Heterogeneous Data Types including digital audio, animations and video

- Consume enormous space and bandwidth (JPEG-compressed movie can require over 2 GByte)

- Are delay-sensitive: when user plays-out or records a time dependent multimedia data object, the system must consume or produce at a constant data gap-free rate

➔ **A new multimedia file system is needed**

File System is the most visible part of an Operating System and consists of:

1. Files ○ : Program Codes

2. Directory Structure □: Organizes files and provides information, e.g.

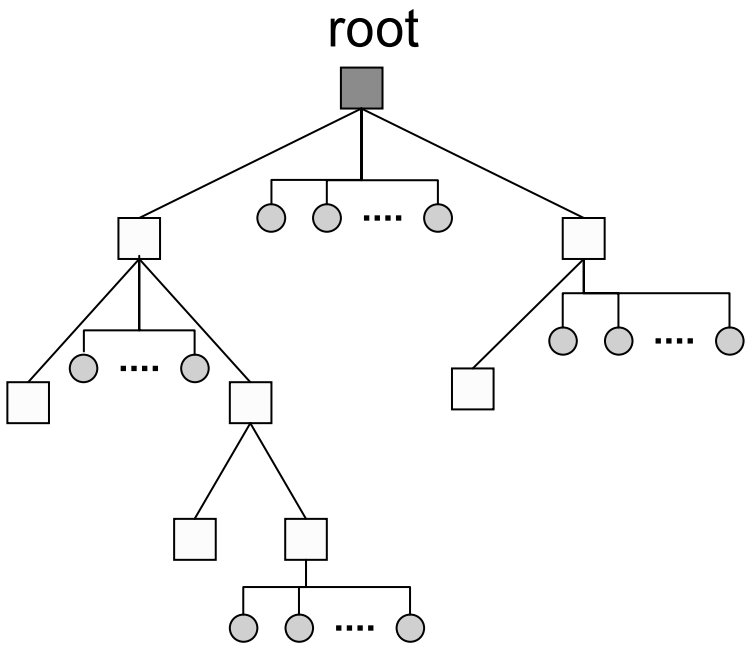
```
-rwxr-xr-x 1 root other 55160 Jul7 1995 gcc*
```

Traditional Files:

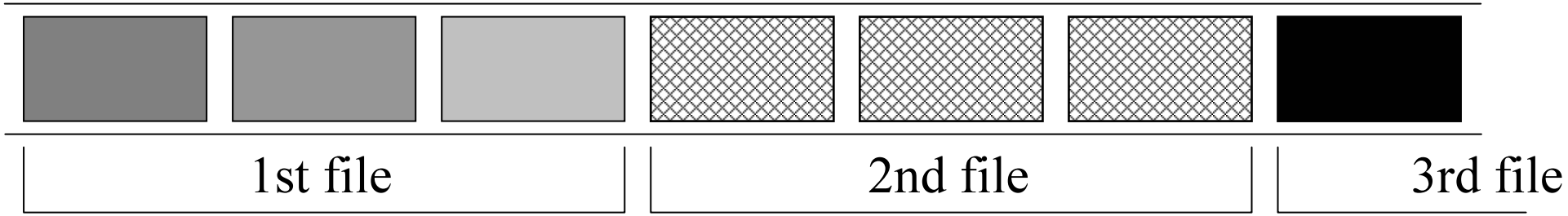
- executables of programs,
- numeric data,
- text,
- objects etc.

Goals:

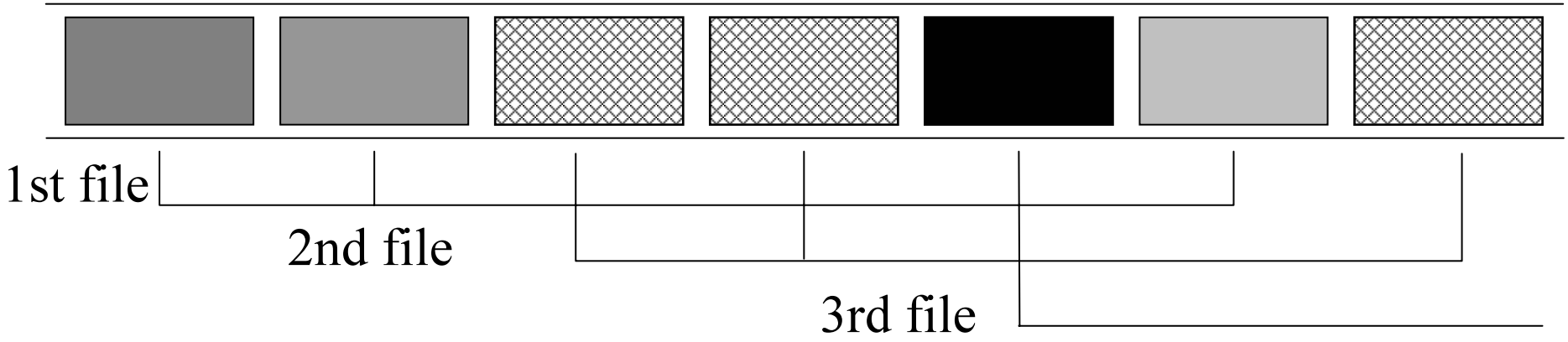
- Provide a comfortable interface for file access
- Make efficient use of storage media



Contiguous Placement



Non-contiguous Placement



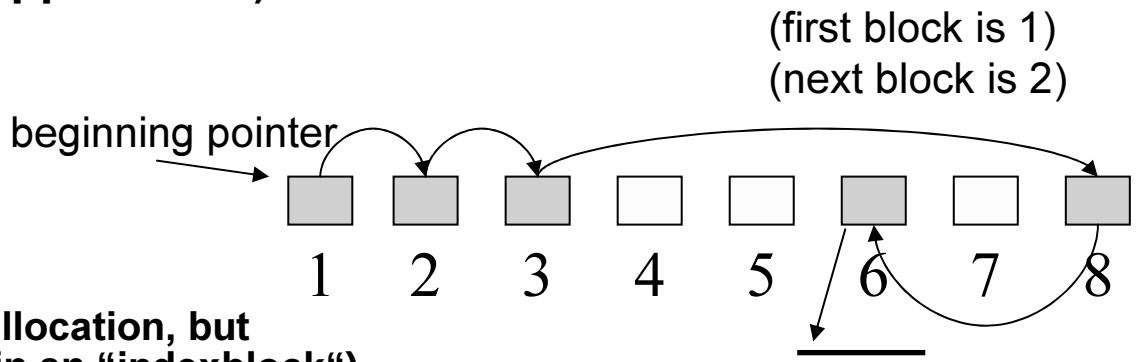
Sequential Storage:

- Disc access time for reading and writing is minimized
- On tape the only possible way
- Major disadvantage for file creation, deletion and size modification

Non-sequential Storage (two main approaches):

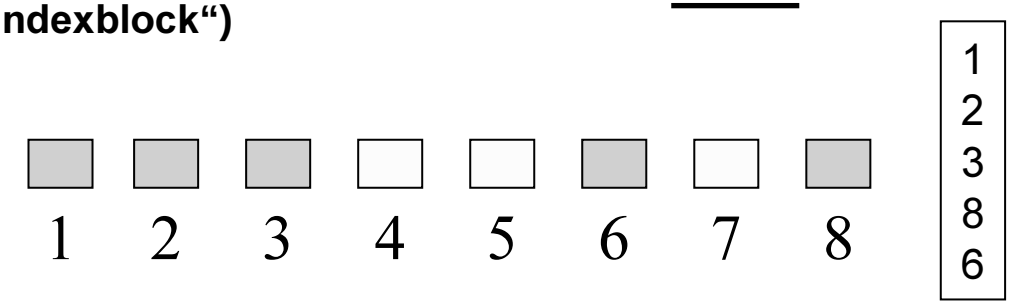
Linked Allocation:

- Fine for sequential access
- Random access is costly

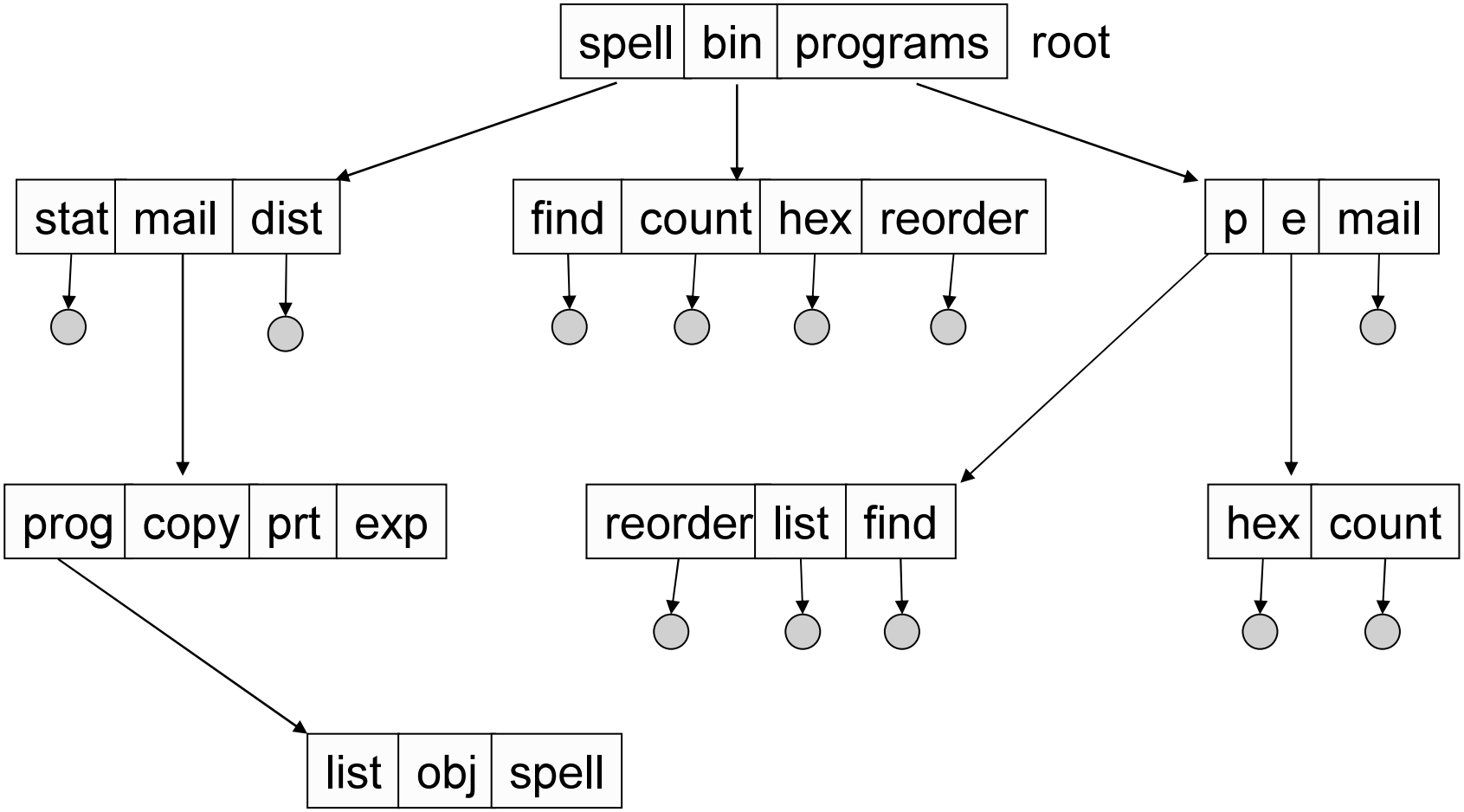


Indexed Allocation: **(similar to linked allocation, but "links" are stored in an "indexblock")**

- Complex
- Performance depends on the index structure and size of the file



Most of today´s operating systems provide tree-structured directories.



