Audio / Video Playback: Task Scheduling and Jitter Buffers

N. C. State University

CSC557 ♦ Multimedia Computing and Networking

Fall 2001

Lecture # 15

Some Definitions

- *Tasks* in multimedia
  - recording or "capture" (conversion from analog to digital form)
  - retrieving a block of data from a storage device
  - compression
  - decompression
  - processing
  - transmission across a network
  - display or playback (conversion from digital to audio)
- *Repetitive* or *periodic* tasks
  - tasks that have to be done over and over for the duration of a video or a sound
  - ex.: capture and compress the next frame of video
  - ex.: send the next block of audio samples to a sound card

Definitions (cont'd)

- *Period* of a repetitive task
  - interval of time during which a new *instance* of a repetitive task is started or initiated
  - ex.: for playing a video, a new frame must be displayed every 1/30 of a second
- *Devices*
  - a component which performs a task
  - ex.: sound card
  - ex: processor
  - ex.: storage device
  - ex.: a video capture card
  - ex.: a router

Definitions (cont'd)

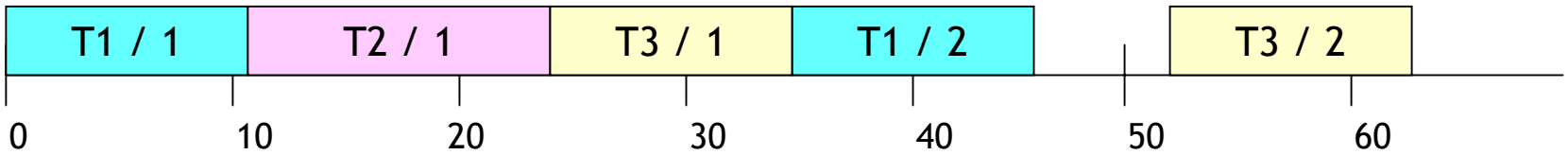
- Maximum *processing time* of a task
  - amount of time for a task to be completed by a device, without interruptions occurring
  - better be less than the period of the task!
  - Ex.: decompressing a frame of a MPEG-encoded video might take 1/60th of a second on processor X
- *Deadline* of a task
  - time after initiation of a task by which it must be completed
  - once video or audio capture / playback begins, you're on a treadmill!
  - ex.: the i^{th} frame of video must be decompressed / displayed by time i * 1/30 seconds after the video playback begins
  - tasks whose execution fails to meet their deadline have no "value"

Definitions (cont'd)

- *Preemptability* of tasks
  - possibility of suspending, then resuming, task execution
  - ex.: "time-slicing" in multiprogramming systems
  - ex.: servicing I/O interrupts
  - there is always overhead in suspending and resuming
  - some tasks may be non-preemptable
- *Utilization* of a device
  - a task which uses a device for t_1 time units every period of t_2 time units utilizes that device $t_1/t_2*100\%$ of the time
  - ex.: if decompressing a frame of video takes $1/60^{th}$ of a second, and occurs every $1/30^{th}$ of a second, processor utilization for decompressing the video = 1/60 / 1/30 = 50%
  - total processor utilization from all tasks must not exceed 100%!

Definitions (cont'd)

- Task execution *schedule*
  - a schedule of the times at which a device is used, and by which tasks
  - Ex.: T2 / 1 = task 2, instance 1



- Schedulability of a set of tasks
  - A set of tasks is "schedulable" if they can be arranged to execute such that all tasks meet their deadlines
- Finding a feasible schedule of tasks
  - Create a schedule under which all tasks meet their deadlines

