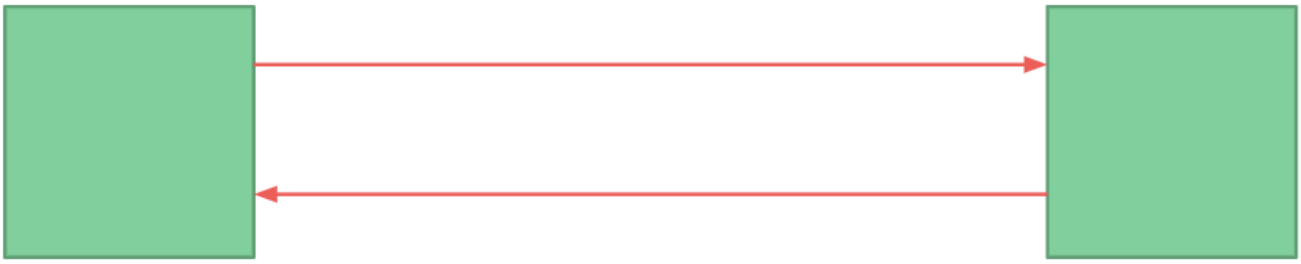
Quick Recap

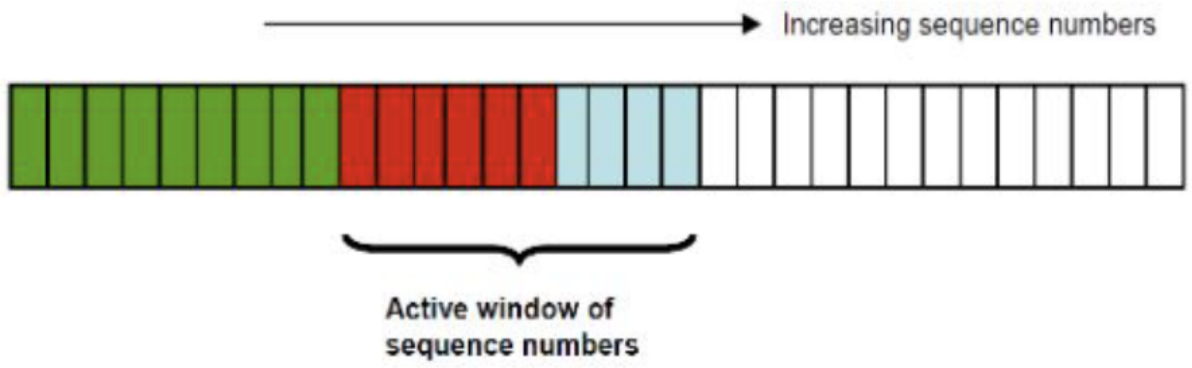
Transport Layer

- Performs packetizing – breaking messages and data from application layer into packets that need to be transported
- Three types of protocol:
  - Stop and wait
  - Pipelined transmission
  - Reliable pipelined transmission with windowing
- Two protocols we use
  - TCP
  - UDP

Stop and wait:



Reliable Pipelined Protocol:



(pipelined transmission is just this without acks)

Network Layer

Focus on getting packets from places to places

A **router** is a device that performs the functionalities of the network layer

Routing Algorithms: determine a route for packets to take from source to destination

- Link-state routing
- Distance vector routing
- Hierarchical routing

Service model: the network layer should forward an incoming packet on an incoming link to the appropriate outgoing link

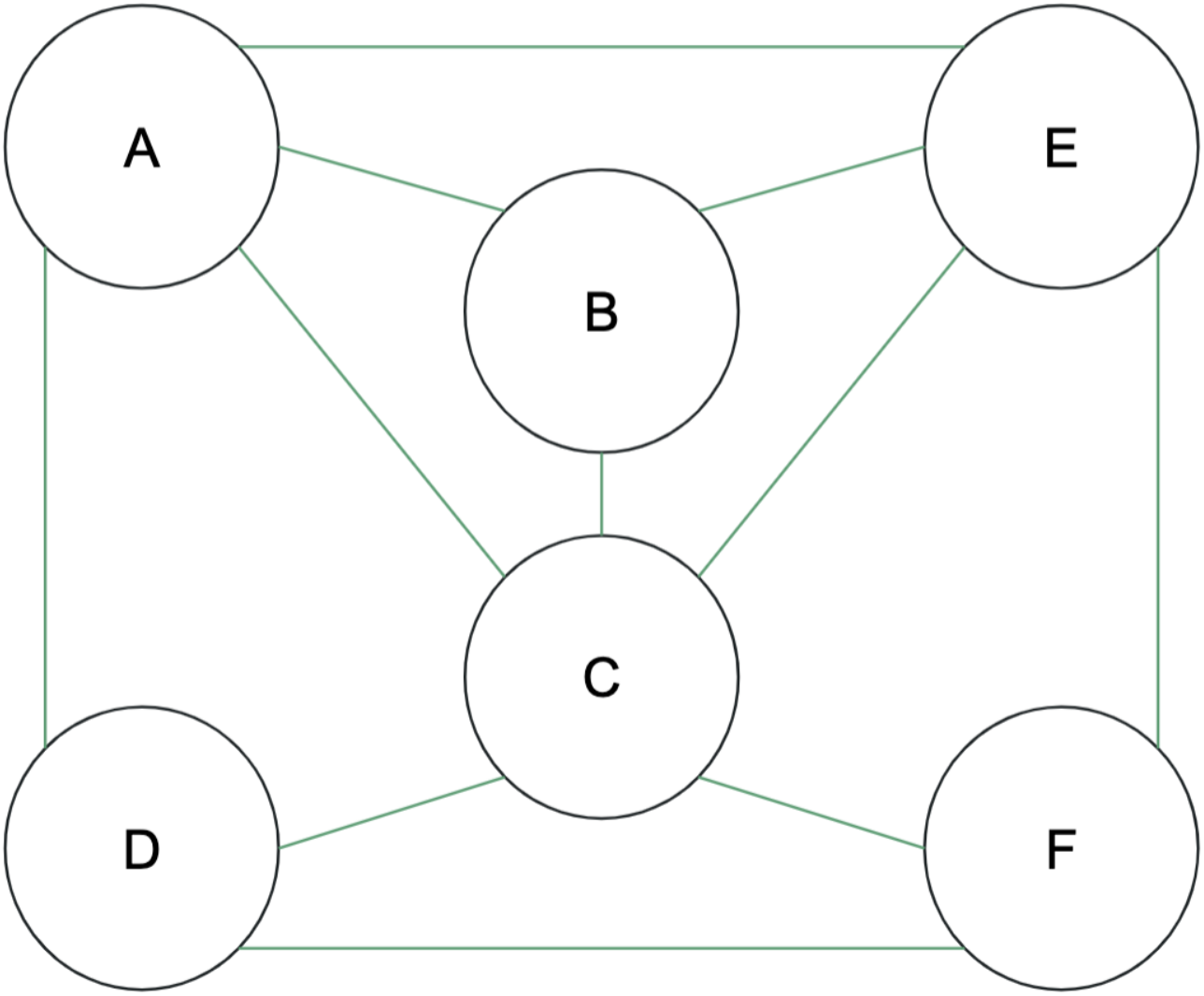
- Virtual circuit
- Datagram

Link State Routing Algorithm

Dijkstra's Link State (LS) Routing:

- Every node knows the global network states

    - LS routing is a local algorithm that uses global information of the link state of the network
- The link state for a node = cost of the paths connected to the node



Distance Vector Routing

Problems with link state routing

- Expensive
- Hard to scale
- Can’t expect every node to have global knowledges

Each device keeps a distance vector table, which is updated periodically to store only the costs to every other node from its neighbors

- DV routing is sync and works with partial knowledge of the link state of the network

We need to update the DV table if one of the table following happens

- Node observes change in link state to neighbors
- Neighbors communicated a change of its costs

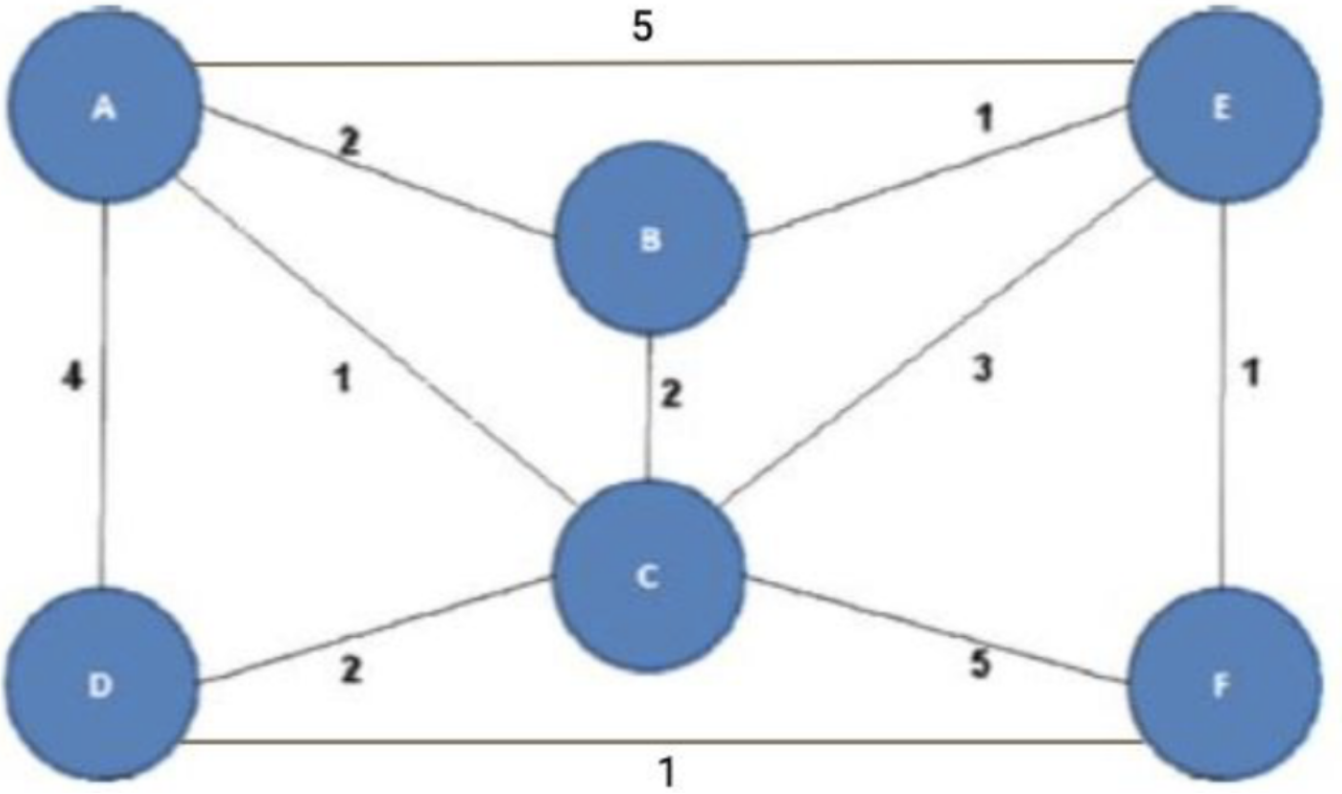


Table for cost through immediate neighbors for Node E

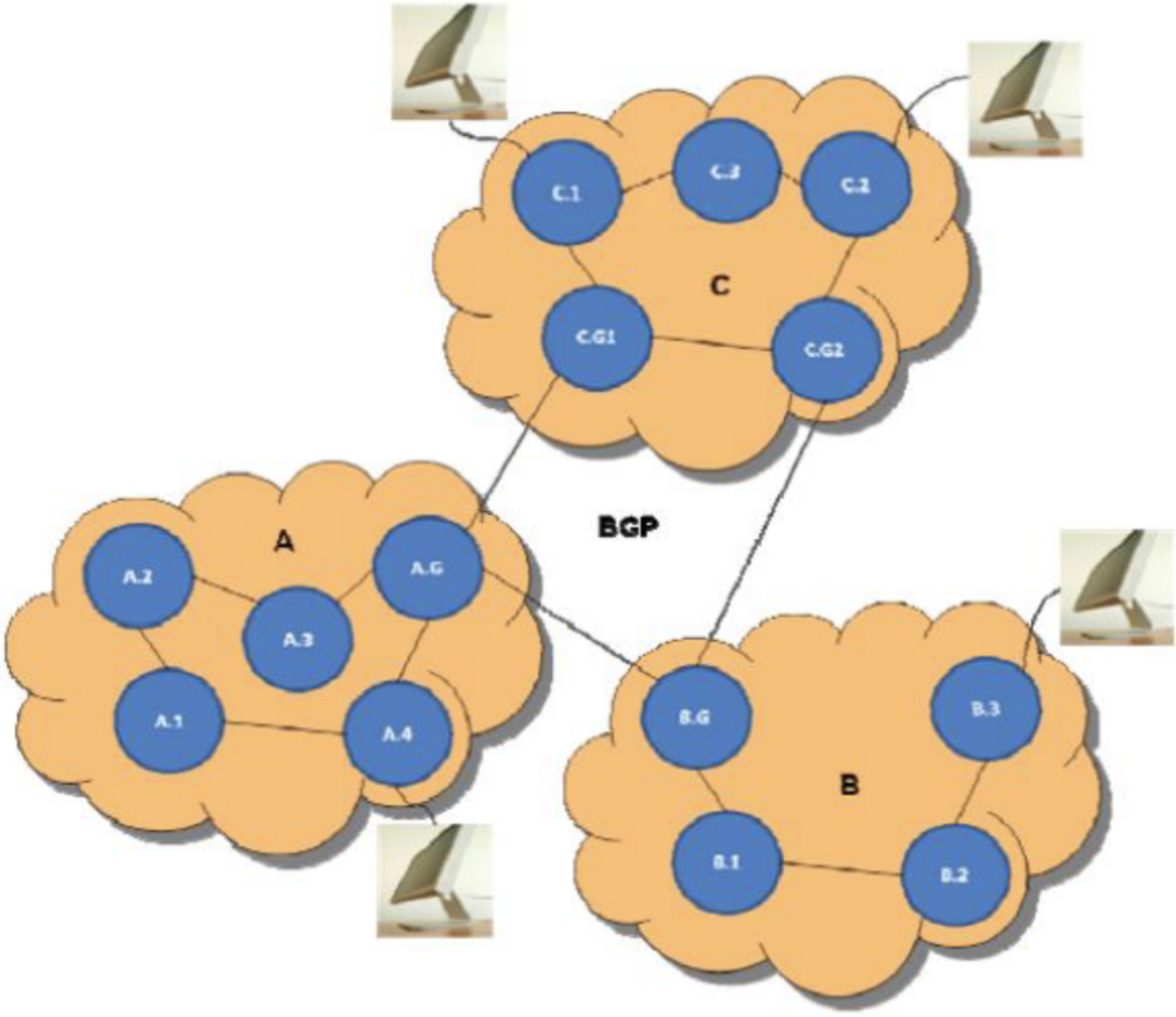
Destination	A	B	C	F
A	5(EA)	3(BA)	4(ECA)	5(EFDCA)
B	7(EAB)	1(EB)	5(ECB)	6(EFDCB)
C	6(EAC)	3(EBC)	3(EC)	4(EFDC)
D	8(EACD)	4(EBEFD)	5(ECD)	2(EFD)
F	9(EABEF)	2(EBEF)	7(ECBEF)	1(EF)

