Resources Classification

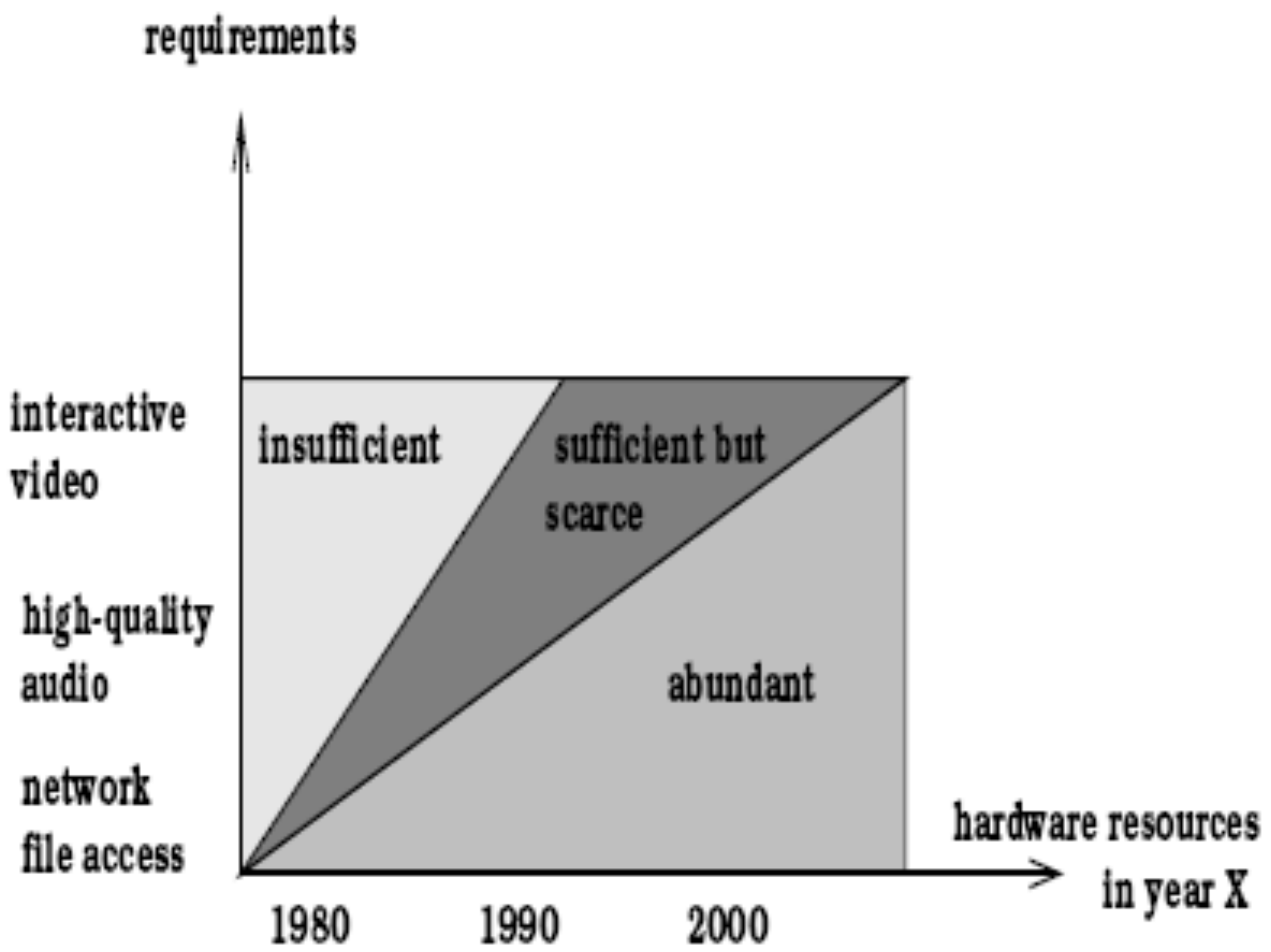
- Resource is a system entity required by tasks for manipulating data
  - Active versus Passive Resources (CPU – active; main memory – passive)
  - Shared versus Exclusive Resources (CPU – shared; audio device – exclusive)
  - Single Versus Multiple Resources (single CPU in PC, multiple CPU in multi-core system

Slides courtesy Prof. Nahrstedt

Resource Management (Why do we need resource management?)

- Limited capacity in digital distributed systems despite data compression and usage of new technologies
- Need adherence for processing of continuous data by every hardware and software component along the data path
- Competition for resources exist in an integrated multimedia system

Window of Insufficient Resources



Process abstraction

- Processes are programs in execution
  - Kernel keeps track of processes and resources assigned to a process using process control blocks
  - PCBs are saved and restored at context switch
- Schedulers choose the ready process to run
  - The choice of process to schedule depends on the scheduling criteria: fairness to all processes, fairness to important processes (IO bound, CPU bound etc.)
  - Real time goals are also an important class

Real-Time and Deadlines

- Real-time system – system in which correctness of computation depends not only on obtaining the right results, but also providing them on time
  - Examples: control of temperature in a chemical plant; control of a flight simulator
- Deadline – represents the latest acceptable time for the result delivery
  - Soft deadlines versus hard deadlines

Real-Time and Multimedia

- Difference between RT requirements for traditional RT systems and Multimedia systems
  - Fault tolerance and security
  - Soft deadlines versus hard deadlines
  - Periodic behavior versus random behavior
  - Bandwidth requirements

Real-time Processing Requirements

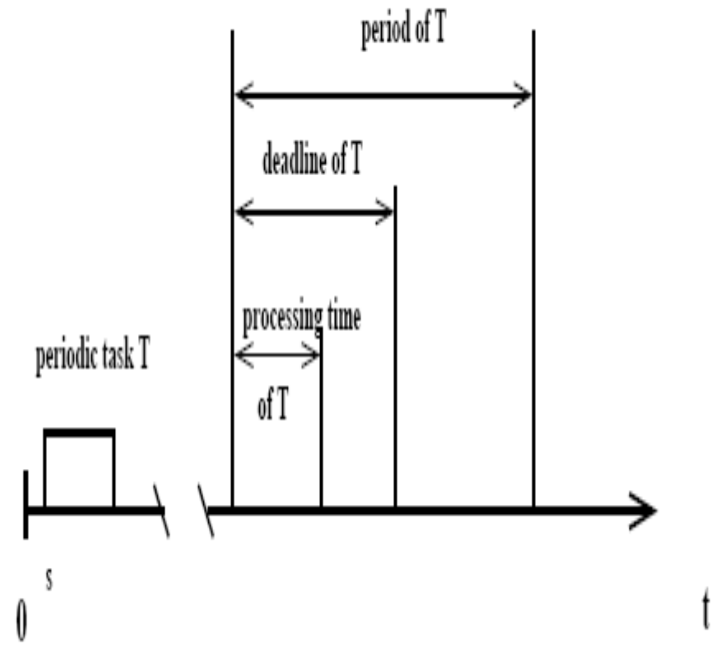
- Need to process continuous multimedia data
  - Processing occurs in predetermined, usually periodic intervals
    - E.g., 30 fps video requires a new frame every 1/30 sec.
  - Processing must be completed by certain deadlines
- Need RT process manager
  - Perform admission control
  - Determine schedule
  - Perform reservation
  - Schedule to give processing guarantees

RT Processing Requirements

- Main Problem: How to find a feasible schedule?
- Conflicting Goals/Problems:
- How do we schedule multimedia (RT) processes so that
  - non-RT processes do not starve when RT process is running
  - RT process is not subject to priority inversion

Model in RT Scheduling

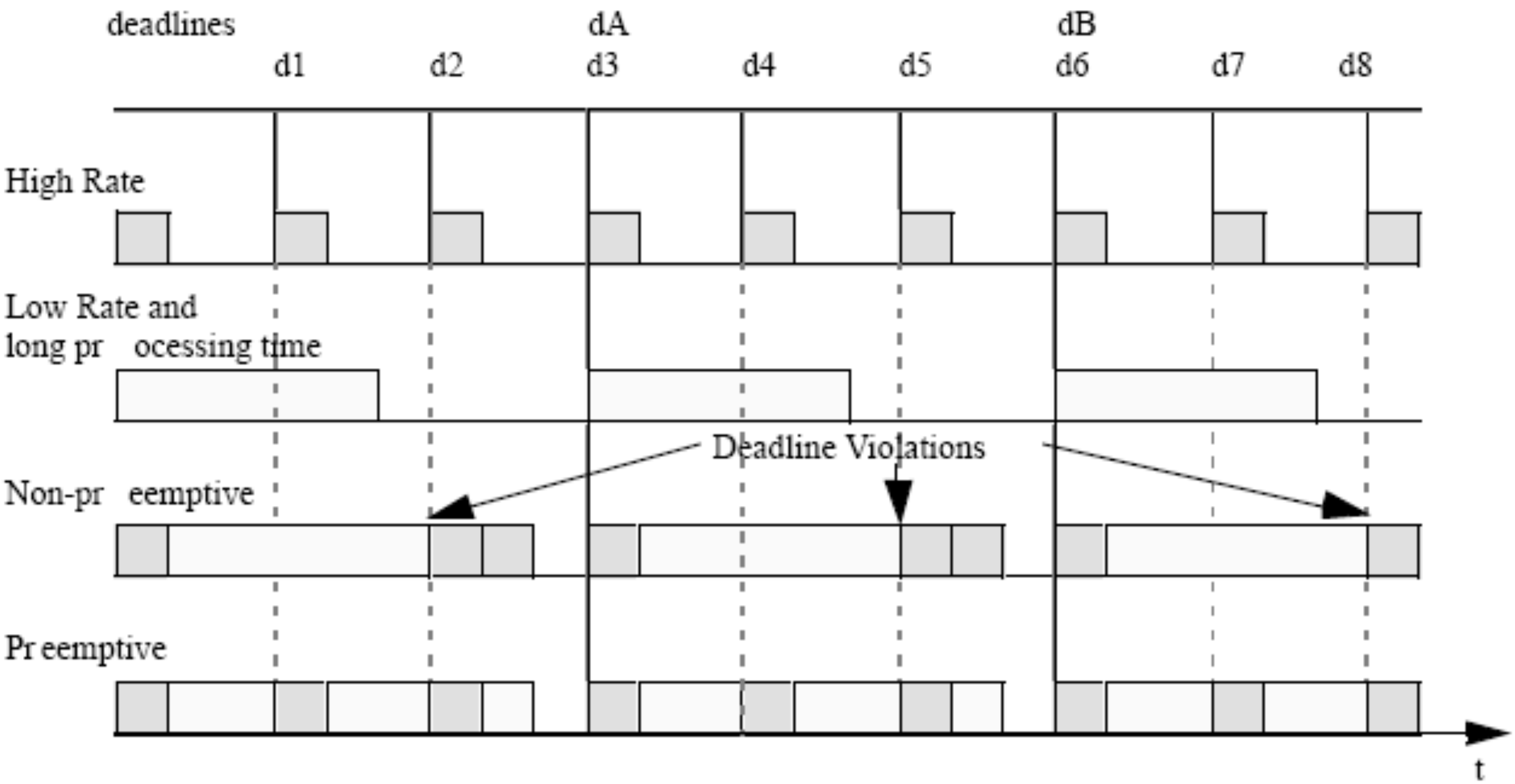
- Task (Process) – schedulable unit
- Task characterized by
  - **Timing constraints**
  - Resource requirements
- Assumptions
  - **Periodic tasks without precedence relations**
- Time constraints
  - s – task starting point
  - e – task processing time
  - d – task deadline
  - p – task period



