



# CS423 Operating Systems and Multimedia Issues

---

# Multimedia Resource Management

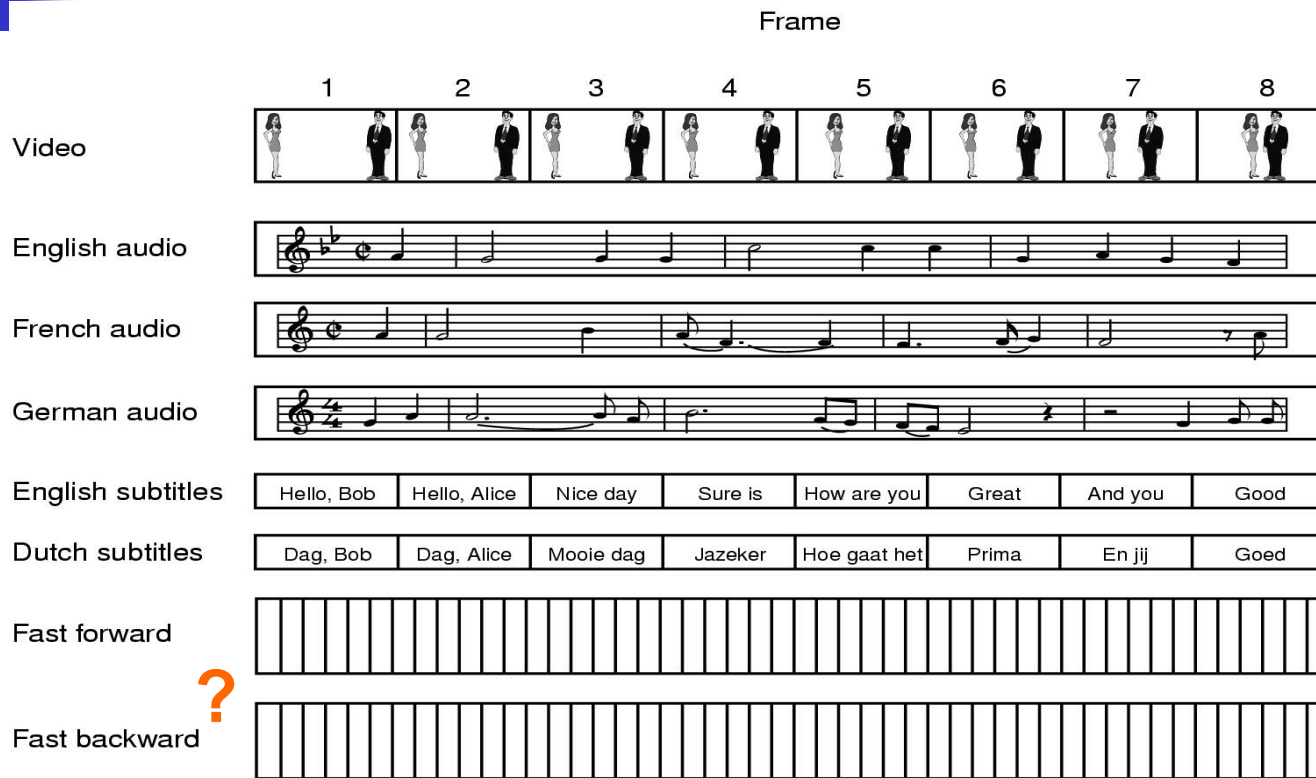


---

- On the client side:
  - Real-time scheduling
  - Non-interrupted playback
- On the server side (e.g., video on demand servers):
  - Maximum throughput delivery
  - Soft real-time operation