Disk access is slow and costly - major bottleneck

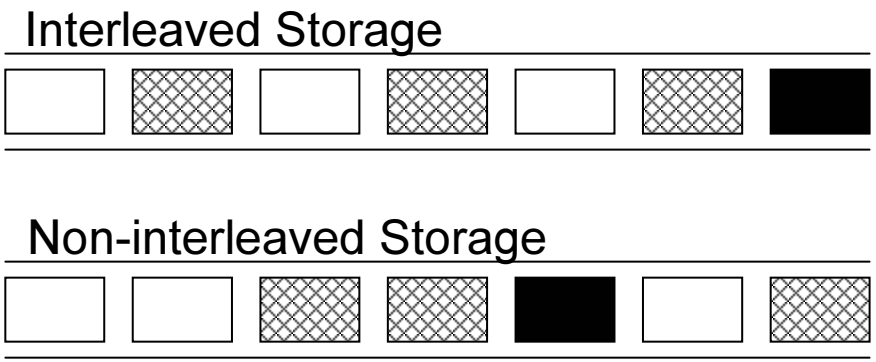
Techniques reducing overall disk access time:

Block caches

- Reduces the number of disk access

Reduce disk arm motion

- Reduces the time for one disk access

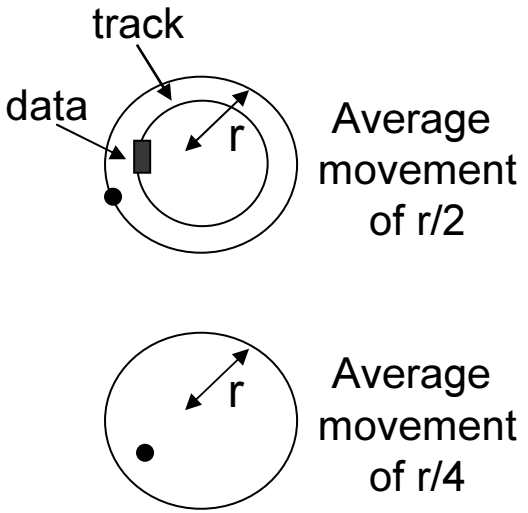


Place consecutive blocks in an interleaved manner.

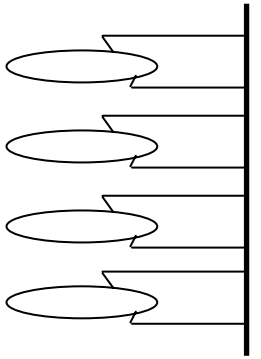
Mapping Tables are placed in the middle of the disk.

Tables and the corresponding blocks are placed on the same cylinder.

1. **head movement**
2. **wait until data appears under the head**

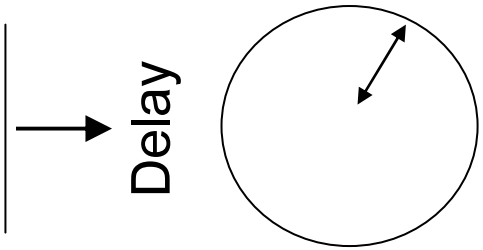


Heads may read in parallel



Total time to service a disk request consist of:

- Seek time, head positioning to appropriate track (diameter)
- Latency (rotation time), time to find the block in the track
- Actual data transfer time



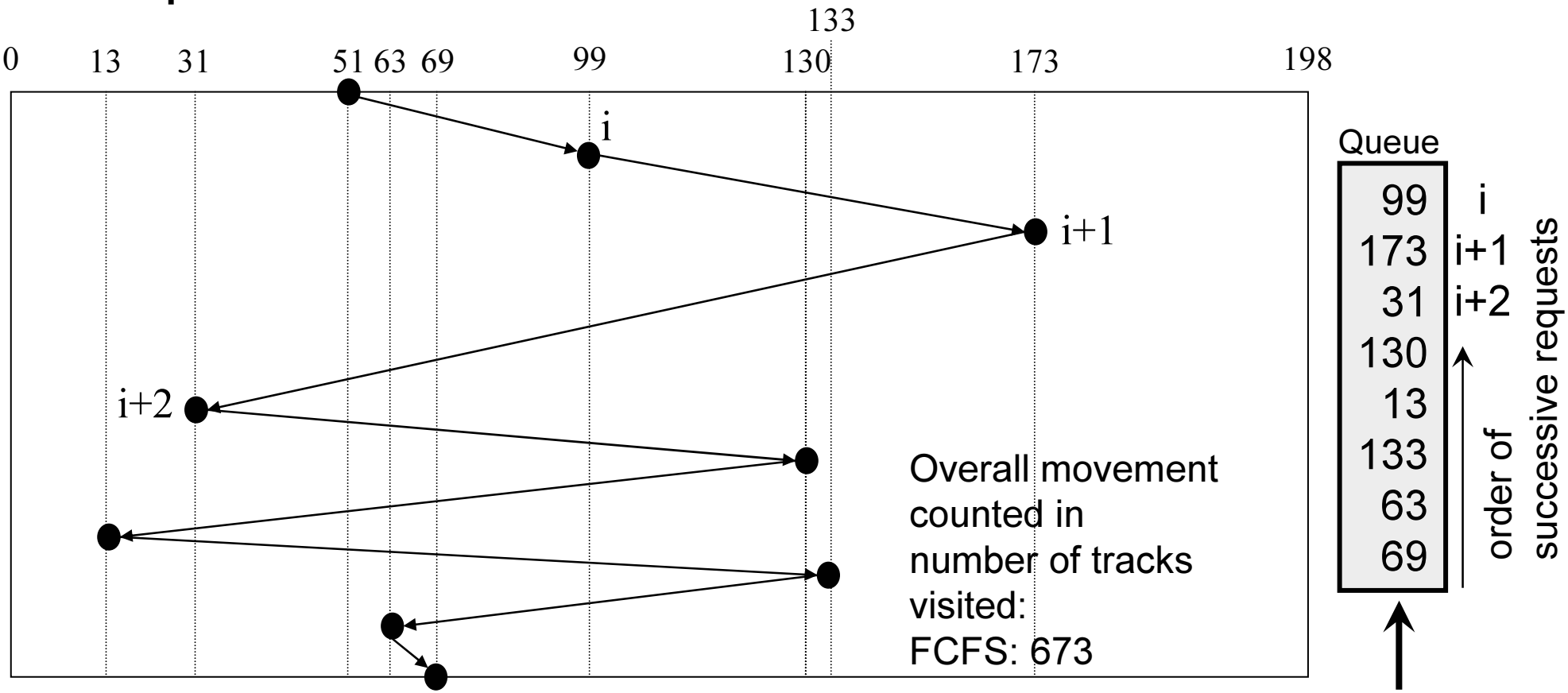
Technique to reduce delay:

- Seek operation ➔ Scheduling algorithms
- Latency ➔ File allocation methods

Sequential storage devices (e.g. tapes) have no scheduling problem.

At first we will consider strategies for minimizing the seek time,
i.e. for the positioning time of the head to the appropriate track.
Tracks are numbered 0,...,N-1. Here 0 is the innermost and N-1 the outermost track.

