Georgia Tech Computer Science

CS2200
Systems and Networks
Spring 2022

Lecture 26:
Disk Scheduling

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

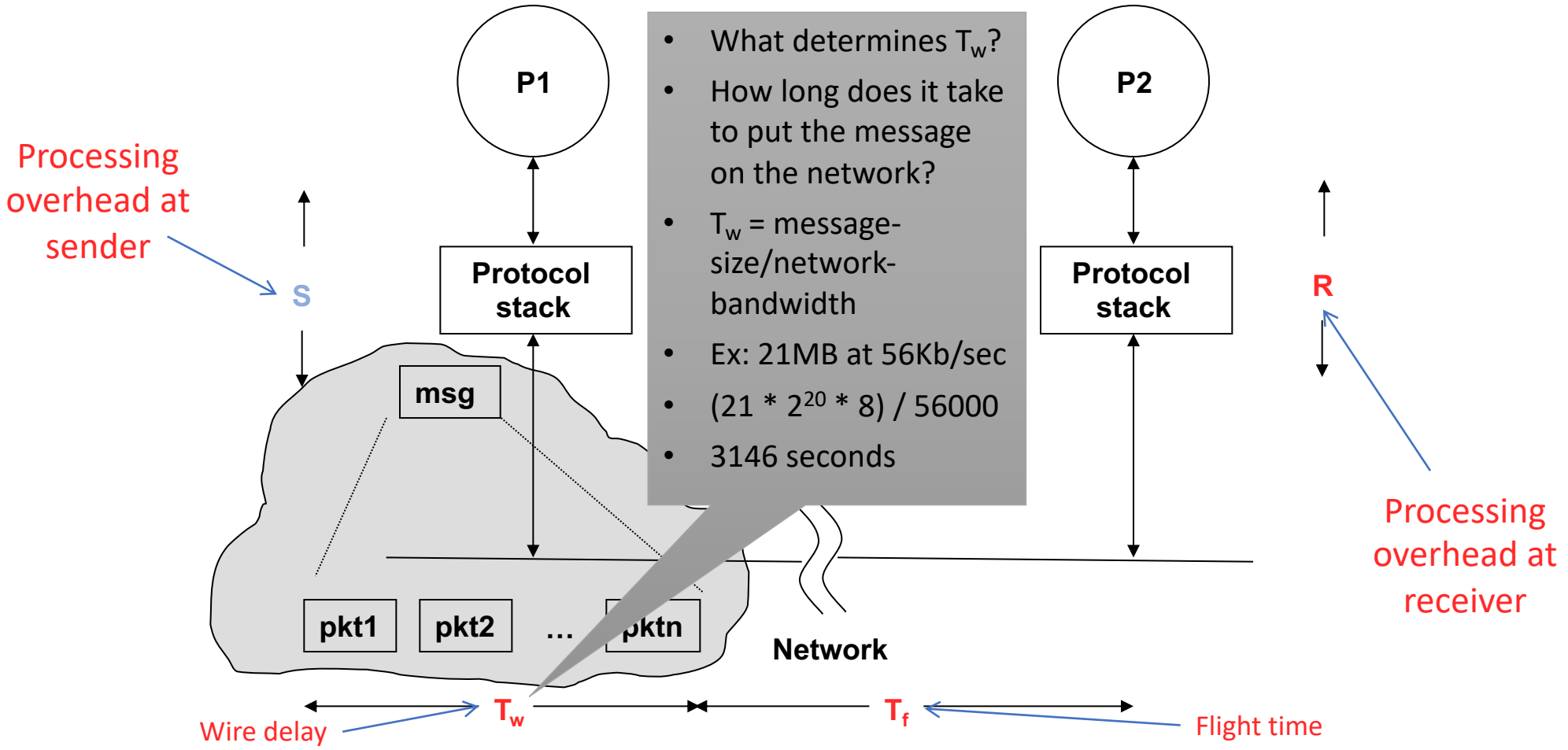
Lecture slides adapted from Bill Leahy and Charles Lively of Georgia Tech

Agenda

- Wrap up Networking
- Disk Scheduling

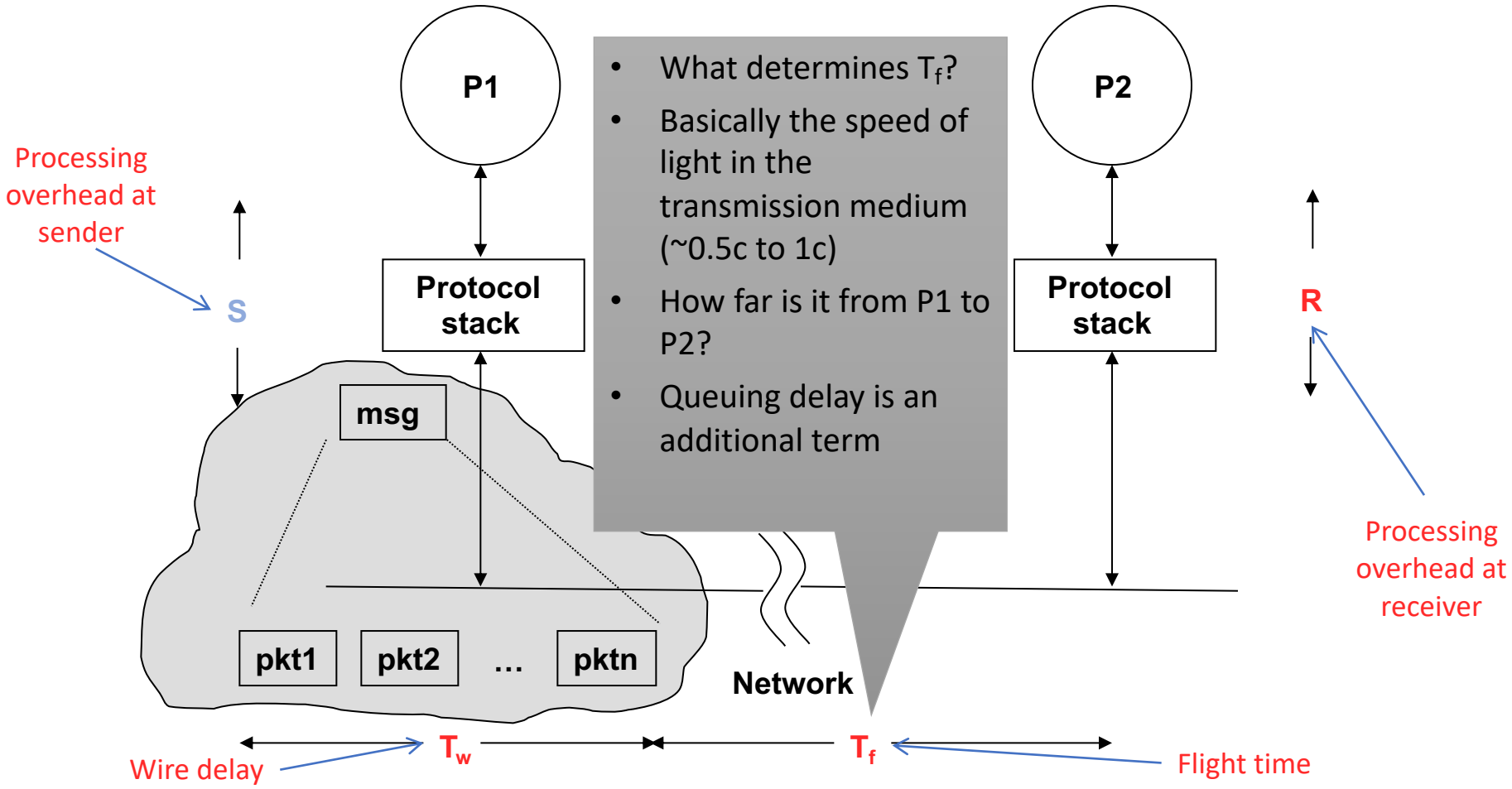
Performance metrics

Transmission time or end-to-end latency = $S+T_W+T_f+R$
Message throughput = message-size/end-to-end-latency



Performance metrics

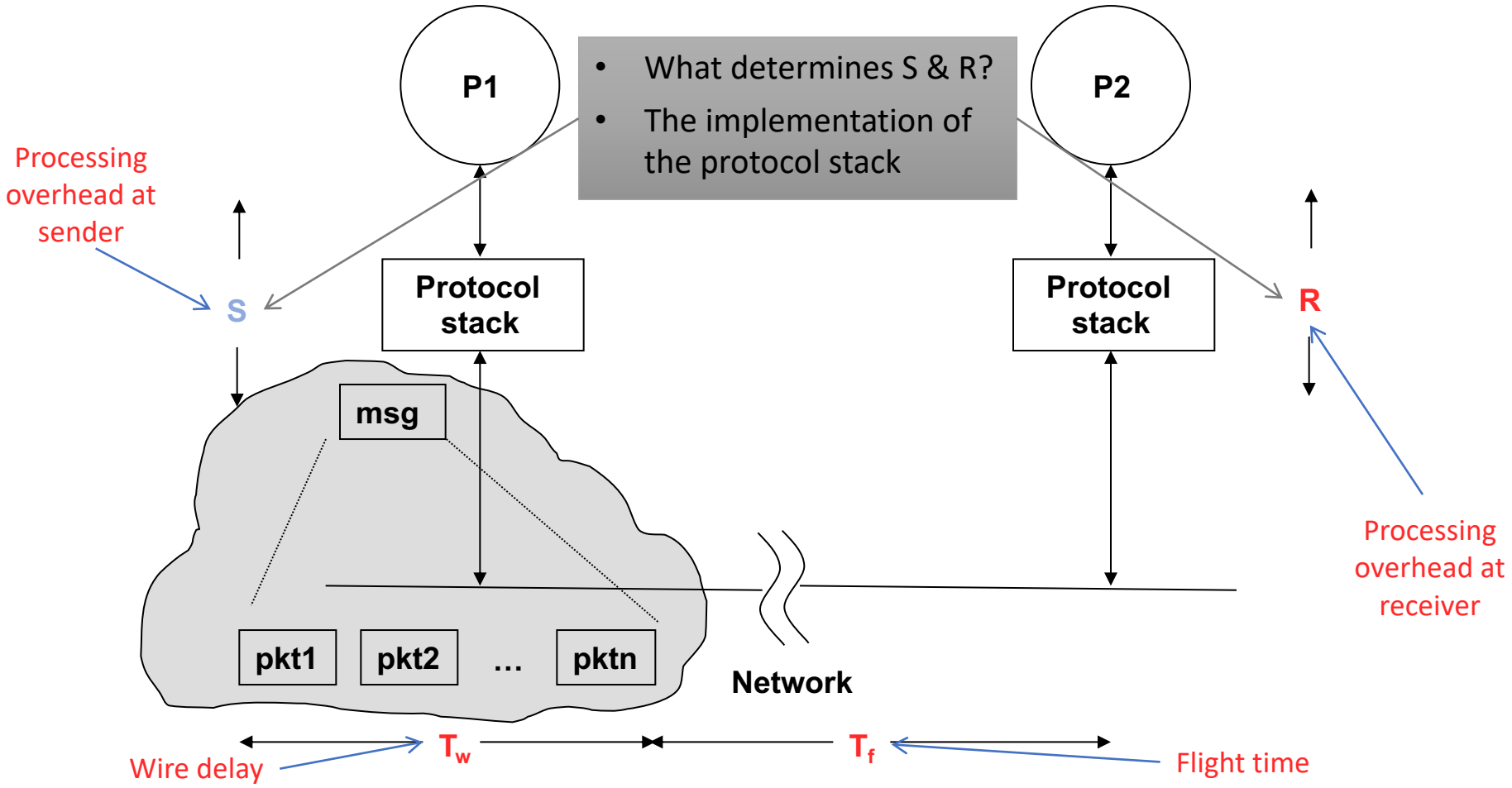
Transmission time or end-to-end latency = $S+T_W+T_f+R$
Message throughput = message-size/end-to-end-latency



- 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

Performance metrics

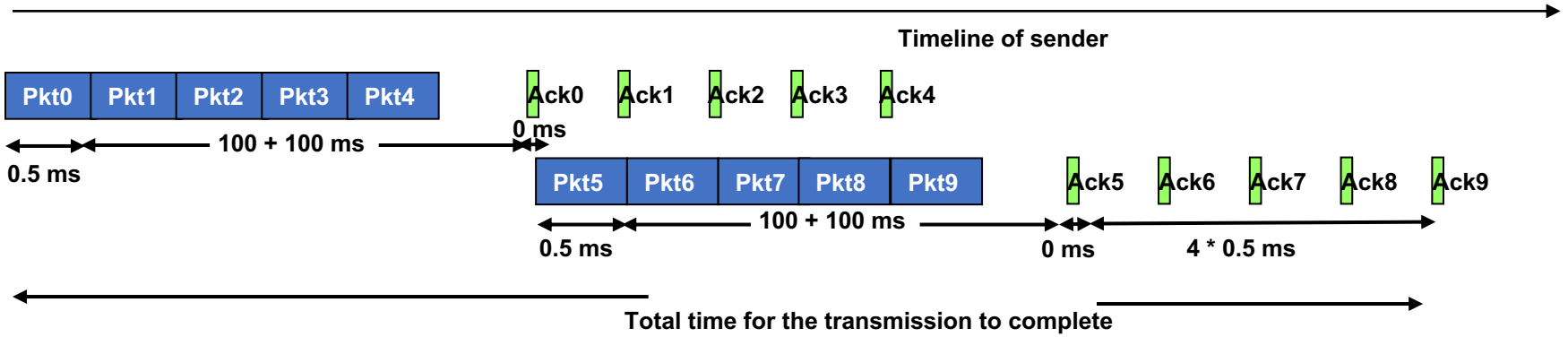
Transmission time or end-to-end latency = $S+T_W+T_f+R$
Message throughput = message-size/end-to-end-latency



Timeline example

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

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



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

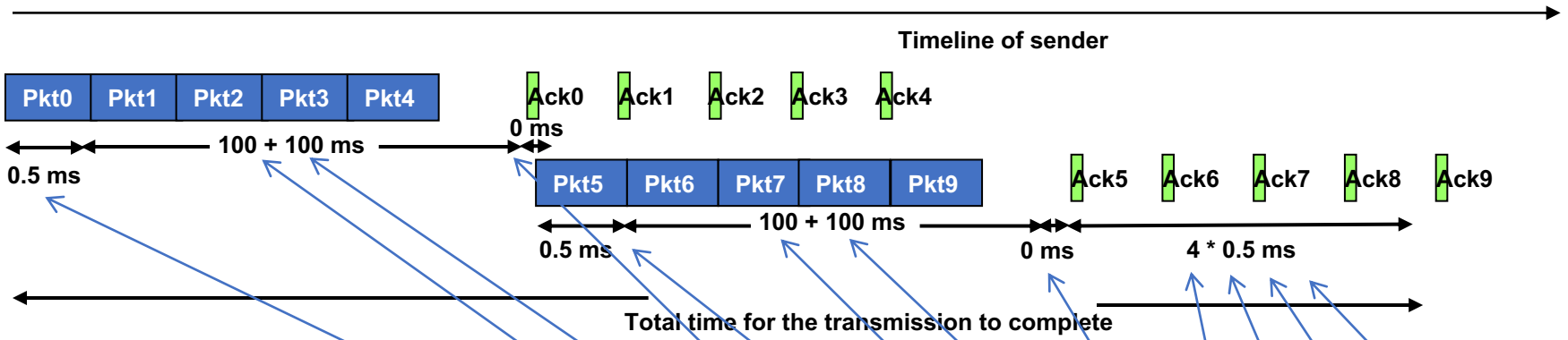
End-to-end latency for Ack0
= $S + T_w + T_f + R$
0 + 0.0 + 100 + 0
100.0 ms

