Protection and Security

Protection

- Processes need to be protected from one another's activities.
- Processes must operate only on those resources for which they have gained proper authorization from the operating system.
- Refers to a mechanism for controlling the access of programs, processes or users to the resources defined by a computer system.

Goals of protection

- To increase the reliability of any complex system that makes use of shared resources.
- To prevent mischievous, intentional violation of an access restriction by a user.
- To ensure that each program component active in a system uses system resources only in ways consistent with the stated policies for the uses of these resources (reliable system).

Goals of protection

- To provide means to distinguish between authorized and unauthorized usage.
- To provide a mechanism for the enforcement of the policies governing resource use.
- Mechanisms determine how something will be done.
- Policies decide what will be done.

Domain of protection

- A computer system consists of processes and objects (both hardware and software).
- Each object has a unique name and it can be accessed only through well-defined and meaningful operations.
- Objects – abstract data types.
- Operations possible may depend on the object.
- Only authorized access of resources by any process.

Domain of protection

- At any time, a process should be able to access only those resources that it currently requires to complete its task (need-to-know principle).
- To limit the amount of damage a faulty process can cause in the system.

Protection domain

- Specifies the resources that a process may access.
- A process operates within a protection domain.
- Each domain defines a set of objects and the types of operations that may be invoked on each object.
- Access right – the ability to execute an operation on an object.

- Domain – collection of access rights each of which is an ordered pair <object-name, rights-set>.
- Domains may share access rights.
- Association between a process and a domain may be either static or dynamic.
- If static, modification of domain contents.
- If dynamic, domain switching.

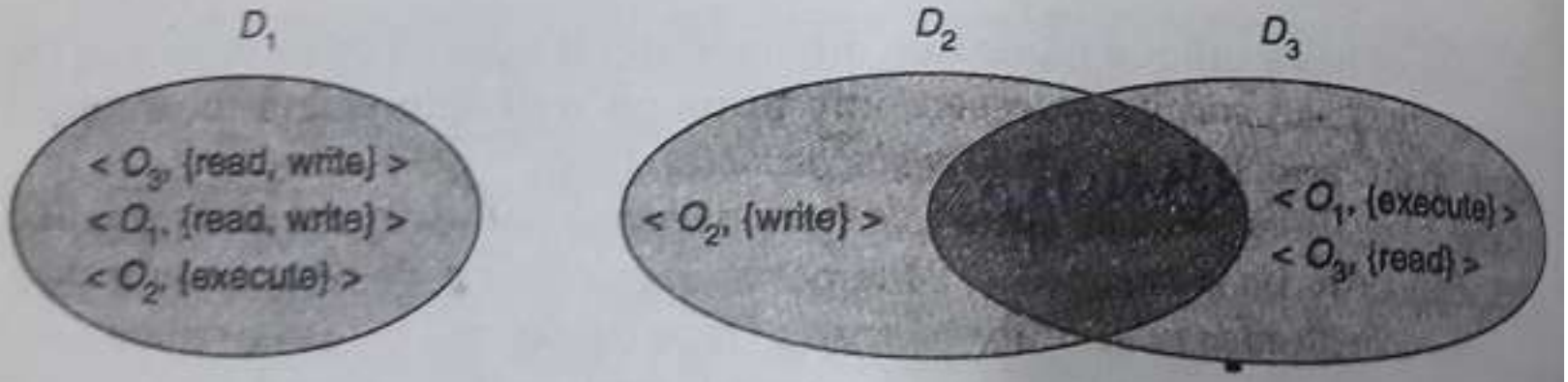


Figure 19.1 System with three protection domains.

Realization of a domain

- Each user may be a domain (set of objects that can be accessed depends on the user's identity).
- Each process may be a domain.
- Each procedure may be a domain (local variables defined within the procedure).

Access matrix

- Protection model can be viewed as a matrix.
- Provides mechanism for specifying a variety of policies.
- Entry access(i,j) defines the set of operations that a process executing in domain D_i can invoke on object O_j.
- Policy decisions involve which rights should be included in the (i,j)th entry.

Access Matrix

object / domain	F_1	F_2	F_3	printer
D_1	read		read	
D_2				print
D_3		read	execute	
D_4	read write		read write	

Access Matrix

- Provides an appropriate mechanism for defining and implementing strict control for both the static and dynamic association between processes and domains.
- Controlling domain switching.
- Domain switching from domain Di to domain Dj is allowed to occur iff the access right switch belongs to access(i,j).

object / domain	F_1	F_2	F_3	laser printer	D_1	D_2	D_3	D_4
D_1	read		read			switch		
D_2				print			switch	switch
D_3		read	execute					
D_4	read write		read write		switch			

Figure 19.4 Access matrix of Figure 19.3 with domains as objects.

- To allow controlled change to the contents of the access-matrix entries, three additional operations are required :

1. copy
2. owner
3. control

- The copy right allows a process to copy some rights from an entry in one column to another entry in the same column (transfer, limited copy).

object / domain	F_1	F_2	F_3
D_1	execute		write*
D_2	execute	read*	execute
D_3	execute		

(a)

object / domain	F_1	F_2	F_3
D_1	execute		write*
D_2	execute	read*	execute
D_3	execute	read	

(b)

Figure 19.5 Access matrix with *copy* rights.

