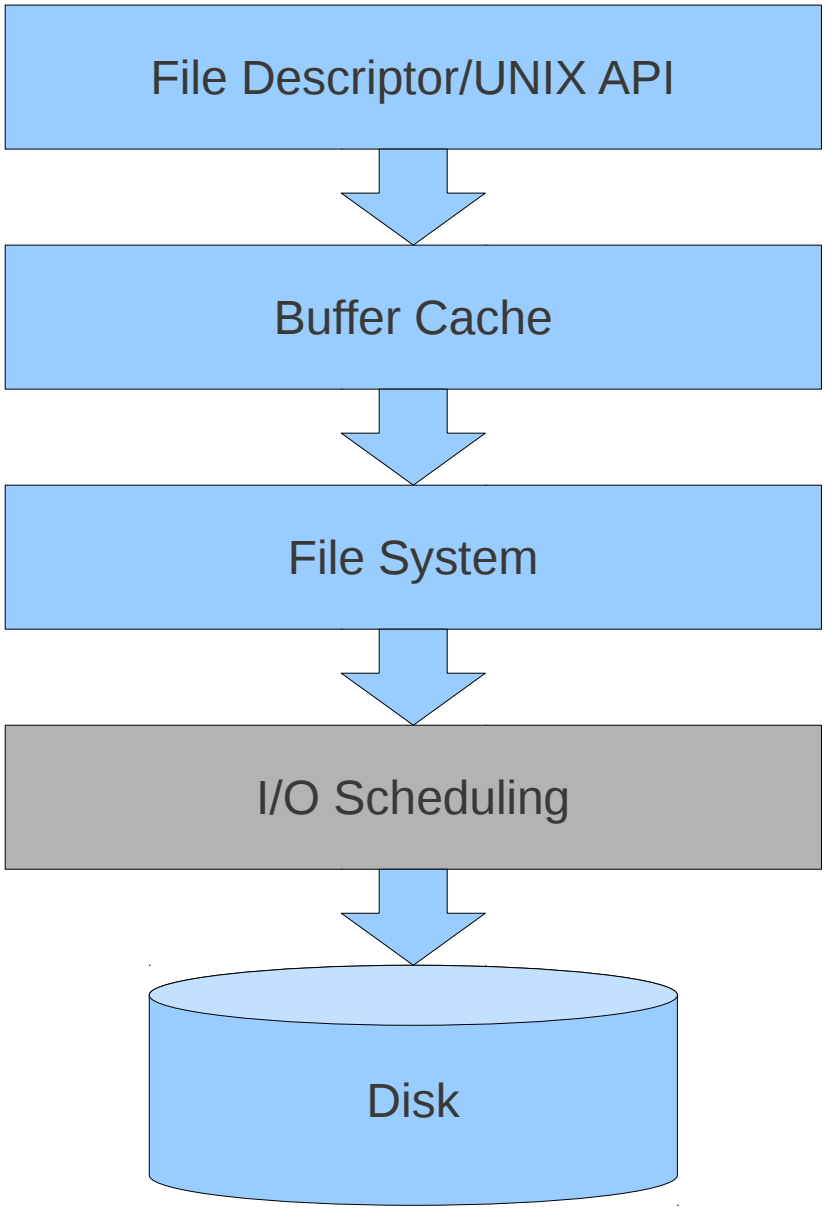
csci 3411: Operating Systems

Disk Scheduling

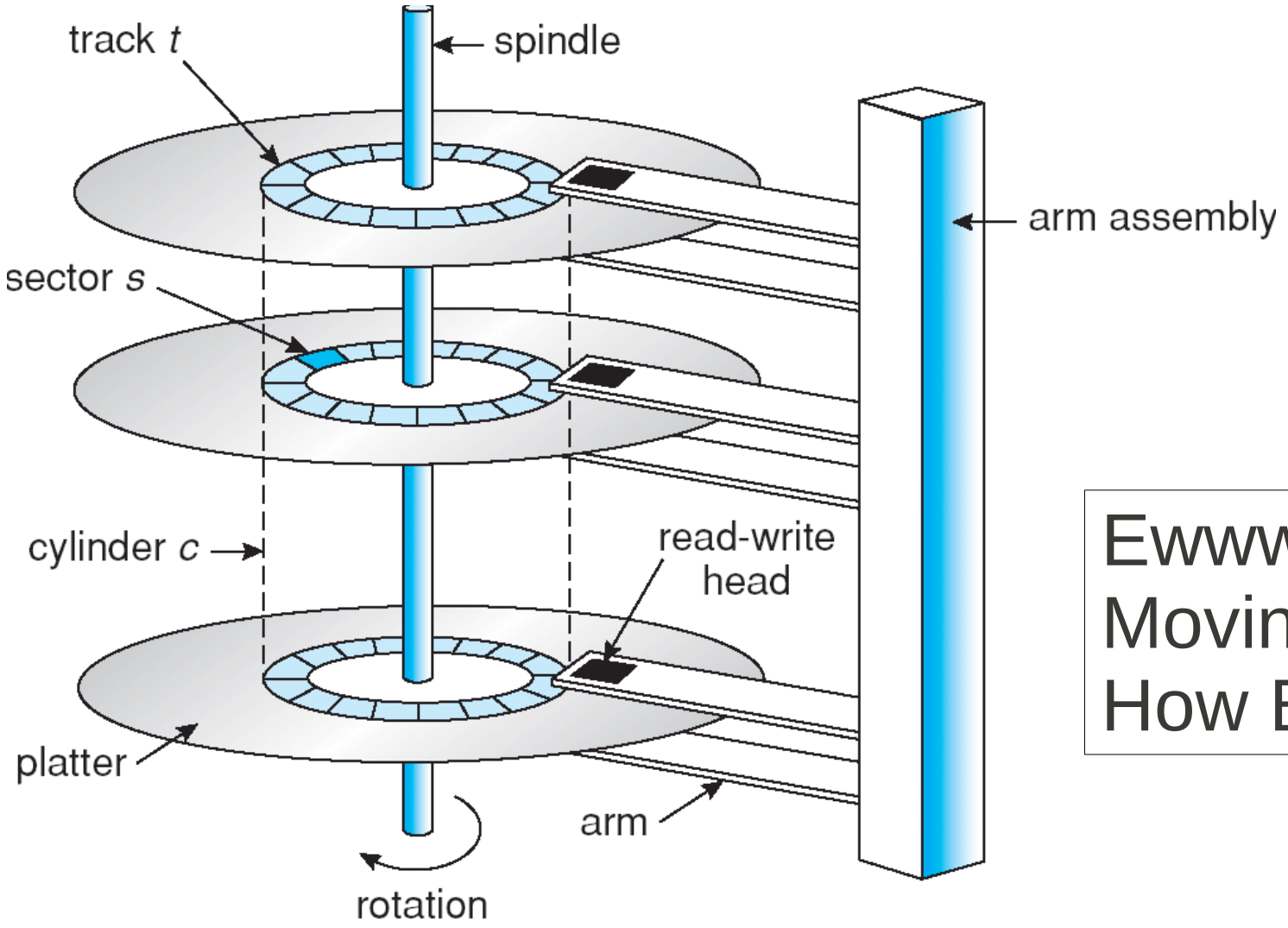
Gabriel Parmer

Slides evolved from Silberschatz

Layered Architecture



Magnetic Media (Disk)



Ewww, Yuck!
Moving Parts!
How Barbaric.

Magnetic Media II

- *Blocks* – granularity of I/O as seen by OS
  - Multiple of sector size
    - Block size == page size is convenient (i.e. 4KB)
- Accessed by OS as 1d array of blocks
- Disks can process I/O requests in *parallel*
  - Many I/O requests for blocks can be pending at a given time

Disk I/O Management

- Disks controllers typically interact with the CPU
  - CPU can request block transfers to/from memory using DMA
  - Interrupts triggered when blocks arrive in memory, or when they are copied to disk
  - Familiar?
  - Why didn't Linux until very recently include an equivalent to NAPI for disks?

