- For each node, if the sum is lower cost than what E has in its route table, E changes its own route to run through the neighbor

Distance vector in action

Routes from E	A	B	C	D	F
Cost/next hop	3/B	1/B	3/C	5/B	1/F

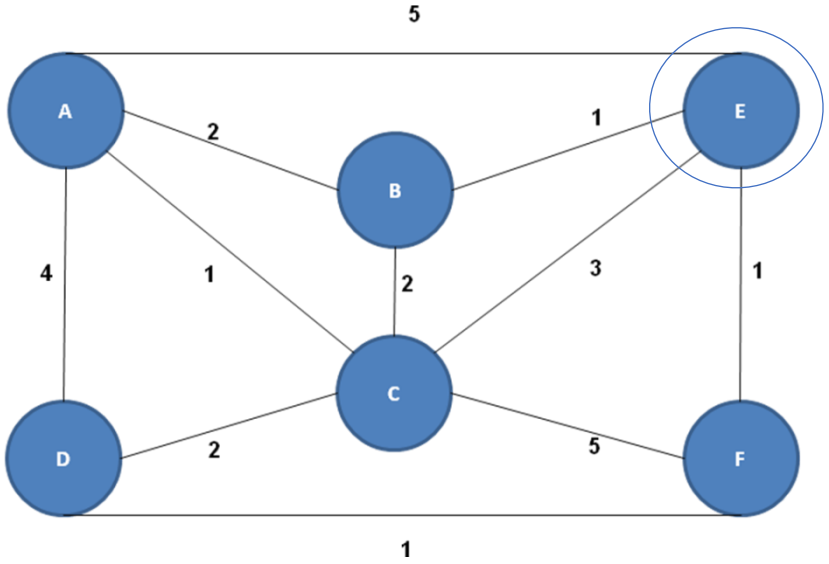


- Reviewing Node F's message

- Node E listens to messages from its neighbors:
  - A says its routes are 0/A, 5/E, 2/B, 1/C, 4/D, 5/F
  - B says its routes are 0/B, 1/E, 2/C, 2/A, 5/F, 4/D
  - C says its routes are 0/C, 1/A, 2/D, 5/F, 3/E, 2/B
  - F says its routes are 0/F, 5/C, 1/D, 1/E, 2/B, 4/A
  - D isn't heard because it isn't connected to E
- Node E looks at each route from each neighbor N; it adds its cost to get to N to N's cost to get to each other node.
- For each node, if the sum is lower cost than what E has in its route table, E changes its own route to run through the neighbor

Distance vector in action

Routes from E	A	B	C	D	F
Cost/next hop	3/B	1/B	3/C	5/B	1/F

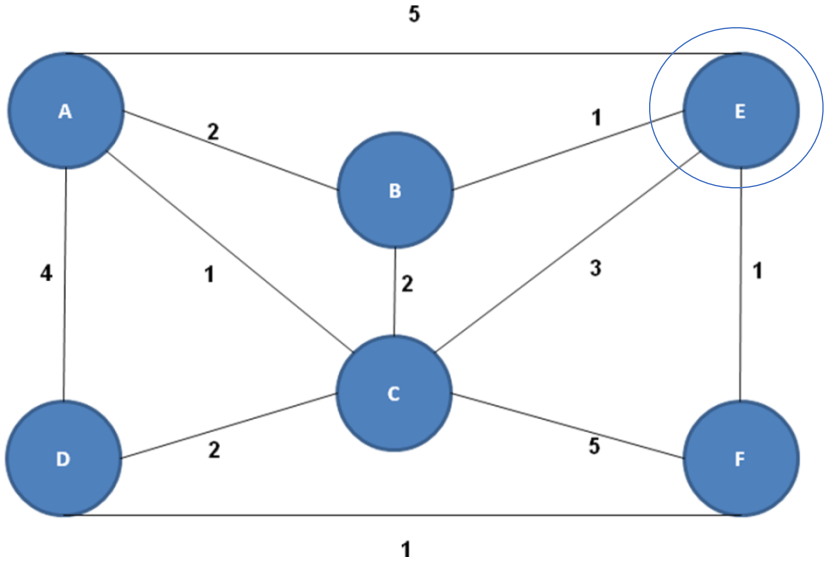


- Node E listens to messages from its neighbors:
  - A says its routes are 0/A, 5/E, 2/B, 1/C, 4/D, 5/F
  - B says its routes are 0/B, 1/E, 2/C, 2/A, 5/F, 4/D
  - C says its routes are 0/C, 1/A, 2/D, 5/F, 3/E, 2/B
  - F says its routes are 0/F, 5/C, 1/D, 1/E, 2/B, 4/A
  - D isn't heard because it isn't connected to E
- Node E looks at each route from each neighbor N; it adds its cost to get to N to N's cost to get to each other node.
- For each node, if the sum is lower cost than what E has in its route table, E changes its own route to run through the neighbor

- Reviewing Node F's message
  - 1+5C, 1+1/D, 1+2/B, 1+4/A

Distance vector in action

Routes from E	A	B	C	D	F
Cost/next hop	3/B	1/B	3/C	5/B	1/F

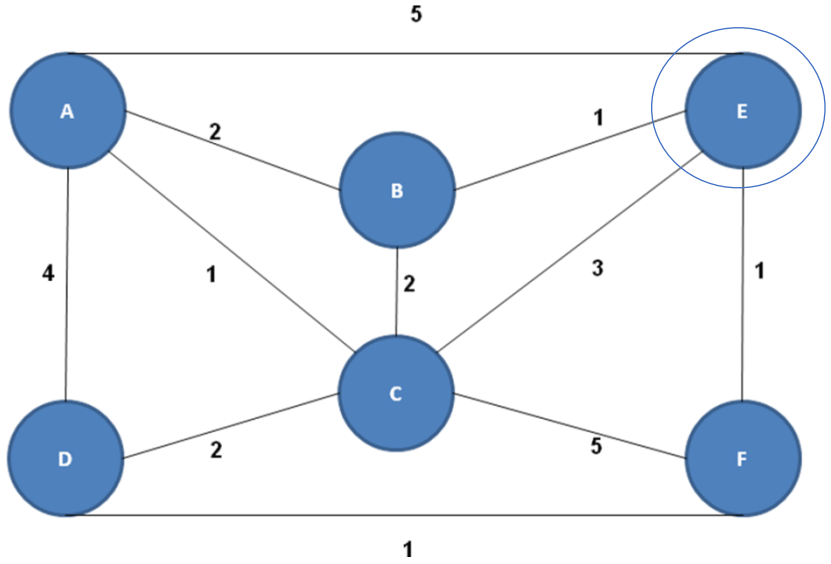


- Node E listens to messages from its neighbors:
  - A says its routes are 0/A, 5/E, 2/B, 1/C, 4/D, 5/F
  - B says its routes are 0/B, 1/E, 2/C, 2/A, 5/F, 4/D
  - C says its routes are 0/C, 1/A, 2/D, 5/F, 3/E, 2/B
  - F says its routes are 0/F, 5/C, 1/D, 1/E, 2/B, 4/A
  - D isn't heard because it isn't connected to E
- Node E looks at each route from each neighbor N; it adds its cost to get to N to N's cost to get to each other node.
- For each node, if the sum is lower cost than what E has in its route table, E changes its own route to run through the neighbor

- Reviewing Node F's message
  - 1+5C, 1+1/D, 1+2/B, 1+4/A
  - 2/D is lower cost

Distance vector in action

Routes from E	A	B	C	D	F
Cost/next hop	3/B	1/B	3/C	5/B	1/F

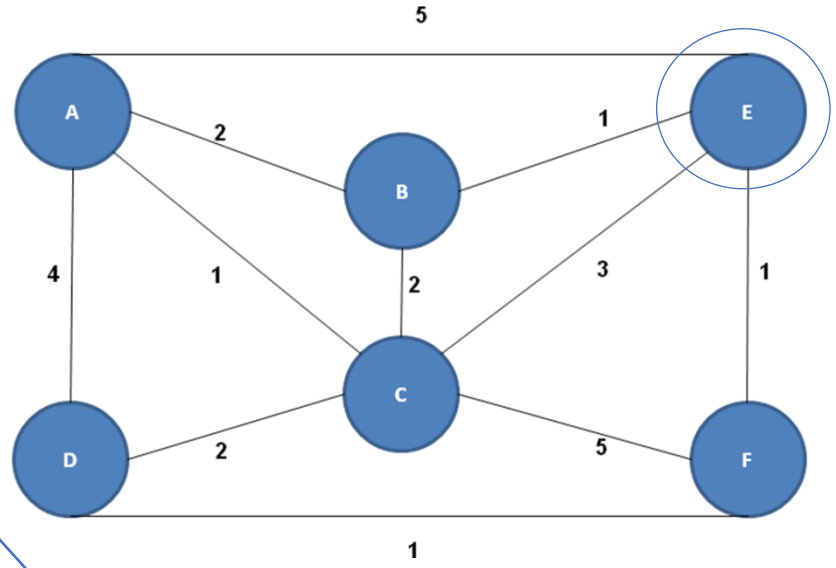


- Node E listens to messages from its neighbors:
  - A says its routes are 0/A, 5/E, 2/B, 1/C, 4/D, 5/F
  - B says its routes are 0/B, 1/E, 2/C, 2/A, 5/F, 4/D
  - C says its routes are 0/C, 1/A, 2/D, 5/F, 3/E, 2/B
  - F says its routes are 0/F, 5/C, 1/D, 1/E, 2/B, 4/A
  - D isn't heard because it isn't connected to E
- Node E looks at each route from each neighbor N; it adds its cost to get to N to N's cost to get to each other node.
- For each node, if the sum is lower cost than what E has in its route table, E changes its own route to run through the neighbor

- Reviewing Node F's message
  - 1+5C, 1+1/D, 1+2/B, 1+4/A
  - 2/D is lower cost
  - We'll set the next hop to F

Distance vector in action

Routes from E	A	B	C	D	F
Cost/next hop	3/B	1/B	3/C	5/B	1/F

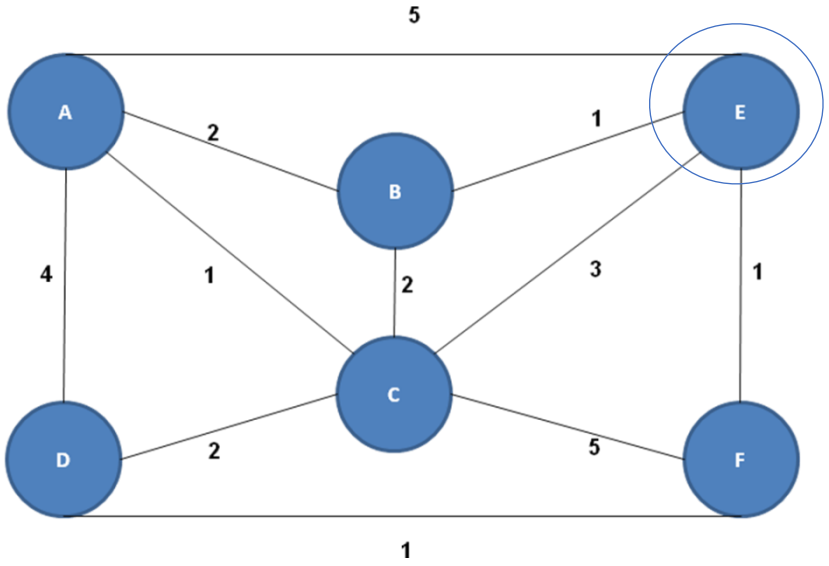


- Node E listens to messages from its neighbors:
  - A says its routes are 0/A, 5/E, 2/B, 1/C, 4/D, 5/F
  - B says its routes are 0/B, 1/E, 2/C, 2/A, 5/F, 4/D
  - C says its routes are 0/C, 1/A, 2/D, 5/F, 3/E, 2/B
  - F says its routes are 0/F, 5/C, 1/D, 1/E, 2/B, 4/A
  - D isn't heard because it isn't connected to E
- Node E looks at each route from each neighbor N; it adds its cost to get to N to N's cost to get to each other node.
- For each node, if the sum is lower cost than what E has in its route table, E changes its own route to run through the neighbor

- Reviewing Node F's message
  - 1+5C, 1+1/D, 1+2/B, 1+4/A
  - 2/D is lower cost
  - We'll set the next hop to F

Distance vector in action

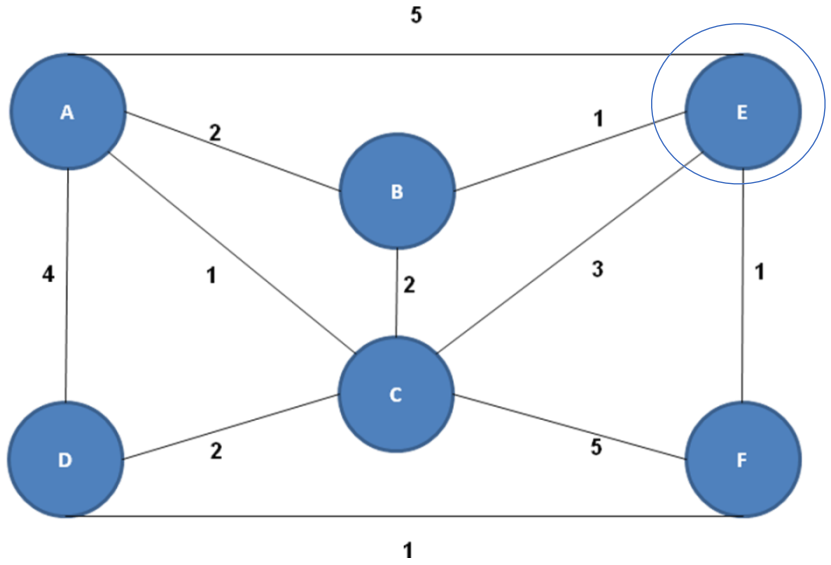
Routes from E	A	B	C	D	F
Cost/next hop	3/B	1/B	3/C	2/F	1/F



- Node E listens to messages from its neighbors:
  - A says its routes are 0/A, 5/E, 2/B, 1/C, 4/D, 5/F
  - B says its routes are 0/B, 1/E, 2/C, 2/A, 5/F, 4/D
  - C says its routes are 0/C, 1/A, 2/D, 5/F, 3/E, 2/B
  - F says its routes are 0/F, 5/C, 1/D, 1/E, 2/B, 4/A
  - D isn't heard because it isn't connected to E
- Node E looks at each route from each neighbor N; it adds its cost to get to N to N's cost to get to each other node.
- For each node, if the sum is lower cost than what E has in its route table, E changes its own route to run through the neighbor

Distance vector in action

Routes from E	A	B	C	D	F
Cost/next hop	3/B	1/B	3/C	2/F	1/F

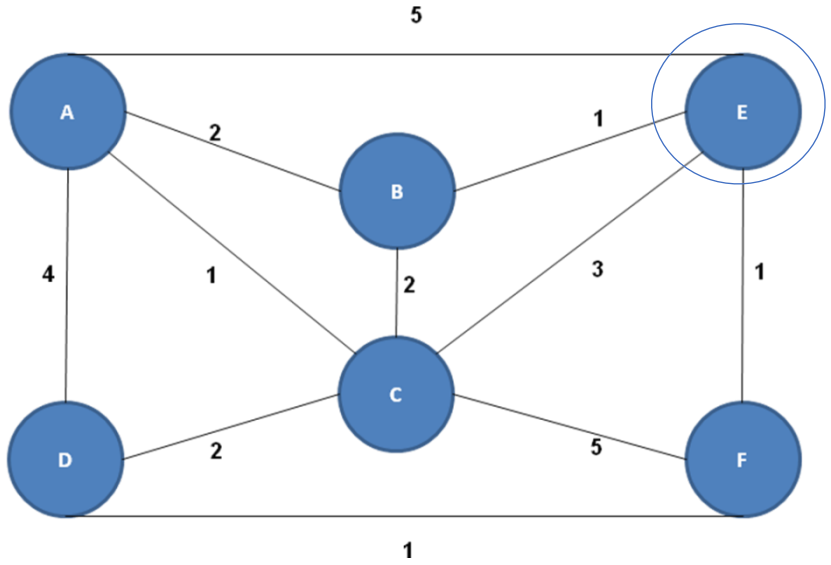


- Node E listens to messages from its neighbors:
  - A says its routes are 0/A, 5/E, 2/B, 1/C, 4/D, 5/F
  - B says its routes are 0/B, 1/E, 2/C, 2/A, 5/F, 4/D
  - C says its routes are 0/C, 1/A, 2/D, 5/F, 3/E, 2/B
  - F says its routes are 0/F, 5/C, 1/D, 1/E, 2/B, 4/A
  - D isn't heard because it isn't connected to E
- Node E looks at each route from each neighbor N; it adds its cost to get to N to N's cost to get to each other node.
- For each node, if the sum is lower cost than what E has in its route table, E changes its own route to run through the neighbor

- Do you agree that E now knows the next hop for the lowest cost routes to the other nodes?