- The owner right allows for the addition of new rights and removal of some rights.
- The copy and owner rights allow a process to change the entries in a column.
- To change the entries in a row, control right.
- The control right is applicable to only domain objects.
- If access(i,j) includes the control right, then a process executing in domain Di can remove any access right from row j.

domain \ object	F_1	F_2	F_3
D_1	owner execute		write
D_2		read* owner	read* owner write*
D_3	execute		

(a)

domain \ object	F_1	F_2	F_3
D_1	owner execute		
D_2		owner read* write*	read* owner write*
D_3		write	write

(b)

Figure 19.6 Access matrix with *owner* rights.

- copy and owner rights provide a mechanism to limit the propagation of access rights but they do not provide appropriate tools for preventing the propagation of information.
- Confinement problem – guaranteeing that no information initially held in an object can migrate outside of its execution environment.
- Unsolvable problem.

Security

- Measure of confidence that the integrity of a system and its data will be preserved.
- Requires an adequate protection system and consideration of the external environment within which the system operates.

Authentication

- Ensures and confirms a user's identity.
- Based on user possession(card), user knowledge(password), user attribute (fingerprint, signature).
- Password vulnerabilities – difficulty of keeping a password secret.

Password vulnerabilities

- Password can be compromised by -
  - ✓ guessing it (intruder having user's information or brute force).
  - ✓ exposure (visual or electronic monitoring).
    - Visual monitoring – shoulder surfing.
    - Network sniffing.
    - Hard-to-remember or long passwords.
  - ✓ human nature.
- System-generated passwords or user-selected passwords.

- Occasional checking of passwords by site administrators.
- Password aging.
- Change of password for each session.
- Encrypted passwords – Given a value x, it is easy to compute the function value f(x) but the reverse is impossible.
- Flaw that the system no longer has control over the passwords.

One-time passwords

- To prevent improper authentication due to password exposure.
- Use of set of paired passwords.
- Use of algorithmic passwords –
  - ✓ system and the user share a secret and seed.
  - ✓ f(secret, seed).
  - ✓ Seed is a random number or alphanumeric sequence and is the authentication challenge from the computer.

Threats

- Trojan Horse - traps user login credentials and stores them to send to malicious user who can later on login to computer and can access system resources.
- Trap door - If a program which is designed to work as required has a security hole in its code and perform illegal action without knowledge of user then it is called to have a trap door.

Threats

- Worm – a process that creates multiple copies of itself.
- Virus – a fragment of code embedded in a legitimate program.
- Self-replicating, designed to infect other programs.

