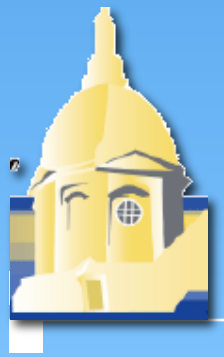
So far…

- RMS - task with highest rate has highest priority
- EDF – earliest deadline first

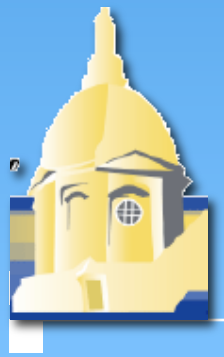
- Tasks can be preemptive or non-preemptive



Slides courtesy Prof. Nahrstedt

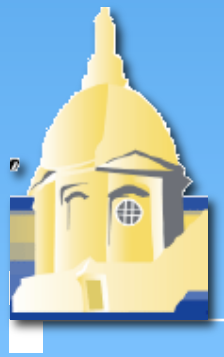
Example

- 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?



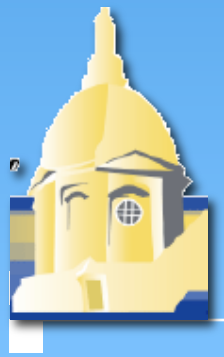
Conclusion

- 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



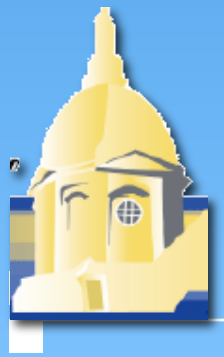
Other In-Time Scheduling Policies

- 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



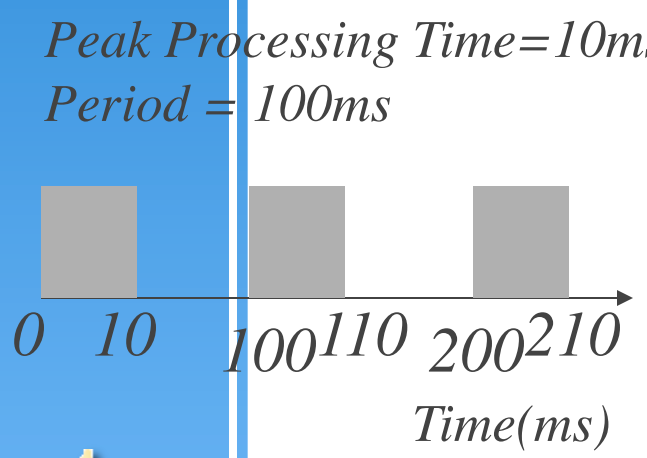
Source: Hao-hua Chu 1999

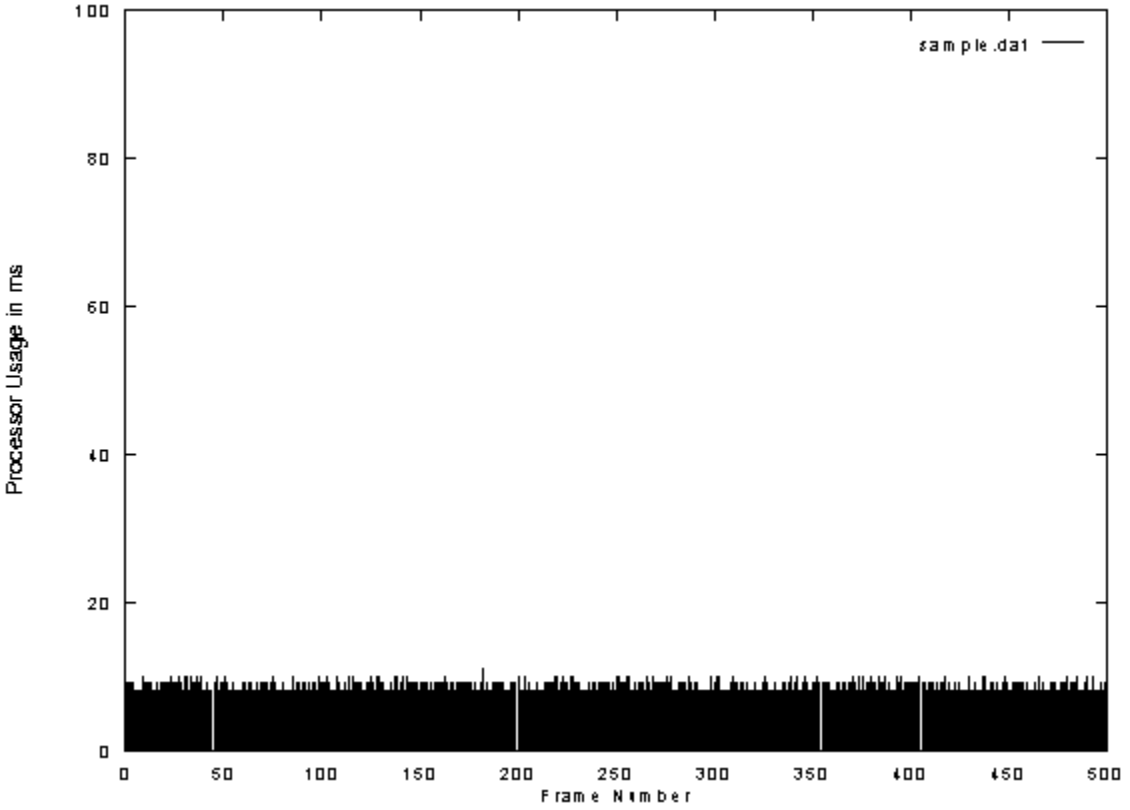
CPU Service Classes

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:

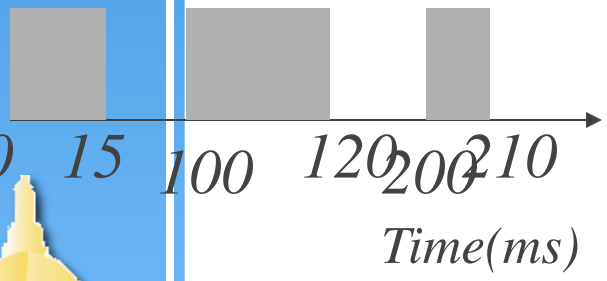


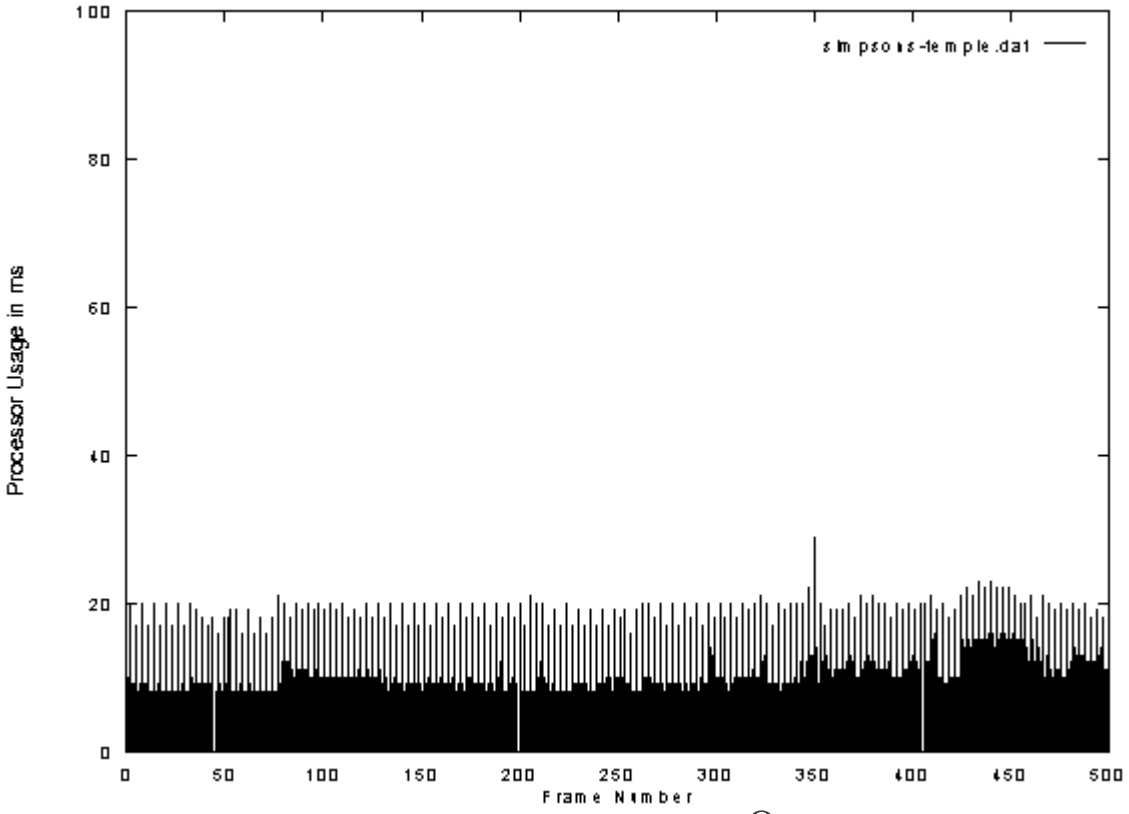


