

# 6. Media Server

**6.1 Media Server Architecture**

**6.2 Storage Devices and Disk Layout**

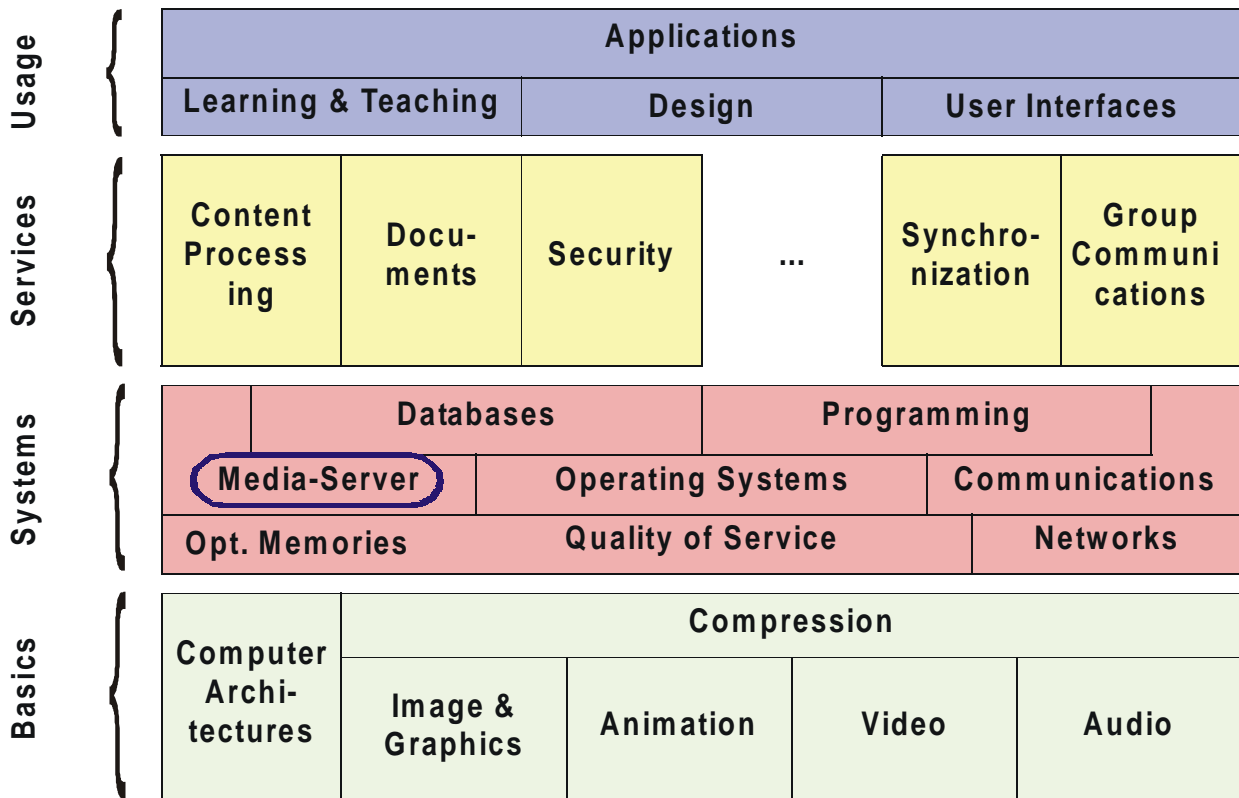
**6.3 Disk Controller and RAID**

**6.4 Storage Management and Disk Scheduling**

**6.5 File Systems, Video File Servers**

A Graduate Course on Multimedia Technology	© Wolfgang Effelsberg, Ralf Steinmetz	6. Media Server	6-1
---	--	-----------------	-----

# Role of the Media Server



# 6.1 Media Server Architecture

## Media server

- A special type of data / file server
- High-volume data transfer
- Real-time access
- Large files (objects, data units)

## Pull model

- The client controls the data delivery
- Suitable for editing of content over networks

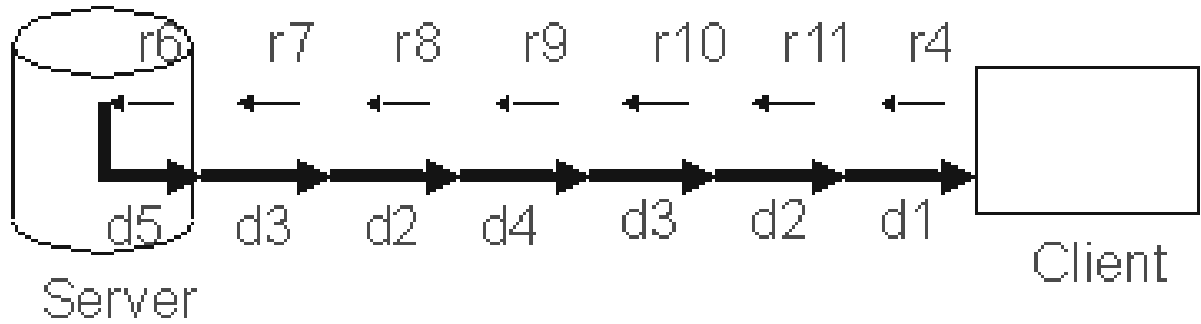
## Push model

- Also known as “data pump“
- The server controls data delivery
- Suitable for broadcasting data over networks

## Basic models: pull & push

- Mainly an application point of view how to interact with media data
- Mixtures possible: application sends “play list” to server
- Same server internals apply to both models (i.e., not treated separately in the rest of this chapter)

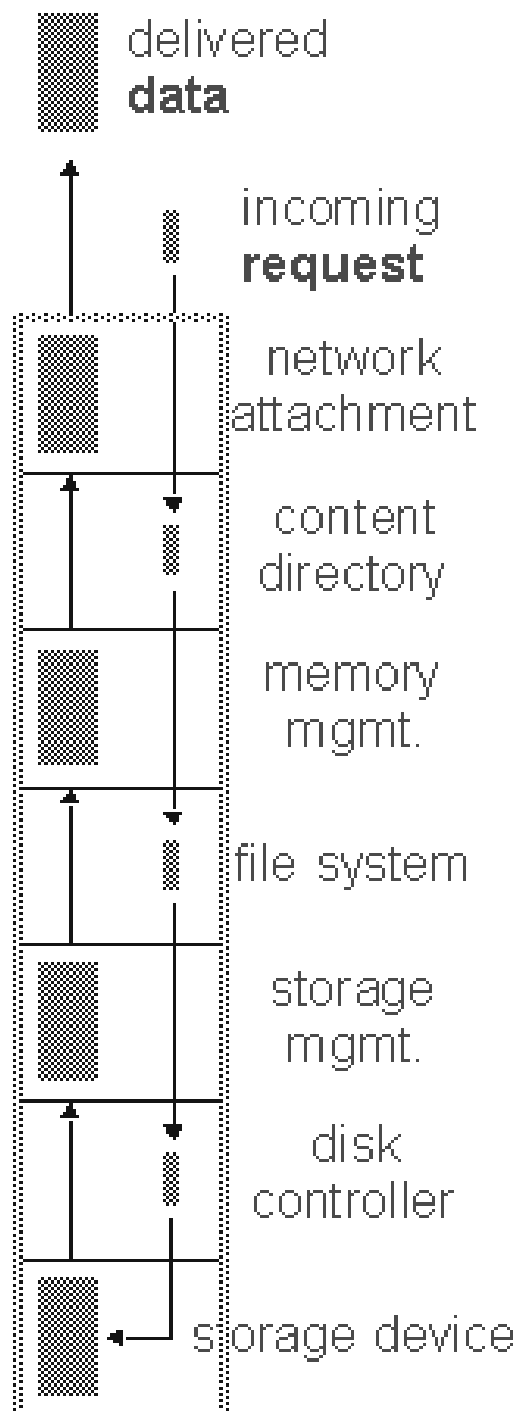
## Media Server - Push and Pull Model



## Media Server Architecture Components (1)

- **Network attachment**
  - typically a network adapter
- **Content directory**
  - responsible for verifying if content is available on the media server, and
  - if the requesting client is allowed to access the data
- **Memory management**
  - caching for large amounts of data and performance improvement
- **File system**
  - handles the organization of the content on the server
  - this includes assignment of sufficient storage space during the upload phase
- **Storage management**
  - abstraction of driver
  - responsible for disk scheduling policies and layout of files
- **Disk controller**
  - handles access to data on the storage device
  - head movement speed, I/O bandwidth, the largest and smallest units that can be read at a time, and the granularity of addressing, (e.g., RAID)

## Media Server Architecture Components (2)



# Scaling of Media Server - Cluster of Server (1)

## Motivation

- Growth of systems implies replication of multiple components

## Approach

- Optimization of each component
- Distributed onto probably heterogeneous components
- Cooperation between distributed components

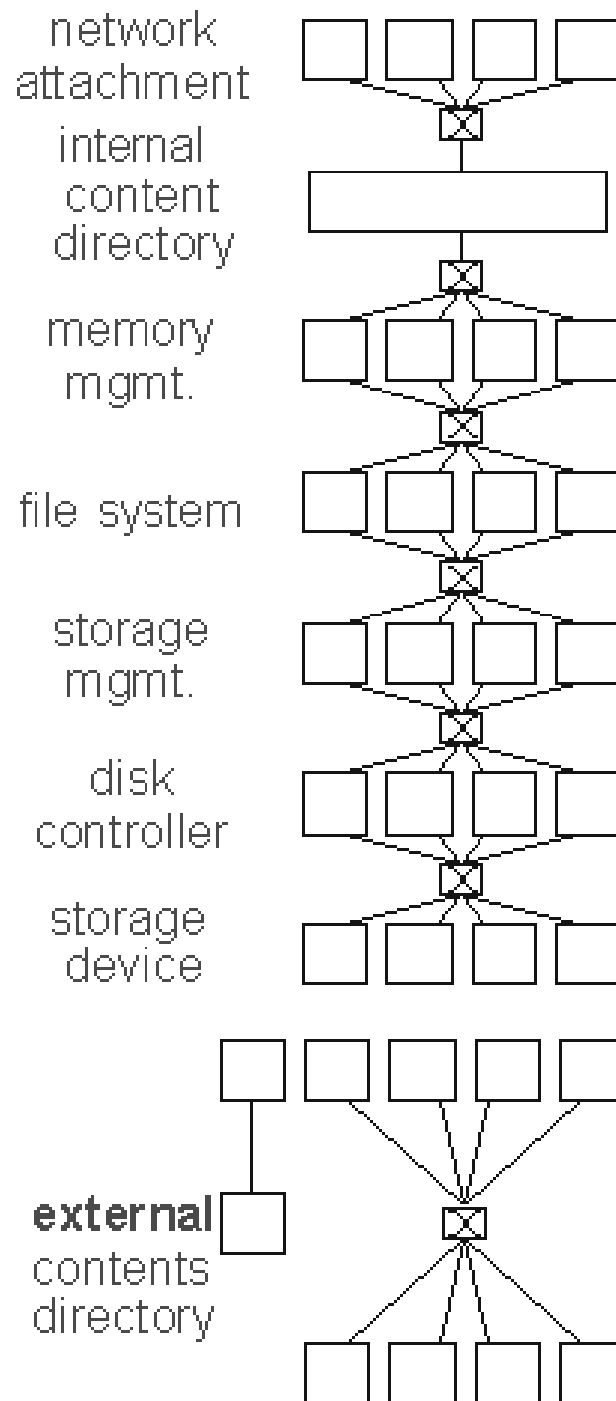
## Issues to be solved

**Example:** Content directory must always be consistent

- Internal content directory, once per media server
- External content directory

A Graduate Course on Multimedia Technology	© Wolfgang Effelsberg, Ralf Steinmetz	6. Media Server	6-7
---	--	-----------------	-----