Distance vector in action

Routes from E	A	B	C	D	F
Cost/next hop	3/B	1/B	3/C	2/F	1/F

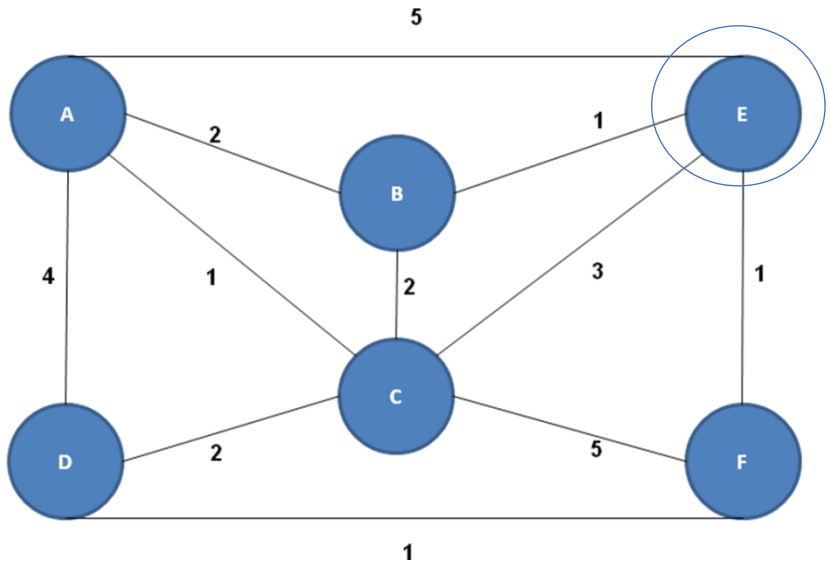


- Node E listens to messages from its neighbors:
  - A says its routes are 0/A, 5/E, 2/B, 1/C, 4/D, 5/F
  - B says its routes are 0/B, 1/E, 2/C, 2/A, 5/F, 4/D
  - C says its routes are 0/C, 1/A, 2/D, 5/F, 3/E, 2/B
  - F says its routes are 0/F, 5/C, 1/D, 1/E, 2/B, 4/A
  - D isn't heard because it isn't connected to E
- Node E looks at each route from each neighbor N; it adds its cost to get to N to N's cost to get to each other node.
- For each node, if the sum is lower cost than what E has in its route table, E changes its own route to run through the neighbor

- Do you agree that E now knows the next hop for the lowest cost routes to the other nodes?
- This of course presumed the other nodes knew their lowest cost routes

Distance vector in action

Routes from E	A	B	C	D	F
Cost/next hop	3/B	1/B	3/C	2/F	1/F



- Node E listens to messages from its neighbors:
  - A says its routes are 0/A, 5/E, 2/B, 1/C, 4/D, 5/F
  - B says its routes are 0/B, 1/E, 2/C, 2/A, 5/F, 4/D
  - C says its routes are 0/C, 1/A, 2/D, 5/F, 3/E, 2/B
  - F says its routes are 0/F, 5/C, 1/D, 1/E, 2/B, 4/A
  - D isn't heard because it isn't connected to E
- Node E looks at each route from each neighbor N; it adds its cost to get to N to N's cost to get to each other node.
- For each node, if the sum is lower cost than what E has in its route table, E changes its own route to run through the neighbor

- Do you agree that E now knows the next hop for the lowest cost routes to the other nodes?
- This of course presumed the other nodes knew their lowest cost routes
- When this runs on all nodes simultaneously, they will ***converge*** on this solution in a relatively short time (convergence depends on diameter of the network)

Distance Vector vs Link State Routing

Distance Vector vs Link State Routing

- Distance Vector
  - Each node knows the cost to reach each of its neighbors
  - Each node sends its routing table to only its neighbors
  - Each node revises its own routing table every time it receives a routing table from a neighbor
  - Slower convergence but lower traffic

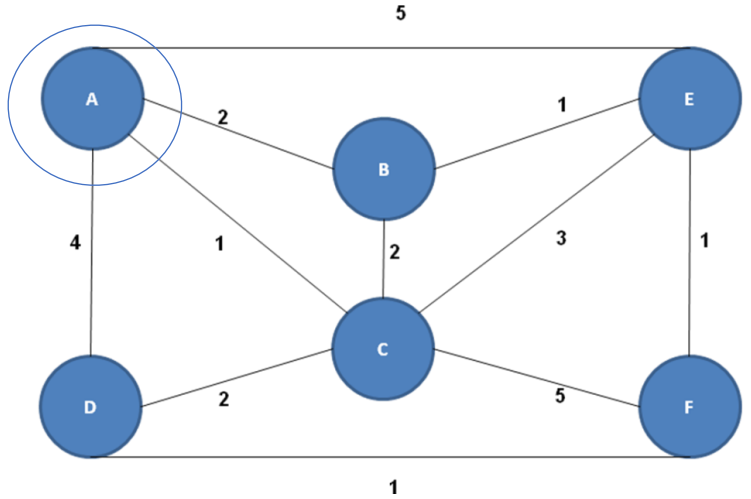
Distance Vector vs Link State Routing

- Distance Vector
  - Each node knows the cost to reach each of its neighbors
  - Each node sends its routing table to only its neighbors
  - Each node revises its own routing table every time it receives a routing table from a neighbor
  - Slower convergence but lower traffic

- Link State
  - Each node advertises its neighbor links and costs to every other node in the network
  - Each node collects the entire topology of the network from these advertisements
  - Using a "shortest path" algorithm, each node calculates its own routing table
  - Faster convergence but more traffic

Link state in action

- Dijkstra's shortest path algorithm
- Uses global info, local algorithm (i.e. every node has this graph)
- n-1 iterations (for n-node network)
- One least-cost route to a destination
- Runs simultaneously on each node



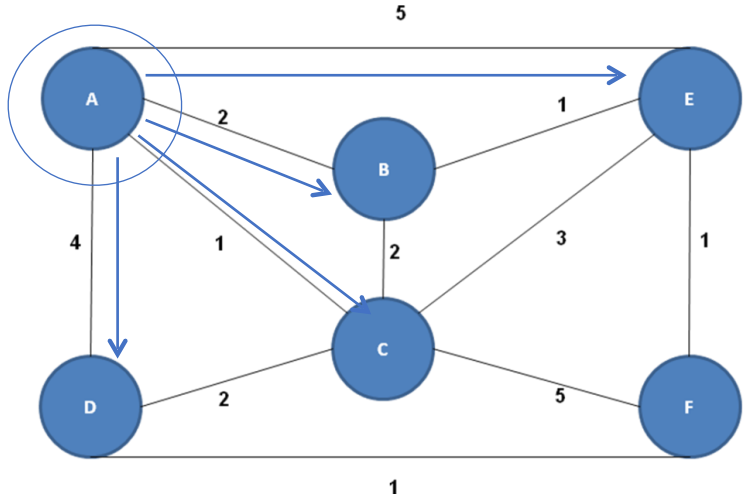
Calculating on Node A

(each node does its own calculation)

Iteration Count	Nodes to which least-cost routes known	B Cost/ route	C Cost/ route	D cost/ route	E Cost/ route	F Cost/ route
Init	A	2/AB	1/AC	4/AD	5/AE	∞
1						
2						
3						
4						
5						

Link state in action

- Dijkstra's shortest path algorithm
- Uses global info, local algorithm (i.e. every node has this graph)
- n-1 iterations (for n-node network)
- One least-cost route to a destination
- Runs simultaneously on each node



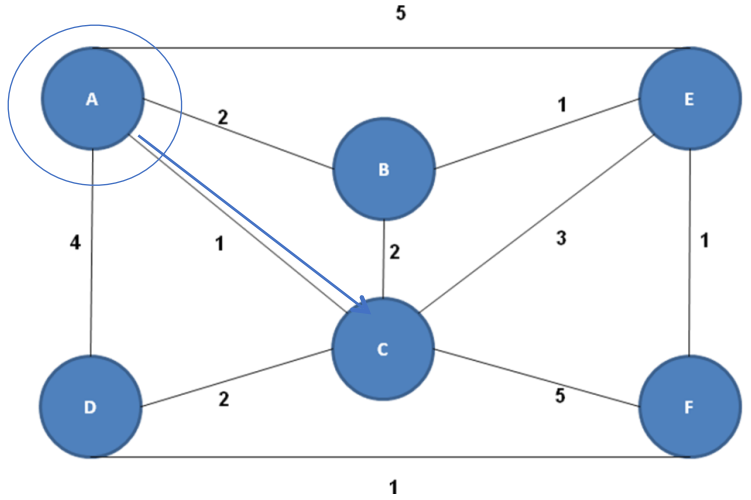
Calculating on Node A

(each node does its own calculation)

Iteration Count	Nodes to which least-cost routes known	B Cost/ route	C Cost/ route	D cost/ route	E Cost/ route	F Cost/ route
Init	A	2/AB	1/AC	4/AD	5/AE	∞
1						
2						
3						
4						
5						

Link state in action

- Dijkstra's shortest path algorithm
- Uses global info, local algorithm (i.e. every node has this graph)
- n-1 iterations (for n-node network)
- One least-cost route to a destination
- Runs simultaneously on each node



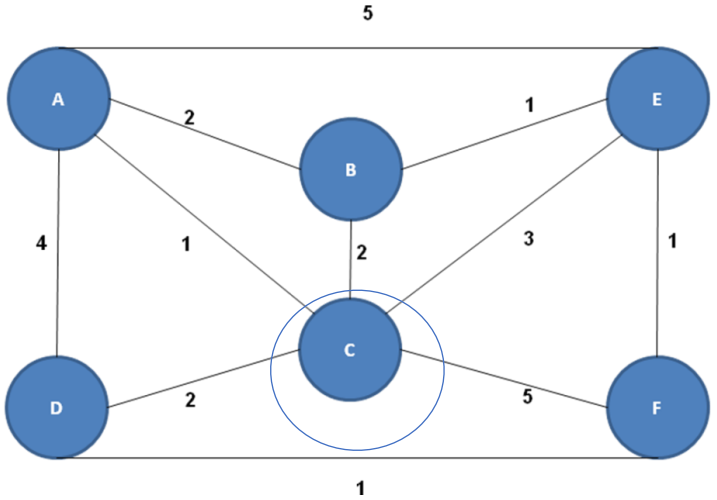
Calculating on Node A

(each node does its own calculation)

Iteration Count	Nodes to which least-cost routes known	B Cost/ route	C Cost/ route	D cost/ route	E Cost/ route	F Cost/ route
Init	A	2/AB	1/AC	4/AD	5/AE	∞
1						
2						
3						
4						
5						

Link state in action

- Dijkstra's shortest path algorithm
- Uses global info, local algorithm (i.e. every node has this graph)
- n-1 iterations
- One least-cost route to a destination
- Runs simultaneously on each node



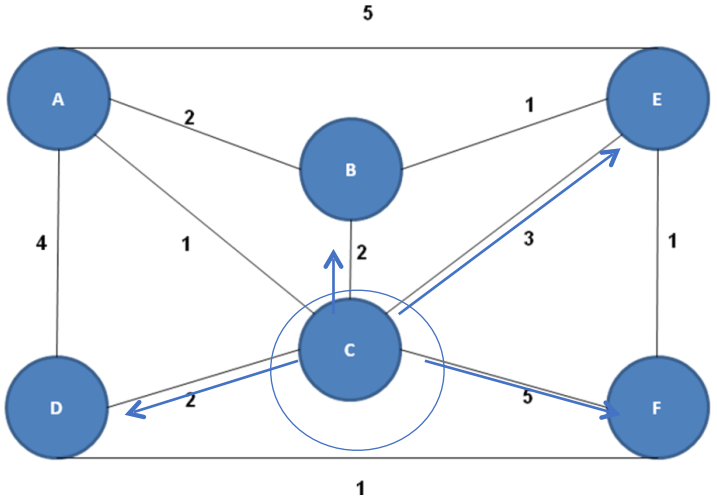
Calculating on Node A

(each node does its own calculation)

Iteration Count	Nodes to which least-cost routes known	B Cost/ route	C Cost/ route	D cost/ route	E Cost/ route	F Cost/ route
Init	A	2/AB	1/AC	4/AD	5/AE	∞
1	A,C	2/AB	1/AC	3/ACD	4/ACE	6/ACF
2						
3						
4						
5						

Link state in action

- Dijkstra's shortest path algorithm
- Uses global info, local algorithm (i.e. every node has this graph)
- n-1 iterations
- One least-cost route to a destination
- Runs simultaneously on each node



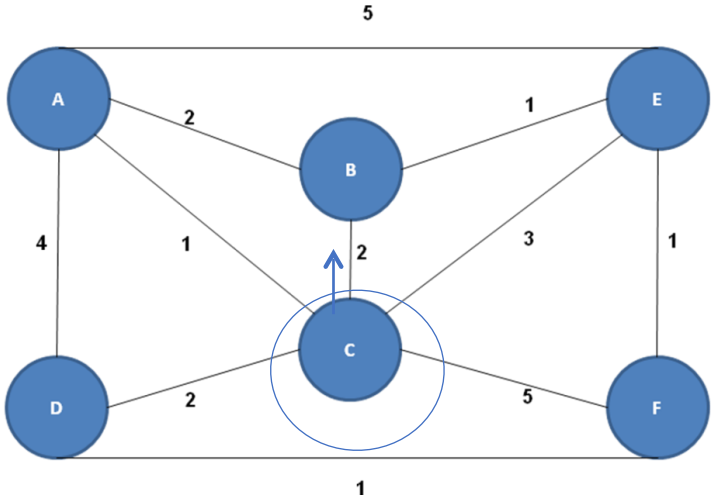
Calculating on Node A

(each node does its own calculation)

Iteration Count	Nodes to which least-cost routes known	B Cost/ route	C Cost/ route	D cost/ route	E Cost/ route	F Cost/ route
Init	A	2/AB	1/AC	4/AD	5/AE	∞
1	A,C	2/AB	1/AC	3/ACD	4/ACE	6/ACF
2						
3						
4						
5						

Link state in action

- Dijkstra's shortest path algorithm
- Uses global info, local algorithm (i.e. every node has this graph)
- n-1 iterations
- One least-cost route to a destination
- Runs simultaneously on each node



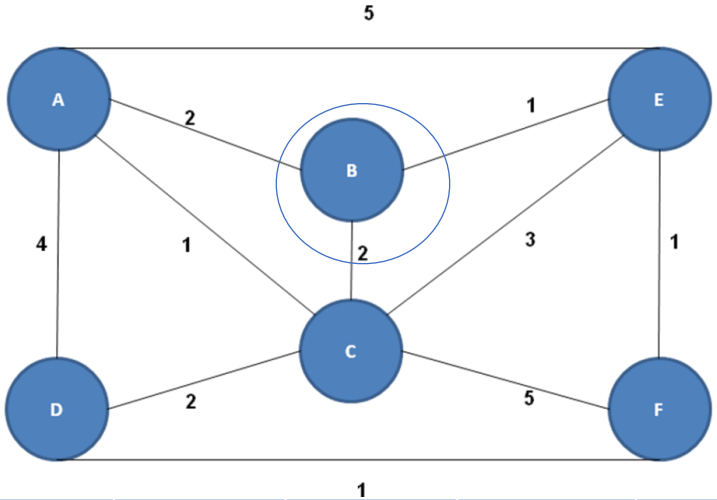
Calculating on Node A

(each node does its own calculation)

Iteration Count	Nodes to which least-cost routes known	B Cost/ route	C Cost/ route	D cost/ route	E Cost/ route	F Cost/ route
Init	A	2/AB	1/AC	4/AD	5/AE	∞
1	A,C	2/AB	1/AC	3/ACD	4/ACE	6/ACF
2						
3						
4						
5						