# Multimedia Files



A movie may consist of several files



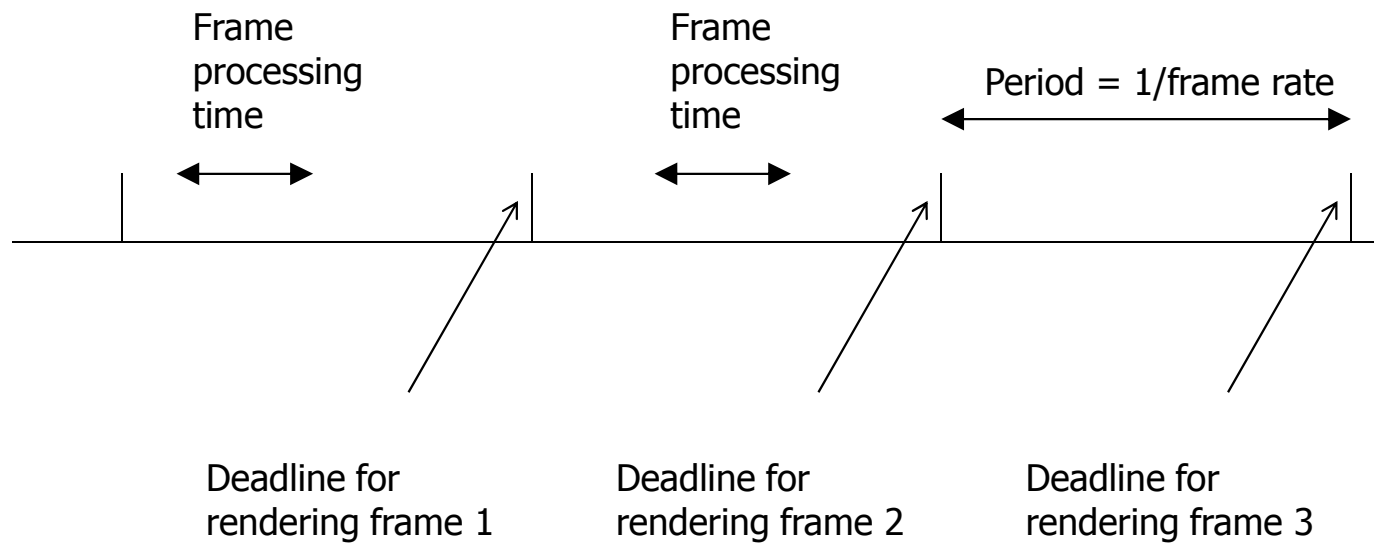
# Scheduling Environments

---

- Traditional scheduling on time-sharing computers
  - optimal throughput, optimal resource utilization, and fair queuing.
- Hard RT scheduling
  - guarantee *hard deadlines* in the worst case.
- Soft RT scheduling
  - *Guarantee soft deadlines* that are met with high-probability in a dynamic environment with adaptation to changes of the workload
- In multimedia: frames have deadlines (no drops, no delays, or video “freezes”)

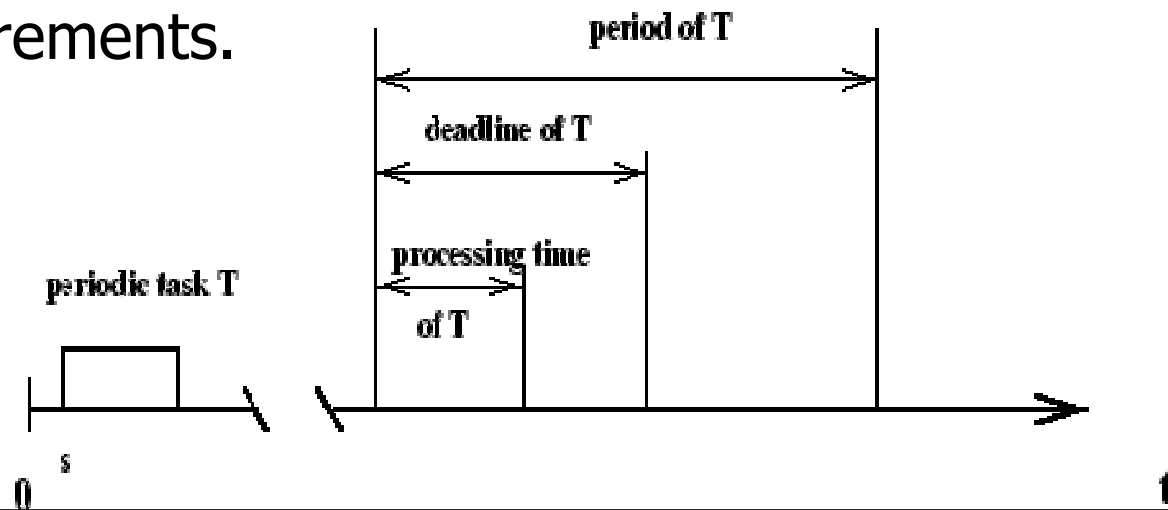


# Movie Playback



# System Model

- Consider a task invocation to be the rendering of a frame on screen; a movie is a periodic task
- Tasks are the schedulable units of the system.
- Each task has a timing constraint and resource requirements.





