CIS 452 - Operating Systems Concepts

Nathan Bowman

Images taken from Silberschatz book

Disk Scheduling

Possible to have several pending requests for disk reads or writes

Important to service these requests in way that minimizes overall time

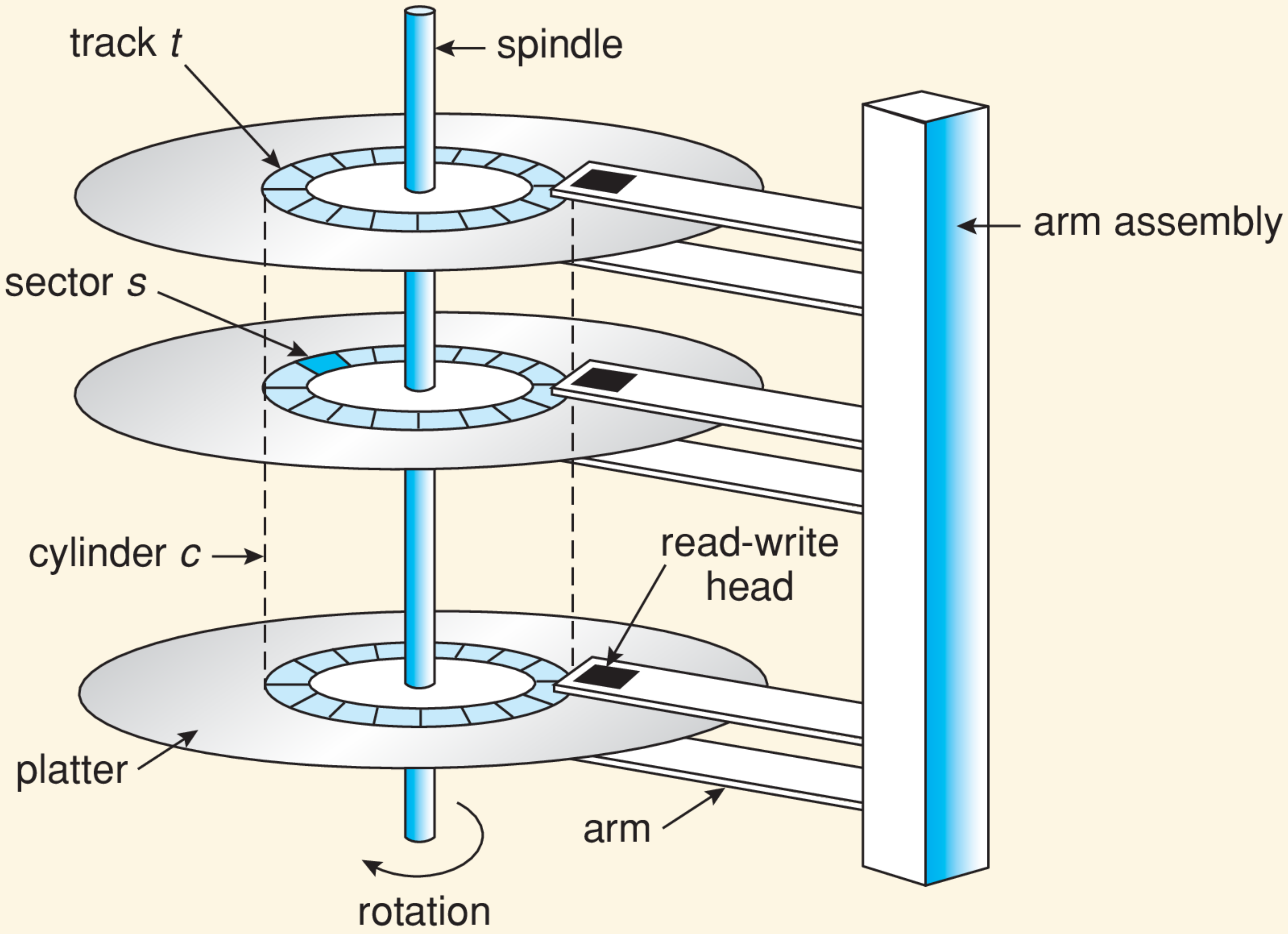
Note that this applies only to hard disks -- SSDs do not have seek time or rotational latency, so random access can be performed efficiently without considering order