Link state in action

- Dijkstra's shortest path algorithm
- Uses global info, local algorithm (i.e. every node has this graph)
- n-1 iterations
- One least-cost route to a destination
- Runs simultaneously on each node



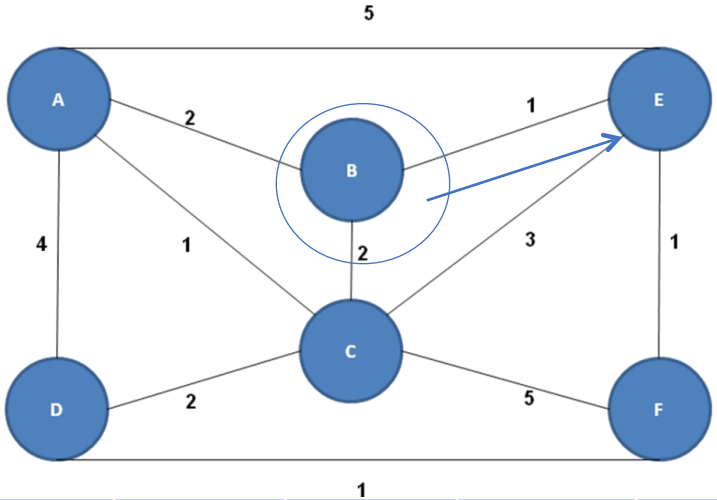
Calculating on Node A

(each node does its own calculation)

Iteration Count	Nodes to which least-cost routes known	B Cost/ route	C Cost/ route	D cost/ route	E Cost/ route	F Cost/ route
Init	A	2/AB	1/AC	4/AD	5/AE	∞
1	A,C	2/AB	1/AC	3/ACD	4/ACE	6/ACF
2	A,C,B	2/AB	-	3/ACD	3/ABE	6/ACF
3						
4						
5						

Link state in action

- Dijkstra's shortest path algorithm
- Uses global info, local algorithm (i.e. every node has this graph)
- n-1 iterations
- One least-cost route to a destination
- Runs simultaneously on each node



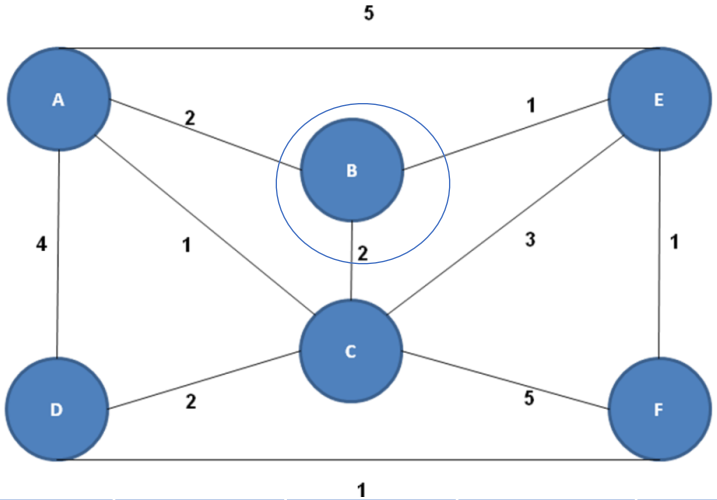
Calculating on Node A

(each node does its own calculation)

Iteration Count	Nodes to which least-cost routes known	B Cost/ route	C Cost/ route	D cost/ route	E Cost/ route	F Cost/ route
Init	A	2/AB	1/AC	4/AD	5/AE	∞
1	A,C	2/AB	1/AC	3/ACD	4/ACE	6/ACF
2	A,C,B	2/AB	-	3/ACD	3/ABE	6/ACF
3						
4						
5						

Link state in action

- Dijkstra's shortest path algorithm
- Uses global info, local algorithm (i.e. every node has this graph)
- n-1 iterations
- One least-cost route to a destination
- Runs simultaneously on each node



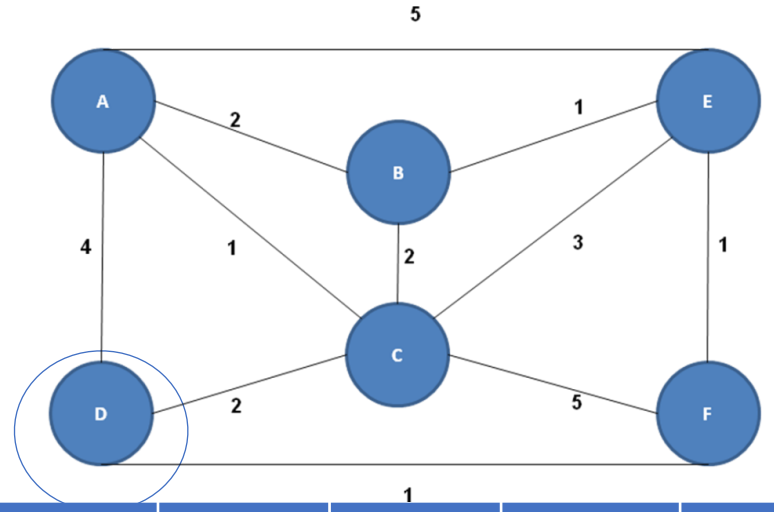
Calculating on Node A

(each node does its own calculation)

Iteration Count	Nodes to which least-cost routes known	B Cost/ route	C Cost/ route	D cost/ route	E Cost/ route	F Cost/ route
Init	A	2/AB	1/AC	4/AD	5/AE	∞
1	A,C	2/AB	1/AC	3/ACD	4/ACE	6/ACF
2	A,C,B	2/AB	-	3/ACD	3/ABE	6/ACF
3						
4						
5						

Link state in action

- Dijkstra's shortest path algorithm
- Uses global info, local algorithm (i.e. every node has this graph)
- n-1 iterations
- One least-cost route to a destination
- Runs simultaneously on each node



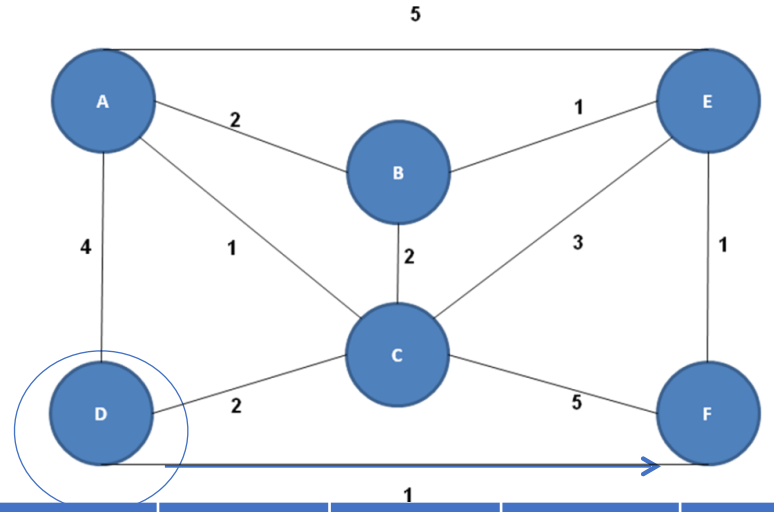
Calculating on Node A

(each node does its own calculation)

Iteration Count	Nodes to which least-cost routes known	B Cost/ route	C Cost/ route	D cost/ route	E Cost/ route	F Cost/ route
Init	A	2/AB	1/AC	4/AD	5/AE	∞
1	A,C	2/AB	1/AC	3/ACD	4/ACE	6/ACF
2	A,C,B	2/AB	-	3/ACD	3/ABE	6/ACF
3	A,C,B,D	-	-	3/ACD	3/ABE	4/ACDF
4						
5						

Link state in action

- Dijkstra's shortest path algorithm
- Uses global info, local algorithm (i.e. every node has this graph)
- n-1 iterations
- One least-cost route to a destination
- Runs simultaneously on each node



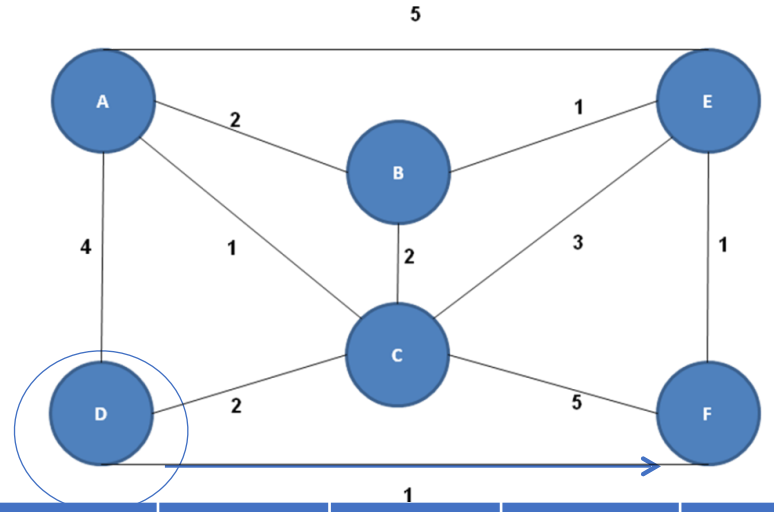
Calculating on Node A

(each node does its own calculation)

Iteration Count	Nodes to which least-cost routes known	B Cost/ route	C Cost/ route	D cost/ route	E Cost/ route	F Cost/ route
Init	A	2/AB	1/AC	4/AD	5/AE	∞
1	A,C	2/AB	1/AC	3/ACD	4/ACE	6/ACF
2	A,C,B	2/AB	-	3/ACD	3/ABE	6/ACF
3	A,C,B,D	-	-	3/ACD	3/ABE	4/ACDF
4						
5						

Link state in action

- Dijkstra's shortest path algorithm
- Uses global info, local algorithm (i.e. every node has this graph)
- n-1 iterations
- One least-cost route to a destination
- Runs simultaneously on each node



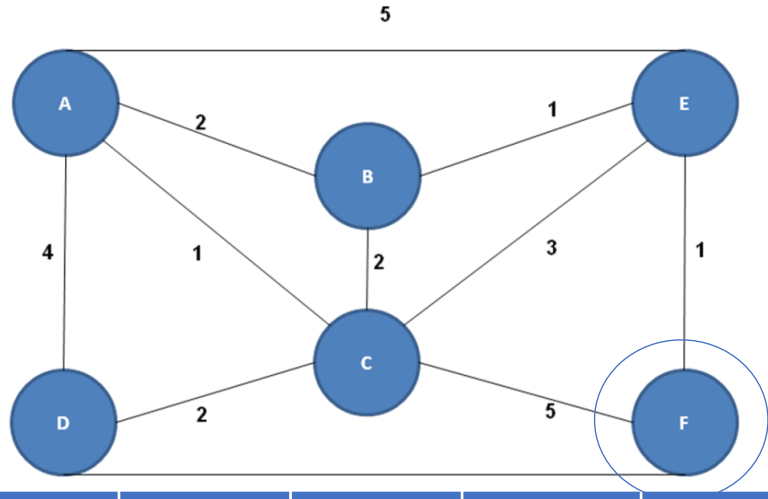
Calculating on Node A

(each node does its own calculation)

Iteration Count	Nodes to which least-cost routes known	B Cost/ route	C Cost/ route	D cost/ route	E Cost/ route	F Cost/ route
Init	A	2/AB	1/AC	4/AD	5/AE	∞
1	A,C	2/AB	1/AC	3/ACD	4/ACE	6/ACF
2	A,C,B	2/AB	-	3/ACD	3/ABE	6/ACF
3	A,C,B,D	-	-	3/ACD	3/ABE	4/ACDF
4						
5						

Link state in action

- Dijkstra's shortest path algorithm
- Uses global info, local algorithm (i.e. every node has this graph)
- n-1 iterations
- One least-cost route to a destination
- Runs simultaneously on each node



Calculating on Node A

(each node does its own calculation)

Iteration Count	Nodes to which least-cost routes known	B Cost/ route	C Cost/ route	D cost/ route	E Cost/ route	F Cost/ route
Init	A	2/AB	1/AC	4/AD	5/AE	∞
1	A,C	2/AB	1/AC	3/ACD	4/ACE	6/ACF
2	A,C,B	2/AB	-	3/ACD	3/ABE	6/ACF
3	A,C,B,D	-	-	3/ACD	3/ABE	4/ACDF
4	A,C,B,D,E	-	-	-	3/ABE	4/ACDF
5	A,C,B,D,E,F	-	-	-	-	4/ACDF

Hierarchical routing algorithms

Hierarchical routing algorithms

- Network of networks

Hierarchical routing algorithms

- Network of networks
- Very large scale, frequent state changes

Hierarchical routing algorithms

- Network of networks
- Very large scale, frequent state changes
- Compartmentalize so routing information protocol issues don't spread and cause widespread outages

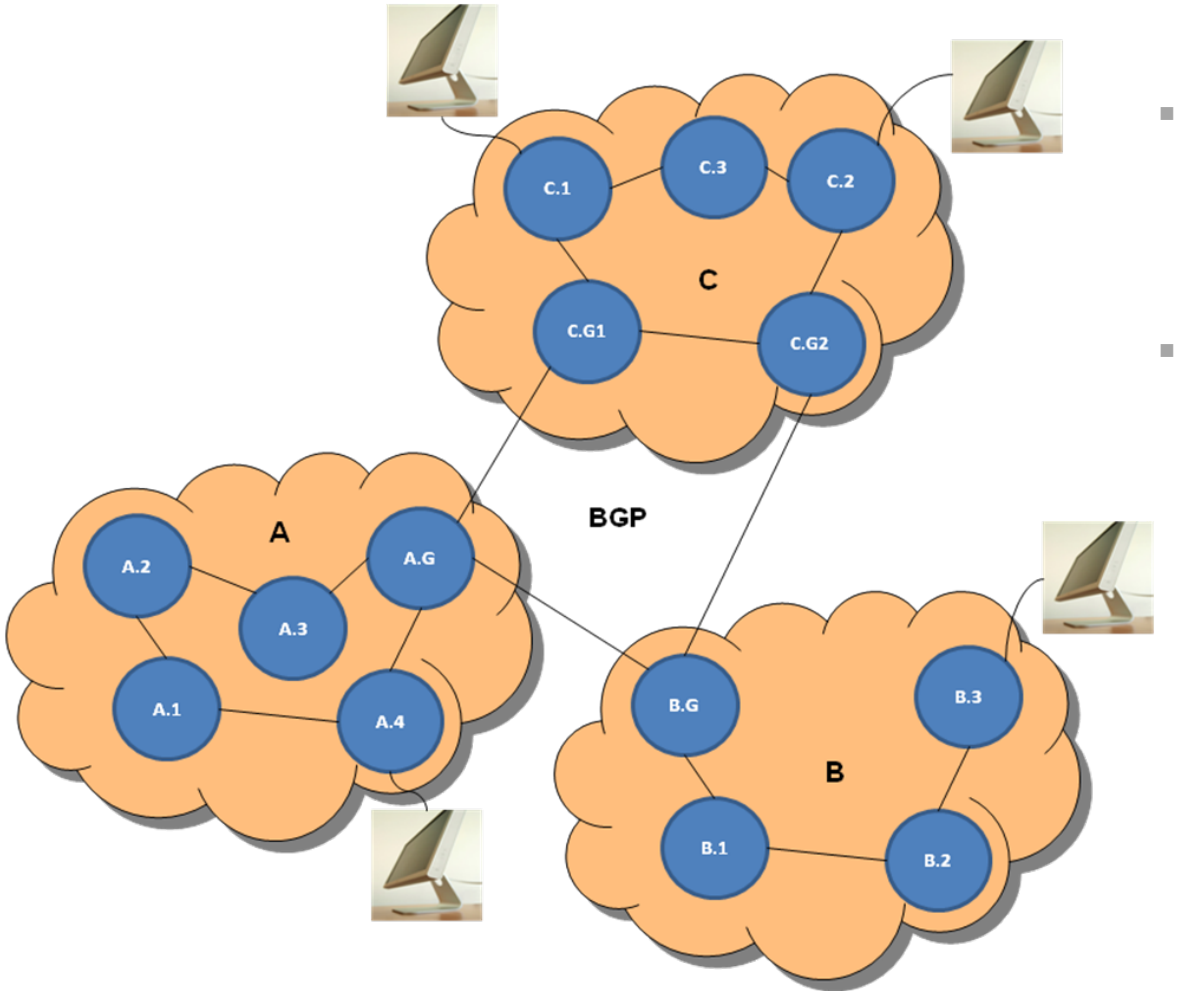
Hierarchical routing algorithms

- Network of networks
- Very large scale, frequent state changes
- Compartmentalize so routing information protocol issues don't spread and cause widespread outages
- What better example than the Internet itself!

