Hệ Điều Hành

(*Nguyên lý các hệ điều hành*)

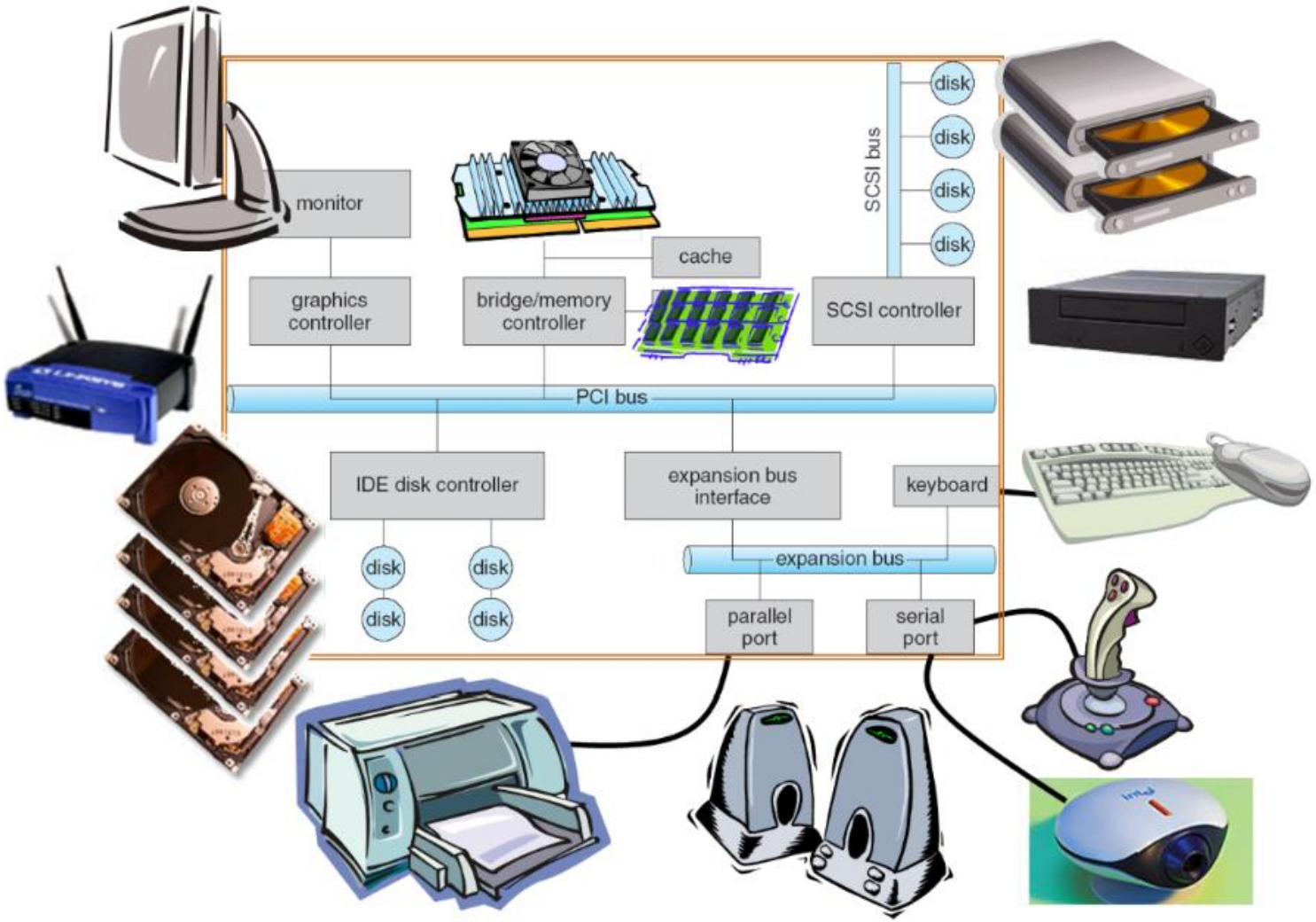
Đỗ Quốc Huy
huydq@soict.hust.edu.vn
Bộ môn Khoa Học Máy Tính
Viện Công Nghệ Thông Tin và Truyền Thông