Absolute vs. Relative Time

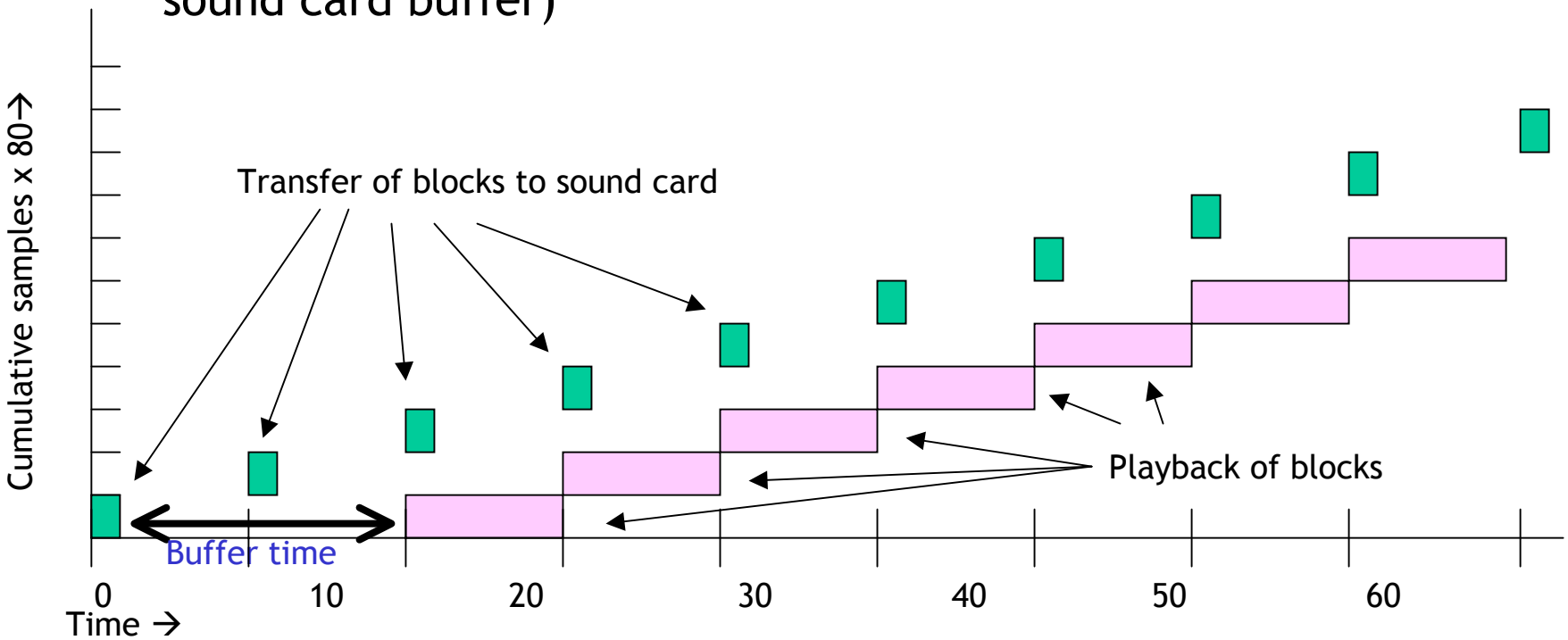
- Absolute time reference
  - advantage: simplicity
  - example: SMPTE Time Code (hh:mm:ss:ff)
  - example: Universal Time Code (UTC)
- Relative time reference
  - advantage: flexibility
  - example: "after the 10th scene change in the video"
  - example: "when the song finishes"

Quality of Service (QoS) Requirements

- What Is "good enough" video / audio playback?
- Requirement #1
  - a task must be processed before some maximum delay has elapsed
- Requirement #2
  - the *variation* in starting times of successive task instantiations must not exceed some maximum
  - Variation = "jitter" (more later)
- Requirement #3
  - the percentage of tasks which fail to be executed must not exceed some maximum amount