Hierarchical routing algorithms

- Network of networks
- Very large scale, frequent state changes
- Compartmentalize so routing information protocol issues don't spread and cause widespread outages
- What better example than the Internet itself!
- Autonomous Systems (AS)
  - It is a collection of IP routing prefixes under the control of common administrative policy that presents a common, clearly defined routing policy to the internet.
  - Gateway nodes connect to other AS gateway nodes for inter-AS routing

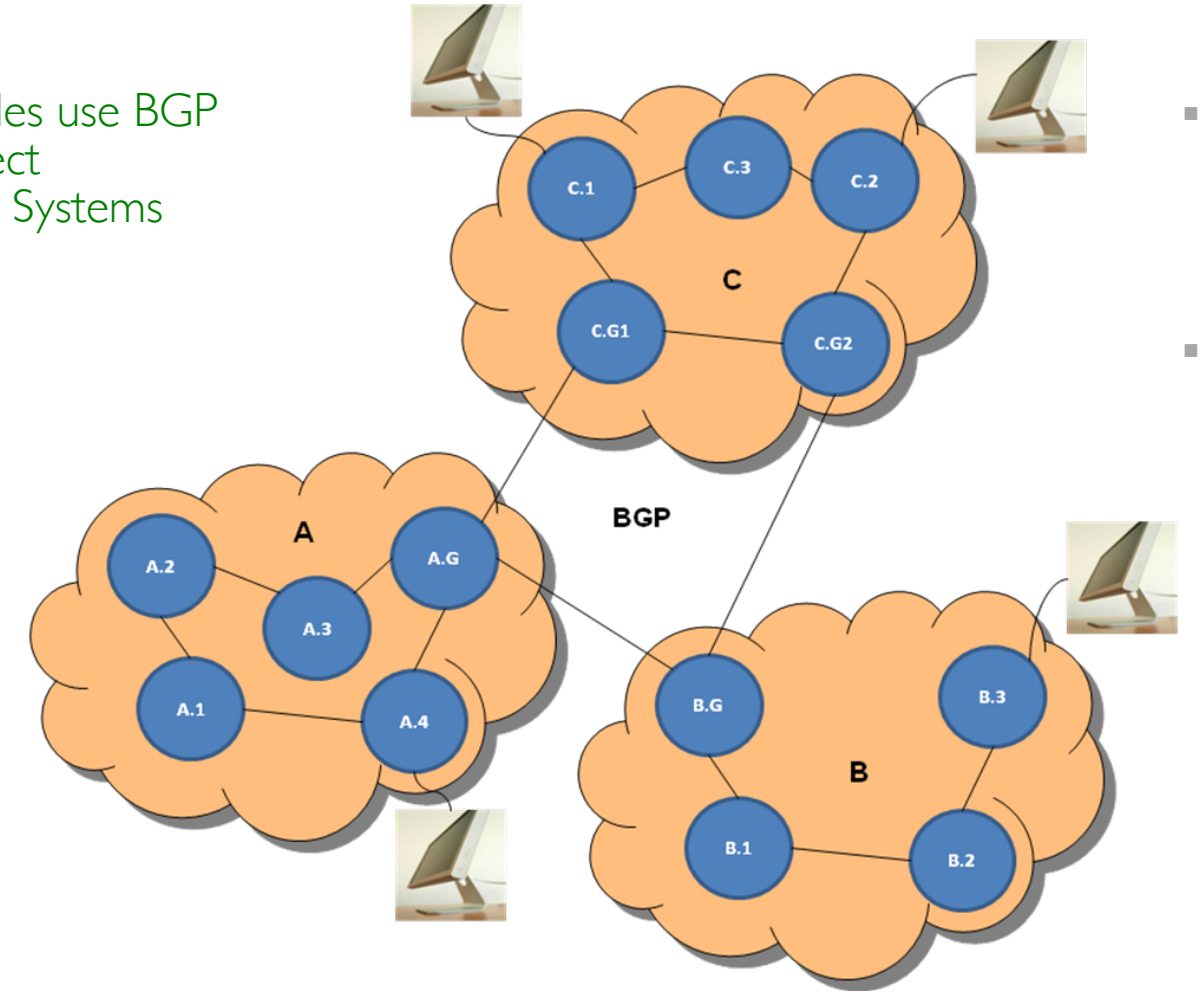
Hierarchical routing algorithms



- ASs are assigned numbers by IANA/ ARIN
- Nodes within an AS use Link State or Distance Vector (or other) routing protocol

Hierarchical routing algorithms

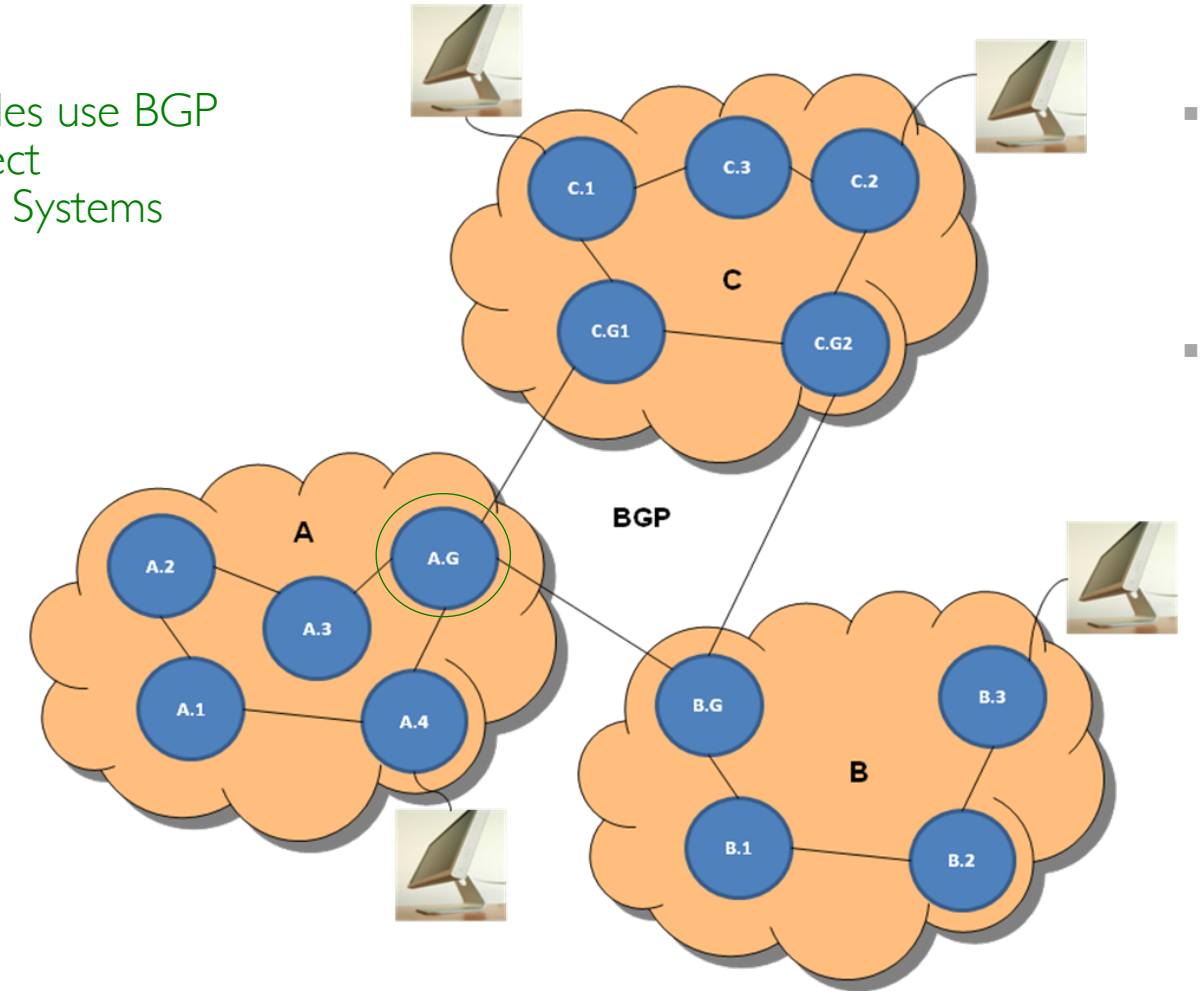
- Gateway nodes use BGP to interconnect Autonomous Systems



- ASs are assigned numbers by IANA/ ARIN
- Nodes within an AS use Link State or Distance Vector (or other) routing protocol

Hierarchical routing algorithms

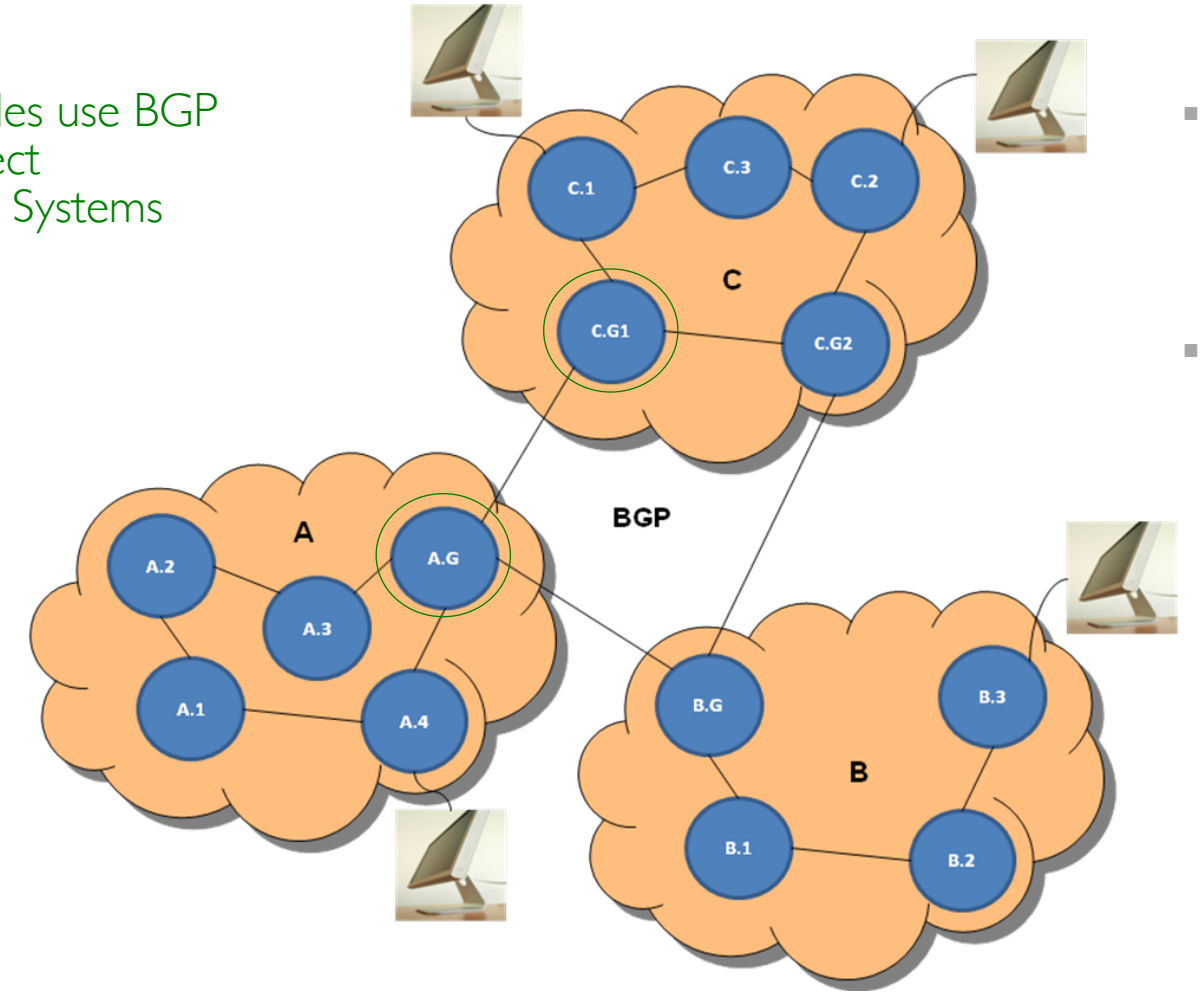
- Gateway nodes use BGP to interconnect Autonomous Systems



- ASs are assigned numbers by IANA/ ARIN
- Nodes within an AS use Link State or Distance Vector (or other) routing protocol

Hierarchical routing algorithms

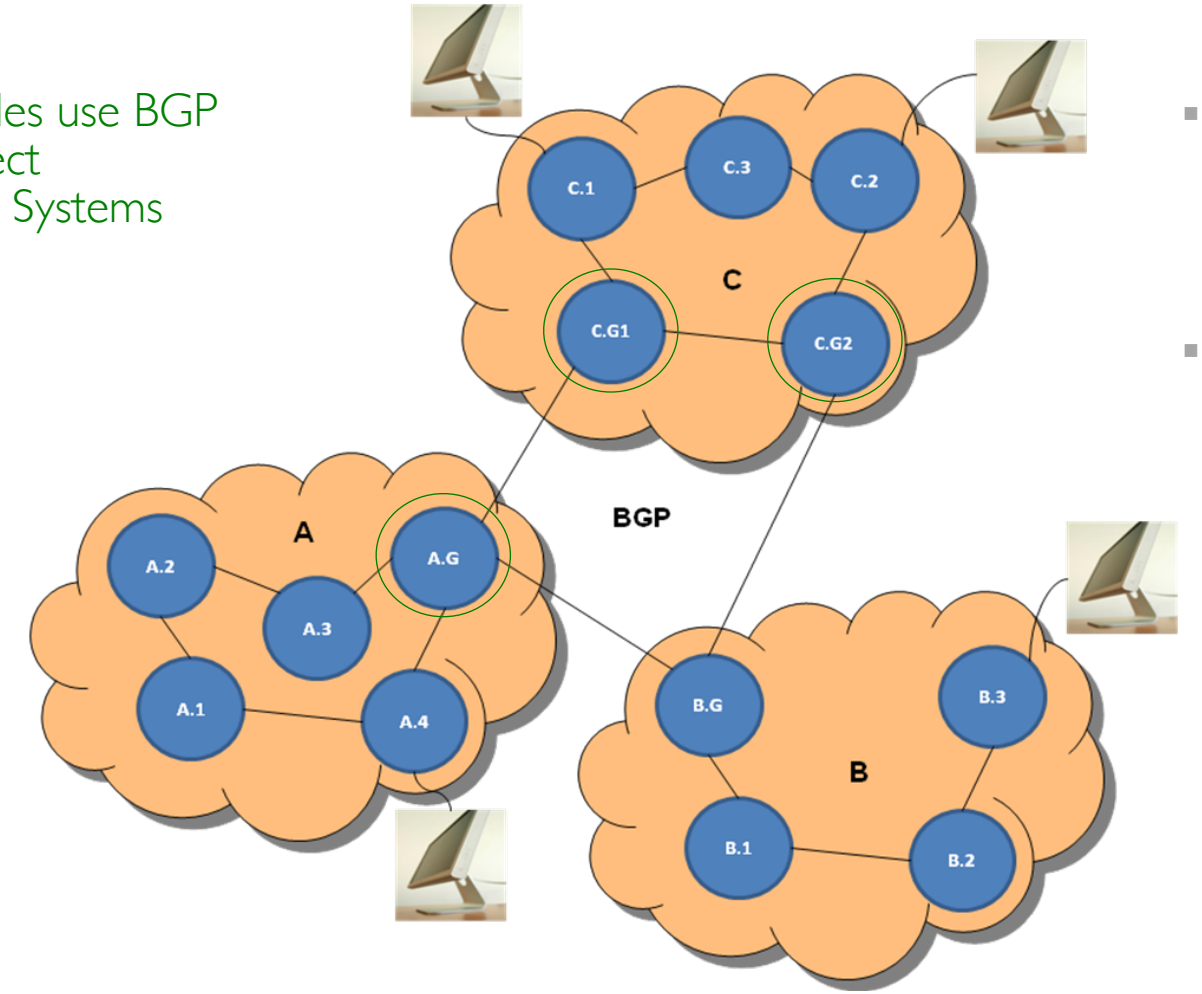
- Gateway nodes use BGP to interconnect Autonomous Systems