Model of RT Scheduling

- Congestion avoidance deadline
  - If period at (k-1) step is equal to ready (start) time of period k
- Tasks: preemptive vs. non-preemptive
- Main goal of RT Scheduling:
  - find feasible schedule of all periodic tasks so that newly arriving task and all previous admitted tasks finish processing in every period according to their deadline

Preemptive vs. Non-preemptive Scheduling



Model of RT Scheduling

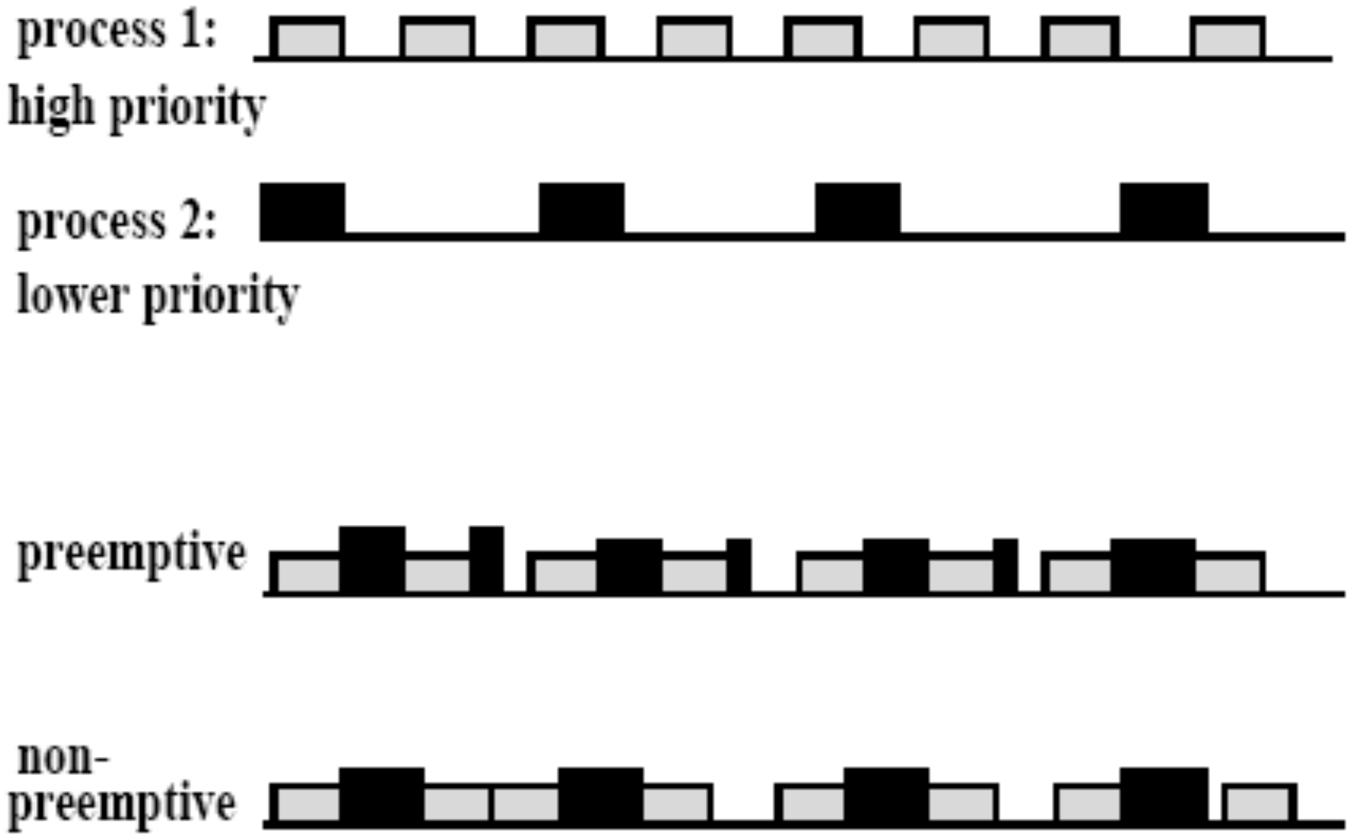
- Must have Schedulability (Admission) Test for RT tasks
- What is the performance metric for RT tasks?
  - Guarantee ratio := number of guaranteed tasks/total number of tasks
  - Process utilization (U):

$$U = \sum_{i=1}^{n} \frac{e_i}{p_i}$$

Scheduling Policies of RT Tasks

- Rate- Monotonic Scheduling (RMS)
  - Designed/proved by C.L. Liu and Layland 1973
  - Policy: task with highest rate has highest priority
  - Static and optimal, priority-driven
    - Optimal means that there no other static algorithm that is able to schedule a RT task which can't be scheduled by RMS algorithm
    - Assumptions:
      - Tasks are periodic
      - Each task must complete before next request
      - All tasks are independent
      - Run-time of each task request is constant
      - Any non-periodic task has no required deadline