Goal of scheduling algorithms will be to minimize seek time

Consider only which cylinder is requested, not specific sector

Put simply, trying to minimize amount disk head must move to satisfy requests

Work with example list of cylinder requests, such as 98, 183, 37, 122, 14, 124, 65, 67


track *t*
spindle
arm assembly
sector *s*
cylinder *c*
read-write head
platter
arm
rotation

As usual, begin with **first-come, first-served** (FCFS) scheduling

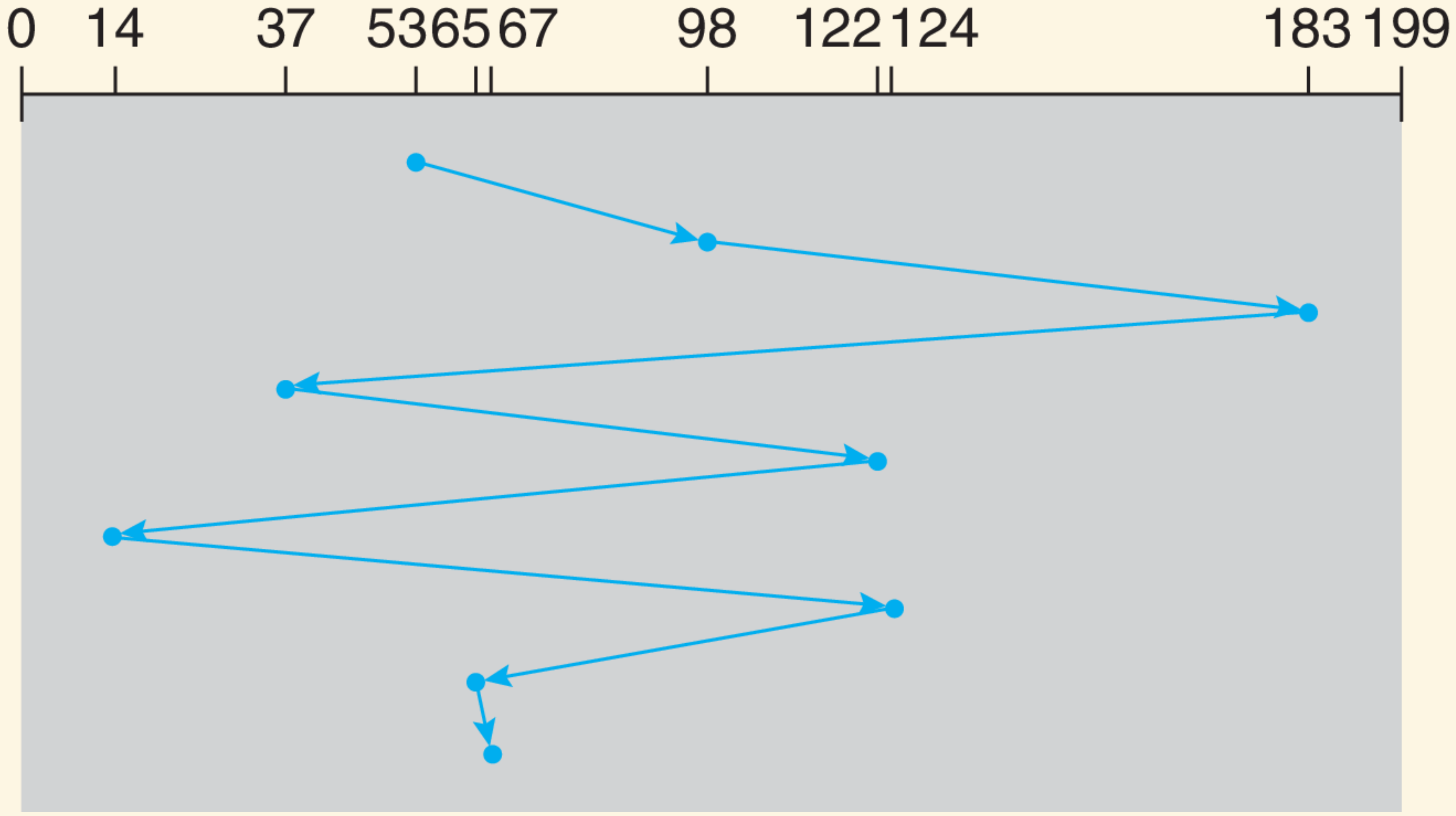
Fair, but not efficient

Consider performance on example request:

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

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

head starts at 53



A lot of back and forth movement that does not take into consideration nearby requests

Next idea: make locally optimal choice

Shortest-seek-time-first (SSTF) -- fulfill request with least seek time from current head position

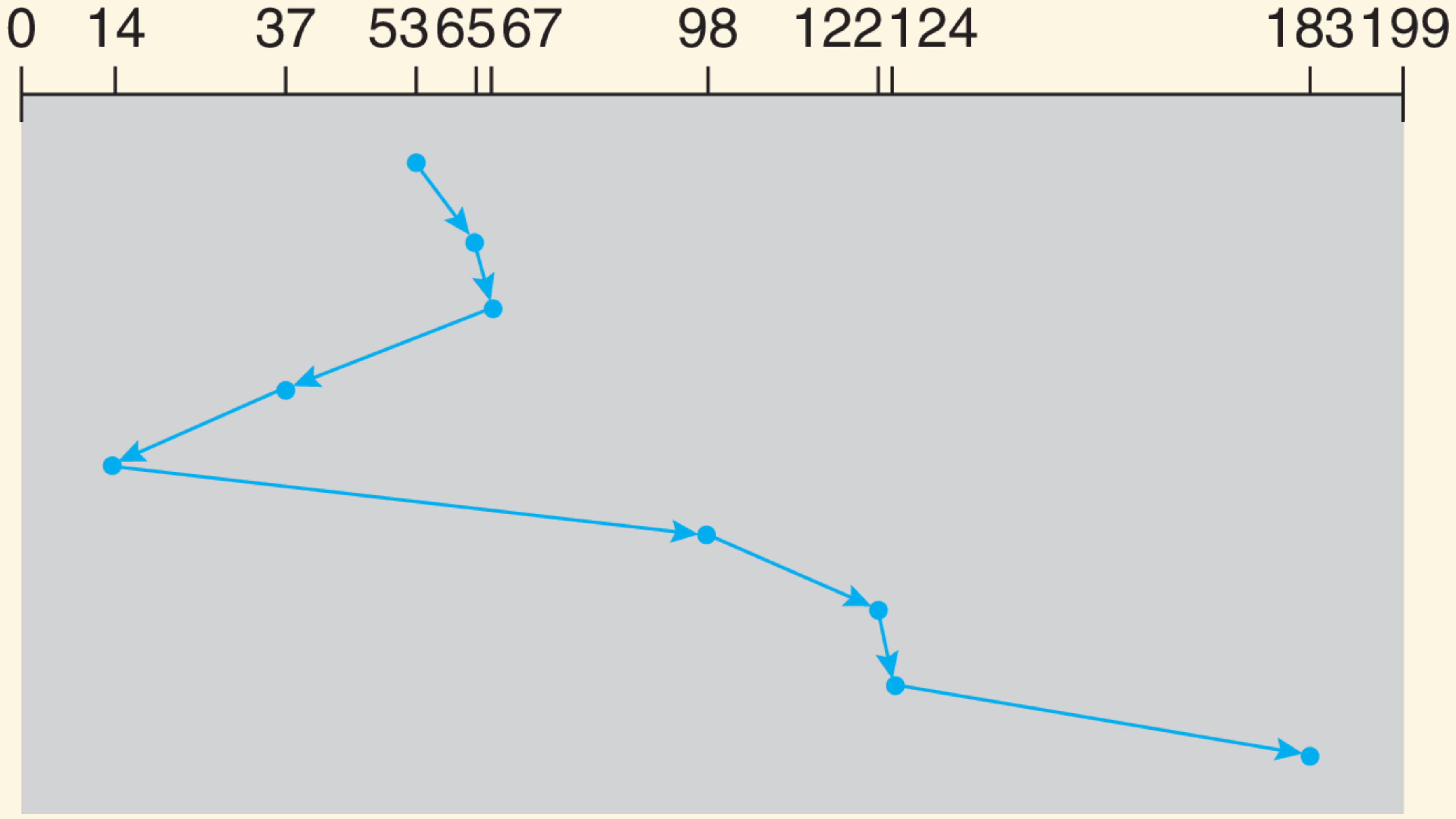
Put simply -- go to closest request

Consider same requests:

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

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

head starts at 53



Much more efficient -- about 1/3 of distance traveled compared to FCFS for this example