# Assumptions and Notation

---

- **Assumptions** : periodic tasks are mutually independent (playback of one video stream does not depend on another).
- **Time Constraints:**  $(e, d, p)$  where  $e$  is processing time,  $d$  is deadline, and  $p$  is period.
- **Multimedia Scheduling Algorithm:** must determine sharing of the resources used by different tasks so that all of them can meet their deadlines (all frames displayed at the right times)



# Rate Monotonic Scheduling

---

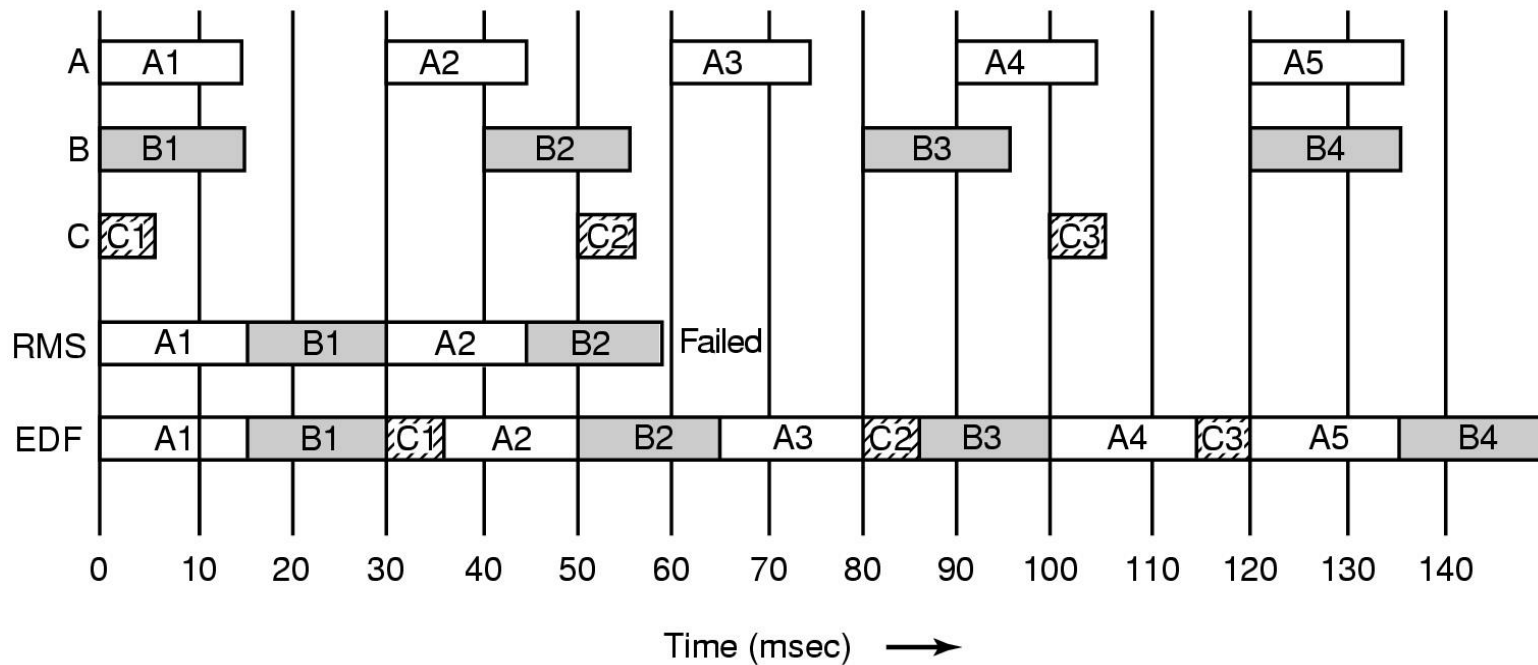
- This algorithm was designed and proved correct by C.L. Liu and Layland in 1973.
- This algorithm is static and optimal, priority-driven for preemptive periodic jobs.
  - Optimal means that there is no other static algorithm that is able to schedule a task set which can't be scheduled by the rate-monotonic algorithm.
- The algorithm was proven under the following assumptions:
  - tasks are periodic
  - each task invocation must be completed before the next request occurs
  - all tasks are independent
  - run-time of each task request is constant
  - any non-periodic task in the system has no required deadlines

# Earliest Deadline First Algorithm



- One of the best known algorithms for RT processing
- It is an optimal dynamic algorithm. It produces a valid schedule whenever one exists.
- Upper bound of process utilization is 100%.
- If priorities should be used, then the earliest deadline gets the highest priority.