Example of RMS

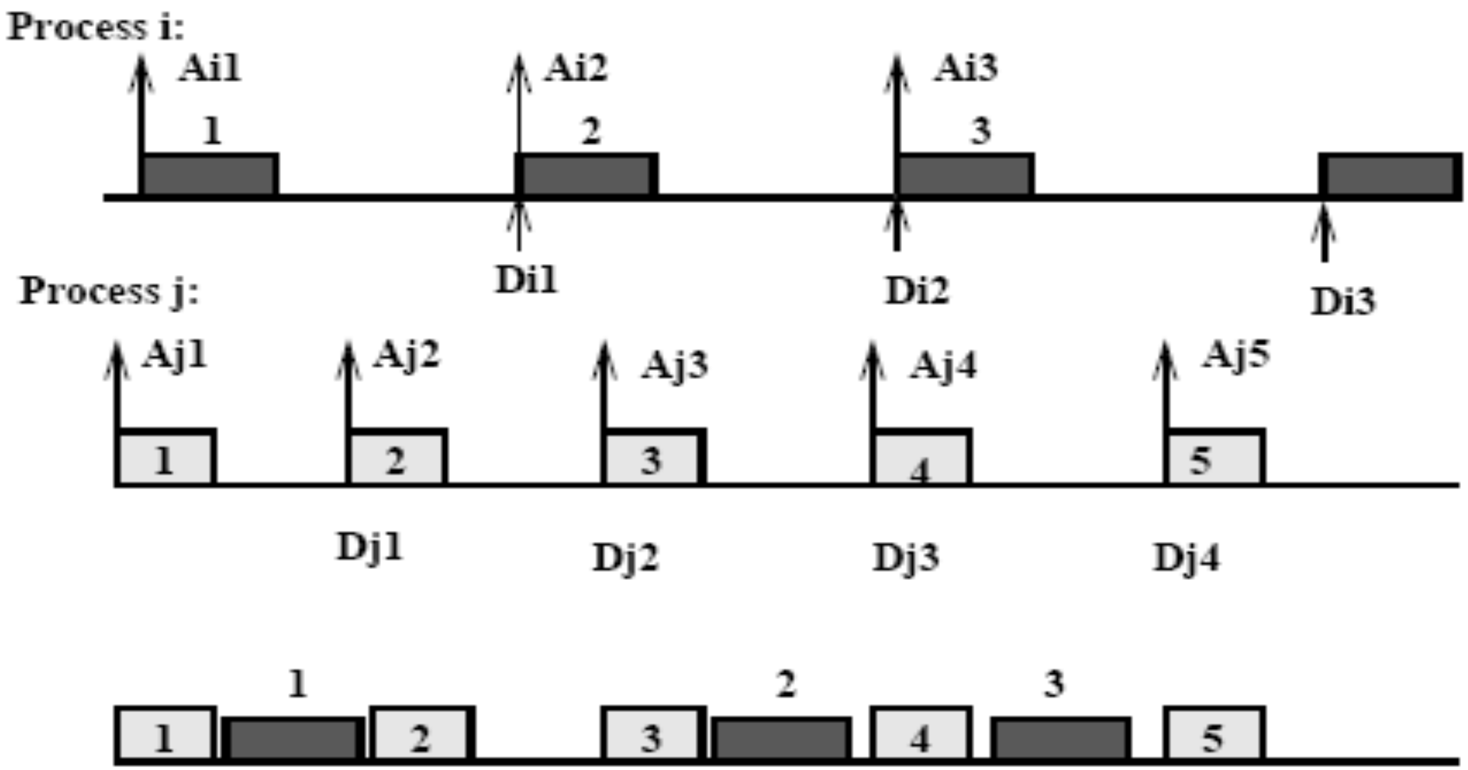


=> Relative Stream priority is fixed

Scheduling Policies for RT Tasks

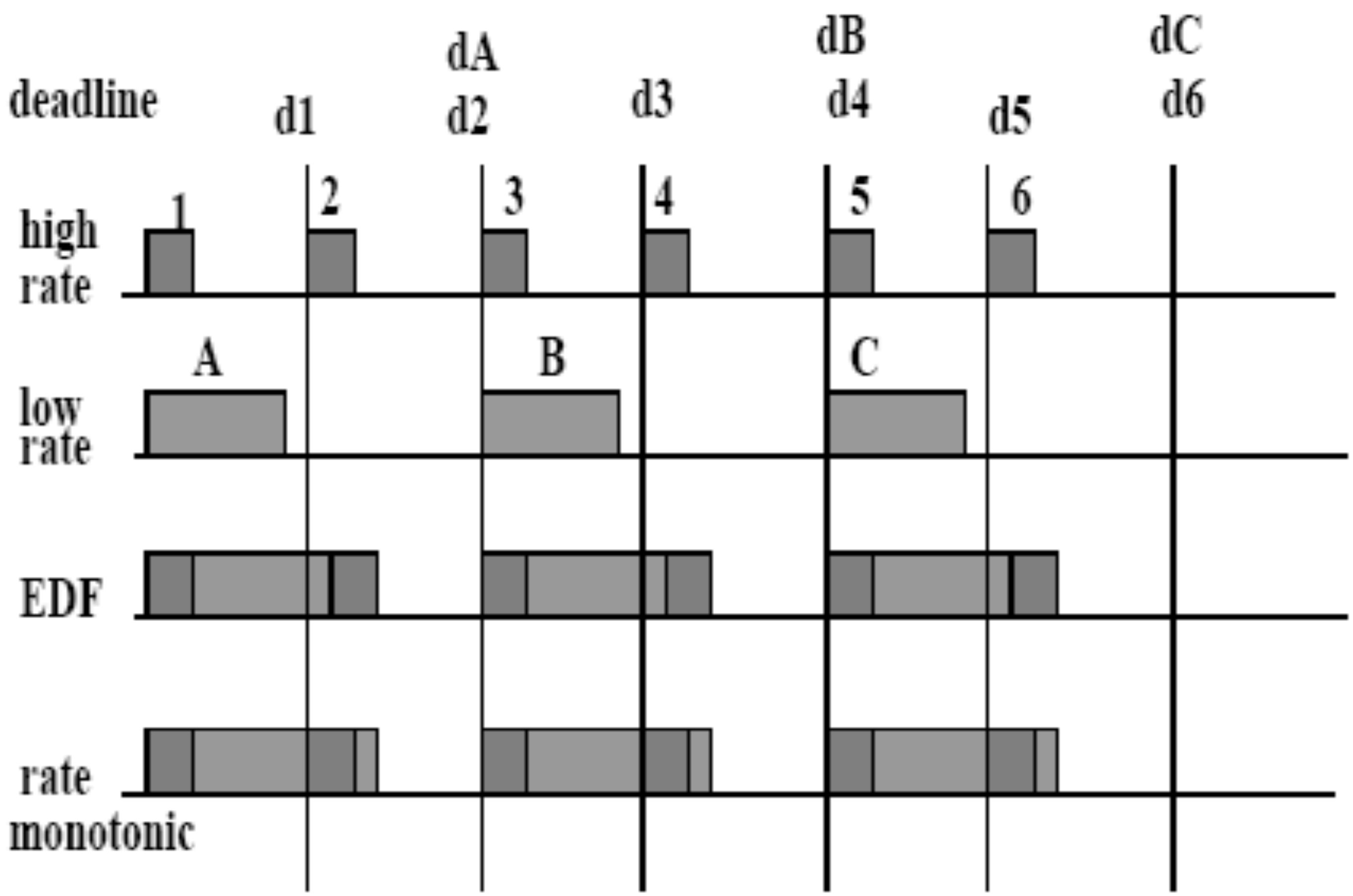
- Earliest Deadline First (EDF) Policy
  - Optimal dynamic algorithm
  - Produces valid schedule if one exists
  - Complexity O(n2)
  - Upper bound of process utilization 100%
  - Policy: task with earliest deadline has highest priority

Example of EDF



Both streams scheduled according to their deadlines

Comparison between RMS and EDF



Example

- Consider the following preemptive RT tasks and their characteristics
  - T1: p1 = 50ms, e1=10ms
  - T2: p2 = 100ms, e2=20ms
  - T3: p3 = 200ms, e3=50ms
  - T4: p4 = 100ms, e4=20ms
- Are these tasks schedulable via RMS?
  - If yes, what is the feasible schedule?
- Are these tasks schedulable via EDF?
  - If yes, what is the feasible schedule?

Conclusion

- RMS and EDF are basic policies for real-time scheduling systems
- For multimedia systems, soft-real-time scheduling (SRT) concepts needed, connecting reservation-based and adaption-based SRT

Other In-Time Scheduling Policies

- Least Laxity First Scheduling Policy
  - Policy: calculate laxity of tasks and task with shortest remaining laxity is scheduled first
  - Laxity lk := (s + (k-1)p + d) – (t + e)
    - k – kth period, t – actual time
  - Optimal dynamic algorithm
  - Problems:
    - Determination of laxity is inexact as algorithm assumes always worst case when calculating laxity
    - Laxity of waiting tasks changes over time, hence tasks can preempt each other several times without dispatching new task (high number of context switches)
    - Laxity calculation means additional overhead

DSRT (Dynamic Soft Real-Time Scheduling)

- CPU Service Classes
- Mapping CPU Service Classes into a Multiprocessor Partitioning Design
- Execution Flow of a SRT Process

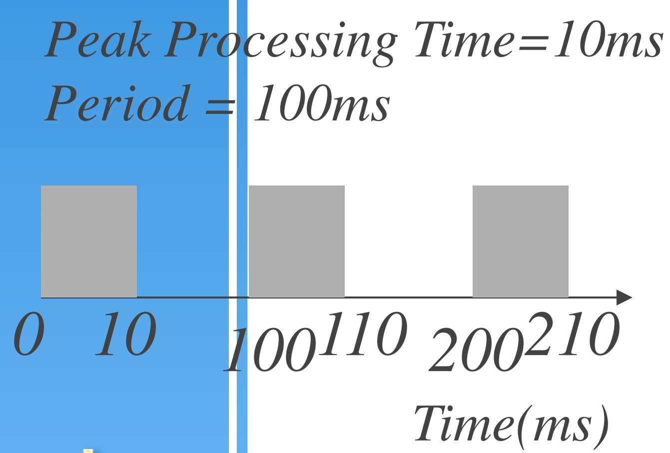
Source: Hao-hua Chu 1999

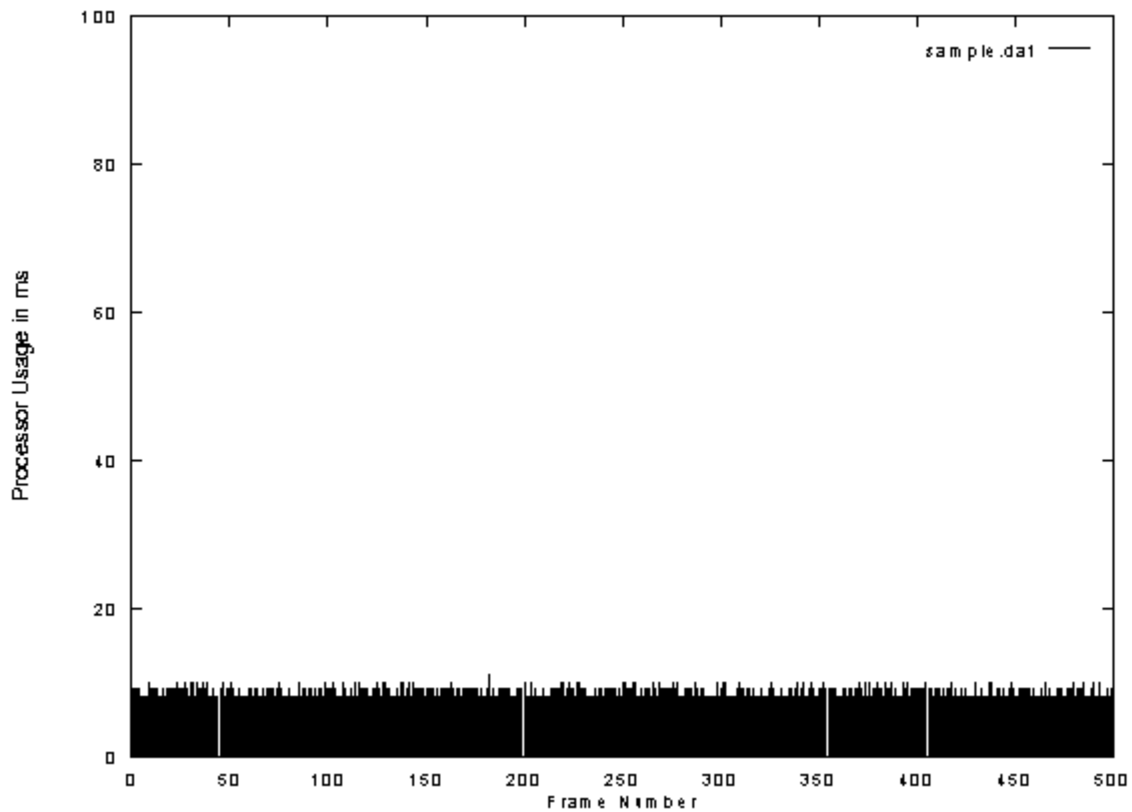
CPU Service Classes

Service Classes	*Specification Parameters*	*Guaranteed*
PCPT *(Periodic Constant Processing Time)*	***P*** *= Period* ***PPT*** *= Peak Processing Time*	***PPT***
PVPT *(Periodic Variable Processing Time)*	***P*** *= Period* ***SPT*** *= Sustainable Processing Time* ***PPT*** *= Peak Processing Time* ***BT*** *= Burst Tolerance*	***SPT***
ACPU *(Aperiodic Constant Processor Utilization)*	***PPU*** *= Peak Processor Utilization*	***PPU***
Event	*Relative Deadline* ***PPT*** *= Peak Processing Time*	***PPT***

