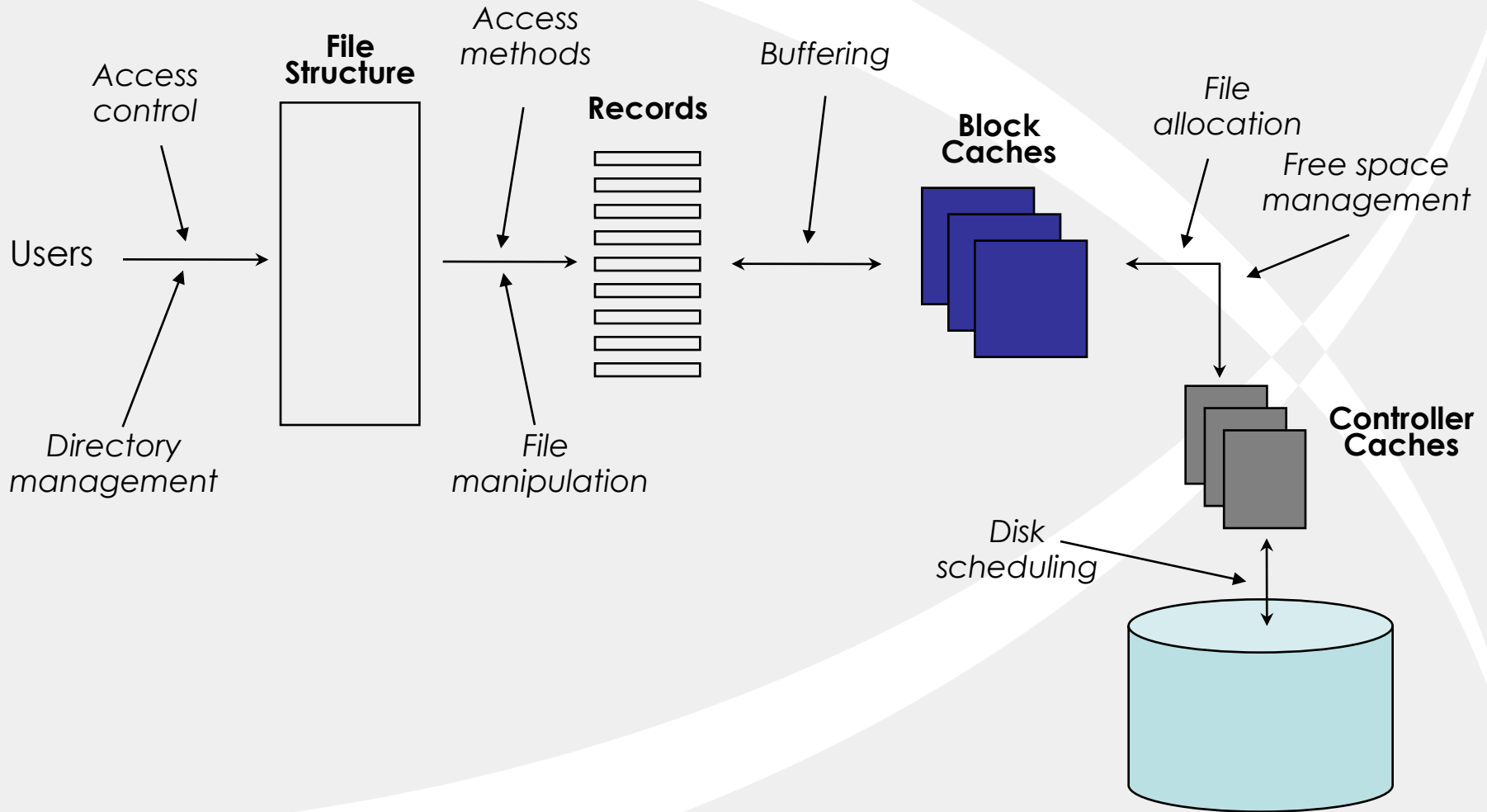
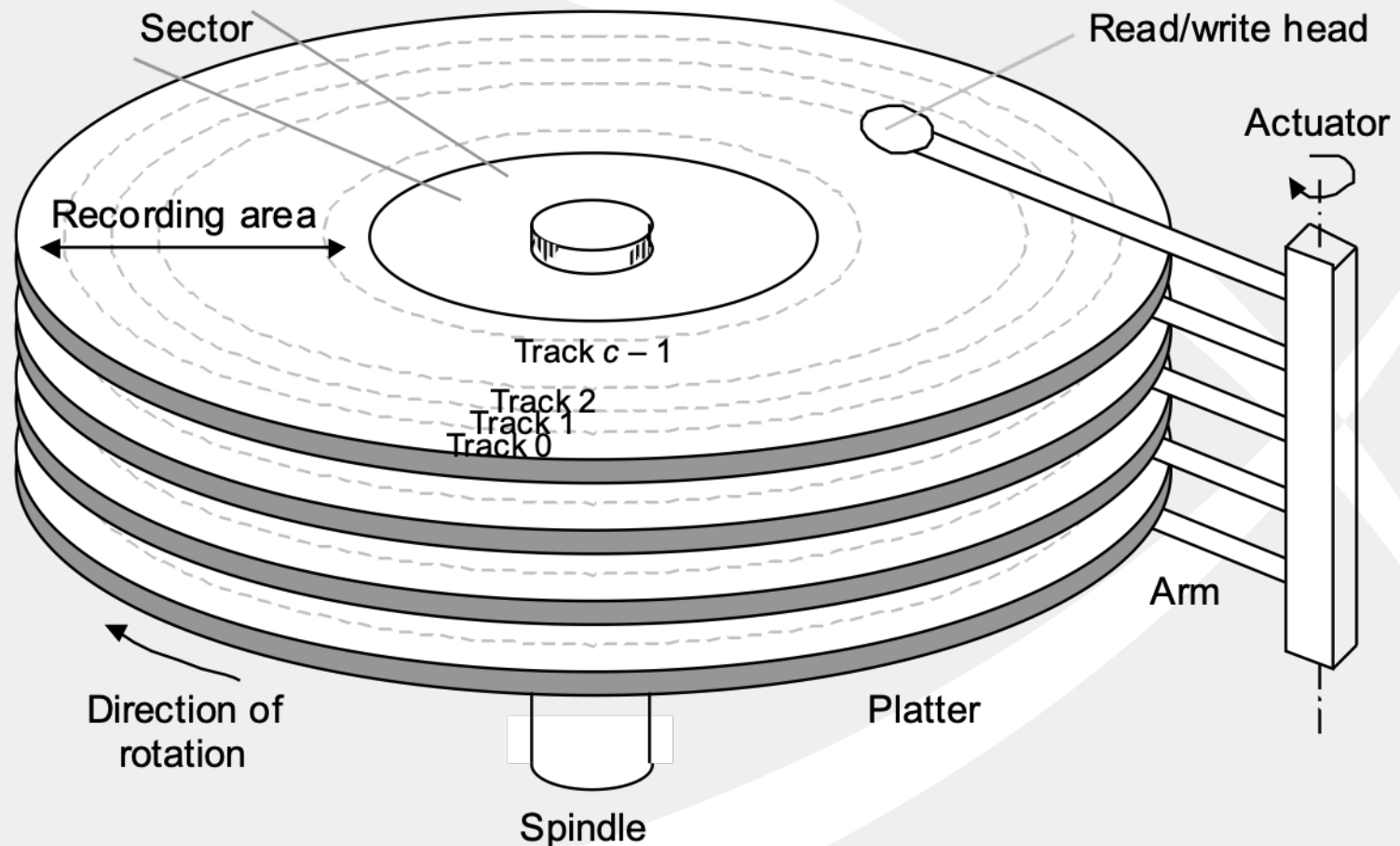