## Scaling of Media Server - Cluster of Server (2)





# 6.2 Storage Devices and Disk Layout

## Tape

- Cannot provide multiple streams in parallel
- Random access is slow

## Disk

- Access times:
  - Seek time typically 8 ms magnetic vs. 150 ms optical disk
- CLV vs. CAV:
  - Magnetic disks usually have constant rotational speed
    - constant angular velocity, CAV
    - more space on outside tracks
  - Optical disks have varying rotational speed
    - constant linear velocity, CLV
    - same storage capacity on inner and outer tracks
- Capacity vs. cost: Optical cheaper than magnetic
- Type of persistence (Rewritable, Write-once, Read-only, e.g., CD-ROM)

# Disk Layout (1)

## Determines

- the way in which content is addressed
- how much storage space on the media is actually addressable and usable
- the density of stored content on the media

## Multiple track vs. single track (CD)

- changes on single track data are expensive

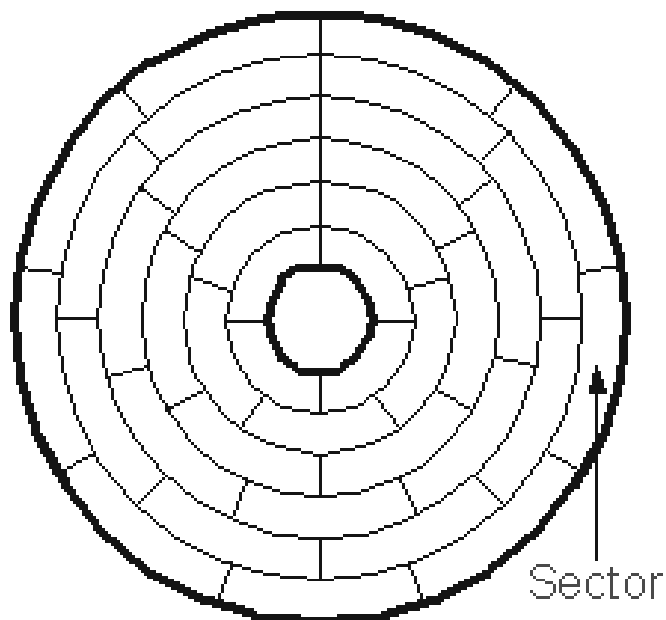
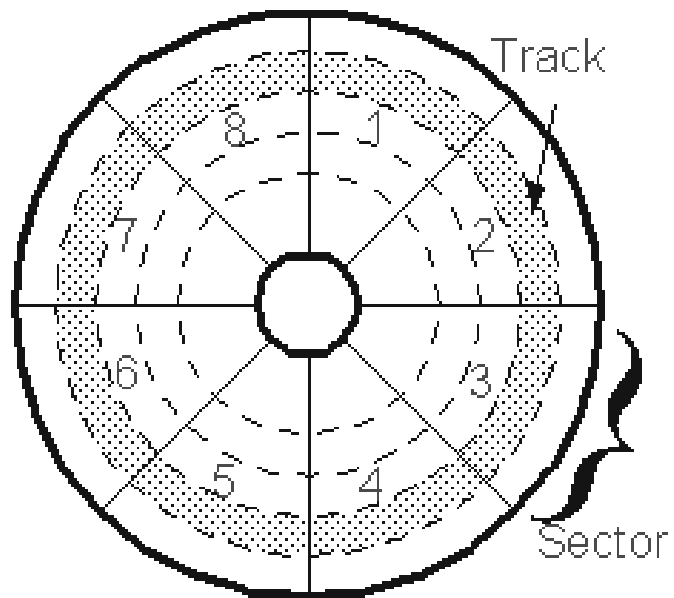
## Tracks and sectors

- access restricted to the unit of a sector
- unused space of a sector is wasted

## Zone bit recording

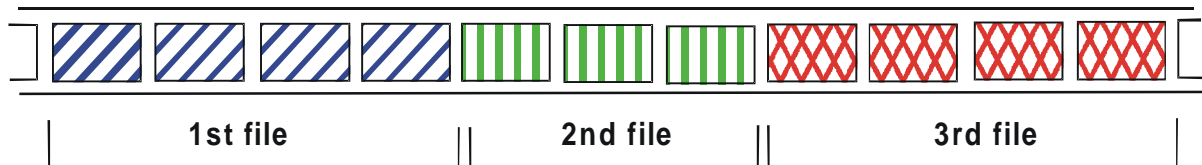
- motivation: a sector at an outer radius has the same (sector) data amount, but more raw capacity
- constant angular velocity
- i.e. same access time to inner/outer tracks
- different read/write throughputs
- Can be used to place
  - more popular media (movies) on an outer track
  - less popular on an inner track

## Disk Layout (2)

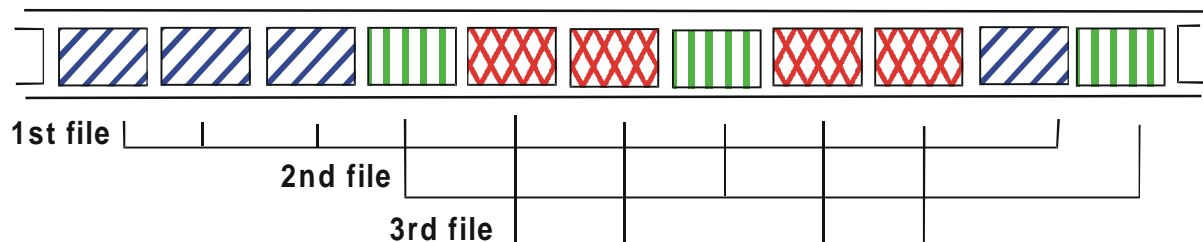


## Placement of Files at Storage Device Level (1)

contiguous placement:



non-contiguous placement:



File organization. A file is a sequence of bytes with a special “end of file” symbol.

- Contiguous (sequential) placement: stored in the order in which it will be read
  - like on a tape
  - fewer seek operations during playback, i.e., good for “continuous” access
  - less flexibility, problematic when data needs to be changed.

## Placement of Files at Storage Device Level (2)

- Non-contiguous placement, i.e. scatter blocks across disk:
  - avoids external fragmentation (“holes” between contiguous files)
  - same data can be used for several streams via references
  - long seek operations during playback

A Graduate Course on Multimedia Technology	© Wolfgang Effelsberg, Ralf Steinmetz	6. Media Server	6-13
---	--	-----------------	------

## 6.3 Disk Controller and RAID

### Redundant Array of Inexpensive Disks

#### Motivation

- disks become more and more inexpensive
- better to provide a set of disks instead of one large disk
- i.e. for “striping“

#### Goals: to enhance storage size AND

- primarily: fault tolerance (availability, stability)
  - by redundancy
  - related to (as low as possible) additional expenses
- secondarily: performance
  - by data striping
    - by distributing data transparently over multiple disks and making them appear as a single fast disk
  - fast read and write
  - for small and large amounts of data

#### RAID and multimedia

- RAID can help to improve multimedia data delivery from servers

# Redundant Array of Inexpensive Disks

## Granularity of data interleaving

- fine grained
  - small units to be interleaved
  - any I/O request (regardless of data unit size) involves all disks
- coarse grained
  - larger units of data to be interleaved
  - a small file (total data request) may involve only some disks

## Method and pattern of placing redundant data

- Computing redundancy data: most often parity, sometimes Hamming or Reed-Solomon codes
- Distribution/placement
  - either concentrate redundancy on some disks
  - or distribute it uniformly

## Reference

E.g., Chen et al: RAID: High-Performance, Reliable Secondary Storage, ACM Computing Surveys, Vol. 26, No. 2, June 1994

A Graduate Course on Multimedia Technology	© Wolfgang Effelsberg, Ralf Steinmetz	6. Media Server	6-15
---	--	-----------------	------

## Non-Redundant (RAID Level 0) (1)

### Goal and Usage

- to enhance pure I/O performance
- Mainly for use in supercomputers

### Approach

- data striping among a set of e.g. 4 disks
- **A block of data is split**, different parts of it are stored on different devices
- 4 disks of 1 GB provide in total a capacity of 4 GB
- Implementation
  - i.e., SCSI allows for up to 8 daisy-chained controllers and up to 56 logical units

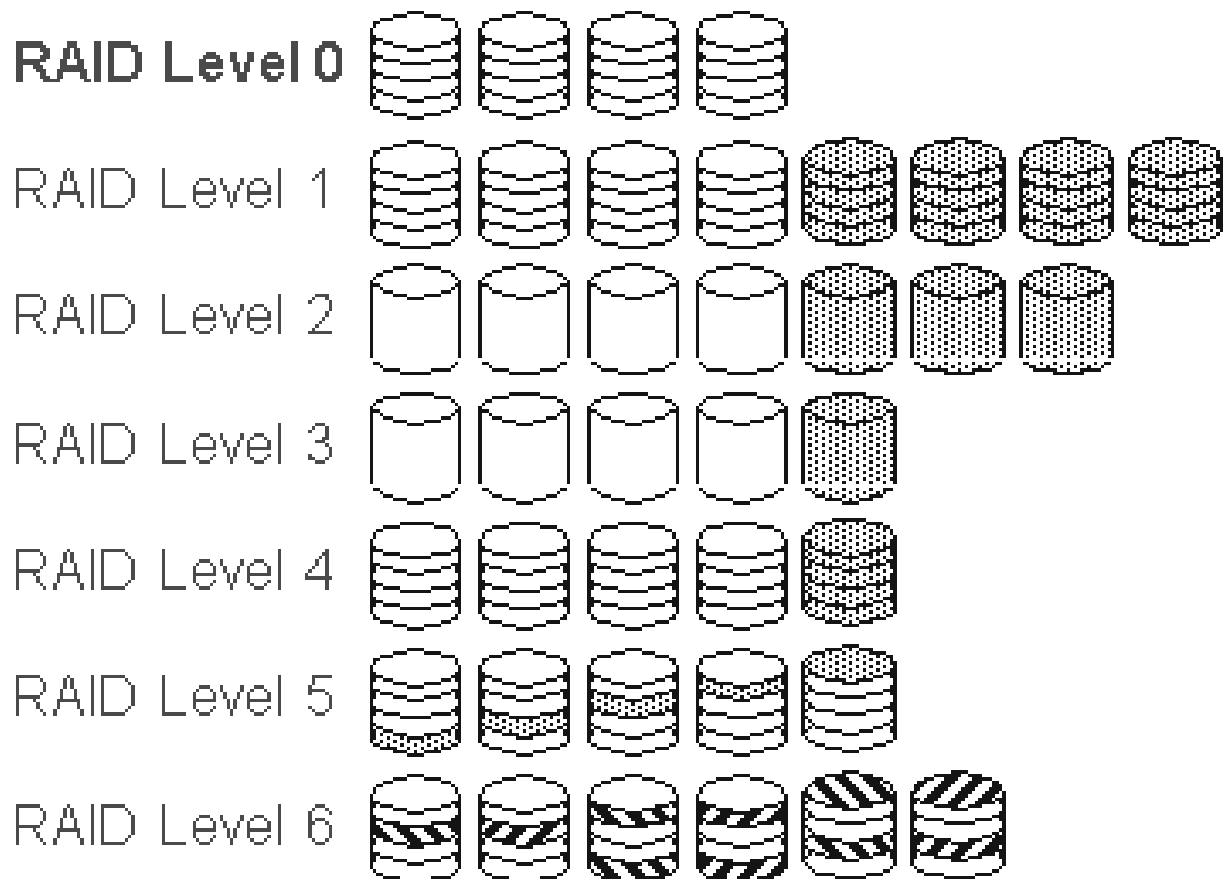
### Performance

- read
  - very good but mirrored disks may be better (if appropriate schedules are used)
- write
  - best of all RAID performances (no need to update any redundant data)

A Graduate Course on Multimedia Technology	© Wolfgang Effelsberg, Ralf Steinmetz	6. Media Server	6-16
---	--	-----------------	------



## Non-Redundant (RAID Level 0) (2)



# Mirrored (RAID Level 1) (1)

## Goal and Usage

- better fault tolerance
- frequently used for databases (when availability is more important than storage efficiency)