Periodic Constant Processing Time Class

- Example Usage Pattern:

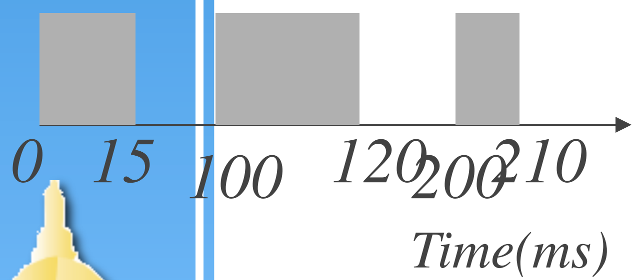


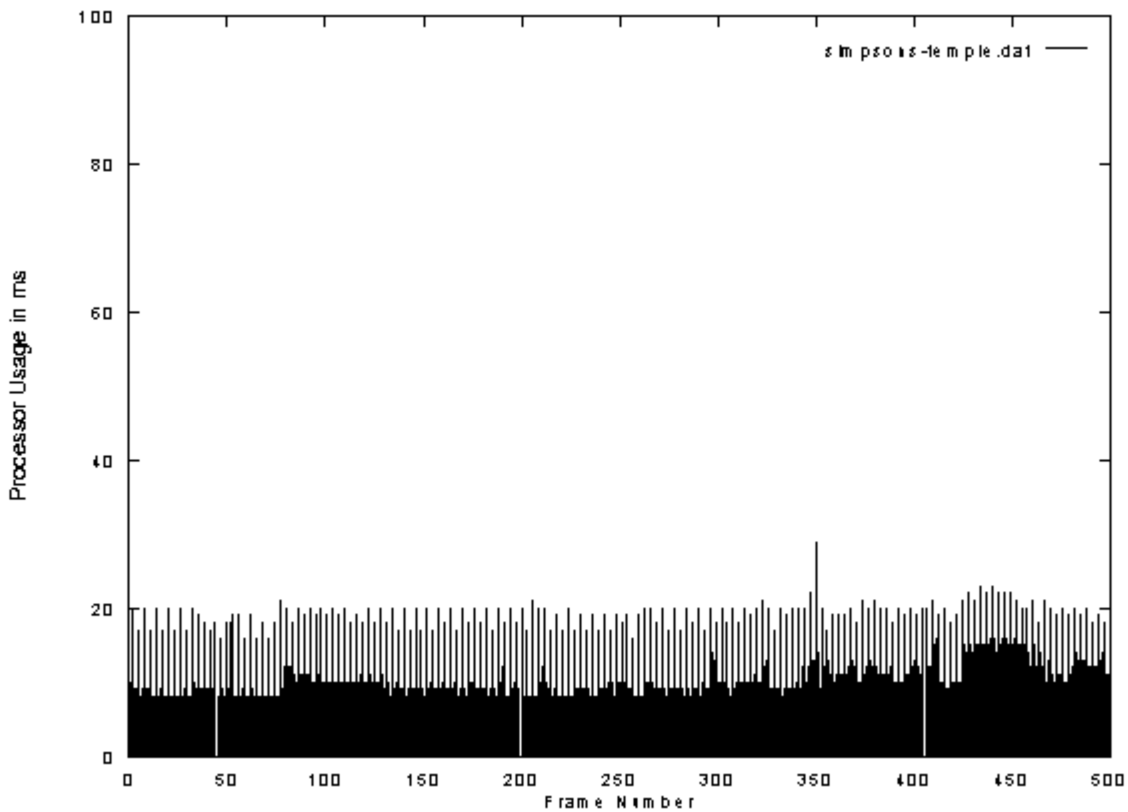


Periodic Variable Processing Time Class

- Example Usage Pattern:

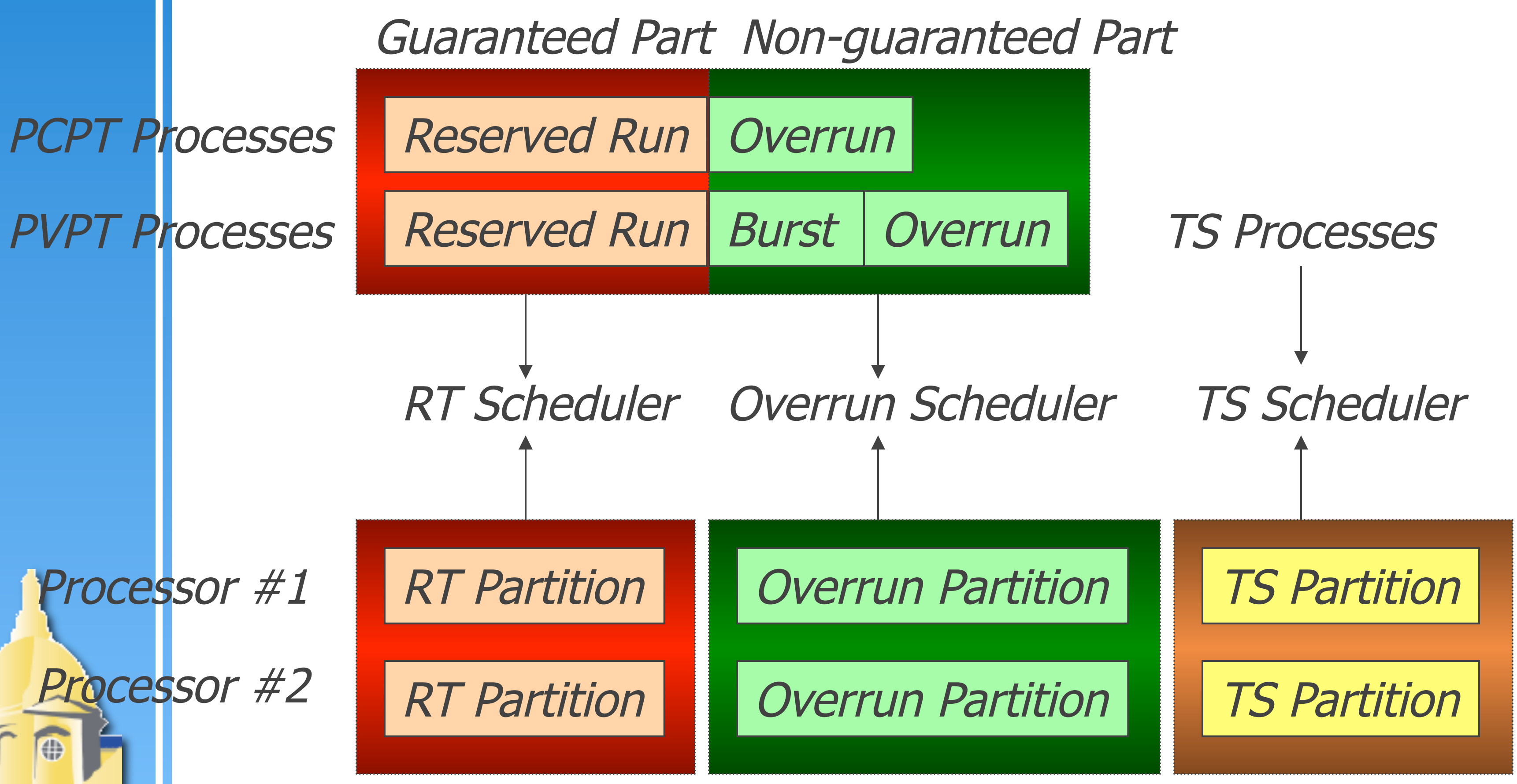
Period = 100ms
Peak Processing Time = 30ms
Sustainable Processing Time = 15ms
Burst Tolerance = 7ms