# Comparison



Example of real-time scheduling with RMS and EDF

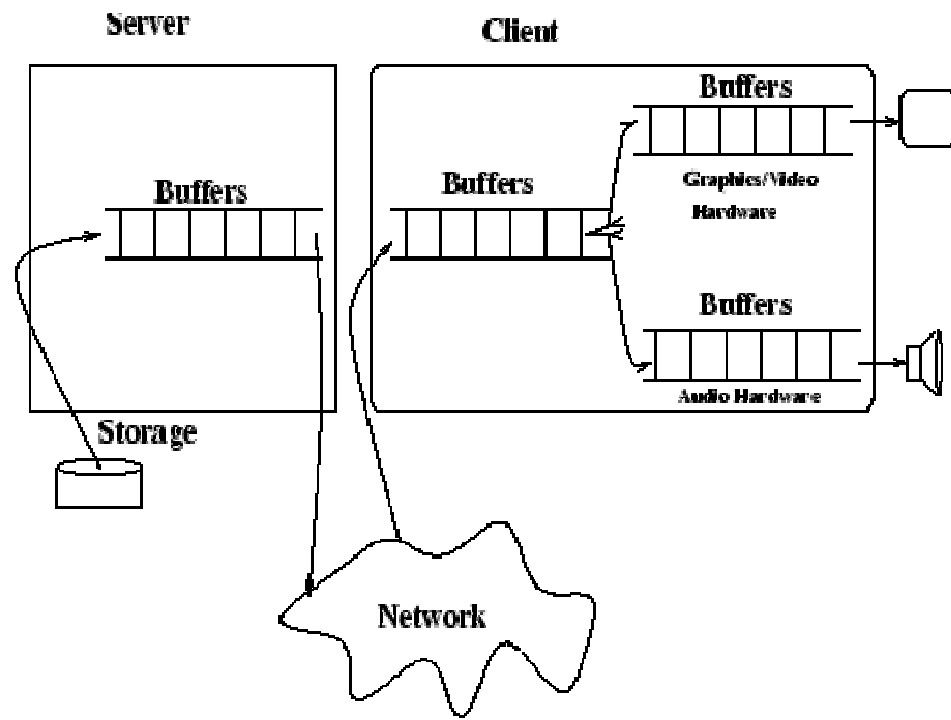
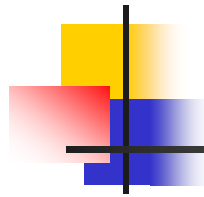
# Condition for Meeting Deadlines

- The RM schedulability test is

$$\sum_{i=1}^n \frac{e_i}{p_i} \leq \ln 2 \quad U \leq \ln 2$$

- With EDF algorithm, the 100% utilization can be achieved :  $U \leq 1$
- The schedulability test is then  $\sum_{i=1}^n \frac{e_i}{p_i} \leq 1$