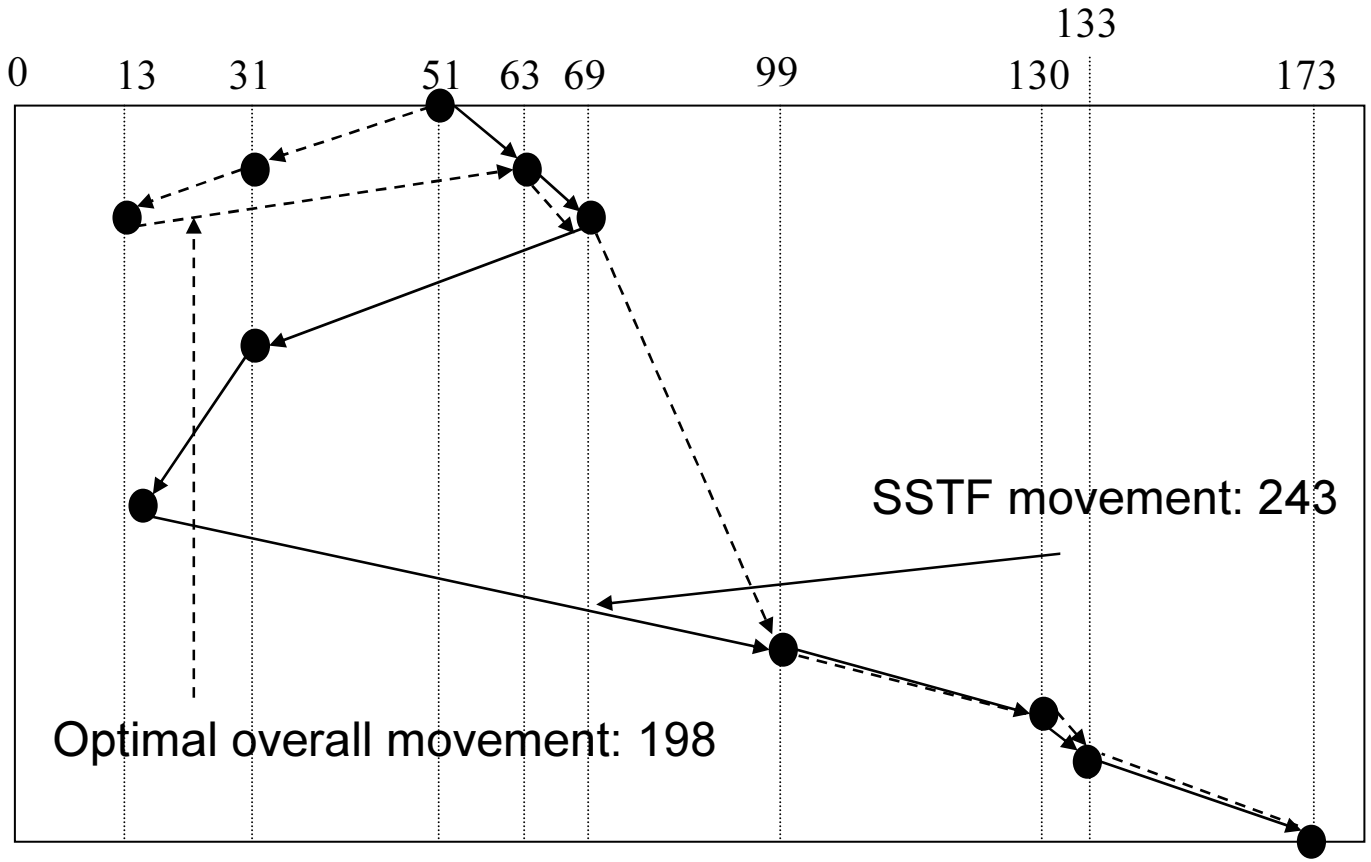
Serve requests in order of arrival



+ Easy to program
+ Fair algorithm
- Not optimal ➔ High average seek time

8.2 Disk Scheduling
Shortest-Seek-Time First (SSFT)

SSFT = Serve “nearest“ request



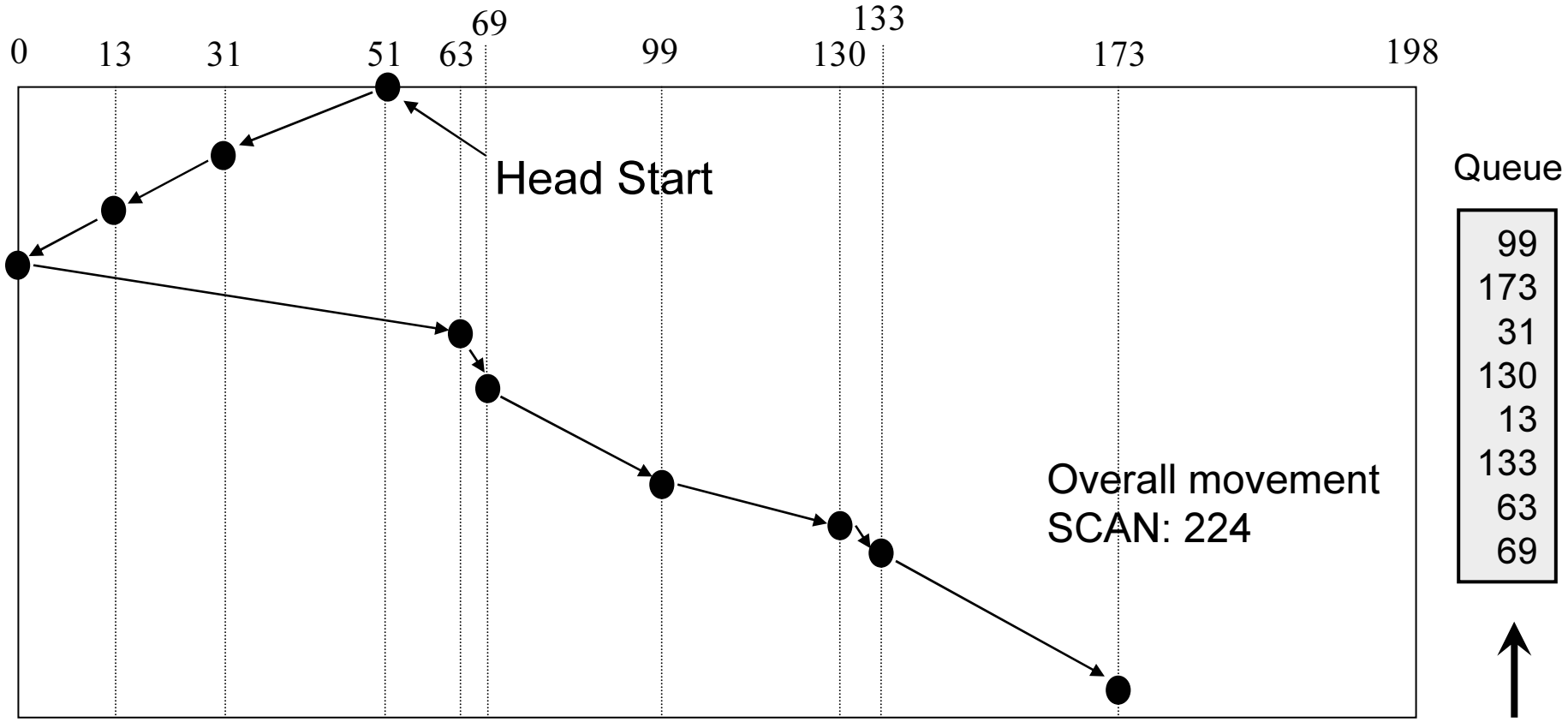
Queue

99
173
31
130
13
133
63
69

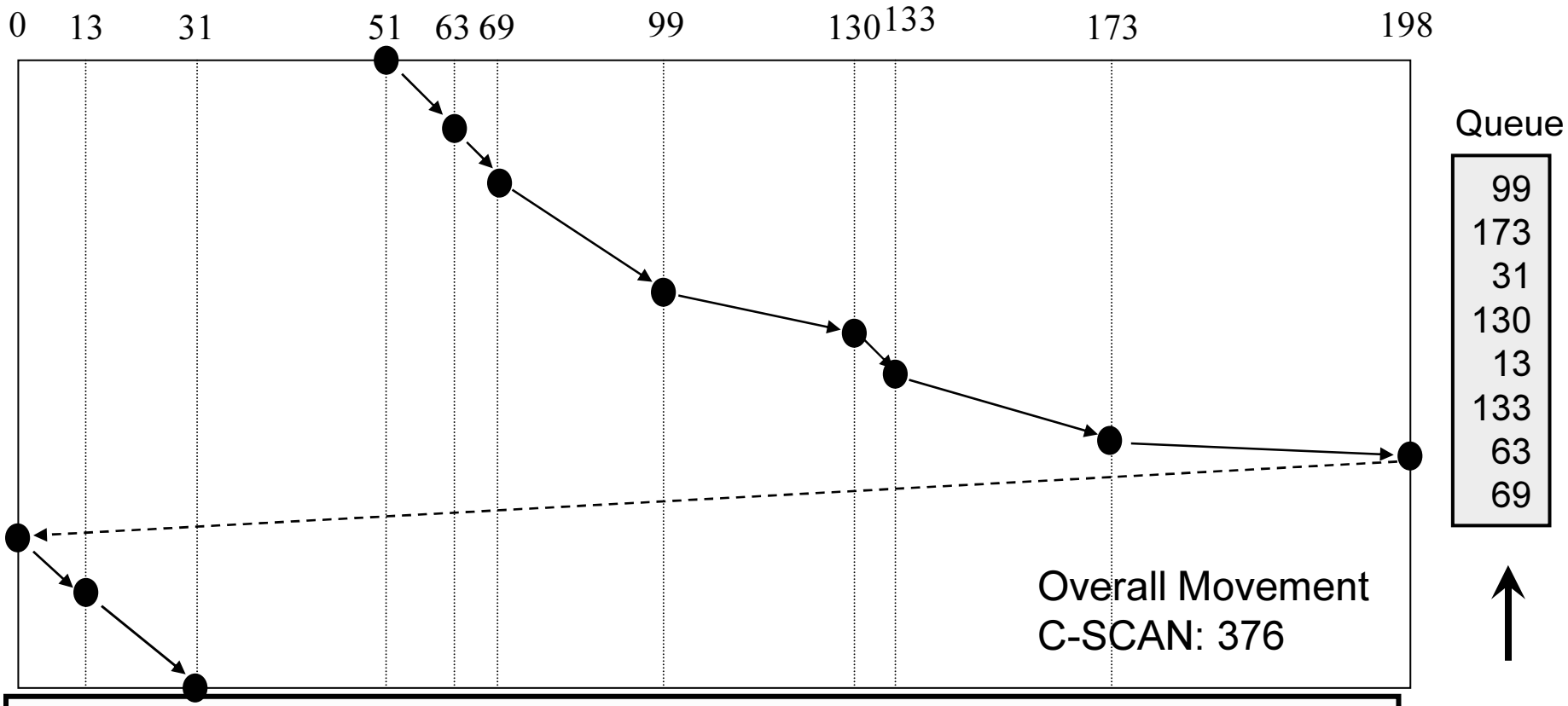
+ Substantial improvement over FCFS
- Still not optimal
- Starvation (of some tracks if there is always a track with shorter seek time available)

SCAN = serve requests in one direction; then reverse the movement.

Move from one end to the other, servicing each request on the way.



C-SCAN is similar to SCAN but returns immediately to the beginning if the end is reached;
One idle head movement from one edge to the other between two consecutive scans.



+ Fair service
- More uniform waiting time
- Performance not as good as SCAN
+ middle tracks don´t get a better service than edge tracks (such as with SCAN or with SSTF)

Requirements of continuous data:

Real Time Characteristic:

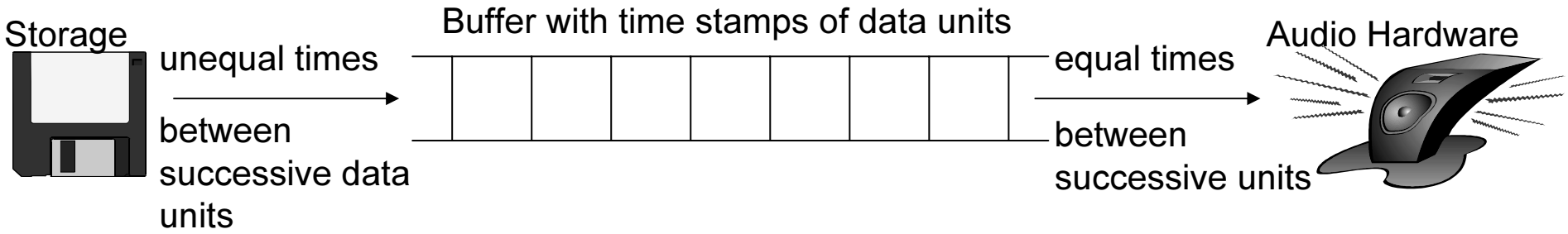
- Constant gap-free rate ➔ Additional buffers

File Size:

- Highly structured data units (e.g. video and associated audio)
  ➔ New organization policies of the data on disk
- Efficient usage of limited storage is necessary

Multiple data Stream:

- For example: retrieval of a movie requires the processing and synchronization of audio and video

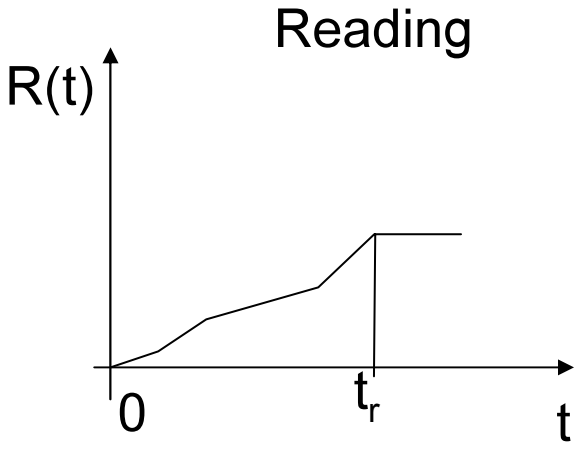


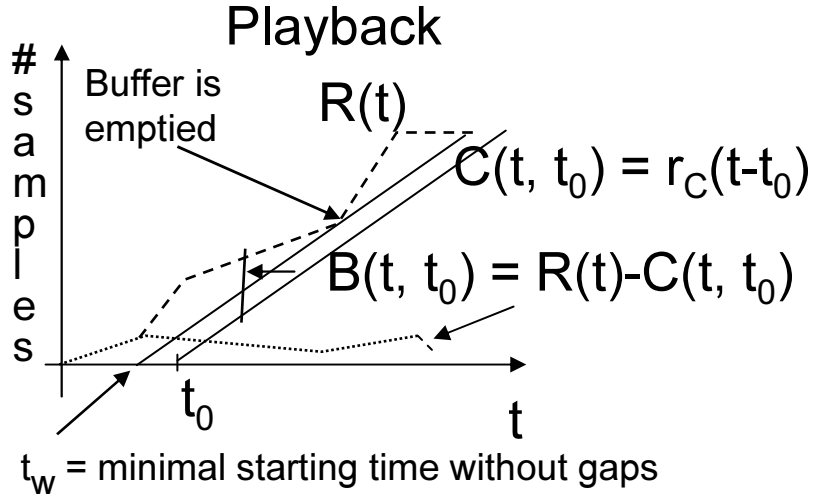
Rate of data retrieved from storage + Rate of data consumed by audio hardware

Determines buffer space requirements

With Voice over IP (VoIP) or with Internet radio the times between successive data can vary even more than with retrieval from conventional storage media. Thus in order to receive as many data of an audio stream the buffer space should be larger (approximately 5 seconds of playout time for Internet radio). Even then, some audio parts can arrive too late but the human ear is quite tolerant against losses (and Internet radio comes for free thus quality degradation is easily tolerated).