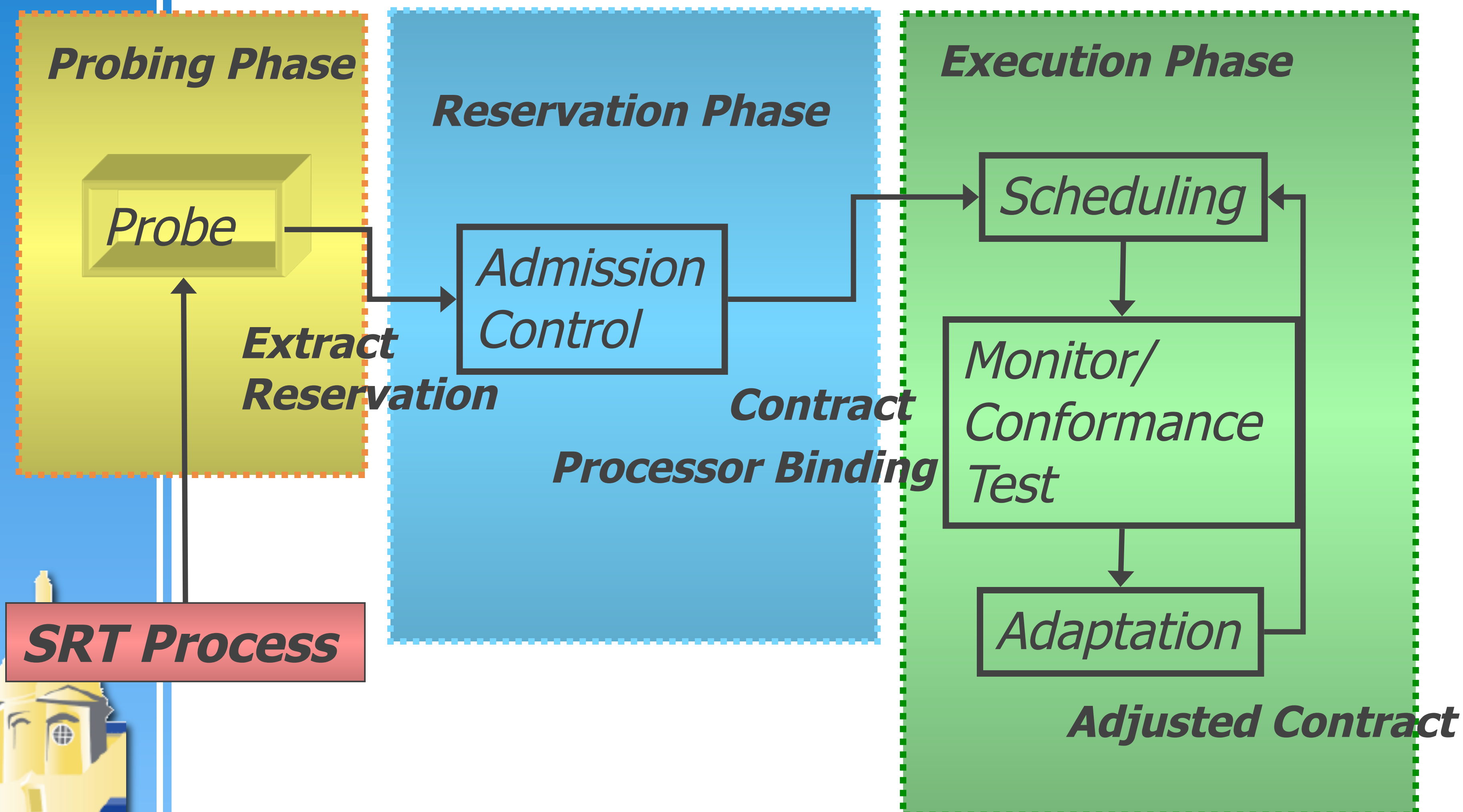


24

Multiprocessor Partitioning Design



Execution Flow of a SRT Process

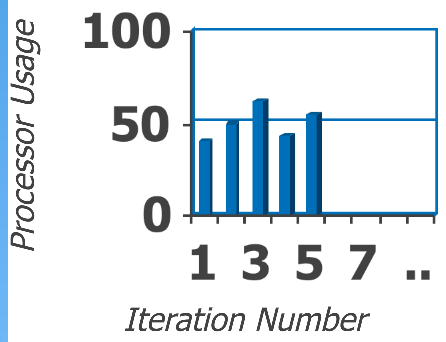


Smart Offline Probing (1)

- Goal: Extract a reservation.
  - Determine the most suitable Service Class and Parameters.
  - Avoid over/under reserve resources.
- Needed because:
  - Processor usage is hardware platform dependent.
  - Processor usage is input dependent.

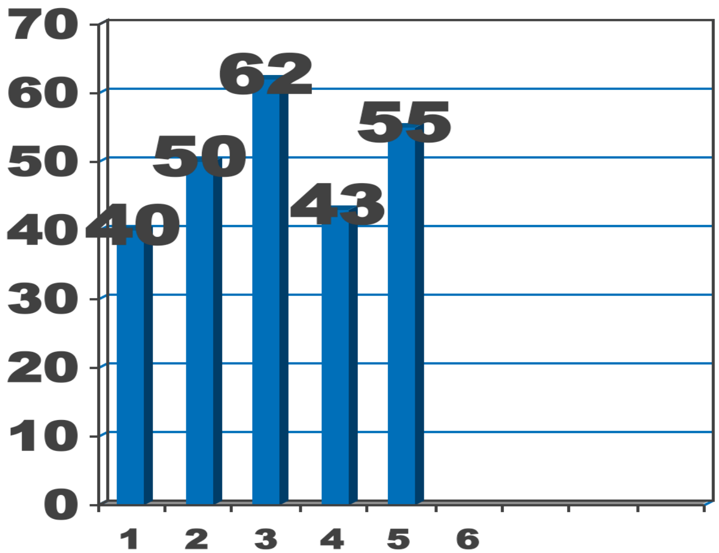
Smart Probing (2)

- DSRT runs a few iterations of SRT applications without reservation.
- DSRT monitors the usage iteration by iteration.
- DSRT analyzes the usage history.

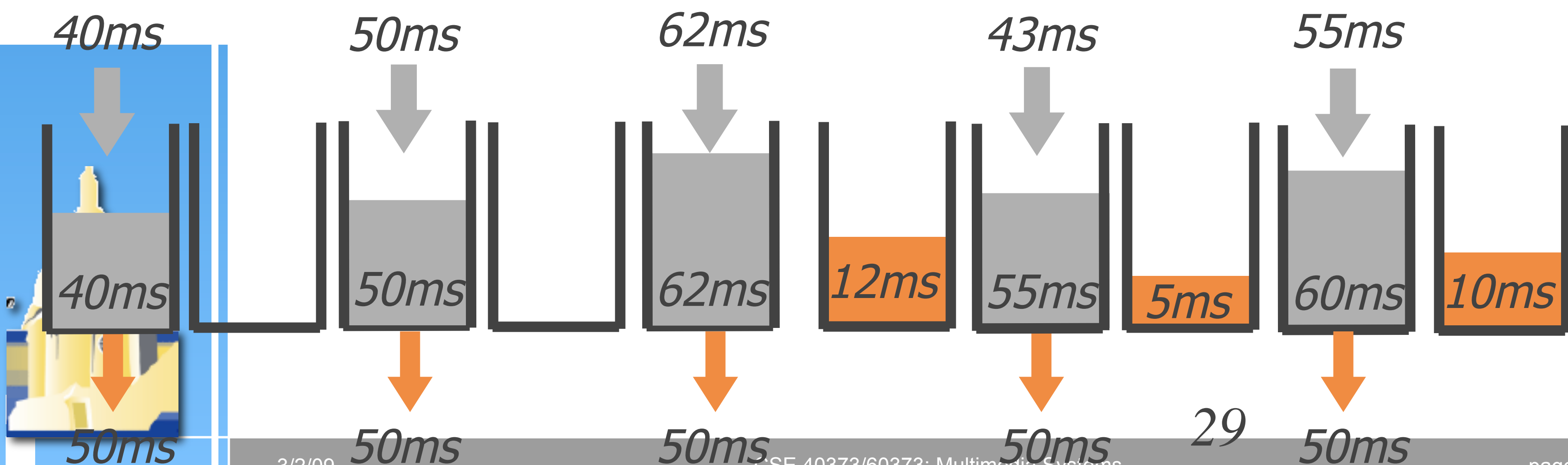


Estimate a Reservation

Smart Probing (3)

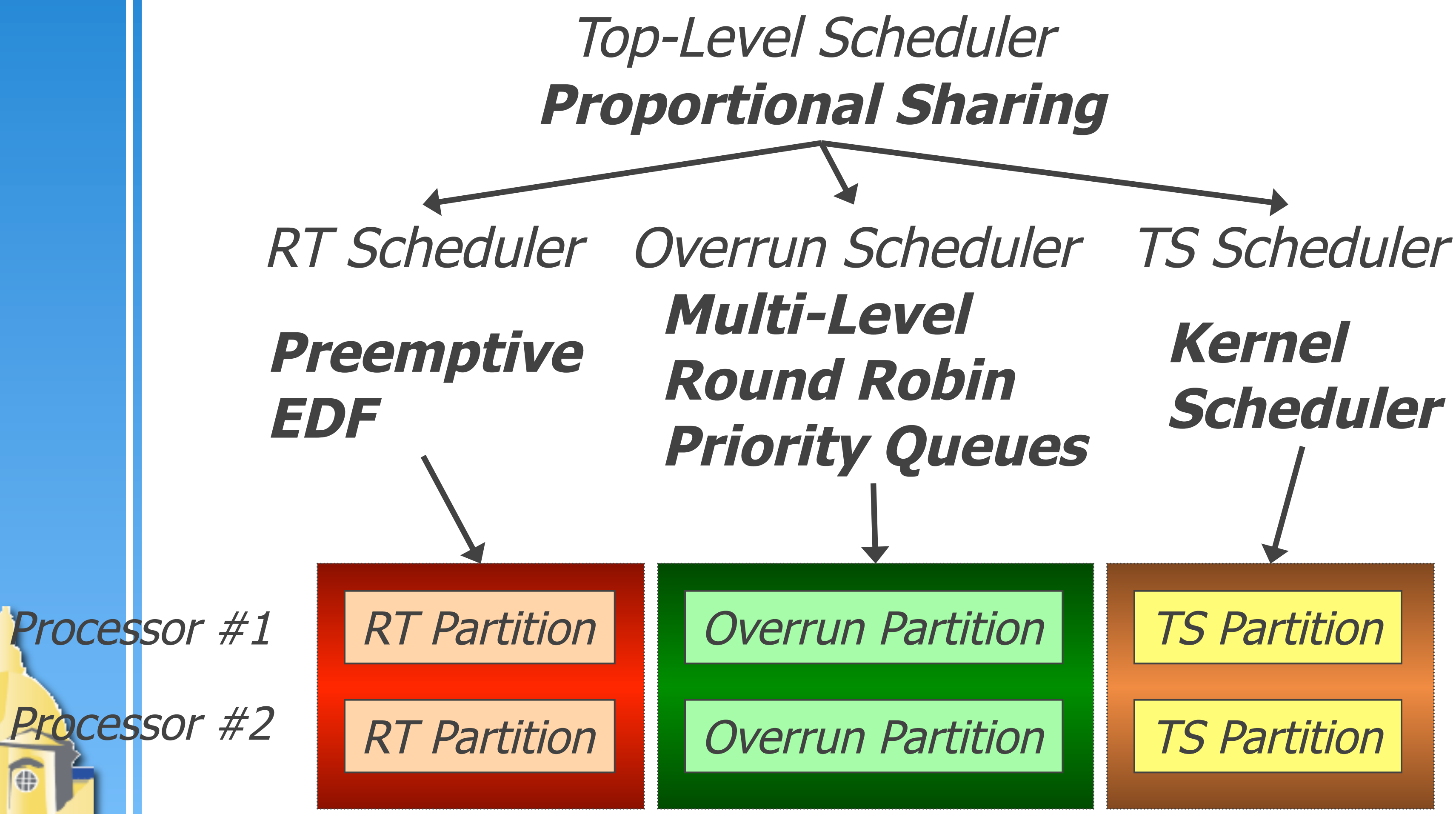


- Compute average processor usage = 50ms
- Compute peak processor usage = 62ms.
- Max Burst = 12ms