Chapter 5 I/O Management

① General management principle
② System I/O service
③ Disk I/O system

- Introduction
- Interrupt and Interrupt handle

Chapter 5 I/O management



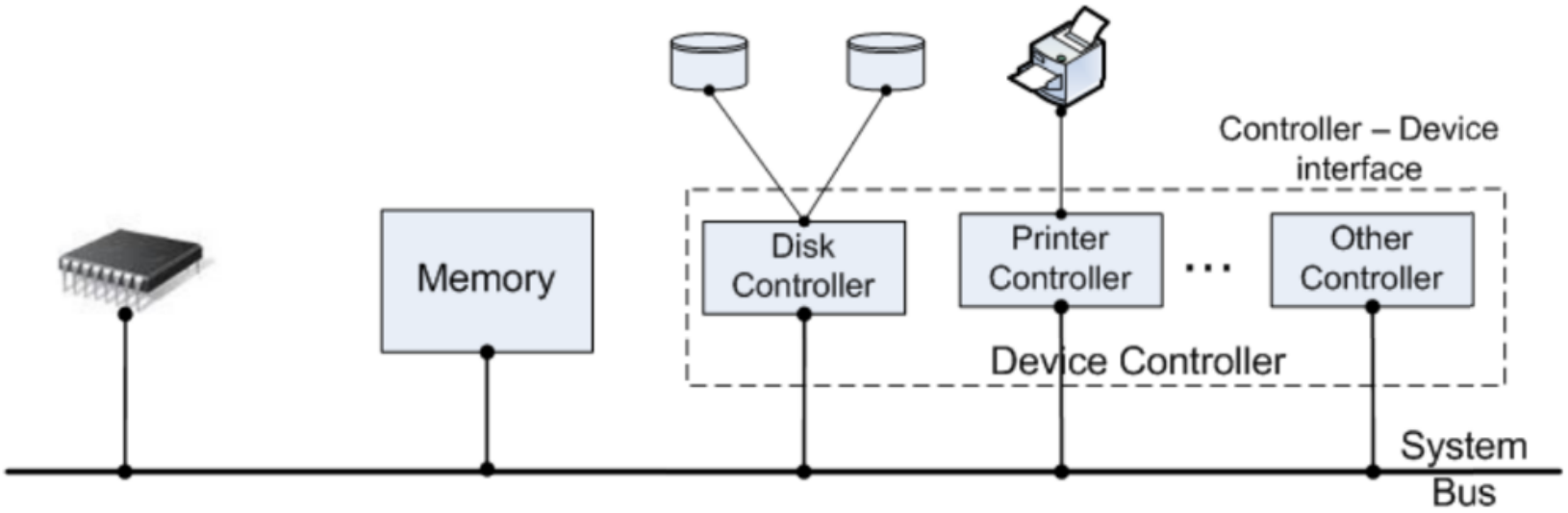
IO device

- Diversity, many kinds, different types
- Engineering perspective: device with processor, motor, and other parts
- Programming perspective: Interface like software to receive, executive command and return result
- Catergorize
  - Block device (disk, magenetic tape)
    - Info mation is stored with fixed size and private address
    - Possible to read/write a block independent from other
      - Operation to locate information exist (seek)
  - Character device (printer, keyboard, mouse,..)
    - Accept a stream of chracter, without block structure
    - No information localization operation
  - Other type: Clock

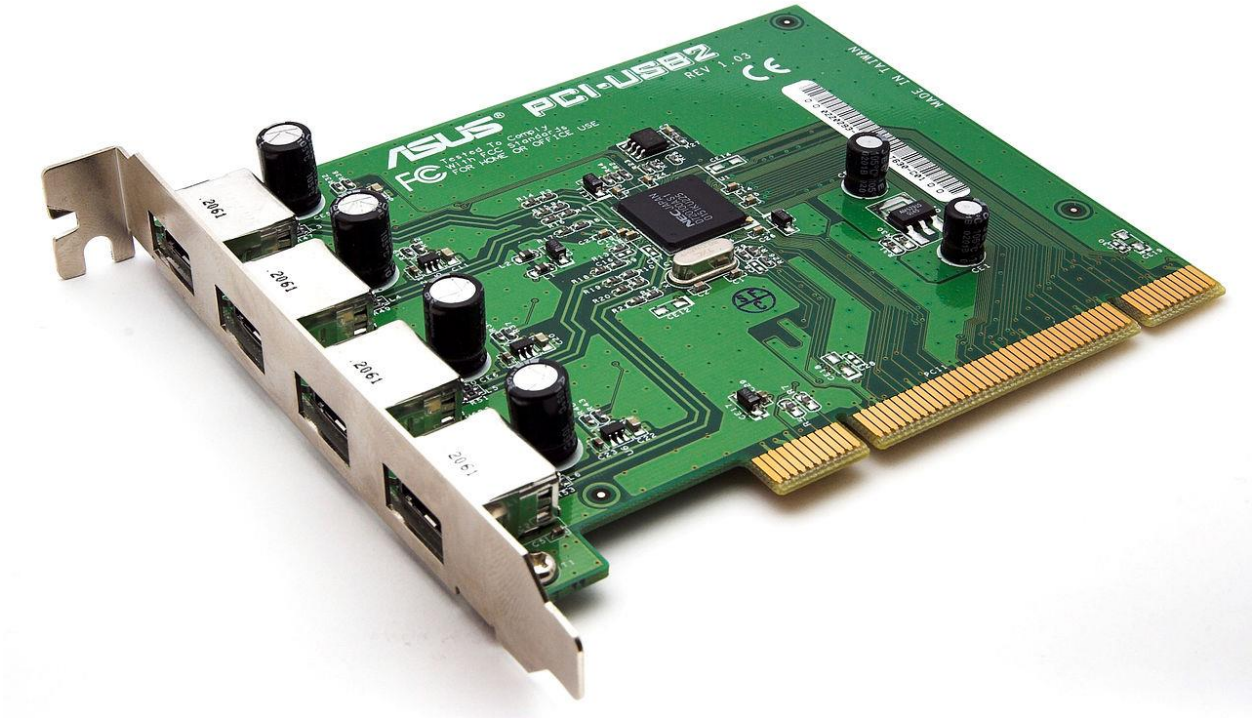
Controller device I

- Peripheral devices are diversity with many types
  - CPU do not know them all ⇒ No individual signal for each device
- Processor do not control device directly
  - Peripheral device is connected to the system via Device controller (DC)
    - Electrical circuit attached to the mainboard's slot
  - Each DC can control 1,2,4,.. peripheral devices
    - Depend on the number of connector on the DC
    - If the controller interface is standard (ANSI, IEEE, ISO,...) -> can connect to different devices
  - Each DC has its own register to work with CPU
    - Use special address space for registers: IO port