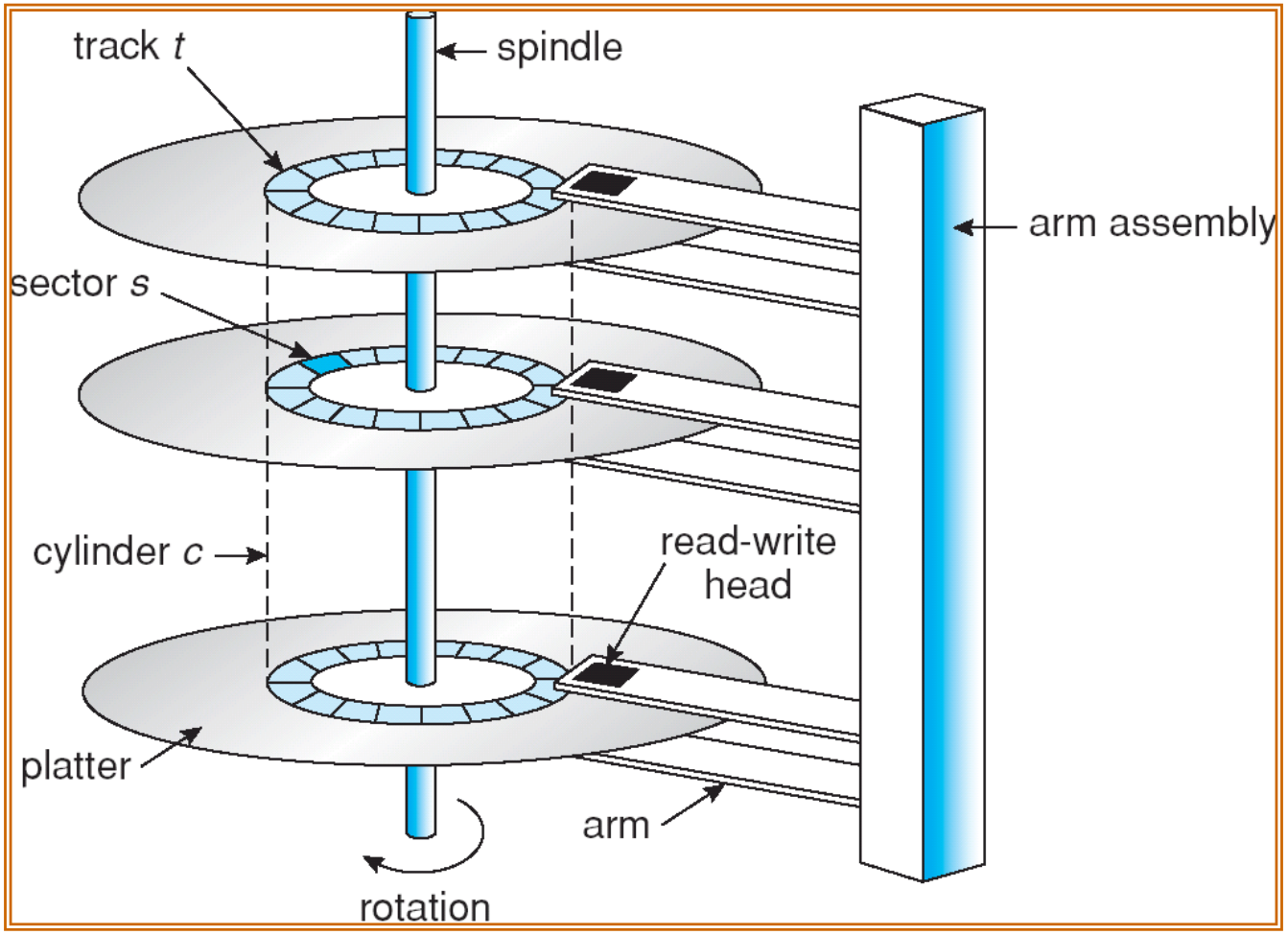
DEVICE MANAGEMENT

Magnetic disks

- Provide the bulk of secondary storage for modern computer systems.
- Consist of several disk platters which have a flat circular shape.
- Platter surfaces covered with a magnetic material.
- Information recorded magnetically on the platters.

- Each platter surface is divided into circular tracks which are subdivided into sectors.
- Set of tracks at one arm position form a cylinder.
- A read-write head flies just above each surface of every platter.

Moving-head disk mechanism



- A drive motor spins the disk at a high speed (60 to 150 times per second).
- Two parts of disk speed –
  - ✓ *transfer rate* – rate at which data flows between the drive and the computer.