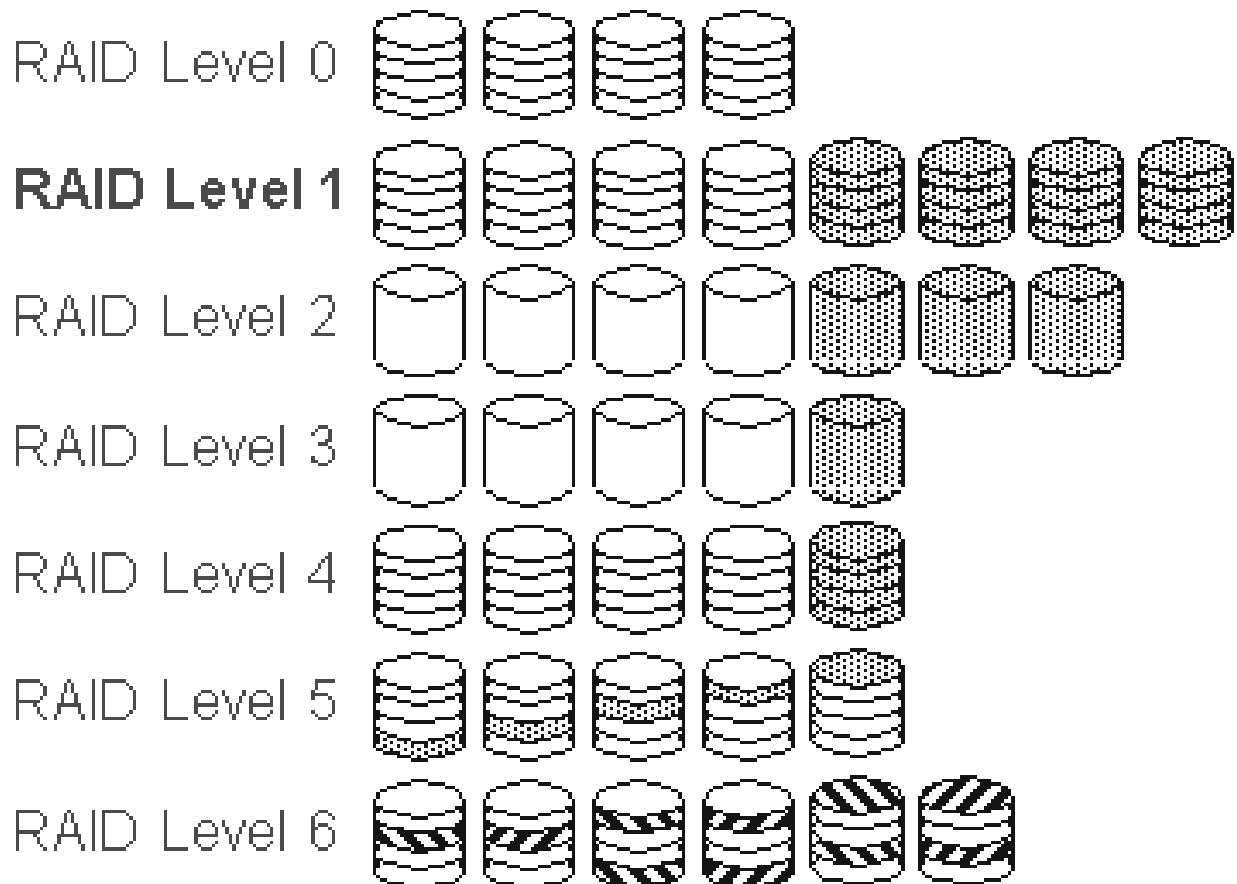
## Approach

- **Mirrored disks** (or shadowing)  
duplicate data written to second disk
- Every sector on the primary disk is also stored on the secondary disk

## Performance

- read
  - parallel reads can increase the I/O performance, or the disk with shorter queues, rotational delay, seek time can be selected
  - if different controllers are used
- write
  - slowed down (write must be done on two devices simultaneously)

## Mirrored (RAID Level 1) (2)



## Memory-Style ECC (RAID Level 2) (1)

### Goal

- to enhance fault tolerance
- to reduce RAID level 1 hardware costs

### Approach

- **Bit striping among various disks with additional error correction codes**
- error detection:
  - Single parity disk
- but here error correction used is proportional to log (number of disks)

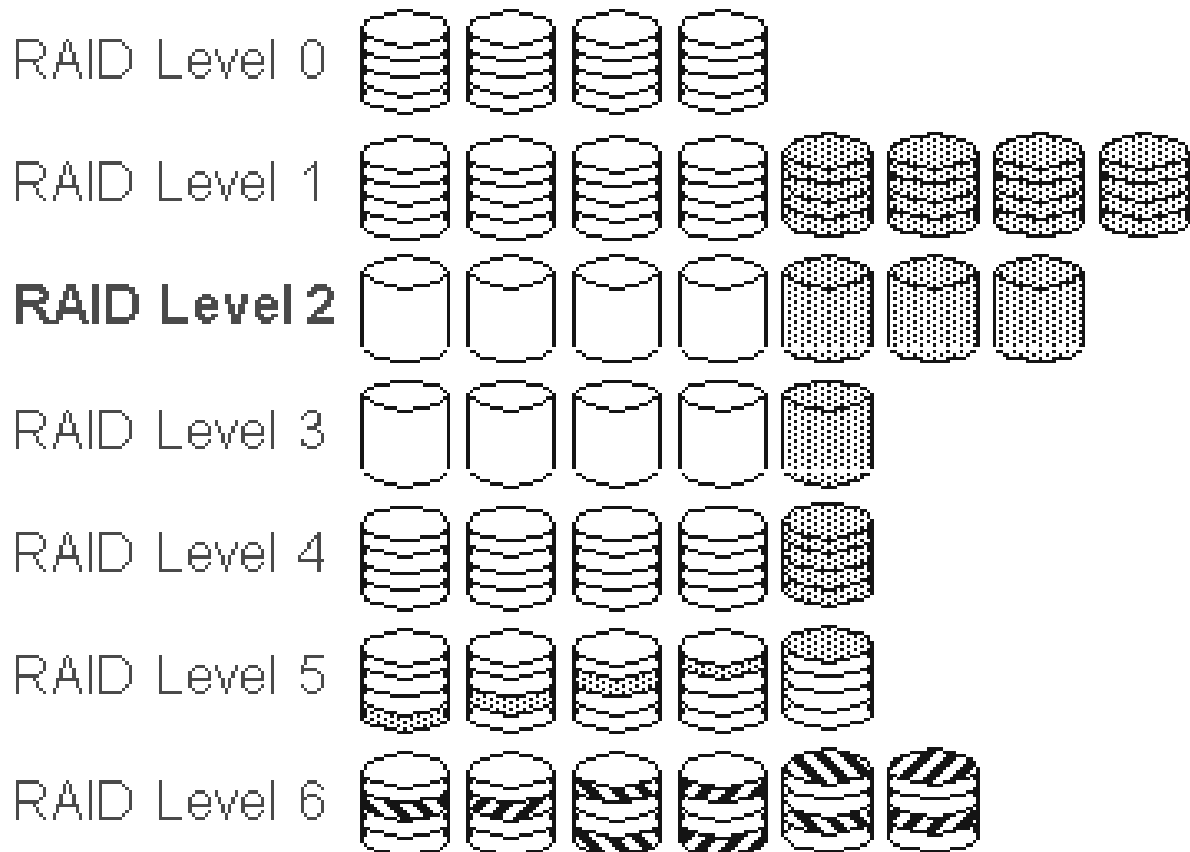
Example 1: 10 data disks with 4 parity disks

Example 2: 23 data disks and 5 parity disks

### Performance

- minimum amount of data that must be transferred is related to the number of disks (one sector on each disk)
- large amount leads to better performance
- slower disk recovery

## Memory-Style ECC (RAID Level 2) (2)



## Bit-Interleaved Parity (RAID Level 3) (1)

### Goal and use

- to enhance fault tolerance
- to reduce RAID level 2 hardware costs
- application when
  - high bandwidth is demanded
  - but not a high I/O rate

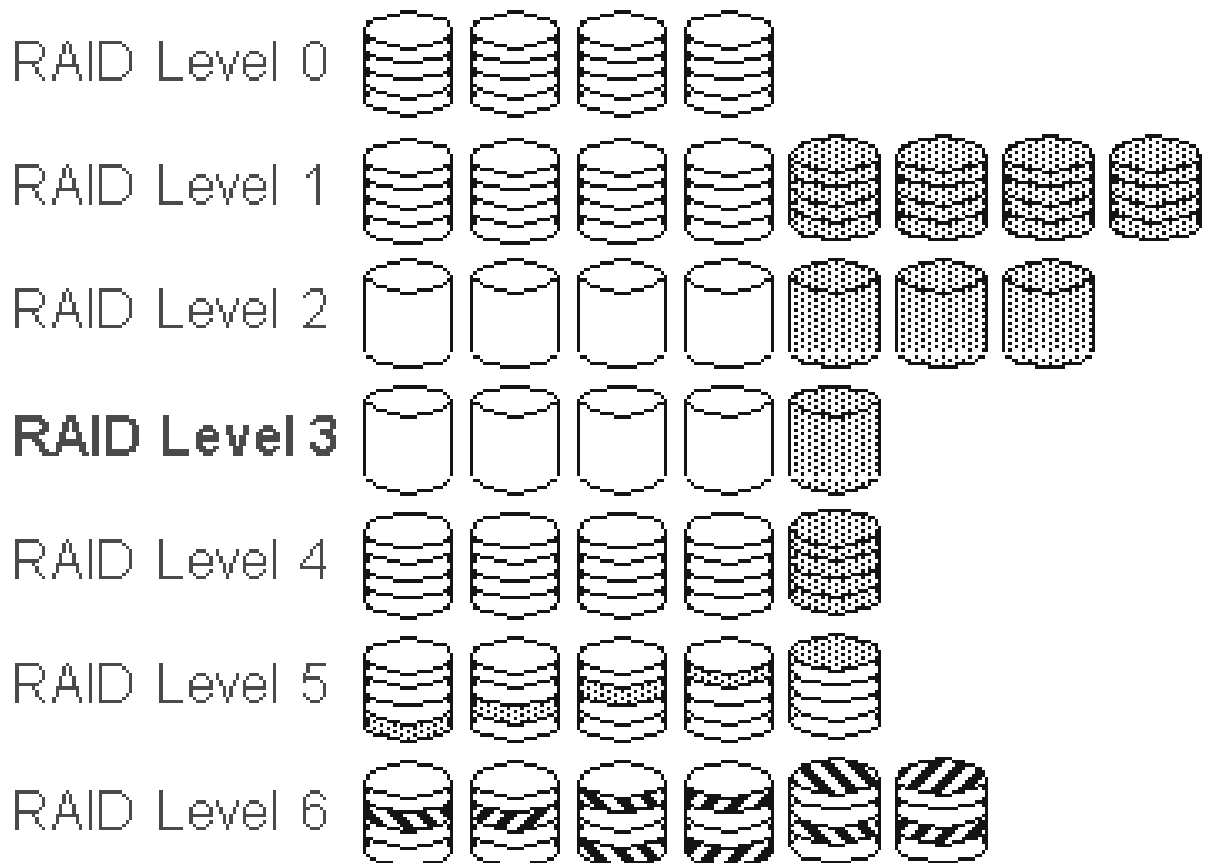
### Approach

- **Bit striping across disks**
- a single parity disk for any group/array of RAID disks
- makes use of build-in CRC checks of all disks

### Performance (similar to RAID level 2)

- slower disk recovery
- no interleaved I/O
- note: disks should be synchronized to reduces seek and rotational delays

## Bit-Interleaved Parity (RAID Level 3) (2)



## Block-Interleaved Parity (RAID Level 4) (1)

### Goal

- to provide fault tolerance
- to enhance RAID level 3 performance in case of a fault