Controller device II



Controller device III



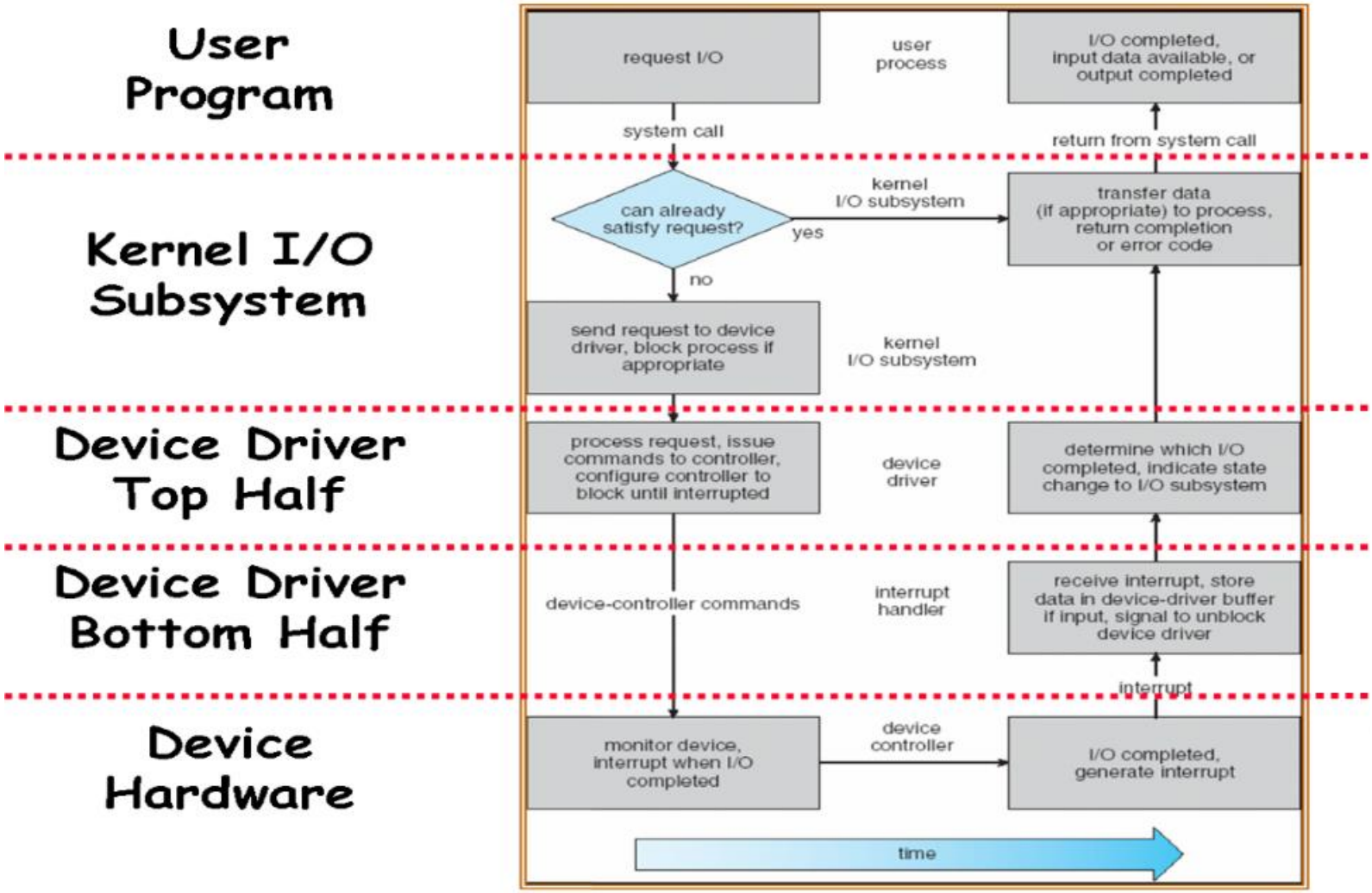
Controller device III

- Controller and device interface: Low level interface
  - Sector = 512bytes = 4096bits
  - Disk controller must read/write bits and group them into sectors
- OS only work with controller
  - Via device's registers
  - Commands and parameters are putted into controller's registers
  - When a command is accepted by the controller, CPU let the controller work itself and turn to other job
  - When command is finished, controller notify CPU via interrupt signal
  - CPU take result and device status via controlling device's register

Device driver

- Code segment in system's kernel allow interactive with hardware device
  - Provide standard interface for different I/O devices
- Categorized into 2 levels
  - High level: Access via system calls
    - Implement standard calls: open(), close(), read(), write()...
    - Interface between kernel and driver
    - High level thread wake up IO device then put control device thread into temporary sleep
  - Low level: Perform via interrupt procedure
    - Read input data or bring out next data block
    - Wake up the High level's temporary sleep thread when IO finish

IO request cycle



Peripheral device – Operating system interact

- After sending request to device, OS need to acknowledge
  - When device finish request
  - If device has error
- 2 methods to acknowledge
  - I/O interrupts
    - Device generate an interrupt signal to let CPU know
    - IRQ: physical path to interrupt manager
      - Map IRQ signal to interrupt vector
      - Call to interrupt handle routine
  - pooling
    - OS timely check device's status register
    - Waste checking period if the IO operation is not frequent
- Nowadays device can combine 2 methods (E.g. high bandwidth network device)
  - Send interrupt when first packet arrive
  - Pooling next coming packet until the buffer is empty

- Introduction

- Interrupt and Interrupt handle

Interrupt definition

Mechanism to help device let the processor know its status

Phenomenon that a process is suddenly stopped and the system executive other process correspond to an event

Classification

- Based on Source
  - Internal interrupt
  - External interrupt
- Based on device
  - Hard
  - Soft
- Based on handling ability
  - maskable
  - unaskable
- Based on interrupt moment
  - Request
  - Report

Interrupt handle

① Write characteristic of event caused the interrupt into defined memory area

② Save interrupted process 'state

③ Change address of interrupt handle routine to instruction pointer register

- Utilize interrupt vector table (IBM-PC)

④ Run interrupt handle routine

⑤ Restore interrupted process

- Interrupt >< procedure !?