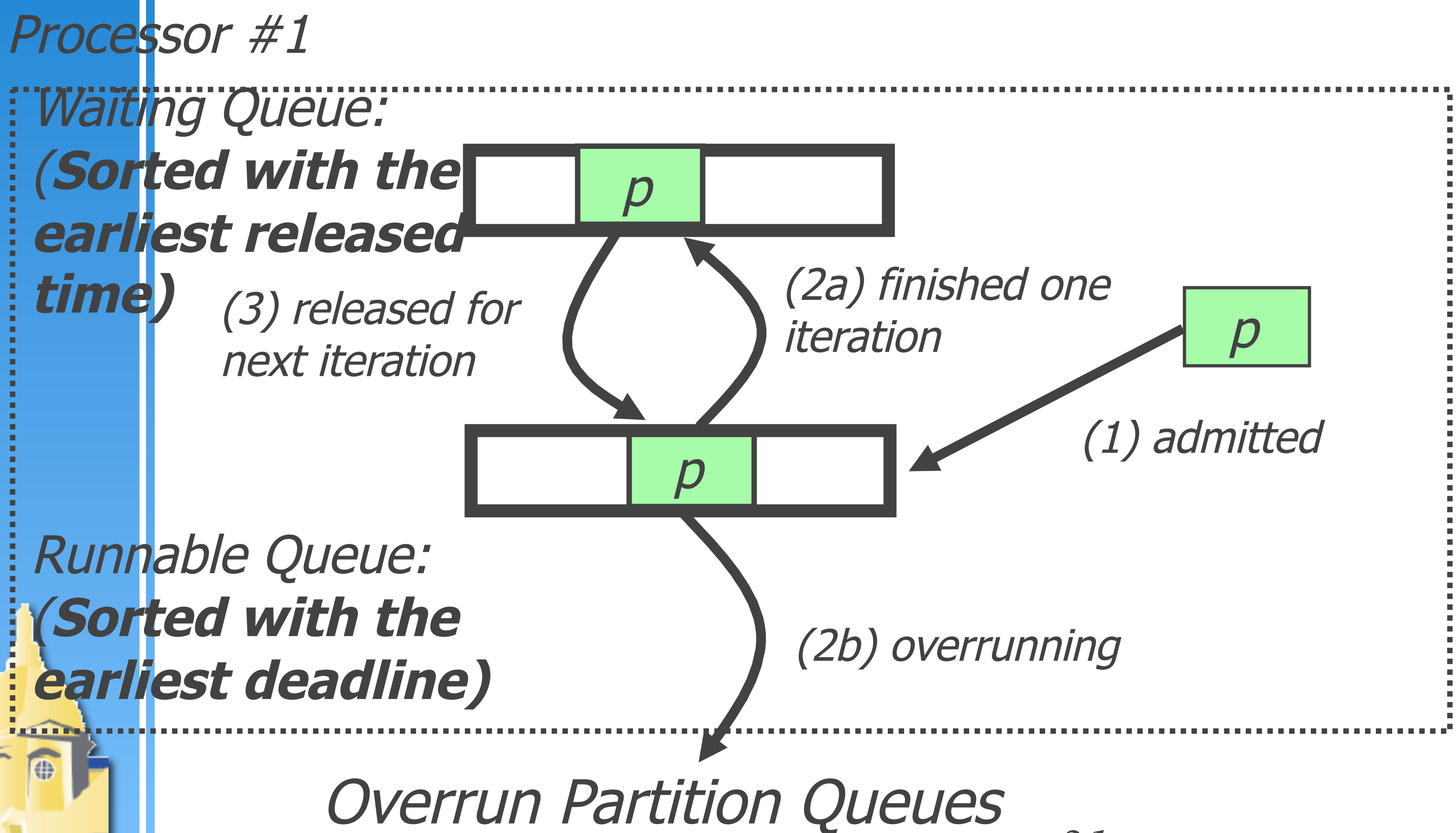


29

Partition Scheduling

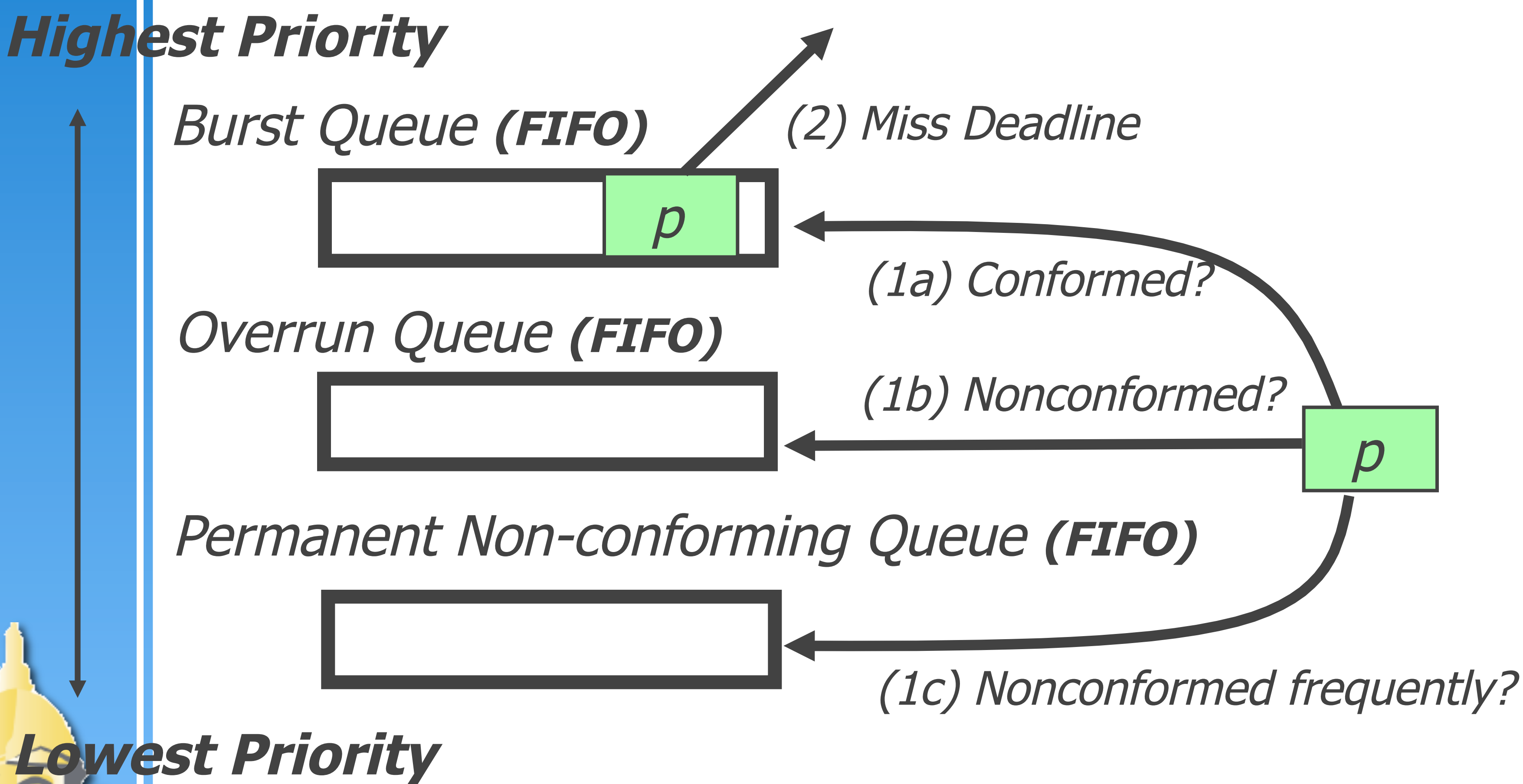


RT Partition Scheduler

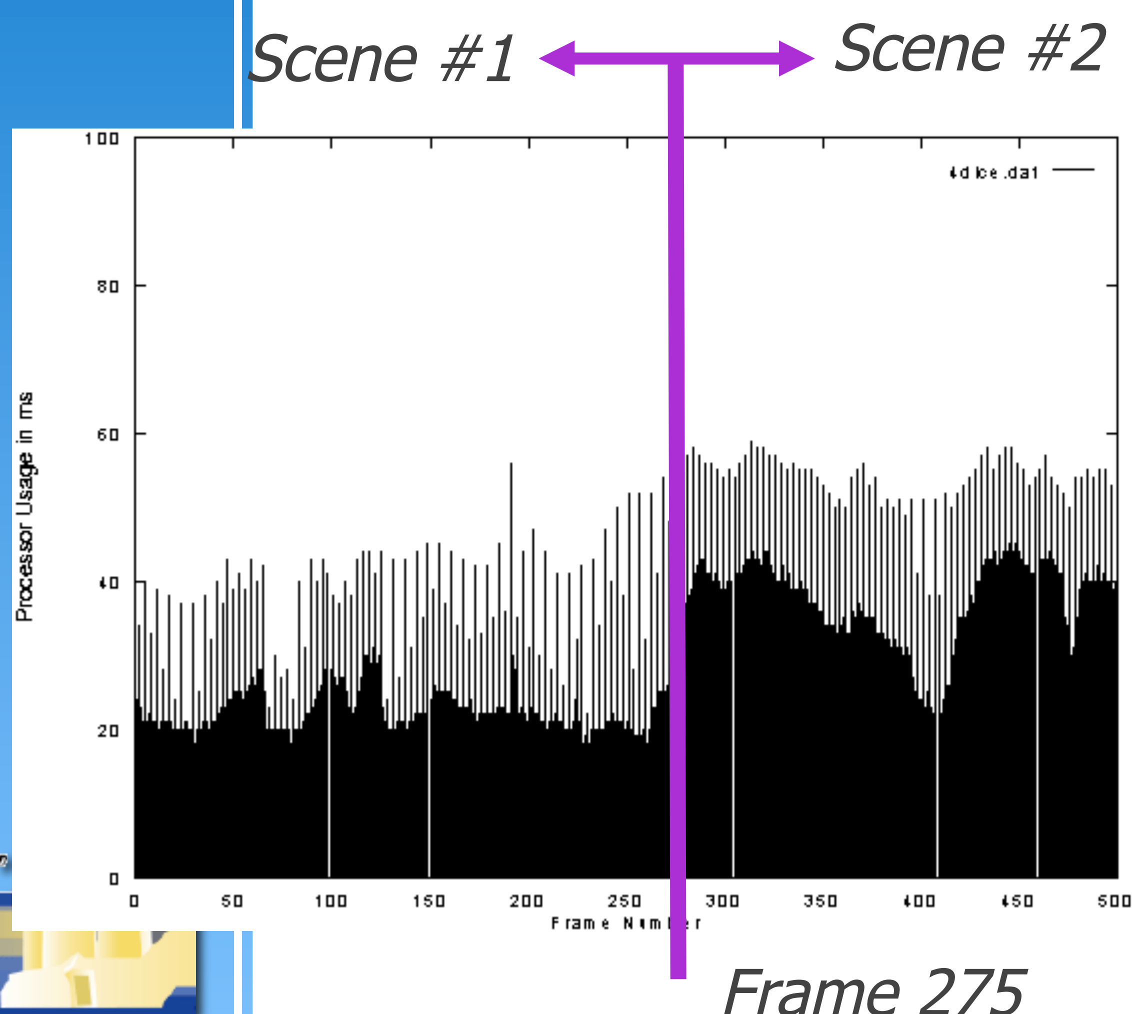


31

Overrun Partition Scheduler

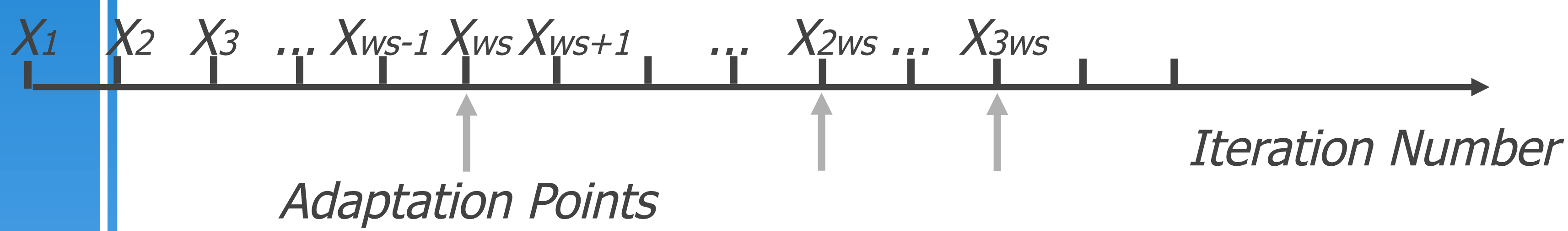


Adaptation



- Adjust Reservation.
- 3 Strategies
  - Exponential Average
  - Range
  - Statistical

Exponential Average Adaptation Strategy

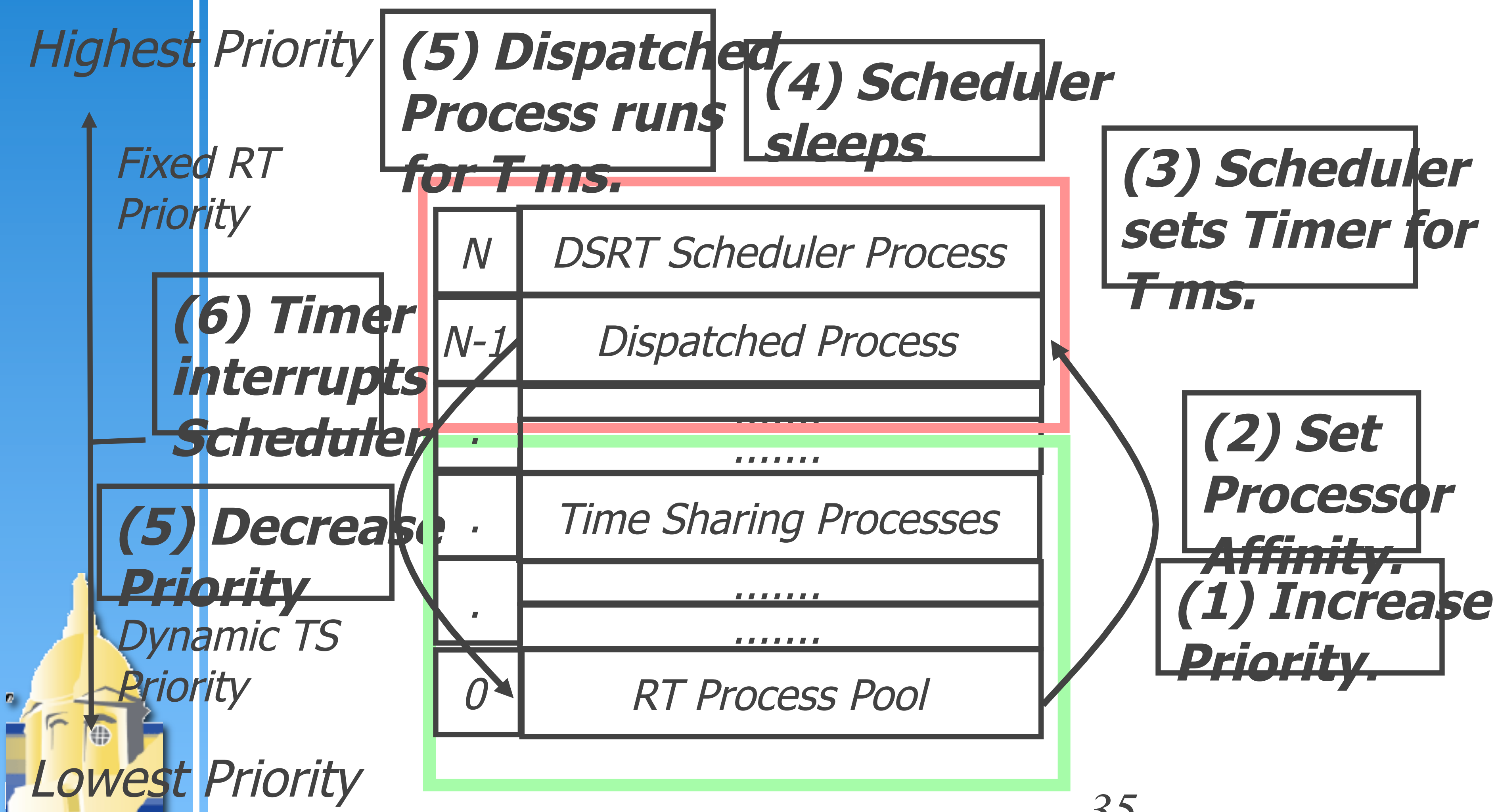


- Specification:
  - Window Size (ws).
  - Alpha (α)
  - X_i= Guaranteed Parameter in a reservation.