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

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

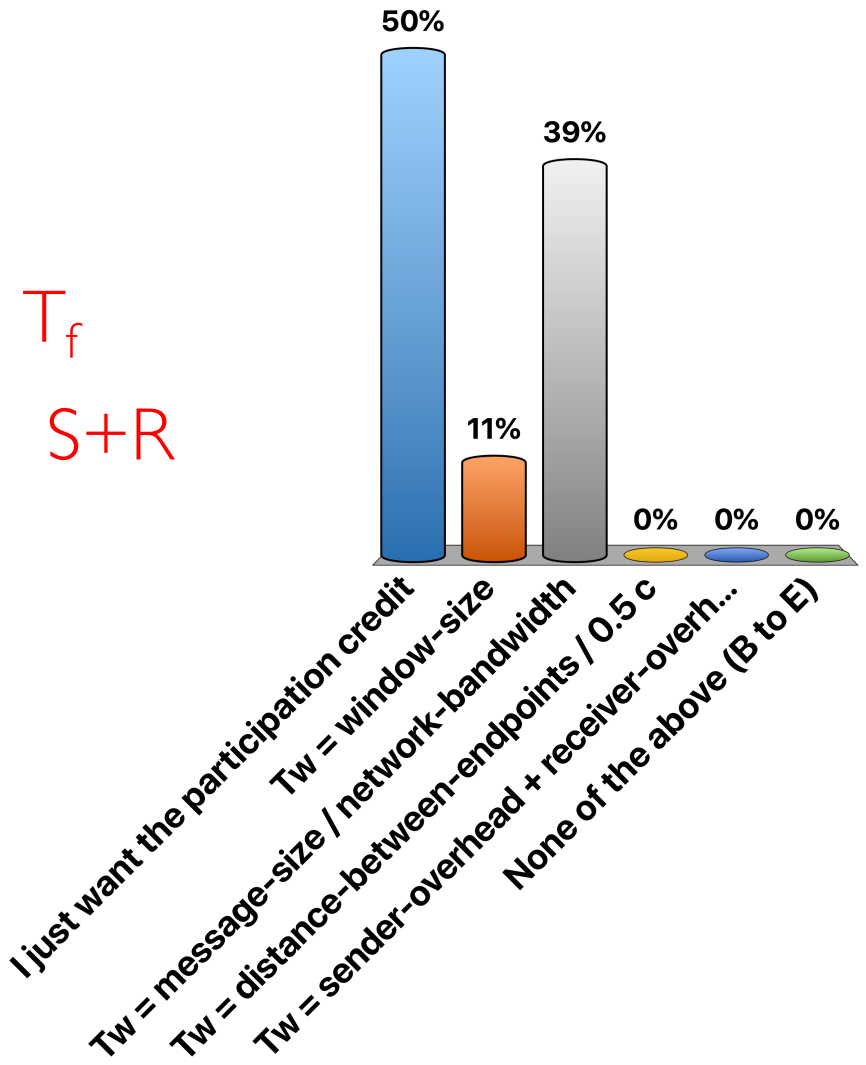
End-to-end latency for Ack0
= $S + T_w + T_f + R$
0 + 0.0 + 100 + 0
100.0 ms

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

Session ID: dag2200

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

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

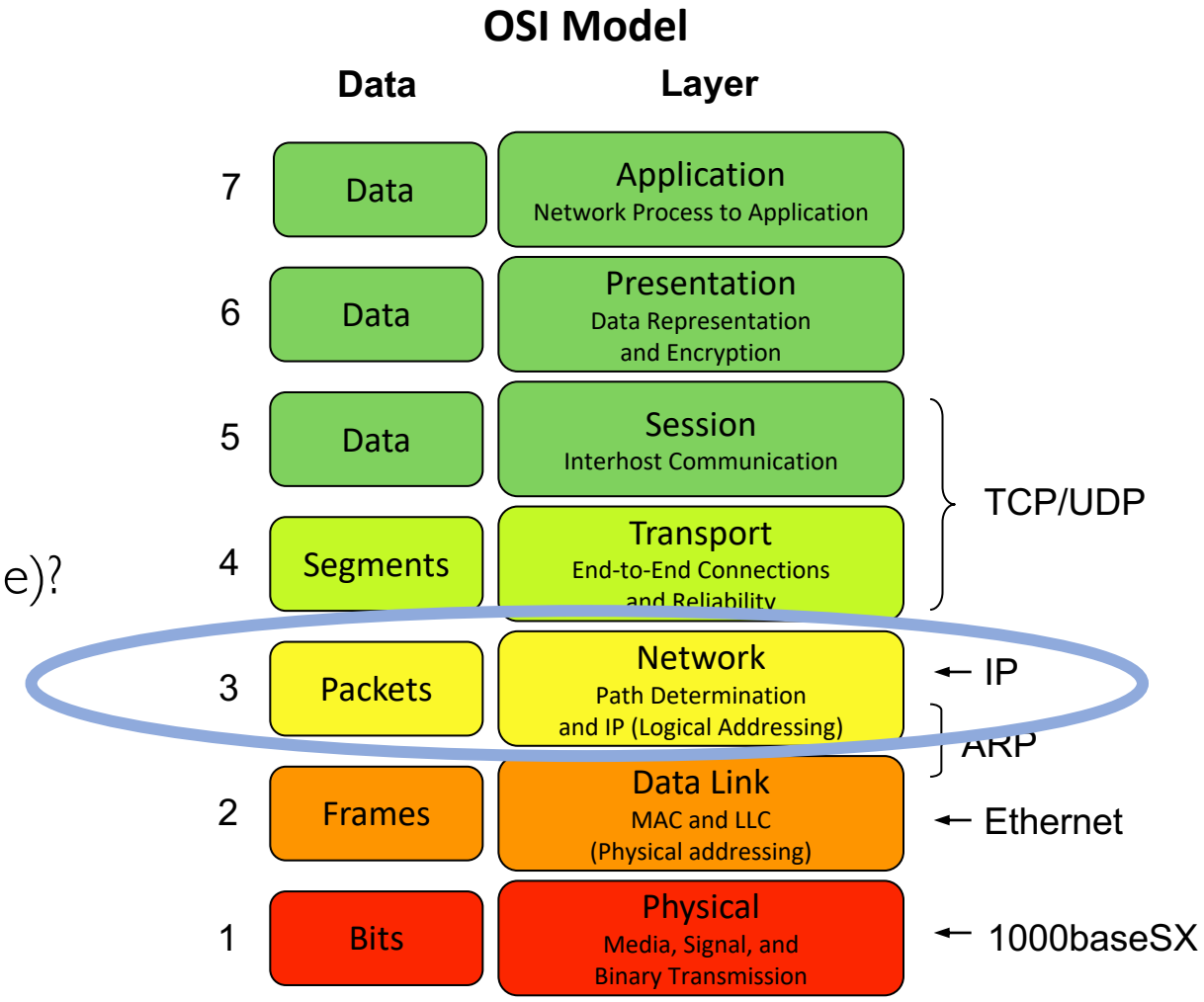


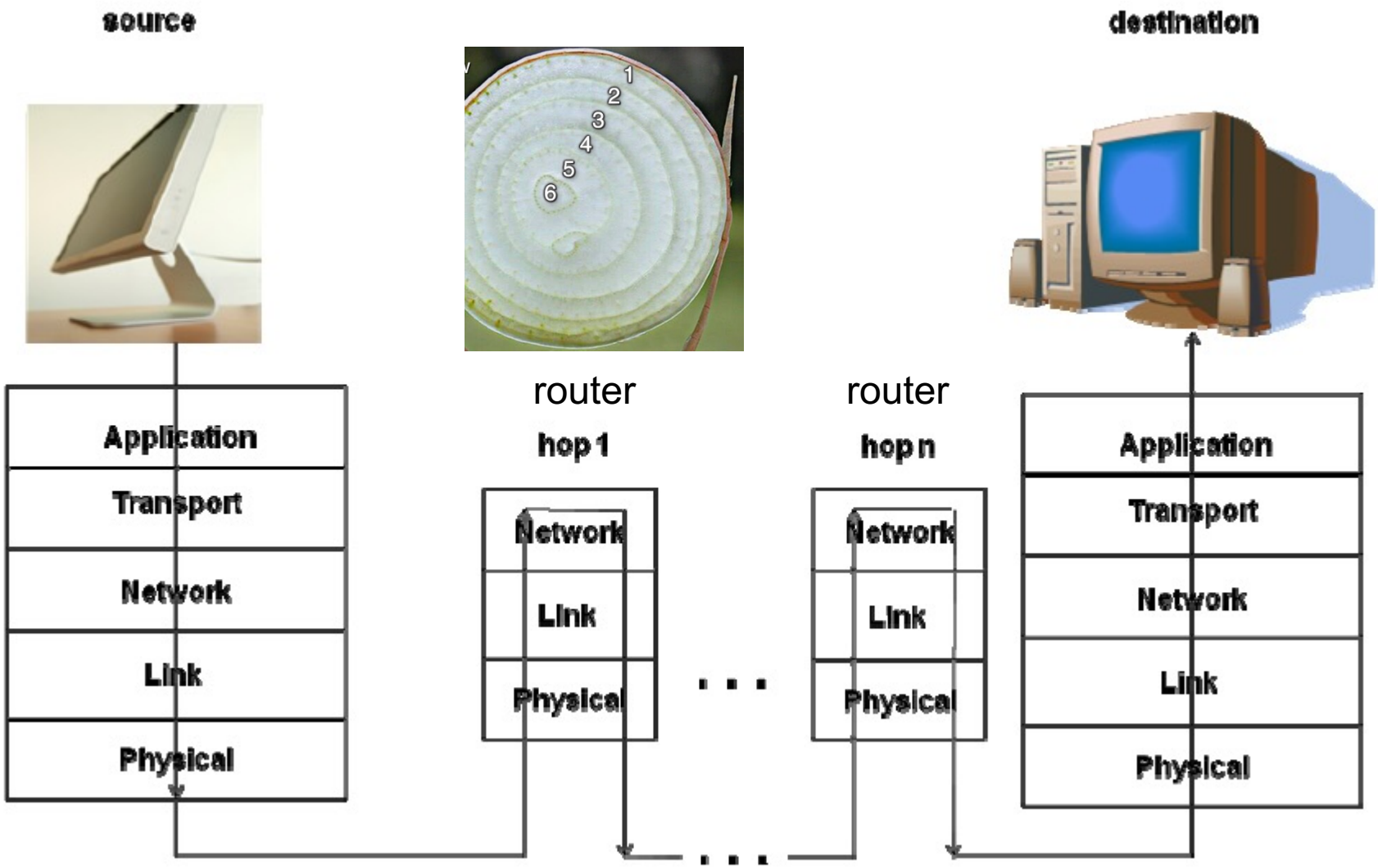
Acknowledgement example

- A transport protocol uses a window of 10 packets.
- For a message that consists of 101 packets, how many acknowledgement packets does the destination generate?
- Assume no packets are lost.
- 101

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)?





PHY (Layer 1) Header

Ethernet (Layer 2) Header

IP (Layer 3) Header

TCP (Layer 4-5) Header

TCP Payload

Ethernet Trailer

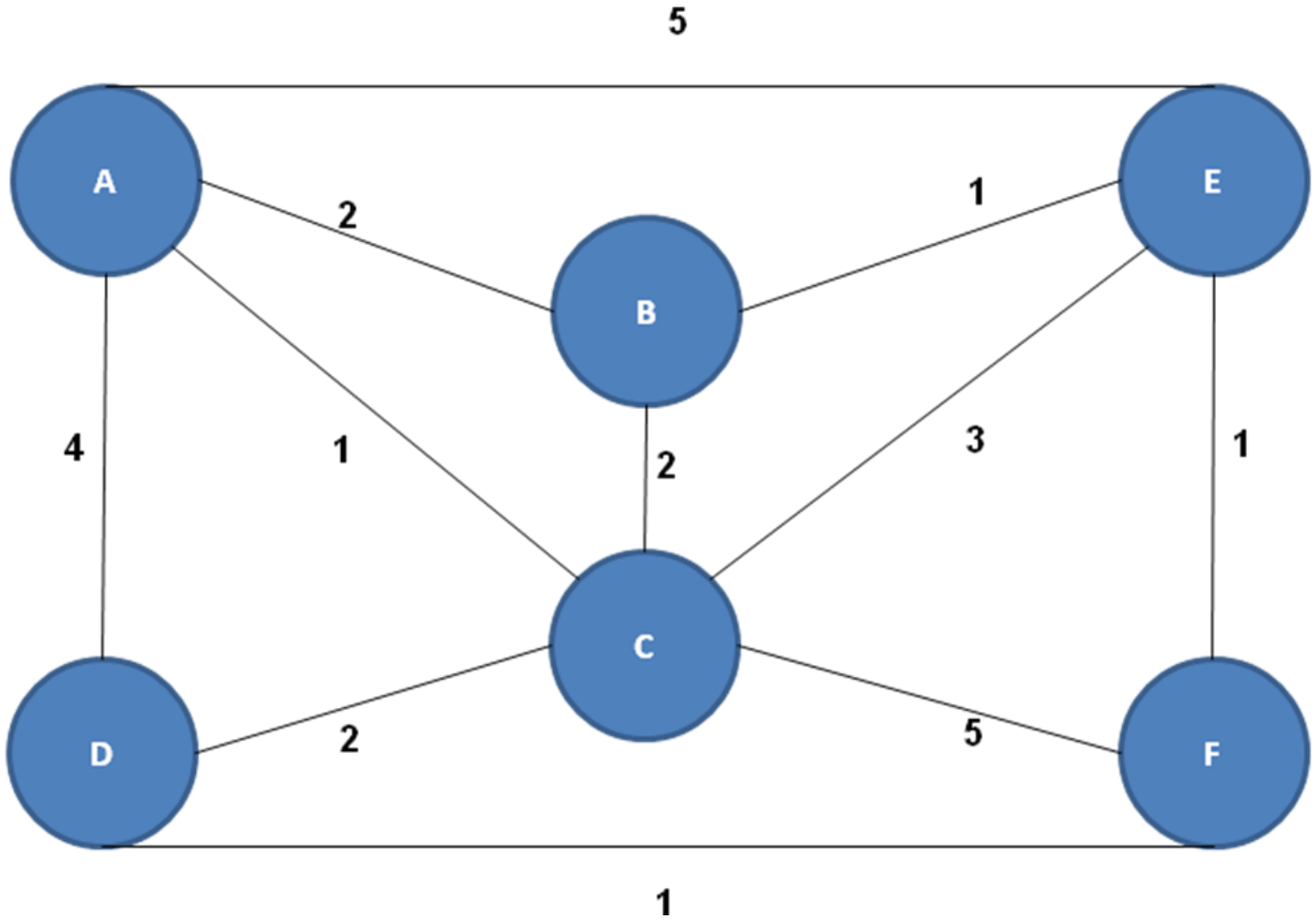
PHY Trailer

Network layer

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

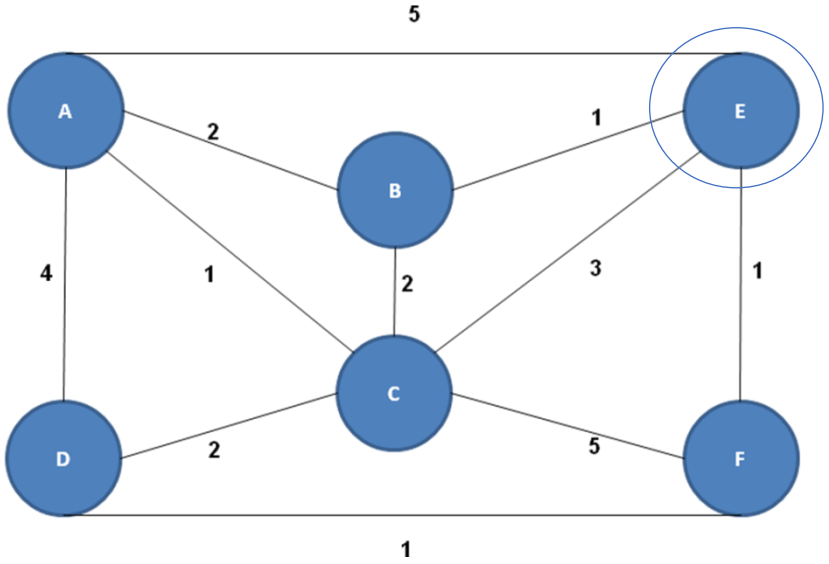
Intuition behind routing algorithms (link state, distance vector)

What's the latency to get to your neighbor?