Chapter 5 I/O Management

① General management principle
② System I/O service
③ Disk I/O system

Chapter 5: IO Management
2. System I/O service
2. 1 Interrupt and Interrupt handle

- Buffer
- SPOOL mechanism

General notion

- Peripheral device's characteristic: operate slow
  - Active the device
  - Wait for device to get to proper working status
  - Wait for IO operation to be perform
- To Guarantee the system's performance -> need to
  - Reduce number of IO operations, work with block of data
  - Perform IO operations parallelly with other operations
  - Perform accessing operation in advance

Buffer: Intermediate memory area, utilized for storing information during IO operation

Buffer classification I

- Input buffer
  - Can perform data access command
  - Example: read data from disk
- Output buffer
  - Information is putted into buffer, when buffer full, buffer content is then write to device

Buffer classification II

- Buffer attached to device
  - Constructed when open device/file
  - Serve device only, cleared when device is close
  - Good when devices have different physical record's structures
- Buffer attached to system
  - Constructed when the system start, not attached to a specific device
  - Exist during system working process
  - Open file/device ⇒ attach to already available buffer
  - Close device/file ⇒ buffer returned to system
  - Good for devices have same physical record's structure

Buffer organization

- Value buffer
  - Input buffer
  - Output buffer
- Process buffer
- Circular buffer
  - Input buffer
  - Output buffer
  - Processing buffer

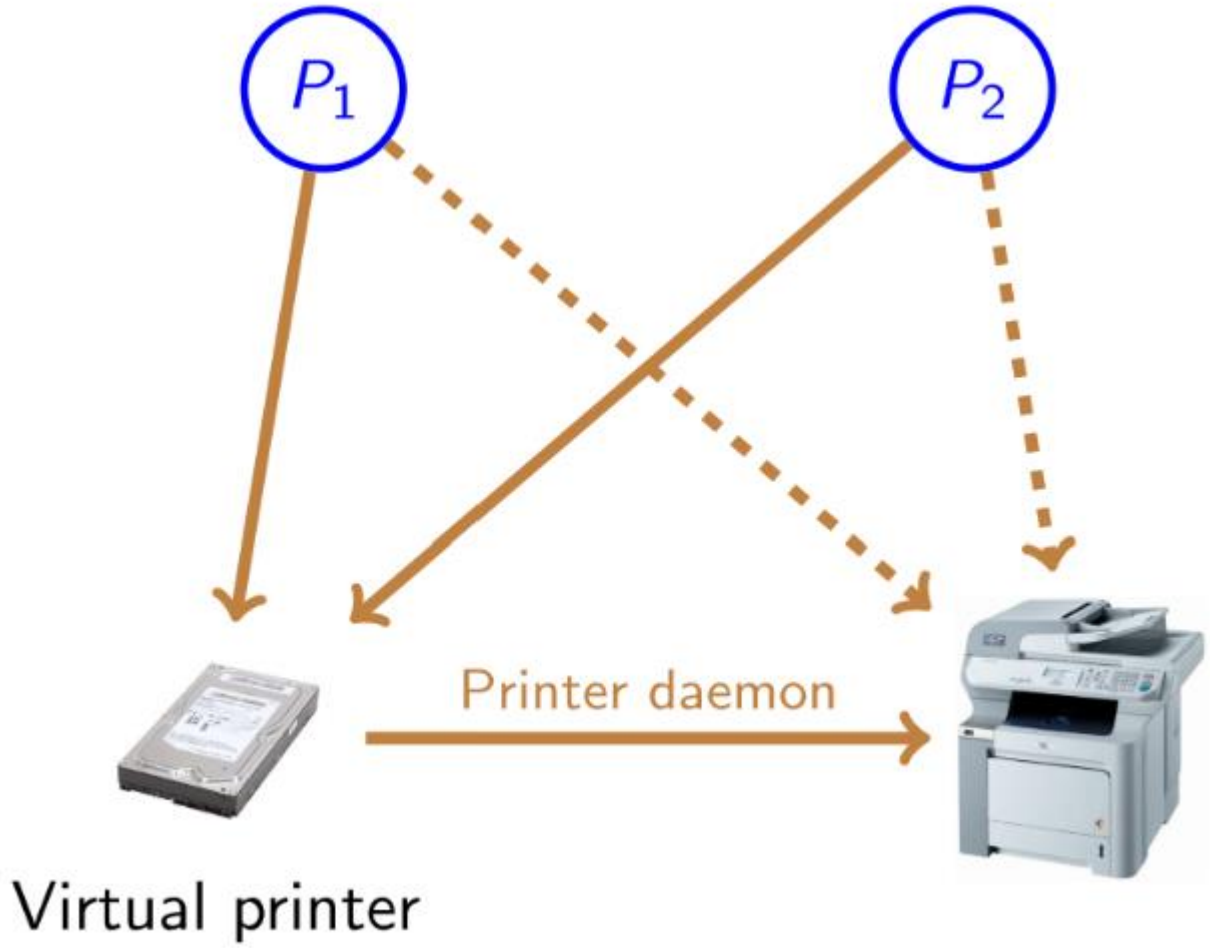
Chapter 5: IO Management
2. System I/O service
2. 1 Interrupt and Interrupt handle

- Buffer
- SPOOL mechanism

SPOOL (Simultaneous Peripheral Operation On-line)

- From programming perspective, IO device is
  - Station to receive request from program and perform
  - Return status code to be analyzed by the system
- -> use software to simulate IO device
  - IO device can be treated as process
    - Synchronized like in process management
- Objective
  - Simulate process of controlling and managing peripheral device
    - Check creating device working status
  - Create parallel effect for sequence device