Note similarity to shortest-job-first scheduling algorithm for CPU

Like SJF, SSTF algorithm can result in starvation

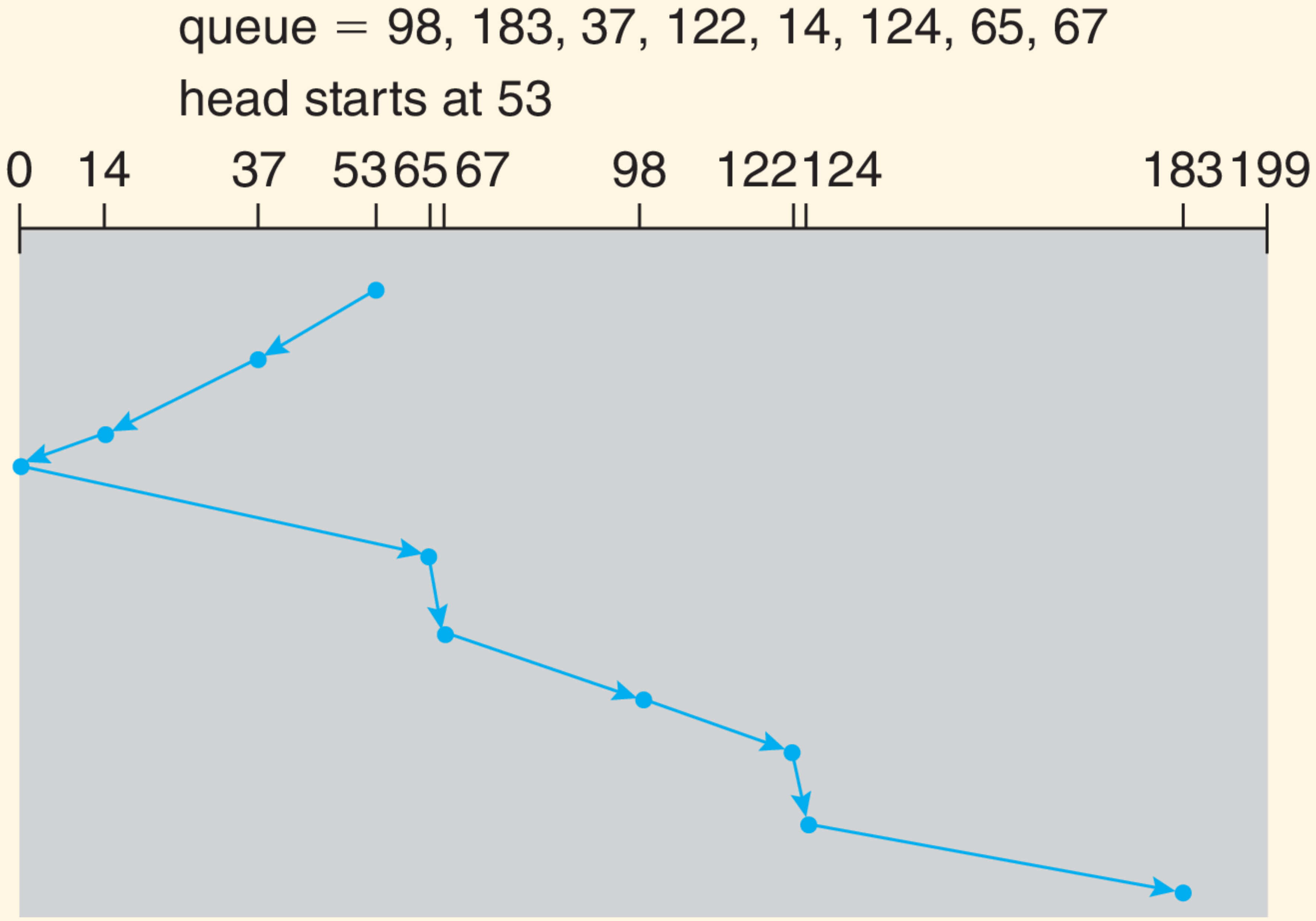
Unlike SJF, SSTF not optimal

Next algorithm tries to maintain fairness while servicing nearby requests

SCAN algorithm moves from one end of disk to other in back-and-forth scan, servicing requests as it encounters them

Consider same requests:

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


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

Back-and-forth motion of SCAN leads to strange access pattern for ends of disk

Consider disk with 200 cylinders (numbered from 0)