### Approach

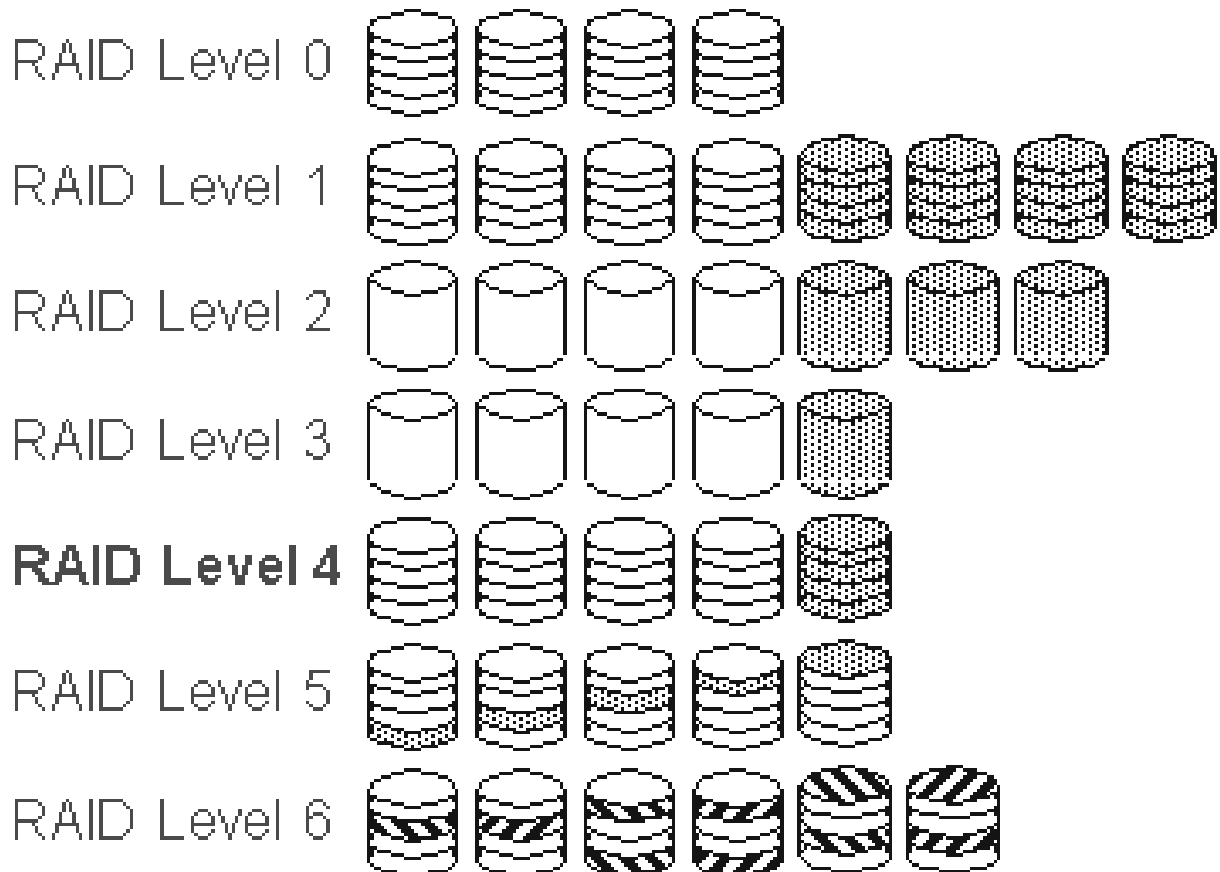
- **Sector striping across disks**
- parity sectors stored on one disk

### Performance

- faster disk recovery possible
- small writes
  - only two disks affected (not the entire set)
  - not in parallel (only one write per disk group as parity disk is affected)
- small reads are improved
  - from one disk only
  - may occur in parallel



## Block-Interleaved Parity (RAID Level 4) (2)



# Block-Interleaved Distributed Parity (RAID Level 5) (1)

## Goal

- to provide fault tolerance
- to remove the write bottleneck of RAID level 4

## Approach

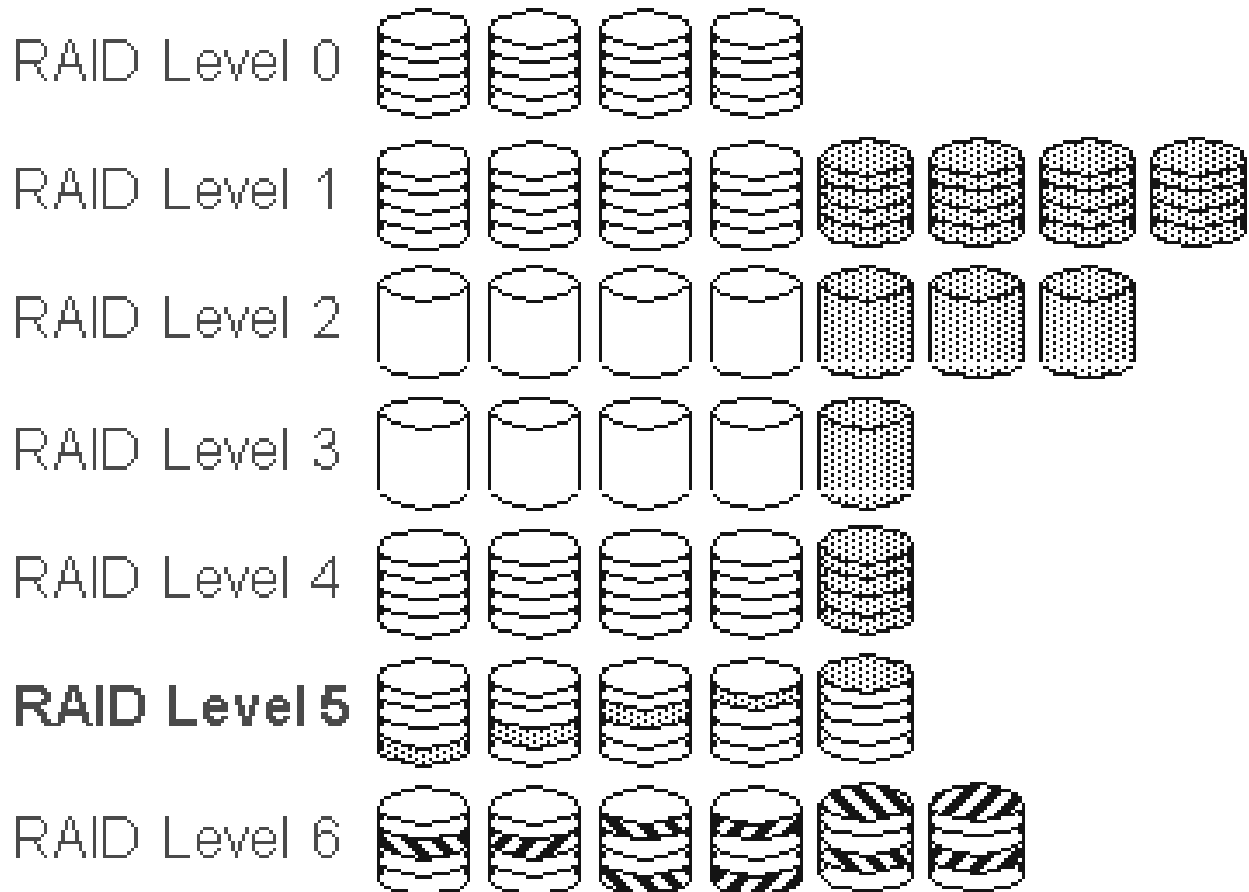
- **Sector striping across disks, parity information distributed over disks**

## Performance

- read and write: allow parallel operations
- small read or write
  - very good: similar to RAID level 1
- large amount of data
  - very good: similar to RAID 3 and 4

A Graduate Course on Multimedia Technology	© Wolfgang Effelsberg, Ralf Steinmetz	6. Media Server	6-26
---	--	-----------------	------

## Block-Interleaved Distributed Parity (RAID Level 5) (2)



## P+Q Redundancy (RAID Level 6) (1)

### Goal

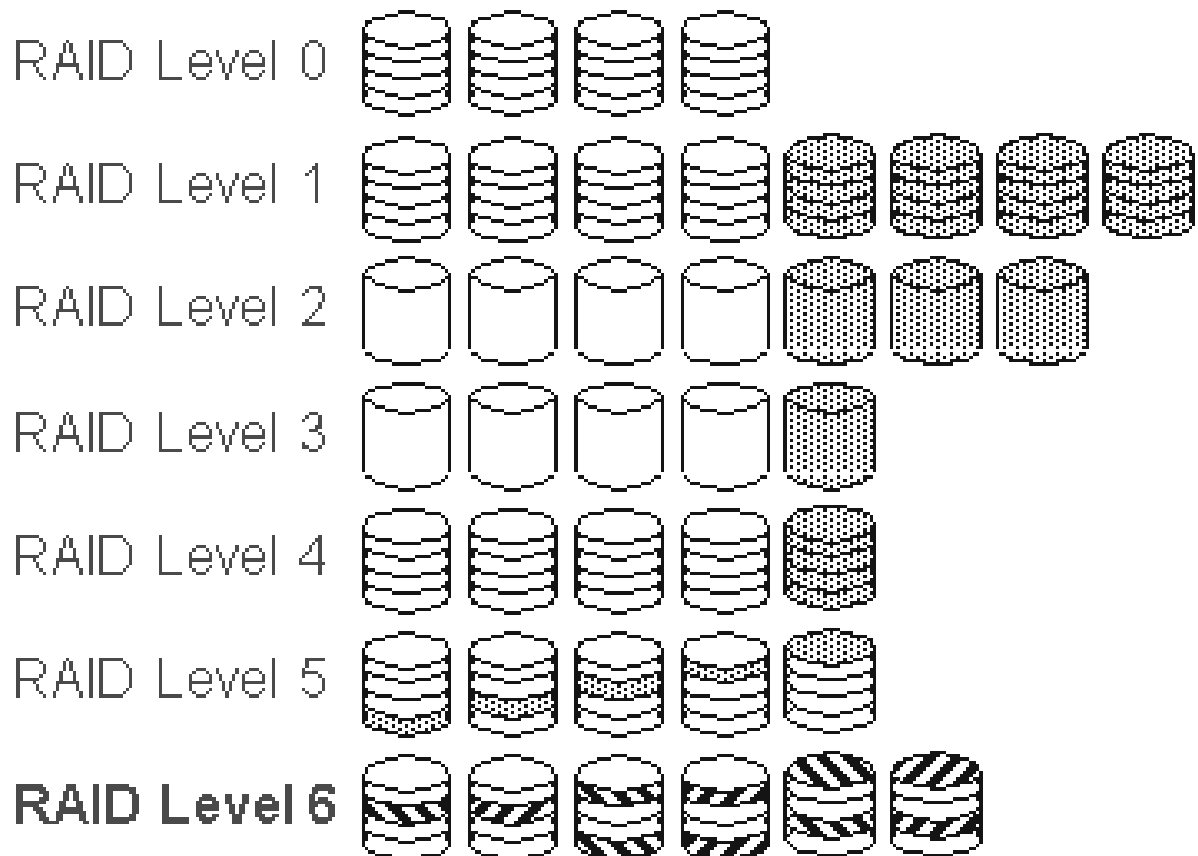
- Motivation:
  - Very large arrays may contain more than one disk with failures
- ECC is required in order to maintain availability

### Approach

- ECC
  - “P+Q redundancy” based on Reed-Solomon
  - protects against failure of two disks at the same time
- two additional disks
- otherwise similar to RAID level 5

A Graduate Course on Multimedia Technology	© Wolfgang Effelsberg, Ralf Steinmetz	6. Media Server	6-28
---	--	-----------------	------