For VoIP (i.e. dialogue conversation) the buffer space (and the corresponding delay) should not be longer than 0,5 seconds (late audio parts must be dropped at the expense of quality degradation).





R(t) = Total number of samples read up to time t
t_r : Reading is completed
t_0 : Playback starts to use the digital to analog converter

$C(t, t_0)$ = Number of consumed samples up to time t.
r_C = Rate of consumed samples

Consumption order and physical placement are known ➔ Optimal for buffer requirements and throughput

$R(t) \geq C(t, t_0)$ for all t is required for a gap-free playback

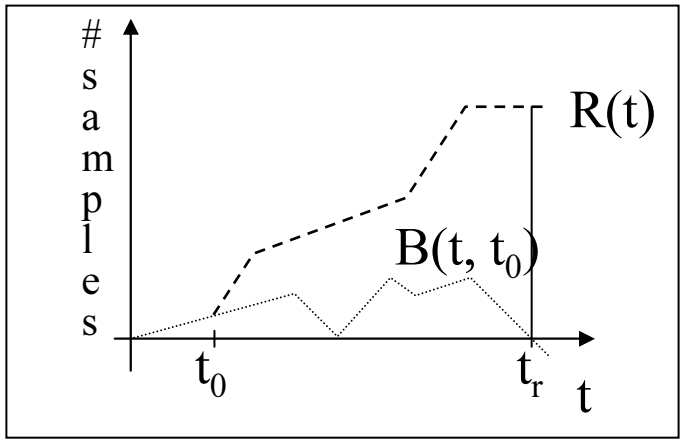
⇒ Buffer needed if playout is started at time t_0:
$B(t, t_0) := R(t) - C(t, t_0)$

- We want to have:
  - a playback to begin as early as possible
  - a small buffer size

Problem: Find minimum buffer size such that buffer never gets empty

“The earlier the first sample is played out, the less buffer requirement“

Theorem 1:
The solution using the minimum start time for playback also requires the least buffer space at any point in time.



PROOF:

Let t_w be a given solution of the minimum start time.

We have for any possible plackback without gaps:

$$B(t, y) = \begin{cases} R(t) & \text{if } t < y \\ R(t) - r_C (t - y) & \text{if } t \geq y \rightarrow R(t) - r_C t + r_C y \geq 0 \end{cases}$$

This must also hold for y = t_w and since t_w is the minimum starting time we have:

$r_C t_W \leq r_C y$ and $B(t, t_w) \leq B(t,y)$

Theorem 2:

The minimum value for t_0 is as follows: consider B(t,0)

1. If $B(t,0) \geq 0$ for $0 \leq t \leq t_r$ ➔ $t_0 = 0$ is the solution
2. $-m := \min(B(t,0))$ at time t_{min} ➔ $B(t_{min}, 0) = -m$
3. m is the number of missing buffers if we would begin immediately, i.e. at time t = 0, with the playout.

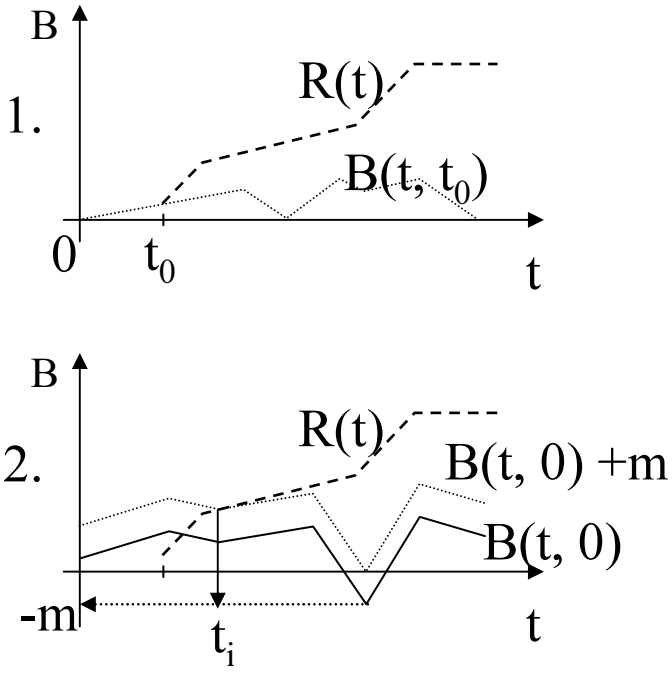
Then the start time t_i of a gap free playout is the intersection of $r_C t - m$ with the t-axis.

Or alternatively t_i is given by the t-abszissa value where R(t) and B(t, 0)+m intersect.

Proof:

1. If B(t,0) is a solution, then it must be optimal according to Theorem 1.
2. $r_C t - m$ is the highest (i.e. earliest starting) linear curve which is below R(t) for all t.

   At the intersection point of R(t) and B(t, 0)+m we have the necessary buffers, namly m, for the first time available in order to make a gap-free playout.



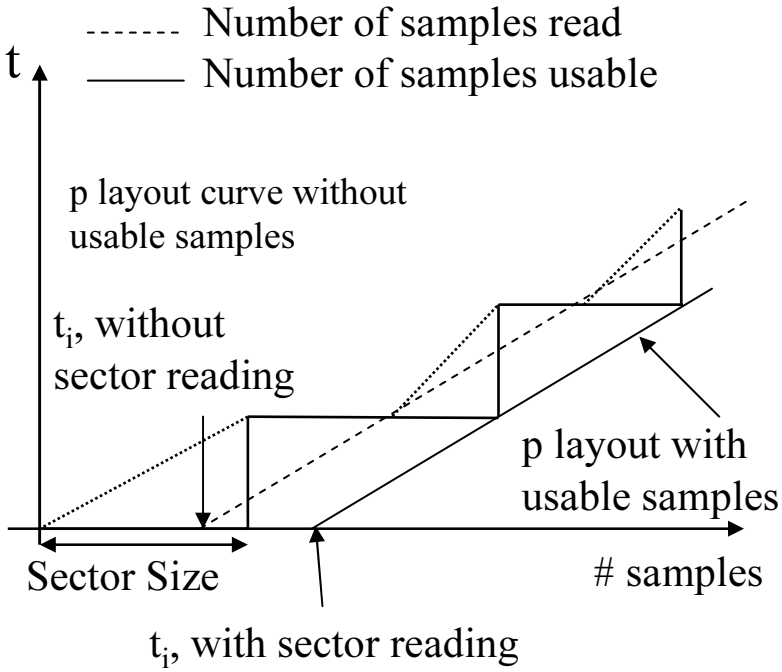
Usually we read samples in sector units of e.g. 512 bytes.

A sector contains various in addition to the data various other information such as error correction codes. Thus the data of the sector is valid only when all the bytes of it have been received.

As a consequence R(t) is a staircase function.

The minimum playout curve is the parallel to $r_C t$ which

Meets the staircase only at one (or more) lower edges.



Types of Storage Devices:

- Tapes are inadequate for multimedia systems
- Conventional magnetic storage devices are not sufficient (may not be up to date)
- CD-ROMs are often used (defined in ISO 9660)

Discs can be classified based on:

- Storage Method
  - Re-writable (magnetic and optical)
  - Write-once (Write Once Read Many: WORM)
- Physical medium
  - Magnetic disk (better seek time)
  - Optical disk

Provide a constant and timely retrieval of data:

- Organization of files on disk is optimized
- Admission control is performed for different service classes
- Special disk scheduling algorithms and sufficient buffers

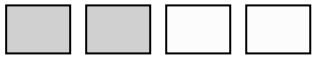
Problem: restricted transfer rate

Management of disk storage

- Optimal placing of data blocks on disk
- Using multiple disks

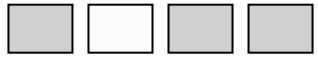
- Storage hierarchies should be possible
- Adding tertiary storage should be possible

Contiguous Placement

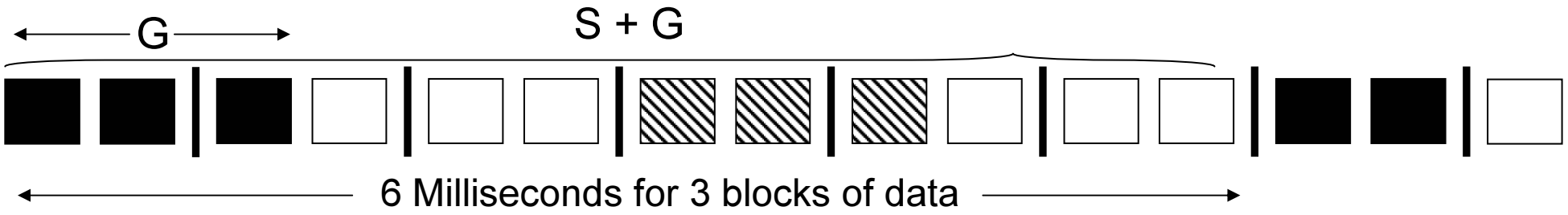


- Easy to implement / But subject to fragmentation
- Overhead by insertion and deletion
- Easy to read
  ➔ Suitable for read only systems

Scattered Placement



- Avoids fragmentation
- Avoids copying overhead
- Seek operation in worst case for each block
  ➔ Additional buffering is needed



➔ play back rate: 0.5 ms/block

Intermediate approach:

constrained placement “bounds the separation between successive media blocks“

Parameters:

the size of media block (granularity parameter G)
blocks: separation between successive blocks (scattering parameter S)

Continuity requirement:

r_D data transfer rate from disk
r_C playback rate

$$\underbrace{\frac{S+G}{r_D}}_{\text{playback duration}} \leq \frac{\overbrace{G}}{r_C}$$

i.e. time to skip over a gap and to read the next media block is smaller than or equal to the duration of the playback