Distance vector in action

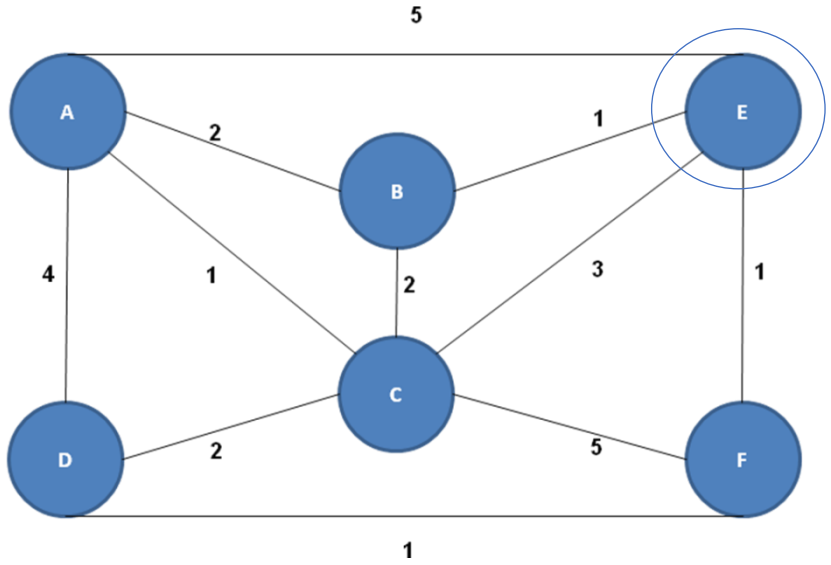
- Uses neighbor info, local algorithm (i.e. every node only sees its neighbors)
- Each node knows the cost to each of its neighbors
- Each node sends its idea of routing to each of its neighbors
- Known colloquially as "routing by rumor"
- For every other node, each node determines the lowest cost next-hop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

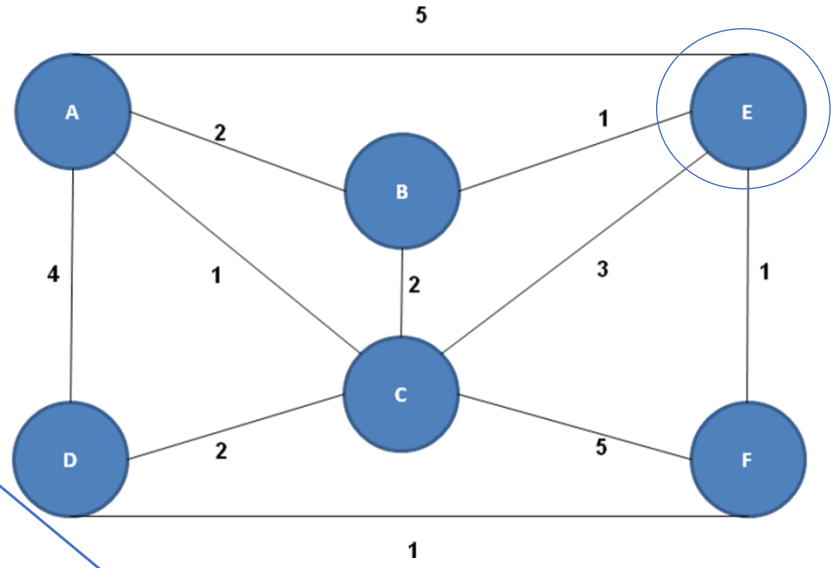


- Node E listens to messages from its neighbors:
  - A says its routes are 0/A, 5/E, 2/B, 1/C, 4/D, 5/F
  - B says its routes are 0/B, 1/E, 2/C, 2/A, 5/F, 4/D
  - C says its routes are 0/C, 1/A, 2/D, 5/F, 3/E, 2/B
  - F says its routes are 0/F, 5/C, 1/D, 1/E, 2/B, 4/A
  - D isn't heard because it isn't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

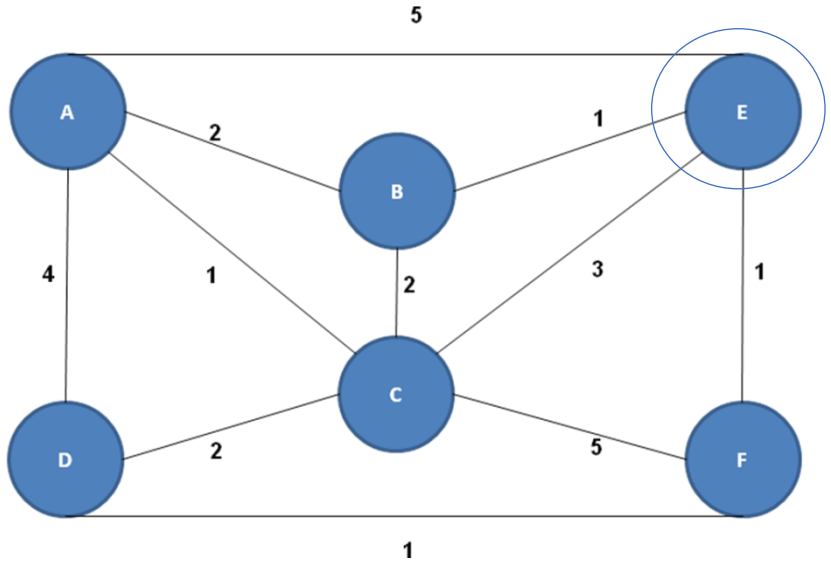


- Node E listens to messages from its neighbors:
  - A says its routes are 0/A, 5/E, 2/B, 1/C, 4/D, 5/F
  - B says its routes are 0/B, 1/E, 2/C, 2/A, 5/F, 4/D
  - C says its routes are 0/C, 1/A, 2/D, 5/F, 3/E, 2/B
  - F says its routes are 0/F, 5/C, 1/D, 1/E, 2/B, 4/A
  - D isn't heard because it isn't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

- 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

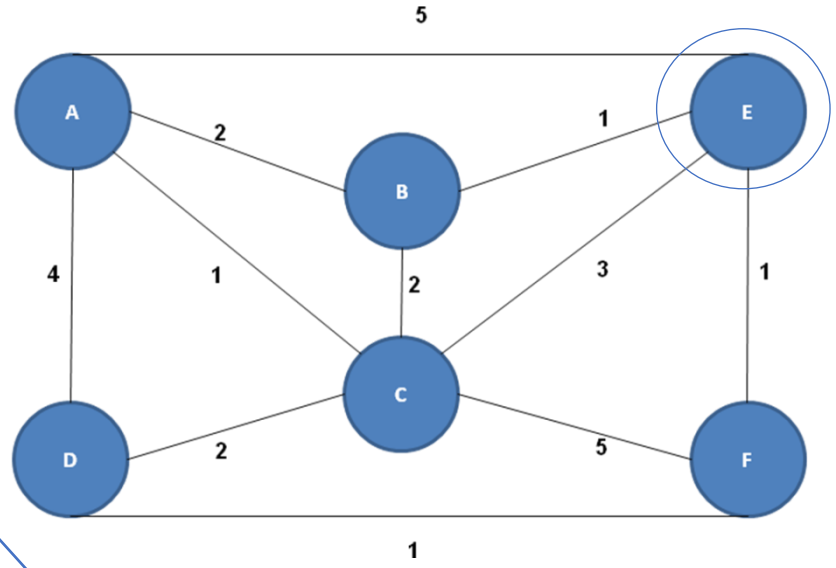


- Node E listens to messages from its neighbors:
  - A says its routes are 0/A, 5/E, 2/B, 1/C, 4/D, 5/F
  - B says its routes are 0/B, 1/E, 2/C, 2/A, 5/F, 4/D
  - C says its routes are 0/C, 1/A, 2/D, 5/F, 3/E, 2/B
  - F says its routes are 0/F, 5/C, 1/D, 1/E, 2/B, 4/A
  - D isn't heard because it isn't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

- 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

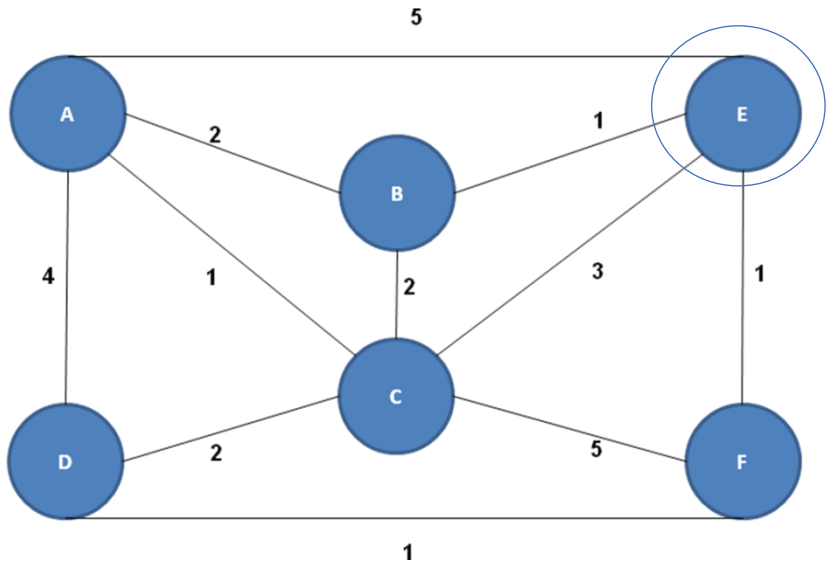


- Node E listens to messages from its neighbors:
  - A says its routes are 0/A, 5/E, 2/B, 1/C, 4/D, 5/F
  - B says its routes are 0/B, 1/E, 2/C, 2/A, 5/F, 4/D
  - C says its routes are 0/C, 1/A, 2/D, 5/F, 3/E, 2/B
  - F says its routes are 0/F, 5/C, 1/D, 1/E, 2/B, 4/A
  - D isn't heard because it isn't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

- 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

- 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)

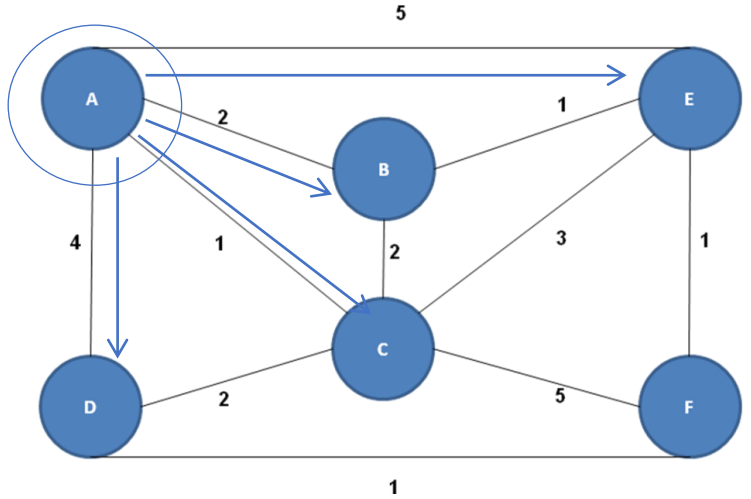
Distance Vector vs Link State Routing

- Distance Vector
  - 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
- One least-cost route to a destination
- Runs simultaneously on each node



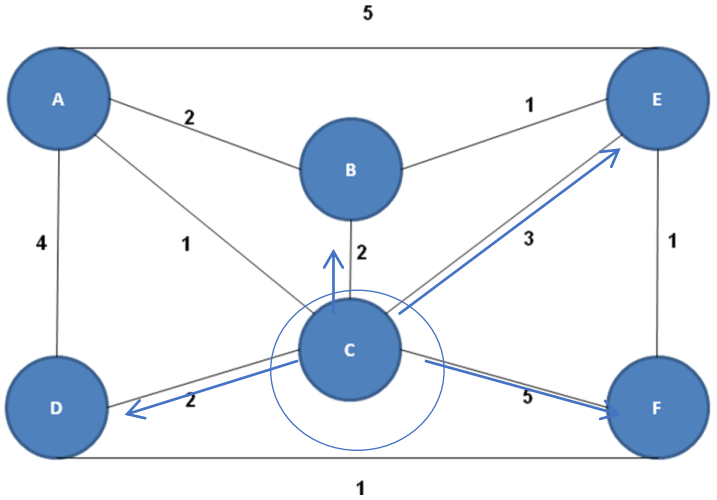
Calculating on Node A

(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



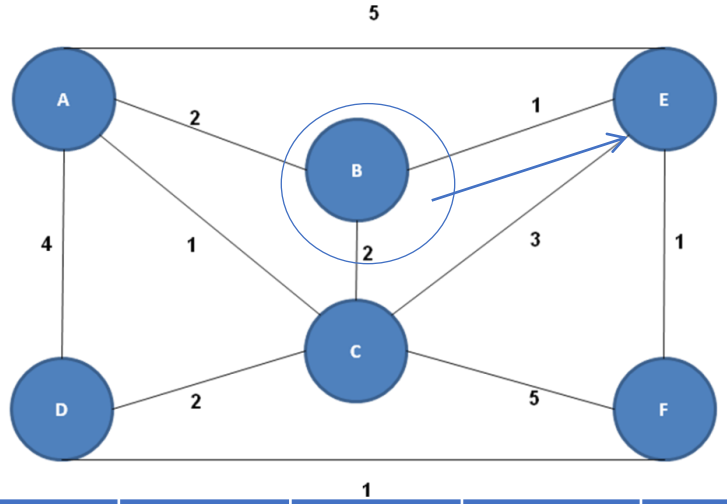
Calculating on Node A

(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



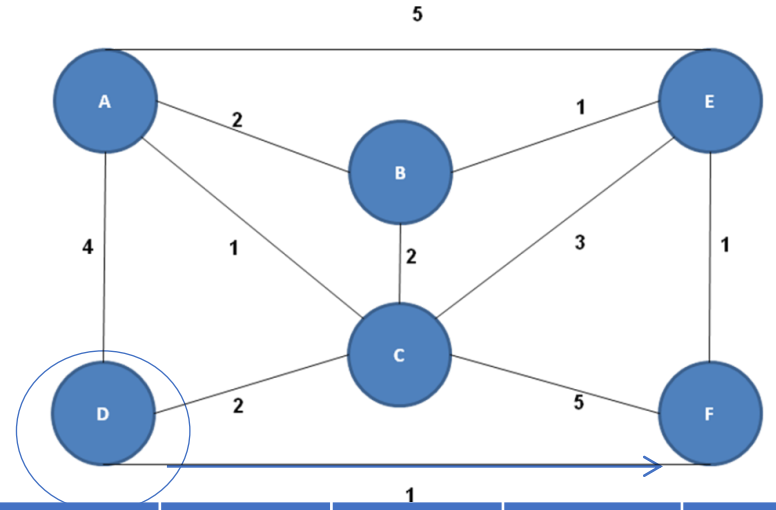
Calculating on Node A

(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	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



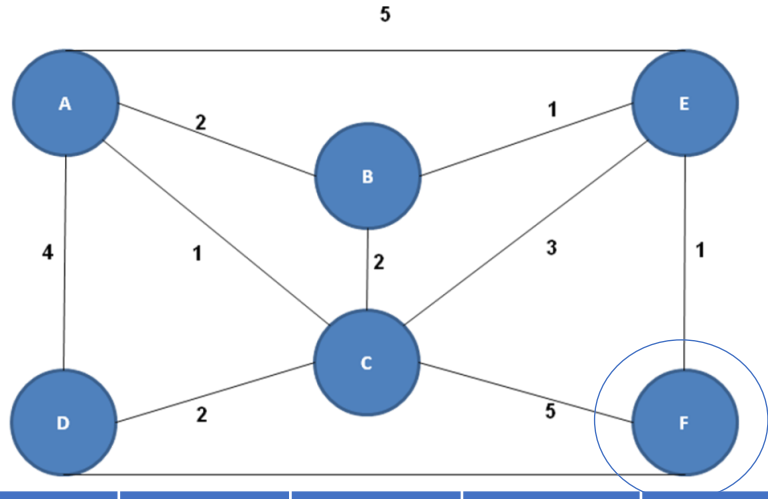
Calculating on Node A

(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)