Hierarchial Routing

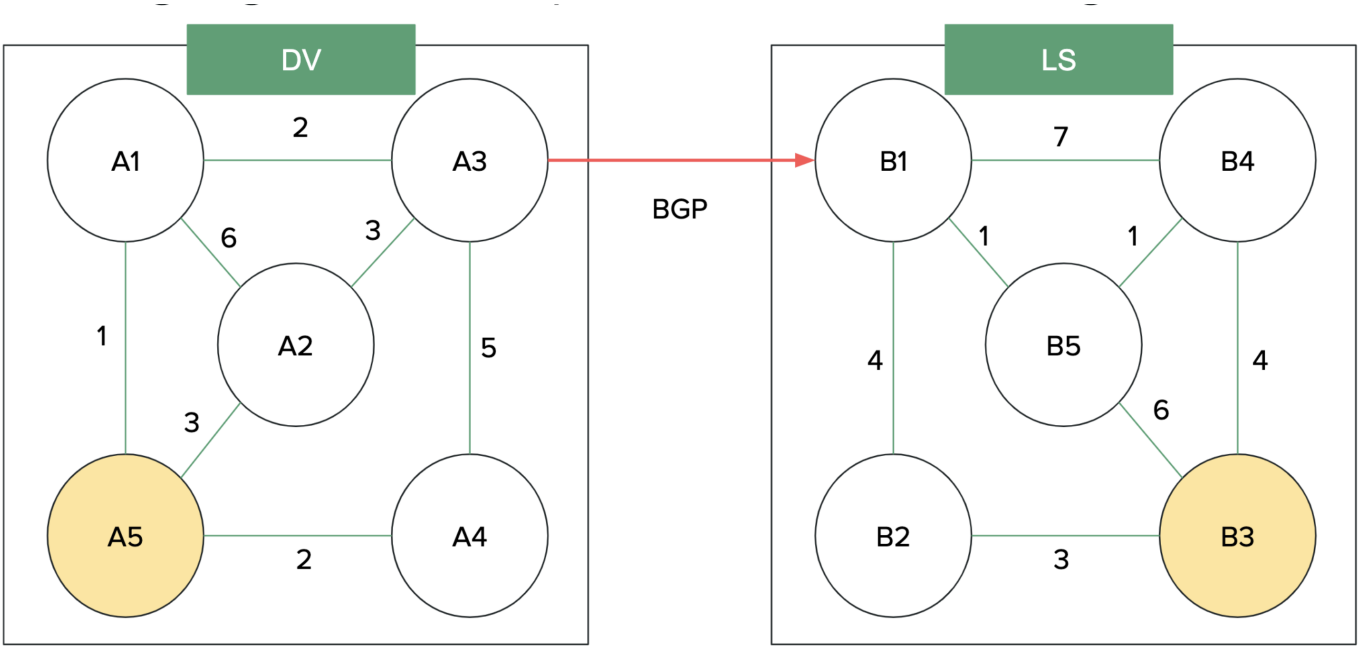
Both link-state and distance vector routing algorithms require us to keep track of lots of information

Divide the network into regions called Autonomous Systems (AS)

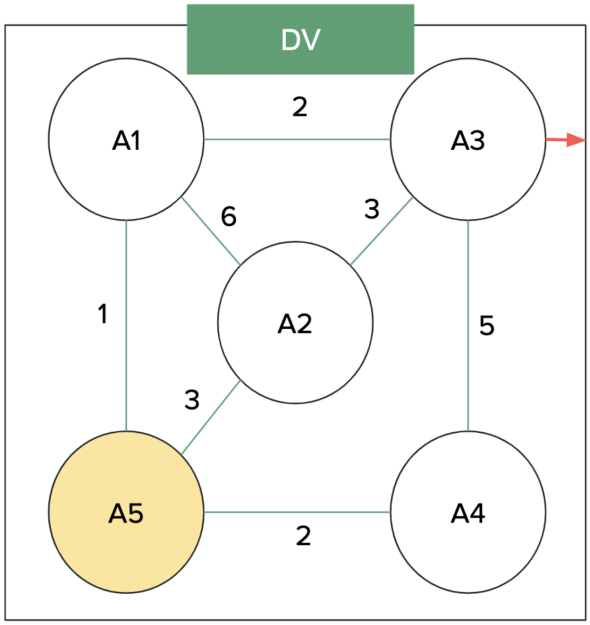
- The network layer features two protocols:
  - Intra-AS: within ASes, can use the first two protocols to organize and search through nodes
  - Inter-AS: routing table!
  - Gateway router = responsible for connecting to the other ASes with the Border Gateway Protocol (BGP)
  - Each AS has a number assigned to it



Routing Algorithm Example: Hierarchical Routing



Let's start with autonomous system A



We must find a way to get to A3 so that we can use a routing table to get to autonomous system B. Let's construct a DV table:

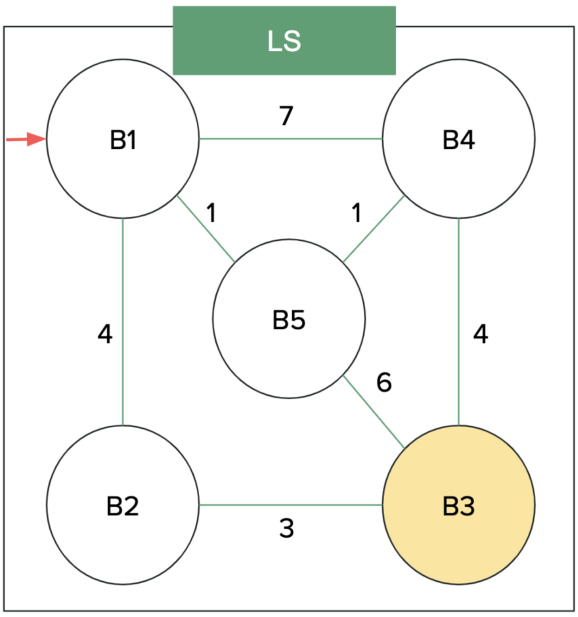
	via		
Dest.	A1	A2	A4
A1	1	7	5
A2	6	3	7
A3	3	6	7
A4	4	8	2

BGP contains a table that includes the current node and its "next hop," that is, what "jumps" to different autonomous systems are supported by the protocol and the hierarchy

How about autonomous system B?

B1 is our starting point, as it is connected to the BGP. Remember, for an LS protocol, all local nodes have global information.

i	New node	B2	B3	B4	B5
0	1	4	inf	7	1
1	1 → 5	4	7	2	x
2	1 → 5 → 4	4	6	x	x
3	1 → 5 → 4 → 3	x	6	x	x

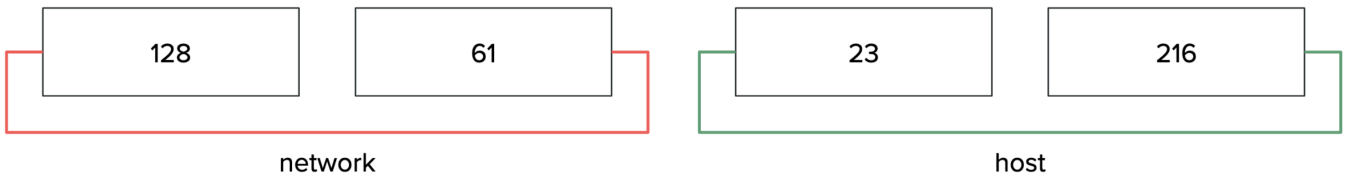


Addressing

Internet Protocol (IP) Address

- IPv4
  - 32-bit addresses
  - Dotted decimal notation
    - Example: 1.1.1
  - CIDR notation: address ranges
    - Example: 10.0.0.1/24
- IPv6
  - 128-bit addresses

But what does each “segment” mean? Let’s take a look at a 32-bit addr example:



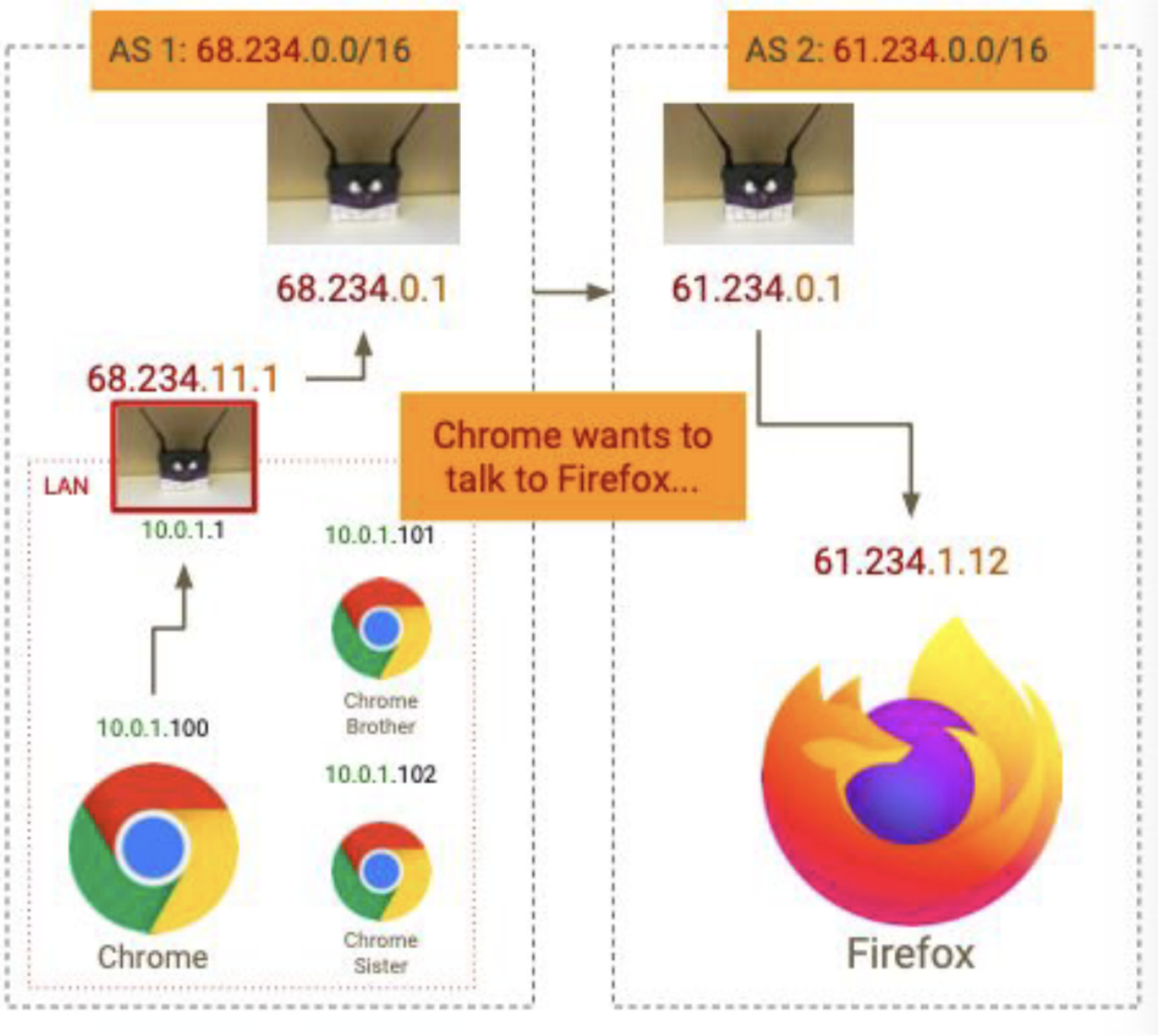
Routing With IP Addresses

- Machines within the same autonomous system are assigned the same network part of the IP address

- Routers must either redirect requests to the link layer or reference the routing table to send packets to the destination address
- We use **port numbers** to track conversations across applications
- **Firewalls** can allow or disallow IP traffic based off of addresses, protocols, etc. They can also decide what permissions each port or address gets

Network Address Translation (NAT)

- Allows us to expose an entire local network as a single IP
- Each device within the local LAN still has its own IP
- The router keeps a NAT translation table to keep track of connections
  - Replace the source IP with the router's IP for outgoing traffic
  - Replace destination IP with device IP for incoming traffic
- Different LANs can have the same address range!



Does the packet have all the information to support this?

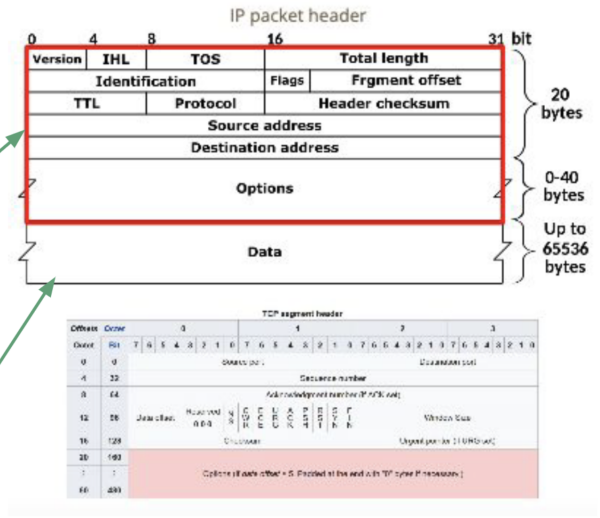
Absolutely! Let's look at how a packet is structured:

Header:

- dest and src addresses
  - where did you come from where did you go
- number of packets
  - this is done with "total length"
- sequence number
- packet size
- checksum
- and more!