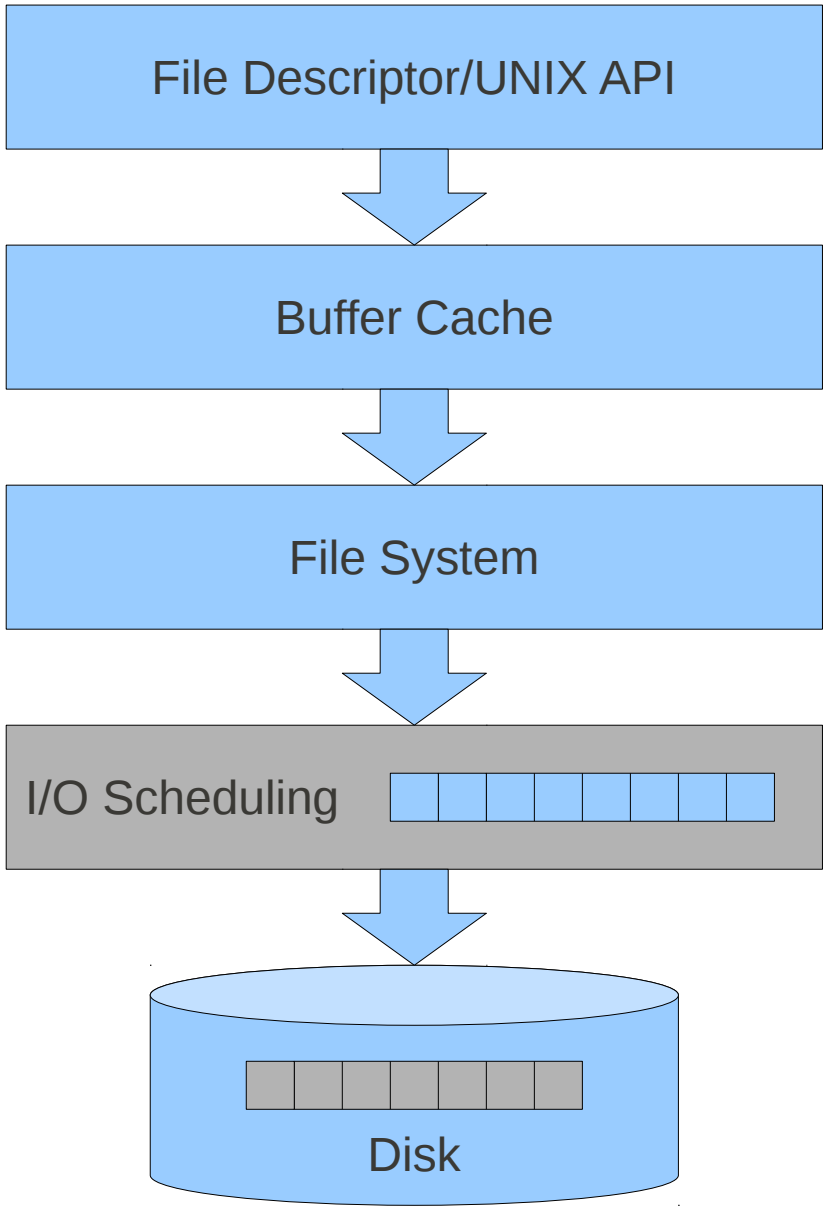
Disk Scheduling

- Oses goal is to maximize disk's bandwidth, minimize latency
- OS encourages concurrency
  - Multiple threads each making disk requests
  - Each filesystem request can cause multiple disk requests
    - Multiple index blocks (ls), multiple data blocks for sequential access (cat)