## P+Q Redundancy (RAID Level 6) (2)



# 6.4 Storage Management and Disk Scheduling

## Disk Management - File Placement on Disk

**Goal: to reduce read and write times by**

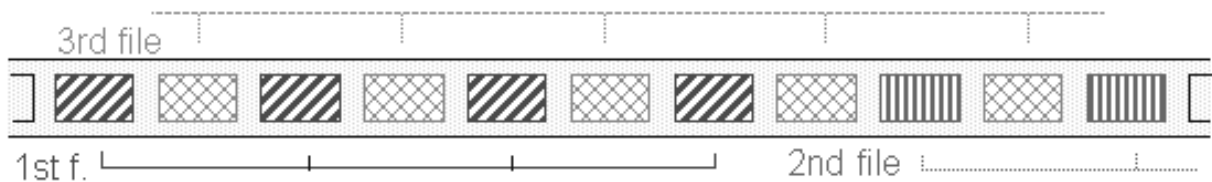
- fewer seek operations
- lower rotational delay or latency
- high actual data transfer rate (can not be improved by placement)

**Method: store data in a specific pattern**

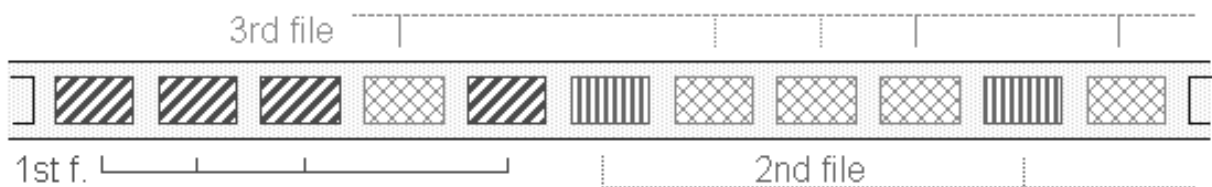
- Regular distance
- Combine related streams
- Larger block size
  - fewer seek operations
  - smaller number of requests
  - but higher loss due to internal fragmentation (last block used only 50% on the average)

## Interleaved Placement

interleaved storage:



non-interleaved storage:



### Interleaved files

- Interleaving several streams (e.g., channels of audio)
- All  $n^{\text{th}}$  samples of each stream are in close physical proximity on disk
- Problem: changing (inserting / deleting) parts of a stream

### Interleaved vs. non-interleave and contiguous vs. non-contiguous/scattered

- Contiguous interleaved placement
- Scattered interleaved placement

## 6.4.1 Traditional Disk Scheduling

### Definition

**Disk scheduling** determines the order by which requests for disk access are serviced.

### Disk service model

Requests are buffered and can be re-ordered before they are served by the disk.

### General goals of scheduling algorithms

- Short response time
- High throughput
- Fairness (e.g., requests at disk edges should not starve)

### Multimedia Goals (in general)

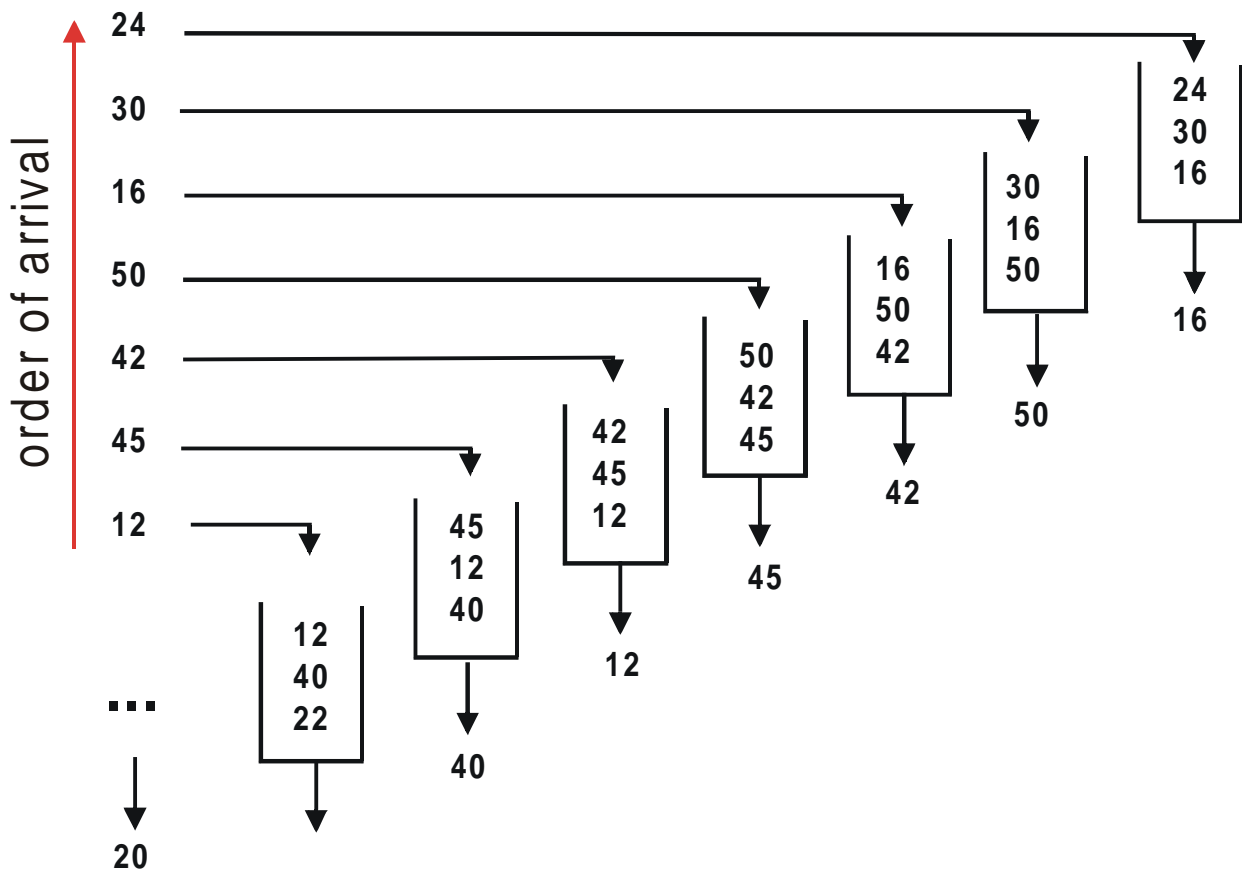
- continuous throughput (must not be fair)
- short maximal (not average) response times
- high throughput

### Typical trade-off

- Seek & rotational delay vs.
- maximum response time



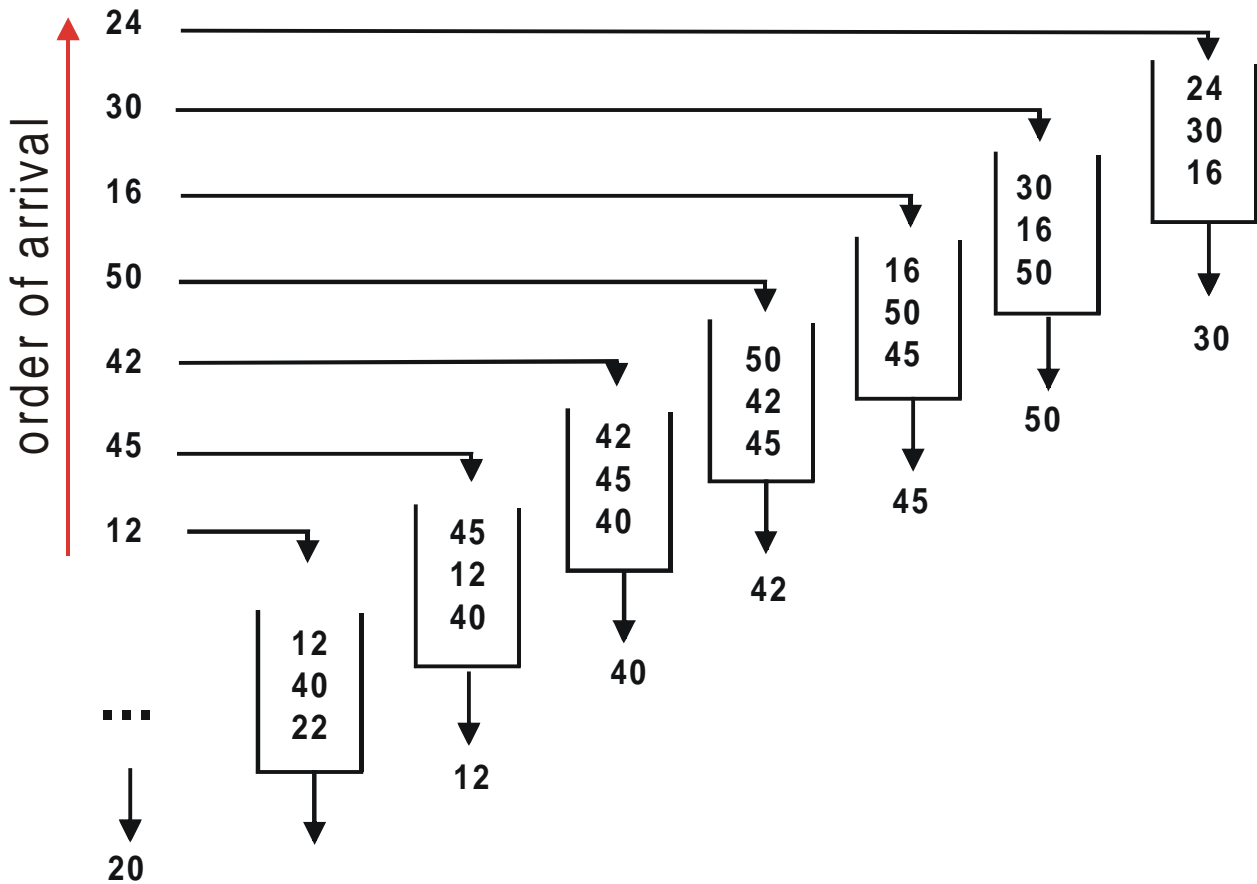
## First Come First Serve (FCFS) Disk Scheduling



### Properties

- Long seek times (since non-optimal head movement occurs)
- Short (individual) response times

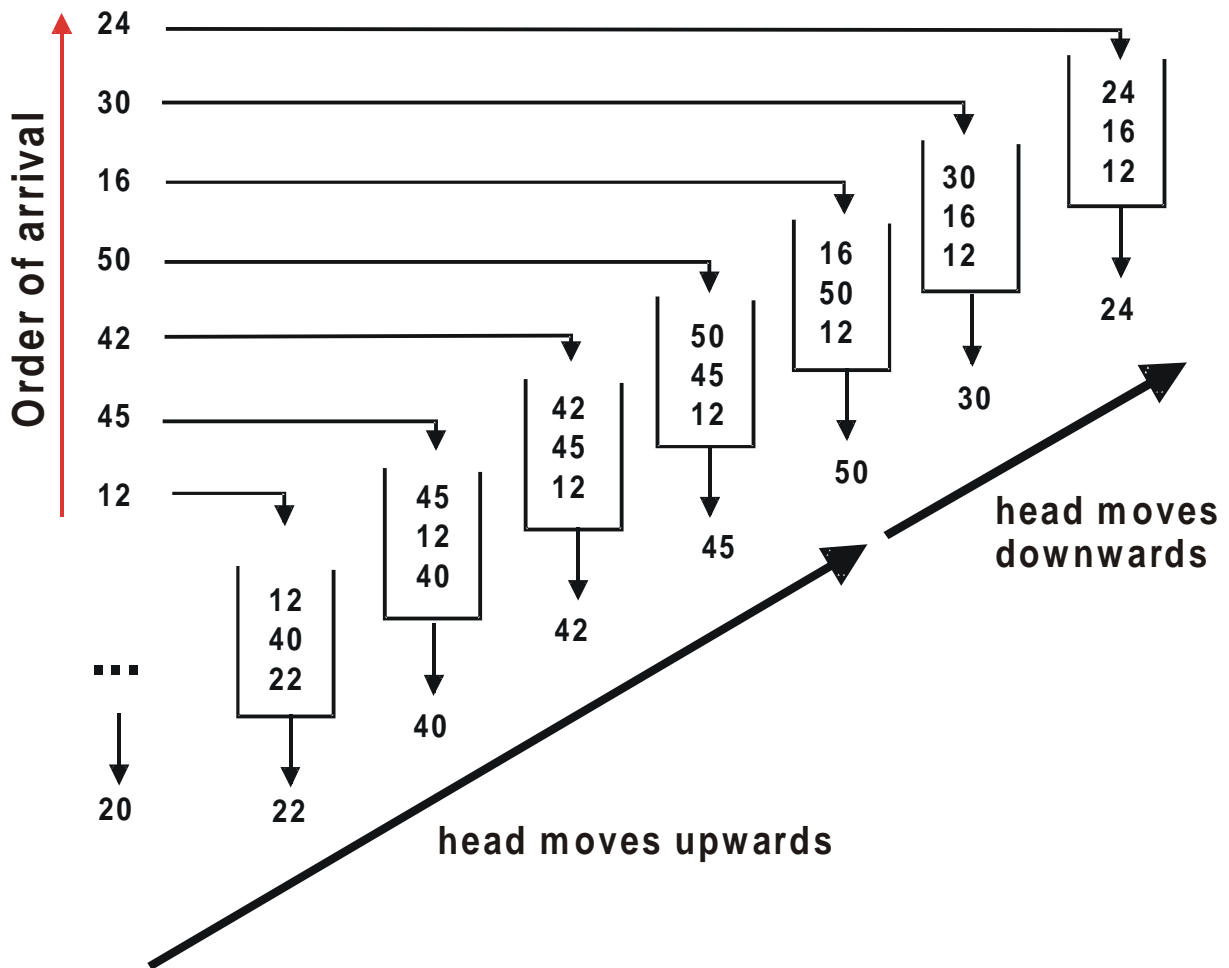
## Shortest Seek Time First (SSTF) Disk Scheduling