Payload:

- The actual data!



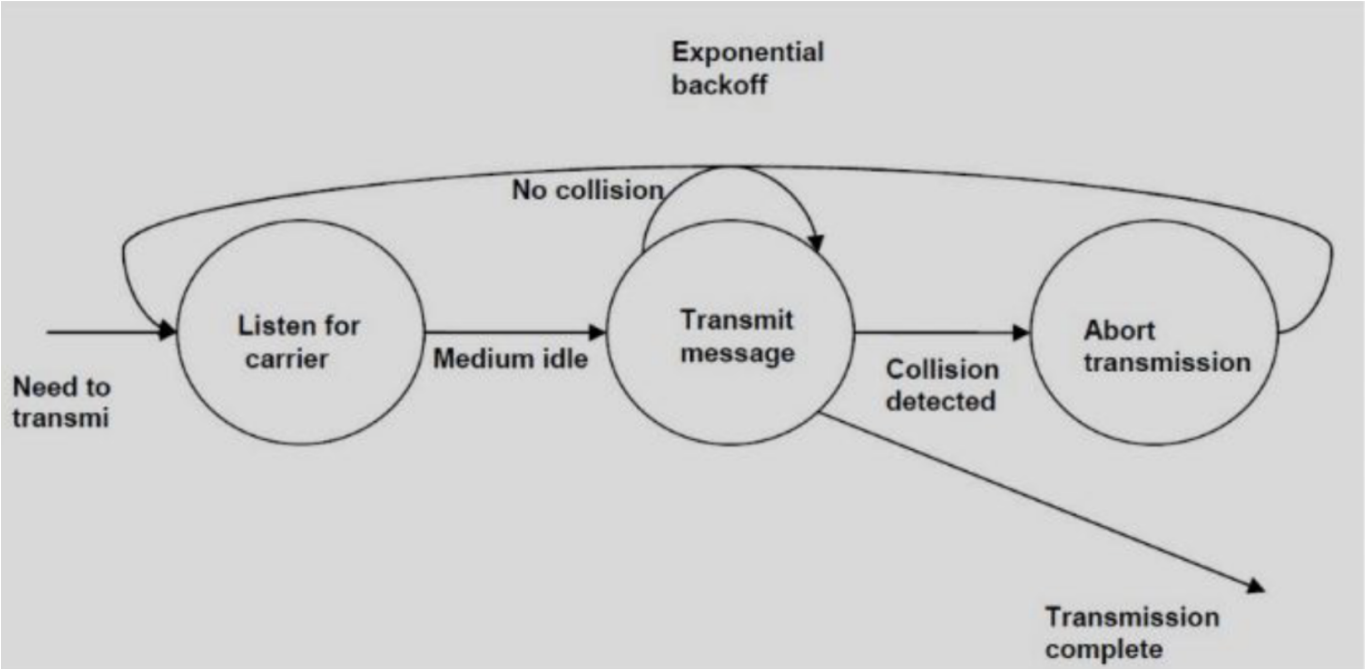
Link Layer

Term	Definition
Collision Domain	All nodes that share a physical ethernet; only one can transmit at a time
Broadcast Domain	All nodes that can hear a broadcast frame
MAC Address	Unique address that identifies a computer at the Link Layer. Stored on the Network Interface Card (NIC)
XBASEN	Describes baseband transmission – XMBps on BASE band with physical medium N (usually T for twisted pair)

Ethernet

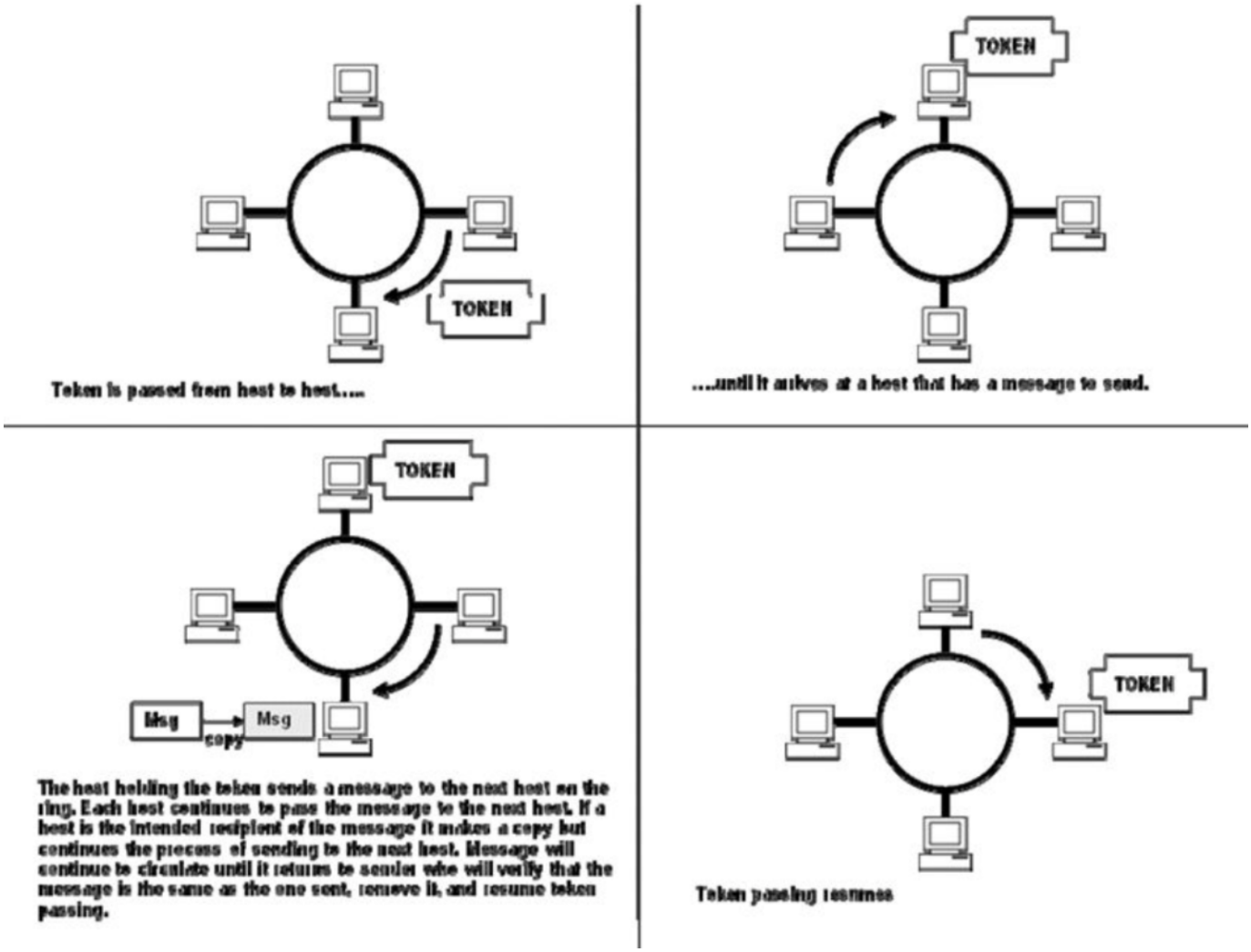
- Ethernet is the most common Link Layer protocol
- Has random access & an arbitrary number of nodes a potentially hundreds of meters long bus
- Transmit one frame on a wire
- Uses a protocol called carrier sense multiple access/collision detect (CSMA/CD) to ensure no collisions on this bus/no communications are sent at the same time

- This means that computers within a collision domain can talk to each other (under this protocol)
- Nowadays, use switched Ethernet so there are no collisions (more to come)
- Wireless communications use CSMA/CA (avoidance instead of detect)



Token Ring

- A token (special bit pattern) keeps circulating on a wire
- A node can only send when it has the token; when it transmits a frame, it puts the token back on the wire for someone else to send a frame



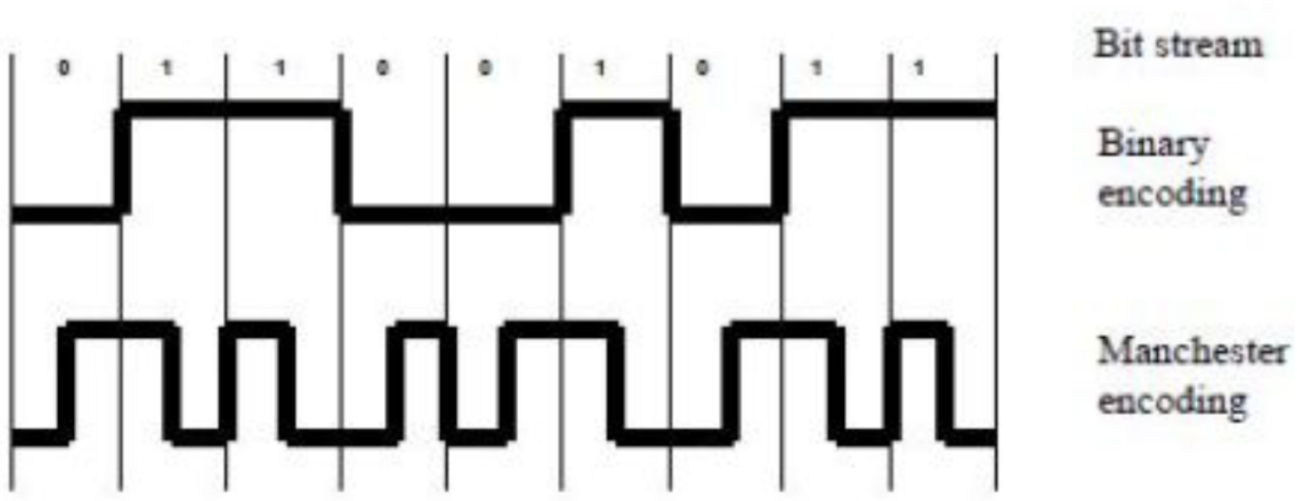
Connecting Link Layer to Transport Layer

- The payload of Ethernet frames is the IP packet, so we see both
- Ethernet frame header has MAC address; IP packet header has IP address
- How does Ethernet know what IP address to put/use?
    - Address Resolution Protocol (ARP) is used to discover MAC addresses
        - When you need to know an MAC address, broadcast a message (network layer) asking who has it
        - Host responds that they have it – that frame also contains their MAC address (it's in the header!)
        - Keep track of an ARP Table with these mappings; discard entries when they are stale

Physical Layer

We use the Manchester Encoding to represent logical 0s and 1s

- A logical 0 is a low → high transition
- A logical 1 is a high → low transition



Networking Hardware

Hub: Relays messages from one host to others

- What datapath structure is most like this?
- Computers connected to each other with hubs share a collision domain
- A **repeater** repeats signals – electrical signals decay in strength over distance, so repeaters keep signals strong along a transmission medium
    - Basically a hub that boost signal strength – you can use either term

Bridge/switch: Connects collision domains to other collision domains so that there are no collisions between them

- Switch = multi-way bridge, generalization
- Can be used to create Virtual LANs (VLANs) so that computers that are physically distributed and connected to different switches to form a LAN

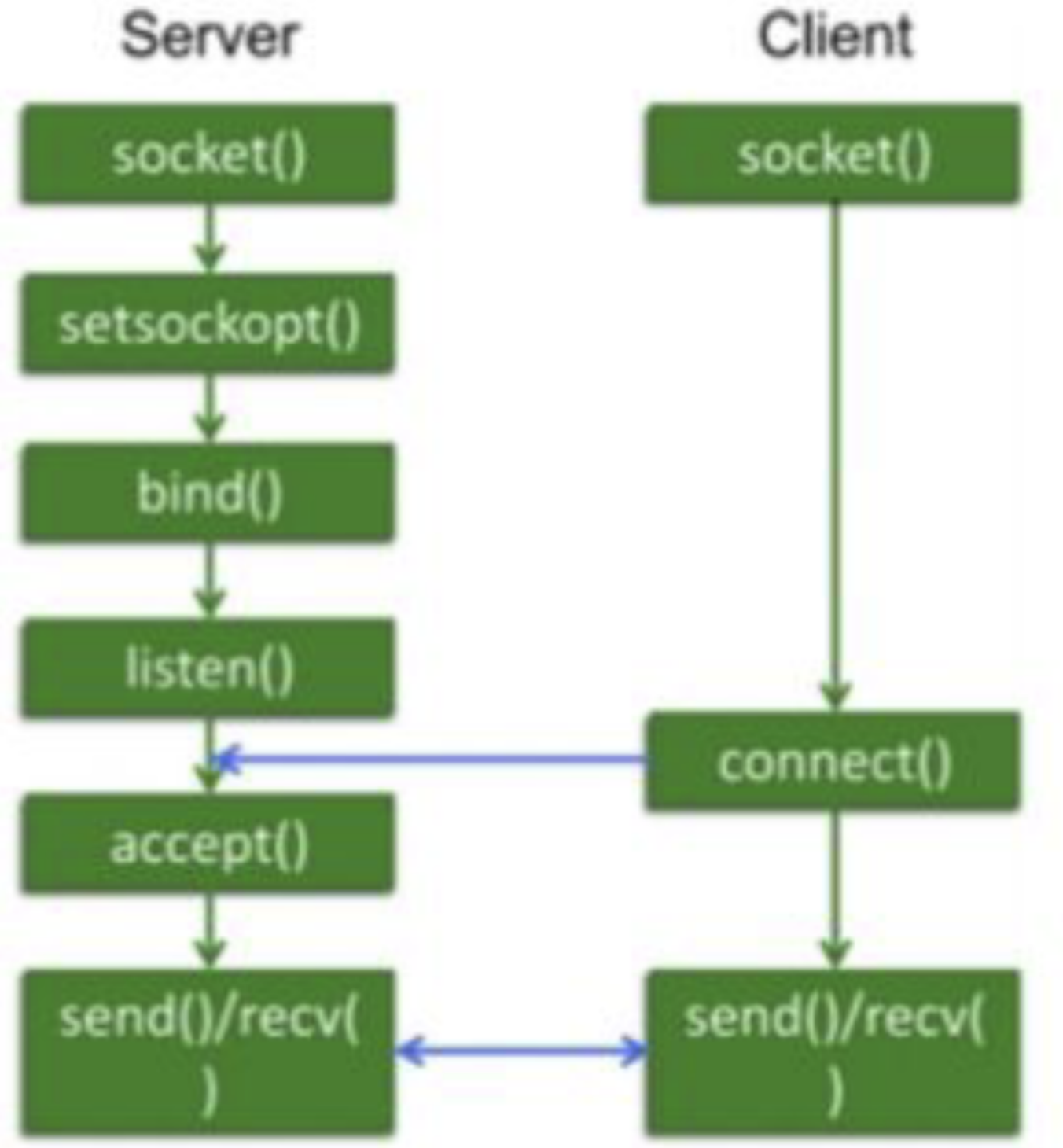
Router: A host on a LAN that understands IP addresses and can communicate with other routers from other networks