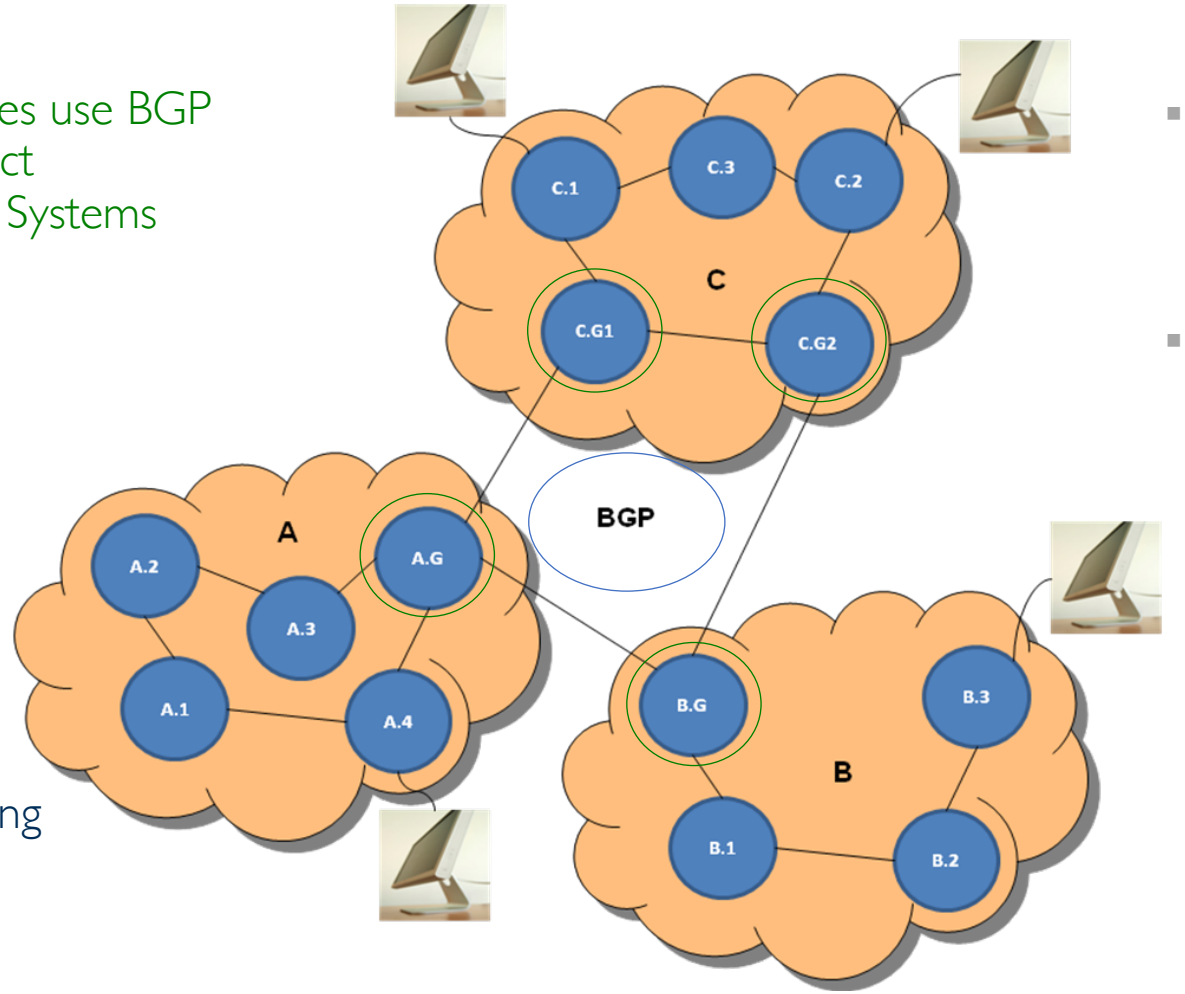
Iteration Count	Nodes to which least-cost routes known	B Cost/ route	C Cost/ route	D cost/ route	E Cost/ route	F Cost/ route
Init	A	2/AB	1/AC	4/AD	5/AE	∞
1	A,C	2/AB	1/AC	3/ACD	4/ACE	6/ACF
2	A,C,B	2/AB	-	3/ACD	3/ABE	6/ACF
3	A,C,B,D	-	-	3/ACD	3/ABE	4/ACDF
4	A,C,B,D,E	-	-	-	3/ABE	4/ACDF
5	A,C,B,D,E,F	-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

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

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

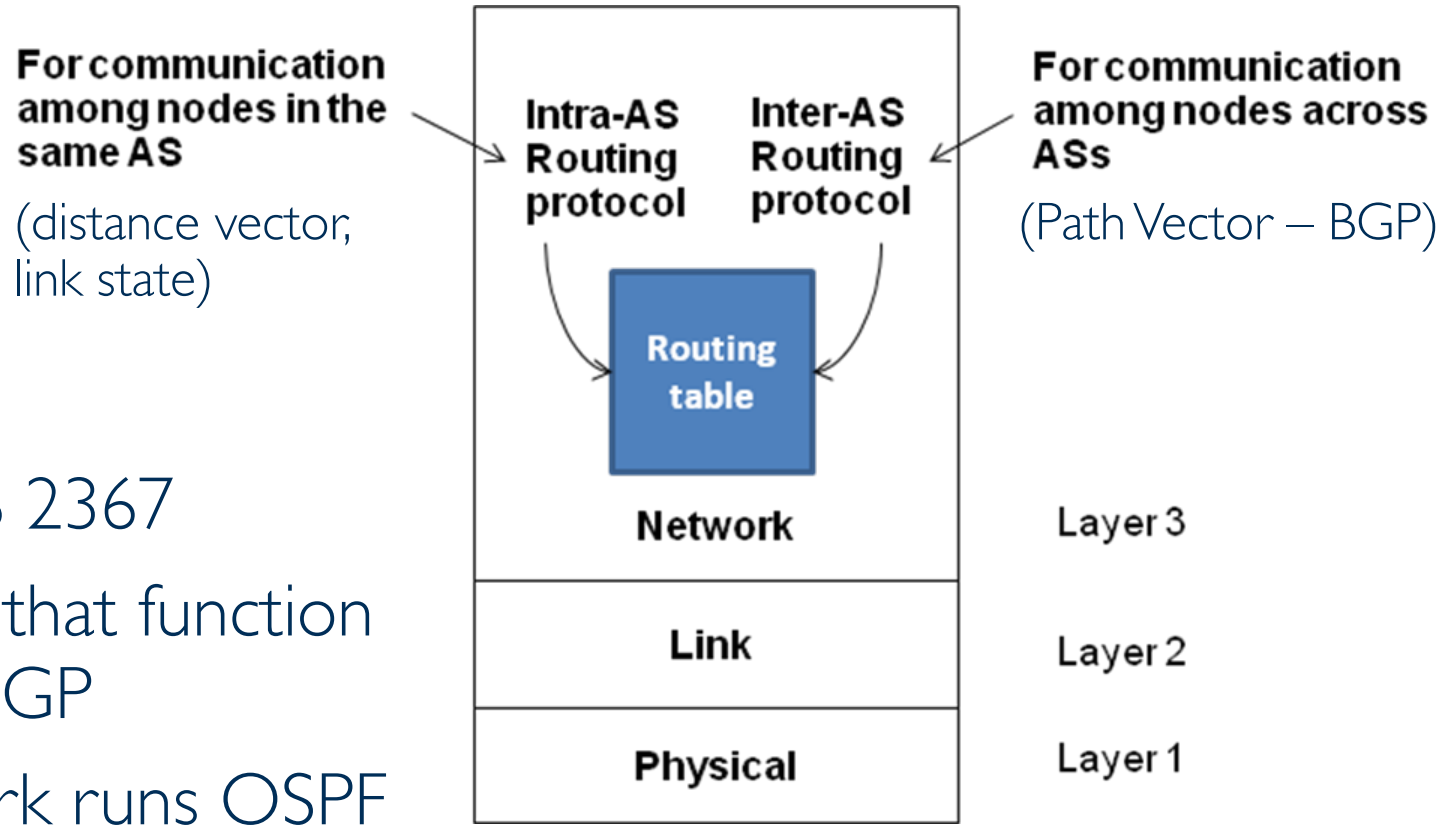


- ASs are assigned numbers by IANA/ARIN
- Nodes within an AS use Link State or Distance Vector (or other) routing protocol

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.

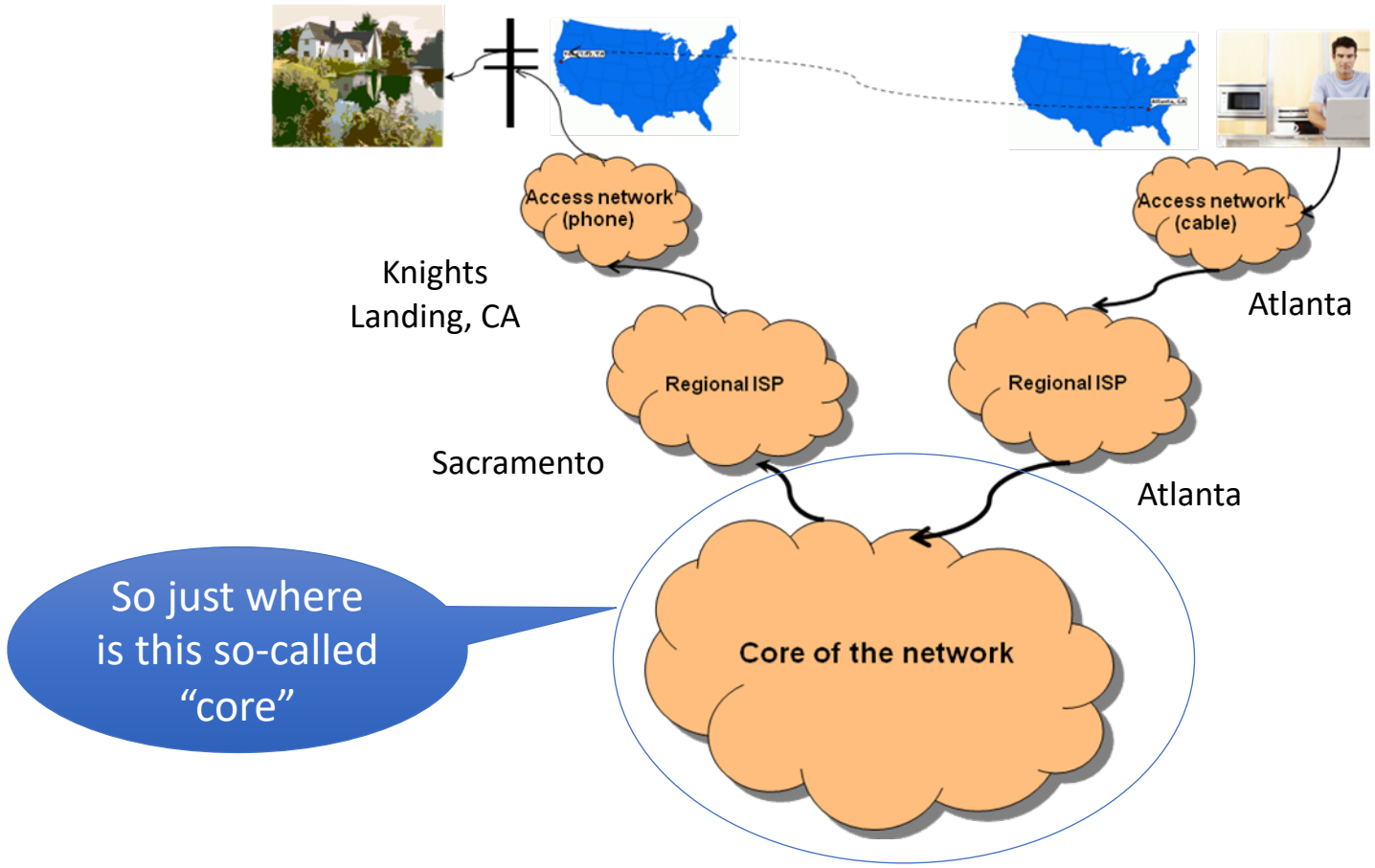
Routing at GT



- 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




This Jen, is “The Internet”.

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

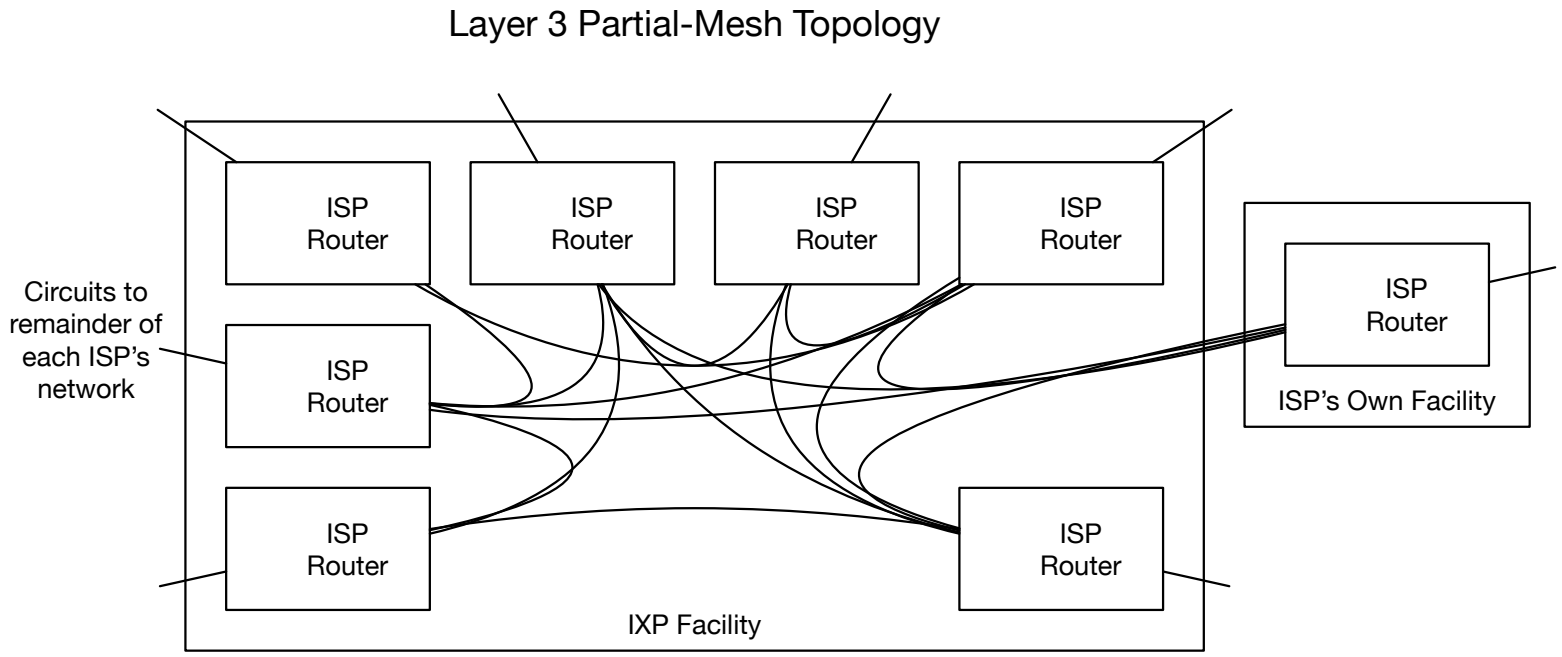
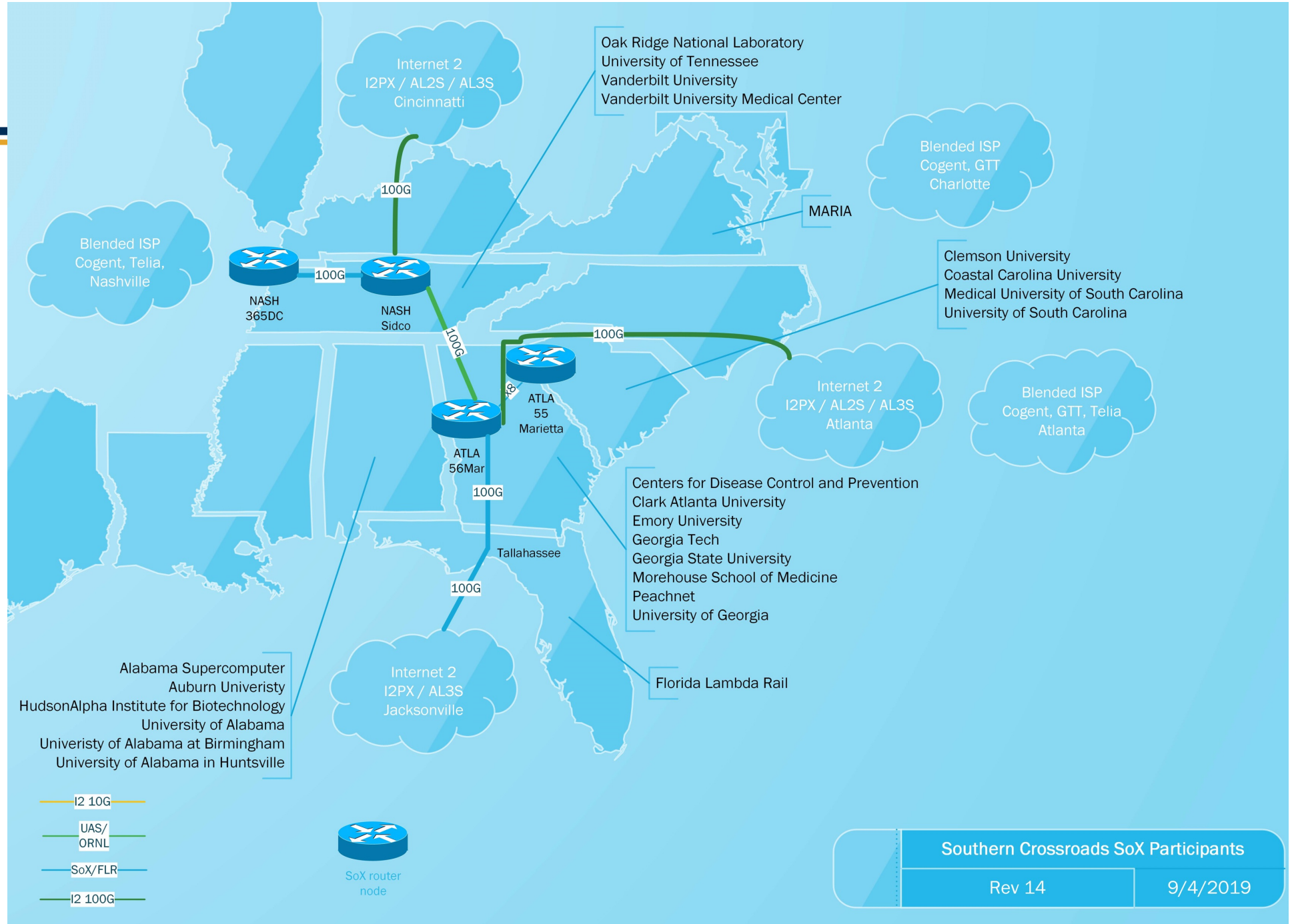