  - ✓ *positioning time (random access time)* –
    - *Seek time* – time to move the disk arm to the desired cylinder.
    - *Rotational latency* – time for the desired sector to rotate to the disk head.

- Head crash – if the disk head comes in contact with the disk surface, it can damage the surface.
- Cannot be repaired, disk needs to be replaced.

Disk Scheduling

- Operating system needs to use the hardware efficiently.
- In case of disk drives, fast access time and disk bandwidth.
- *Disk bandwidth* – total number of bytes transferred divided by the total time between the first request for service and the completion of the last transfer.

Disk Scheduling Algorithms

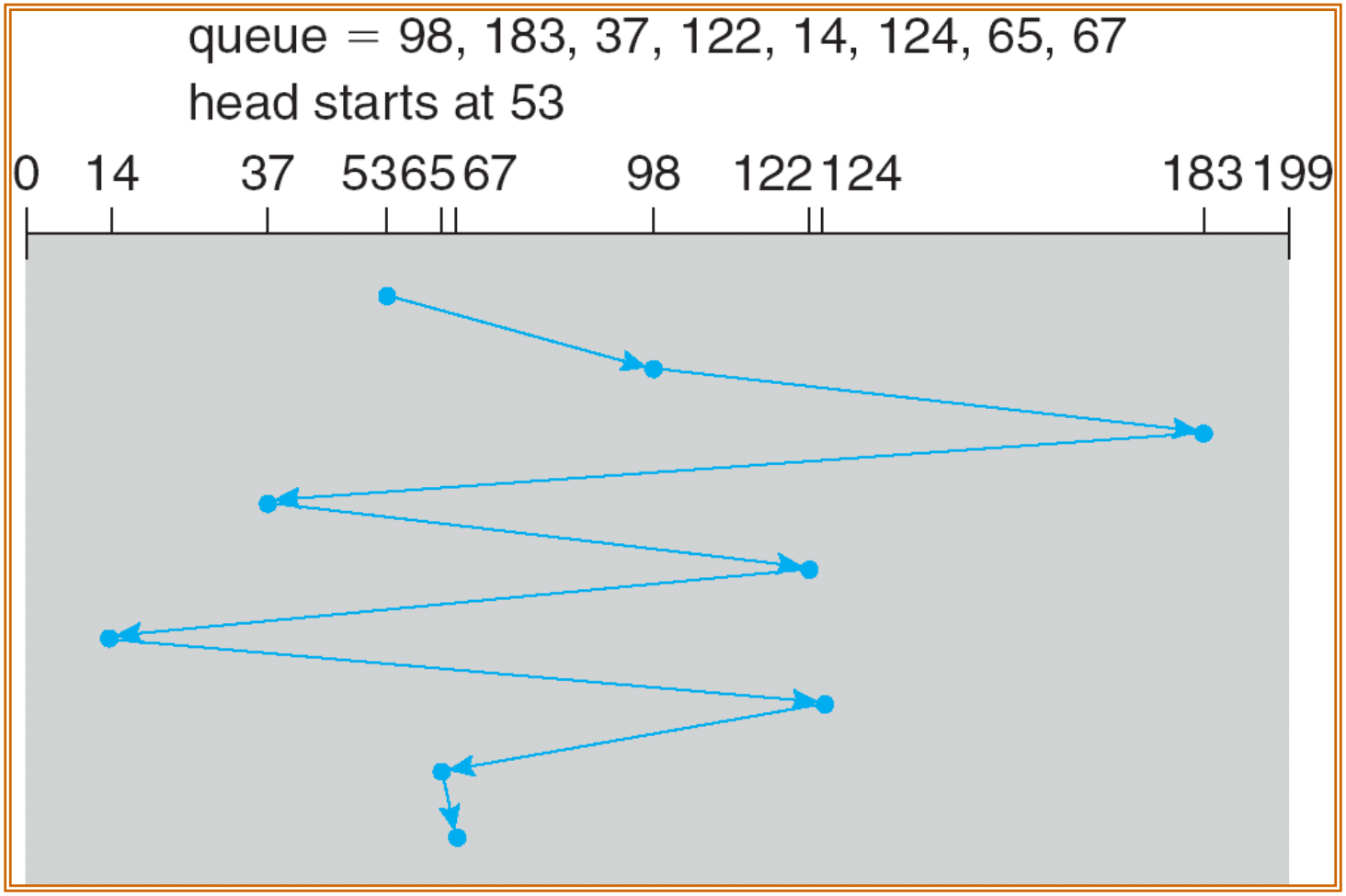
1. *First-come, first-served (FCFS) Scheduling –*