e.g.
G = 3
r_D = 2
r_C = 0,5
(S+G)/2 ≤ G/0.5
S+G ≤ 12
S ≤ 9

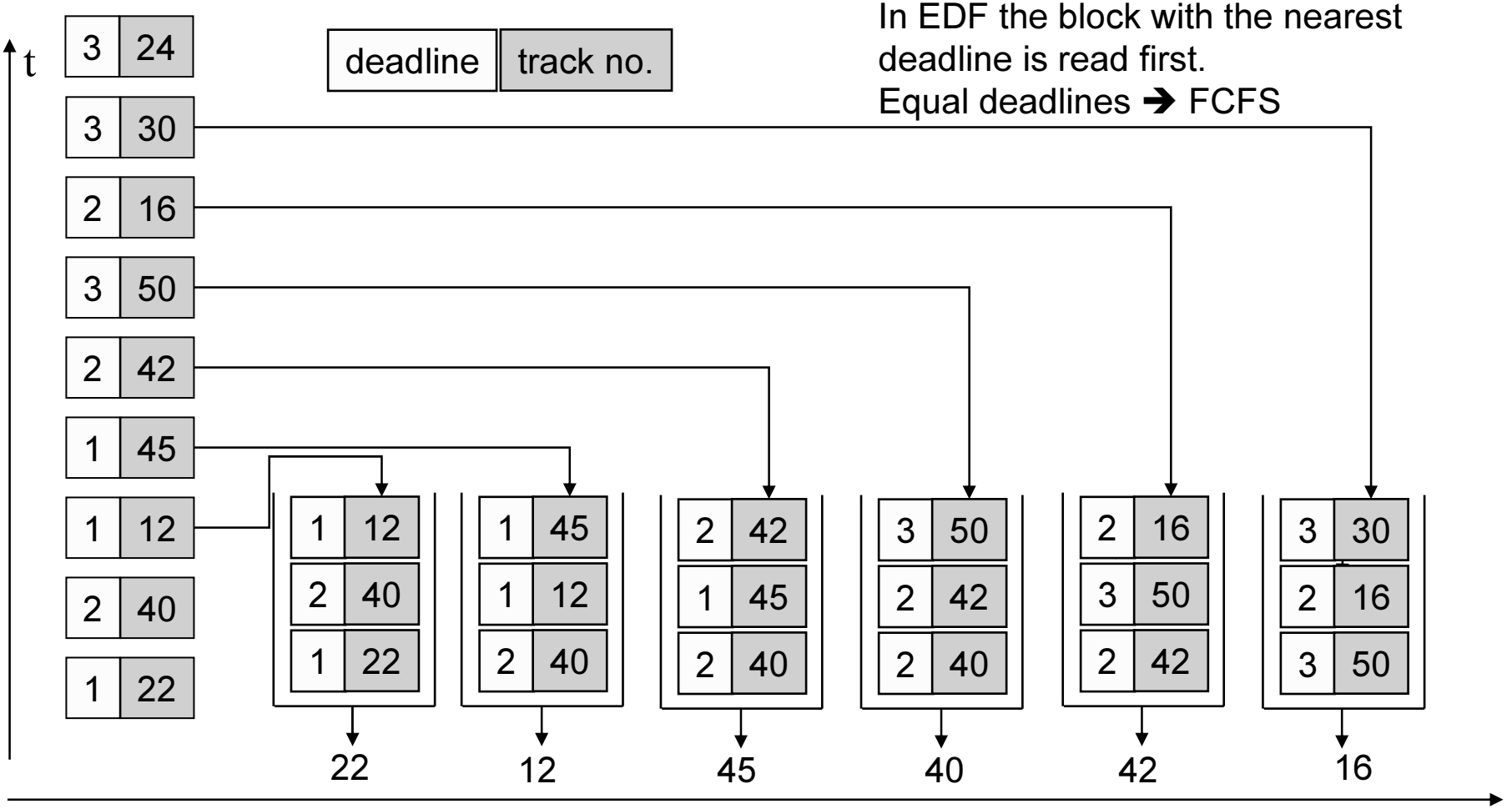
Goals in Traditional File Systems:

- Reduce cost of seek time
- Achieve fair throughput
- Provide fair disk access
- Standard scheduling algorithms are: FCFS, SSTF, SCAN, C-SCAN

Goals in Multimedia File Systems are different:

- Meet deadlines of all time-critical tasks
- Keep the necessary buffer space requirements low
- Find balance between time constraints and efficiency

8.4 Disk Scheduling
Earliest Deadline First (EDF)



Poor throughput due to excessive seek time. Only deadlines are taken into account, but not track number. Very similar to FCFS: inefficient. Does not reflect the geographical position of tracks.

SCAN-EDF is a combination of:

- Deadline scheduling (as in EDF earlier deadlines are served first).
- Scanning (tasks with same deadline are served according to the actual scan direction).

Problem: SCAN (i.e. use of scanning directions for tie break among equal deadlines) does not make much sense if too many different deadlines exist.

Thus:

It has to be enforced that many requests have the same deadline.

In order to do so, all requests are grouped in a few groups which can be scanned together.

We require that deadlines D_i are multiples of a common period p ➔ $D_i \in \{1, 2, 3, ...\}$.

Then deadlines with the same period can be grouped and served together by SCAN.

Implementation of SCAN-EDF by „Perturbation of deadlines“ (in order to apply EDF).

Let D_i the deadline of task i and N_i be the track number ($0 \leq N_i < N_{max}$, e.g. N_{max} =100)

Assume that $D_i \in N$.

Modify D_i towards D_i' (D_i'= perturbed deadline)

$D_i' = D_i + f(N_i)$.

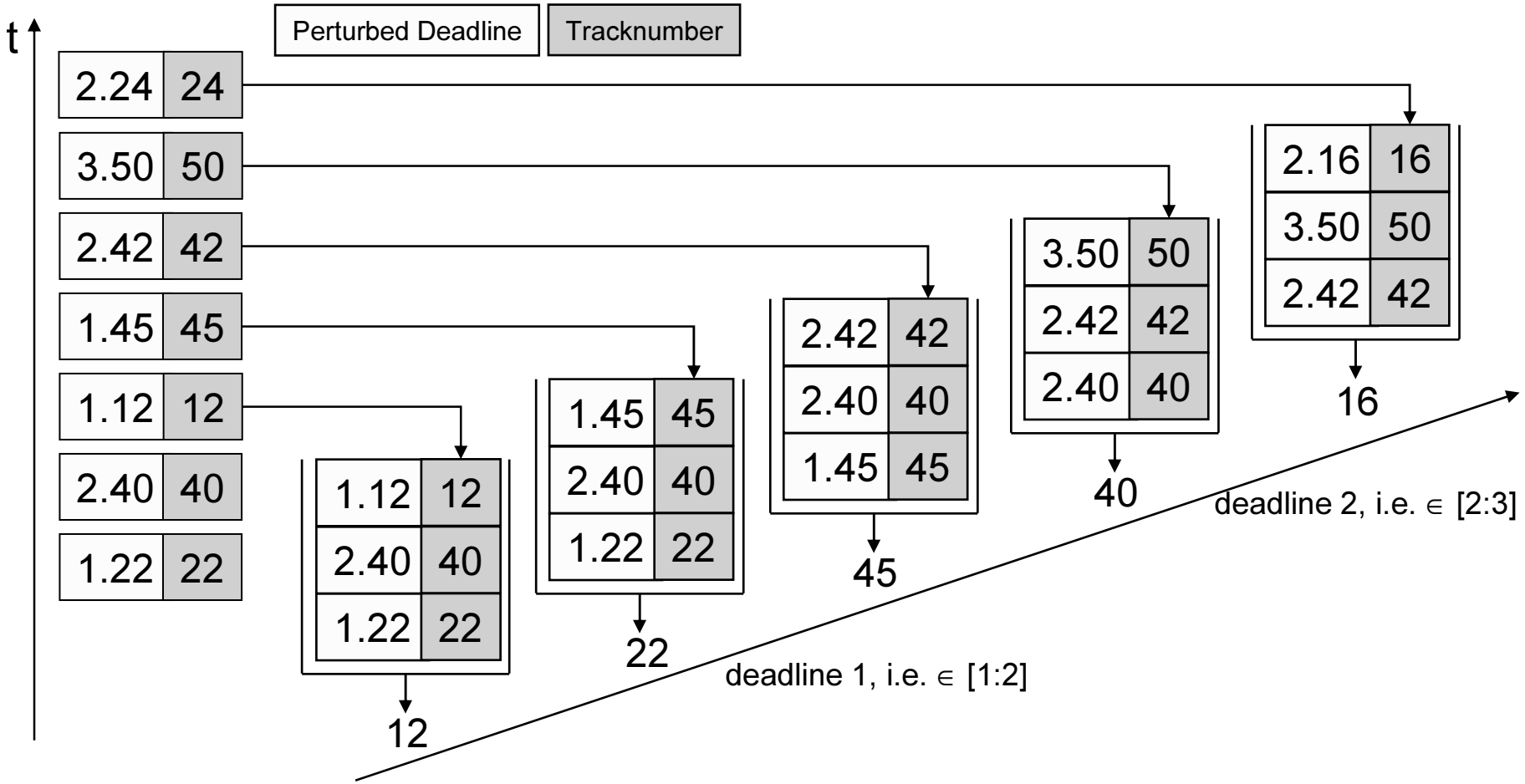
$f(N_i)$ converts the track number of i into a small perturbation of the deadline such that for equal deadlines the scanning is automatically applied.

If we choose (for example)

$$f(N_i) = \frac{N_i}{N_{\max}}$$
$$\Rightarrow 0 \leq f(N_i) < 1$$

Thus if the deadline for a task on track 42 is equal to 3 then the perturbed deadline is $3 + \frac{42}{100} = 3{,}42$

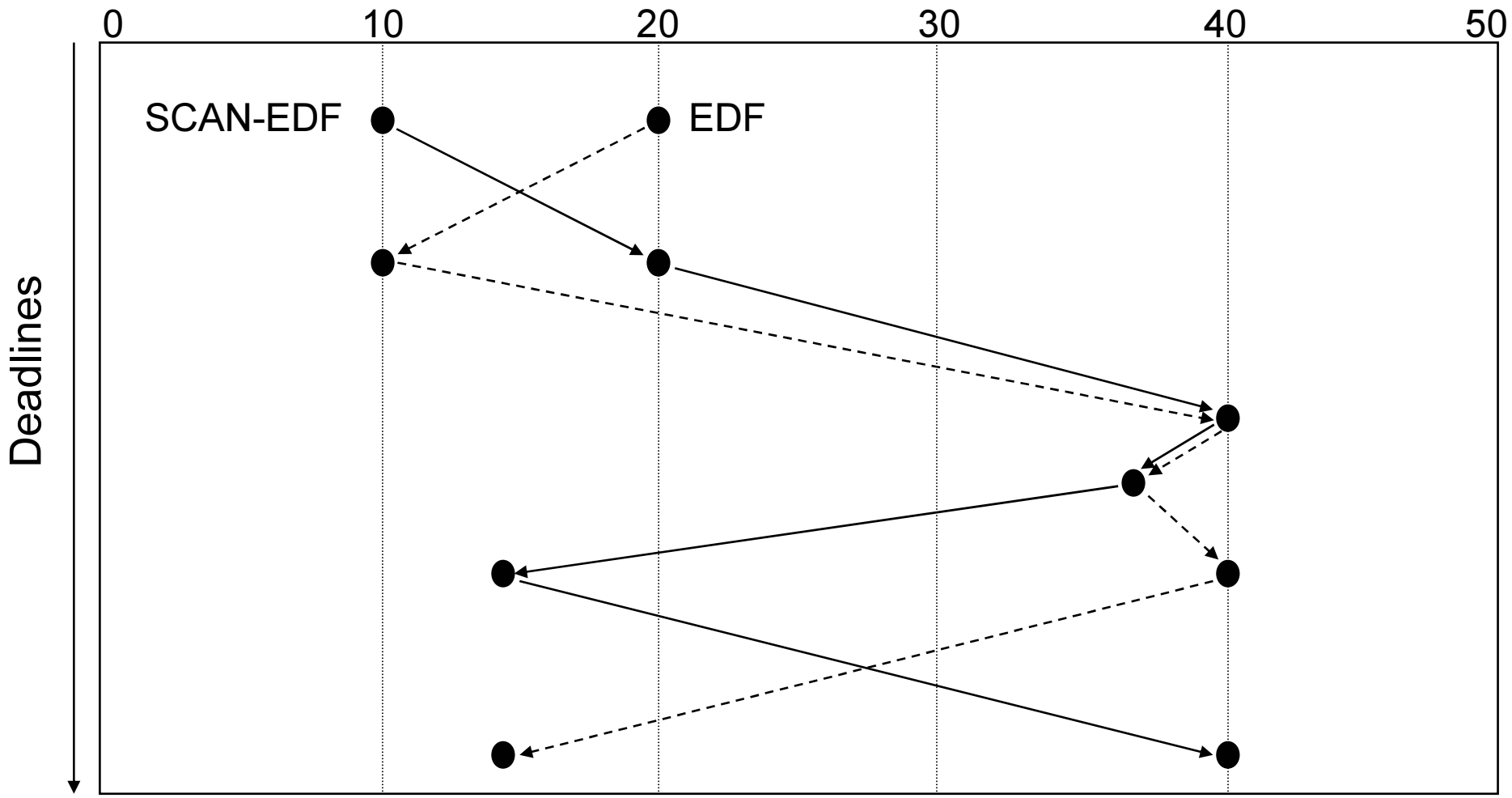
This deadline is given to the task at arrival time.