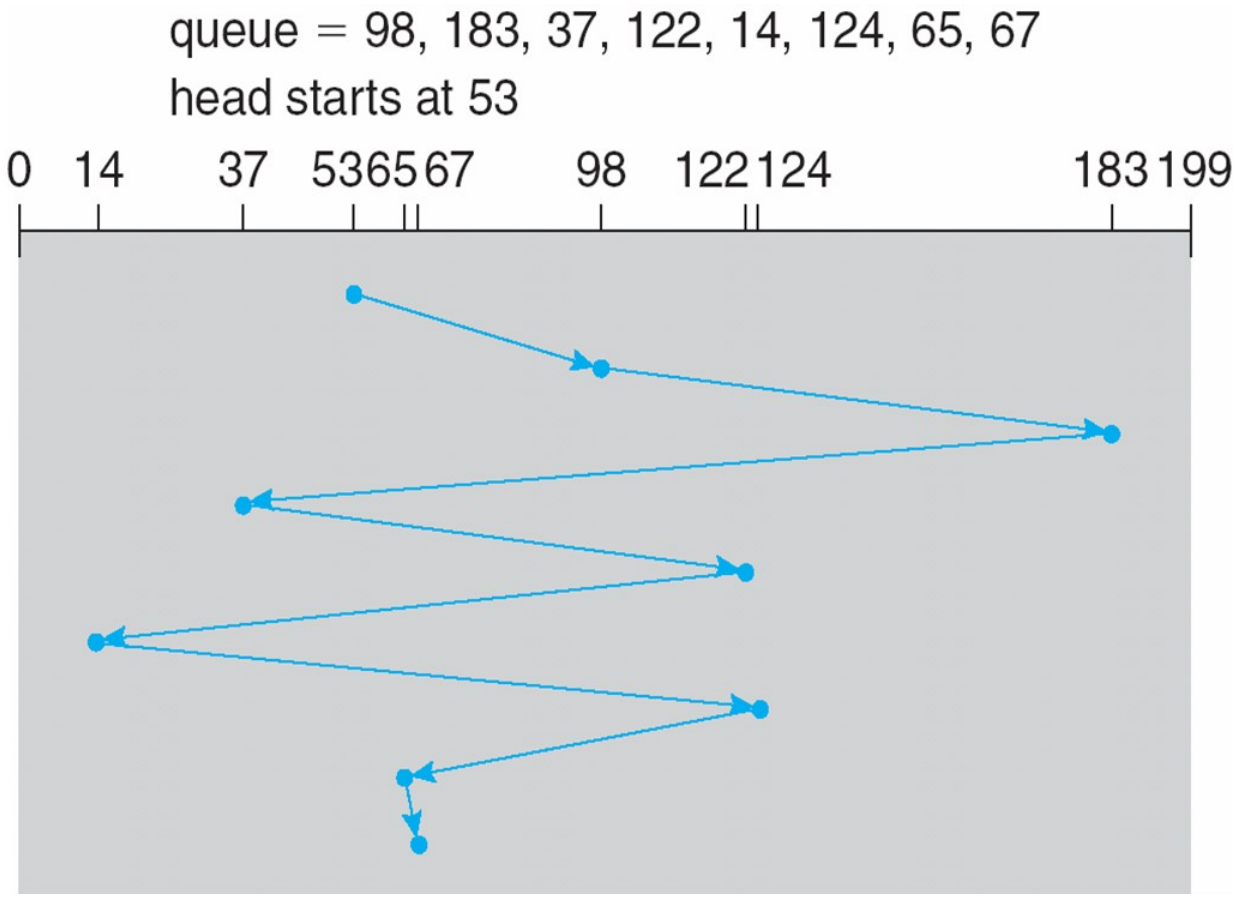
Disk Scheduling II

- *In what order should the OS issue these concurrent disk requests to disk?*
  - Consider disk physical characteristics (seek time)
  - Elevator analogy
- Goals
  - Performance (bandwidth/latency)
  - Fairness (between different processes making reqs)
    - Starvation
  - Performance/Fairness often at odds