Diagram by Bill Woodcock, Packet Clearing House, rights released for use on Wikipedia

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
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

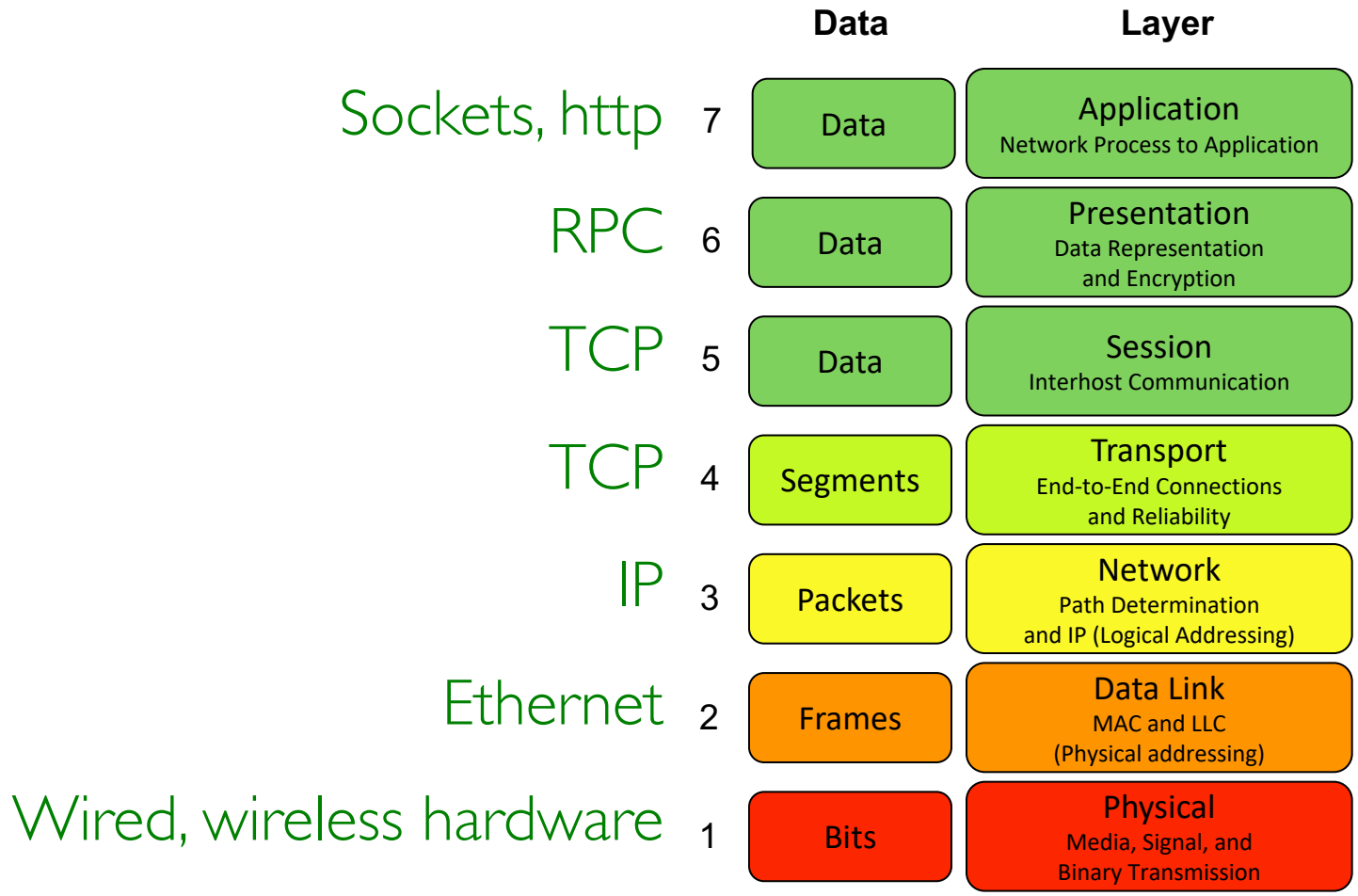
Network Layer Summary

Network Terminology	Definition/Use
Circuit switching	A network layer technology used in telephony. Reserves the network resources (link bandwidth in all the links from source to destination) for the duration of the call; no queuing or store-and-forward delays
Packet switching	A network layer technology used in wide area Internet. It supports best effort delivery of packets from source to destination without reserving any network resources en route.
Message switching	Similar to packet switching but at the granularity of the whole message (at the transport level) instead of packets.
Switch/Router	A device that supports the network layer functionality. It may simply be a computer with a number of network interfaces and adequate memory to serve as input and output buffers.
Input buffers	These are buffers associated with each input link to a switch for assembling incoming packets.
Output buffers	These are buffers associated with each outgoing link from a switch if in case the link is busy.
Routing table	This is table that gives the next hop to be used by this switch for an incoming packet based on the destination address. The initial contents of the table as well as periodic updates are a result of routing algorithms in use by the network layer.

Network Layer Summary

Network Terminology	Definition/Use
Delays	The delays experienced by packets in a packet-switched network
Store and forward	This delay is due to the waiting time for the packet to be fully formed in the input buffer before the switch can act on it.
Queuing	This delay accounts for the waiting time experienced by a packet on either the input or the output buffer before it is finally sent out on an outgoing link.
Packet loss	This is due to the switch having to drop a packet due to either the input or the output buffer being full and is indicative of traffic congestion on specific routes of the network.
Service Model	This is the contract between the network layer and the upper layers of the protocol stack. Both the datagram and virtual circuit models used in packet-switched networks provide **best effort delivery** of packets.
Virtual Circuit (VC)	This model sets up a virtual circuit between the source and destination so that individual packets may simply use this number instead of the destination address. This also helps to simplify the routing decision a switch has to make on an incoming packet. (ATM and X.25)
Datagram	This model does not need any call setup or tear down. Each packet is independent of the others and the switch provides a best effort service model to deliver it to the ultimate destination using information in its routing table. (IP)