- A disk queue with requests for I/O to blocks on cylinders

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

- Disk head is initially at cylinder 53.
- Total head movement of 640 cylinders.

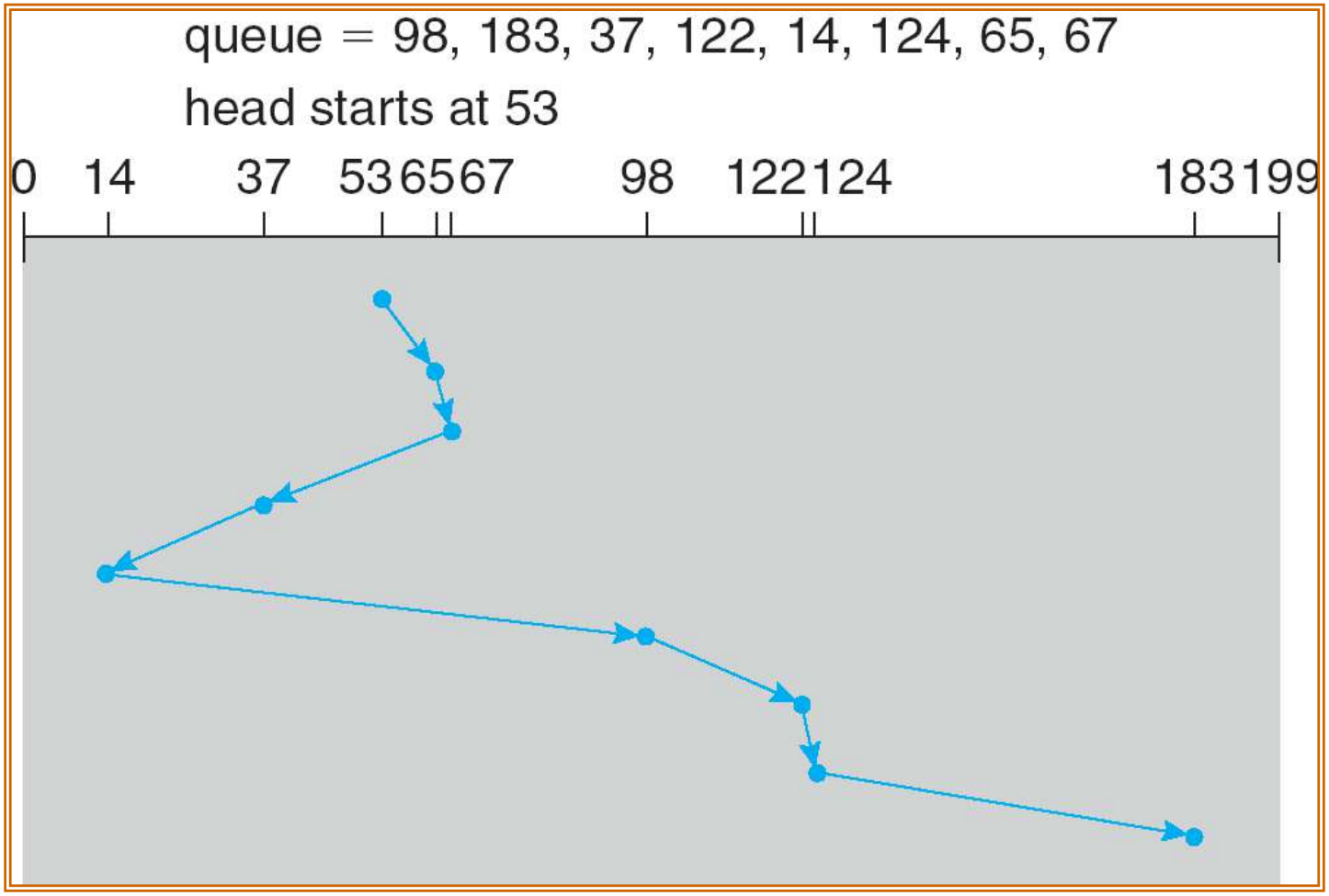
FCFS disk scheduling



2. Shortest-seek-time-first (SSTF) scheduling -

- Selects the request with the minimum seek time from the current head position.
- May cause starvation of some requests.
- Significantly better than FCFS but not optimal.