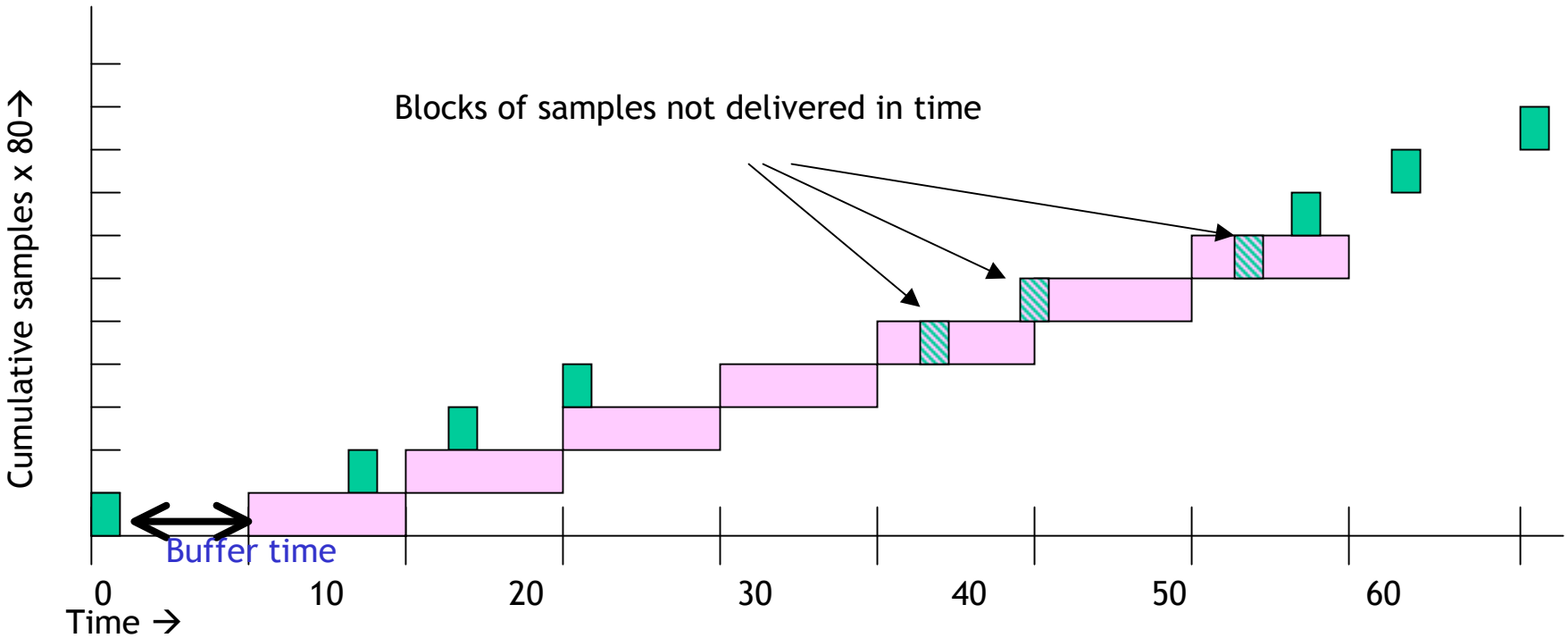
Jitter

- Audio playback
  - every 10ms interval…
    - read one block of 80 audio samples from a storage device
    - transfer this block to a memory buffer on the sound card
  - play this block of samples in some later 10ms interval (empties the sound card buffer)