Secondary Storage System

Elements of storage a system

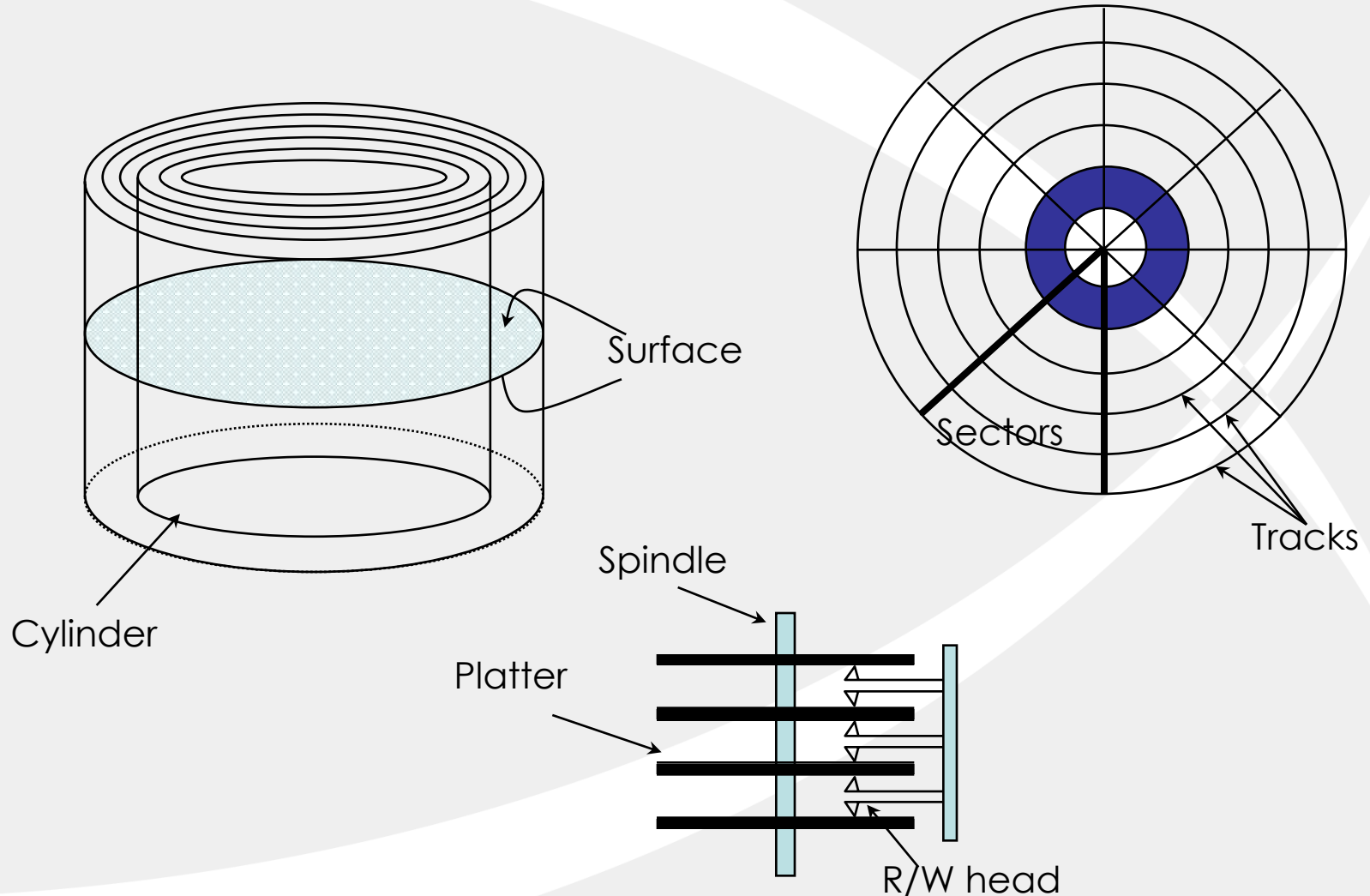


Disk Memory Basics

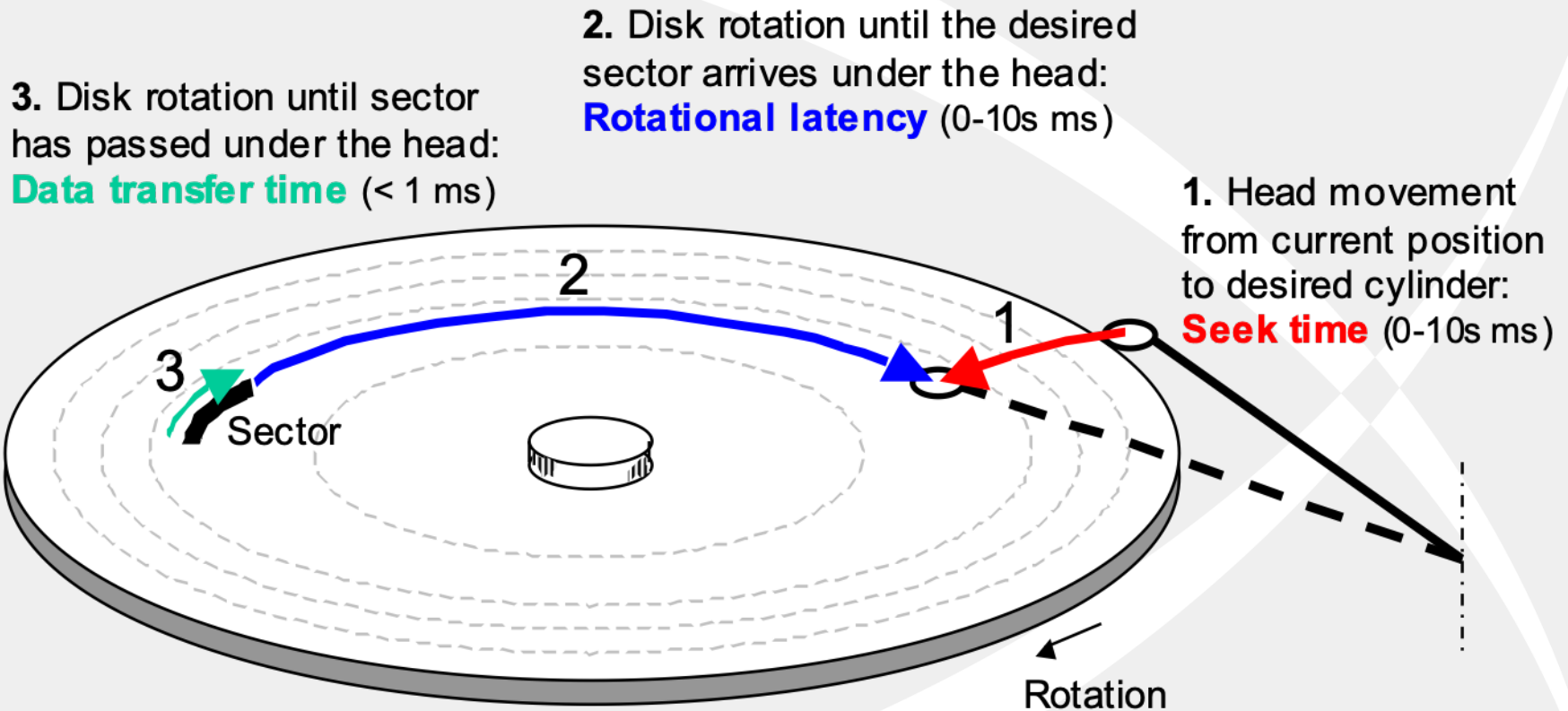


Disk memory elements and key terms.

Disk structure (another view)



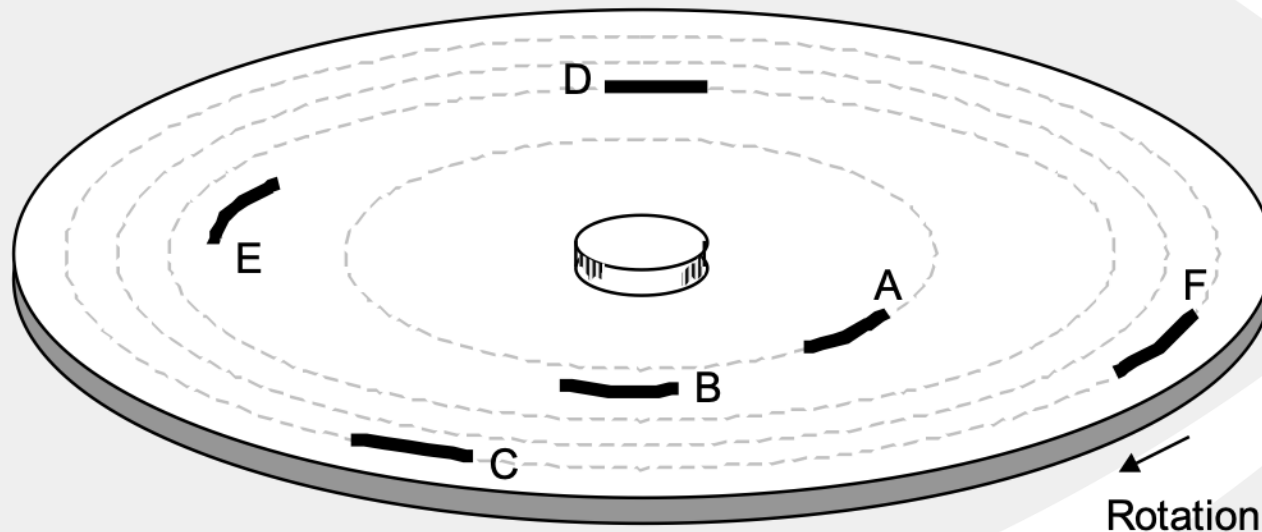
Access Time for a Disk



The three components of disk access time. Disks that spin faster have a shorter average and worst-case access time.

Disk Performance

Average rotational latency = $30 / \text{rpm} \text{ s} = 30\,000 / \text{rpm} \text{ ms}$



Arrival order of
access requests:

A, B, C, D, E, F

Possible out-of-
order reading:

C, F, D, E, B, A

Reducing average seek time and rotational latency by performing disk accesses out of order.

Disk Caching

Same idea as processor cache: bridge main-disk speed gap