SPOOL: Virtual printer



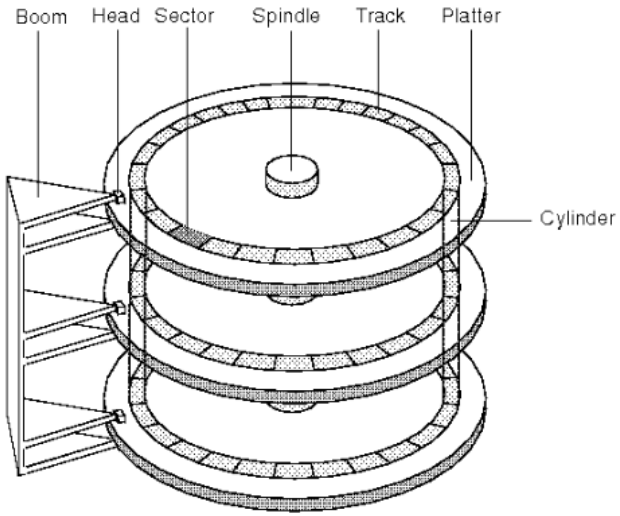
Chapter 5 I/O Management

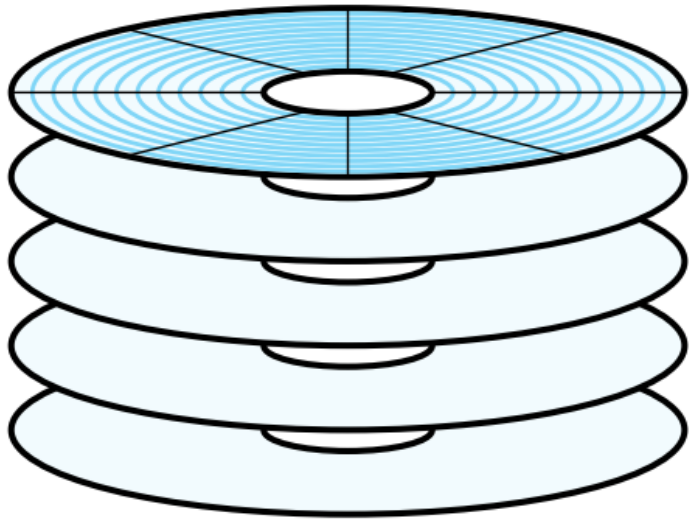
1. General management principle
2. System I/O service
3. Disk I/O system

Chapter 5: IO Management
3. Disk I/O device

- Disk structure
- Disk accessing scheduling

Structure





- Modelled as array of logic block
  - logic block is the smallest exchange unit
- Map continuous logic block to disk's sector
  - Block 0 is first sector header 0 outer most track/Cylinder
    - Mapping follow an order: Sector → Header → Track/Cylinder
  - Reading header do not need to move much when read sector next to each other

Disk accessing problem

- OS is respond for effectively exploit the hardware
  - For disk: Fast access time and high bandwidth
- Bandwidth is calculated based on
  - Total bytes exchanged
  - Time from the first service request until the request is completed
- Access time consist of 2 part
  - seek time : Time to move header to cylinders contain required sector
  - Rotational latency: Time to wait until disk rotate to required sector

Disk accessing problem

- Hệ điều hành có trách nhiệm sử dụng hiệu quả phần cứng
  - Với đĩa: Thời gian truy nhập nhanh và băng thông cao
- Băng thông được tính dựa trên
  - Tổng số bytes đã trao đổi
  - Khoảng thời gian từ y/cầu dịch vụ đầu tiên cho tới khi hoàn thành
- Thời gian truy nhập gồm 2 phần
  - Thời gian định vị (seek time) : Thời gian dịch chuyển đầu từ tới cylinders chứa sector cần truy nhập
  - Độ trễ quay (Rotational latency) :Thời gian chờ đợi để đĩa quay tới sector cần truy nhập
- Mục đích: cực tiểu hóa thời gian định vị
  - Thời gian định vị ≈khoảng cách dịch chuyển
- Hàng đợi yêu cầu
  - Đĩa và bộ đ/khiển sẵn sàng, y/cầu truy nhập đc thực hiện ngay
  - Đĩa/bộ đ/khiển chưa sẵn sàng, yêu cầu đc đặt trong hàng đợi
  - Hoàn thành một yêu cầu truy nhập đĩa, lựa chọn y/cầu nào?

- Disk structure
- Disk accessing scheduling

Algorithm

- Objective: minimize seek time
  - Seek time ≈ moving distance
- Algorithm for disk IO request scheduling