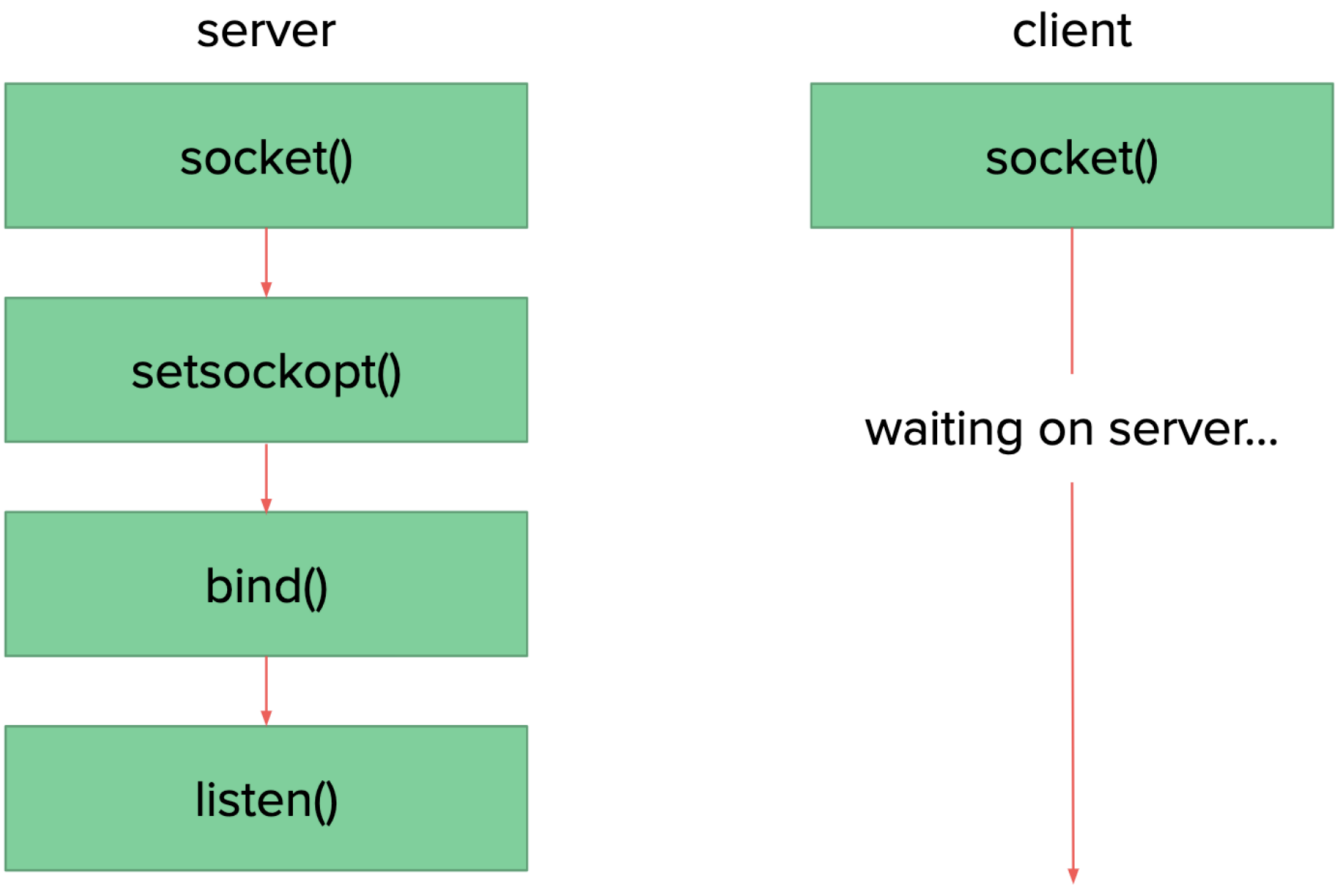
Client Server Communication

Networking Socket Programming

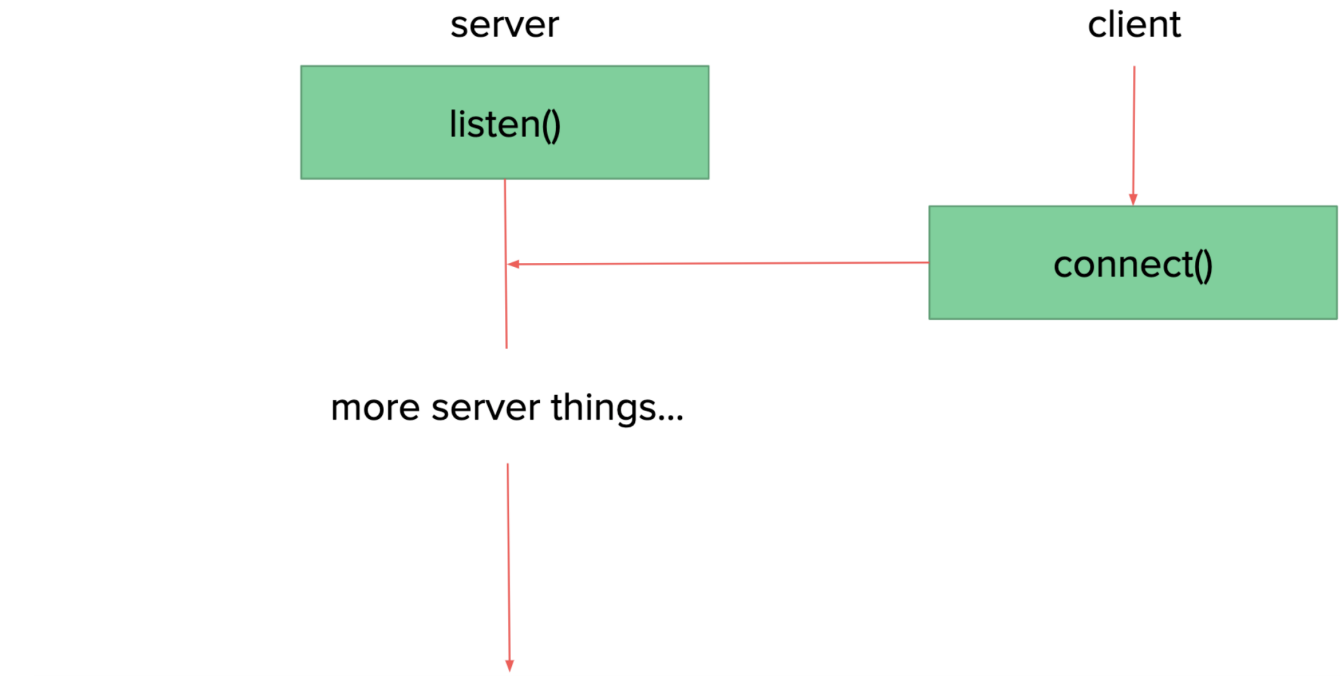
- Determines how to write programs to deal with networking (at the 3rd layer – network layer)

- The paradigm:
  - Server sets up a listener bound to a specific destination port
  - Client makes a connect request to the listener
  - The listener accepts the connection
  - Client and server can communicate using send and recv (write and read) syscalls!

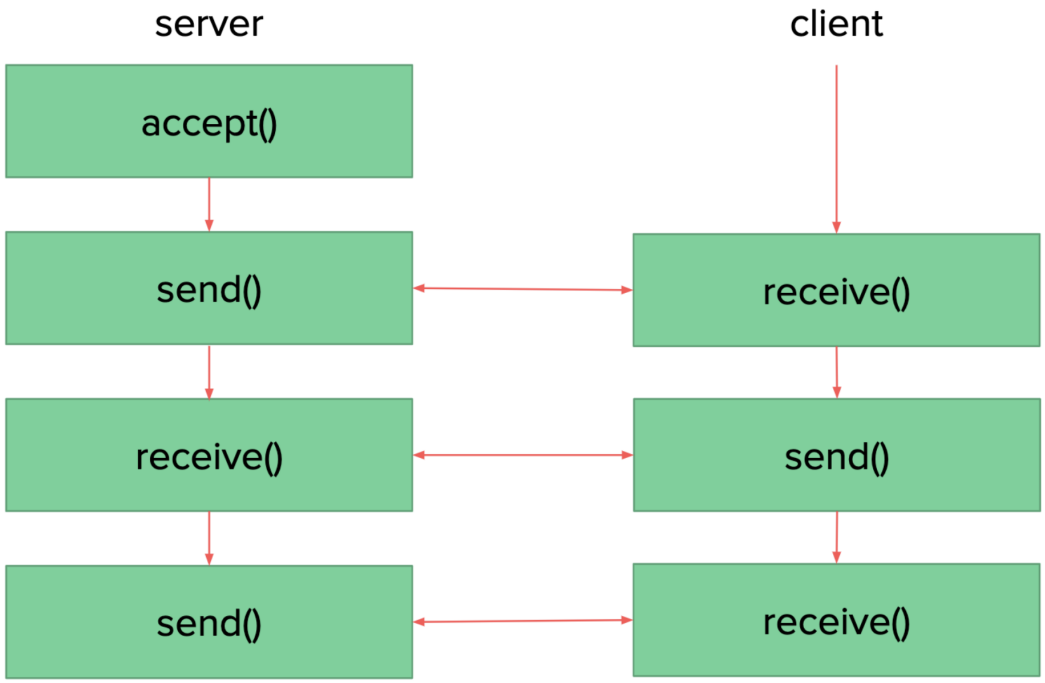




What happens once the server is listening?



Connection successful!



Network Sockets

We can use a higher-level abstraction for a client to call a procedure on a server – a remote procedure call (RPC)

- Basically puts the function calling into the payload of a packet and receives the result back in another packet

Example of a use:

- Network File System (NFS) uses servers as file systems

File Systems

What is a file?

- Collection of information with attributes associated with the information
- Abstraction for I/O devices, since a device serves as either a source or sink of information
- Allows a user program to interact with I/O in a device independent manner

What is a file system?

- Contains a lot of files (#noshitsherlock)
- Stores files in a particular organized way to ensure that files are easy to locate and search through
- How are these files stored?

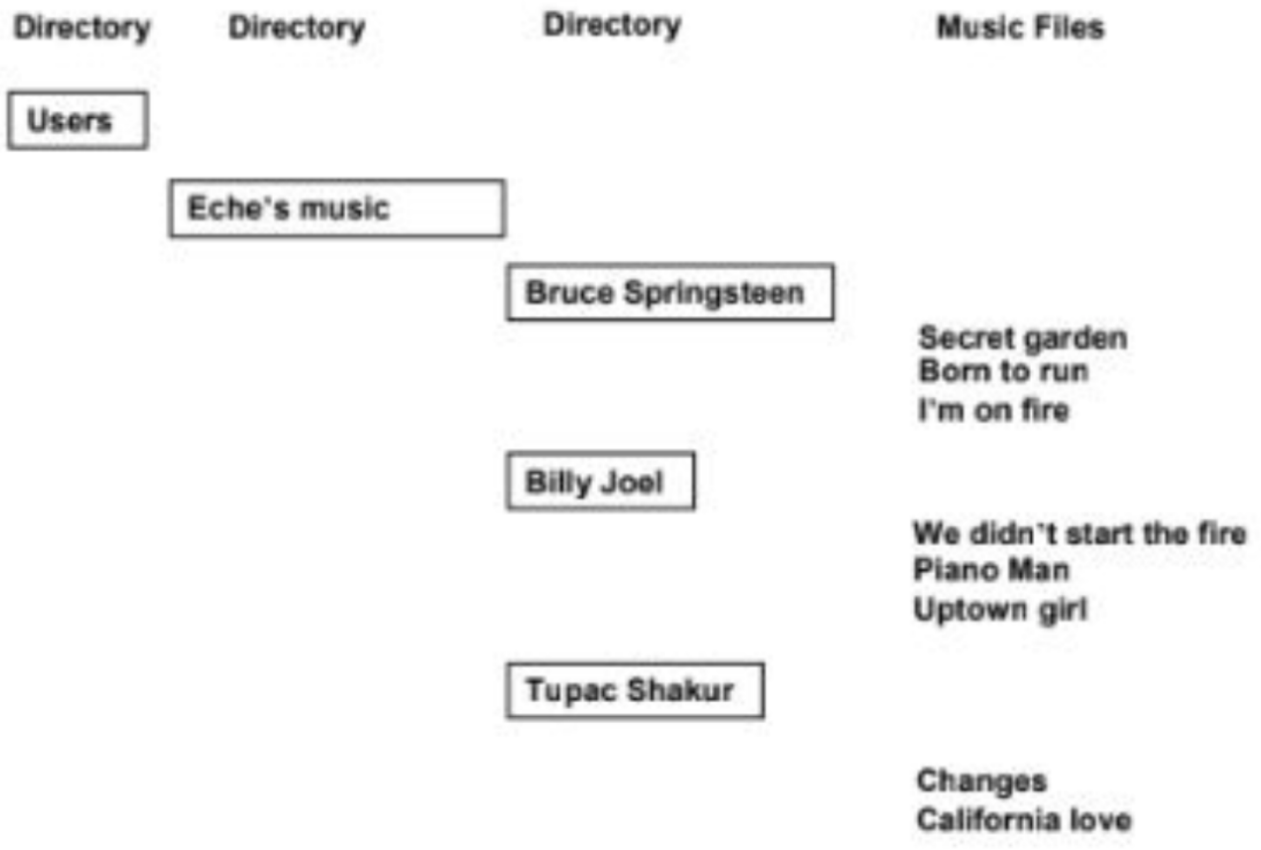
File Attributes

- Called **metadata** = represents space overhead and requires careful analysis as to its utility

Attribute #1: Name

- Allows the logical identification of the contents of a file
- Naming schemes:

  - Single level: names must be unique
  - Two-level: User → specific file
  - Multi-level: each part of the name is unique only with respect to the previous parts of its name (like a tree!)