### Properties

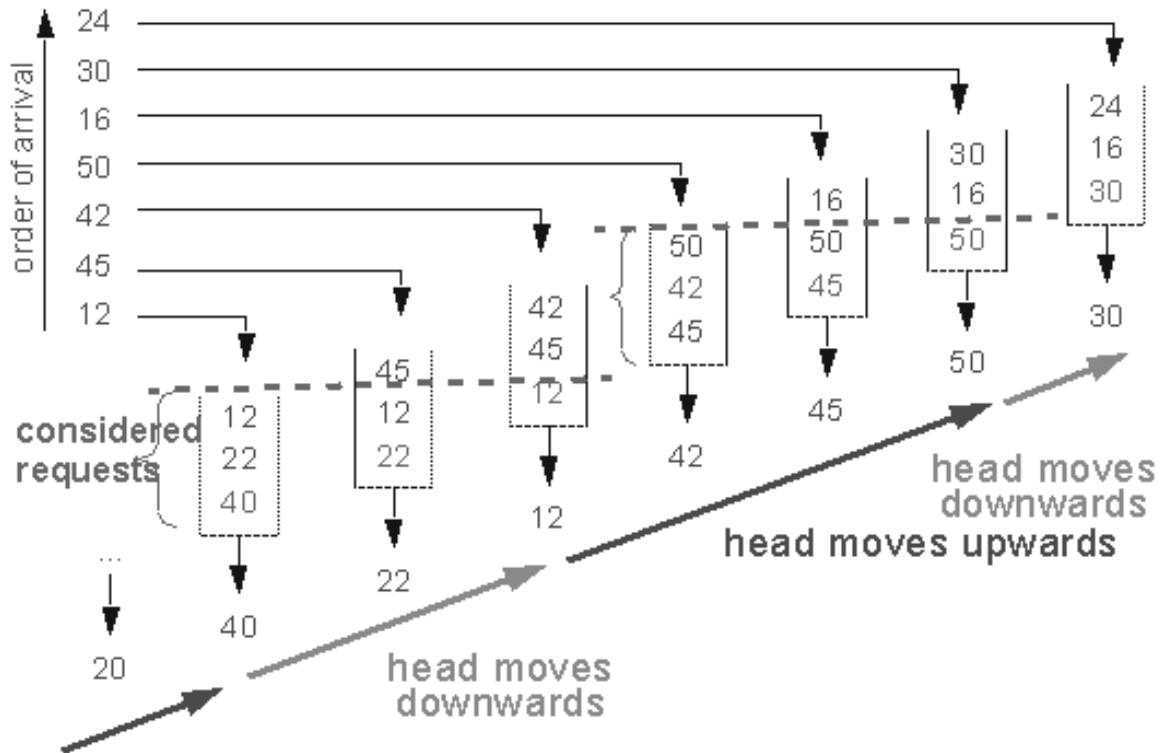
- Short seek times
- Longer maximum (individual) response times
- May lead to starvation

## SCAN Disk Scheduling



- Move disk head always between disk edges (up until the end, then down until the end)
- Read next requested block in disk movement direction
- A compromise between optimization of seek times and response times
- Data in the middle of the disk has better access properties

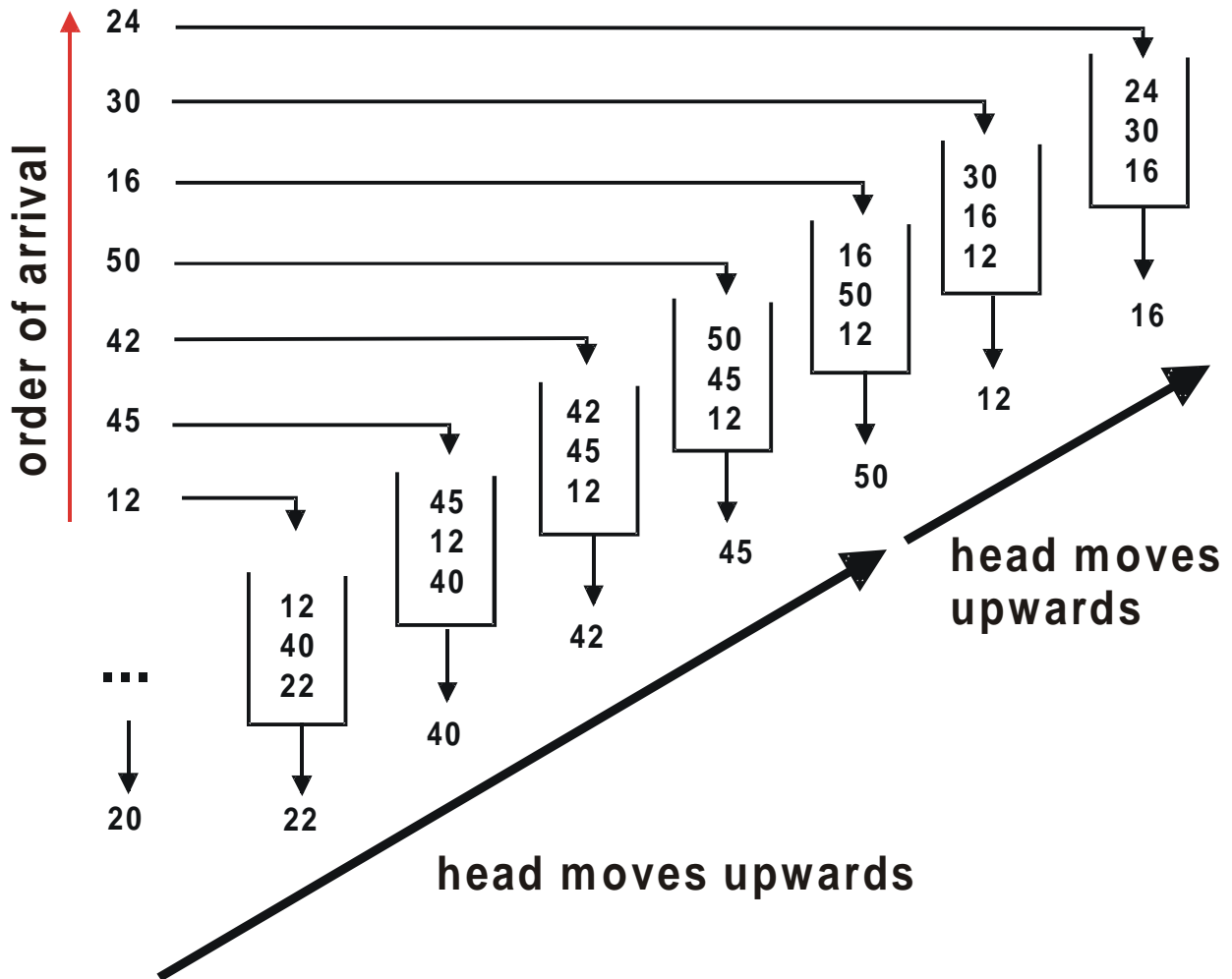
## N-Step-SCAN Disk Scheduling



### Properties

- reduces unfairness for outer and inner tracks
- longer seek time
- shorter response time

## C-Scan Disk Scheduling



### Properties

- Move disk head always between disk edges (UNIdirectional; up to the end, quickly down, then up to the end again)
- Improves fairness (compared to SCAN)

## 6.4.2 Disk Scheduling for Continuous Media

### Suitability of traditional disk scheduling methods

- Effective utilization of the disk arm? short seek time
- No guaranty for / not optimized for deadlines! -> not suitable for continuous streams

### Specific scheduling methods for continuous streams

- Serve continuous media, i.e., periodic requests with deadlines, plus aperiodic requests from other media
- Never miss a deadline of a continuous medium while serving aperiodic requests
- Aperiodic requests should not starve
- Provide high multiplicity (multiple streams) with real-time access
- Balance the trade-off between buffer space and efficiency

A Graduate Course on Multimedia Technology	© Wolfgang Effelsberg, Ralf Steinmetz	6. Media Server	6-38
---	--	-----------------	------

## Disk Scheduling: Dependencies

### Continuous media disk scheduling

#### Efficiency depends on the

- tightness of deadlines
- disk layout
- buffer space available

#### Ability to create a schedule in advance depends on

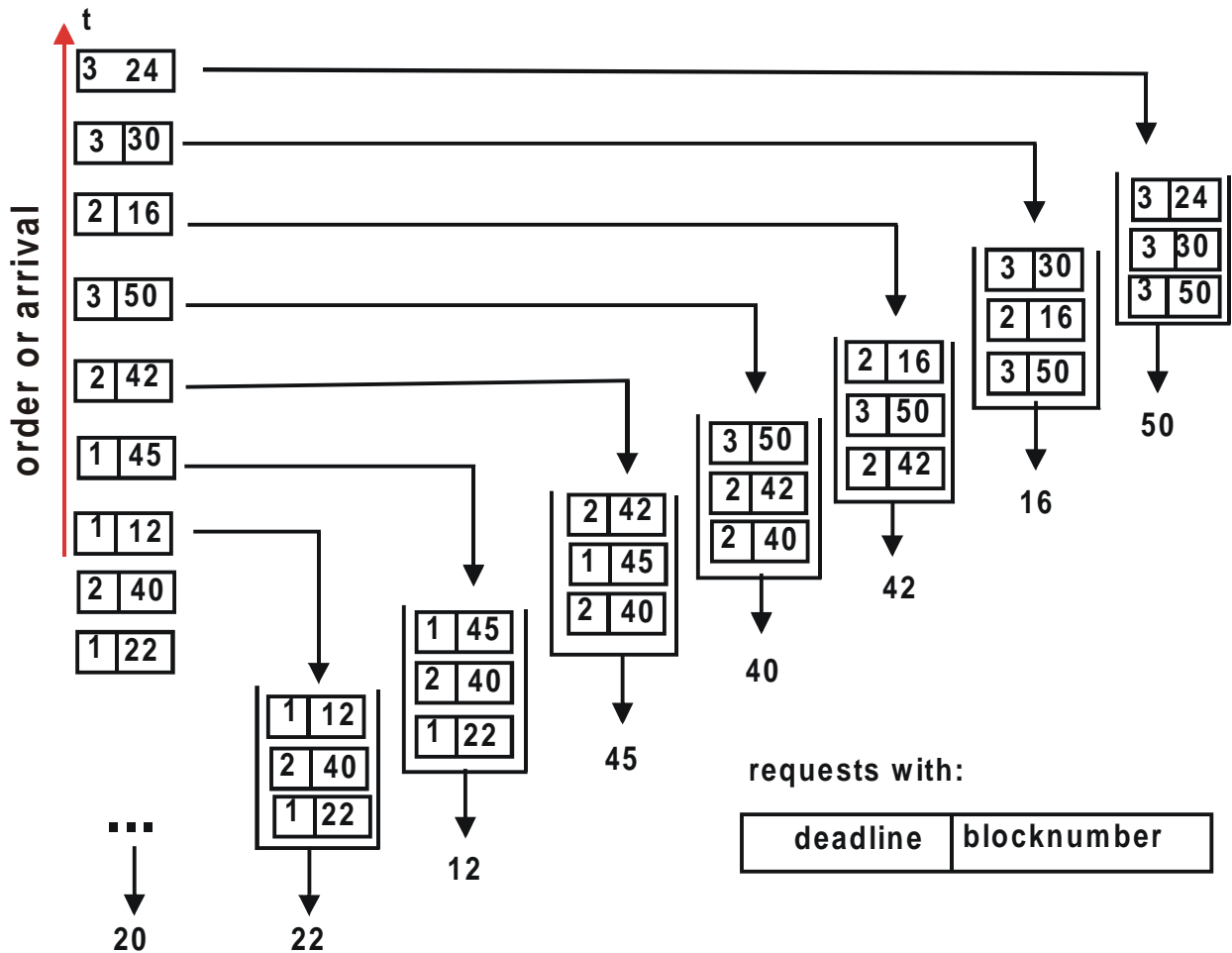
- buffer space and
- stream length

#### Most practical case

- create a schedule on-the-fly (to be considered here)