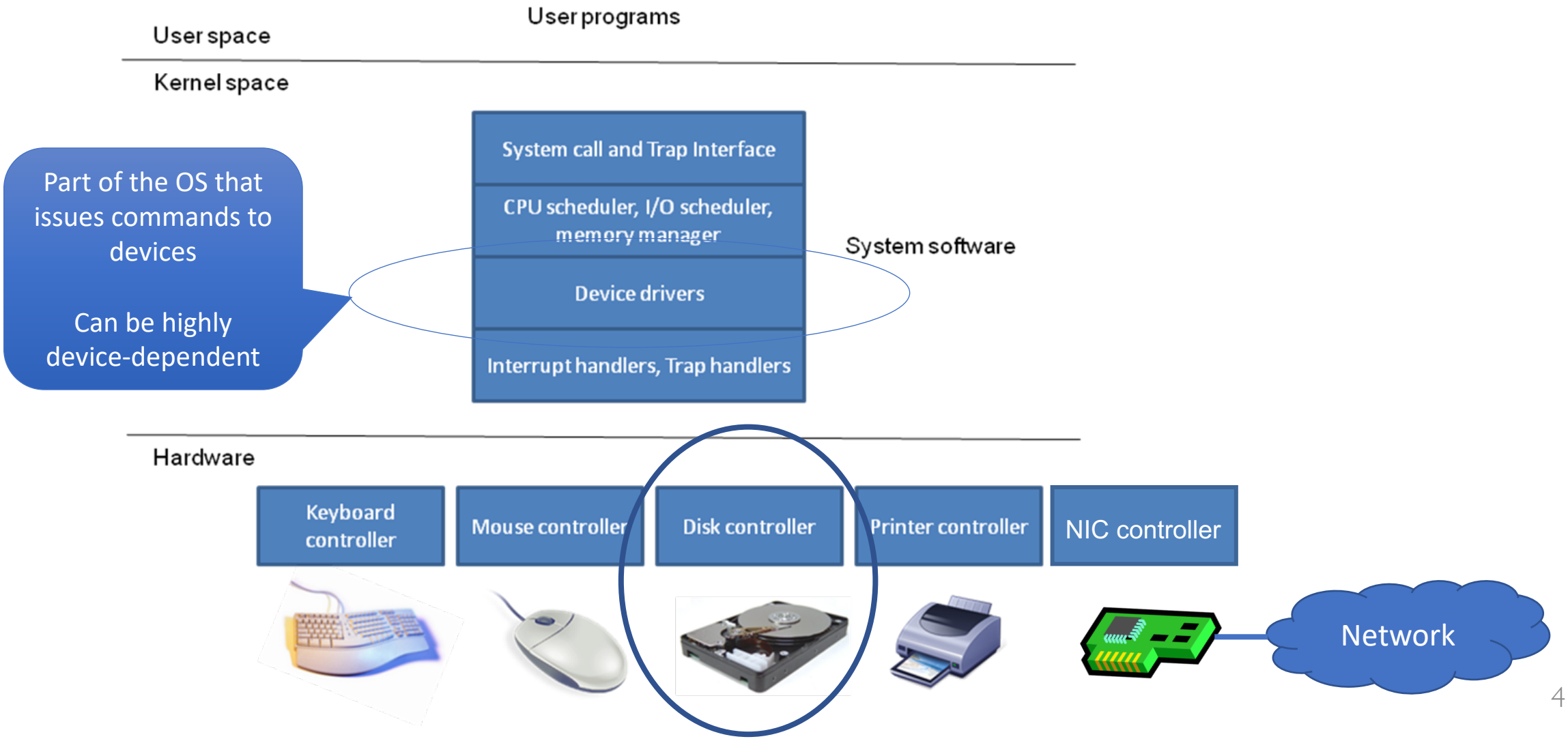
Summary


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

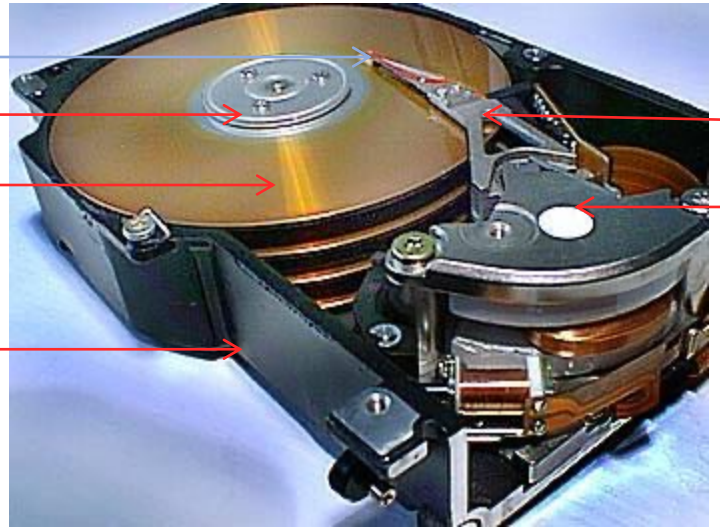
Agenda

- ~~Wrap up Networking~~
- Disk Scheduling

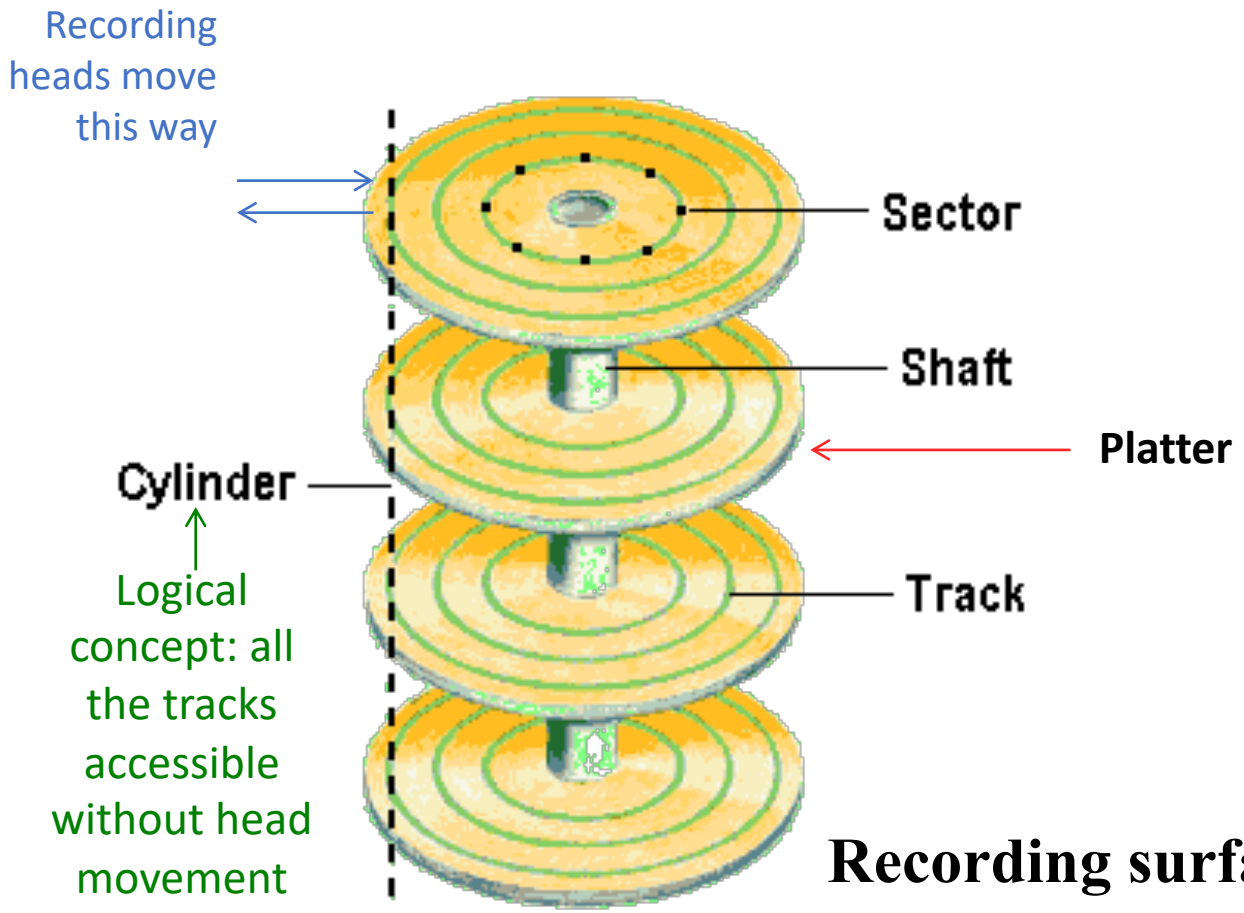
Device drivers and OS



A modern disk drive



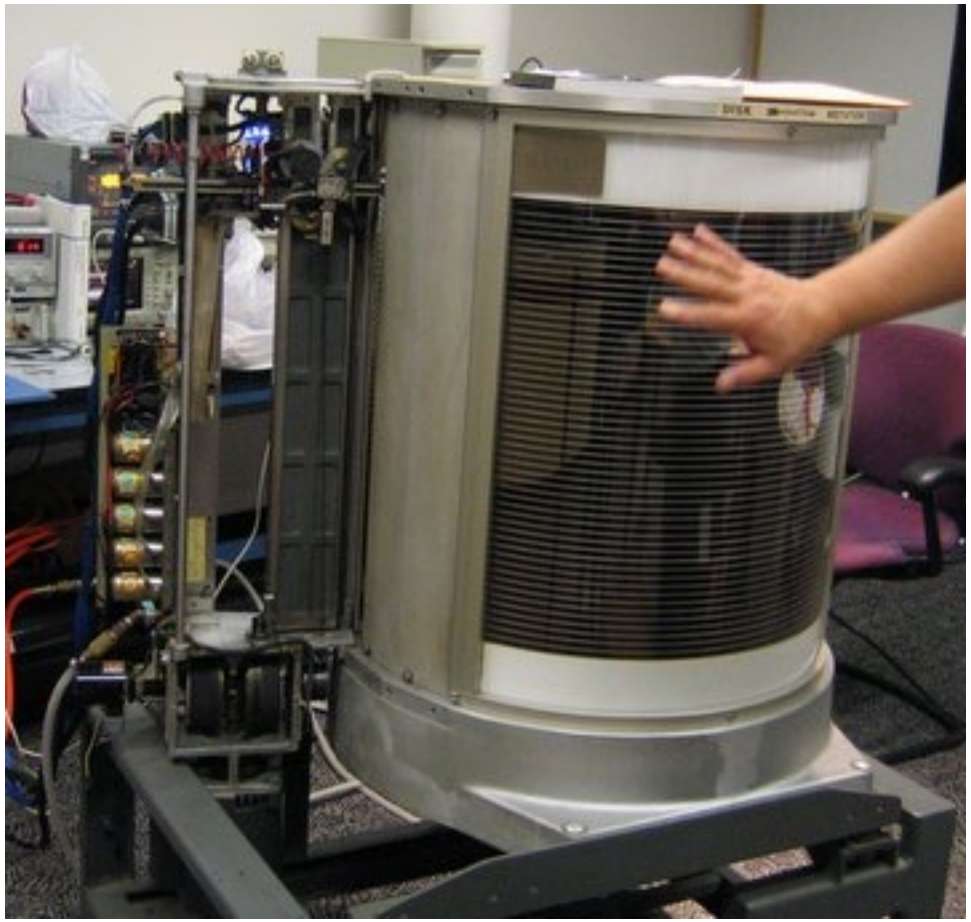
More disk drive terminology



How a modern disk drive works

- Cool 7-min video by Seagate engineer: https://www.youtube.com/watch?v=NtPc0jI21i0

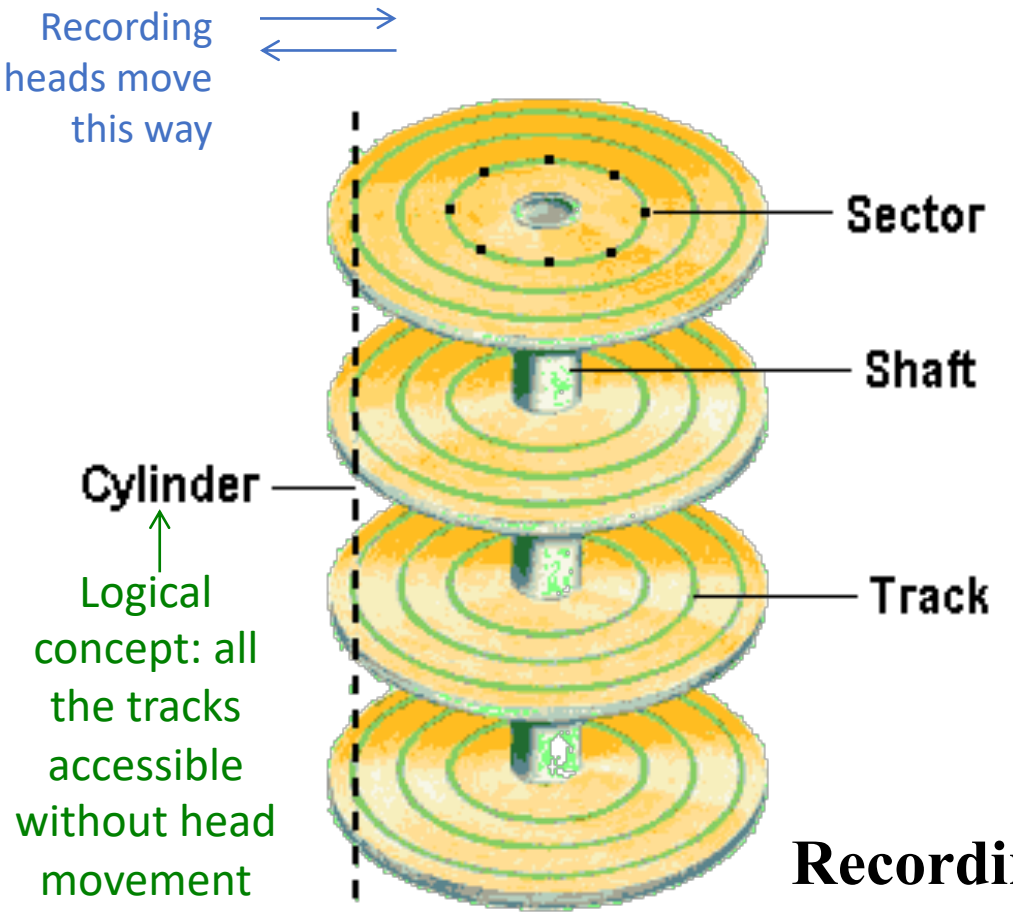
The first magnetic disk drive



- Announced: 1956
- 50 platters
- 3.75 MB
- 1200 rpm
- 8,800 char/sec
- Weight: 2000 lbs.

IBM 305 RAMAC at the Computer History Museum, Mountain View, CA
https://www.youtube.com/watch?v=knFRJP3s8WQ

More disk drive terminology



Typical today:

WD Ultrastar DC530
7200 RPM
Seek: 7.5 ms
Rot lat: 4 ms
Surface size – 1.75 TB
Tot cap – 14 TB
Xfer rate – 600 MB/sec

Recording surface on both sides

Based on those stats…

- How long does it take to find a random block on the disk?
- seek-time + rotational-latency / 2
- In this case, 7.5ms + 2ms = 9.5ms
- Roughly how many random reads can you do per second?
- 1000ms / 9.5 ms = ~100 reads(!)
- And the data transfer rate if blocks are 4K bytes?
- 100 * 4096 = 409 KB/sec
- Very different from the maximum transfer rate, right?

Typical today:
WD Ultrastar DC530
7200 RPM
Seek: 7.5 ms
Rot lat: 4 ms
Surface size – 1.75 TB
Tot cap – 14 TB
Xfer rate – 600 MB/sec

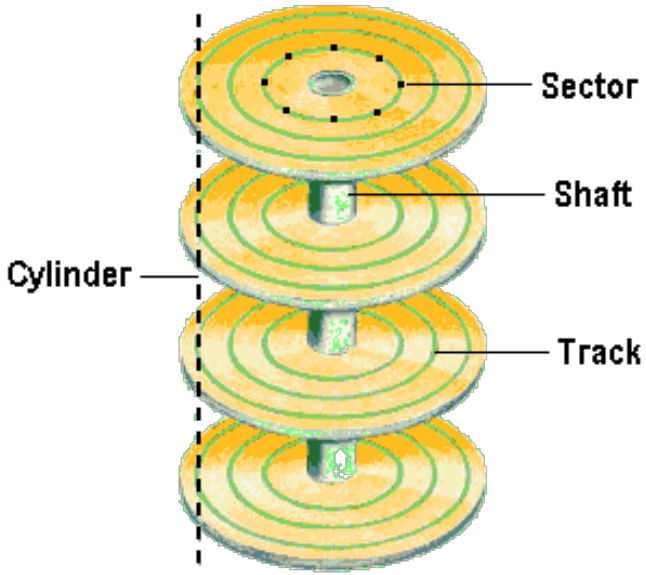
Capacity Metrics

Let,

p – number of platters,
n – number of surfaces per platter (1 or 2),
t – number of tracks per surface,
s – number of sectors per track,
b – number of bytes per sector,

The total capacity of the disk:

Capacity = (p * n * t * s* b) bytes



Example

Suppose a disk drive has 512 bytes per sector, 12 sectors per track, 100 tracks per surface, 2 surfaces per platter, and 6 platters.

What is the total capacity of such a drive in bytes?

How many cylinders does the device have?

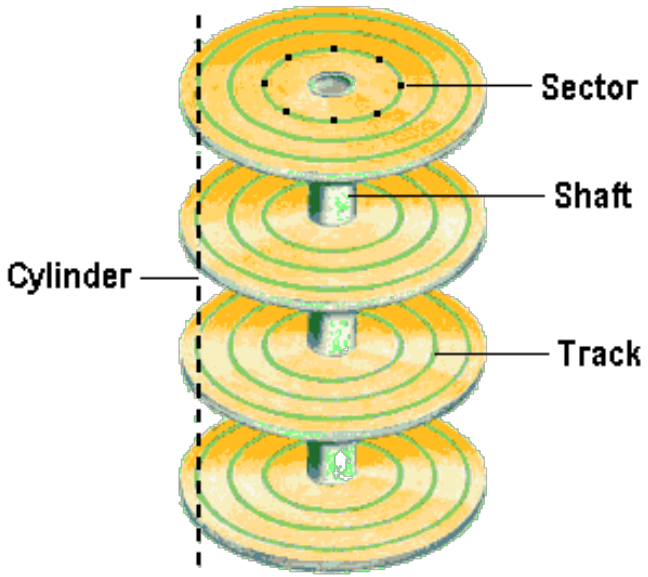
n b s t p

Capacity = ($p * n * t * s * b$) bytes

= 6 * 2 * 100 * 12 * 512 bytes

= 7200 KB (KB = 2^{10} bytes)

And it has t cylinders, or 100



Transfer metrics

Let,

r – rotational speed of the disk in RPM,
s – number of sectors per track,
b – number of bytes per sector,

Time for one revolution = $60/r$ seconds

Amount of data read in one revolution = $s * b$ bytes

The data transfer rate of the disk:

(Amount of data in track) / (time for one revolution)

$= (s * b)/(60/r)$

Data transfer rate

$= (s * b * r) / 60$ bytes/second

Example

Given the following specifications for a disk drive:

- b 512 bytes per sector
- s 400 sectors per track

  6000 tracks per surface

  3 platters
- r Rotational speed 15000 RPM

What is the transfer rate of the disk?

Data transfer rate = (s * b * r) / 60 bytes/second

= (400 * 512 * 15000) /60

= 51,200,000 bytes/second

Which of these attributes

of a hard disk drive is conceptual rather than physical?

- 14% A. I just want the participation credit
- 14% B. p – number of platters
- 14% C. n – number of surfaces per platter (1 or 2)
- 14% D. t – number of tracks per surface
- 14% E. s – number of sectors per track
- 14% F. b – number of bytes per sector
- 14% G. c – number of cylinders

Early 60's disk pack: 5MB

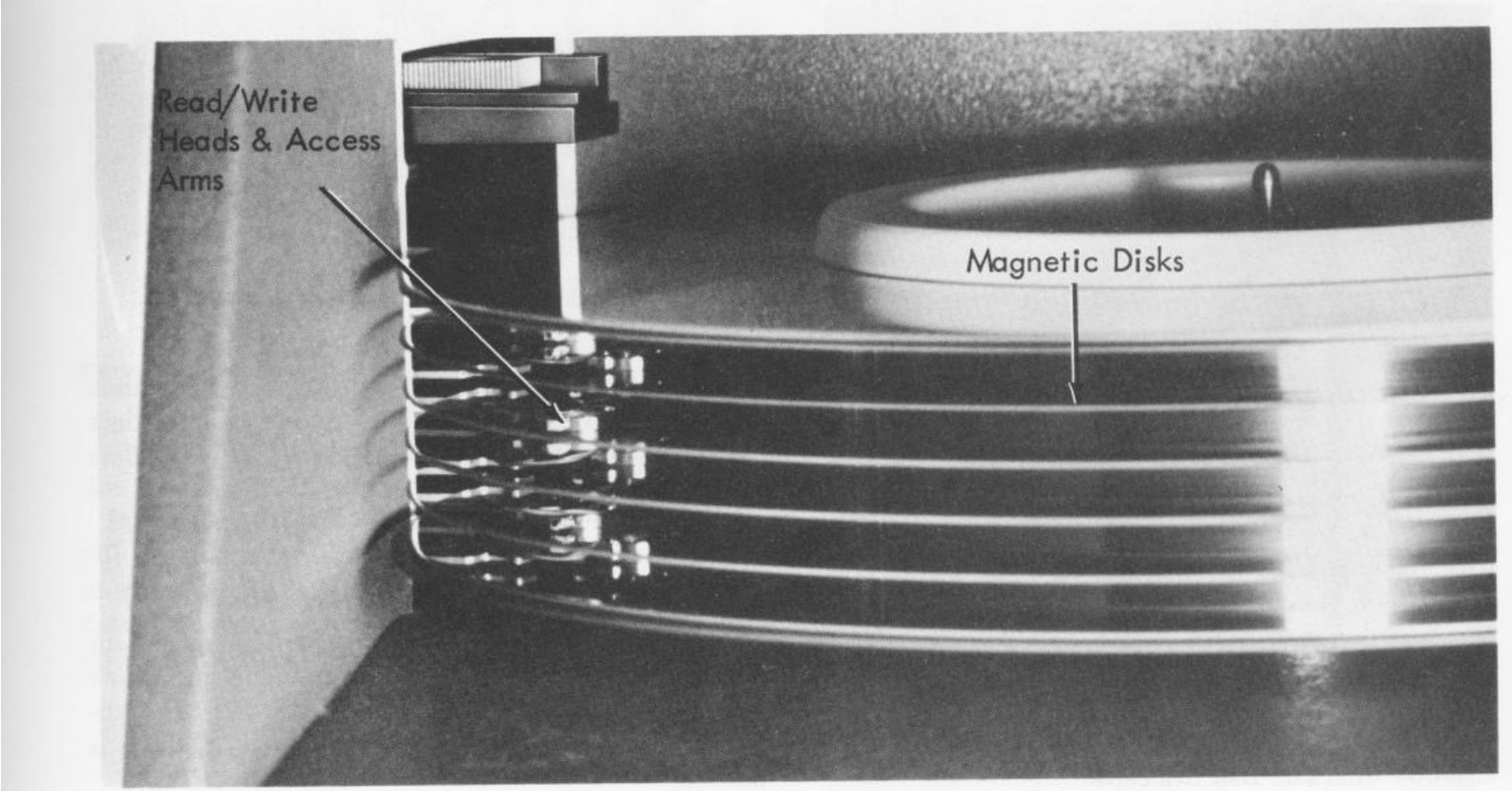
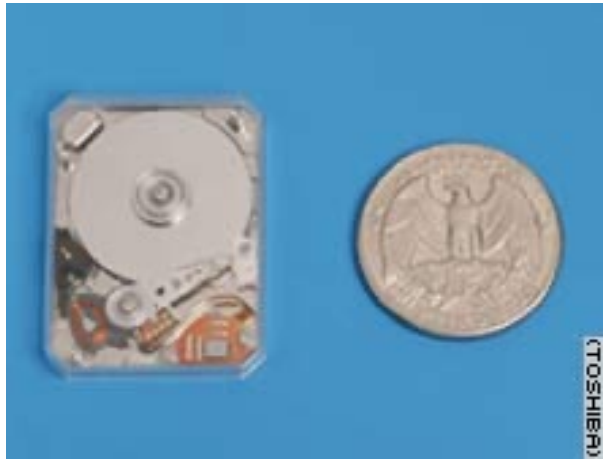