Read/write an entire track with each disk access:

“Access one sector, get 100s free,” hit rate around 90%

Disks listed in above table have buffers from 1/8 to 16 MB

Rotational latency eliminated; can start from any sector

Need back-up power so as not to lose changes in disk cache

Placement options for disk cache

In the disk controller:

Suffers from bus and controller latencies even for a cache hit

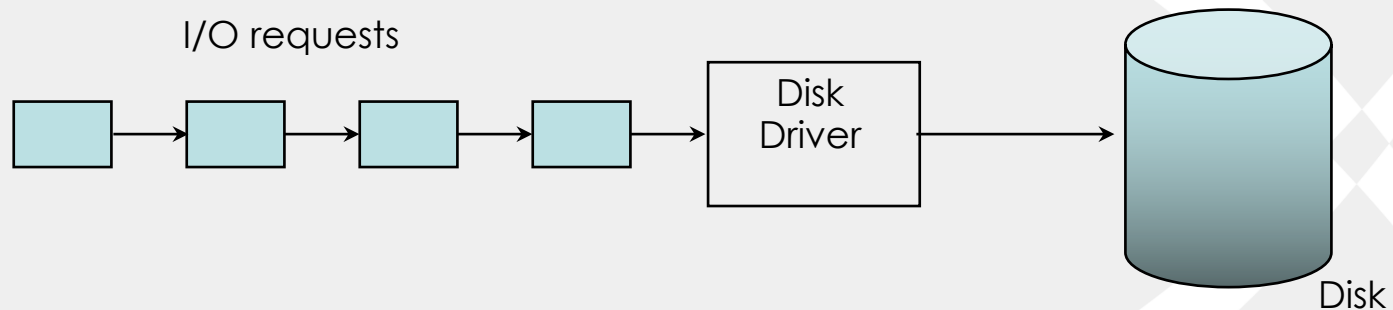
Closer to the CPU:

Avoids latencies and allows for better utilization of space

Intermediate or multilevel solutions

Disk scheduling

- Scheduling problem: maximize the throughput with concurrent I/O requests from



- Strategy: minimize the time spent by disk seeking for data – maximizes the time spent reading/writing data