  FCFS:First Come First Served

  SSTF: Shortest Seek Time First

  SCAN

  C-SCAN: Circular SCAN

  LOOK/C-LOOK

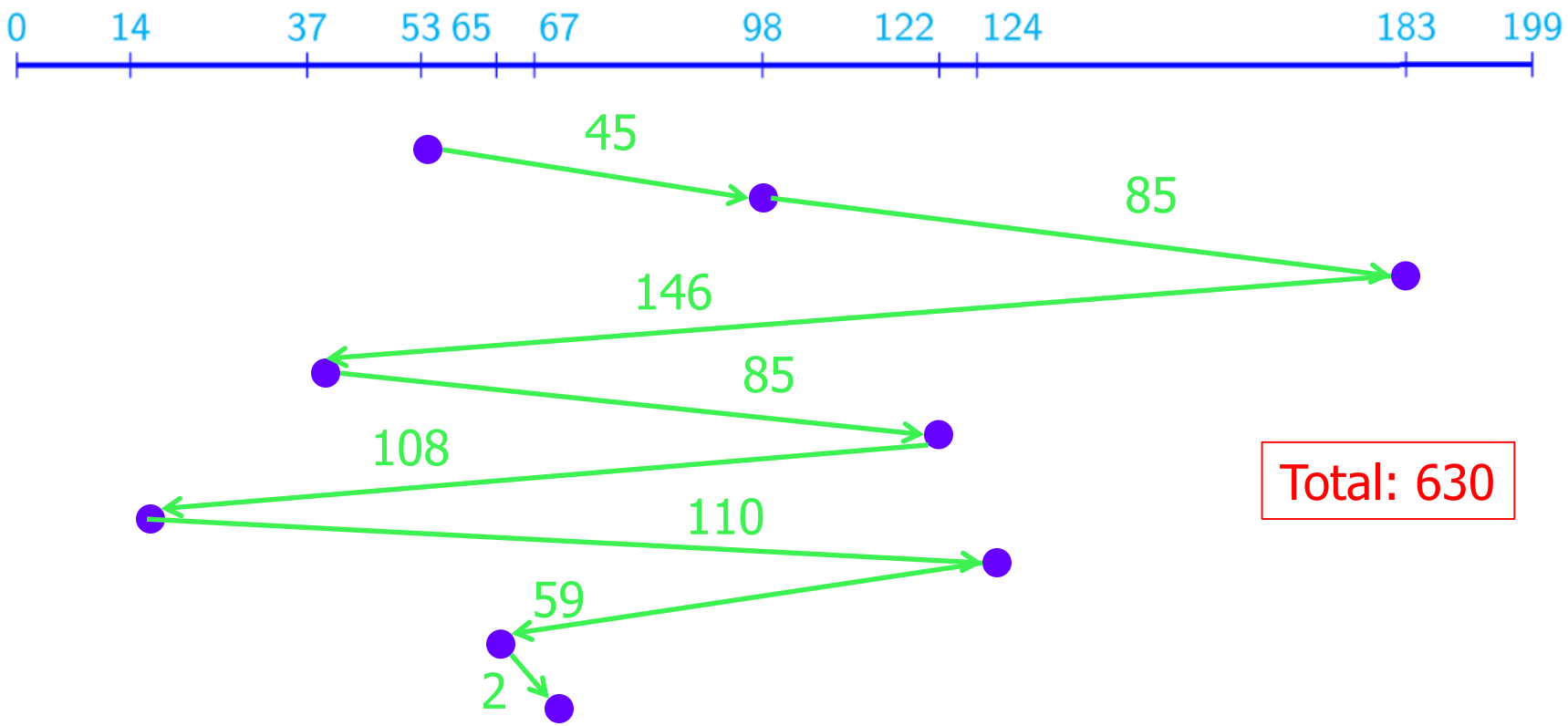
Assumption

- Accessing requests 98, 183, 37, 122, 14, 124, 65, 67
- Header current position at cylinder 53

FCFS

Access follow the request order ⇒Not effective

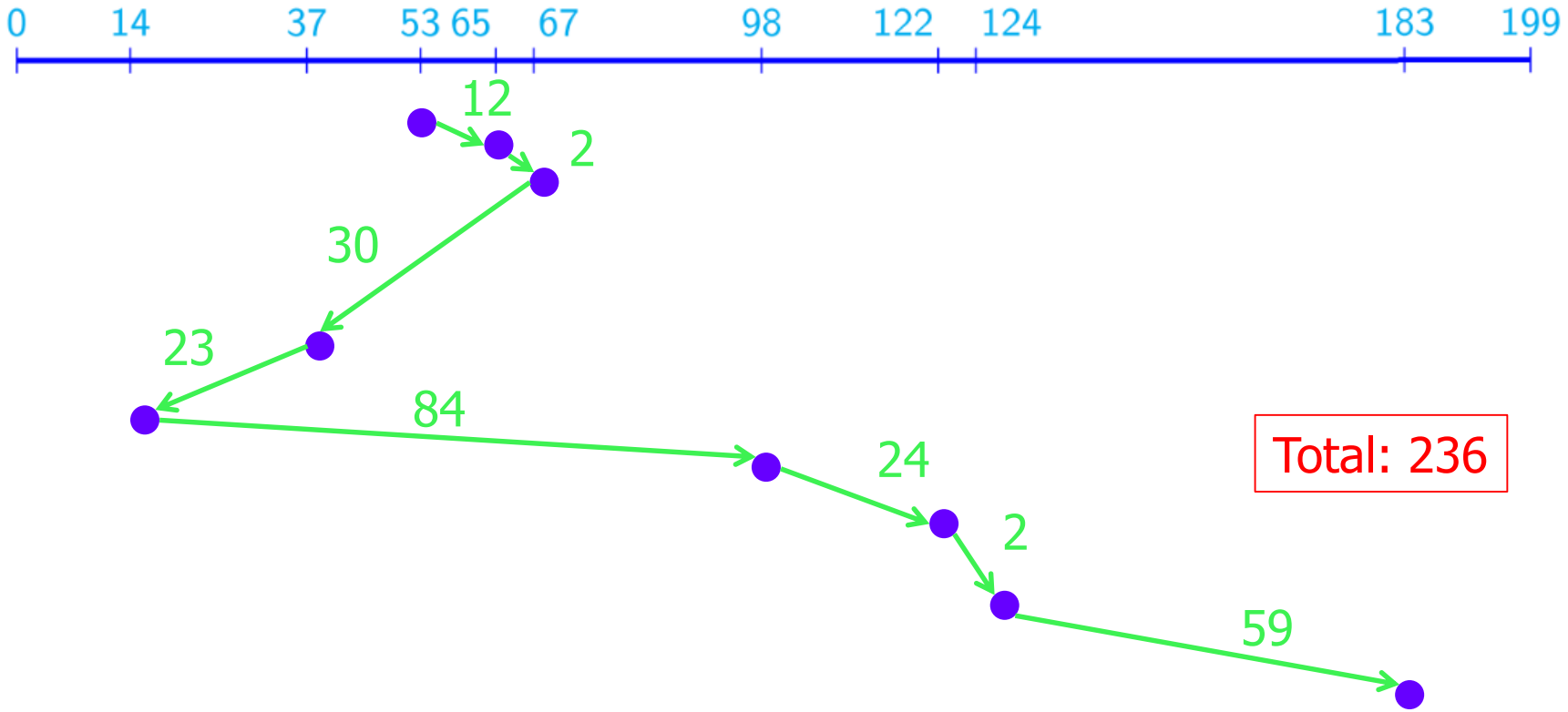
Accessing requests 98, 183, 37, 122, 14, 124, 65, 67



SSTF

Select access has smallest seek time from current position ⇒ A request may wait forever if new appearing requests closer to header (similar to SJF)