Figure 54. IBM 2311 Disk Storage Drive showing mounted disk pack and access arms

When computers were big…

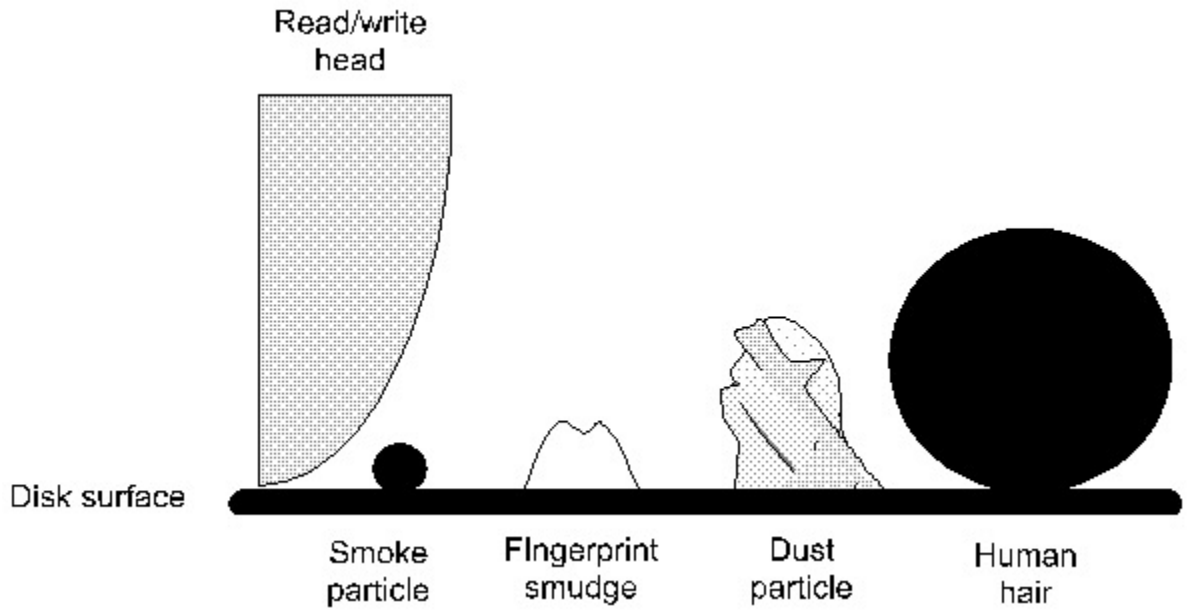
Welcome to modern times



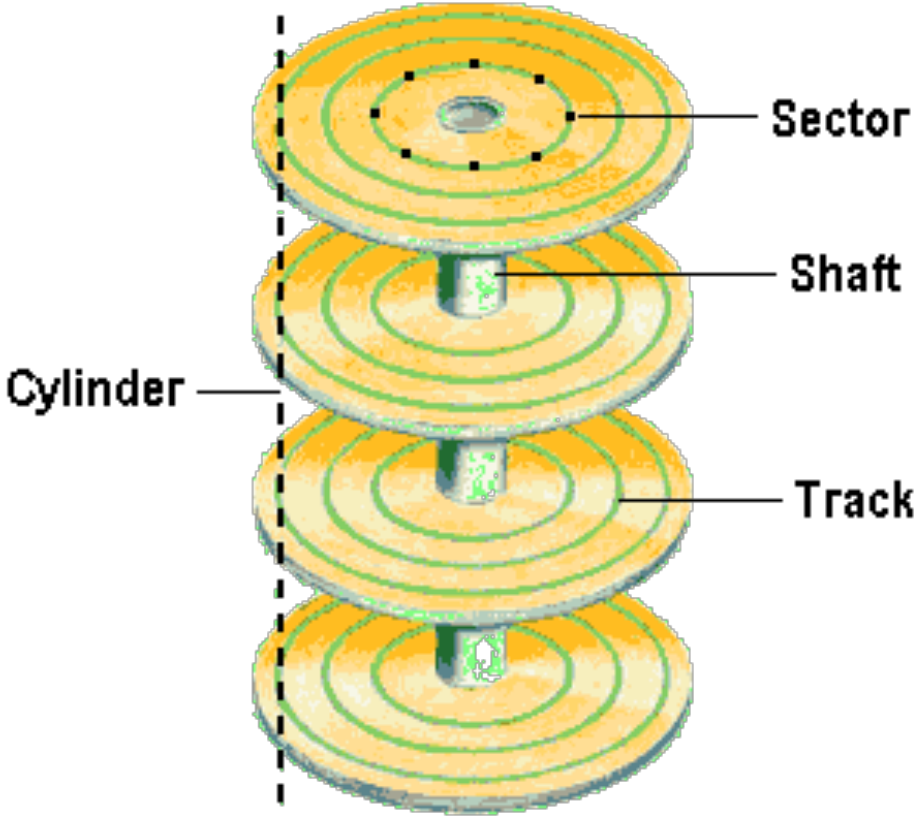
4 GB drive

Why are disk drives sealed?

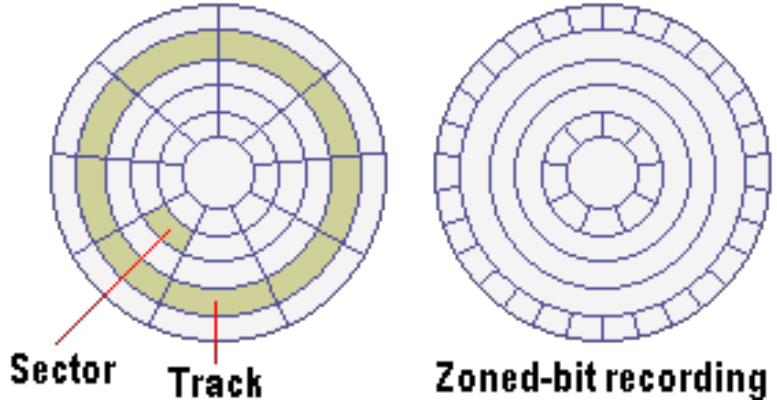
Since the early 60s, read/write heads are aerodynamically shaped so they fly on a very thin cushion of air. Obstacles are not well tolerated.



Zoned recording

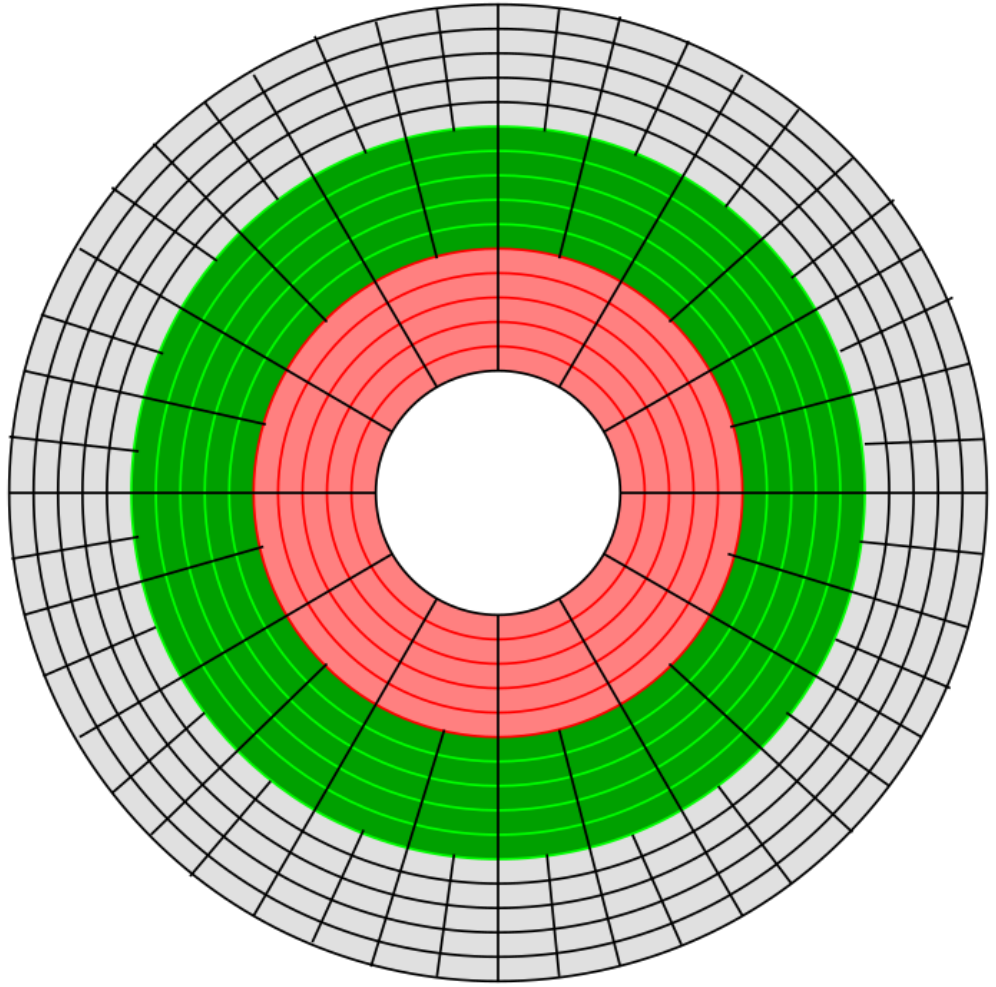


- The outside track of a platter has a larger surface and moves faster than an inside track!
- This allows the outside tracks to hold more data and transfer faster
- Zoned recording (of which there are many schemes) assigns more sectors to the outer tracks



Example of Zoned Recording

- Physical layout of sectors in a zone-bit disc
- As distance from the center increases, the number of sectors in a given angle increases from one (red) to two (green) to four (grey)
- The red zone has 12 sectors/track; green has 24; grey has 48.



https://en.wikipedia.org/wiki/Zone_bit_recording#/media/File:DiskStructure.svg

With zoned recording,

z **– number of zones,**

t_{zi} **– number of tracks at zone** z_i**,**

s_{zi} **– number of sectors per track at zone** z_i**,**

The total capacity of the disk with zoned recording:

$$\text{Capacity} = (p * n * (\sum (t_{zi} * s_{zi}), \text{ for } 1 \leq i \leq z) * b) \text{ bytes}$$

Total access latency

Let,

a – average seek time in seconds

r – rotational speed of the disk in RPM

s – number of sectors per track

Rotational latency = $60/r$ seconds

Average rotational latency = $(60 / (r * 2))$ seconds

Sector read time = rotational latency / number of sectors per track

Sector read time = $(60 / (r * s))$ seconds

Time to get to the desired track

= Average seek time

= a seconds

Time to get the head over the desired sector

= Average rotational latency

= (60 / (r * 2)) seconds

Time to read a sector

= Sector read time

= (60 / (r * s)) seconds

Time to read a random sector on the disk

= Time to get to the desired track +
Time to get the head over the desired track +
Time to read a sector

= a + (60 / (r * 2)) + (60 / (r * s))
seconds

Example

Given the following specifications for a disk drive:

256 bytes per sector
12 sectors per track
20 tracks per surface
3 platters
Average seek time of 20 ms
Rotational speed 3600 RPM

What would be the time to read 6 contiguous sectors from the same track?

What would be the time to read 6 sectors at random?

Disk scheduling

- Ordering of disk requests is quite important for I/O performance just as job scheduling is important for CPU performance
- Seeks and rotational delay (on the order of 5ms each) are mechanical and hence huge compared to the speeds of memory (100 ns) and CPU instructions (1 ns). That's 5×10^{-3} compared with 1×10^{-9}, or more than 6 orders of magnitude·
- However, as disk controllers have become more capable, I/O requests are queued for them by the OS as soon as they come in.
- Disk controllers may have tens of pending I/O operations queued and perform their own scheduling on them.

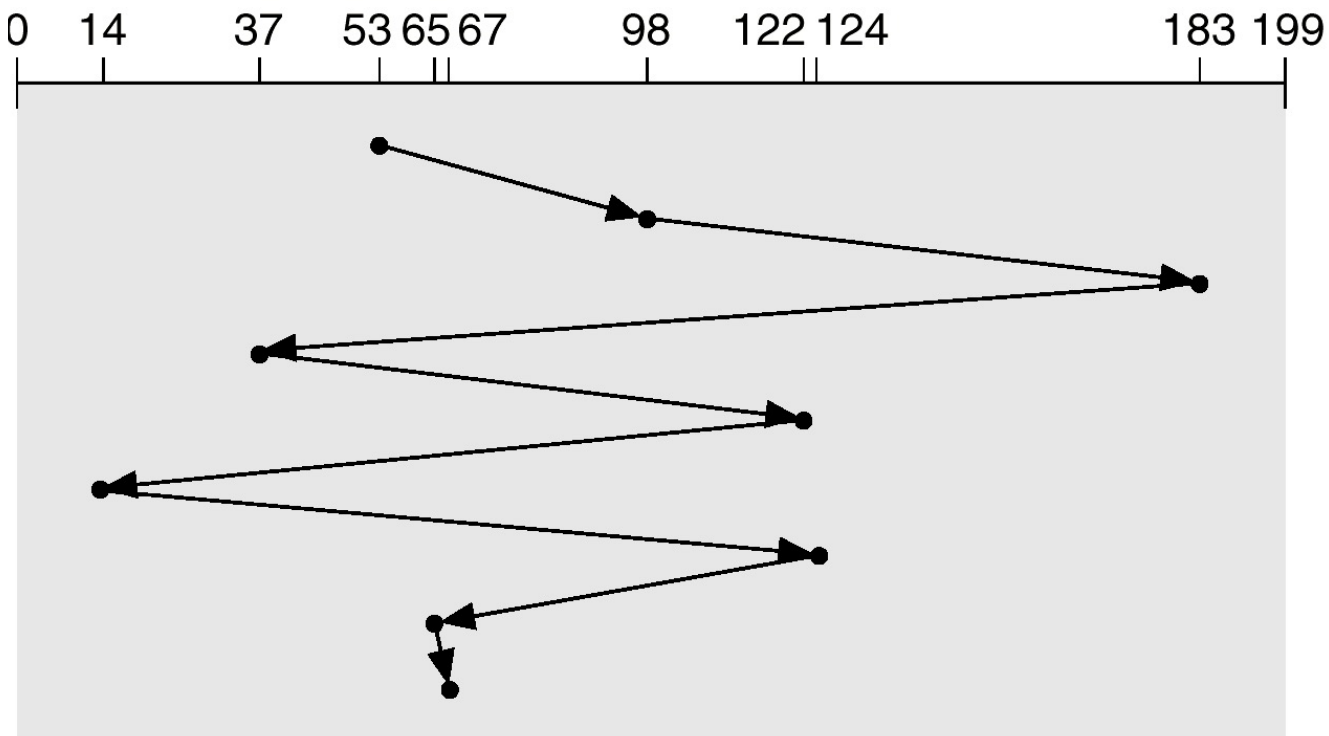
Disk scheduling

- We're going to bypass discussing the metrics for disk scheduling, but know there are many algorithms similar to CPU scheduling
- Some prominent algorithms
  - FCFS – self explanatory
  - SSTF – shortest seek time first
  - LOOK – keep going the same direction until no more requests in that direction (more like a real elevator)
  - SCAN – "elevator algorithm": like LOOK but always go to the top and bottom floors on each trip
  - C-SCAN – always go "up" and don't service any requests on the down cycle
  - C-LOOK – like LOOK but only service requests in one direction