# Data Flow for a Multimedia Server



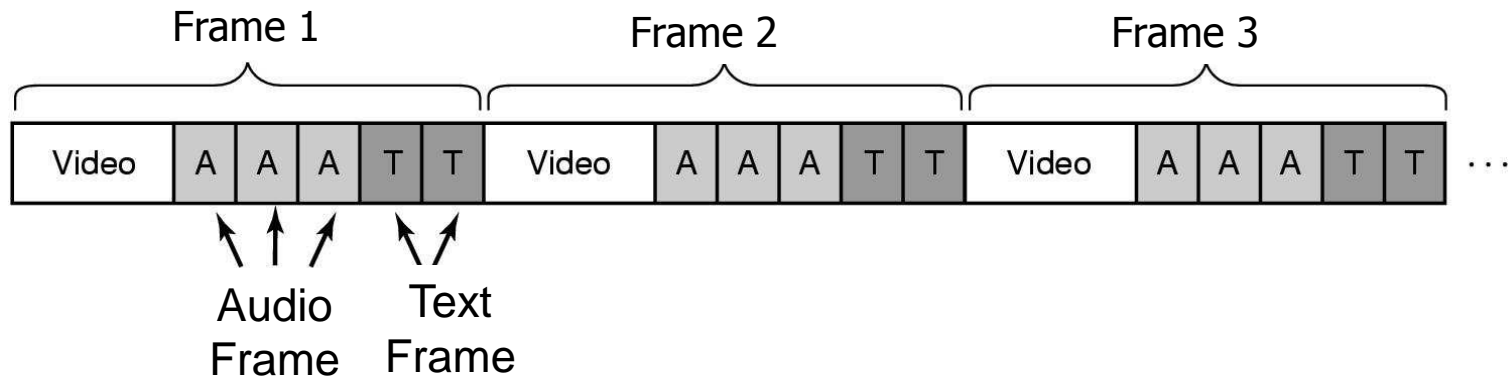


## Management Multimedia Disk Storage

---

- Optimally place data blocks on the disk.
- Use multiple disks.
- Build storage hierarchies.

# Single Disk File Placement (\*)



## Placing a File on a Single Disk

### ■ Interleaving

- Video, audio, text in single contiguous file per movie
- Why?



# Placement of Data Blocks of a File on a Single Disk

---

- Discussion
  - What are the advantages and disadvantages of continuous placement?



# Tradeoff

---

- Continuous Placement

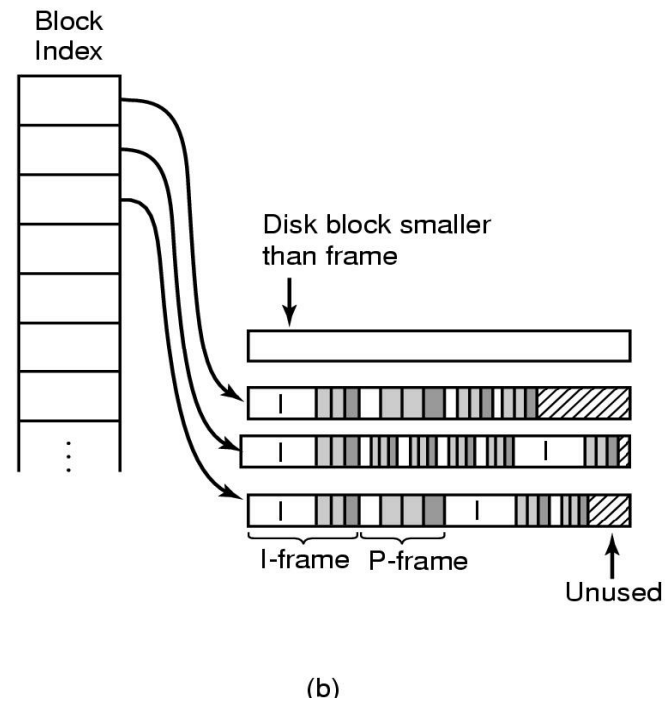
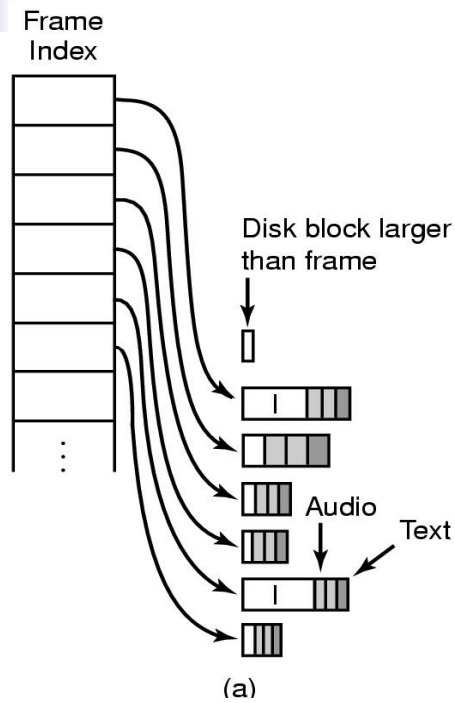
- Pros

- Simple to implement
    - When reading a file, only 1 seek is needed
      - Scattered placements may require a seek for each blockhead---Intrafile seek.
      - Does this advantage still hold for
        - Multi-stream?
        - Video editing?