I/O Buffer Request Queues

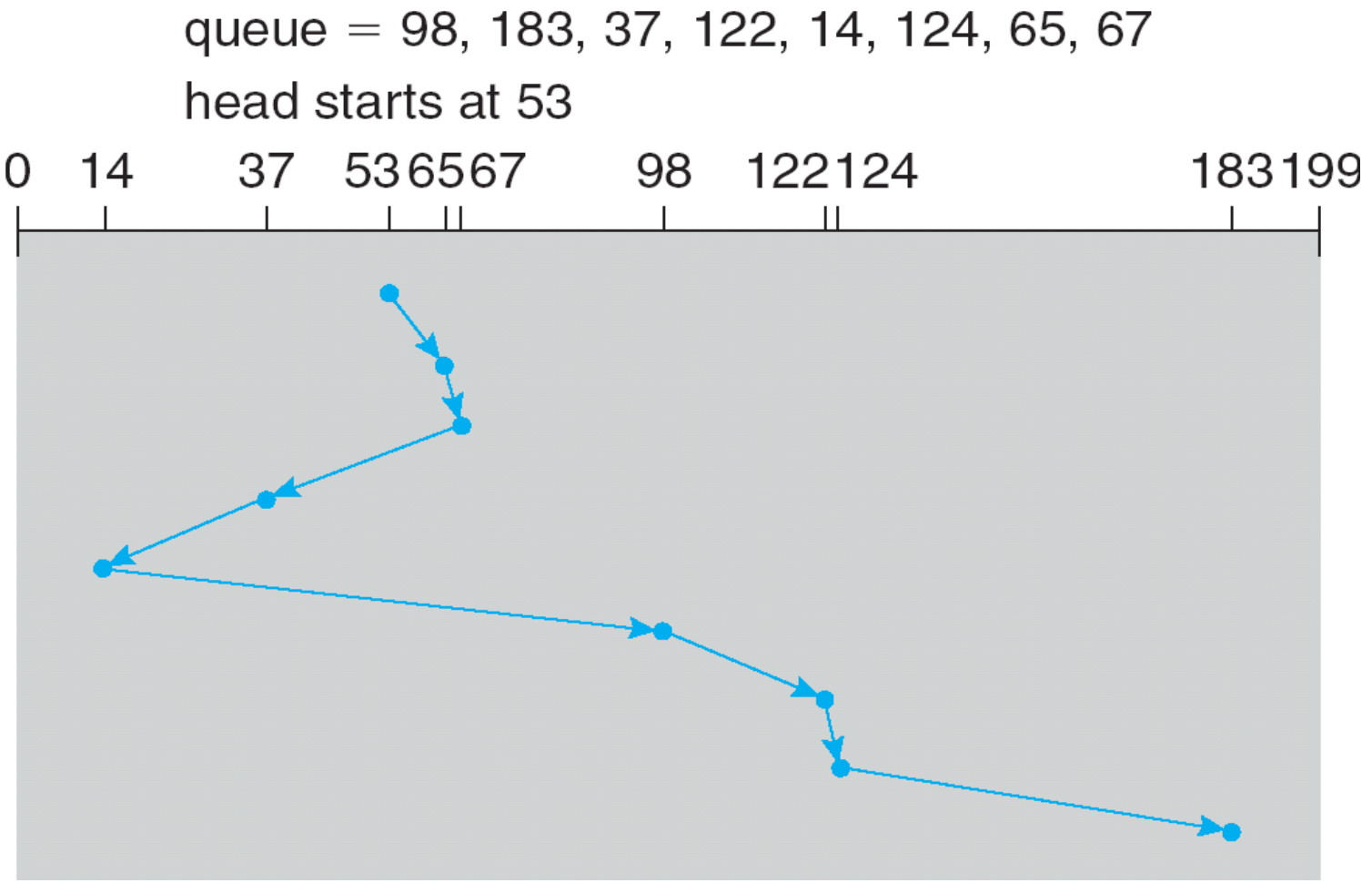


First-Come, First Served