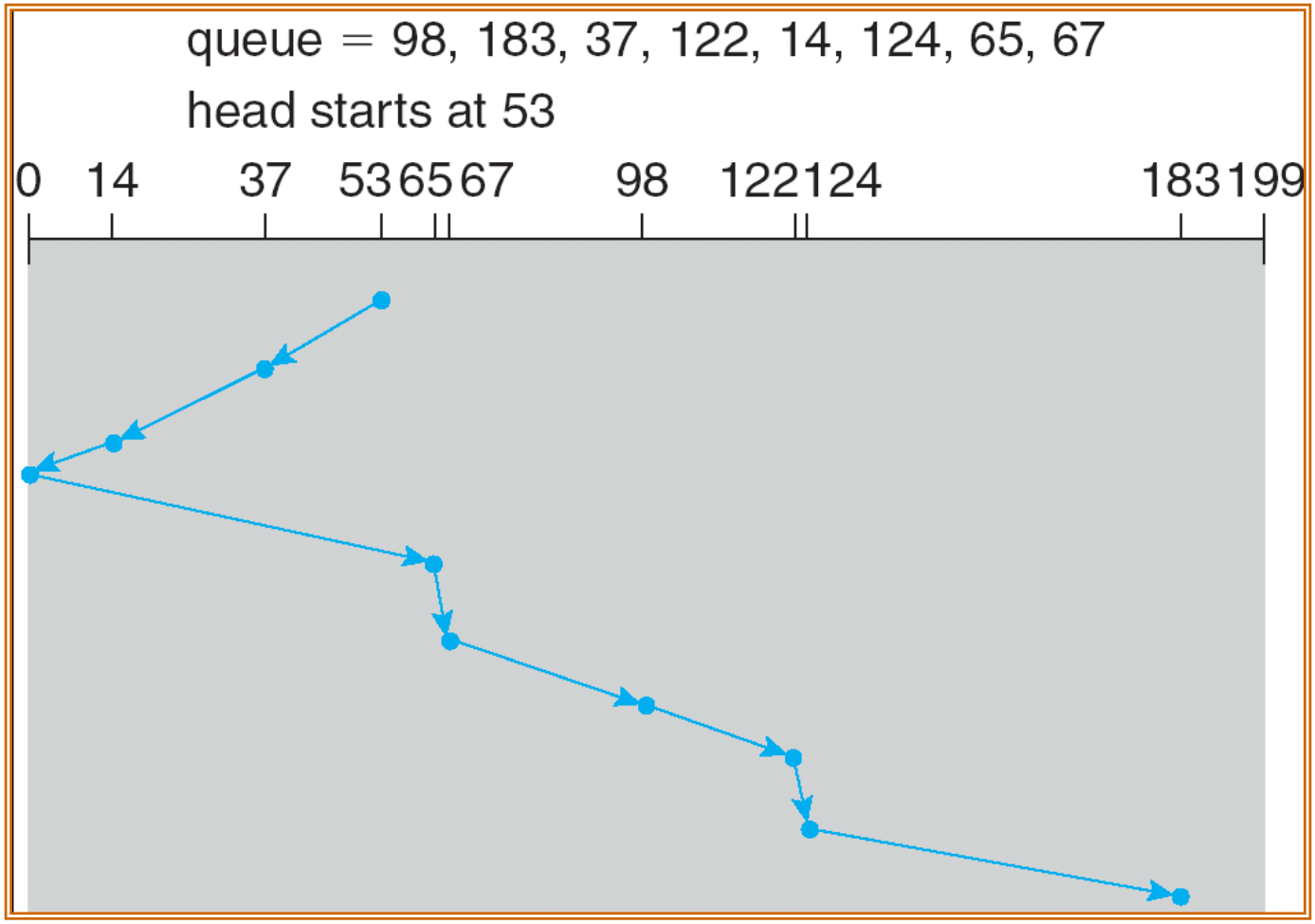
SSTF Scheduling



3. SCAN Scheduling/elevator algorithm

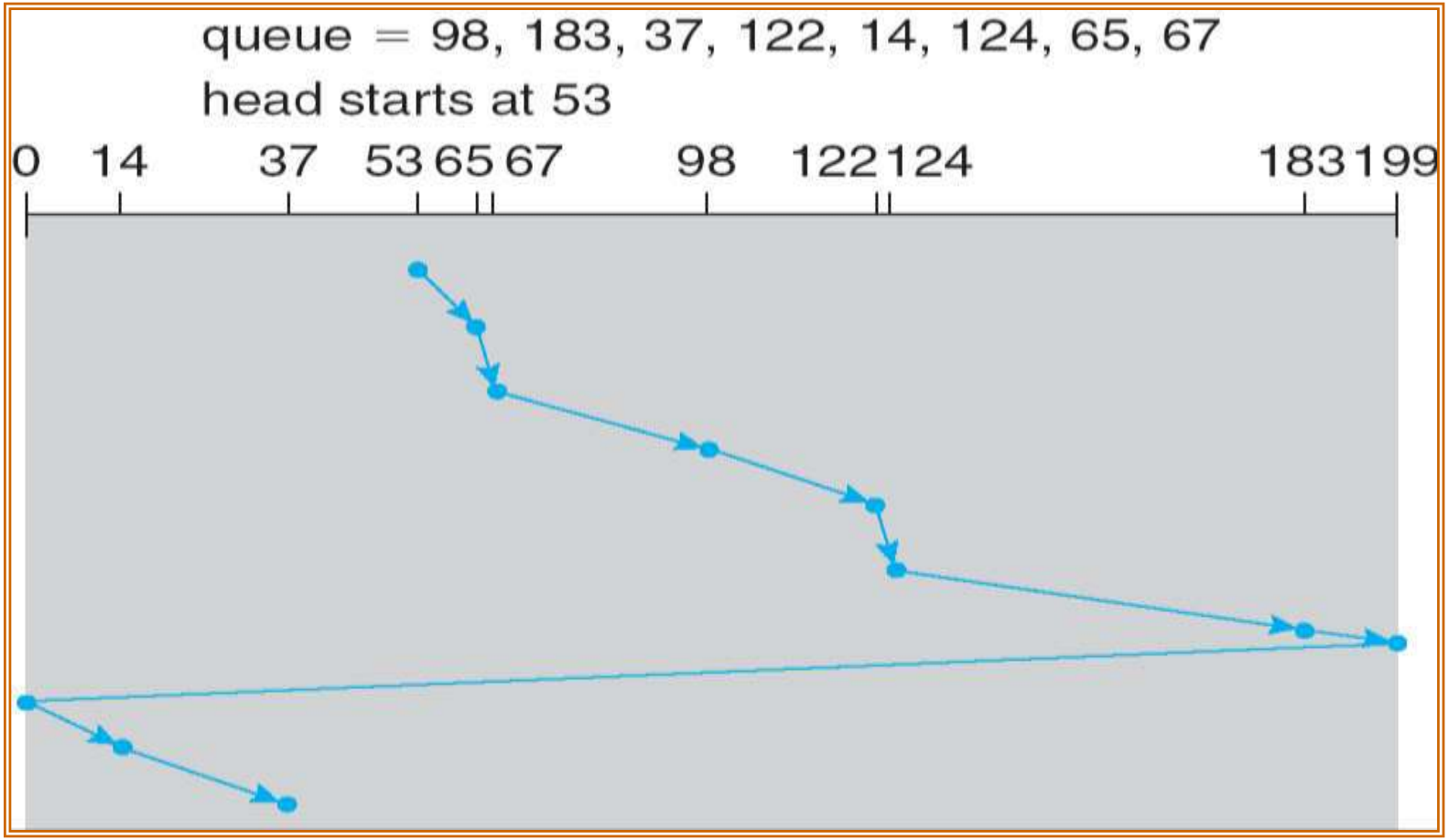
- The disk arm starts at one end of the disk and moves toward the other end servicing requests until it gets to the other end of the disk where the direction of head movement is reversed and servicing continues.
- Direction of head movement plus head's current position.