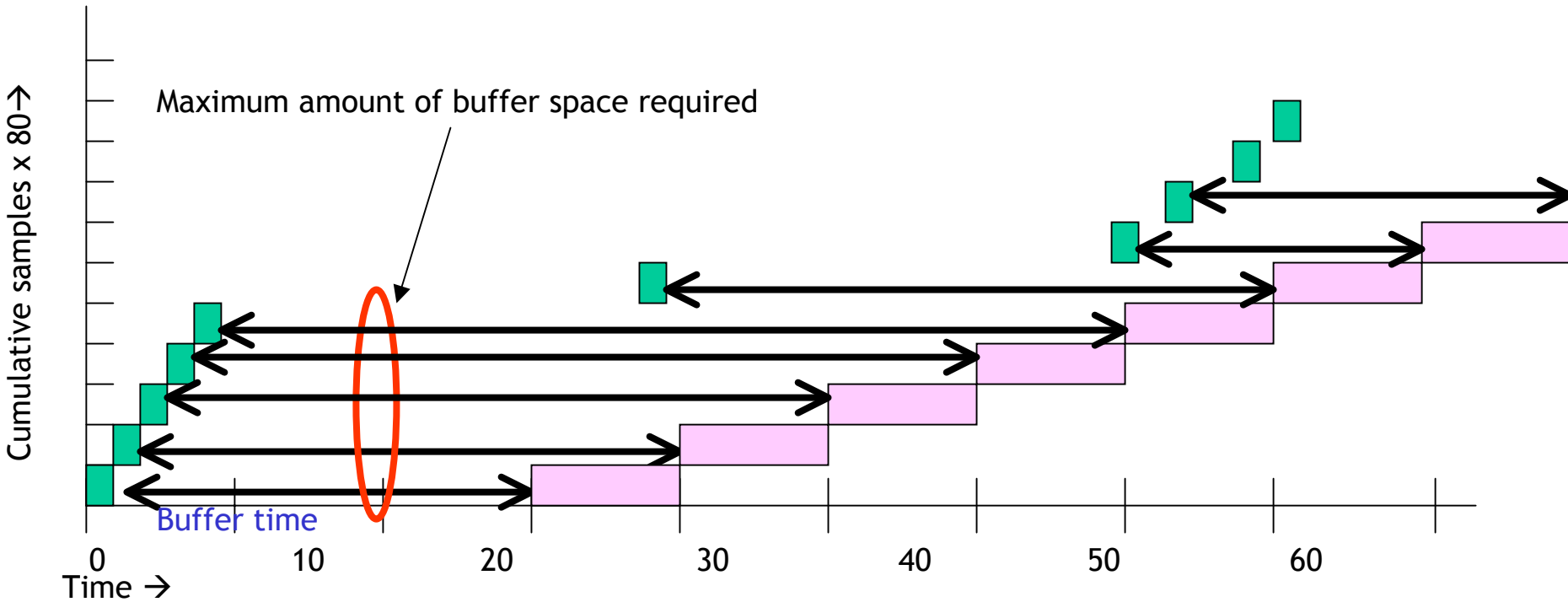
Reducing Jitter

- Problems that are possible
  - reading the block is delayed or interrupted; won't make it to the sound card in time! Or,
  - Reading occurs so fast, that the sound card buffer will fill up!



Jitter

- Dilemma
  - Delay playback long enough to avoid starvation
  - Don't delay playback longer than the size of the buffer
  - Don't delay playback of interactive conversation!



Delay Requirements

- For stored video/audio playback
  - delay as long as needed
- For interactive video/audio
  - 150 msec one-way delay
  - longer interferes too much with interaction
- For audio/video synchronization
  - ~ 50 msec maximum drift

Dealing with QoS Violations

- Freezing the video / sound
  - obviously wrong
- Use jitter buffer, for delay
  - how estimate the size of the jitter buffer?
  - should jitter buffer size be adapted as delay varies?
- Forward error recovery (FEC) for missing data
  - send data more than once (use redundancy)
  - use extra copies to recover from missing bits
  - cost: increased storage and transmission bit rate
  - adaptive the forward error correction to the minimum necessary?

Dealing with Violations (cont.)

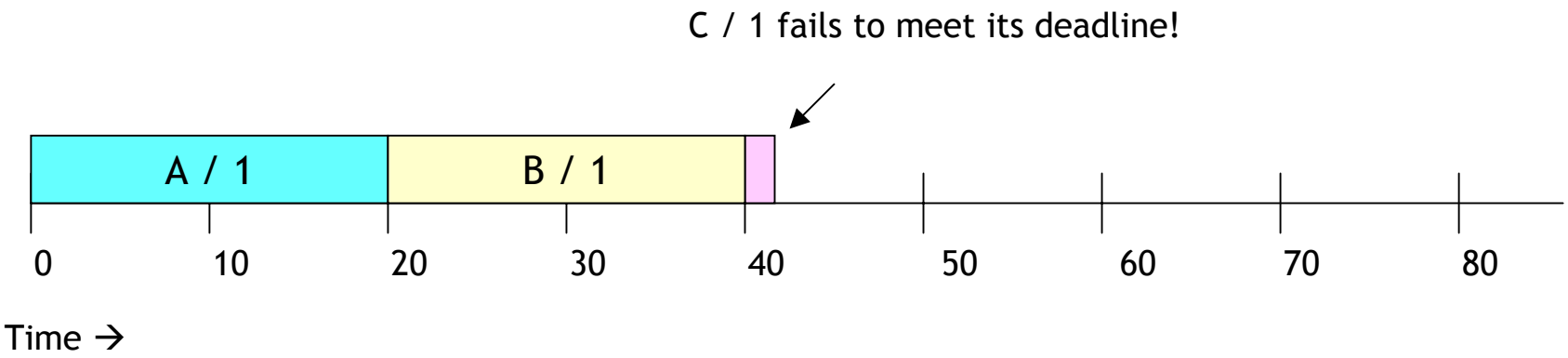
- Reduce bit rate by sacrificing quality
  - reduce video frame rate
  - increase quantization
  - reduce frame size!