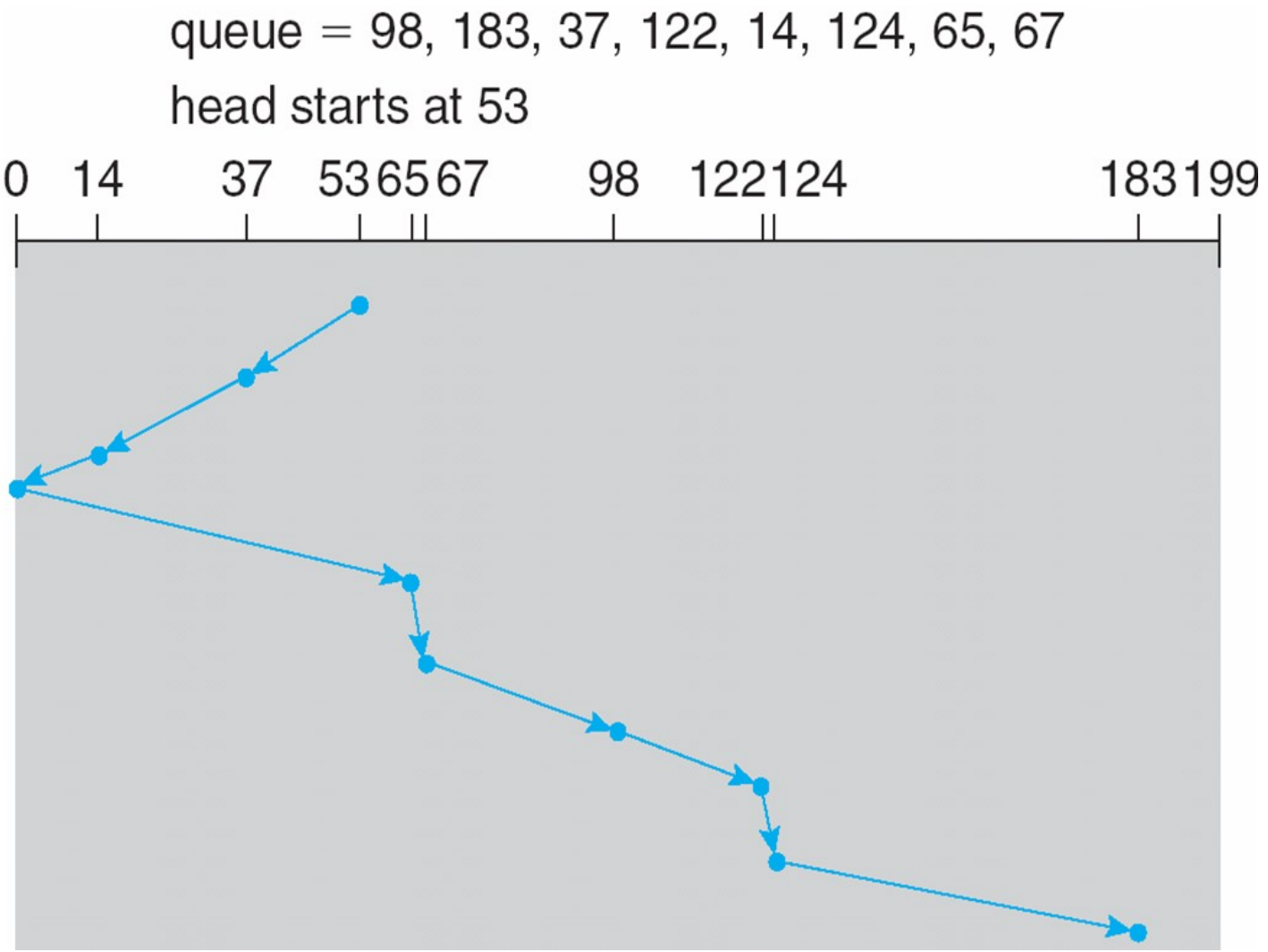
- Head movement: 640 cylinders

Shorted Seek Time First (SSTF)