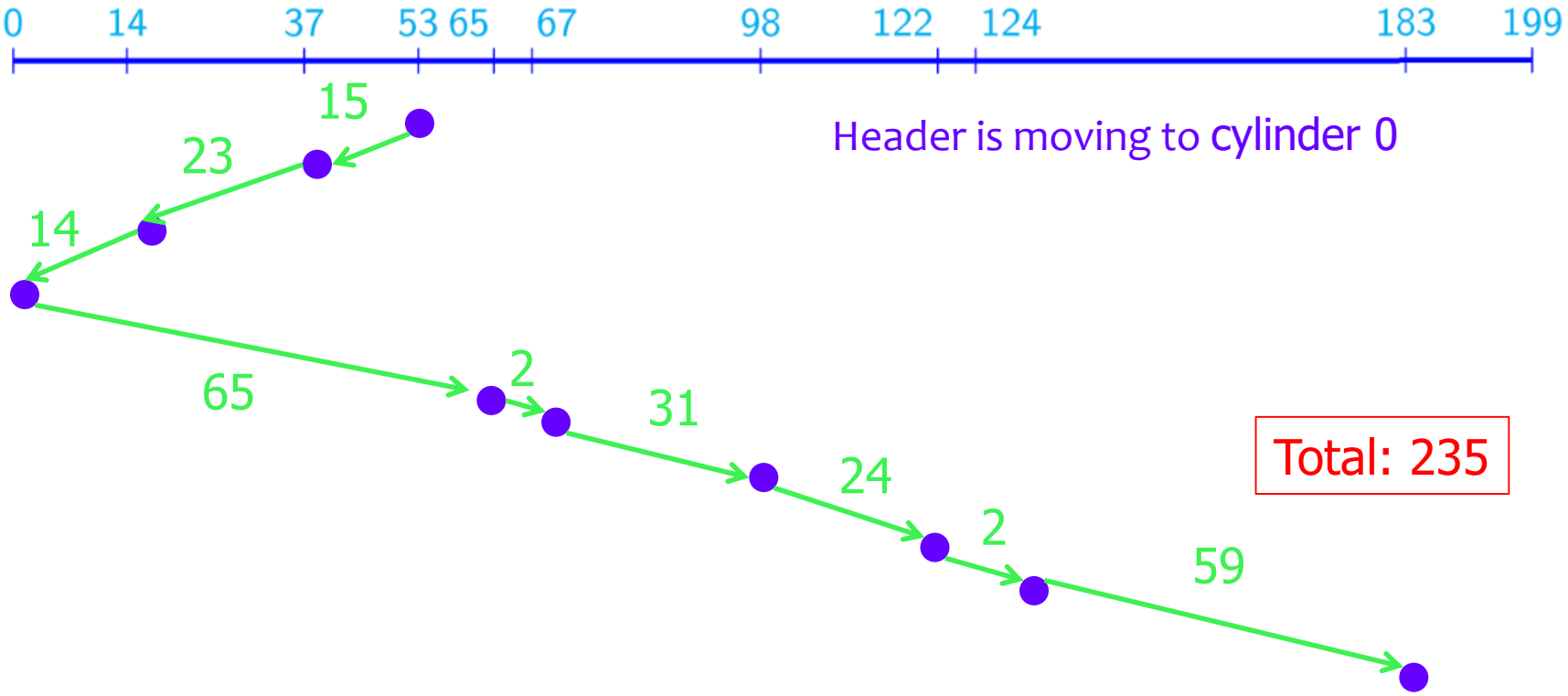
Accessing requests 98, 183, 37, 122, 14, 124, 65, 67



SCAN

Header move from outer most cylinder to innermost cylinder and return. Serve request met on the way

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



C-SCAN

Principle: Treat cylinders like a circular linked list: Outer most Cylinder connect with innermost cylinder

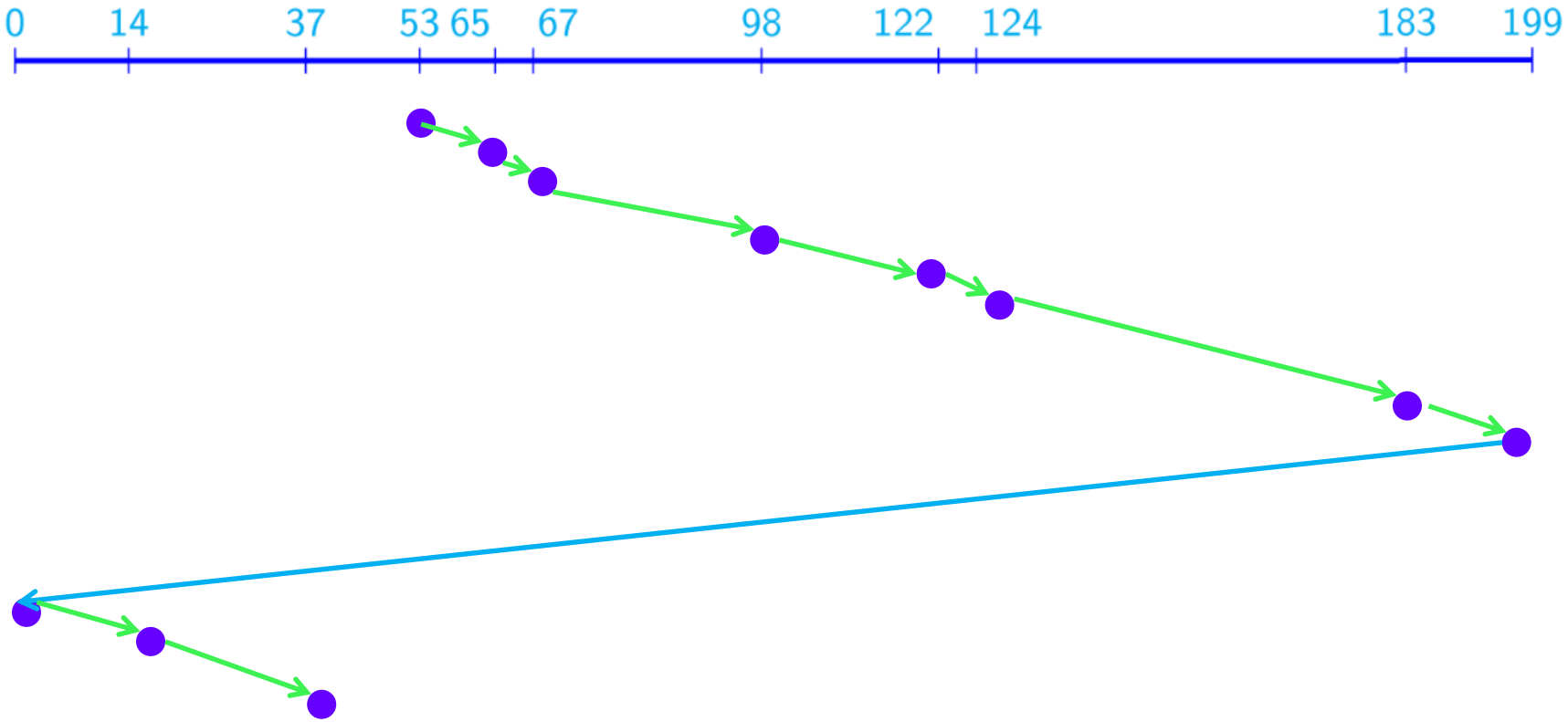
- Header move from outermost cylinder to innermost cylinder
  - Serve request met on the way
- When inner most Cylinder is reached, return to outermost Cylinder
  - Do not serve request met on the way