An Example Task Set for Scheduling

NAME	TYPE	PERIOD	EXEC TIME	UTILIZTN
A	Video	60ms	20ms	.33
B	Video	45ms	20ms	.44
C	Audio	20ms	2ms	.10
D	Animation	100ms	5ms	.05
E	Text Scrolling	50ms	1ms	.02
TOTAL				.94

First-Come First Served (without Preemption)

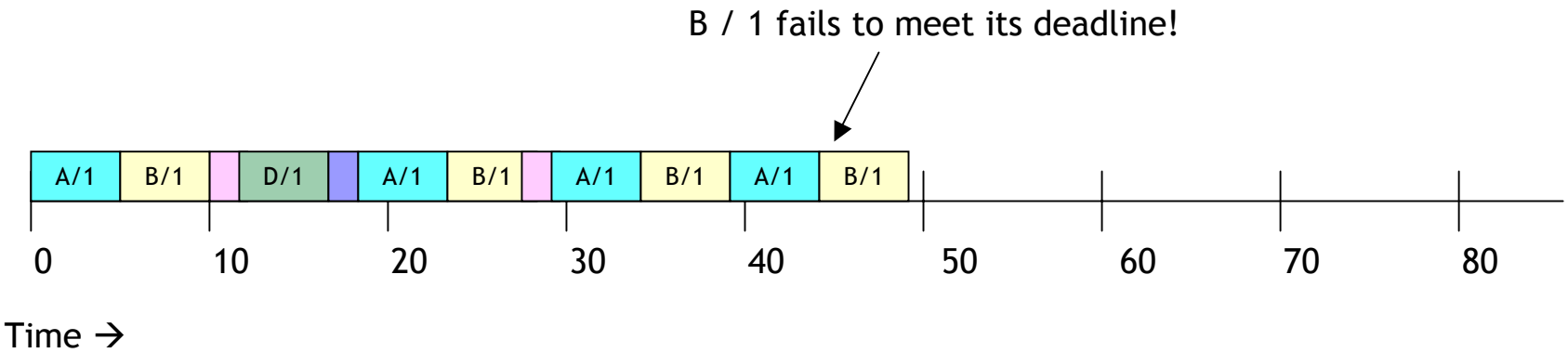
- Problem: no preemptions or priorities



NAME	TYPE	PERIOD	EXEC TIME	UTILIZTN
A	Video	60ms	20ms	.33
B	Video	45ms	20ms	.44
C	Audio	20ms	2ms	.10
D	Animation	100ms	5ms	.05
E	Text Scrolling	50ms	1ms	.02
TOTAL				.94

Round-Robin (Fixed Time Slices)

- Assume: time slice = 5ms
- Problem: no priorities



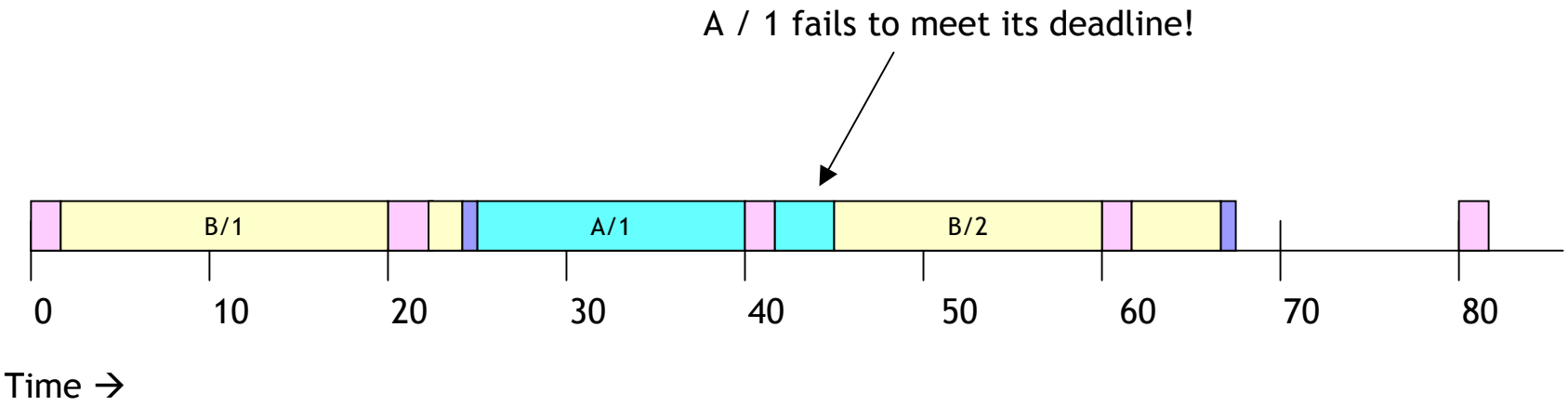
- Another problem: overhead of preemption