Among the same deadline SCAN is applied
Request with the earliest deadline is served
Sensible only for a large number of requests

➔

Optimization only applies for requests with the same deadline.
Increase this probability by grouping the requests.


0
10
20
30
40
50
SCAN-EDF
EDF
Deadlines

A small variation of "deadline perturbation"

The actual deadline given to the task is refined by:

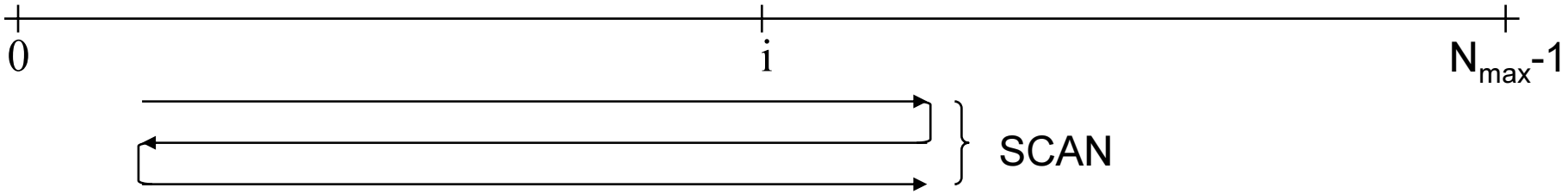
- taking account the actual movement of the head at arrival time (i.e. upwards from 0 to N_{max}-1 or downwards from N_{max}-1 to 0)
- and considering the actual position, N, of the head.

The perturbed deadline for a task which resists on track N_i is given by: $D_i' = D_i + f(N_i)$ where:

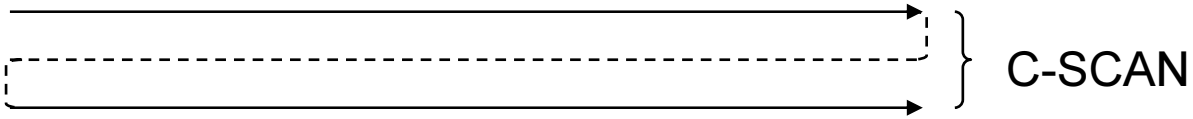
$$f(N_i) = \begin{cases} \dfrac{N_i - N}{N_{\max}} & \text{if } N_i \geq N \text{ and "head moves upwards"} \\ \dfrac{N_{\max} - N_i}{N_{\max}} & \text{if } N_i < N \text{ and "head moves upwards"} \\ \dfrac{N_i}{N_{\max}} & \text{if } N_i > N \text{ and "head moves downwards"} \\ \dfrac{N - N_i}{N_{\max}} & \text{if } N_i \leq N \text{ and "head moves downwards"} \end{cases}$$

This allows to serve new requests as soon as possible.

SCAN: serves request in ascending order of tracks, then in descending order, then in ascending order



C-SCAN: serves in ascending order only (“circular scan“)



EDF (Earliest Deadline First)

SCAN-EDF:

1. Groups requests in few deadline classes

(in order to have many requests for a given deadline)

2. Serves according to EDF (i.e. smallest deadline first) by using SCAN or

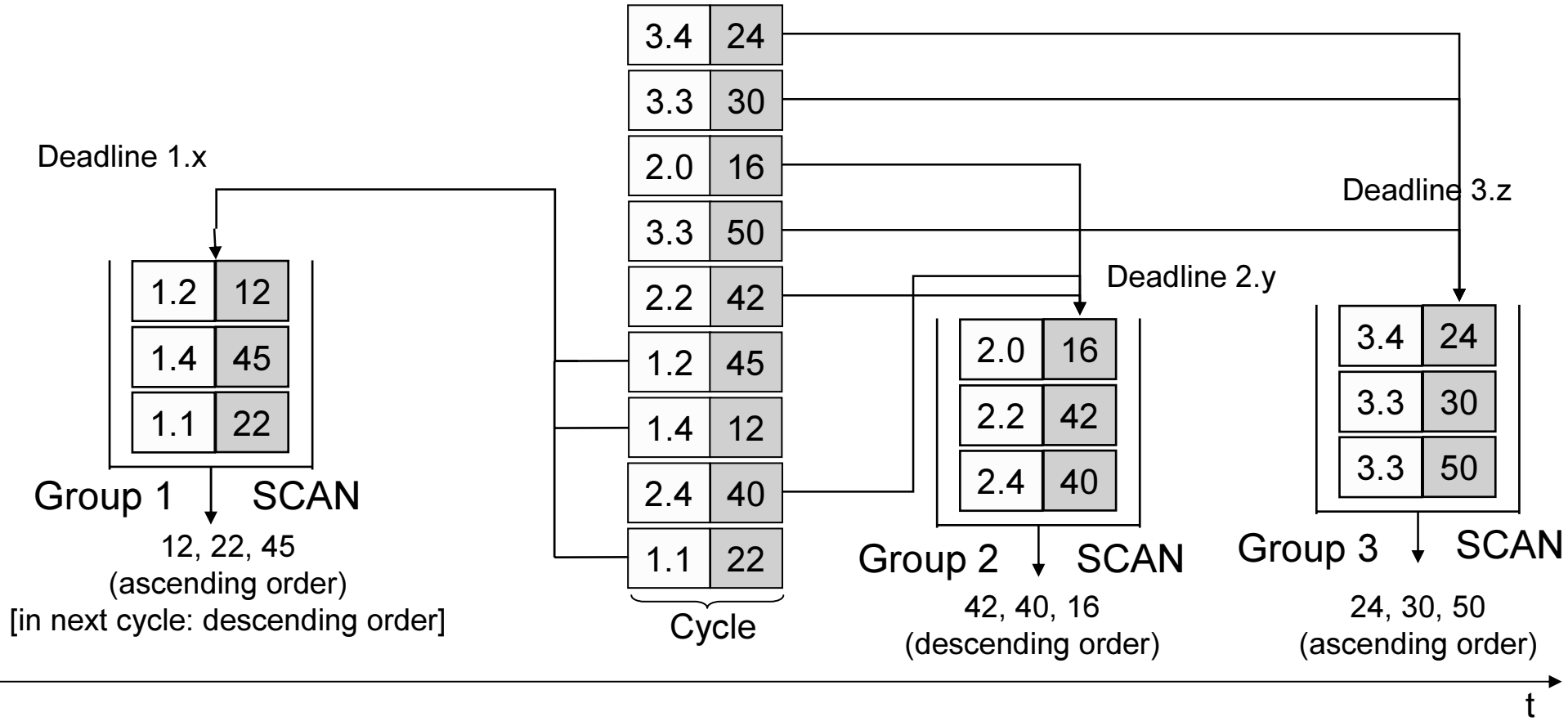
C-SCAN for members of a given deadline group.

For implementation: use Deadline perturbation

$$D_i \in \mathbb{N} \rightarrow D_i' = D_i + f(N_i)$$

where N_i = track number; $0 \leq f(N_i) < 1$

8.4 Disk Scheduling
Group Sweeping Scheduling (GSS)



Requests are served in cycles in a round-robin manner.
In one cycle requests are divided into groups. A group is served according to SCAN.
Serviec in a group may be in ascending or in descending order depending on the other groups.
Thus a smoothing buffer may be needed (to assure continuity).

A particular stream can be

- the first one in its group in a given cycle
- and the last one in its group in the next cycle

This happens if the scan order is reversed, i.e. if we have an odd number of groups.

Thus we need a smoothing buffer in order to achieve continuity of playout.

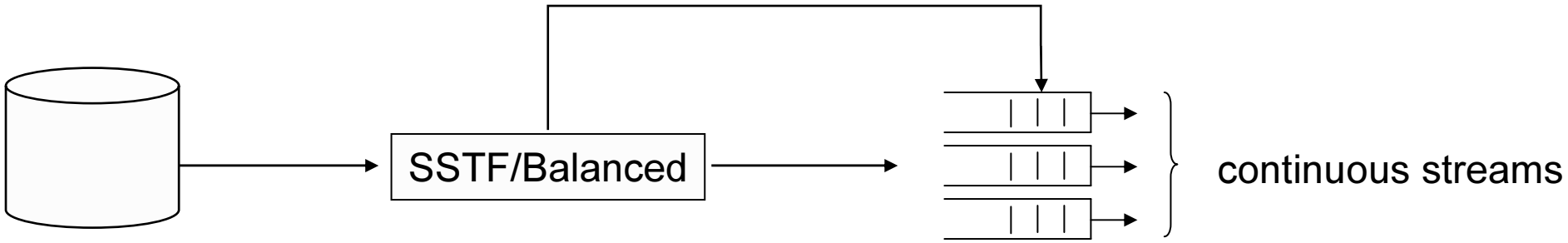
The „mixed strategy“ is a compromise between

- Shortest seek (“greedy“)
- Balanced strategy.

Data retrieved from disk are placed into buffers. Different queues are used for different data streams.

“Shortest seek“ serves the stream whose data block is closest.

“Balanced“ serves the stream which has the lowest utilization of buffers (since this stream risks to run out of data).



Filling Status of Buffers indicate when to switch from SSTF to "Balanced" and vice versa.

"Urgency" criterion:

$$\text{Urgency} = \sum_{(\text{all streams } i)} \frac{1}{\text{Fullness } i}$$

Fullness *i* = small ➔ Urgency = high

➔ Balanced strategy should be used

Continuous data are characterized by consecutive, time-dependent logical data units.

Basic data unit:

- Video - frame (single video image)
- Audio - sample (amplitude of the analog audio signal)

The design of data structure is guided by two requirements:

- Time continuum of media: Media units convey their meaning only when they are presented continuously.
- Synchronization between media: Temporal coordination of different media components is necessary.

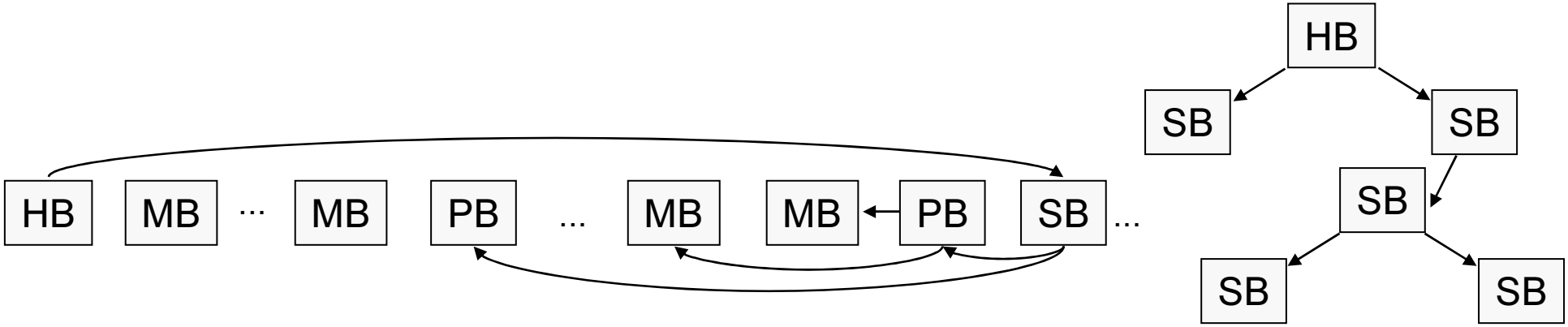
Continuity Requirement ➔ **Define a continuously recorded sequence of media units (video, audio or both) as a Strand. Strands must be partitioned into blocks and stored on disk.**

For providing direct access to any of the media blocks of the strand a hierarchical index structure is used.