A Graduate Course on Multimedia Technology	© Wolfgang Effelsberg, Ralf Steinmetz	6. Media Server	6-39
---	--	-----------------	------

# Earliest Deadline First (EDF) Disk Scheduling



## Real-time scheduling algorithm

- First read the block with nearest deadline

## May result in

- excessive seek time and
- poor throughput



# Scan-EDF Disk Scheduling (1)

## Method

- Group requests by similar deadlines
- Requests with earlier deadlines are served first
- Among all requests with the same deadline, requests are served by track location

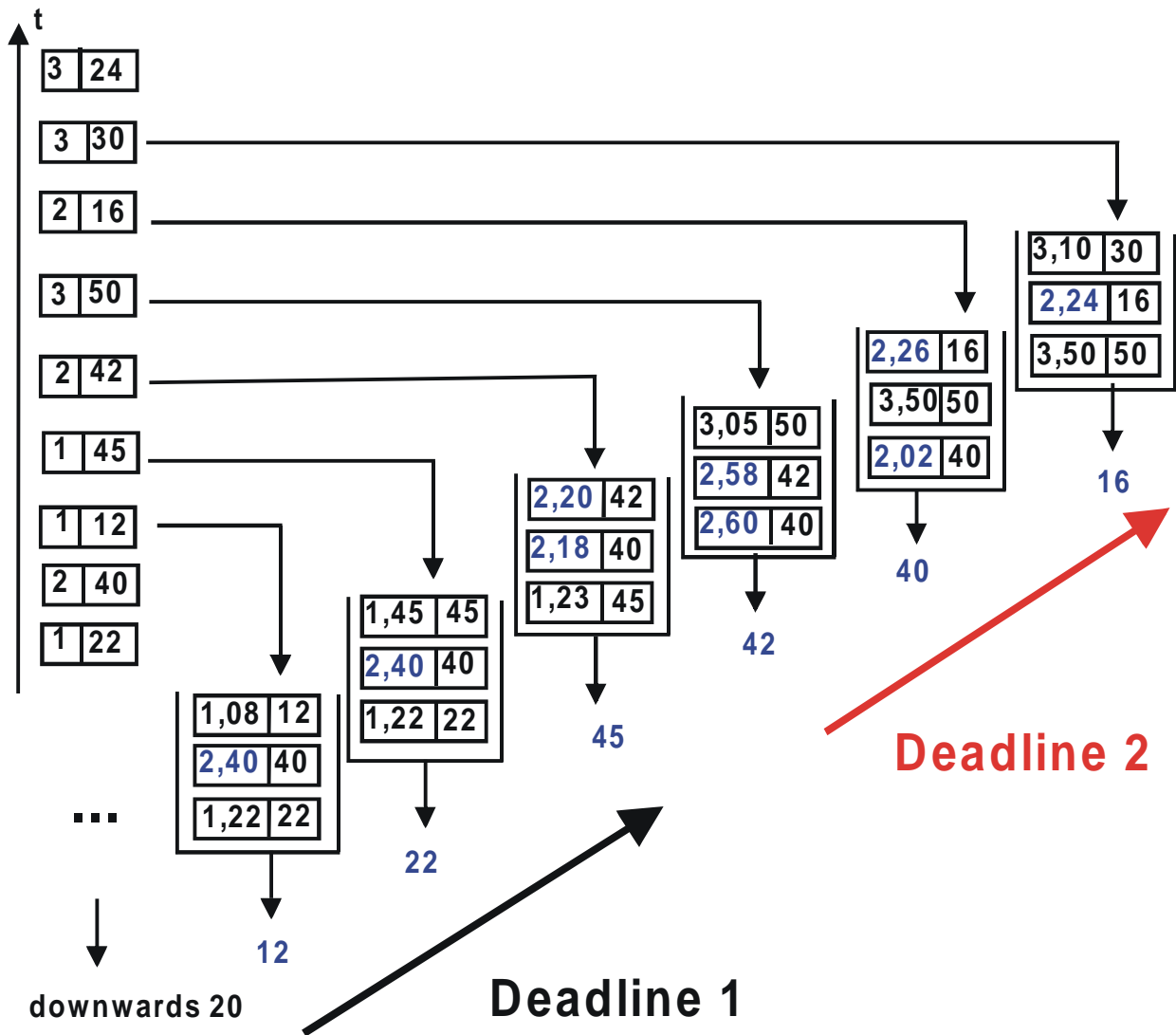
## Combines advantages of

- SCAN (seek optimization) with
- EDF (real-time aspects)

**We increase efficiency by modifying deadlines.**

A Graduate Course on Multimedia Technology	© Wolfgang Effelsberg, Ralf Steinmetz	6. Media Server	6-41
---	--	-----------------	------

## Scan-EDF Disk Scheduling (2)



### Properties

- apply EDF between groups
- for all requests within a group apply SCAN

## Scan-EDF Disk Scheduling (3)

### Map SCAN to EDF (1)

deadline =  $D_i + f(N_i)$ ;

$D_i$  deadline of request  $i$ ,

$N_i$  track position of request  $i$

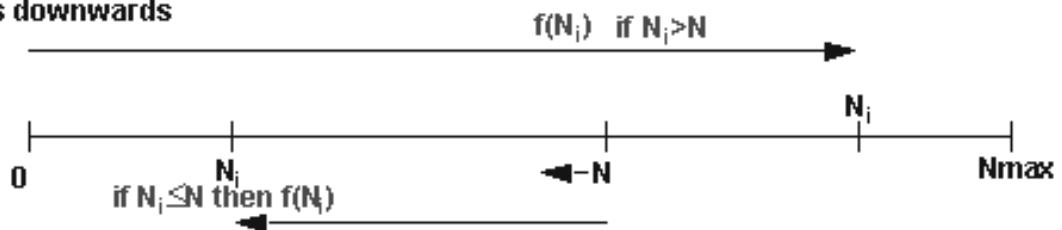
with  $f()$  such that  $D_i + f(N_i) > D_j + f(N_j)$  if  $D_i > D_j$