Periodic Variable Processing Time Class

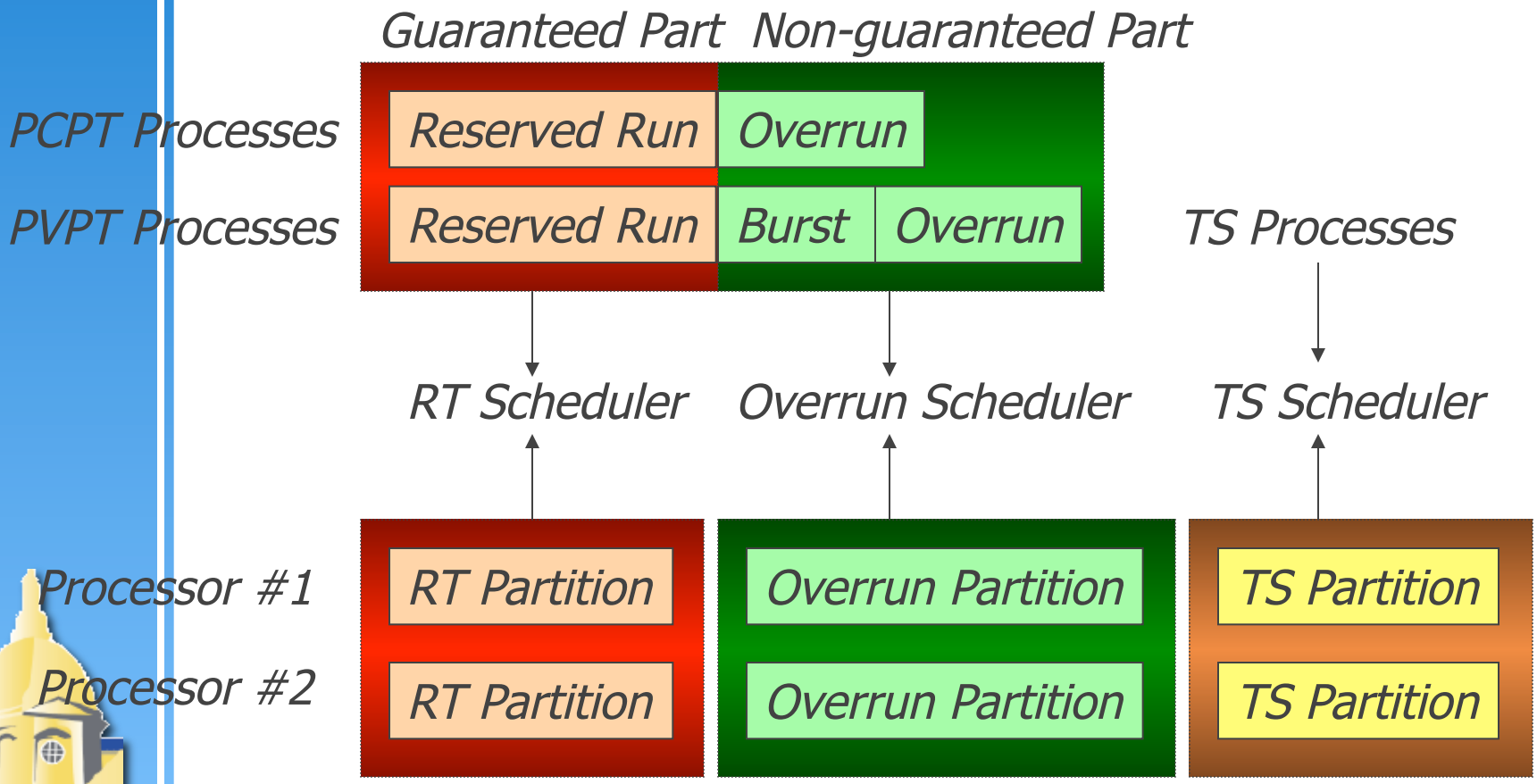
- Example Usage Pattern:

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

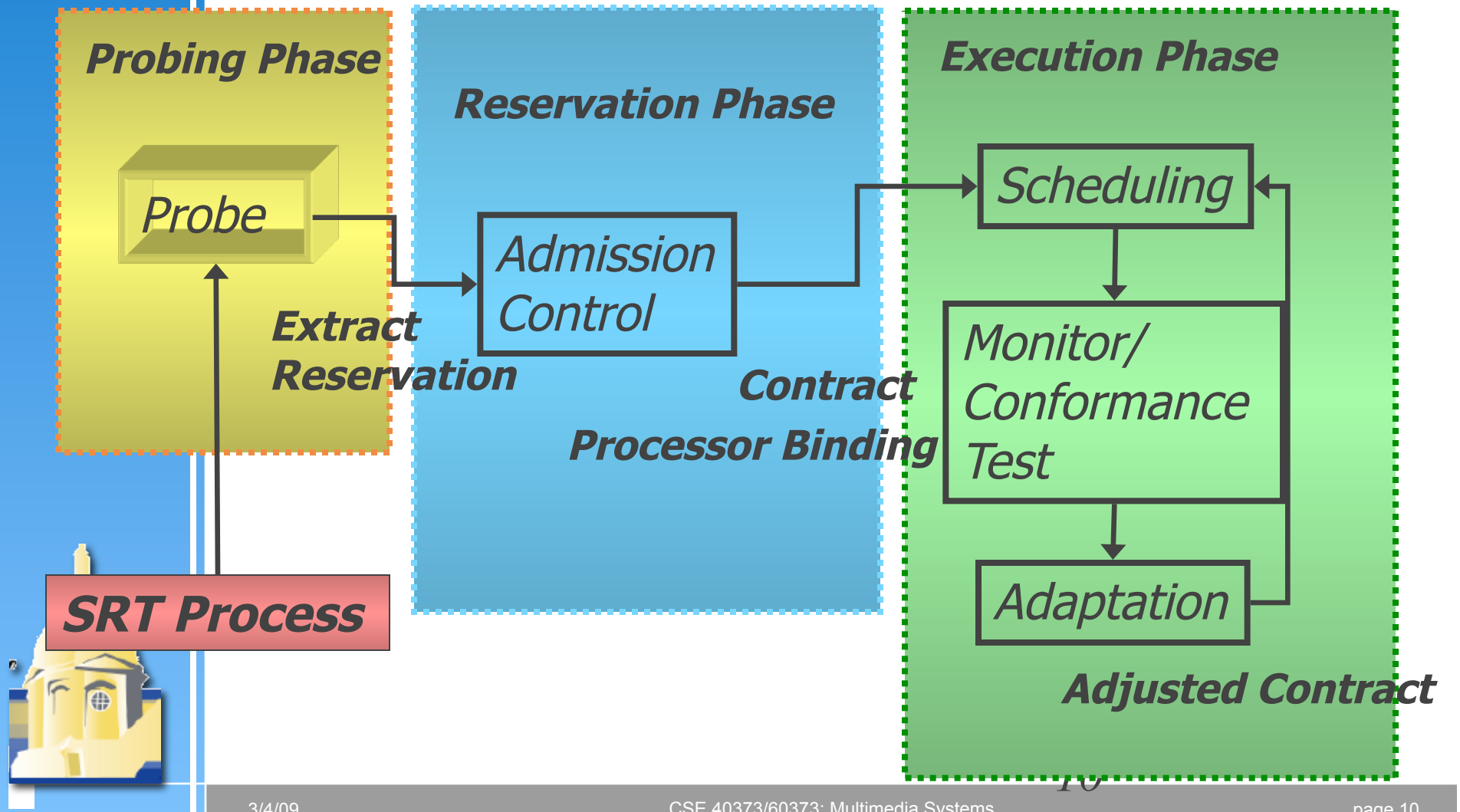




Multiprocessor Partitioning Design

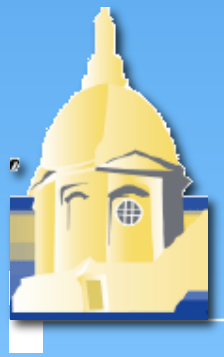


Execution Flow of a SRT Process