SCAN Scheduling



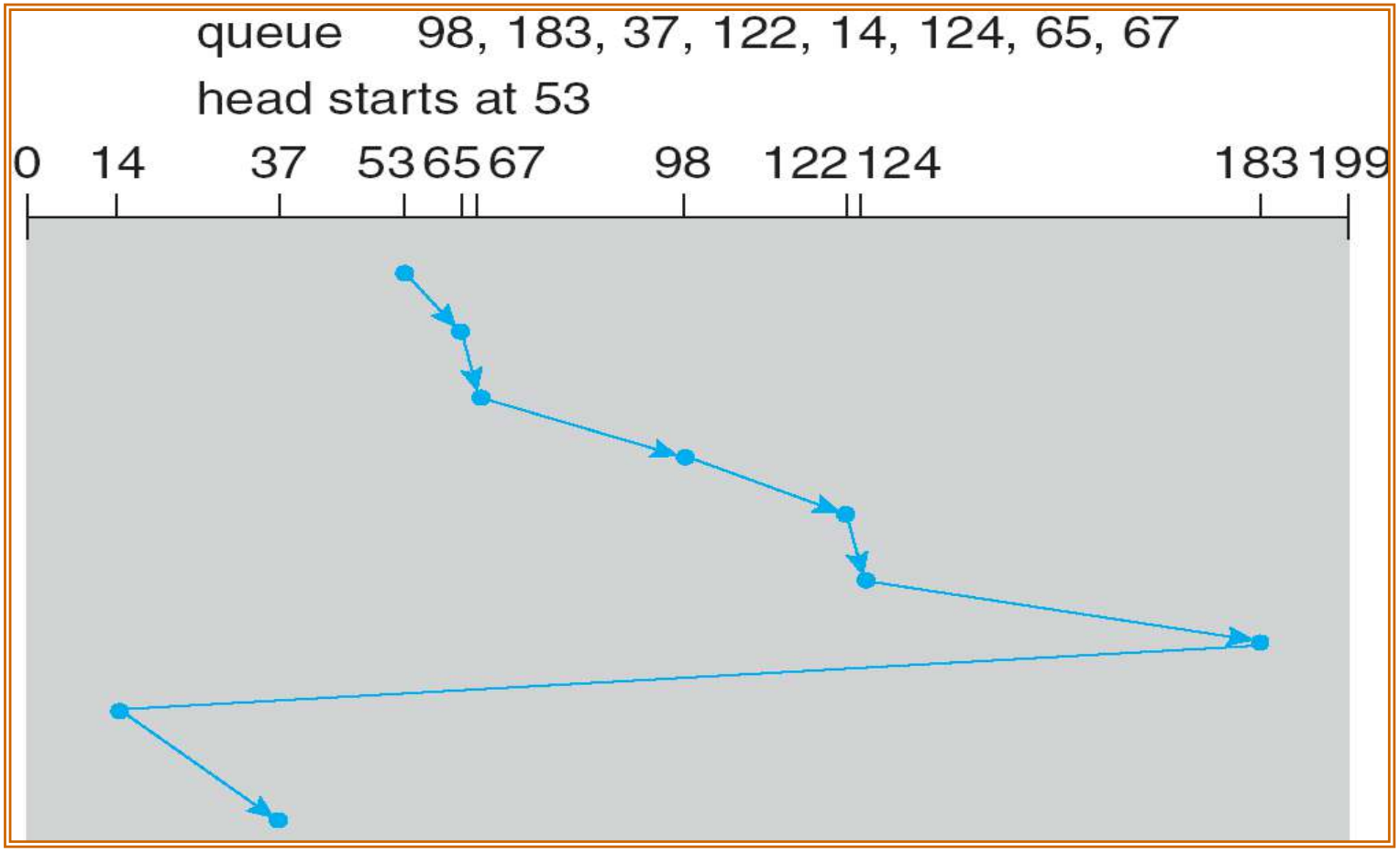
4. C-SCAN Scheduling -

- Provides a more uniform wait time.