On "inward" (199 -> 0) sweep, cylinder 1 accessed, then cylinder 0, then cylinder 1 again

On "outward" (0 -> 199) sweep, cylinder 1 accessed, then cylinders 2-199, then cylinders 198-2, then cylinder 1 again

Wait times very uneven near ends of disk

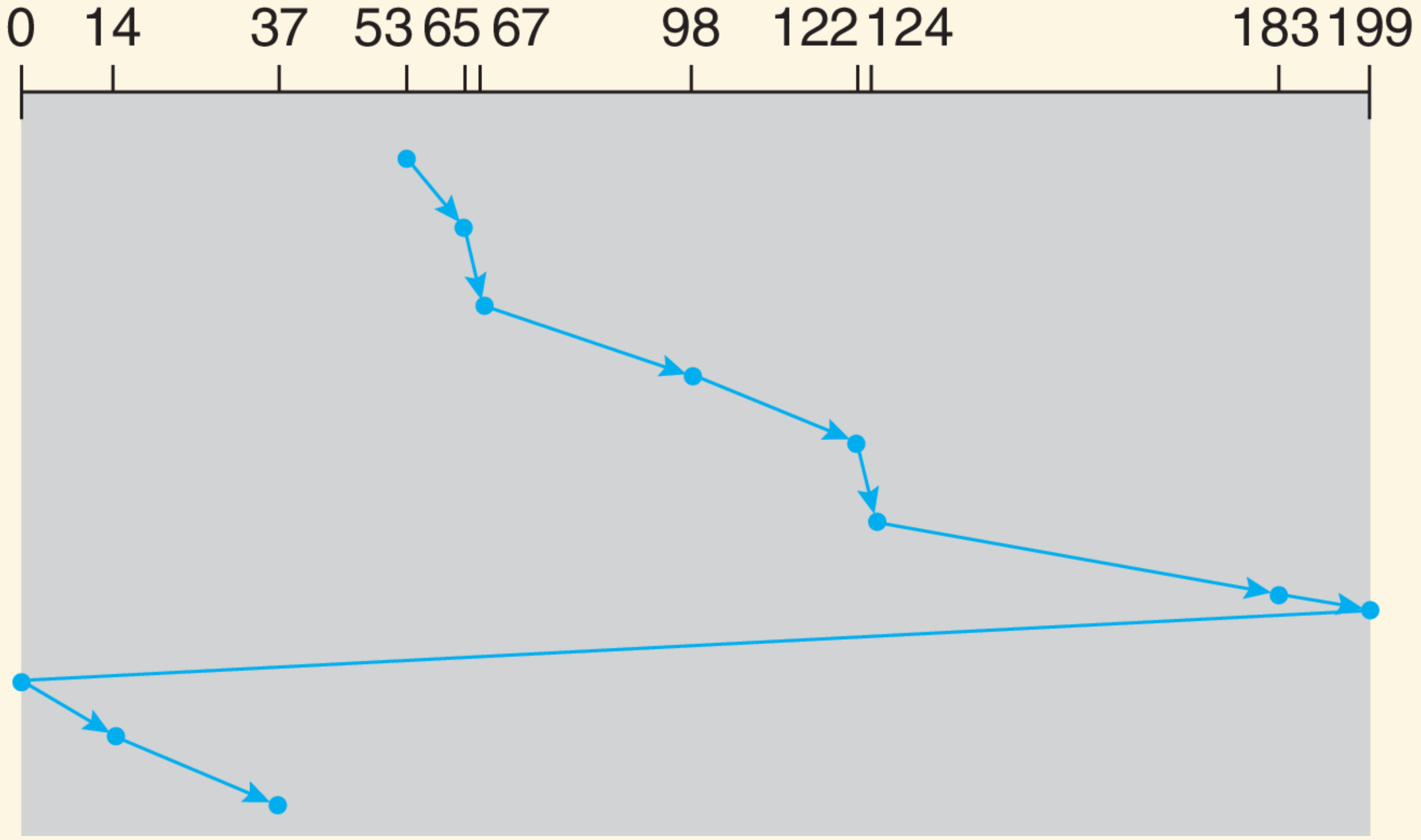
In our example, delay of 2 between accesses and then delay of ~200 until next access

Also, why bother servicing cylinder 1 if it was just accessed? Unlikely to have another request there