- ASs are assigned numbers by IANA/ ARIN
- Nodes within an AS use Link State or Distance Vector (or other) routing protocol

Hierarchical routing algorithms

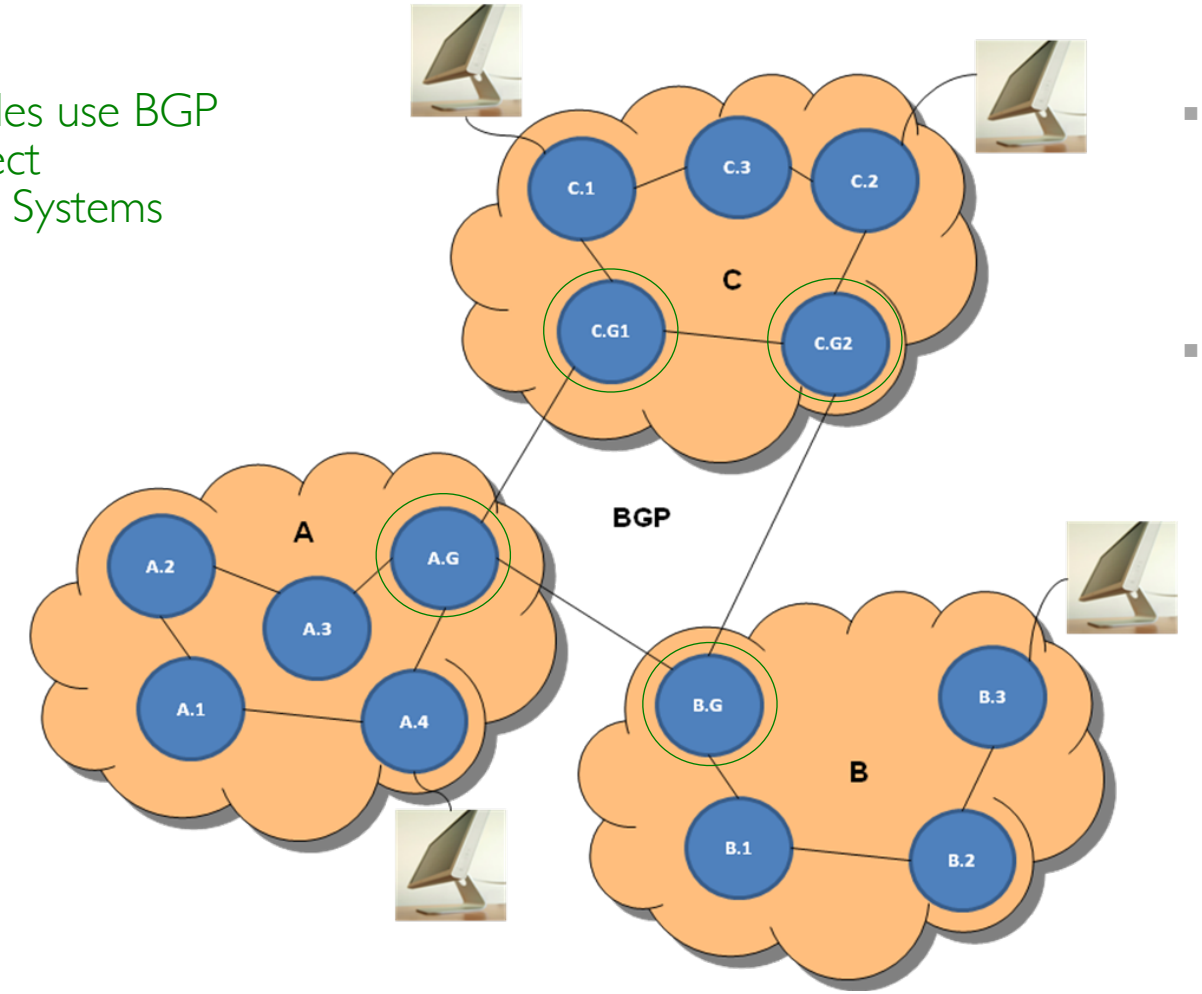
- Gateway nodes use BGP to interconnect Autonomous Systems



- ASs are assigned numbers by IANA/ ARIN
- Nodes within an AS use Link State or Distance Vector (or other) routing protocol

Hierarchical routing algorithms

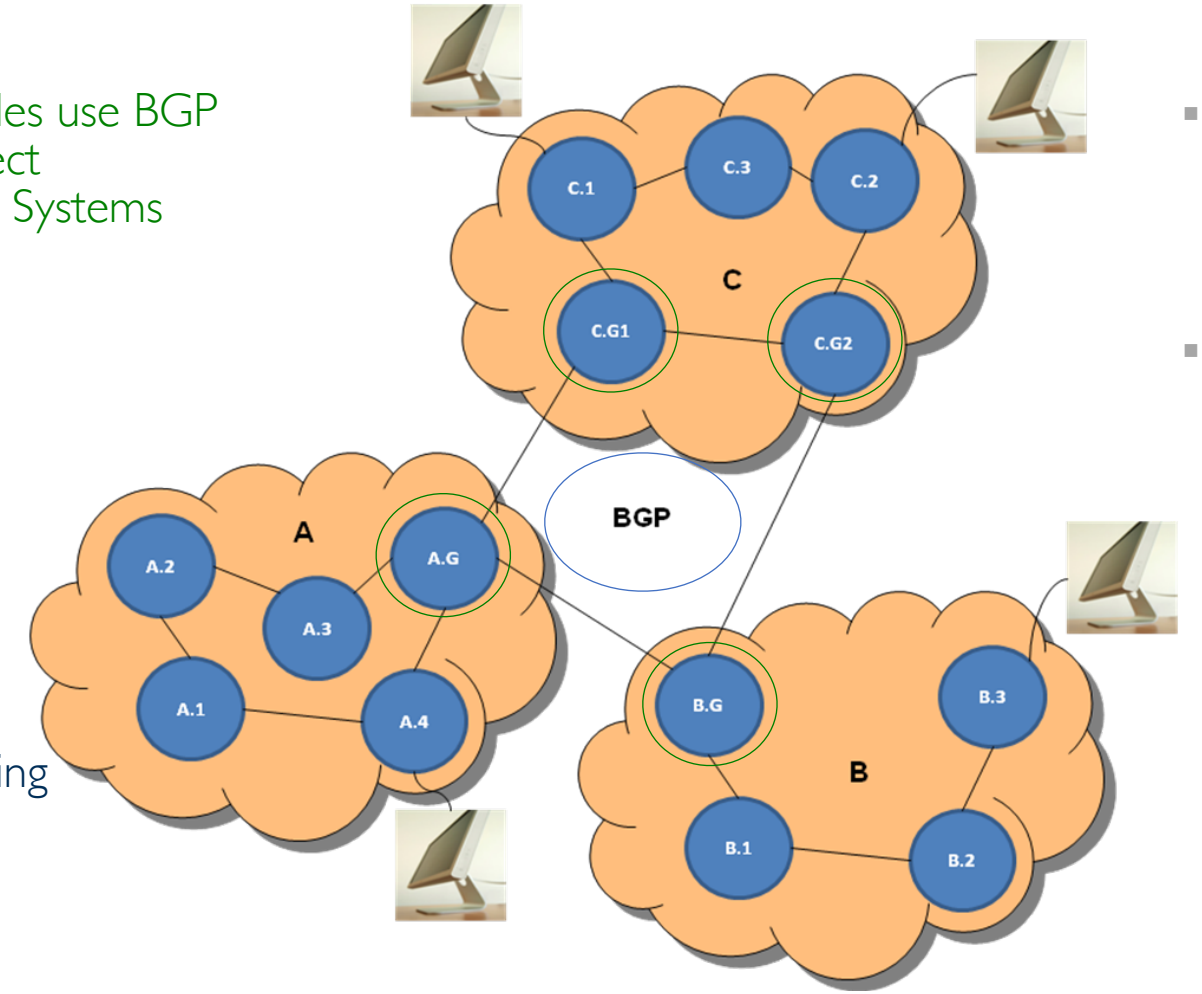
- Gateway nodes use BGP to interconnect Autonomous Systems



- ASs are assigned numbers by IANA/ ARIN
- Nodes within an AS use Link State or Distance Vector (or other) routing protocol

Hierarchical routing algorithms

- Gateway nodes use BGP to interconnect Autonomous Systems
- We use BGP as the internet inter-AS routing protocol

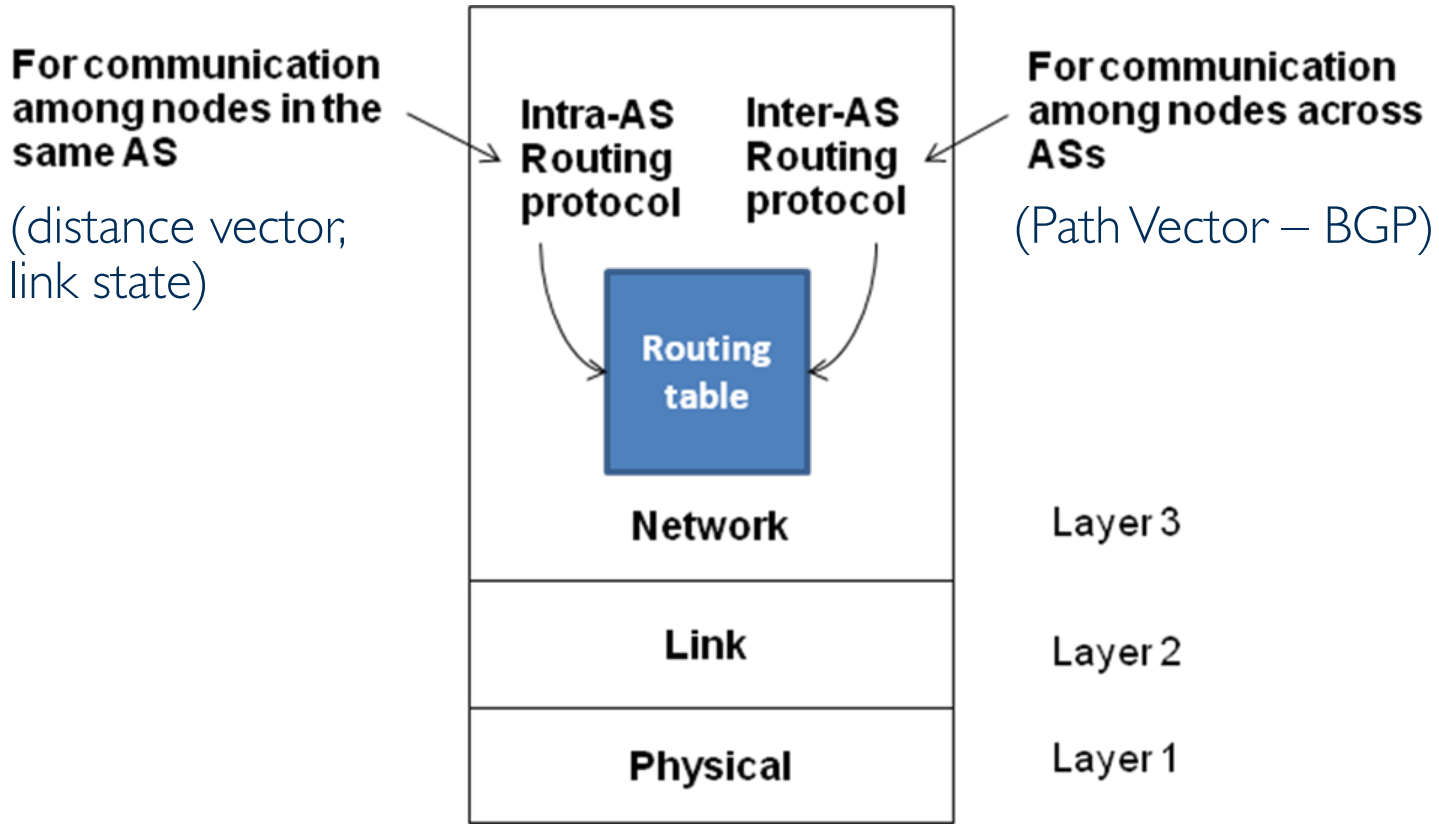


- ASs are assigned numbers by IANA/ ARIN
- Nodes within an AS use Link State or Distance Vector (or other) routing protocol

Border Gateway Protocol (BGP)

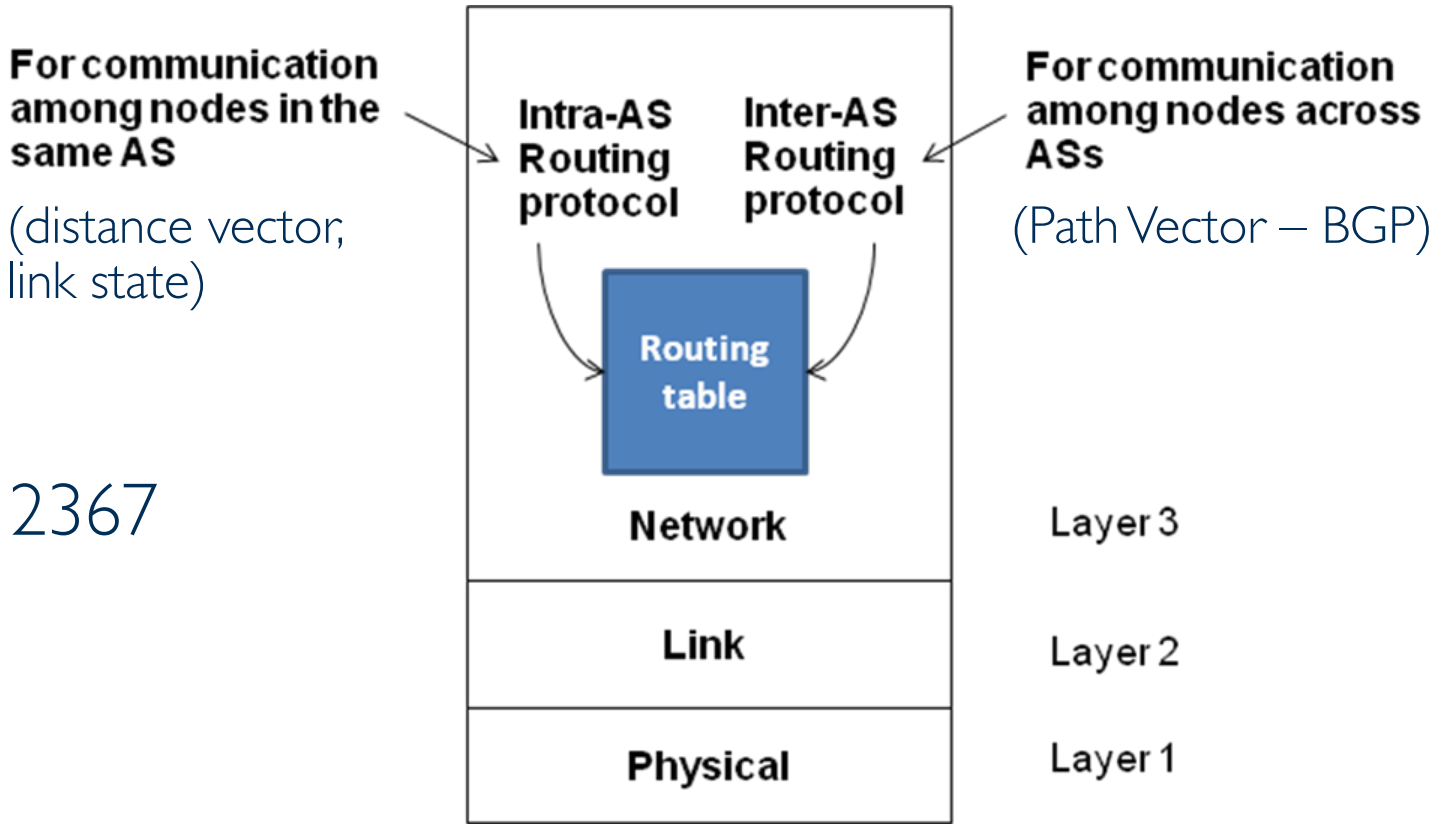
- BGP is a path-vector protocol. (We're not going into detail about that.)
- The gateway routers send path-vector messages to advertise the reachability of networks.
- Each router that receives a path vector message must verify the advertised path according to its policy.
- If the message is compliant, the router modifies its routing table and the message before sending the message to the next neighbor.
  - It modifies the routing table to maintain the autonomous systems that are traversed in order to reach the destination system.
  - It modifies the message to add its AS number and to replace the next router entry with its identification.