Attribute #2: Access Rights

- Who owns the file?

- What kind of privileges does each user with access have?
    - For instance, it's safe to assume that the owner of a file has more access privileges than a random user that the file is shared with
- Common privileges:
    - read
    - write
    - execute
    - change ownership (chown)
    - change permissions (chmod)

Storing Files

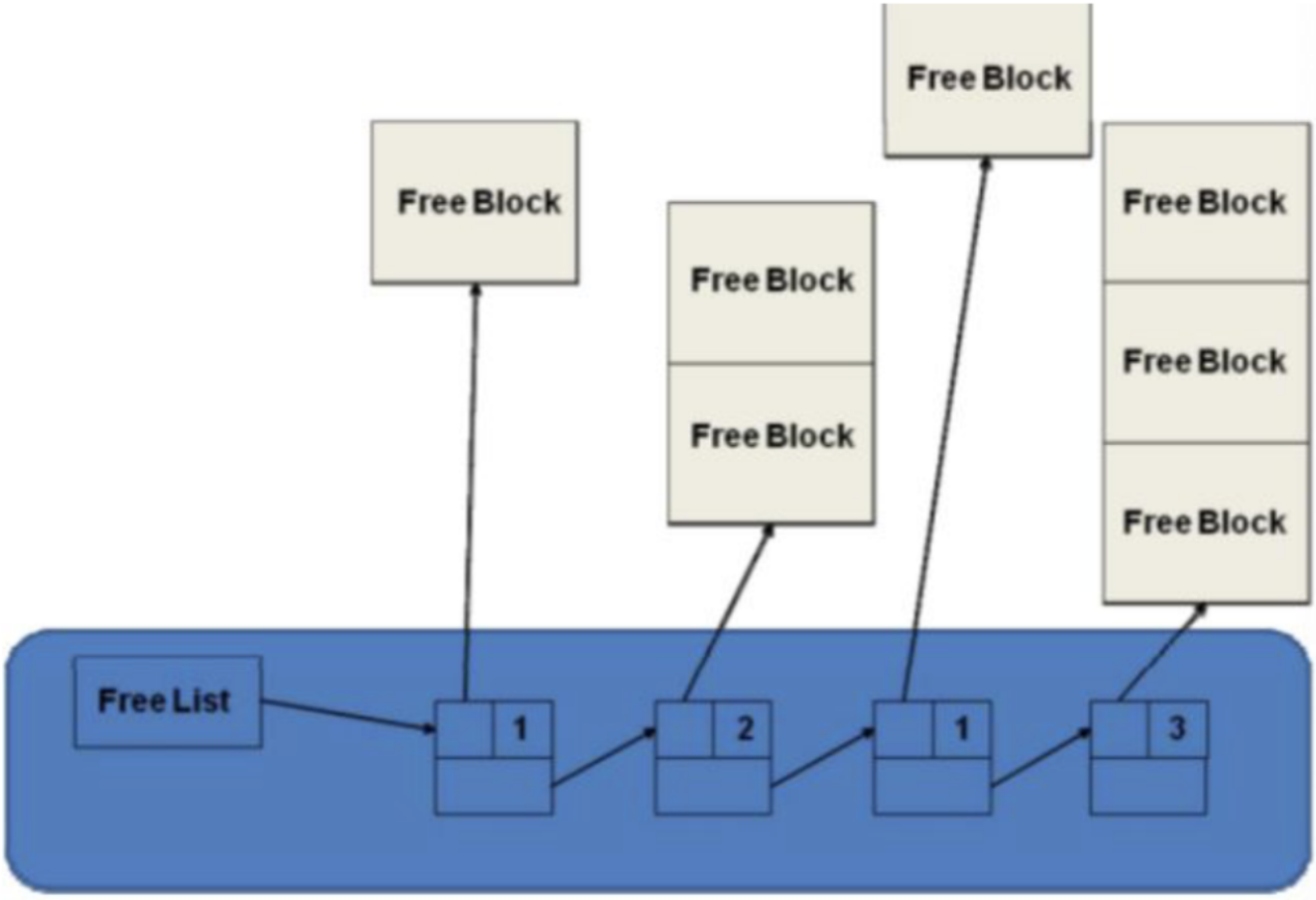
If we make modifications to a file in a filesystem for a particular process, we need to account for storing the file to ensure that no data is lost. The way in which we implement and store file systems on disks depends on 5 merits.

We want:

1. Fast sequential access
2. Fast random access
3. Ability to grow a file
4. Easy storage allocation
5. Efficient space utilization

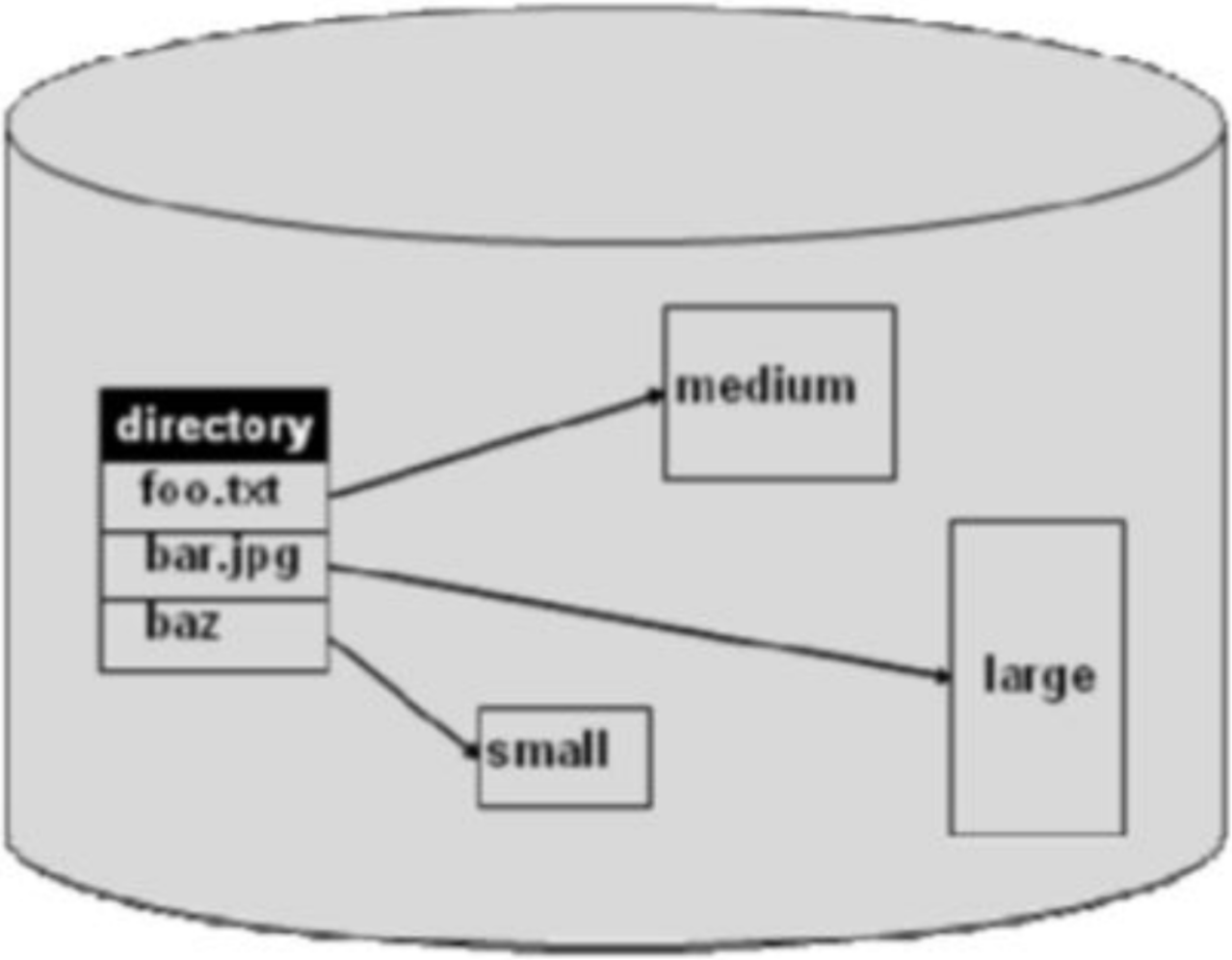
Strategy #1: Continuous Allocation

- maintains a "free list of available disk blocks
- the file system allocates a fixed size for each file (does this remind you of something?)
    - the amount of space allocated depends on what the file system is storing (audio files, text files, etc.)
- the file can't grow beyond this max size
- how do you allocate a file to a free block?
    - first fit
    - best fit



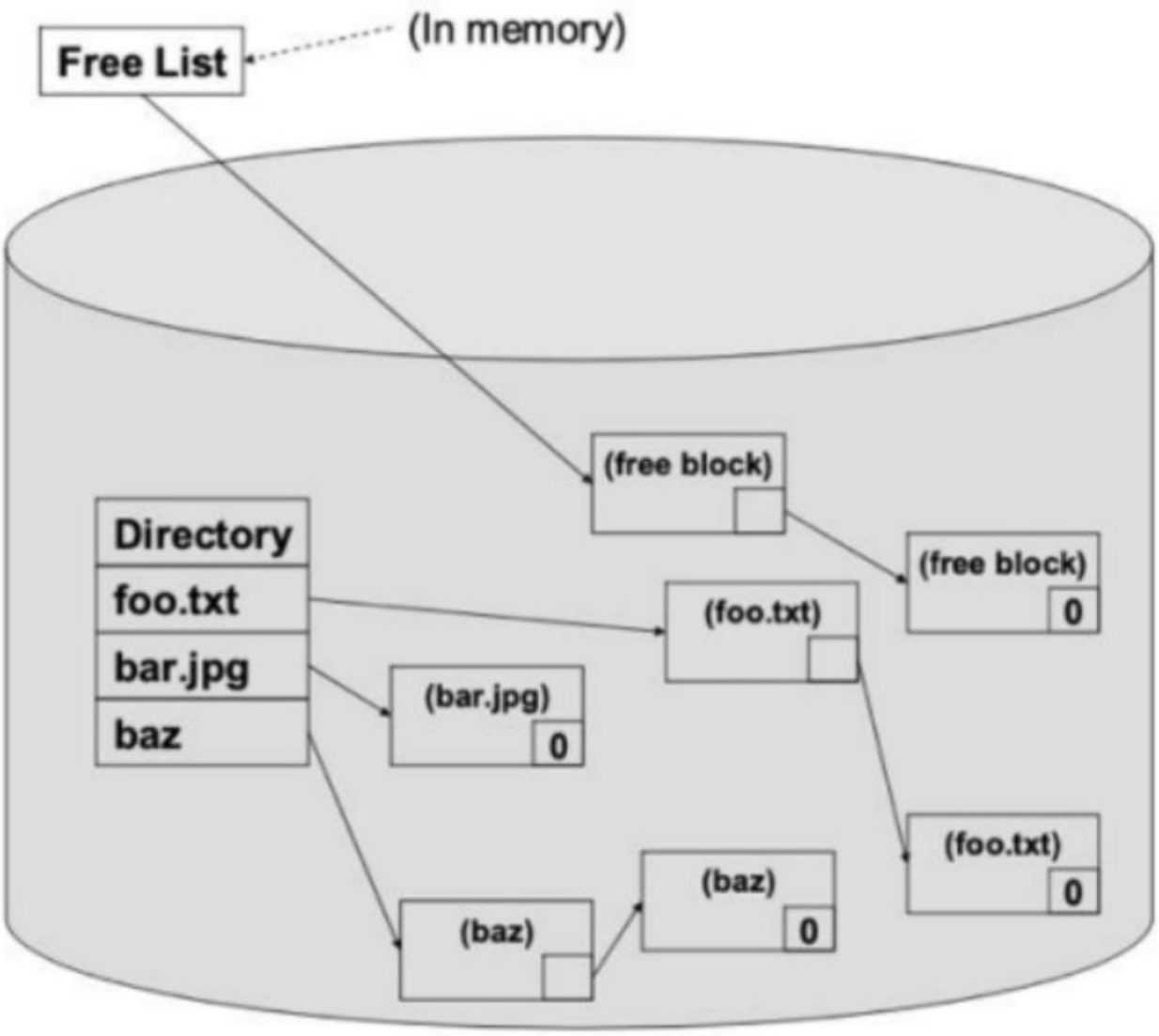
Strategy #2: Contiguous allocation with overflow

- Overflow zone allows for files to exceed their max space because extra information can just go in the overflow zone
- Great because now your files can be bigger! less fragmentation!
- Not so great because now you have to look in the file in the filesystem AND the overflow region for data
    - Search time increases