Circular SCAN (C-SCAN) scheduling "circles back" to other side of disk when head reaches end

Provides more uniform wait time

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



Both SCAN and C-SCAN waste time by moving to end of disk when no accesses requested

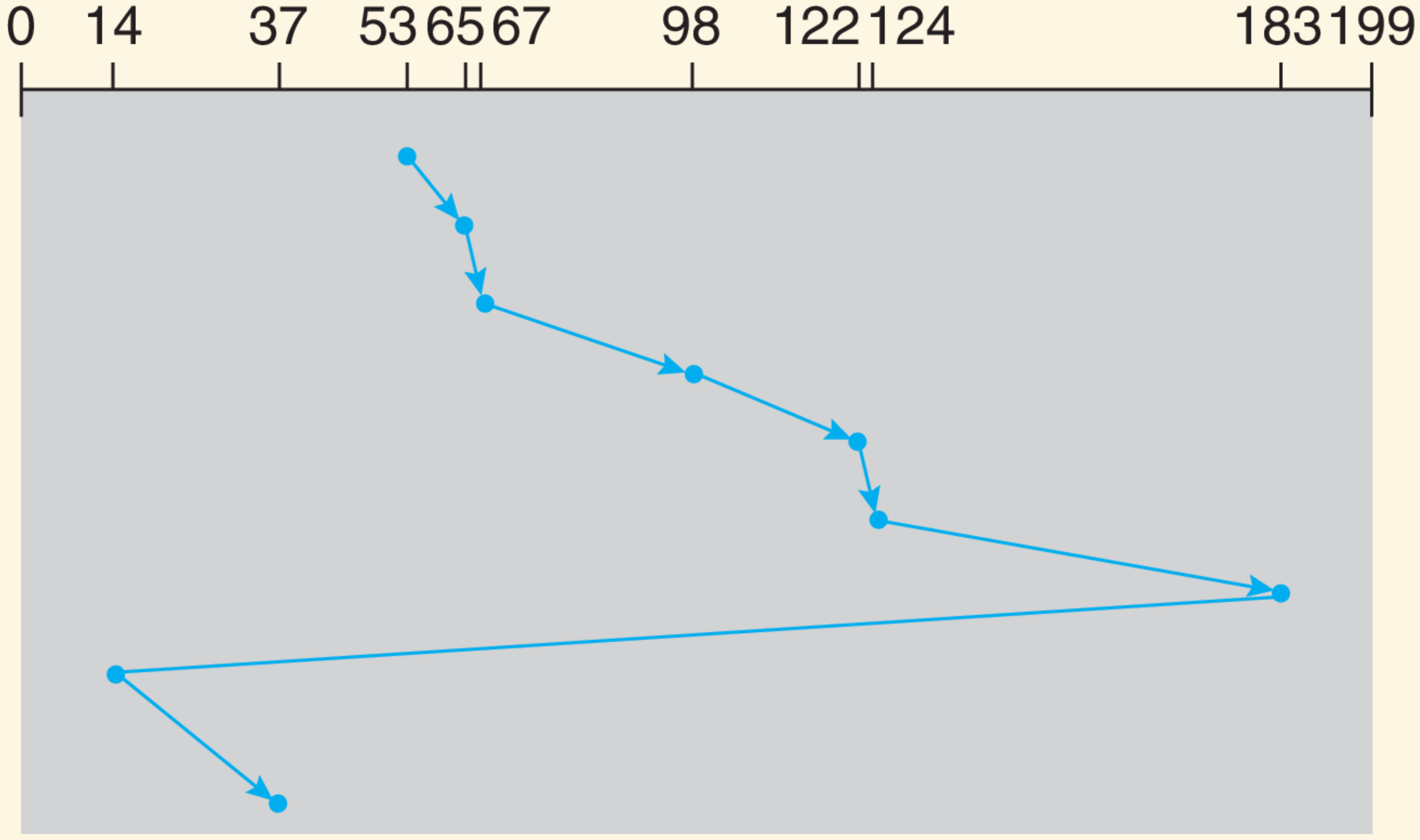
In actual implementation, would only go as far as final request in given direction before turning around (SCAN) or jumping back to start (C-SCAN)

Versions implementing this optimization are LOOK and C-LOOK, respectively

Example of C-LOOK shown next

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

head starts at 53



No one best choice of algorithm, but SSTF or LOOK reasonable choices for default