Routing at GT

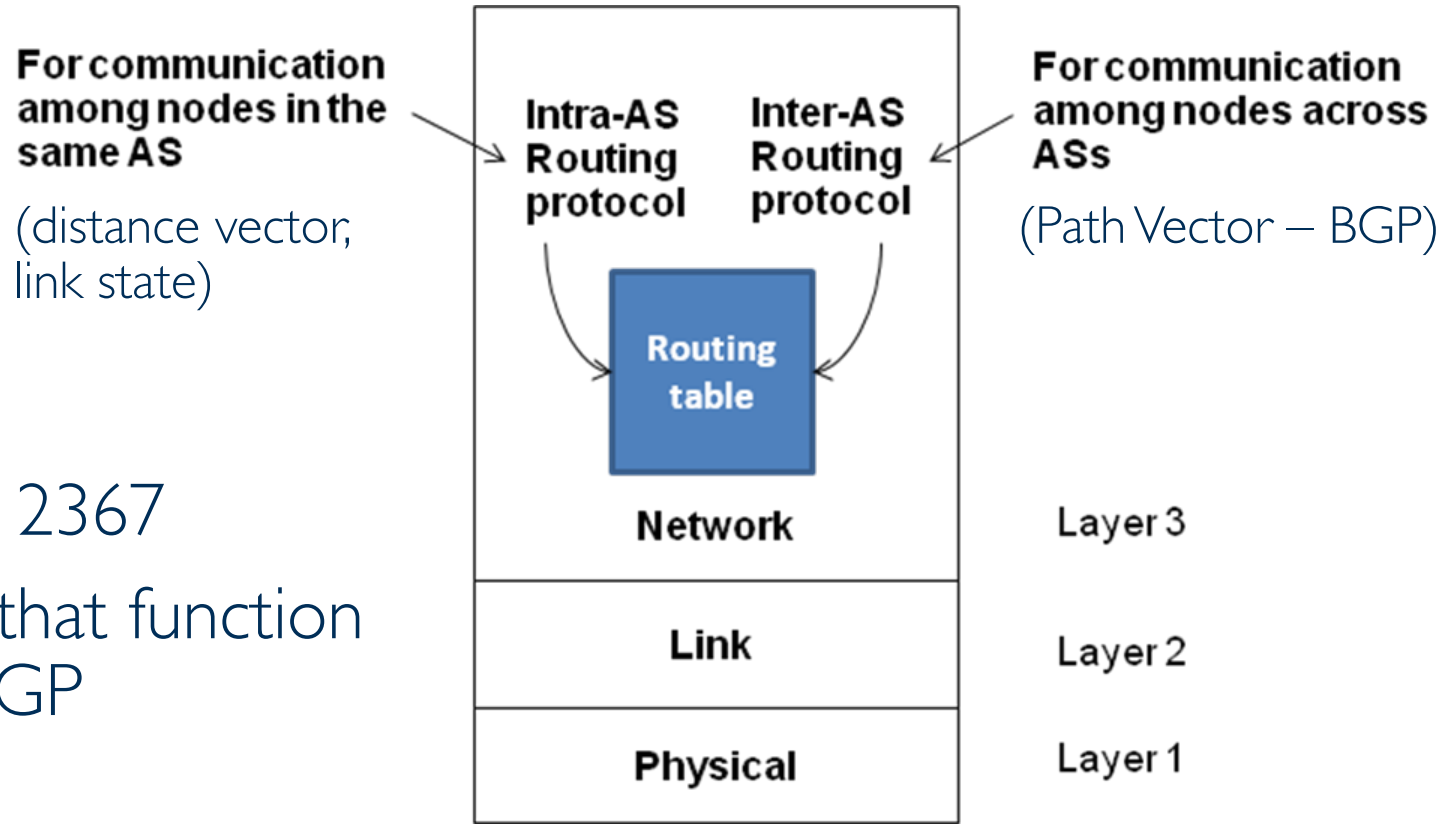


Routing at GT

- The GT campus network is AS 2367

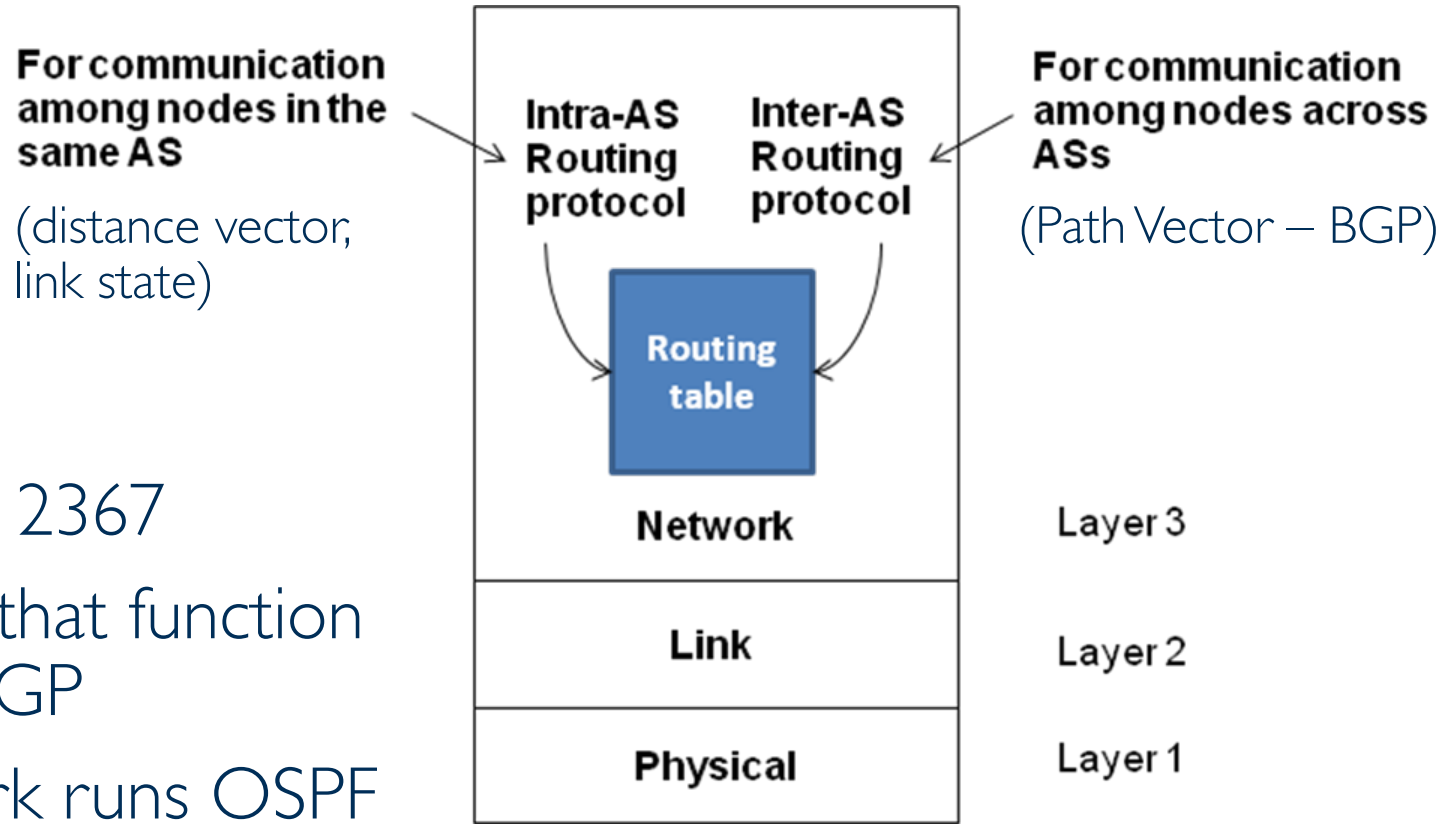


Routing at GT



- The GT campus network is AS 2367
- There are two Juniper routers that function as Gateway Routers and run BGP

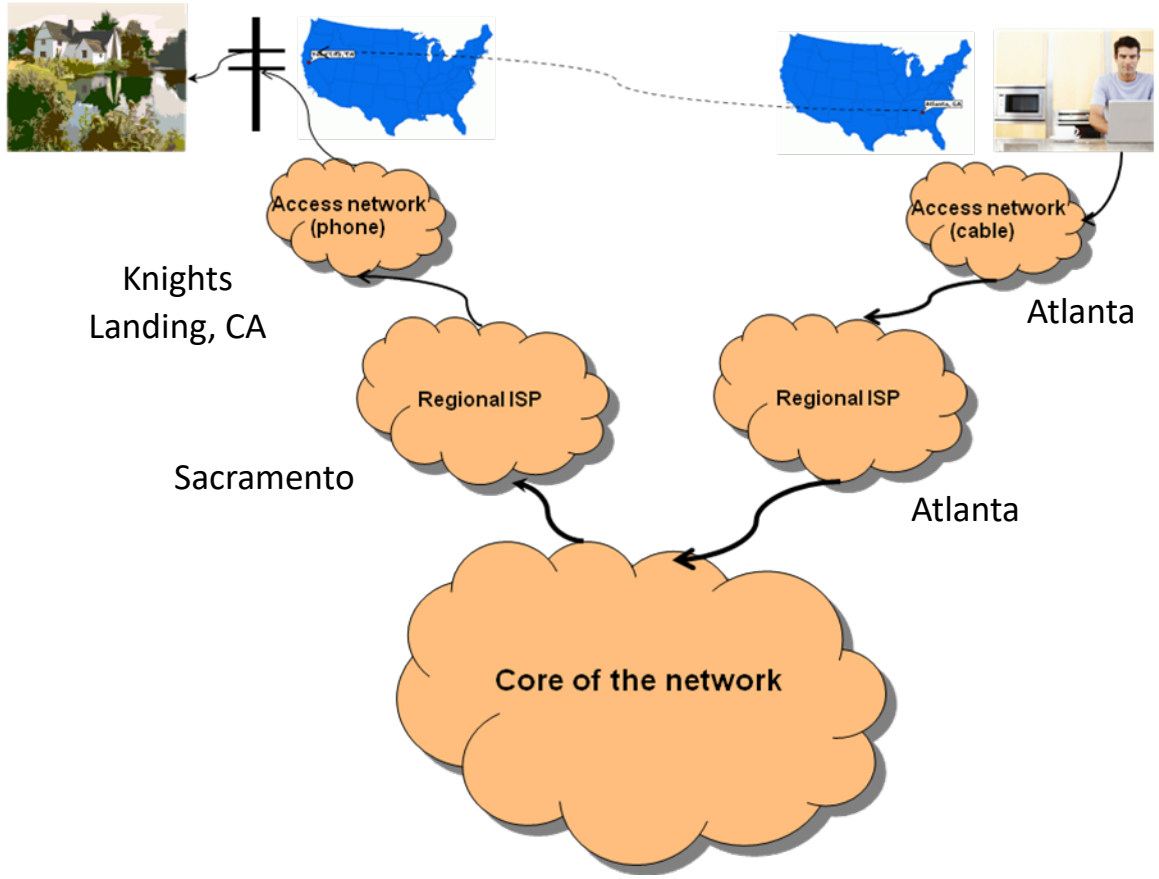
Routing at GT



- The GT campus network is AS 2367
- There are two Juniper routers that function as Gateway Routers and run BGP
- The rest of the campus network runs OSPF (a link-state routing protocol)

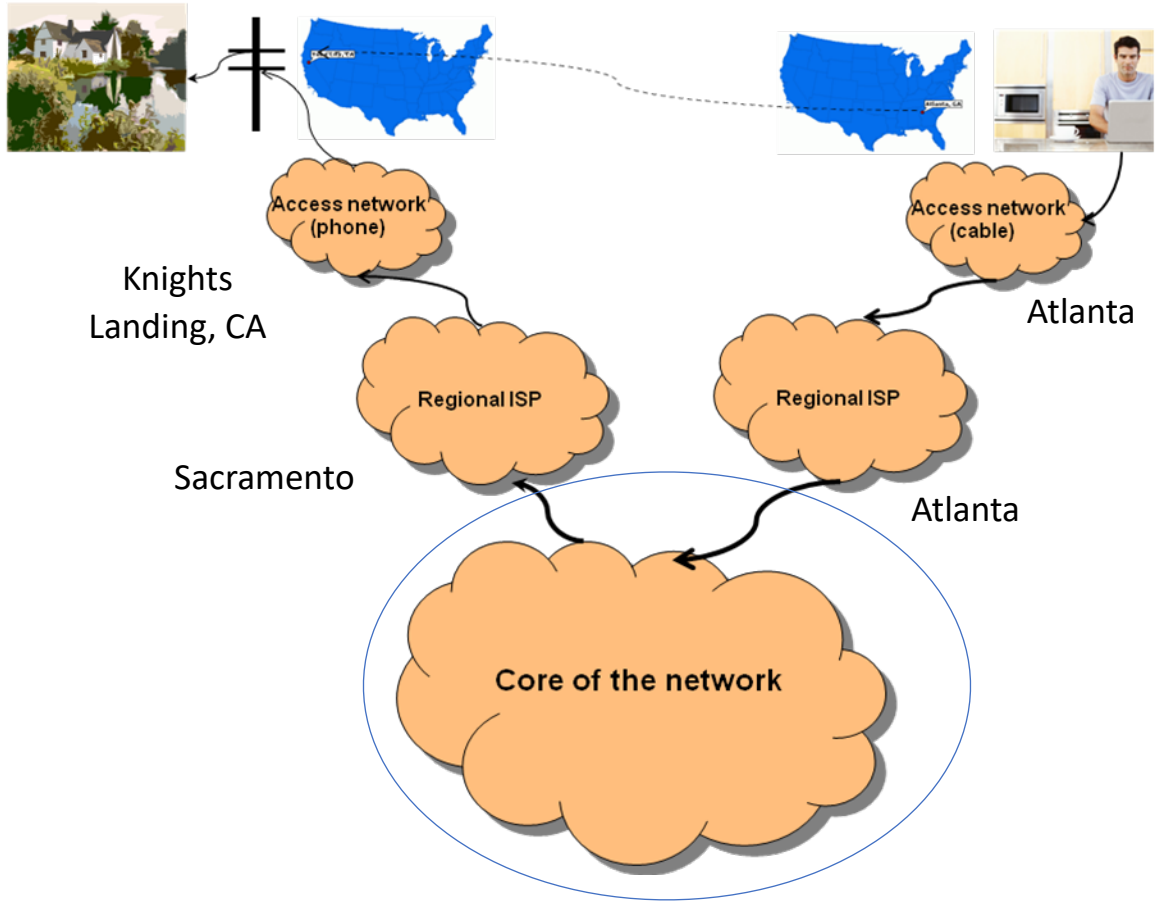
Remember this slide?