5. LOOK Scheduling

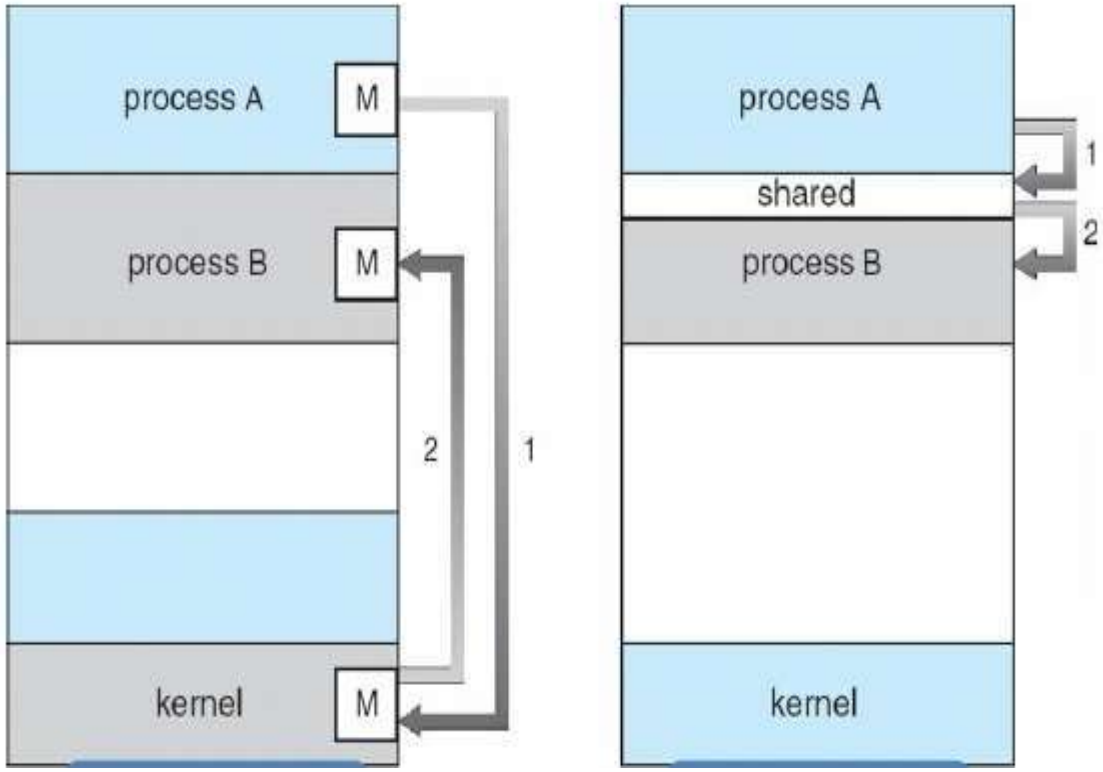
6. C-LOOK Scheduling



Interprocess Communication (IPC)

Introduction to IPC

- In order for the cooperating processes to exchange data and information.
- Two fundamental models – shared memory and message passing.
- In the former, processes exchange information by reading and writing data to the shared region.
- In the latter, messages exchanged between the cooperating processes.



Communication models : message passing (left) and shared memory (right).

Shared-Memory Systems

- A region of shared memory needs to be established by communicating processes.
- Producer-consumer problem.
- A compiler may produce assembly code, which is consumed by an assembler.
- Buffer of items that is filled by the producer and emptied by the consumer.
- Will reside in a region of memory shared by the cooperating processes.

- #define BUFFER_SIZE 10

  typedef struct {

  }item;

  item buffer [BUFFER_SIZE] ;

  int in = 0 ;

  int out = 0 ;
- Shared buffer implemented as a circular array with two logical pointers: in and out.
- in points to the next free position in the buffer.
- out points to the first full position in the buffer.

- The buffer is empty when in == out.
- The buffer is full when ((in + 1) % BUFFER_SIZE) == out.

```
item nextProduced;
while (true) {
while (((in + 1) % BUFFER-
SIZE) == out);
buffer[in] = nextProduced;
in = (in + 1) % BUFFER_SIZE;
```

Producer process

```
item nextConsumed;
while (true) {
while (in == out);
nextConsumed =
buffer[out];
out = (out + 1) %
BUFFER_SIZE;
}
```

Consumer process

Message-Passing Systems

- Does the function of allowing the processes to communicate with each other without the need to use shared variables.
- Particularly useful in a distributed environment where the communicating processes may reside on different computers connected by a network.
- Provision of at least two operations – send(message) and receive(message).
- Communication link.

Methods for logically implementing a link and send()/receive() operations

- Direct or indirect communication
- Synchronous or asynchronous communication
- Automatic or explicit buffering

Naming

- Direct communication – each process that wants to communicate must explicitly name the recipient or sender of the communication.
  - ✓send(P, message)—Send a message to process P.
  - ✓receive (Q, message)—Receive a message from process Q.
- Communication link properties –
  - ✓A link is established automatically between every pair of processes that want to communicate. The processes need to know only each other's identity to communicate.

- ✓ A link is associated with exactly two processes.
- ✓ Between each pair of processes, there exists exactly one link.

- Symmetry in addressing.
- Asymmetry in addressing - only the sender names the recipient, the recipient is not required to name the sender.
  - ✓ send(P, message) - Send a message to process P.
  - ✓ receive(id, message) - Receive a message from any process; the variable id is set to the name of the process with which communication has taken place.
- Disadvantage – limited modularity.

- Indirect communication – messages are sent to and received from mailboxes or ports.
- A mailbox can be viewed abstractly as an object into which messages can be placed by processes and from which messages can be removed.
- Each mailbox has a unique identification.
- A process can communicate with some other process via a number of different mailboxes.
- Two processes can communicate only if the processes have a shared mailbox.

- ✓send(A, message)—Send a message to mailbox A.
- ✓receive(A, message)—Receive a message from mailbox A.

- Communication link properties –
  - ✓A link is established between a pair of processes only if both members of the pair have a shared mailbox.
  - ✓A link may be associated with more than two processes.
  - ✓Between each pair of communicating processes, there may be a number of different links, with each link corresponding to one mailbox.

- Mailbox may be owned by a process or operating system.

Synchronization

- Message passing may be either blocking or nonblocking - also known as synchronous and asynchronous.
  - ✓ Blocking send : The sending process is blocked until the message is received by the receiving process or by the mailbox.
  - ✓ Nonblocking send : The sending process sends the message and resumes operation.
  - ✓ Blocking receive : The receiver blocks until a message is available.
  - ✓ Nonblocking receive : The receiver retrieves either a valid message or a null.

- When both send() and receive() are blocking, we have a rendezvous between the sender and the receiver.

Buffering

- messages exchanged by communicating processes reside in a temporary queue.
- Zero capacity (message system with no buffering).
- Bounded capacity.
- Unbounded capacity.