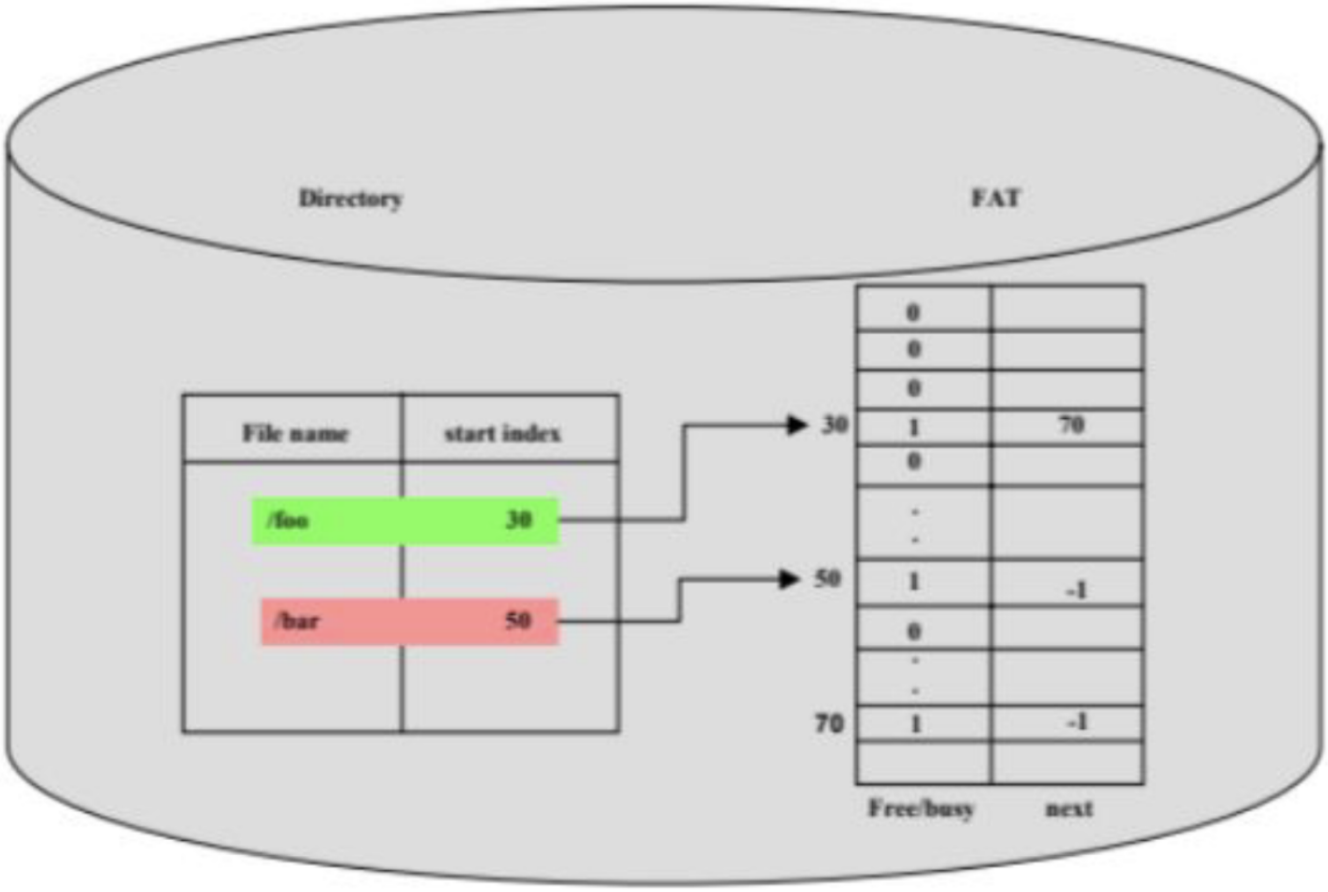
Strategy #3: Linked Allocation

- Deals with allocation at the level of individual disk blocks
- Each block for a given file points to its next block
- We still maintain a free list of the free blocks in memory
- Not the best for random accessing
- Good at file growth and space efficiency



Strategy #4: File Allocation Table (FAT)

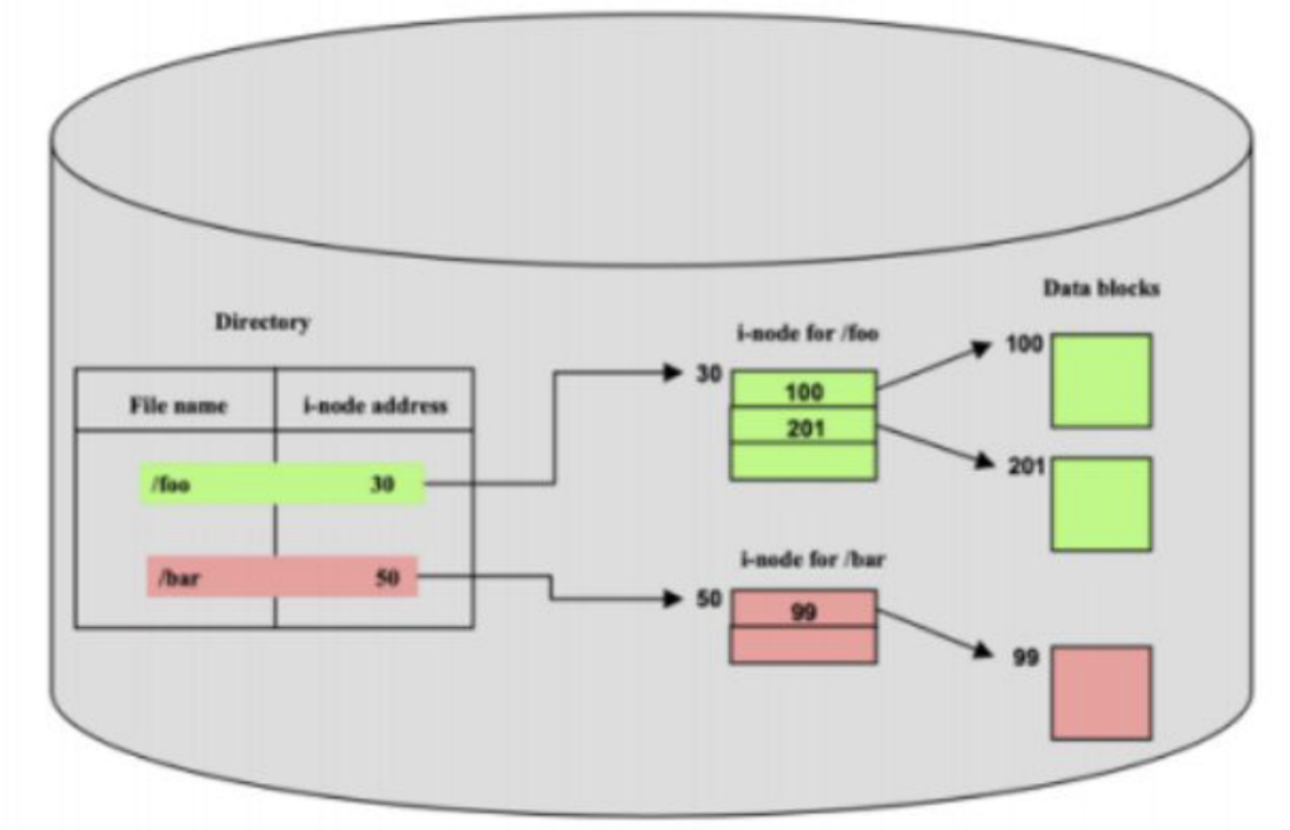
- Variation of linked allocation
- This file allocation table contains the linked list of the files currently populating in the disk
- The table also includes information about whether a particular block is free or not (as opposed to using a freelist)
- This is actually kinda similar to paging!



Strategy #5: Indexed Allocation

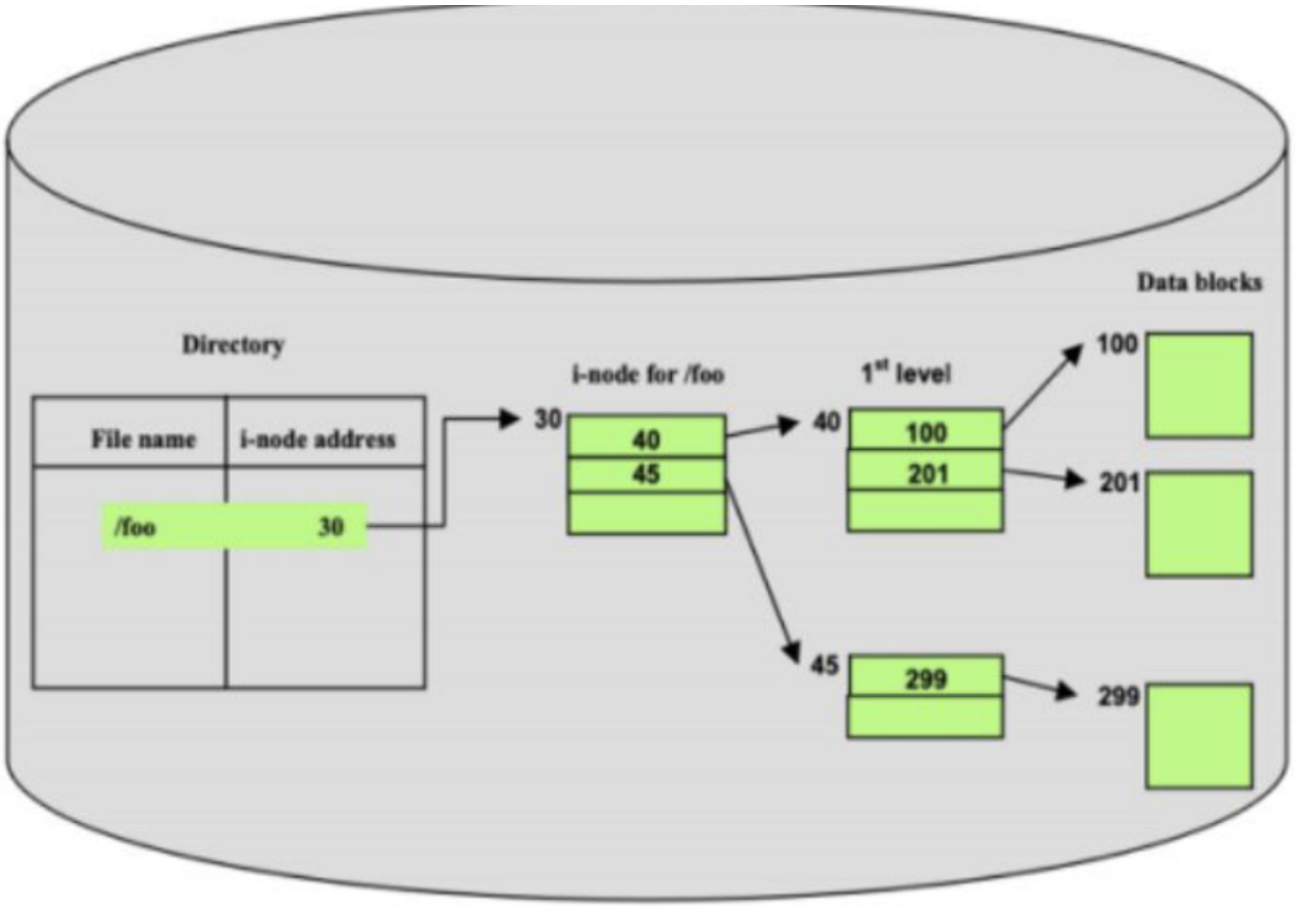
- Each file has an index disk block
  - Fixed-size data structure that contains addresses for data blocks that are part of that file
- Aggregates data block pointers for a file scattered all over the FAT data structure into one table per file (like the image)
- This table is called the i-node (index node)
- keeps a free list like linked allocation

like FAT but a lil better ;)



Strategy #6: Multilevel Indexed Allocation

- Fixes the limitation in the indexed allocation by making the inode for a file an indirection table
  - if there's one level indirection, then each inode entry points to a first level table, which points to the data blocks
  - if there's two, then you go to the first-level table, then a second-level table, then the data block
- Tradeoff
  - too few levels of indirection and this collapses to indexed allocation
  - too many: time and space complexity increases



Strategy #7: Hybrid Indexed Allocation

Combination of 5 (indexed) and 6 (multilevel indexed) allocation strategies

- every file has an inode but if the files are bigger than the capacity of their direct blocks, we use levels of indirection for additional data blocks

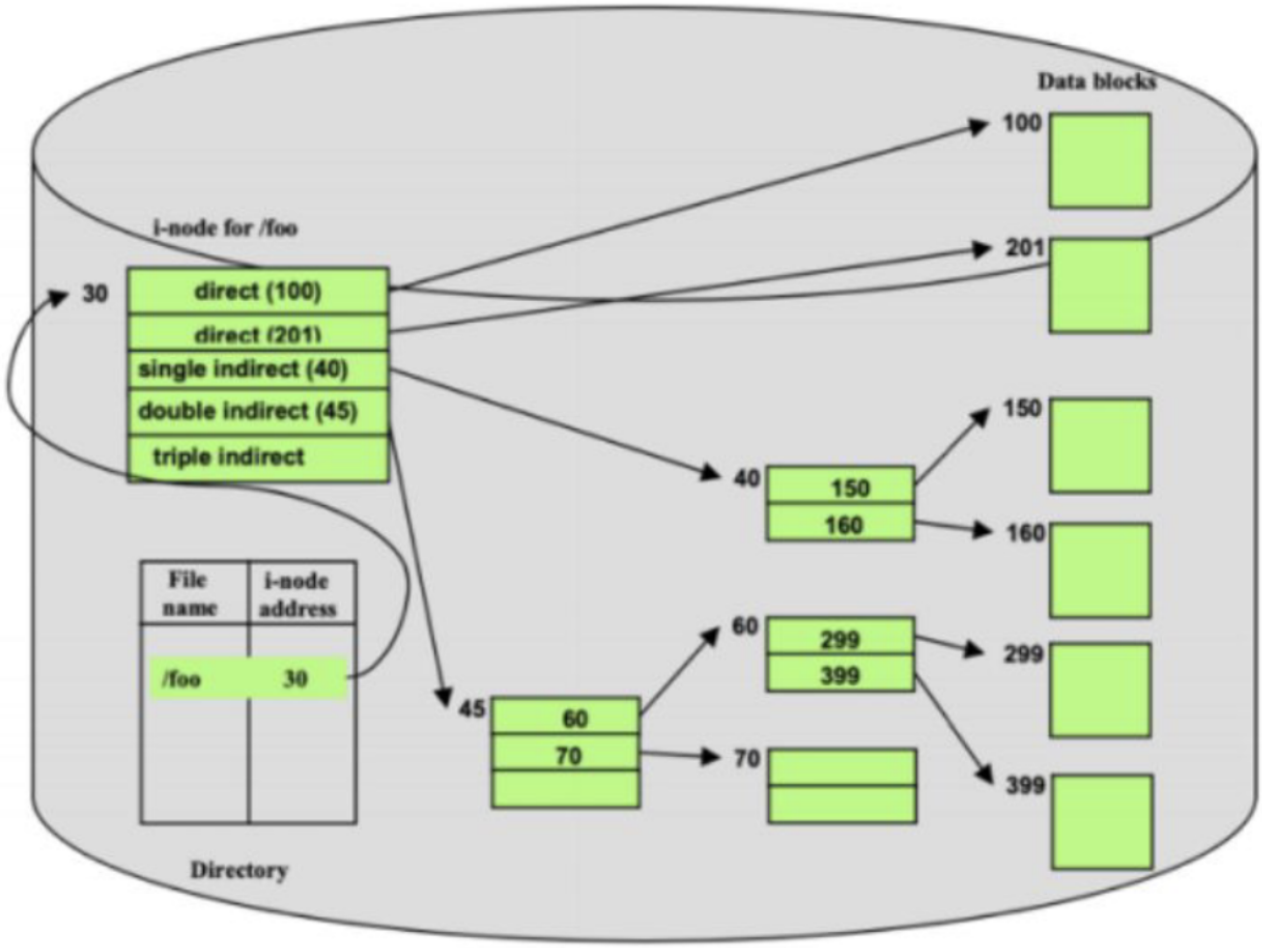


Figure 11.10: Hybrid Indexed Allocation

strategy	file rep.	free list maintenance	#1: sequential access	#2: random access	#3: file growth	#4: allocation overhead	#5: space utilization
Contiguous	contiguous blocks	complex	very good	very good	messy	Medium to high	Internal and external frag.
Contiguous w/ overflow	contiguous for small files	complex	very good for small files	very good for small files	eh	Medium to high	Internal and external frag.
Linked List	non-contiguous	bit vector	good but depends on seek time	no	very good	Small to medium	Excellent
FAT	non-contiguous	FAT	good but depends on seek time	good but depends on seek time	very good	small	Excellent
Indexed	non-contiguous	bit vector	good but depends on seek time	good but depends on seek time	limited	small	Excellent
Multilevel Indexed	non-contiguous	bit vector	good but depends on seek time	good but depends on seek time	good	small	Excellent
Hybrid	non-contiguous	bit-vector	good but depends on seek time	good but depends on seek time	good	small	Excellent