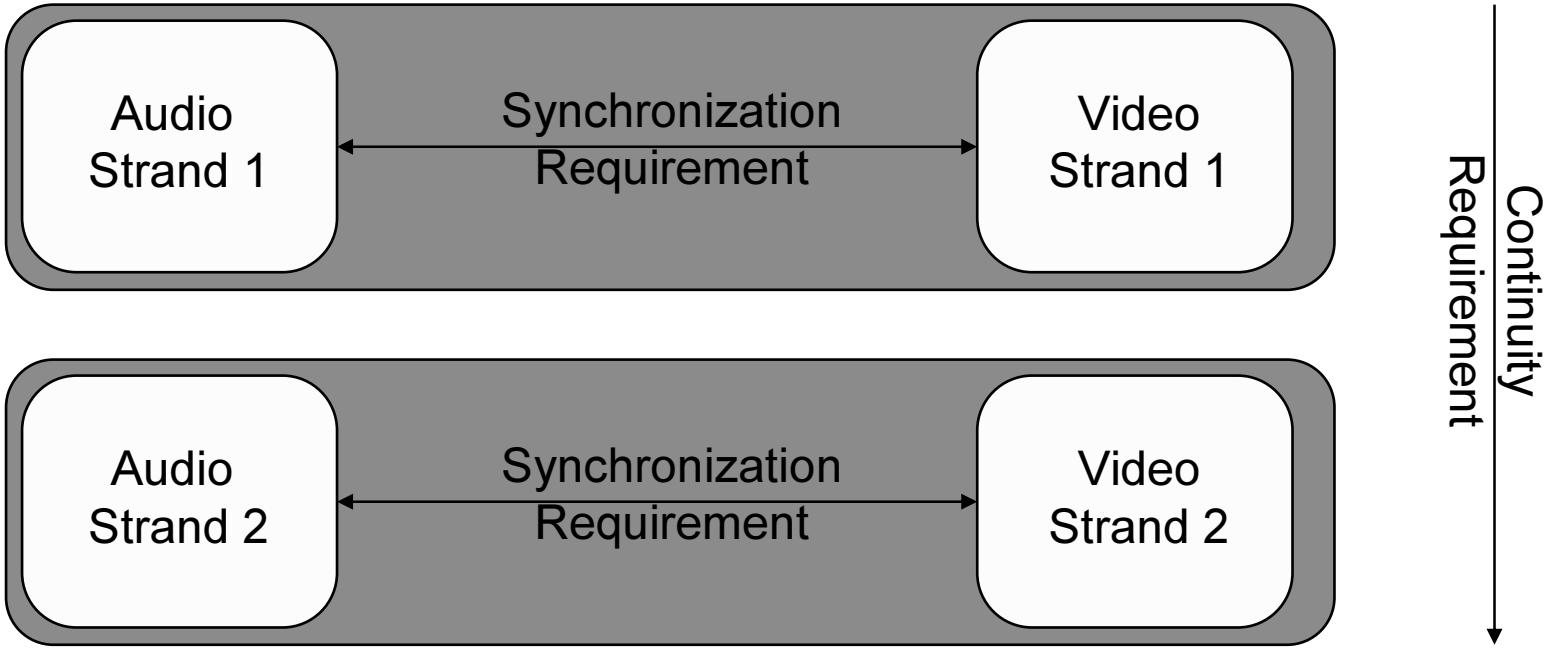
A strand will generally include headers and other information (e.g. about compression).

Components of a strand:

- **Media Blocks (MB) placed according to the placement model**
- **Primary Blocks (PB) contain a sequence of (MB, disk-location) pairs**
- **Secondary Blocks (SB) contains pointers to PB**
- **Header Block (HB) root of the strand (pointers to all SB, recording length, rate)**



Multimedia Rope: all media strands which constitute a logical entity (e.g. video and associated audio of a movie)



Data Structure of a multimedia rope


```
MultimediaRopeID
Creator
Length
PlayAccess
EditAccess
List of [
    List of [
          MediaStrandID
          <StrandIntervalStart,
           StrandintervalEnd>
          Media type
          EncodingFormat
          MediaRecordingRate
          StorageGranularity
          List of [
                    <MediaUnitID, RTS>
          ]
          MaximumAsynchrony
    ]
    List of [
          RTSvalue
    ]
]
```

Unique ID
Identification of the creator.
Length of the rope.
List of users or group ID.

Unique ID of media strand.
Strand interval, in media units.
Medium of the strand.
Strand encoding format: MPEG, JPEG, etc.
Rate of recording, in media units/s.
Media granularity, in media units/s.

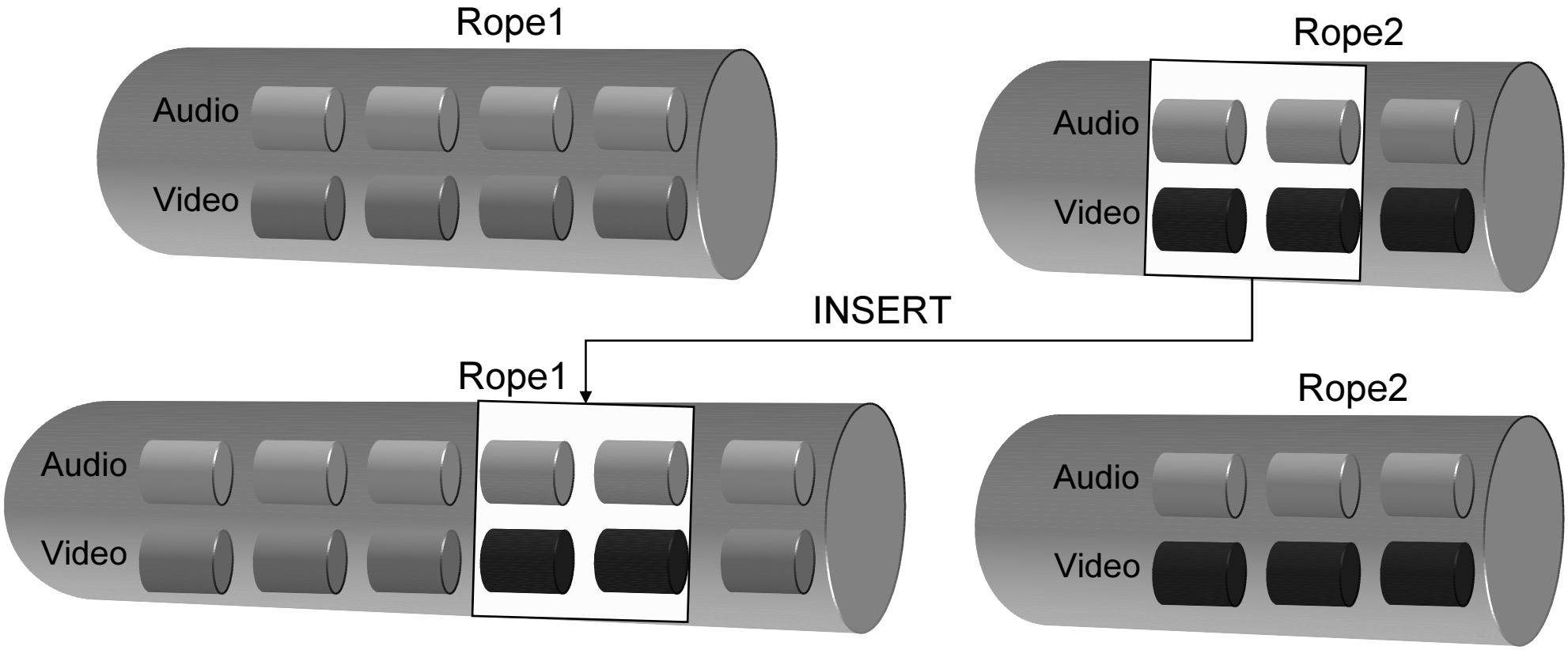
Media unit and its corresponding RTS.
(RTS = Relative Time Stamps).
Tolerable asynchrony threshold.

Discrete synchronization points.
Identities of synchronization points.

At the time of recording temporal relationships among strands (may be recorded at different sites/times) will be represented by RTS = Relative Time Stamp.

During playback all media units must be played in RTS in relation to the other strands.

Editing operations on ropes manipulate pointers to strands.
Intervals of strands can be shared by different ropes.
Strands that are not referenced by any rope can be deleted, and storage can be reclaimed.



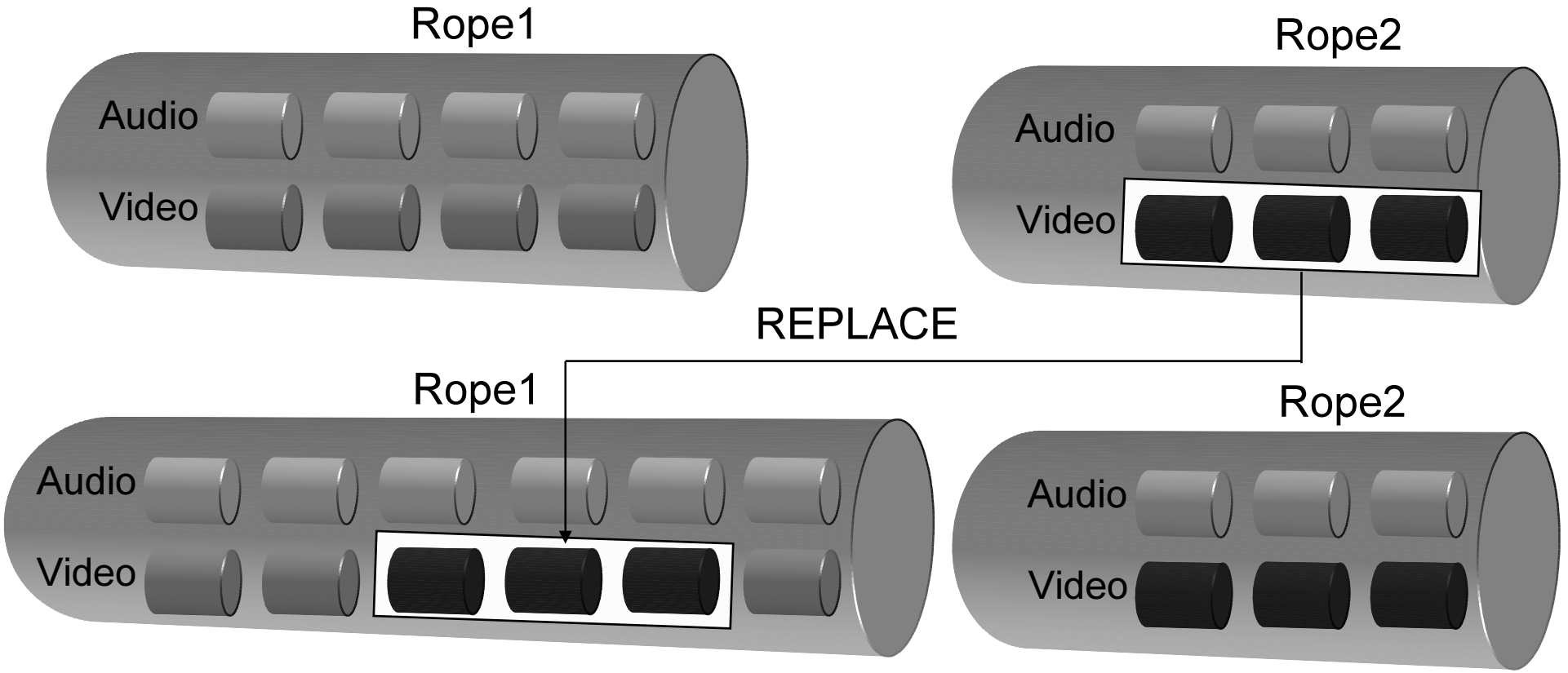
Further operations:

- `RECORD [media][requestID, mmRpeID]`

  Records a Mmrope represented by “`mmRopeID`“ which consists of several media strands until a `STOP` operation is issued.