e.g.,  $f(N_i) = N_i / N_{\max} - 1$

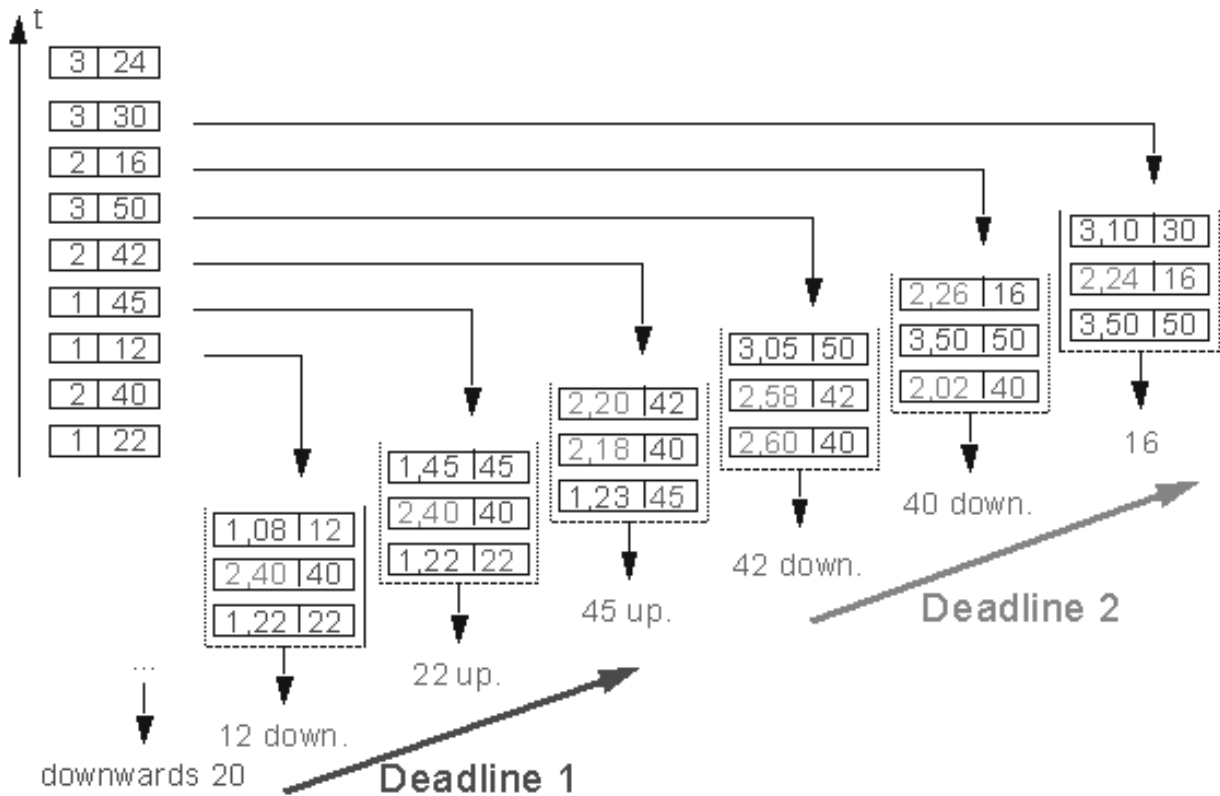
head moves upwards



head moves downwards



## Scan-EDF Disk Scheduling: Example



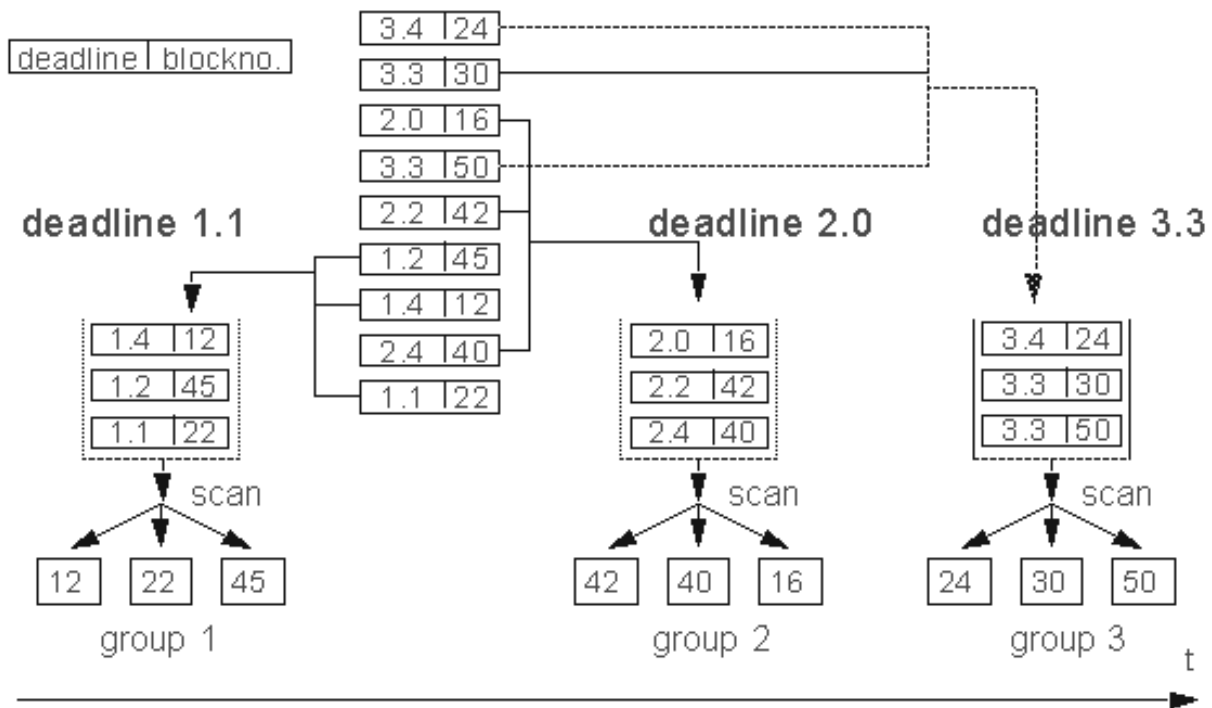
### Example

At “downwards 20” the next deadlines are computed, assume  $N_{\text{Max}}=100$

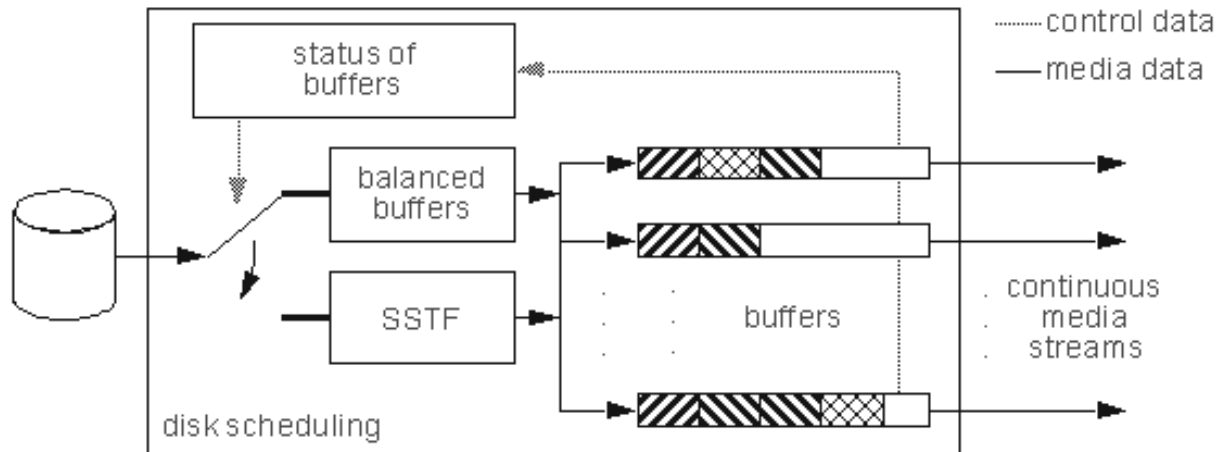
- 1 12: downwards & 12 on the way: position20 - position12 = 08, i.e. 1,08
- 2 40: downwards & 40 not on the way: =40; i.e. 2,40

## Group Sweeping Scheduling

- **Form groups**
  - with deadlines lying closely together
  - or in a round robin manner
- **Apply SCAN to each group**



## Mixed Disk Scheduling Strategies



### Goal

- to maximize transfer efficiency by minimizing seek time and latency
- to serve process requirements with a limited buffer space

### Combines

- shortest seek time first (SSTF)
- buffer underflow and overflow prevention
- by keeping buffers filled in a “similar way”

**Mixed Strategy also known as “greedy strategy”**

## 6.4.3 Data Replication (Content and Access Driven)

### Goal

- to increase the availability in case of disk failures (like RAID)
- to overcome the limit of concurrent accesses to individual titles caused by limits of the throughput of the hardware

### Static Replication

- user has choice of access points
- frequently done in the Internet today
- content provider keeps copies of the original version up to date on servers (proxies) close to the user.

### Dynamic Segment Replication

- read only, segments are replicated
- Since continuous media data is delivered in linear order, a load increase on a specific segment can be used as a trigger to replicate this segment and all following segments to other disks.

### Threshold-Based Dynamic Replication

- considers entire movies on a video server
- takes all disks of the system into account to determine whether a movie should be replicated

# 6.5 File Systems, Video File Servers

## File system

- is said to be the most visible part of an operating system

## Traditional file systems

- MSDOS File Allocation Table FAT, Berkeley UNIX FFS, ...

## Multimedia file systems

- Real Time Characteristics
- File Size
- Multiple Data Streams

## Examples of multimedia file systems

- Video File Server (experimental, outlined here )
- Fellini
- Symphony
- IBM Media Charger ...
- Real Networks ...

A Graduate Course on Multimedia Technology	© Wolfgang Effelsberg, Ralf Steinmetz	6. Media Server	6-48
---	--	-----------------	------



## Example: Video File Server

### Data Structuring - Data Types

- Continuous media data itself (audio, video, ...)
- Meta-data (attributes):
  - Annotations by the author
  - Associations between related files (synchronization)
  - Linking and sharing of data segments:
    - e.g., storing common parts only once

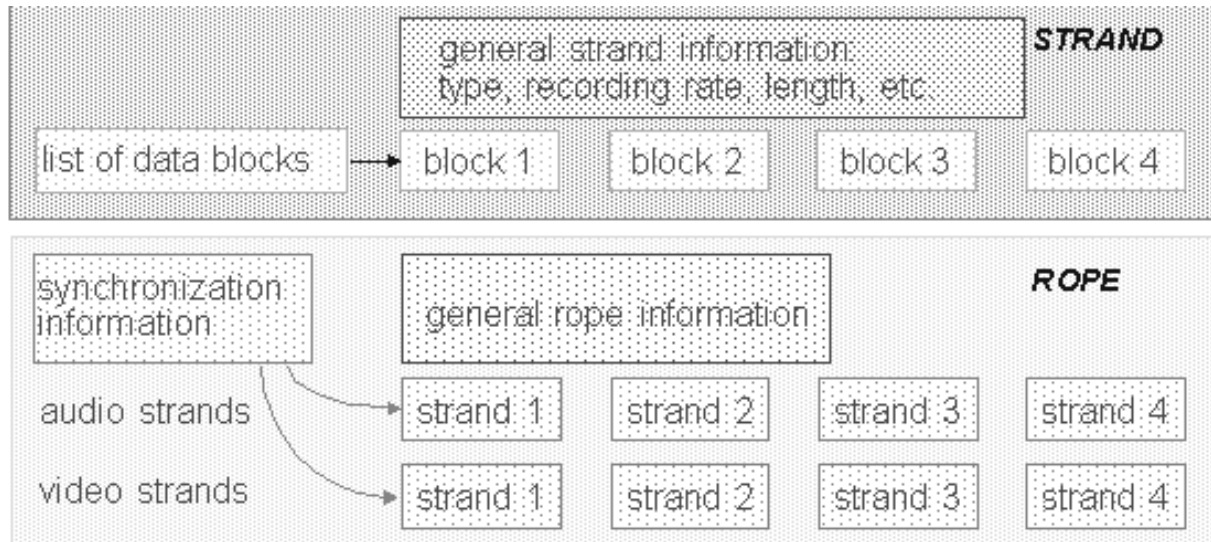
**Frame: Basic unit of video**

**Sample: Basic unit of audio**

**Block: Basic unit of disk storage**

- Homogeneous block:
  - each block contains exactly one medium
  - requires explicit inter-media synchronization
- Heterogeneous block:
  - multiple media stored in one block
  - implicit inter-media synchronization

## Data Structuring: Terminology



### **Strand:**

- An immutable sequence of continuous video frames or audio samples

### **Rope:**

- A collection of multiple strands tied together with synchronization information

### **Strands are immutable:**

- Copy operations are avoided
- Editing on ropes manipulates pointers to strands
- Many different ropes may share intervals of the same media strand
- Reference counters, storage reclaimed when not referenced anymore

## Operations on Multimedia Ropes: Interface

### Example

RECORD [media] → [requestID, mmRopeID]

- Record a new multimedia rope consisting of media strands
- Perform admission control to check resource availability

PLAY [mmRopeID, interval, media] Æ [requestID]

- Playback a multimedia rope consisting of media strands
- Perform admission control to check resource availability

STOP [requestID]

INSERT [baseRope, position, media, withRope,  
withInterval]

REPLACE [baseRope, media, baseInterval, withRope,  
withInterval]

DELETE [baseRope, media, interval]

(from Rangan and Vin, 1991)

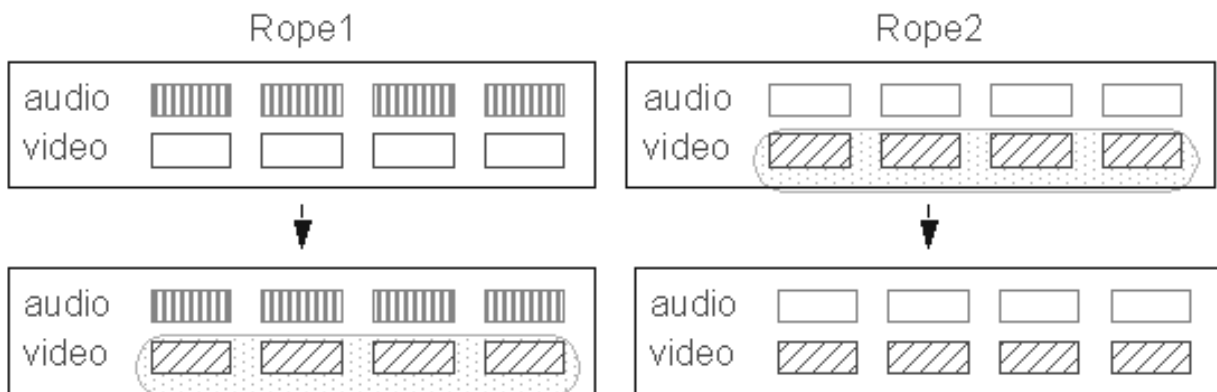
A Graduate Course on Multimedia Technology	© Wolfgang Effelsberg, Ralf Steinmetz	6. Media Server	6-51
---	--	-----------------	------

## Operations on Multimedia Ropes: Example

### Merge audio and video strands:

- Rope1 contains only audio strand
- Rope2 contains only video strand
- Replace (non-existing) video component of Rope1 with video component of Rope2

REPLACE[baseRope: Rope1, media: video,  
baseInterval: [start:0, length: I],  
withRope: Rope2, with Interval: [start:0,  
Length: I]]



# Video File Server: System Structure

## Two major components

- Lower level storage manager
- Higher level rope server

## Multimedia storage manager

- Physical storage of media strands on disk
- Admission control
- Maintains disk layout

## Multimedia rope server

- Performs operations on multimedia ropes
- Communicates with storage manager via inter-process communication mechanisms
- Receives status messages from storage manager and
- sends status messages to application

A Graduate Course on Multimedia Technology	© Wolfgang Effelsberg, Ralf Steinmetz	6. Media Server	6-53
---	--	-----------------	------