Disk Terminology

- Disk - consists of circular platters with top and bottom surfaces
- Platters - concentric tracks around center spindle
- Tracks - broken into sectors; contiguous recording of bytes and information
- Sectors - fundamental unit of recording on a disk
- Logical Cylinder - set of corresponding tracks on surfaces

Magnetic read/write heads do NOT touch surfaces and are connected to the shaft using arms to form the head assembly

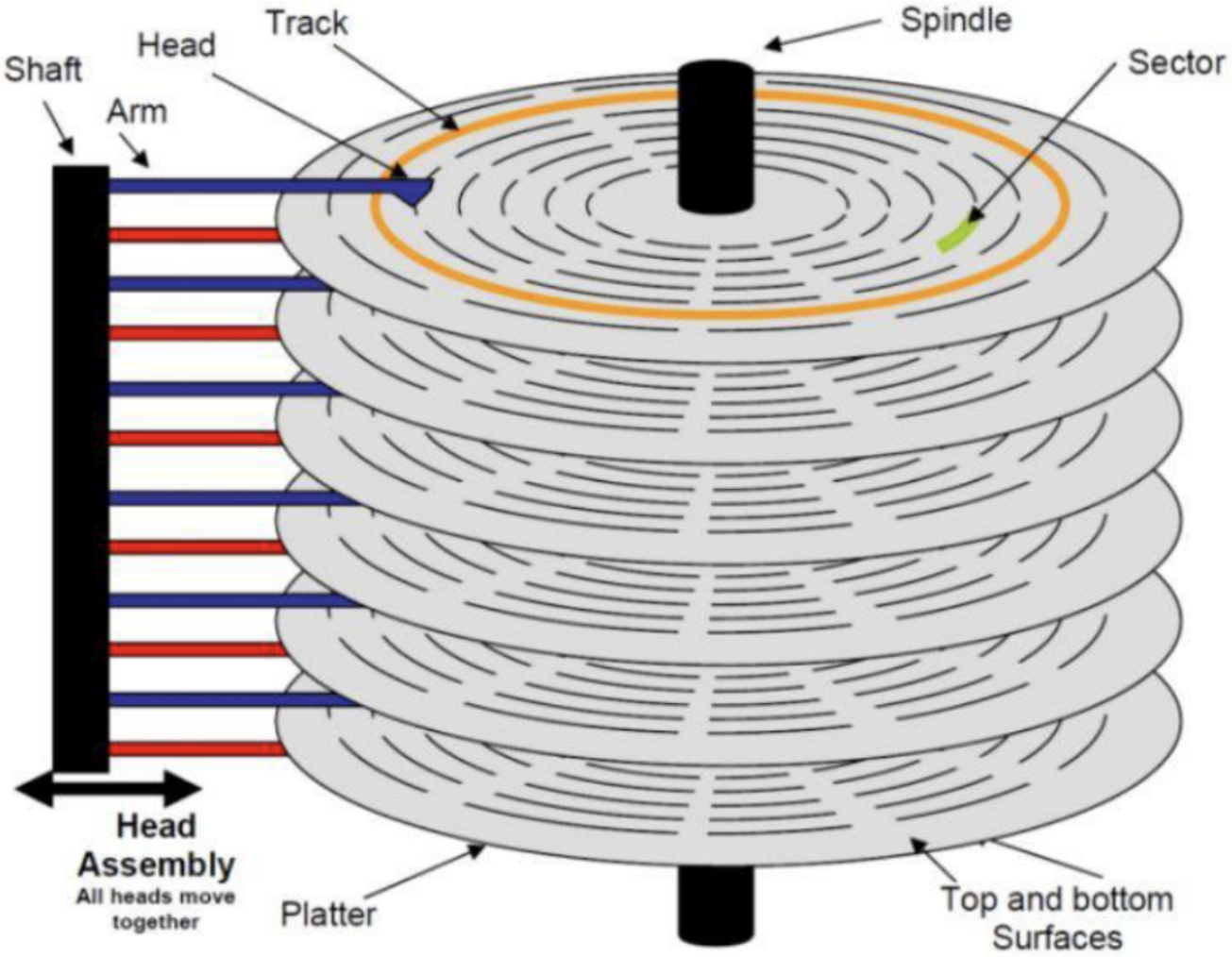


Figure 10.10: Magnetic Disk

Using a Disk

Reading /writing to a disk takes time

- Seek time = time to move the head to the right cylinder

- Rotational Latency = time to spin disk to get the right sector under the head
- Data Transfer Time = time to transfer data between head and disk

Formulas

P = # of platters

N = # of surfaces per platter (1 or 2)

T = # of tracks per surfaces

S = # of sectors per track

B = # of bytes per sector

R = rotational speed of disk in RPM

$$\text{Capacity} = P \cdot N \cdot T \cdot S \cdot B$$

$$\text{Data Transfer Rate} = S \cdot B \cdot R \div 60$$

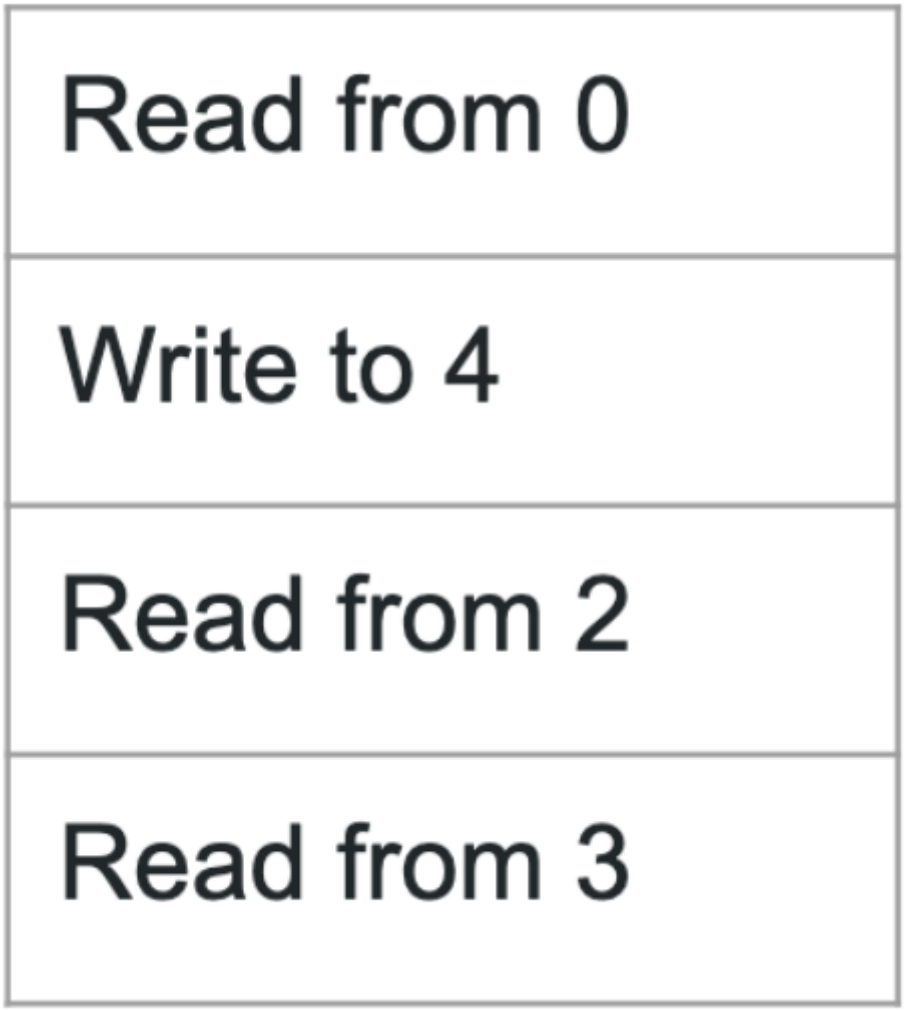
Disk Scheduling

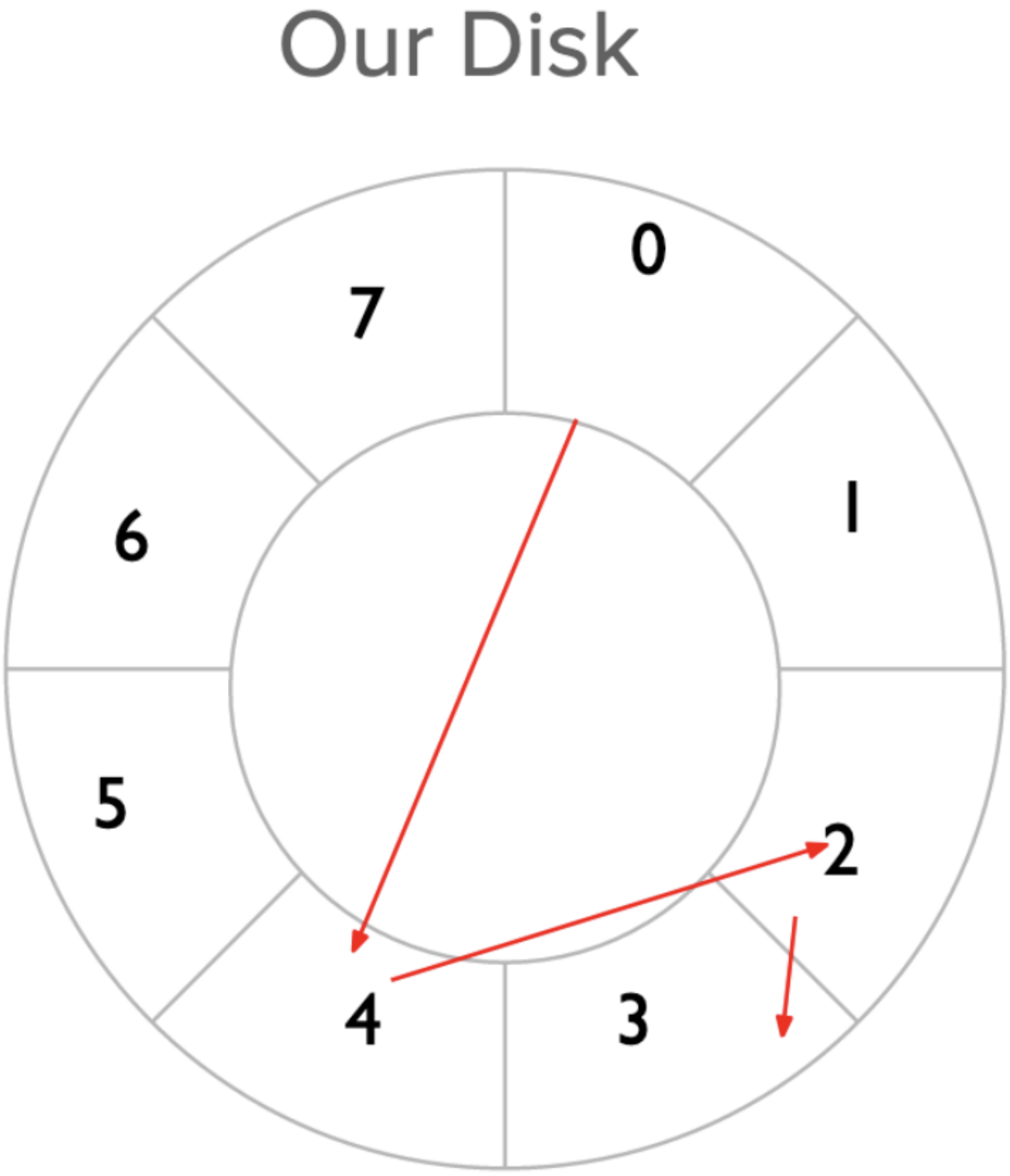
Goal: minimize the time cost of spinning the physical hardware and moving read/write heads