Disk scheduling strategies

■ Commonly used strategies include:

◆ *Random*

- select from pool randomly
- worst performer
- sometimes useful as a benchmark for analysis and simulation

◆ *First Come First Served (FCFS) or FIFO*

- fairest of them all; no starvation; requests are honored in the order they are received
- works well for few processes (principle of locality)
- approaches to *random* as number of processes competing for the given disk increases

Disk scheduling strategies

◆ *Priority*

- access to the disk is not actually controlled by the disk management software
- based on processes' execution priority
- designed to meet job throughput criteria and *not* to optimize the disk usage

◆ *Last In First Out (LIFO)*

- service the most recently arriving request first
- can be useful for processing sequential files
- real danger of starvation; once a process falls behind it can be serviced only if the entire list ahead of it empties

Disk scheduling strategies

- Above algorithms have based disk scheduling decision on the requestor process
- Following algorithms use requested item, i.e. the requested disk address
 - ◆ *Shortest Service Time First (SSTF)*
 - select the item requiring the shortest seek time
 - no guarantee of improved average seek time in any particular circumstance but
 - in general better average seek time than FIFO
 - random tie breaker used, if needed, to decide in which direction to move

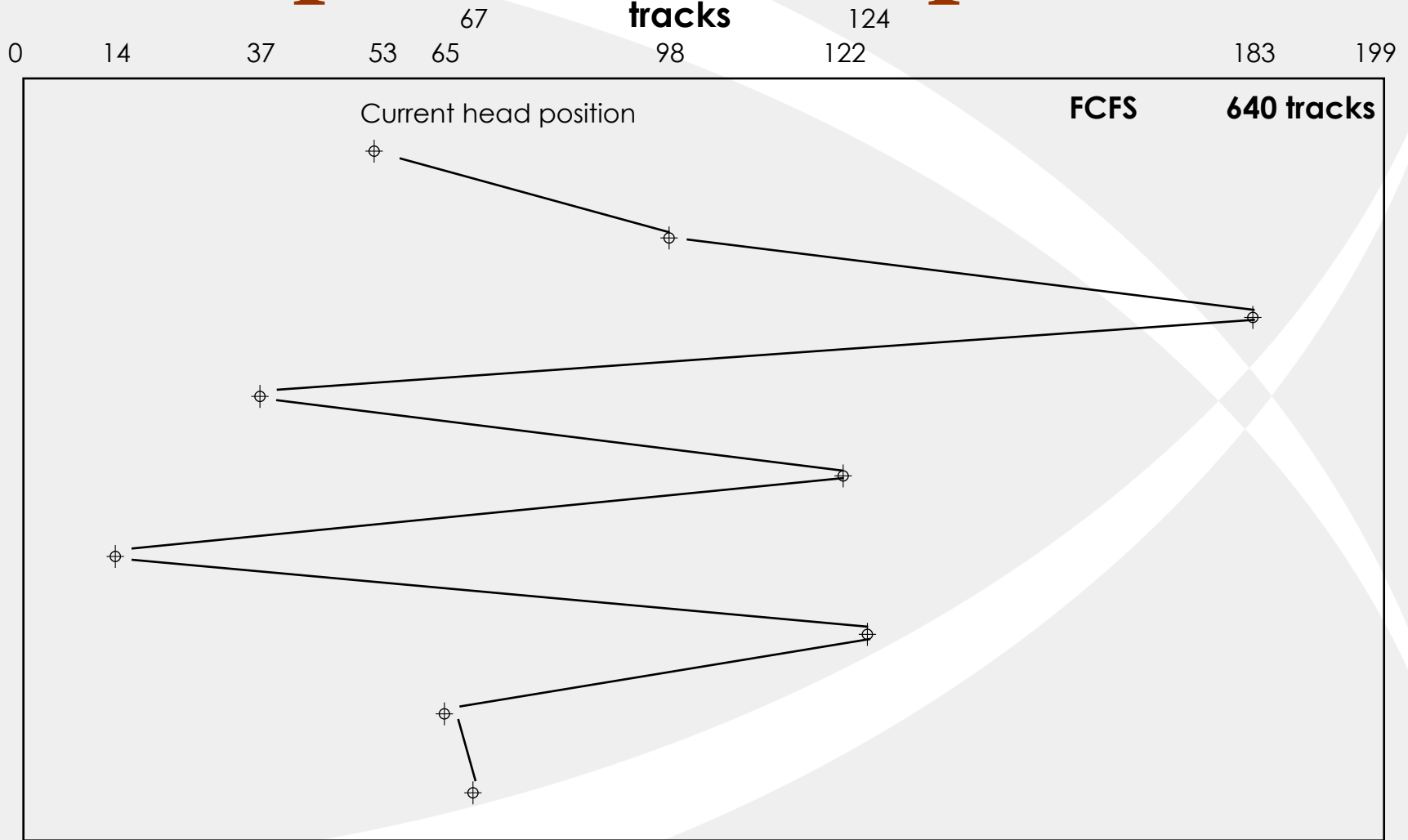
Disk scheduling strategies

- ◆ **SCAN**—back and forth over disk
 - heads move in only one direction until the last track is reached, then the direction is reversed
 - if the direction of the heads' travel is reversed once there are no more requests in that direction this method is called *LOOK*
 - no danger of starvation, but
 - biased **against** the most recently used area on the disk
 - Doesn't exploit locality as well as SSTF though or as LIFO
 - often similar to *SSTF*
- ◆ **C-SCAN**—circular SCAN or one way SCAN and fast return
 - scans in only one direction to the last track and then returns heads quickly to the beginning
 - reduces the maximum delay for new arrivals