- Remark: Retrieve more equal waiting time than SCAN
  - When header reach to one side of disk (innermost/outermost cylinders), density of requests appear at other side will be higher than current place (reason: header just passing by). This request need to wait longer ⇒ Return to other side immediately

C-SCAN

Header move from outermost cylinder to innermost cylinder and return. Serve request met on the way

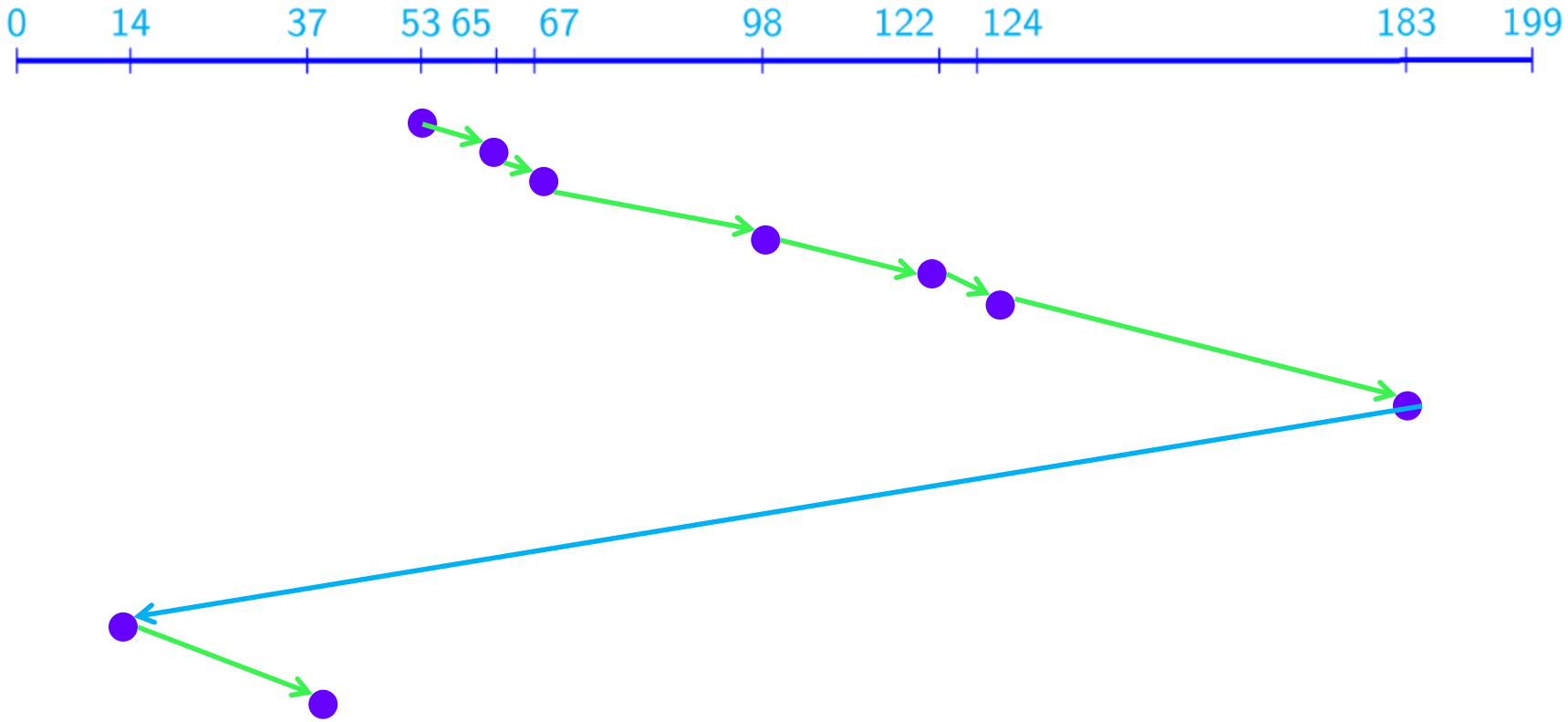
Accessing requests 98, 183, 37, 122, 14, 124, 65, 67



LOOK/ C-LOOK

SCAN/C-SCAN's version: Header does not move to outermost/innermost cylinders, only to farthest request at 2 sides and return

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



Conclusion