- Now consider an email from Joe to his grandmother



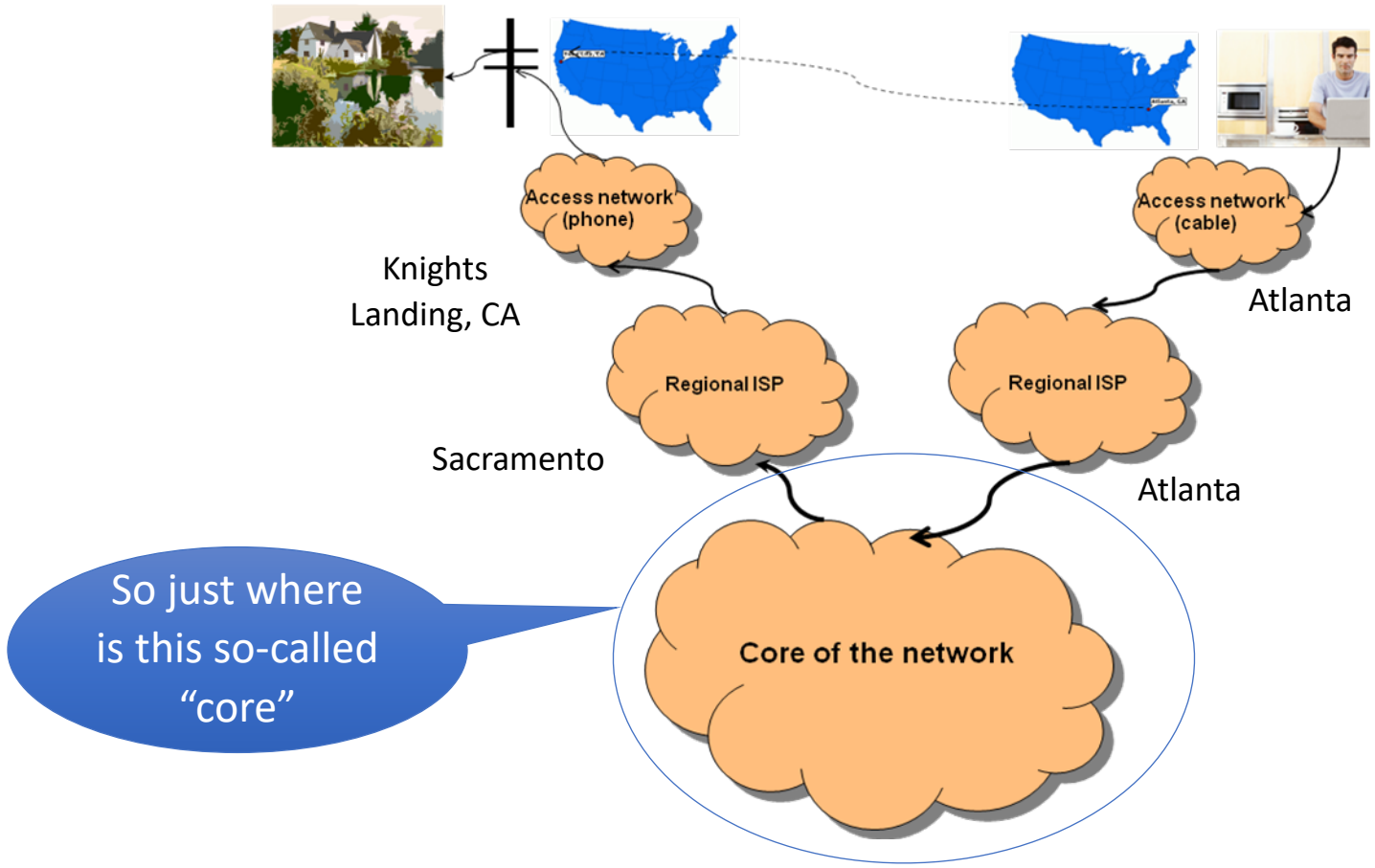
Remember this slide?

- Now consider an email from Joe to his grandmother



Remember this slide?

- Now consider an email from Joe to his grandmother



There really is no internet core …

- Instead there are dozens of Internet Exchange Points (IXPs) where groups of Internet Service Providers (ISPs) interconnect their networks

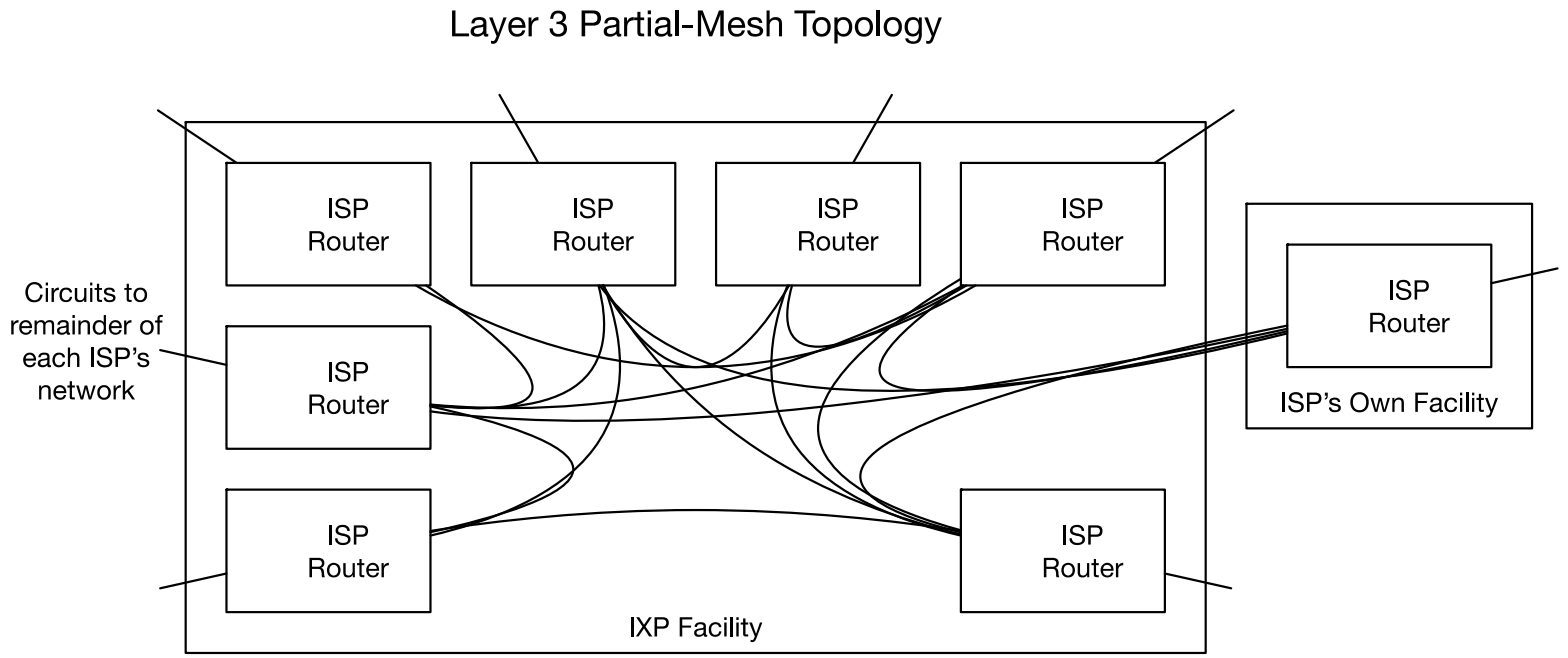
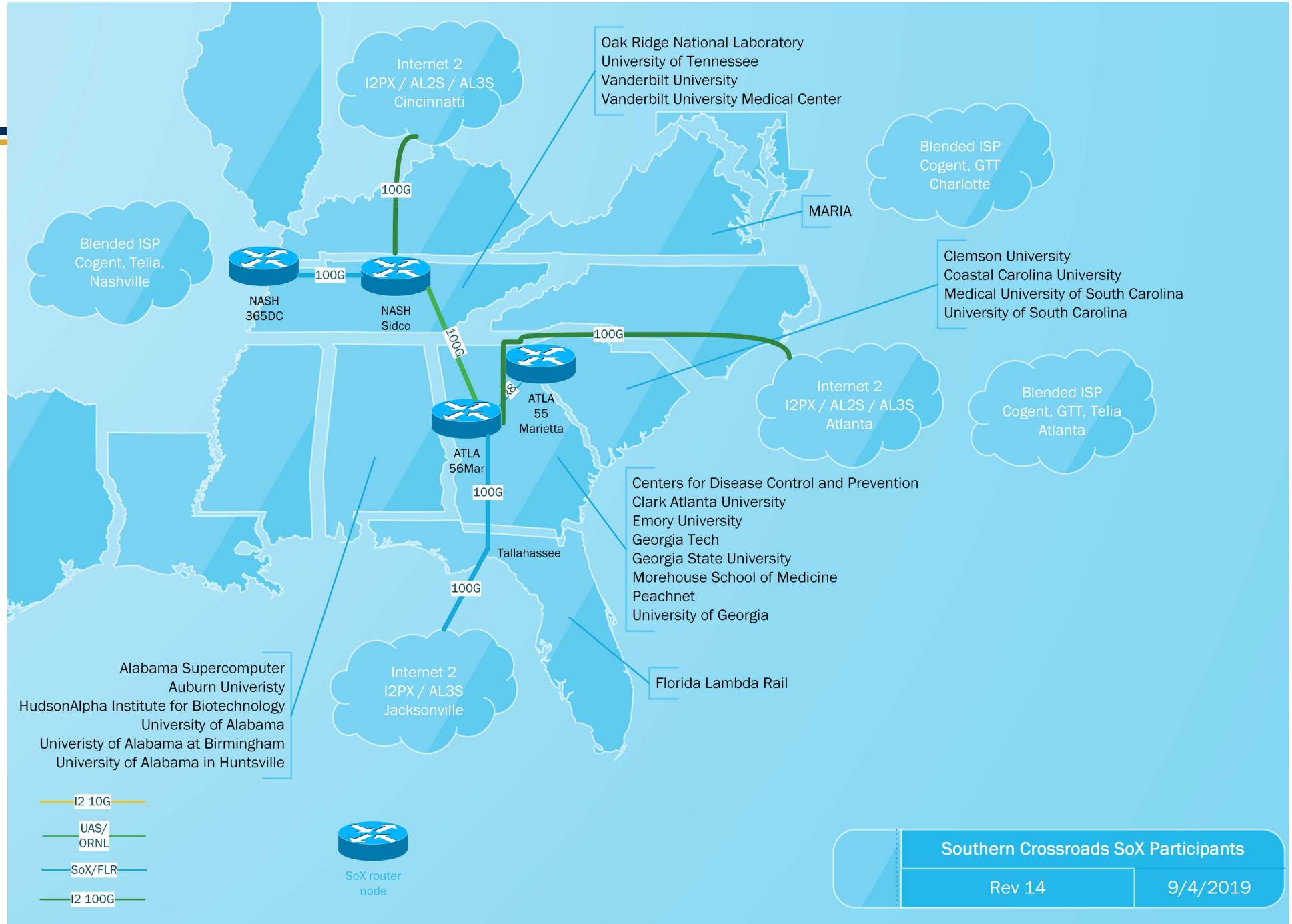


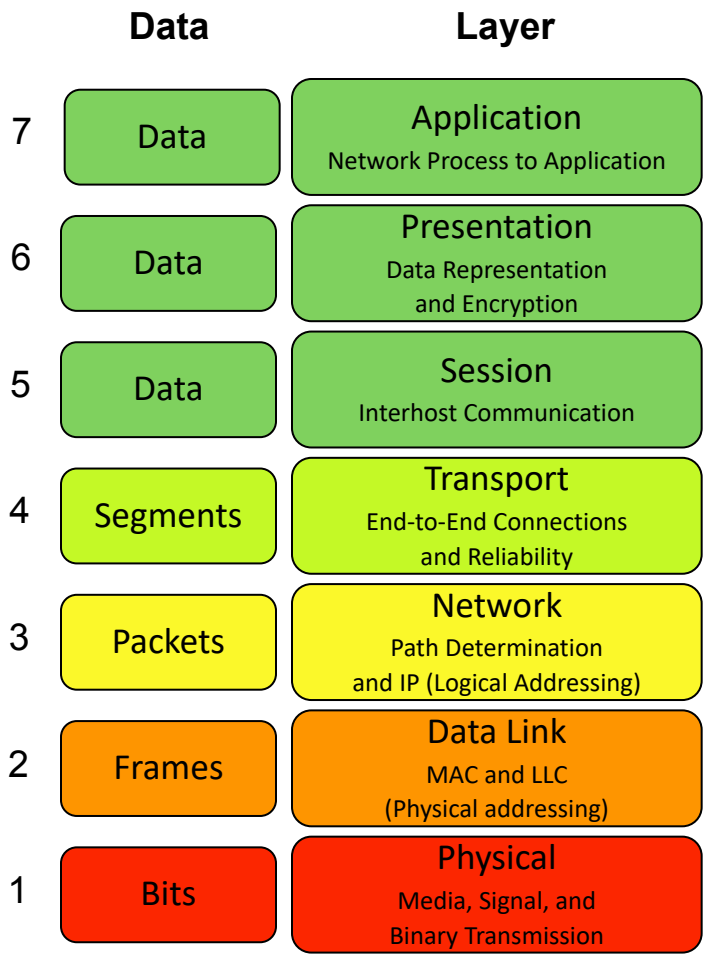
Diagram by Bill Woodcock, Packet Clearing House, rights released for use on Wikipedia

There really is no internet core …

- The ISPs peer using BGP such that all the gateway routers know multiple routes to every regional ISP
- You can begin to see that the Internet has a highly distributed "core" created by multiple IXPs
- There are at least four IXPs in Atlanta; GT participates in and operates an ISP, SoX, which provides network access for about 3 dozen southeastern universities and research centers
- The diagram on the next slide shows just the IXPs in which SoX participates. Note that traffic is allowed to traverse SoX networks to get from one ISP to another.
- The diagram doesn't show the SoX peering with other entities such as Google, AT&T, NASA, Dept of Energy, and Comcast

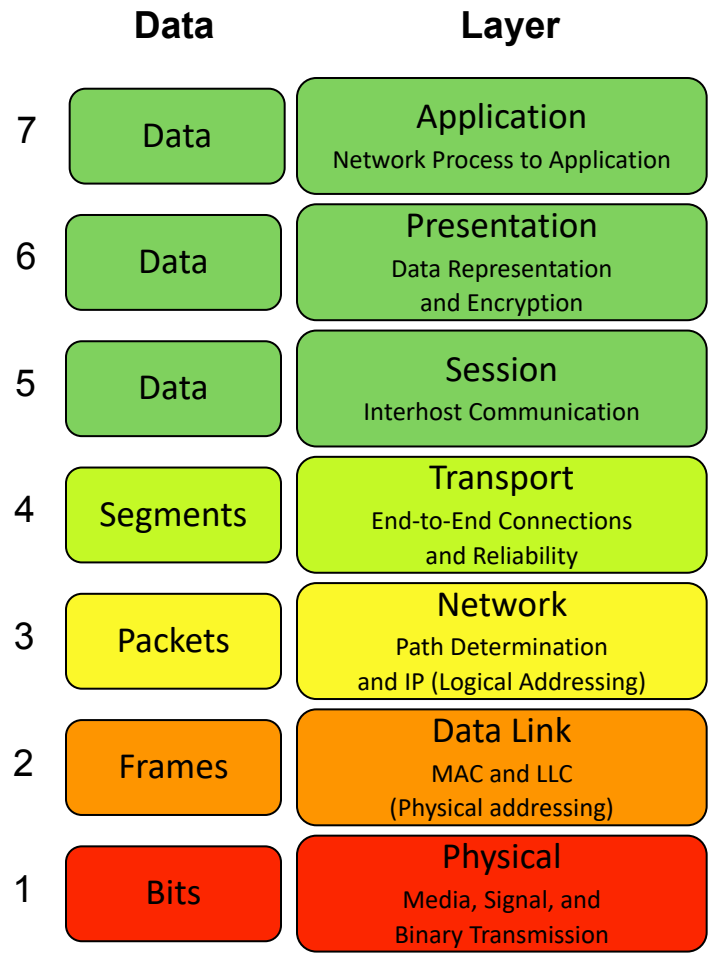

Internet 2
I2PX / AL2S / AL3S
Cincinnati
Oak Ridge National Laboratory
University of Tennessee
Vanderbilt University
Vanderbilt University Medical Center
Blended ISP
Cogent, GTT
Charlotte
MARIA
Blended ISP
Cogent, Telia,
Nashville
100G
NASH
365DC
NASH
Sidco
Clemson University
Coastal Carolina University
Medical University of South Carolina
University of South Carolina
100G
100G
ATLA
55
Marietta
Internet 2
I2PX / AL2S / AL3S
Atlanta
Blended ISP
Cogent, GTT, Telia
Atlanta
ATLA
56Mar
100G
Centers for Disease Control and Prevention
Clark Atlanta University
Emory University
Georgia Tech
Georgia State University
Morehouse School of Medicine
Peachnet
University of Georgia
Tallahassee
100G
Internet 2
I2PX / AL3S
Jacksonville
Florida Lambda Rail
Alabama Supercomputer
Auburn Univeristy
HudsonAlpha Institute for Biotechnology
University of Alabama
Univeristy of Alabama at Birmingham
University of Alabama in Huntsville
I2 10G
UAS/
ORNL
SoX/FLR
I2 100G
SoX router
node
Southern Crossroads SoX Participants
Rev 14
9/4/2019