- `PLAY [mmRopeID, interval, media] requestID`
- `STOP [requestID]`

8.4 Storage Devices / Data Structuring



Operations:

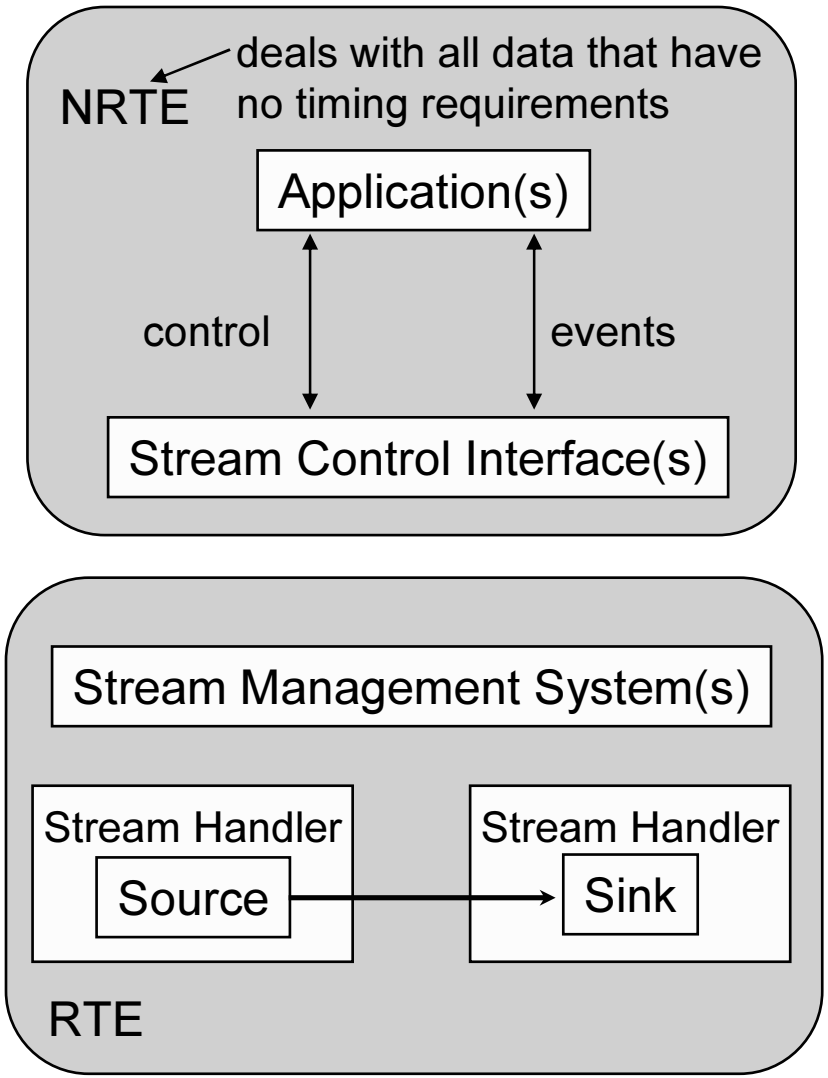
- `INSERT [baseRope, position, media, withRope, withInterval]`
- `REPLACE [baseRope, position, media, withRope, withInterval]`
- `SUBSTRING [baseRope, media, interval]`
- `CONCATE [mmRopeID1, mmRopeID2]`
- `DELETE [baseRope, media, interval]`

Multimedia System must coexist with the conventional Non Real Time Environment (NRTE) data processing

➔ Many operating systems provide extensions to support multimedia application Real Time Environment (RTE)

Application itself never really touches the data

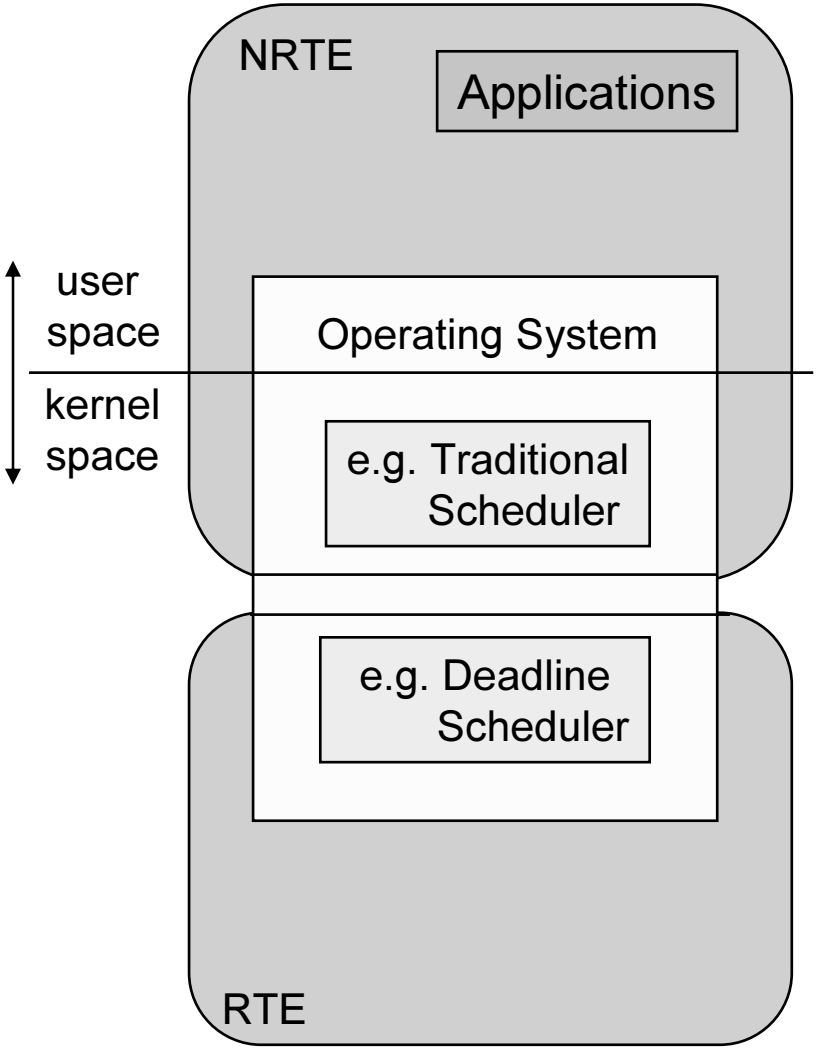
➔ take the shortest possible path from source to the sink



RTE schedules the processes according to the timing requirements

Stream Handlers manage the RTE data flow in accordance with the control operations of the NRTE

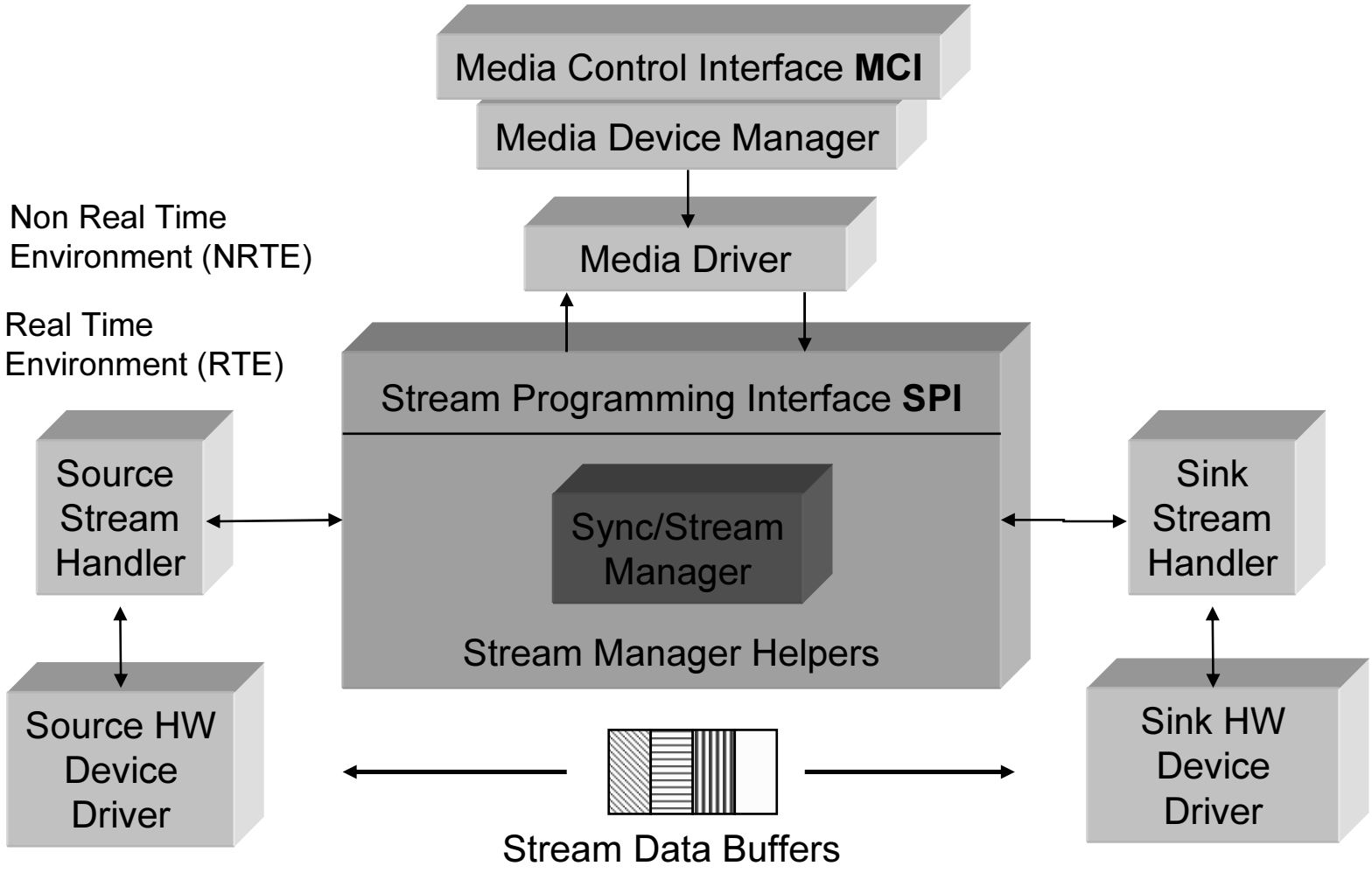
Applications access stream handlers by establishing (creating) sessions.



Applications make use of systems calls in the NRTE

Extensions to the operating system (i.e. RTE) are part of the kernel space

OS/2 Multimedia Presentation Manager/2



Multimedia Presentation Manager/2 (MMPM/2)

- Part of IBM´s Opearting System/2 (OS/2)
- Well-suited for multimedia supporting preemptive multitasking, priority scheduling, demand-paged virtual memory storage, etc.

Media Control Interface (MCI) (device independent programming interface)

- Open, close, status of device (for all devices)
- Play, record, resume, stop (playback, record)
- Set cue point (allows for synchroniation)
- Get table of contents of a CD-ROM (device-specific command)

Stream Programming Interface (SPI)

- Implementation of data streaming and synchronization
- Access to the SyncStream Manager (coordinates and manages the buffers, synchron.)

Ease of use several levels

- Style guide for applications
- Unified graphical interfaces
- Application developers and device providers can intergrate their own I/O processes, stream handlers, etc.