- Cons

- Unnecessary reads

# Two Alternative File Organization Strategies

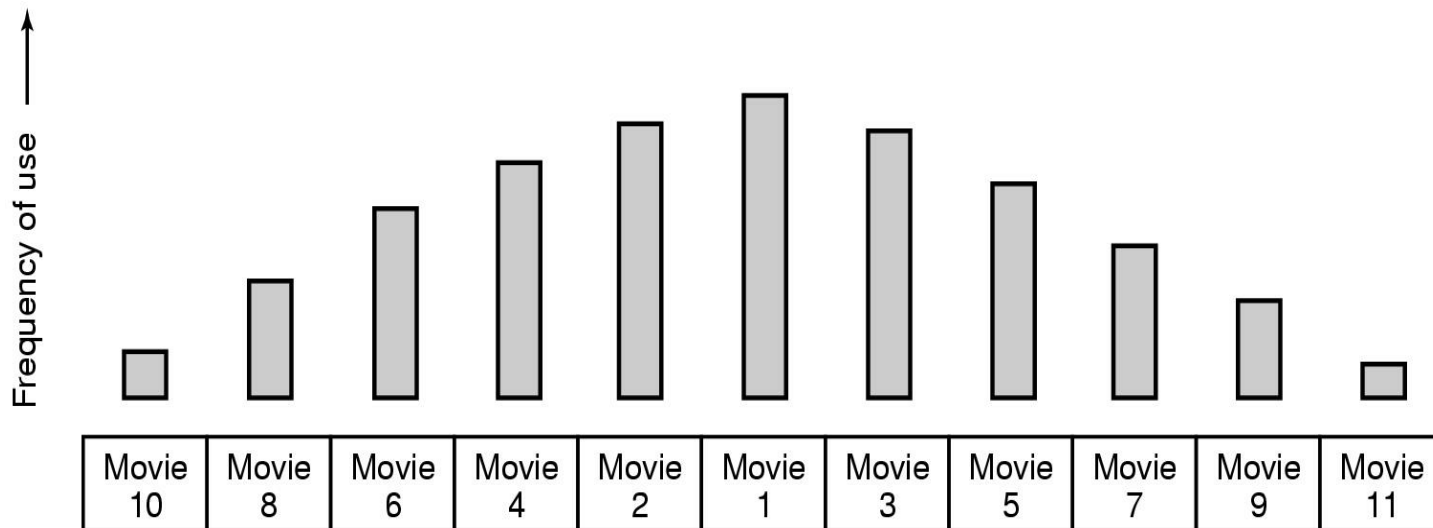


Noncontiguous Movie Storage  
(a) small disk blocks  
(b) large disk blocks

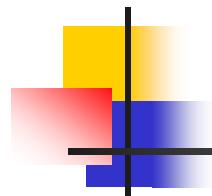
# File Organization Strategies

- **Small-block Organization**, called also *constant time length*, uses small disk blocks and organizes all data per frame. This organization uses a *frame index* per movie, where each entry points to the start of the frame. Each frame consists of video, audio, text tracks for that frame.
- **Large-block Organization**, called also *constant data length*, uses large disk blocks, and puts several frames into one block. This organization uses *block index* to point to each block.
- Comparison: Large-block organization suffers from internal fragmentation, small-block organization yields little waste of disk space. On the other hand, large-block organization has smaller index than the small-block organization.
  - A variation: Start blocks at beginning of new "scenes"