  - X_{i-1} = Actual Usage.

$$X_i = (1-\alpha)X_{i-1} + \alpha \overline{X}_i$$

34

User-level Priority Dispatch

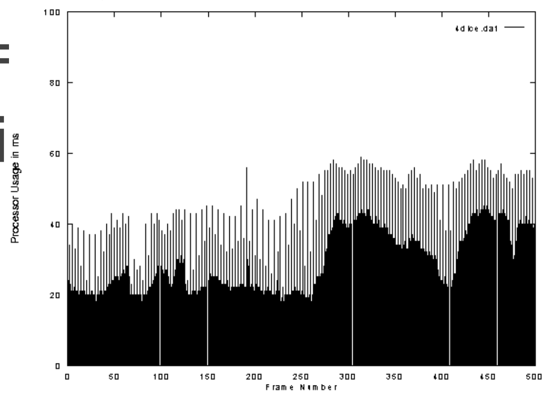


35

Experiment Setup

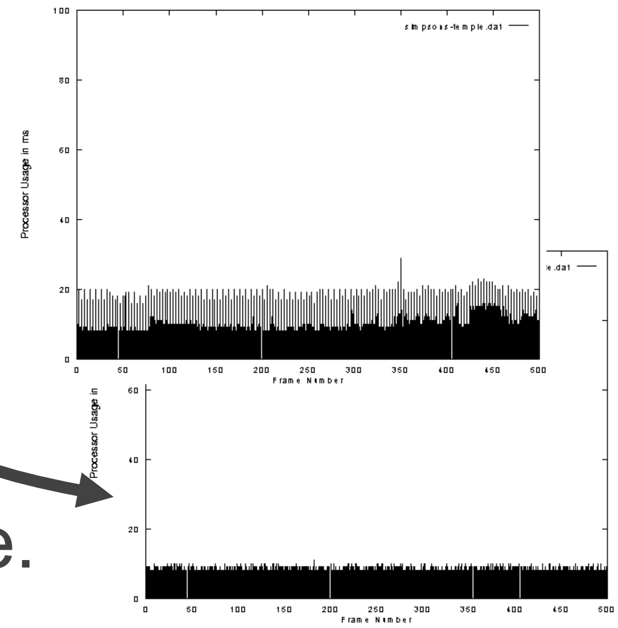
- Run 8 TS processes and 5 SRT processes concurrently.
- $TS_{1\text{-}6}$: Computational intensive programs.
- $TS_{7\text{-}8}$: Compilation programs.
- SRT_1: A MPEG player at 10 FPS.
  - Probing (PVPT class, P=100ms,
  - SPT=28ms, PPT=40ms, BT=
  - Adaptation Strategy: (Statisti

  f = 20%, ws = 20).