- Head moves across 236 cylinders

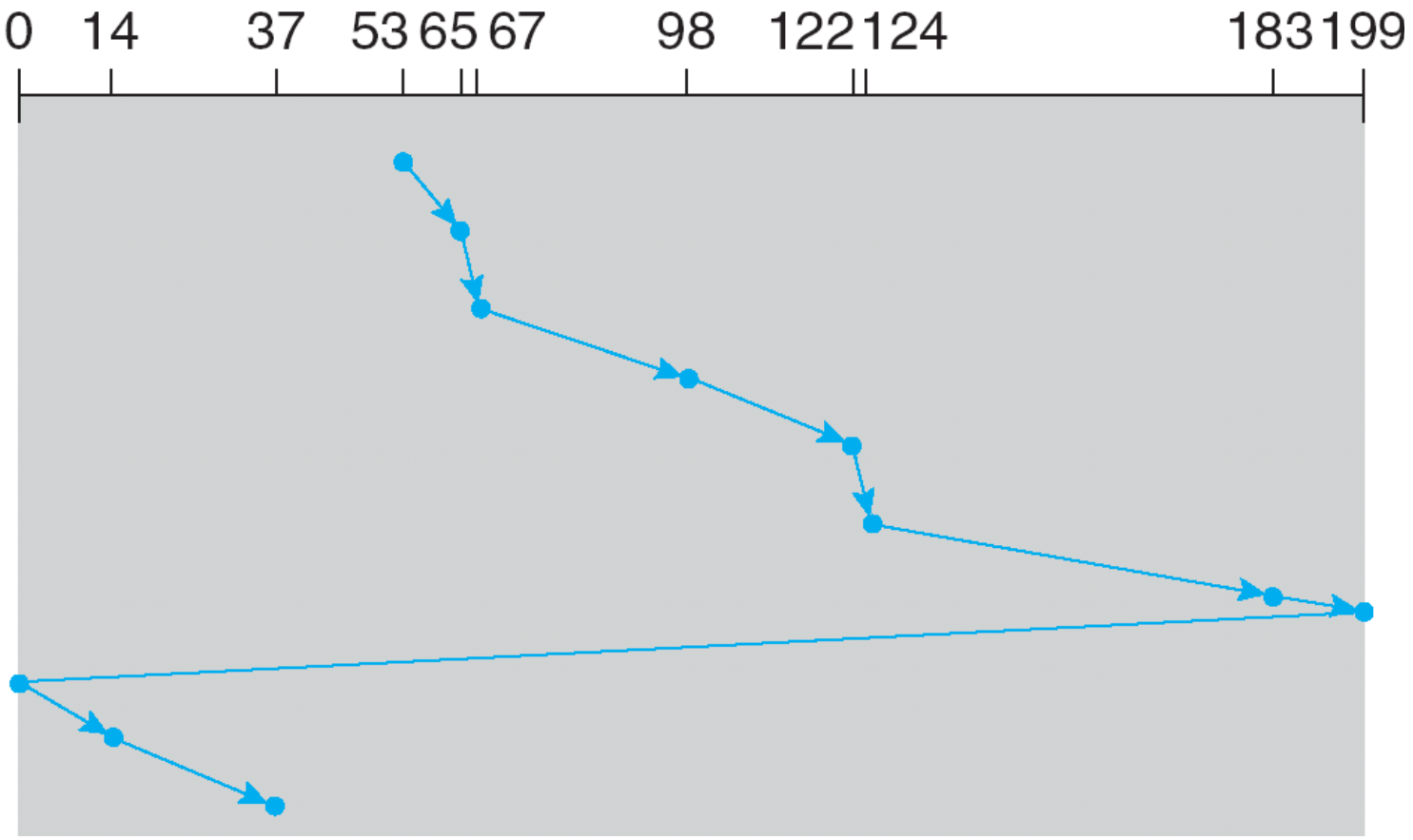
Scan (Elevator) Scheduling



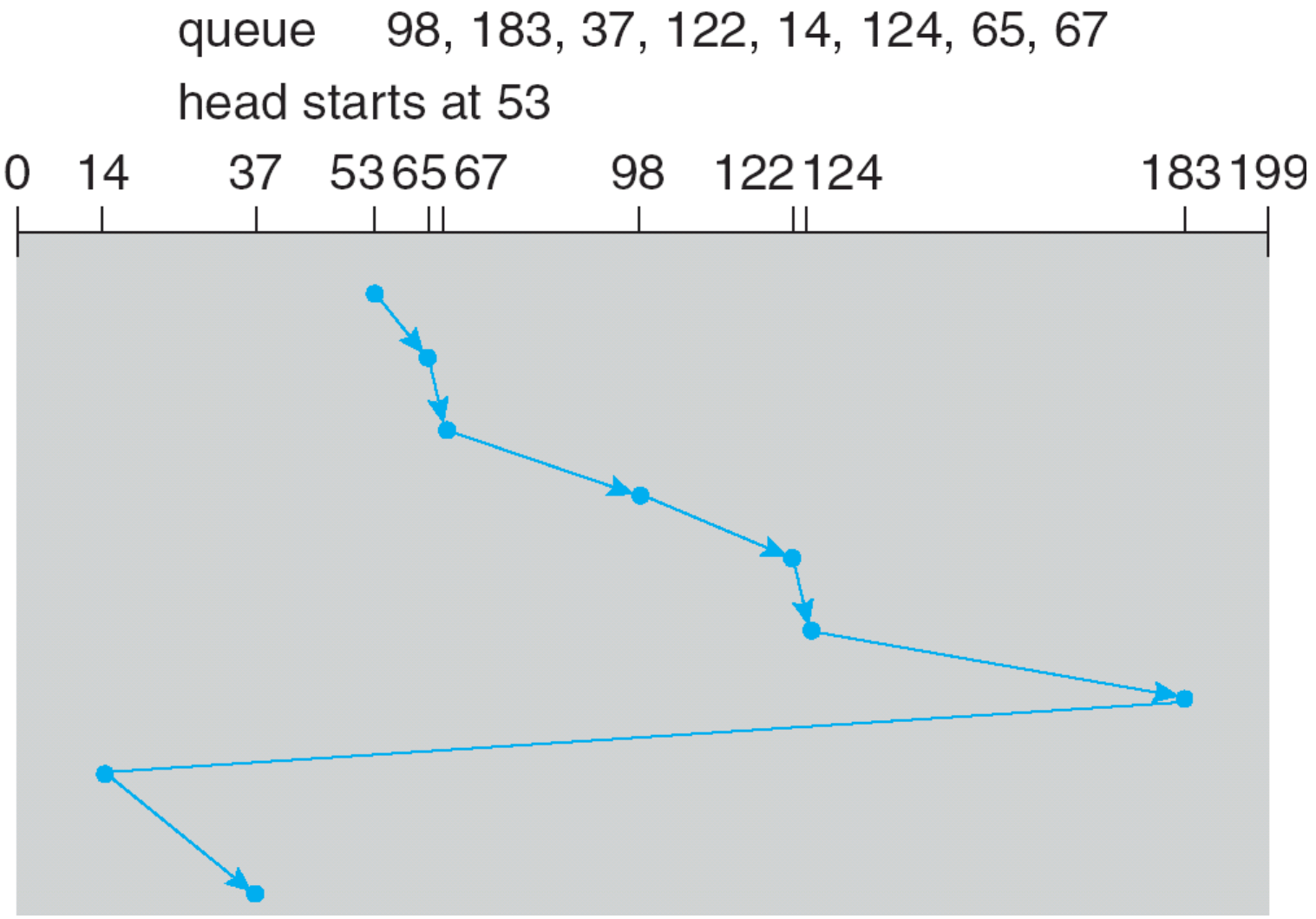
C-Scan (circular scan)

queue = 98, 183, 37, 122, 14, 124, 65, 67

head starts at 53



C-Look Scheduling



Additional Techniques

- Which algorithm is best when there is one outstanding request?
  - Better use disk when there are multiple outstanding requests
    - Motivates *Anticipatory* scheduling
    - A generally beneficial technique?

So wait, which should I use?

- Which is *best*?
  - Throughput
  - Latency
  - Fairness
- Expected workload?
  - Low Throughput?
  - High throughput?
    - Sustained or bursty?
  - Interactive?

- Which do I use???
  - Too hard to decide,
  - I give up
- Recurring theme
  - No one algorithm best for all situations
  - What can do we???

Disks *are* barbaric, what about SSDs

- *Solid State Disks* (aka Flash drives)
  - Organized like memory: Random access array
    - Random vs. Sequential access
    - Implications on disk scheduling algorithms?
  - Reads lightening fast!
  - Writes not so much, but still better than disks!
    - Generally only sequential write is slower than disks
- OMGFTWBBQ – filesystems can be stupid now!
  - right?

[1]

[1] http://arstechnica.com/hardware/news/2009/11/biography-solid-state-disk.ars

Solid State Disks: Different Pain

- What's the same
  - I/O still relatively very expensive: Minimize
    - Buffer cache
    - Indexes should minimize number of accesses
- Different medium, different constraints
  - Write once, *erase* operation on granularity of many blocks (512K erases vs 4K blocks)
  - Limited number of erases per cell – Requires write leveling
- Minimize writes, cluster data changed often
- More info: http://www.anandtech.com/printarticle.aspx?i=3531

Phase-Change Memory (PCM)

- PCM and beyond...
- Trend towards faster random access nonvolatile storage
- Will speeds meet that of RAM?
  - Where will the filesystem be in such a case?
  - Most likely, speeds will lag behind that of RAM