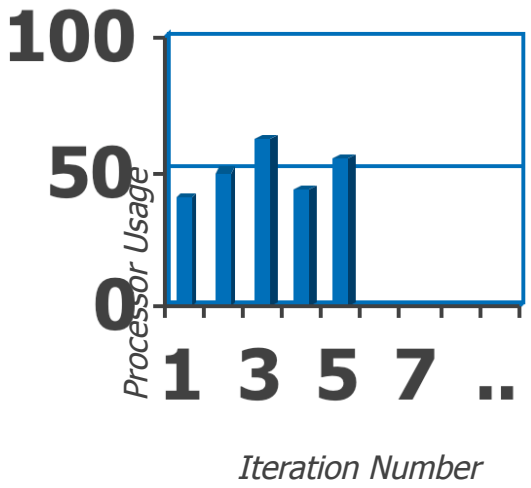
Smart Offline Probing (1)

- 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.



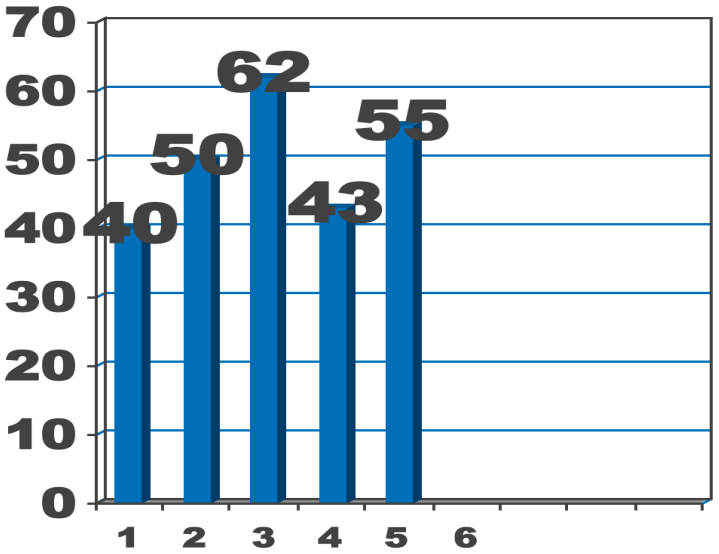
Smart Probing (2)