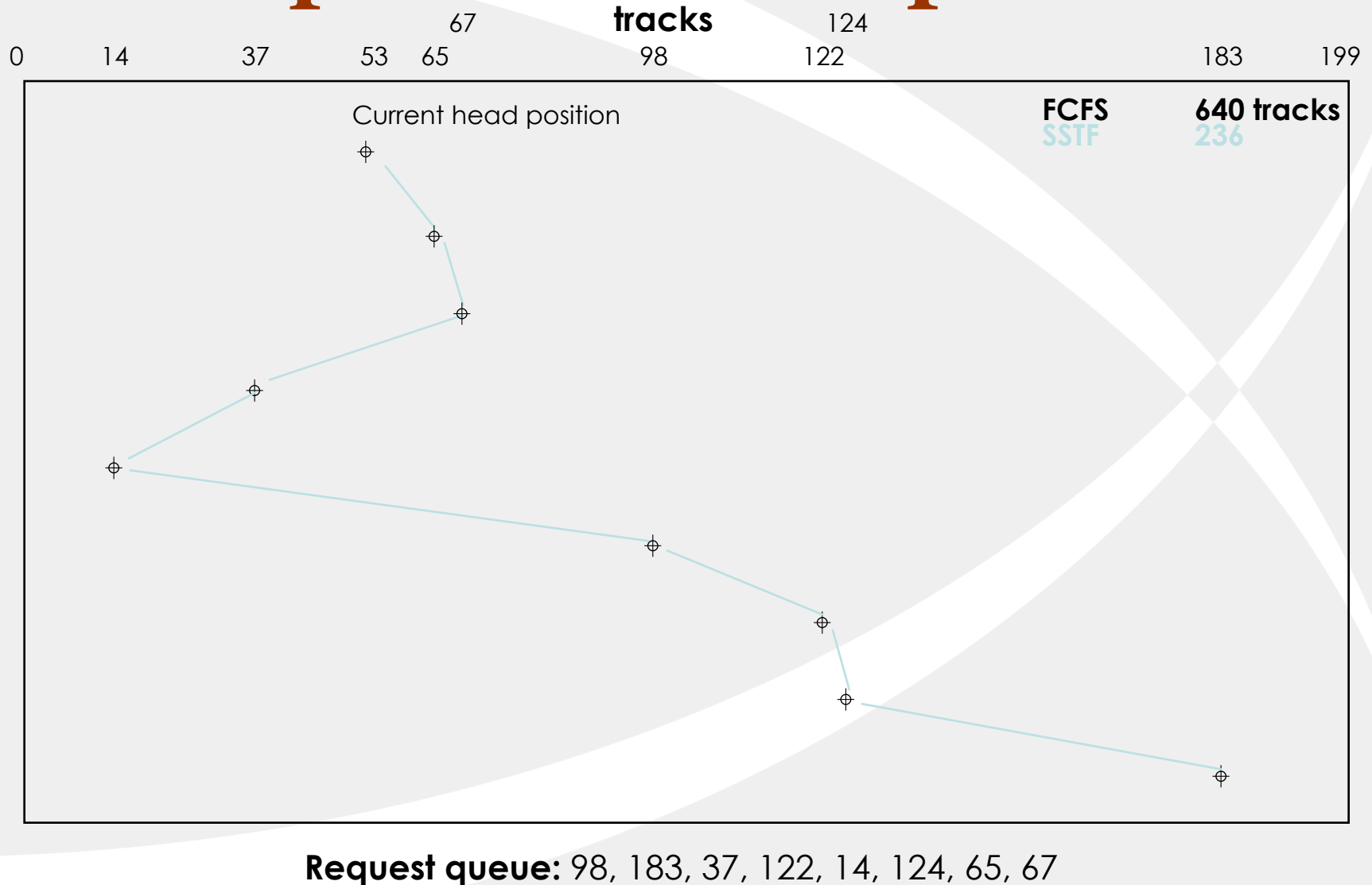
Disk scheduling strategies

- ◆ *LOOK*—look for a request before moving in that direction
 - Heads move in only one direction until there are no more requests in that direction, then the direction is reversed.
- ◆ *C-LOOK*—circular LOOK
 - Scans in only one direction until there are no more requests in that direction and then returns heads quickly to the beginning