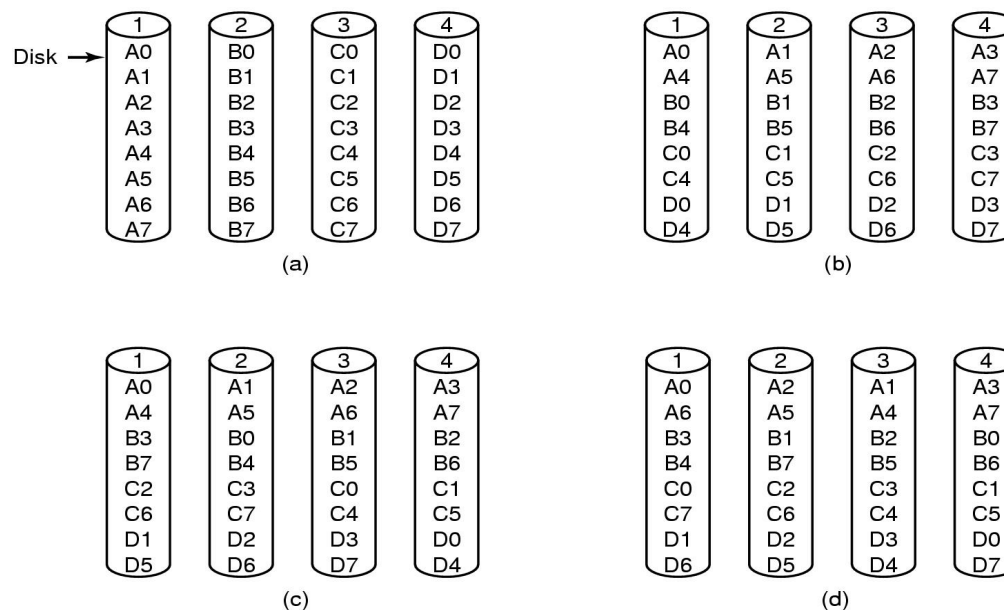
# Placing Multiple files on a Single Disk



- Organ-pipe distribution of files on server
  - most popular movie in middle of disk
  - next most popular either on either side, etc.

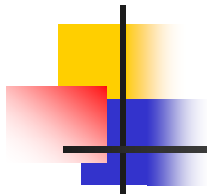


# Placing Files on Multiple Disks

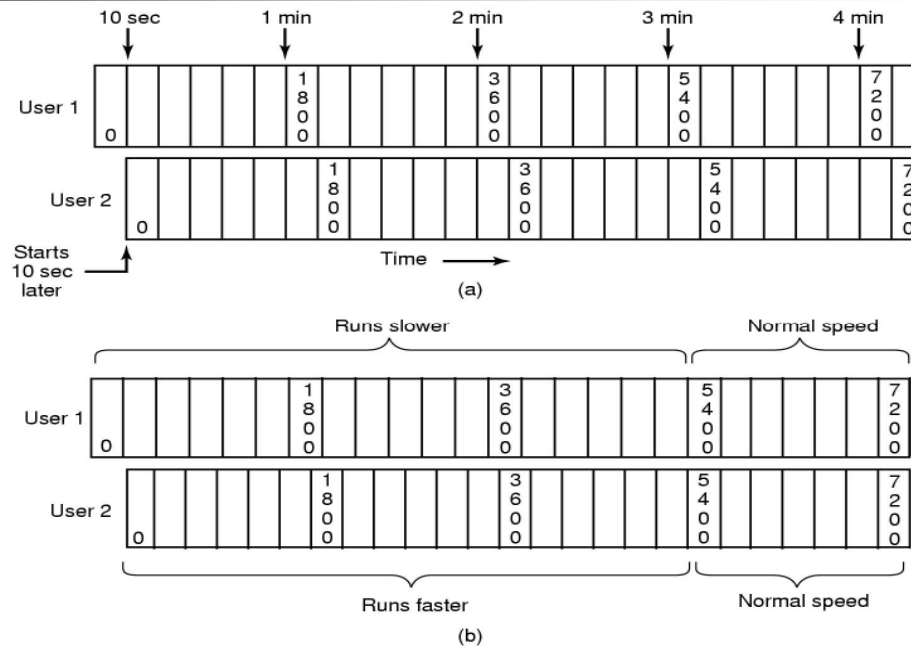


Organize multimedia files on multiple disks

- (a) No striping
- (b) Same striping pattern for all files
- (c) Staggered striping
- (d) Random striping



# Caching



## Block Caching

(a) Two users, same movie 10 sec out of sync

(b) Merging two streams into one



# File Caching

---

- Most movies stored on disk
  - copy to memory when needed
  - results in large startup time
  - prefetch/keep most popular movie parts in memory
- Can keep first few min. of all movies in memory
  - start movie from this while remainder is fetched