FCFS

Disk requests handled in order of when they arrived

- Slow because lots of hopping around


Read from 0
Write to 4
Read from 2
Read from 3



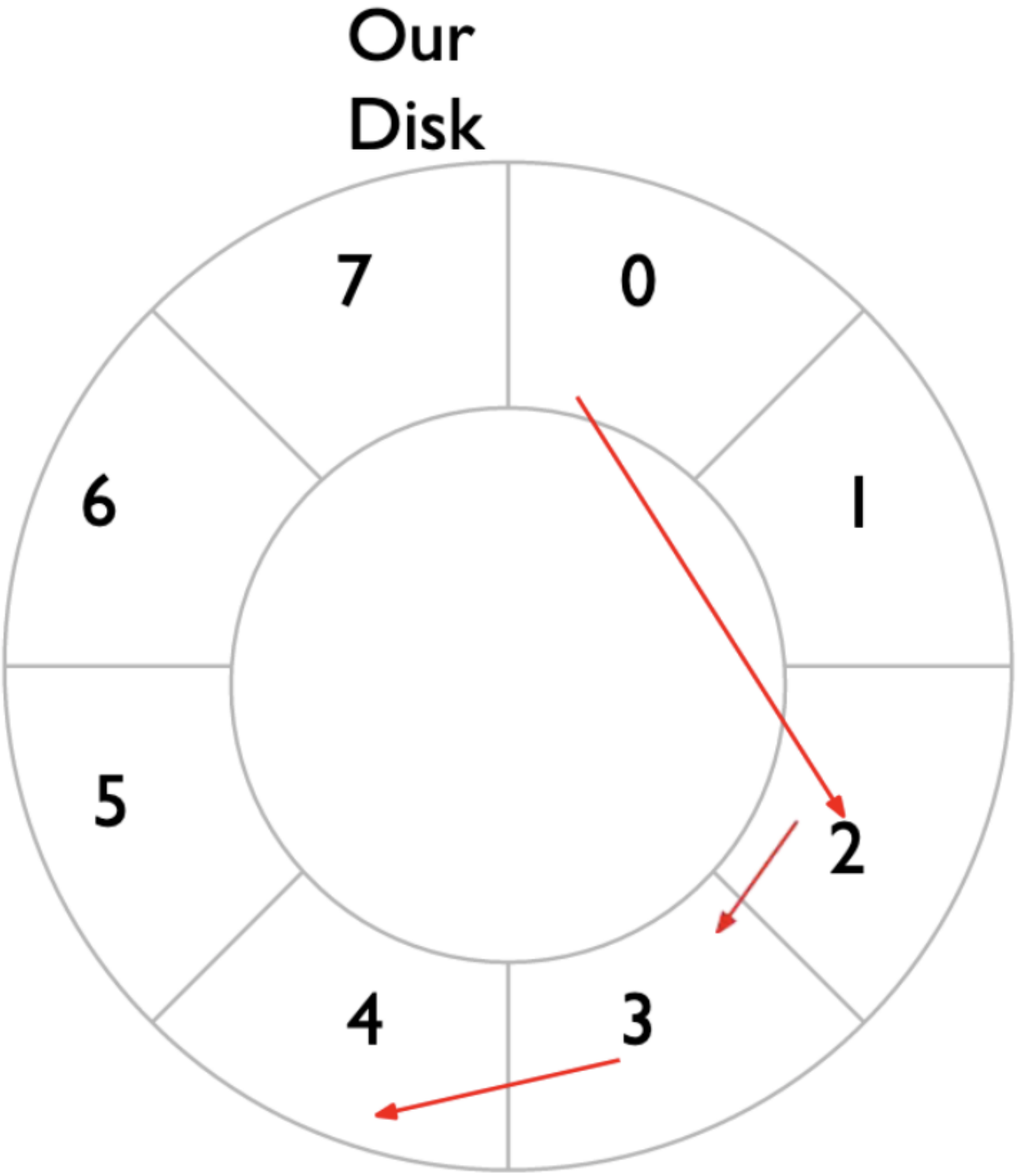
Shortest Seek Time First

Disk requests handled in order of shortest seek time first

- Good throughput
- Short waiting time
- Minimize head movement

BUT

- Doesn’t preserve order of memory accesses
- Starvation possible



SCAN (elevator algorithm)

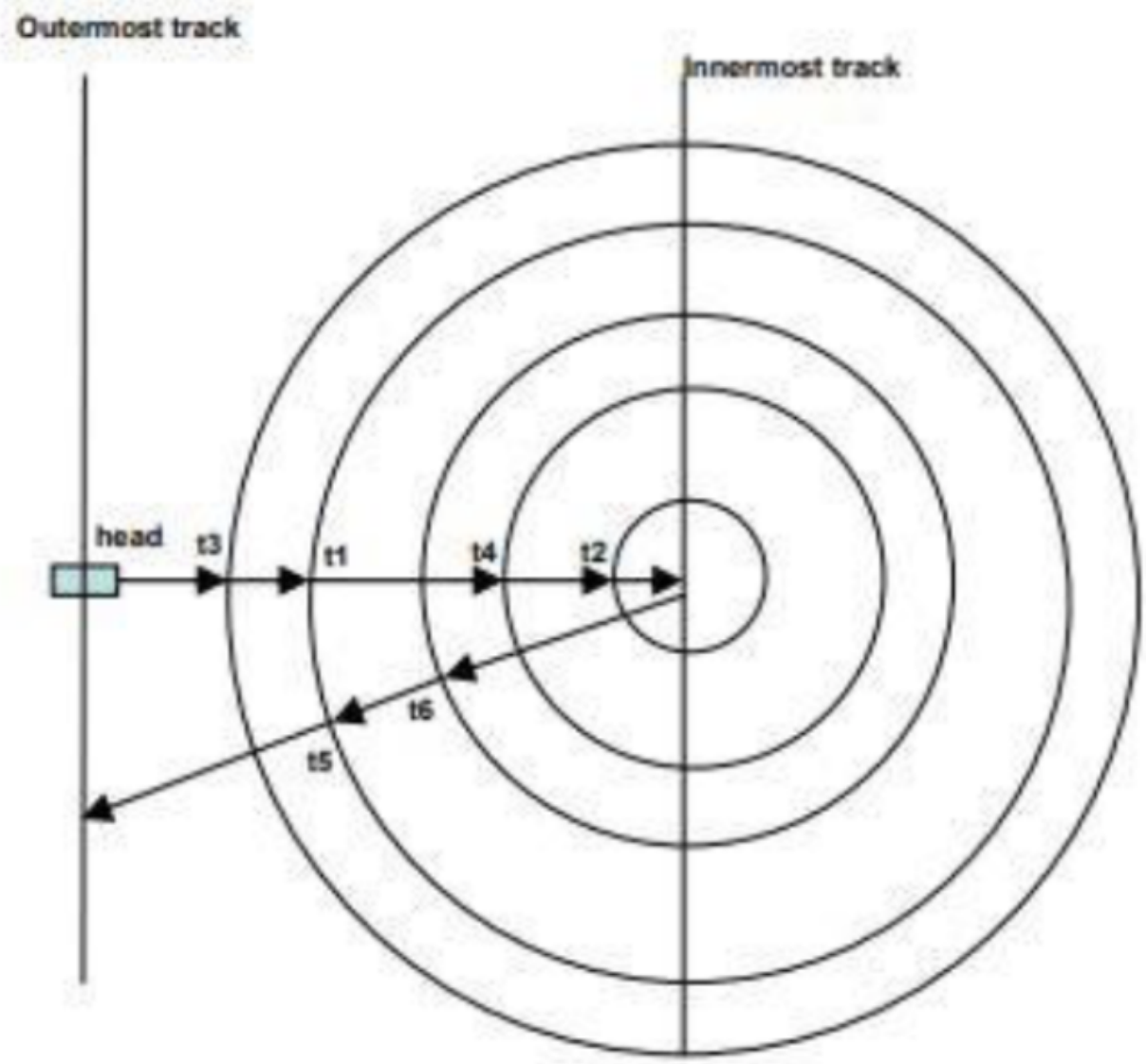
- Head moves from position of rest (outermost) to innermost track
- Handles requests en route regardless of when they appear
- Goes to outermost once it reaches the innermost (services requests on the way back)

Advantages

- Good throughput
- Good waiting time
- Limited starvation

Disadvantages

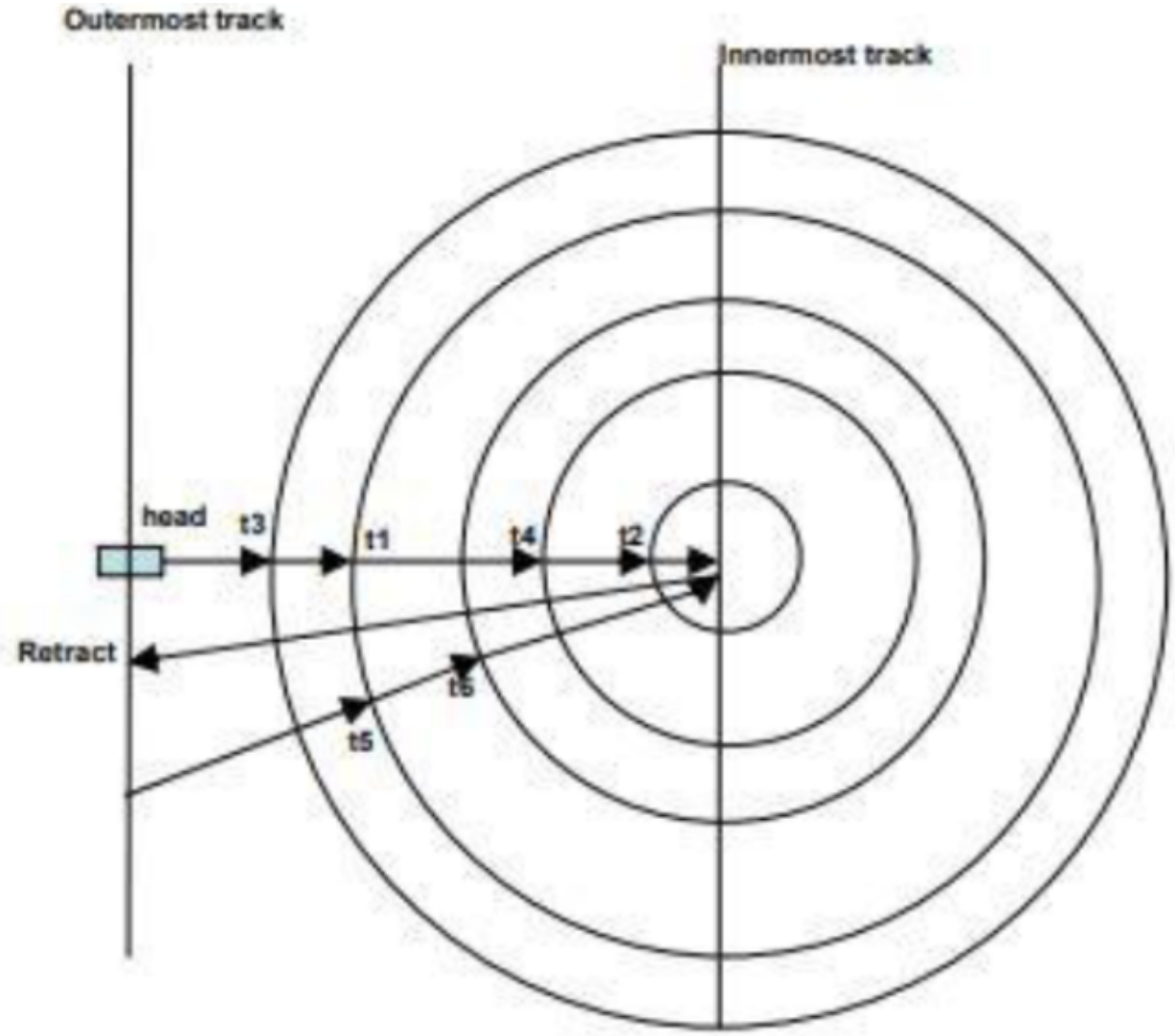
- Doesn't preserve memory access ordering



Circular SCAN (C-SCAN)

Same as SCAN but when it reaches innermost track it goes straight back to the outermost again (does NOT service requests on the return trip)

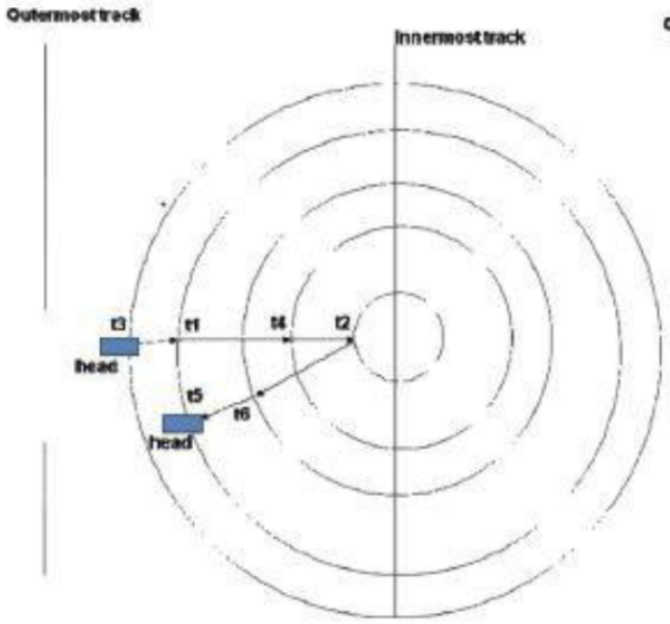
- Preserves order a bit more than SCAN
- Less variance in waiting time
- Same benefits as SCAN



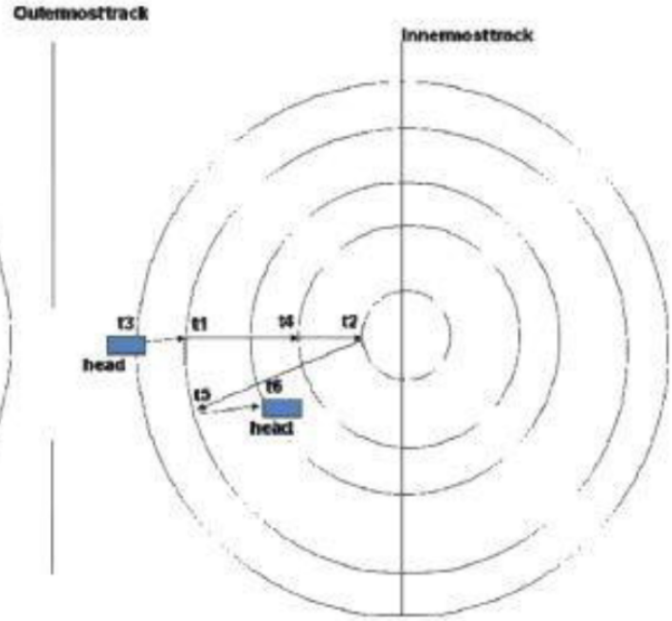
LOOK and C-LOOK

- Similar to SCAN and C-SCAN
- When there are no more requests to be served in a particular direction we reverse direction (no reason to go all the way back if there's no one to service there)

This is what is used on modern disks



(a) LOOK



(b) C-LOOK