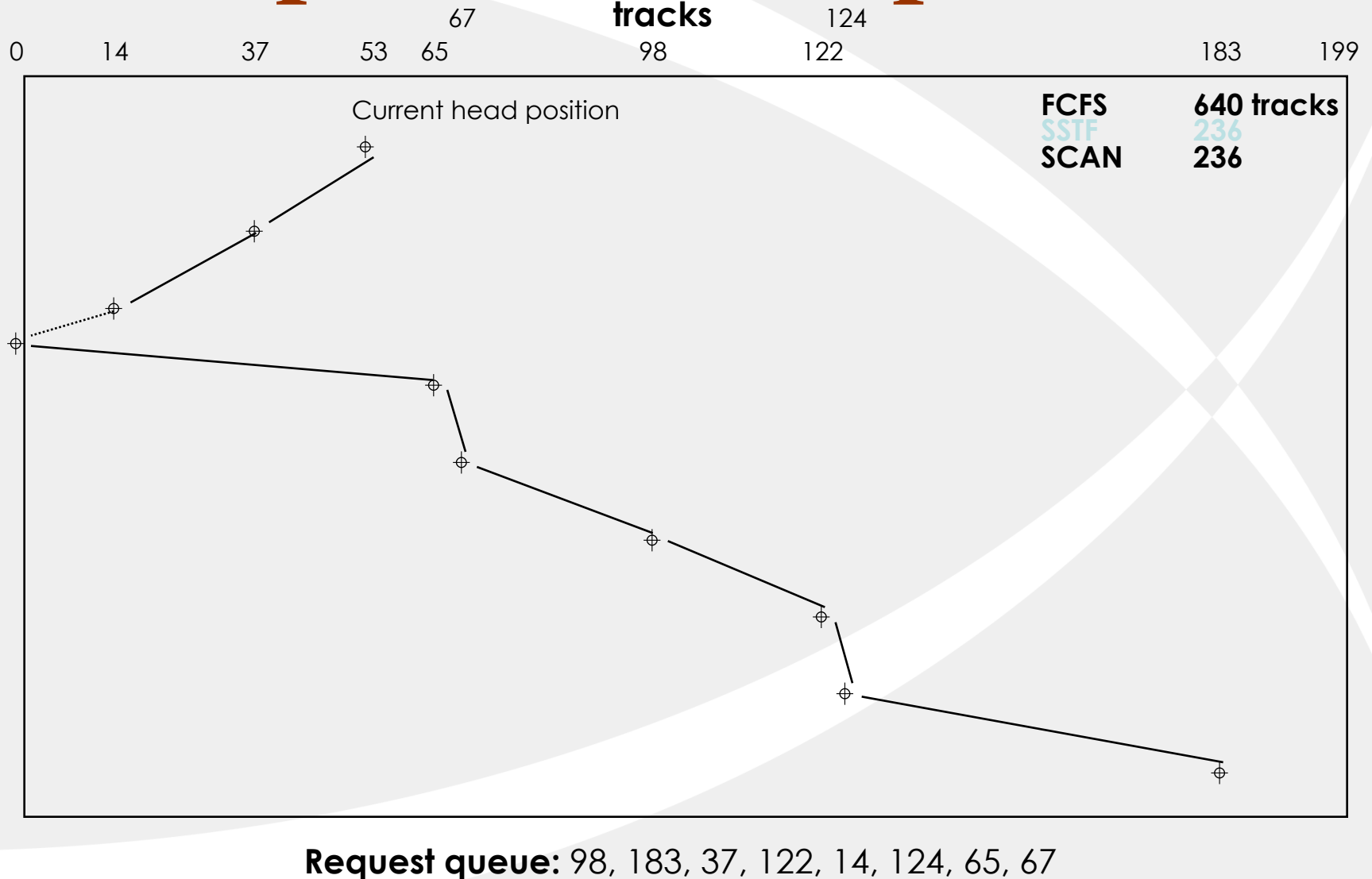
A comparative example—FCFS



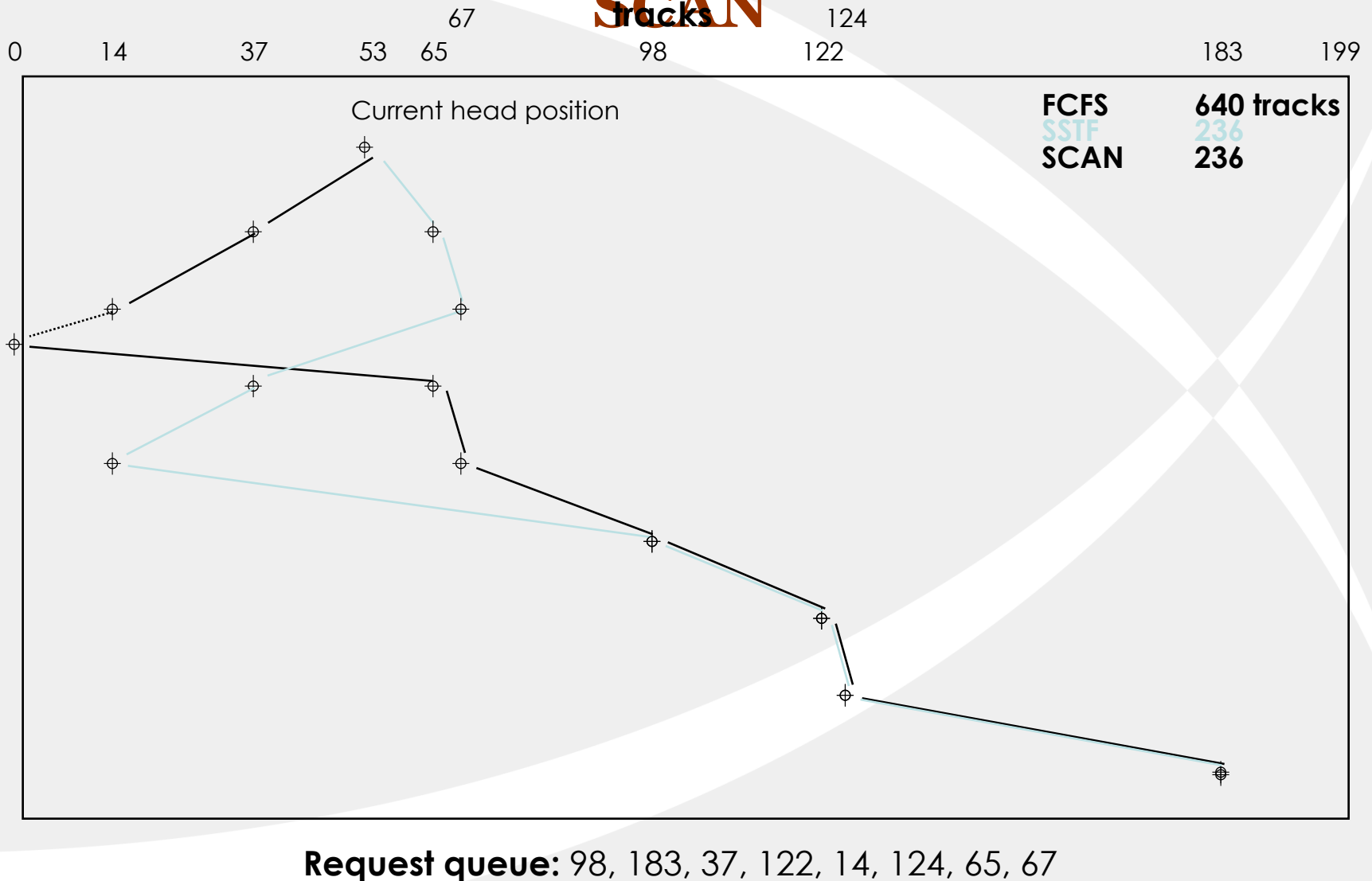
A comparative example—SSTF



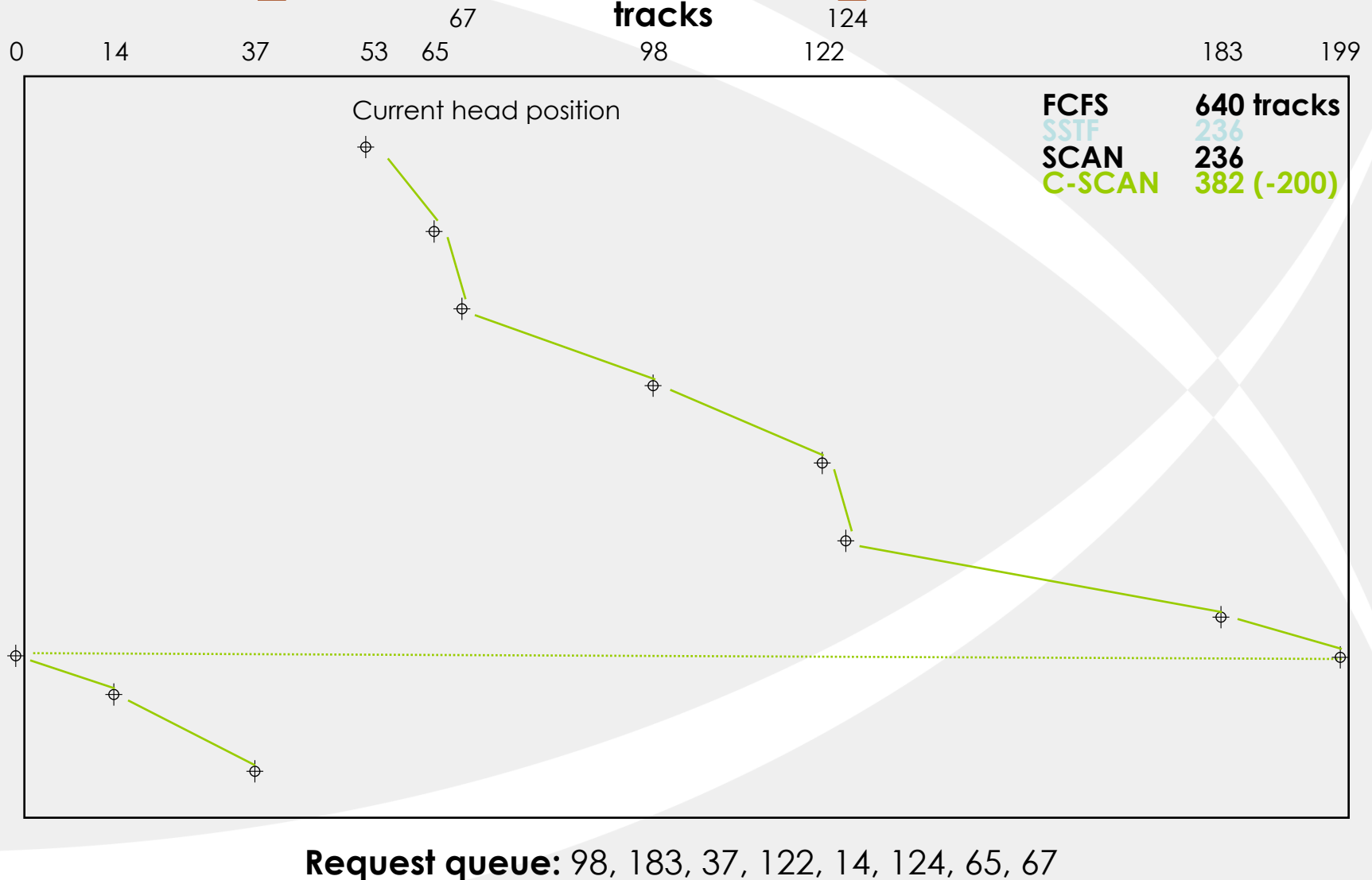
A comparative example—SCAN