# Disk Scheduling Algorithms

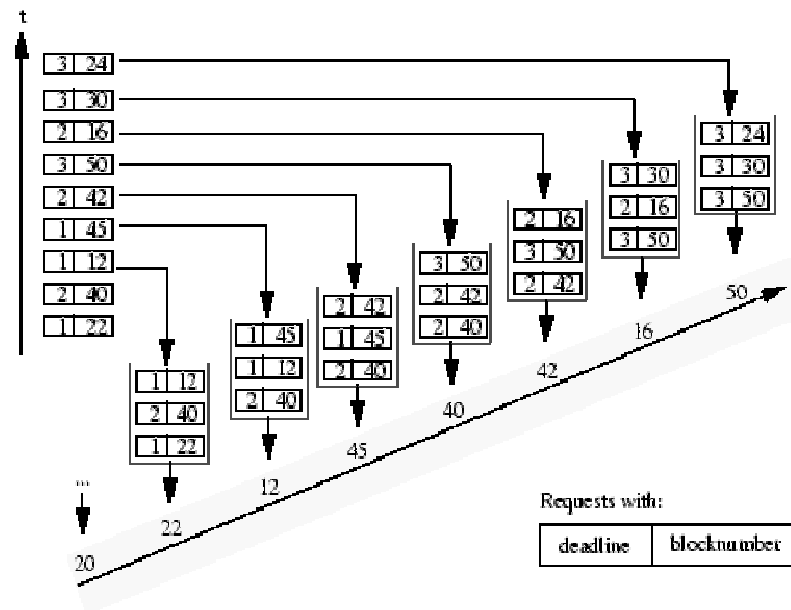
---

- *Goal of scheduling in traditional file systems*
  - reduce cost of seek time
  - achieve high throughput
  - provide fair disk access;
- *Goal of scheduling in multimedia file systems:*
  - meet **deadlines** of all time-critical tasks
  - keep the necessary buffer requirements low
  - serve many streams concurrently
  - find balance between time constraints and efficiency.



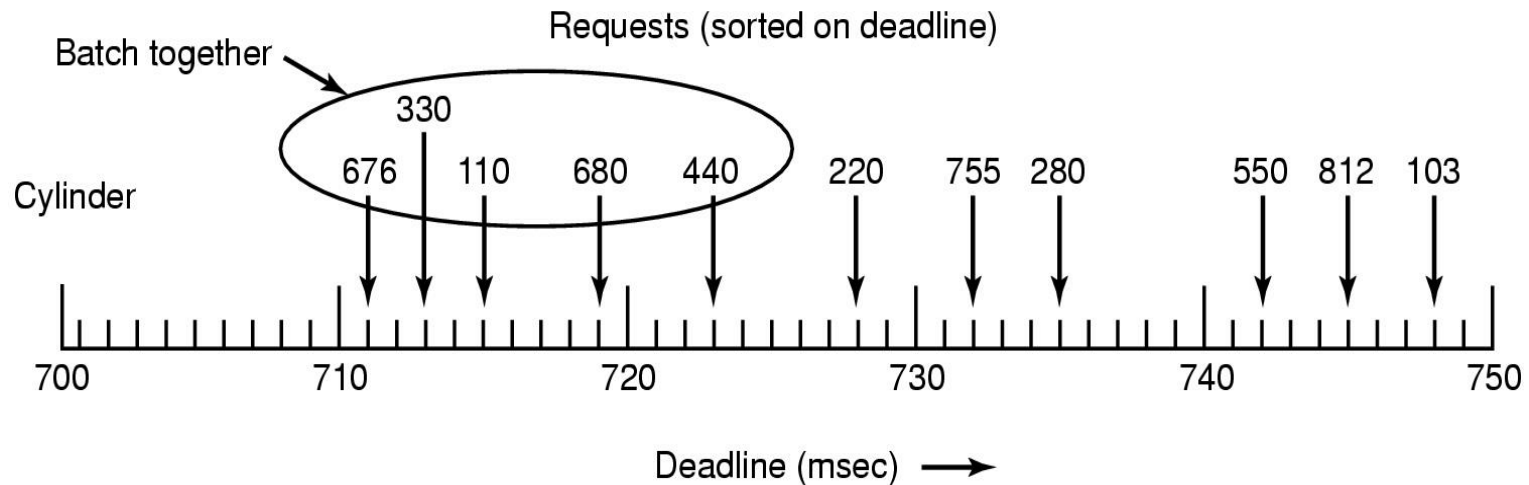
# EDF Scheduling Algorithm

- **Earliest Deadline First**
- Problem: poor throughput and excessive seek time.





# Scan-EDF Scheduling



- Scan-EDF algorithm
  - uses deadlines & cylinder numbers for scheduling