Summary


Data
Layer
7
Data
Application
Network Process to Application
6
Data
Presentation
Data Representation
and Encryption
5
Data
Session
Interhost Communication
4
Segments
Transport
End-to-End Connections
and Reliability
3
Packets
Network
Path Determination
and IP (Logical Addressing)
2
Frames
Data Link
MAC and LLC
(Physical addressing)
1
Bits
Physical
Media, Signal, and
Binary Transmission

Summary

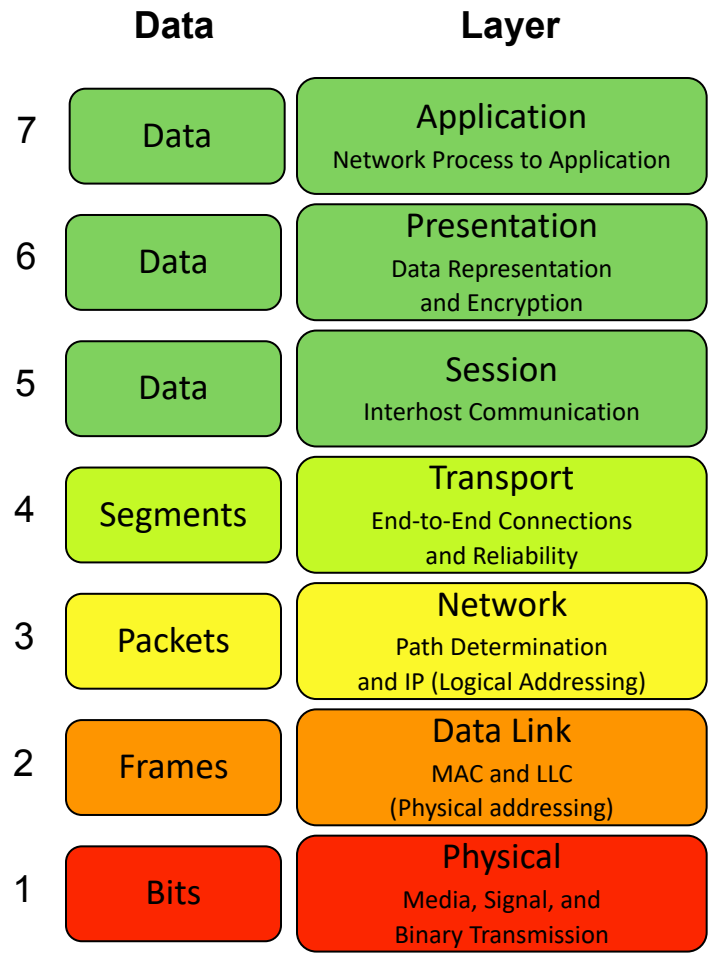
Sockets, http



Summary

Sockets, http

RPC

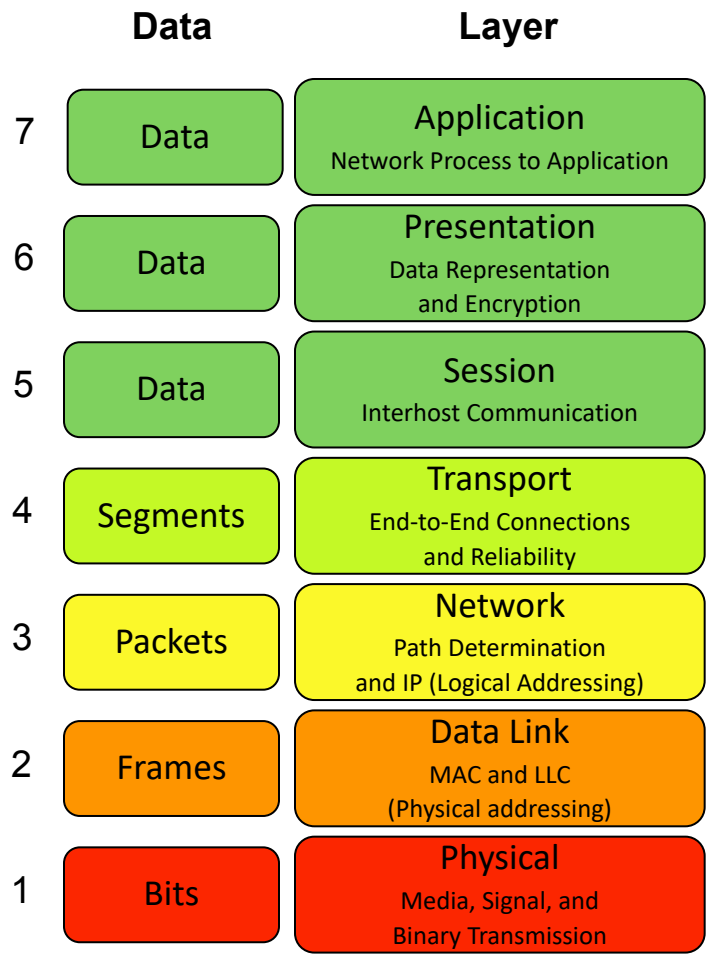


Summary

Sockets, http

RPC

TCP



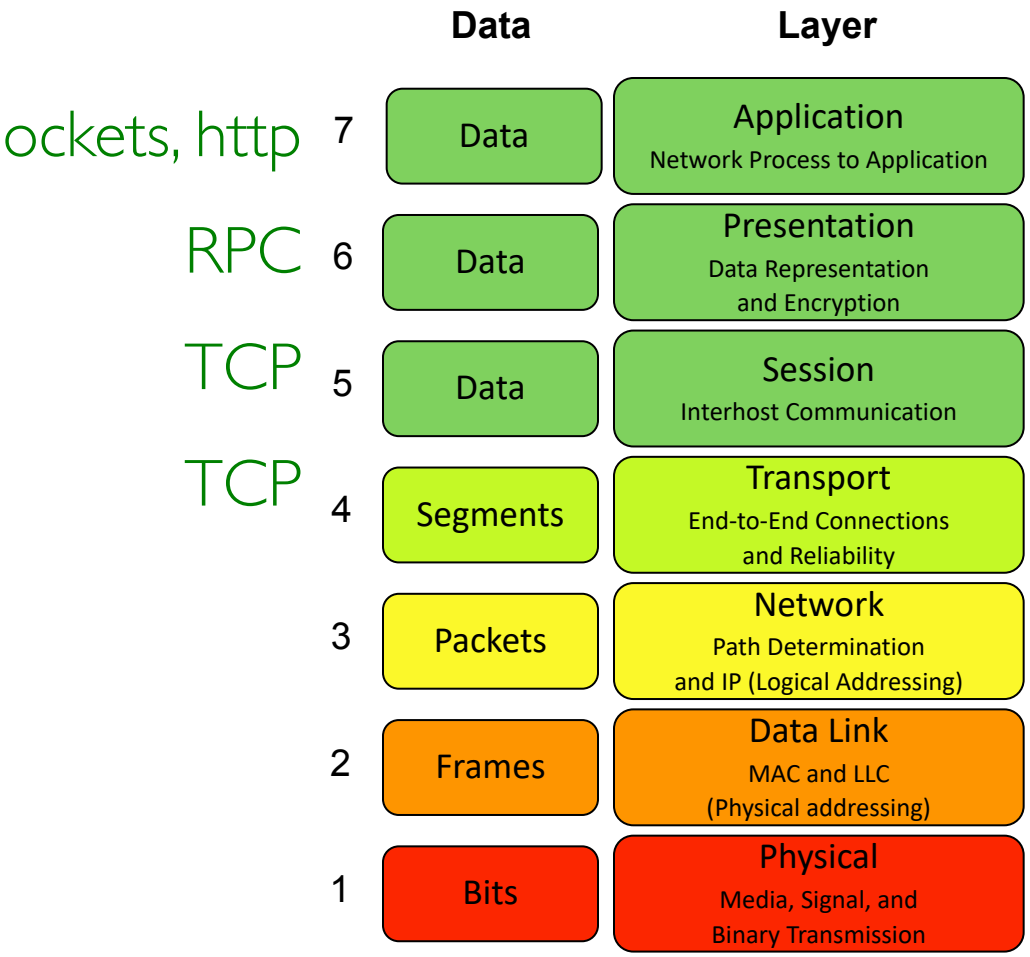
Summary

Sockets, http

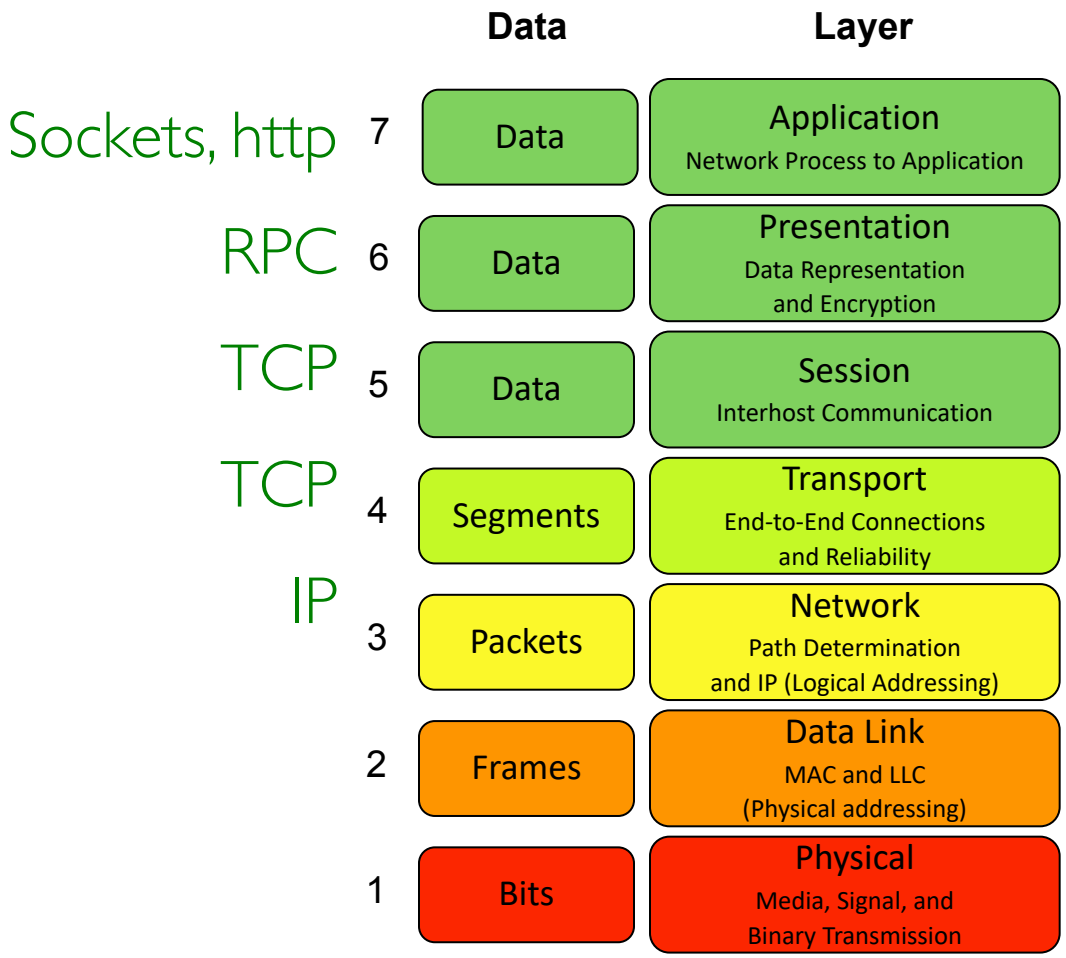
RPC

TCP

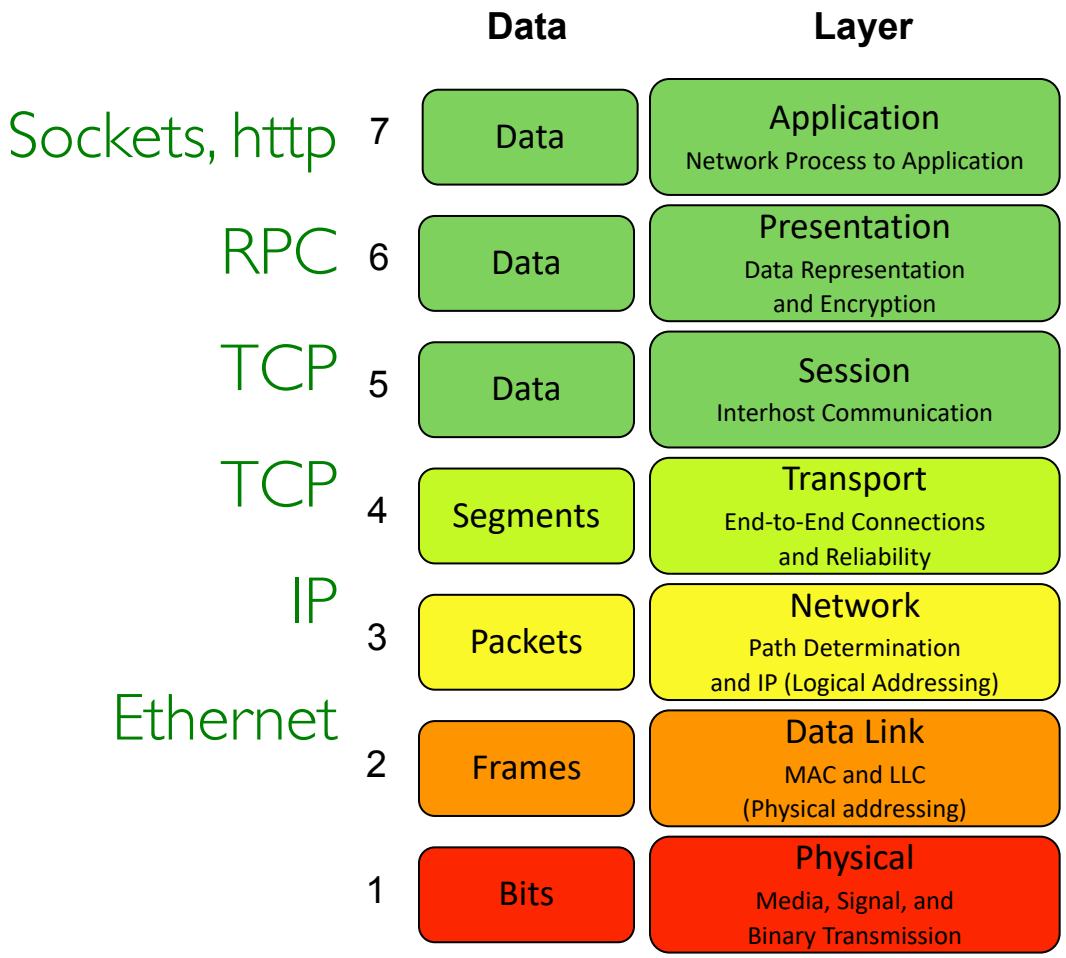
TCP



Summary


Data
Layer
Sockets, http
RPC
TCP
TCP
IP
7
6
5
4
3
2
1
Data
Data
Data
Segments
Packets
Frames
Bits
Application
Network Process to Application
Presentation
Data Representation
and Encryption
Session
Interhost Communication
Transport
End-to-End Connections
and Reliability
Network
Path Determination
and IP (Logical Addressing)
Data Link
MAC and LLC
(Physical addressing)
Physical
Media, Signal, and
Binary Transmission

Summary


Data
Layer
Sockets, http
RPC
TCP
TCP
IP
Ethernet
7
6
5
4
3
2
1
Data
Data
Data
Segments
Packets
Frames
Bits
Application
Network Process to Application
Presentation
Data Representation
and Encryption
Session
Interhost Communication
Transport
End-to-End Connections
and Reliability
Network
Path Determination
and IP (Logical Addressing)
Data Link
MAC and LLC
(Physical addressing)
Physical
Media, Signal, and
Binary Transmission

Summary

Sockets, http

RPC

TCP

TCP

IP

Ethernet

Wired, wireless hardware