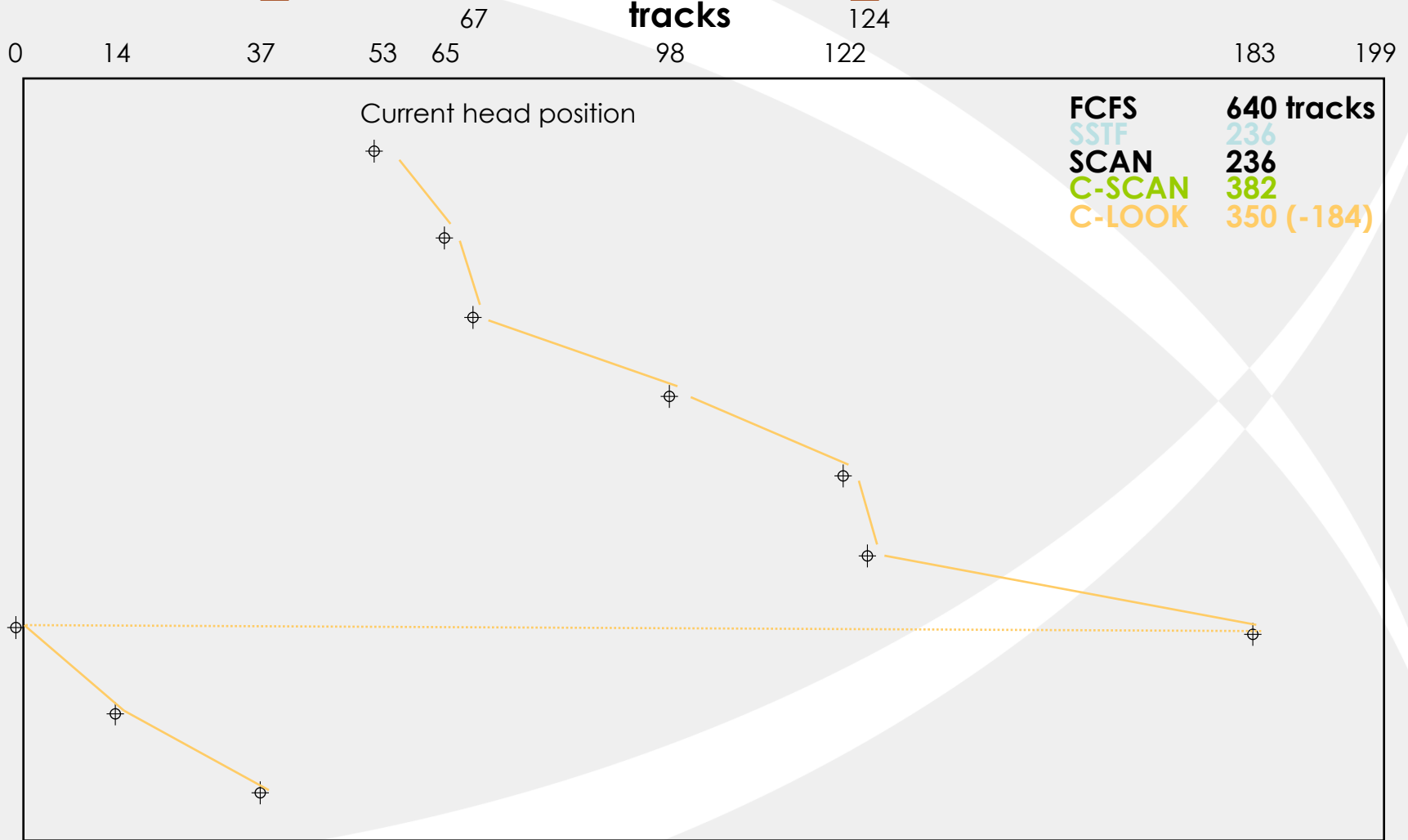
A comparative example—SSTF vs SCAN



A comparative example—C-SCAN

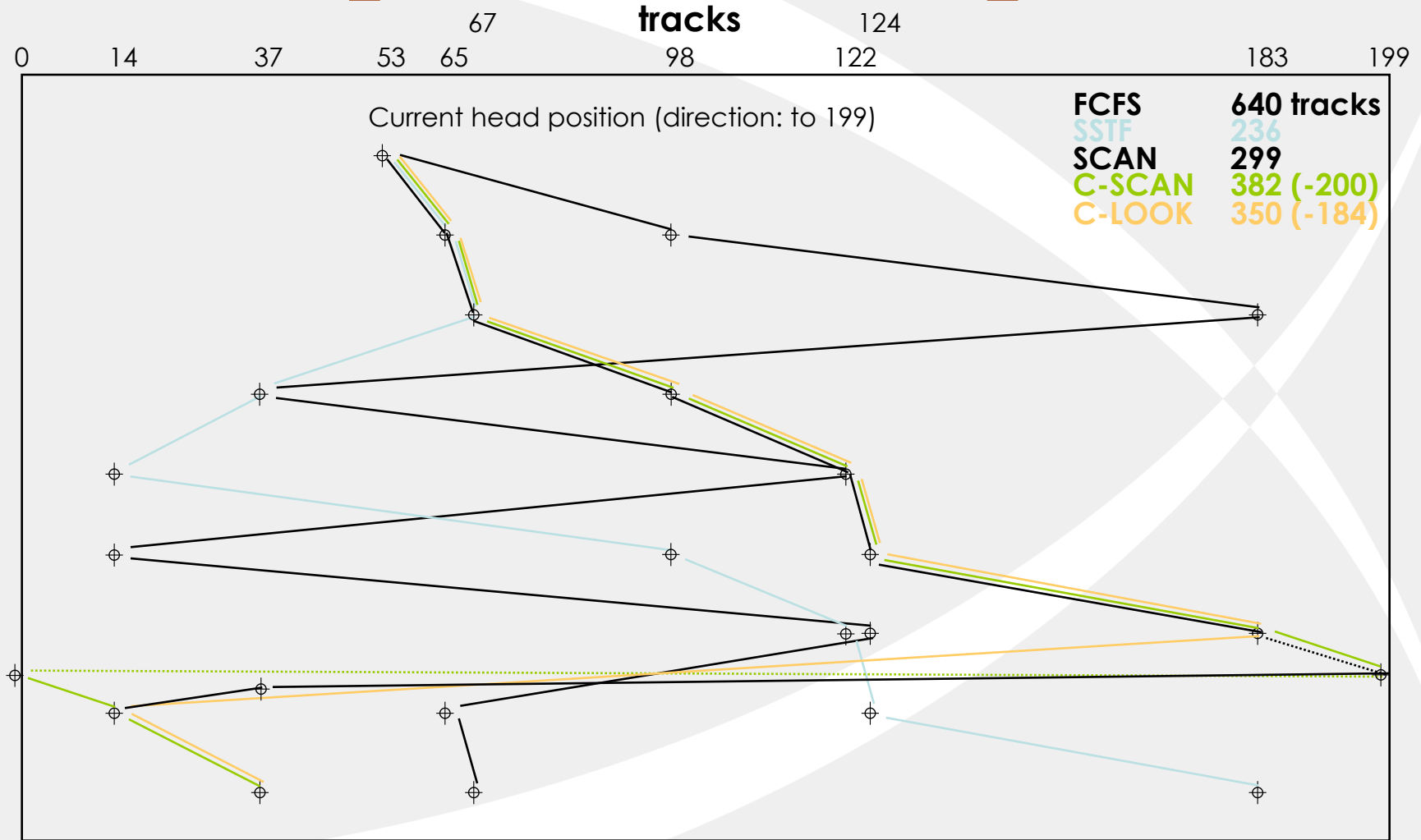


A comparative example—C-LOOK

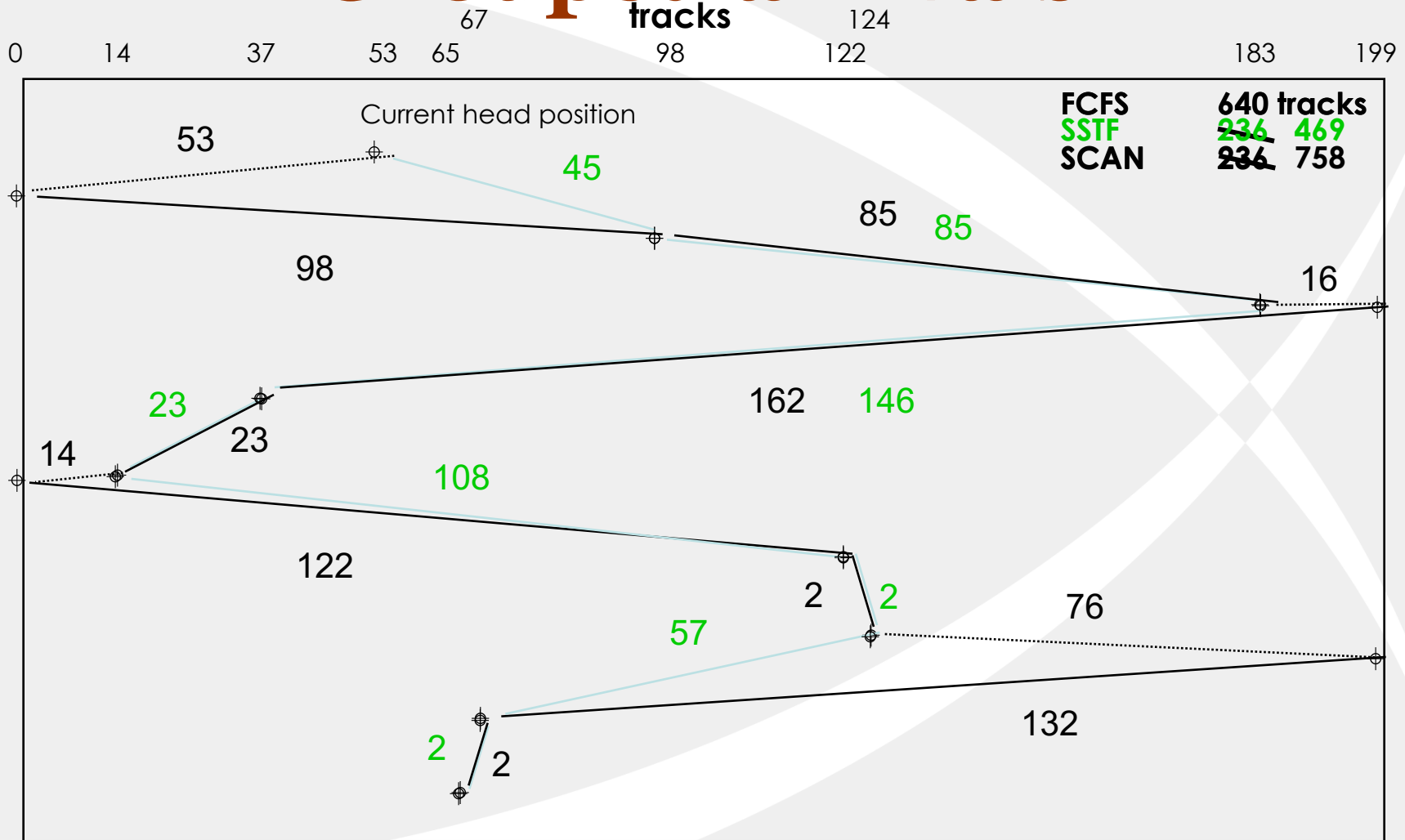


Request queue: 98, 183, 37, 122, 14, 124, 65, 67

A comparative example—All



Grouped arrivals



Request queue: 98, 183, 37, 122, 14, 124, 65, 67

Arrivals: 98, 183; 37; 122, 14; 124, 65, 67