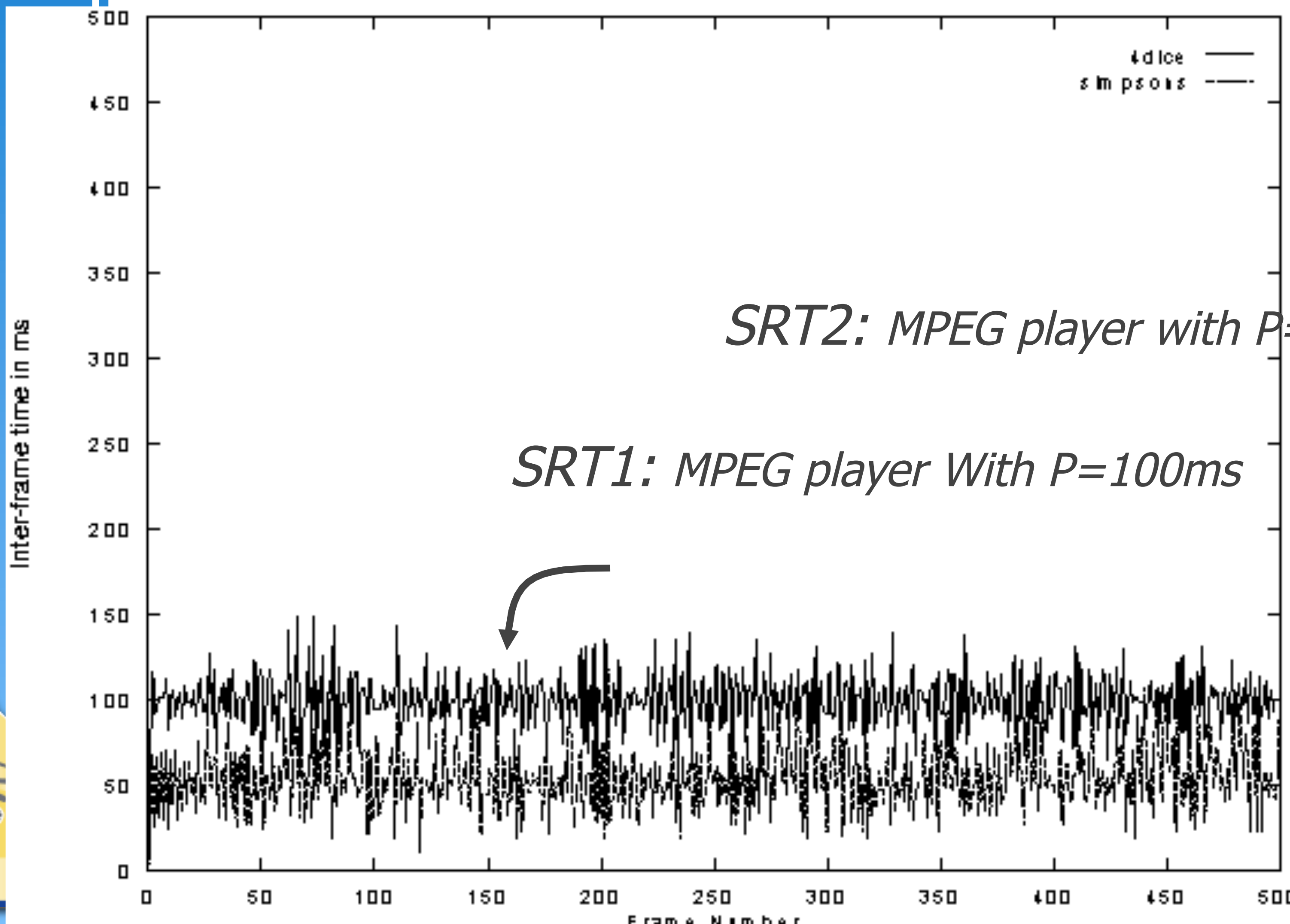
Experimental Setup (Cont.)

- SRT$_2$: A MPEG player at 20 FPS.
  - Probing (PVPT class, P=50ms, SPT=14ms, PPT=21ms, BT=6ms)
- SRT$_3$: A sampling program.
  - Probing(PCPT class, P=50ms, PPT=10ms)
- SRT$_4$: A Java RocksInSpace game.
  - (PCPT class, P=100ms, PPT=30ms).
- SRT$_5$: Misbehaving greedy program.
  - (PCPT class, P=500ms, PPT=10ms).

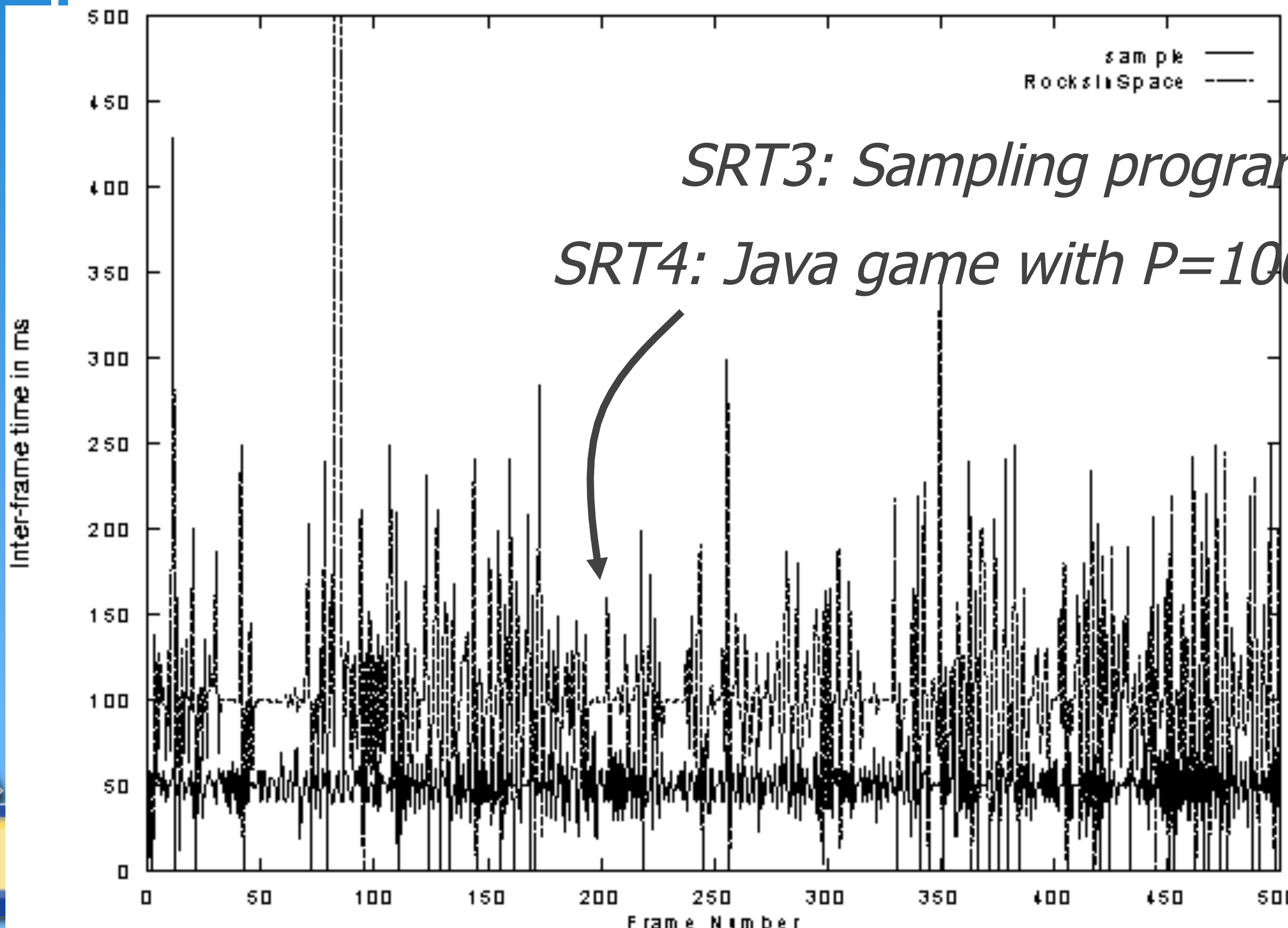


37

Experimental Result



Experimental Result (Cont.)



SRT3: Sampling program with P=50m

SRT4: Java game with P=100ms.

Conclusion (Over-provisioning Approach)

- Current Approach:
  - To achieve throughput - overprovision of resources (fast processors, large disks, fast I/O bus, high-speed networks), and exclusive usage of resources
  - To achieve timing - overprovision CPU, exclusive usage of resources, application-dependent scheduling

Current Thesis: With over-provisioning we get Quality of Service !!

Conclusion (Problems with Over-provisioning)

- Violation of Timing Guarantees in Exclusive Case (head-of-line blocking)
  - Sensory and Video Data Processing/Transmission
- Violation of Timing/Throughput Guarantees in Shared Case (greedy applications, flows)
  - UDP flows versus TCP Flows
- Last Mile Problem (not everywhere is over-provisioning possible)
  - Telescopes, University Campus
- Need support of QoS in OS towards multimedia tasks