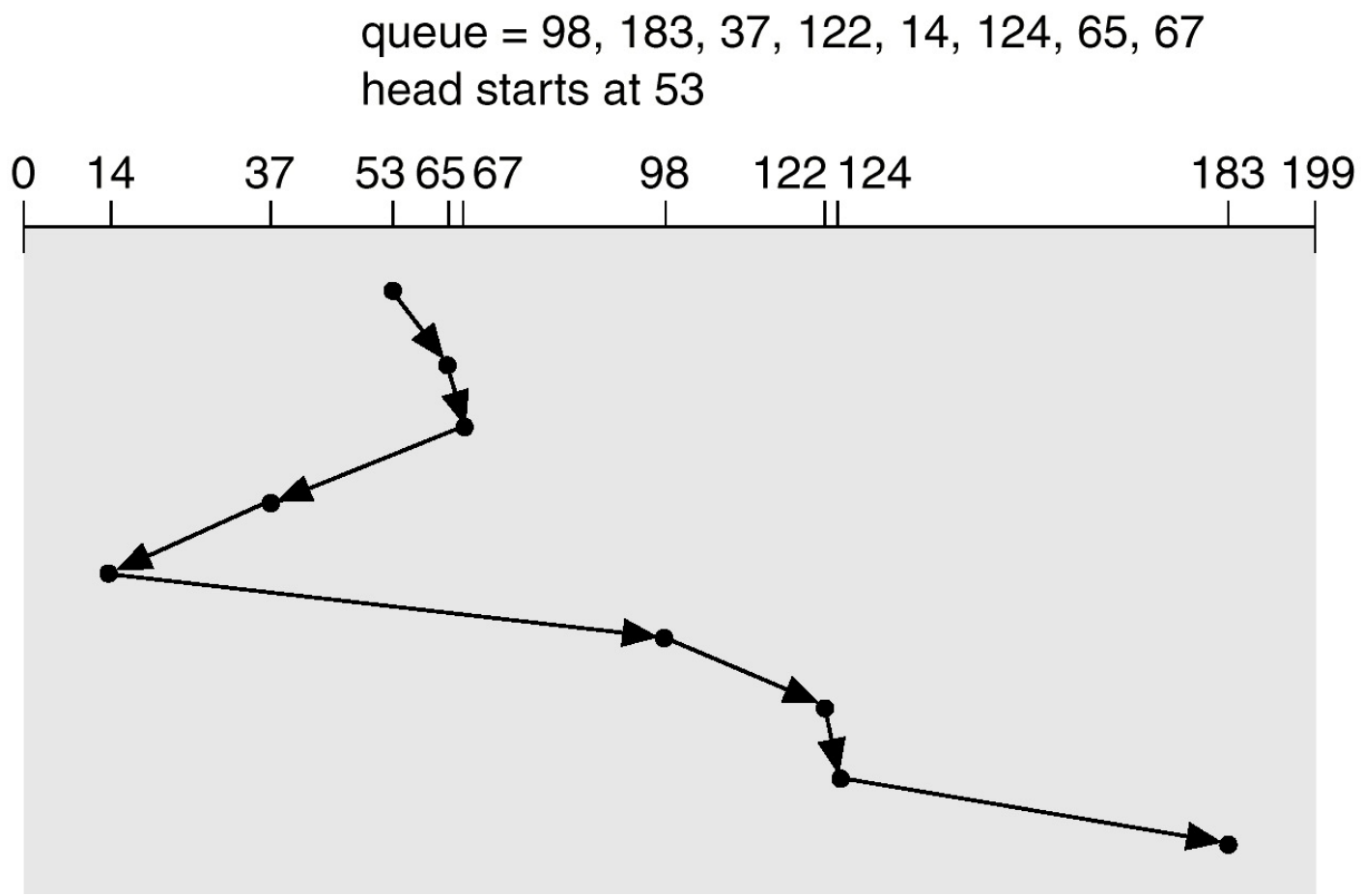
FCFS

queue = 98, 183, 37, 122, 14, 124, 65, 67
head starts at 53

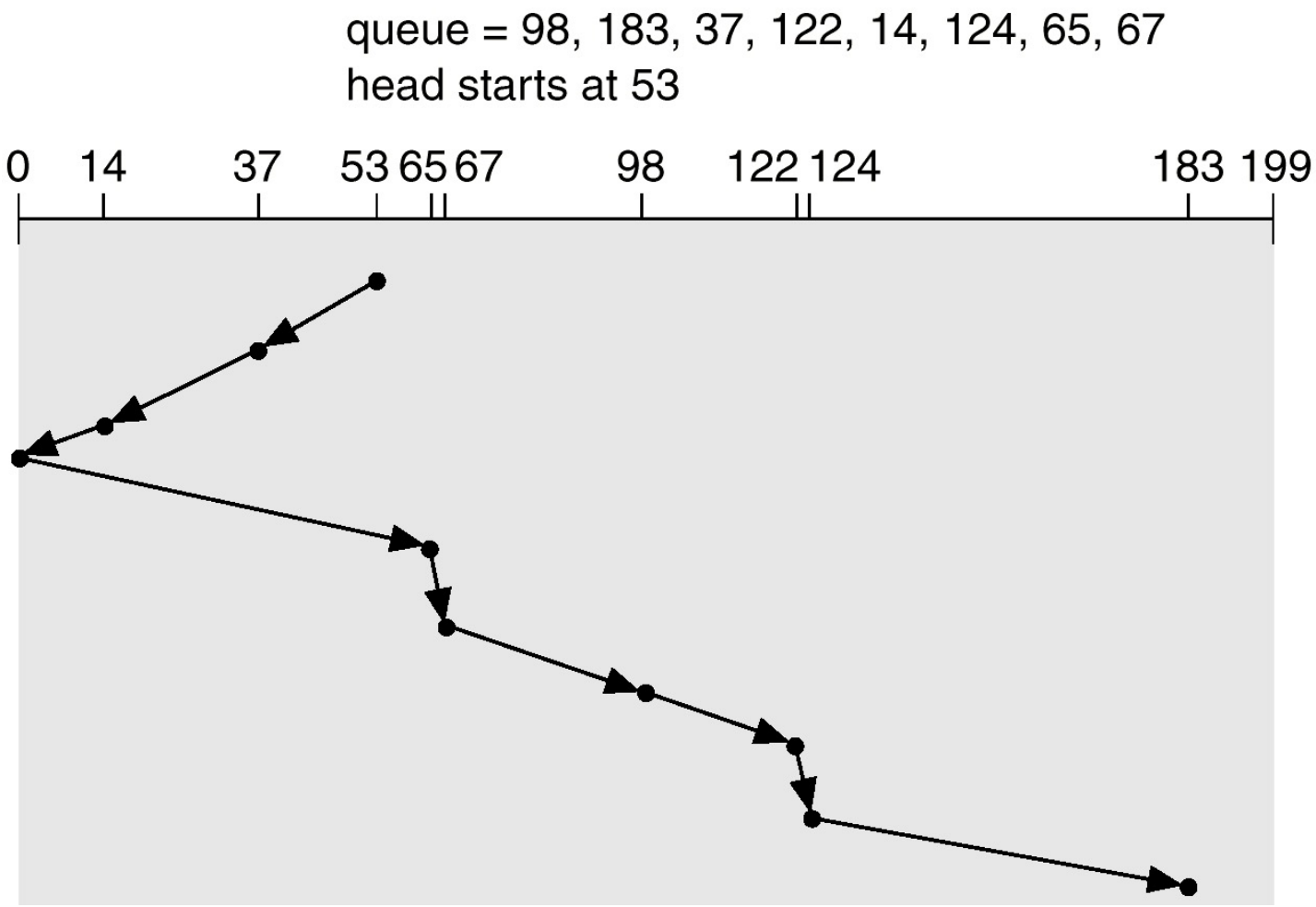
Tracks:



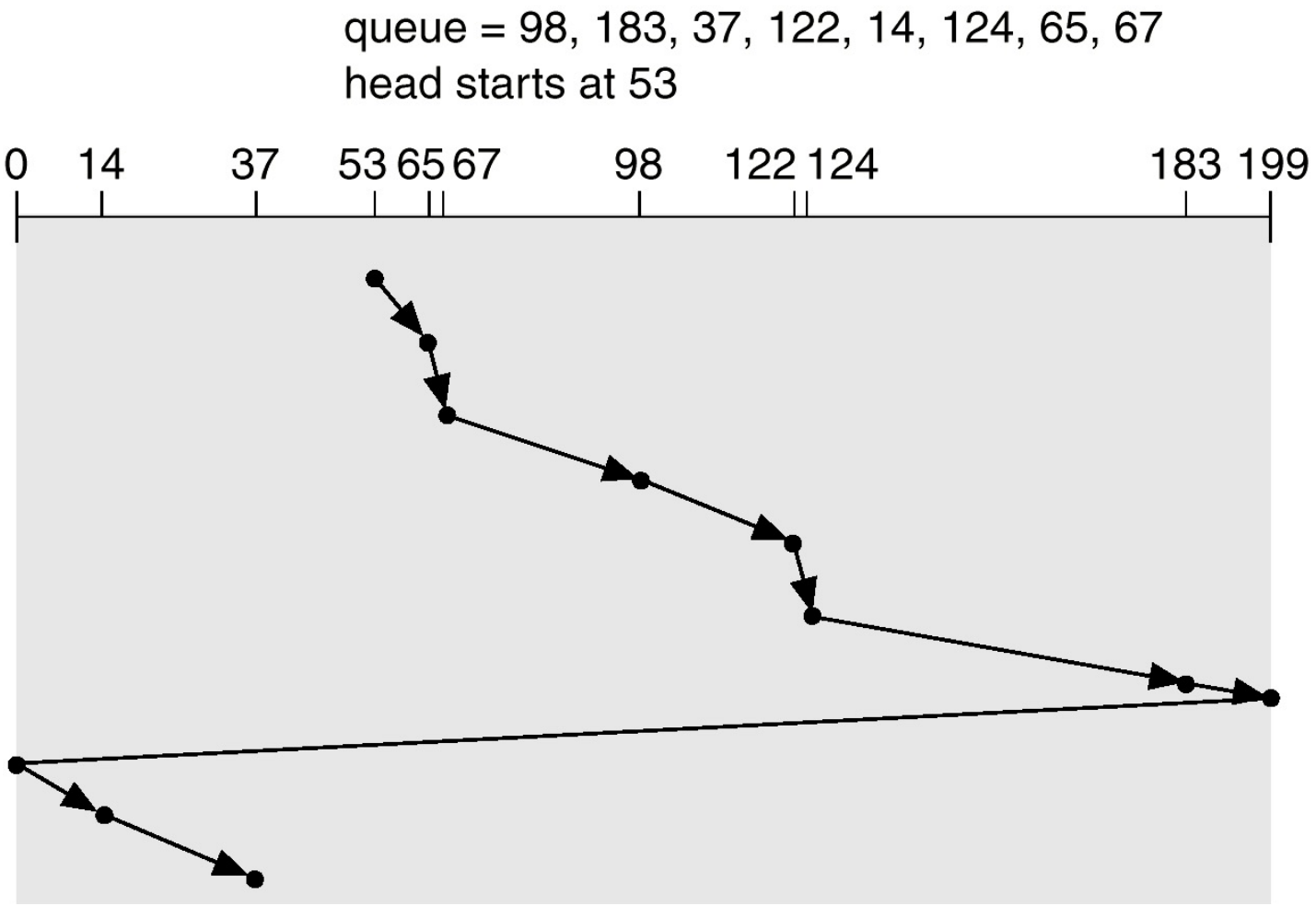
SSTF


queue = 98, 183, 37, 122, 14, 124, 65, 67
head starts at 53
0
14
37
53
65
67
98
122
124
183
199

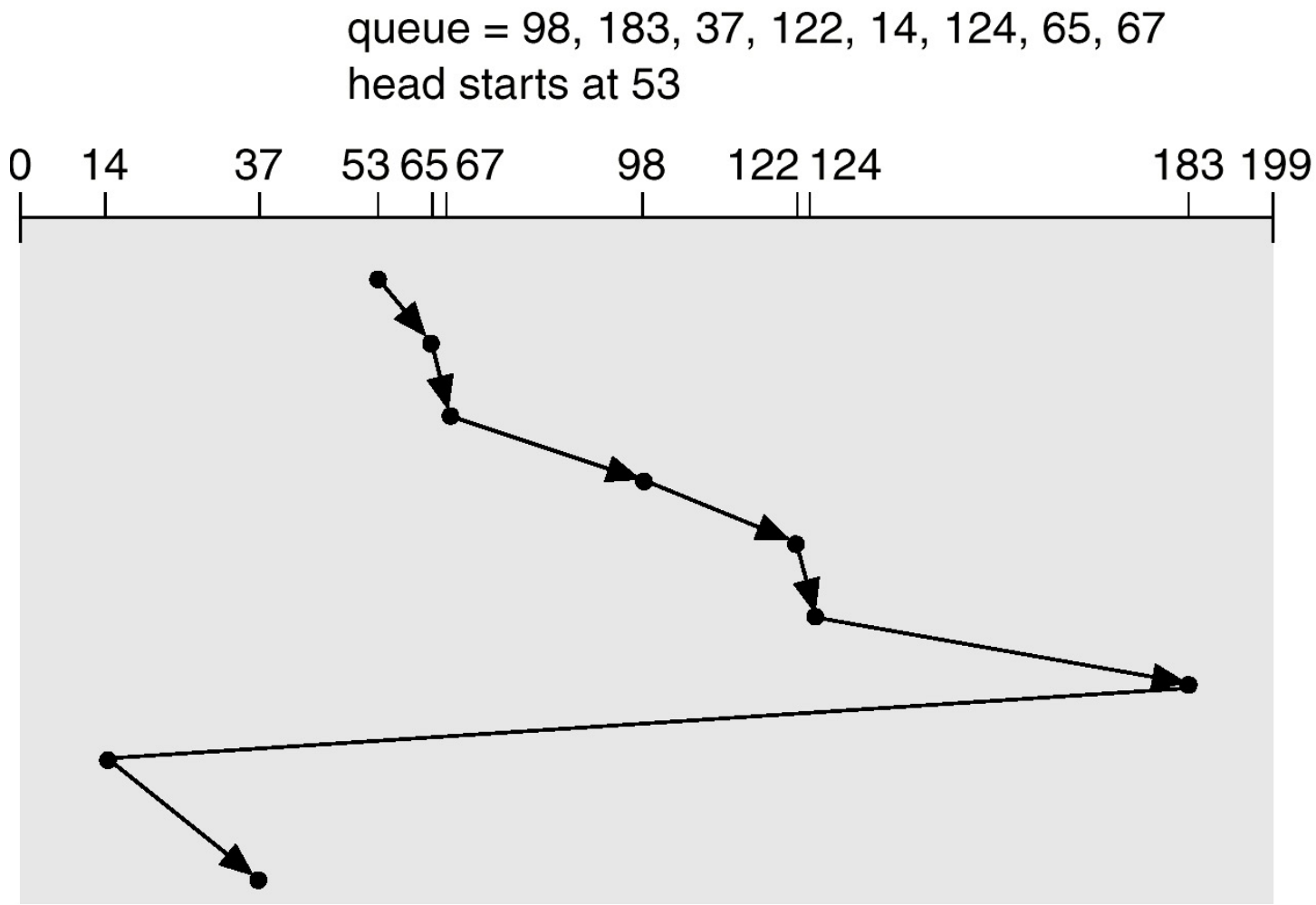
SCAN


queue = 98, 183, 37, 122, 14, 124, 65, 67
head starts at 53
0
14
37
53
65
67
98
122
124
183
199

C-SCAN


queue = 98, 183, 37, 122, 14, 124, 65, 67
head starts at 53
0
14
37
53
65
67
98
122
124
183
199

C-LOOK


queue = 98, 183, 37, 122, 14, 124, 65, 67
head starts at 53
0
14
37
53
65
67
98
122
124
183
199

Session ID: dag2200

Question

Which disk scheduling algorithm isn't called the "elevator algorithm" but works more like a modern elevator?

A. I just want the participation credit
B. SCAN
C. LOOK
D. C-SCAN
E. C-LOOK

Solid-state storage devices (SSDs)

- These have rapidly become effective replacements for small & fast storage applications
- They can act like HDDs and are faster but more expensive
- Look for an NVME connected SSD. Direct connection to PCI-E instead of through a SATA/SAS HDD controller.

	HDD	SSD
Capacity	1 - 18TB	0.25 - 15TB
IOPS (I/O per second)	400	50,000-1,000,000
Max transfer rate	Up to 500 MB/sec	Up to 7 GB/sec
Seek time	8 ms	0.1 ms
Energy use	6-16 W	2-5 W
Cost ($/GB)	0.02	0.10