NAME	TYPE	PERIOD	EXEC TIME	UTILIZTN
A	Video	60ms	20ms	.33
B	Video	45ms	20ms	.44
C	Audio	20ms	2ms	.10
D	Animation	100ms	5ms	.05
E	Text Scrolling	50ms	1ms	.02
TOTAL				.94

Rate Monotonic Scheduling

- Priority-based preemptive scheduling
  - shortest period == highest priority, always
- An optimal static-priority scheduler
- Schedulability check
  - guaranteed schedulable if utilization < 69%
  - possibly schedulable if utilization > 69%
    - checking: start tasks simultaneously, check to earliest common deadline

Rate Monotonic Example

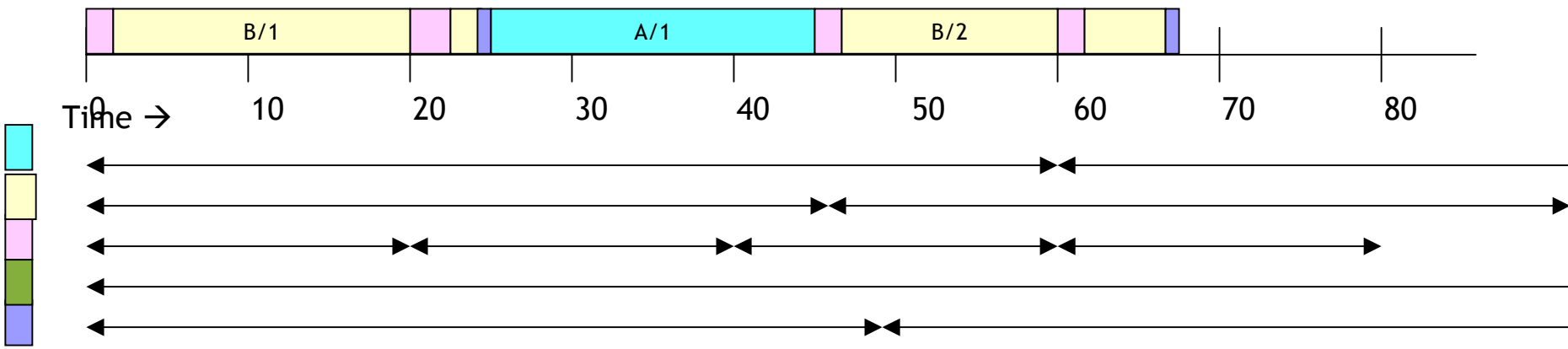


NAME	TYPE	PERIOD	EXEC TIME	UTILIZTN
A	Video	60ms	20ms	.33
B	Video	45ms	20ms	.44
C	Audio	20ms	2ms	.10
D	Animation	100ms	5ms	.05
E	Text Scrolling	50ms	1ms	.02
TOTAL				.94

Earliest Deadline First

- Priority-based preemption
  - Closest or "earliest" deadline == highest priority
- EDF = an optimal dynamic-priority scheduler
- Schedulability check
  - Guaranteed schedulable if utilization <= 100%
- More computation but fewer preemptions than rate monotonic

EDF Example



NAME	TYPE	PERIOD	EXEC TIME	UTILIZTN
A	Video	60ms	20ms	.33
B	Video	45ms	20ms	.44
C	Audio	20ms	2ms	.10
D	Animation	100ms	5ms	.05
E	Text Scrolling	50ms	1ms	.02
TOTAL				.94

Complications

- Pessimistic (conservative) results: worst-case execution time
  - How else can you get predictability
- Non-periodic tasks?
- Preemption overhead

Sources of Info

- *Multimedia systems*, by J. Buford, Chapter 7
- *Multimedia : computing, communications, and applications*, by R. Steinmetz
- On the web
  - http://research.microsoft.com/~mbj/papers/tr-2000-89.pdf
  - http://www-courses.cs.uiuc.edu/~cs314/Lectures/MMOS/Process/talk/talk.html
  - http://cs-www.bu.edu/faculty/best/crs/cs835/S96/lectures/scheduling.html