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

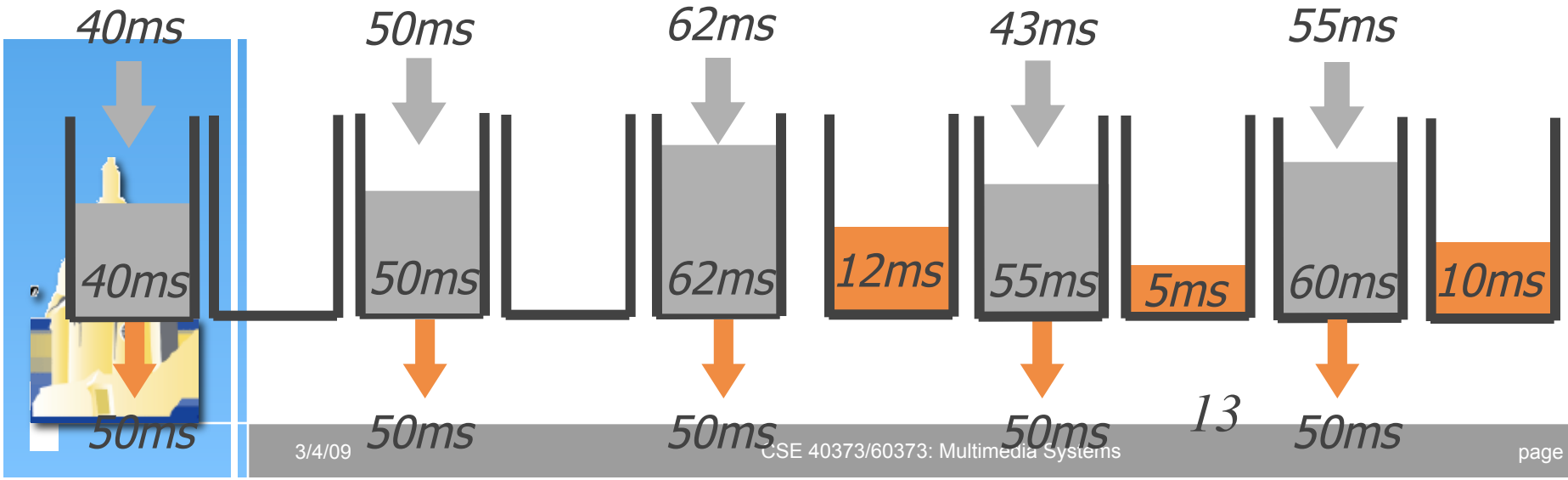


Estimate a Reservation

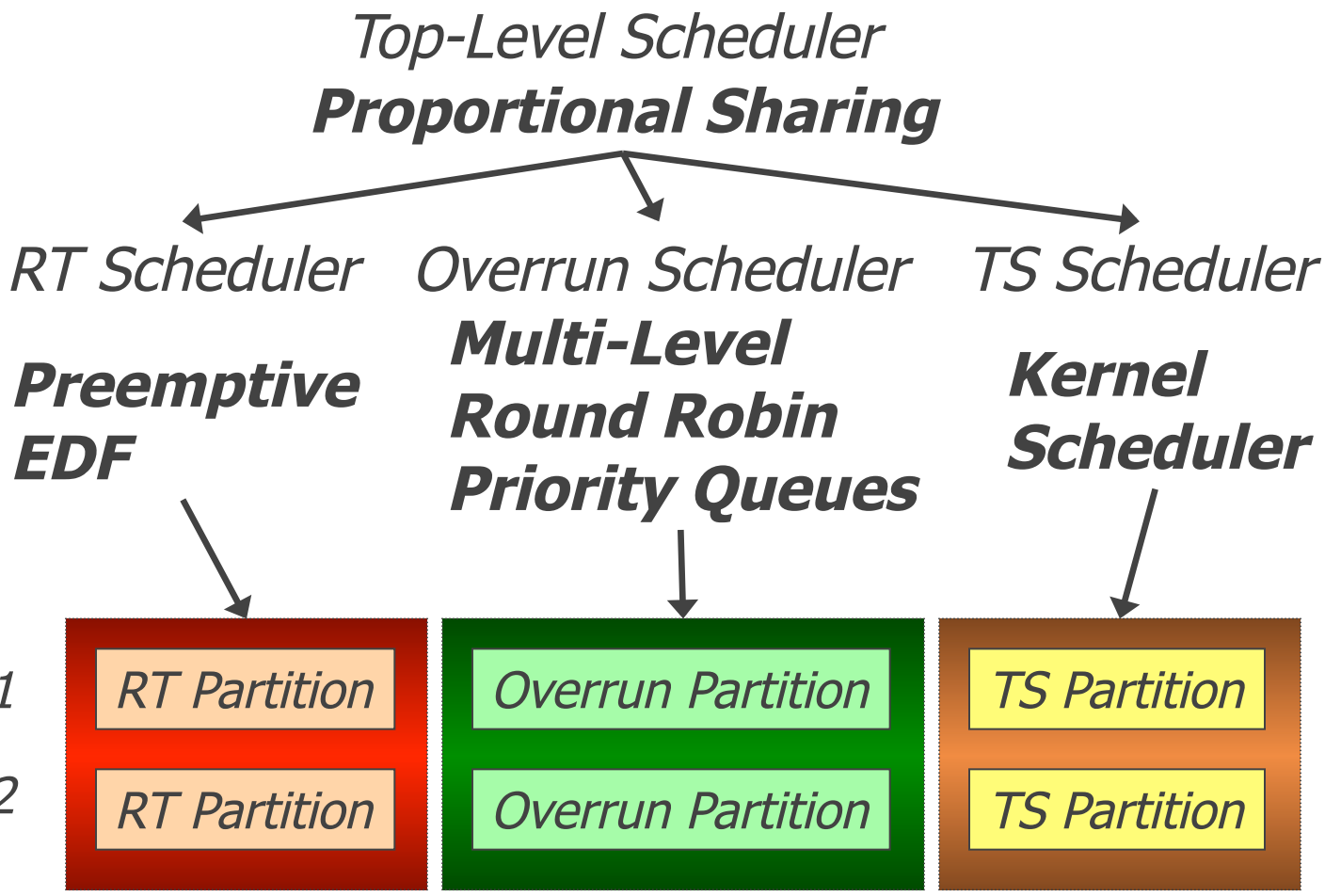
Smart Probing (3)



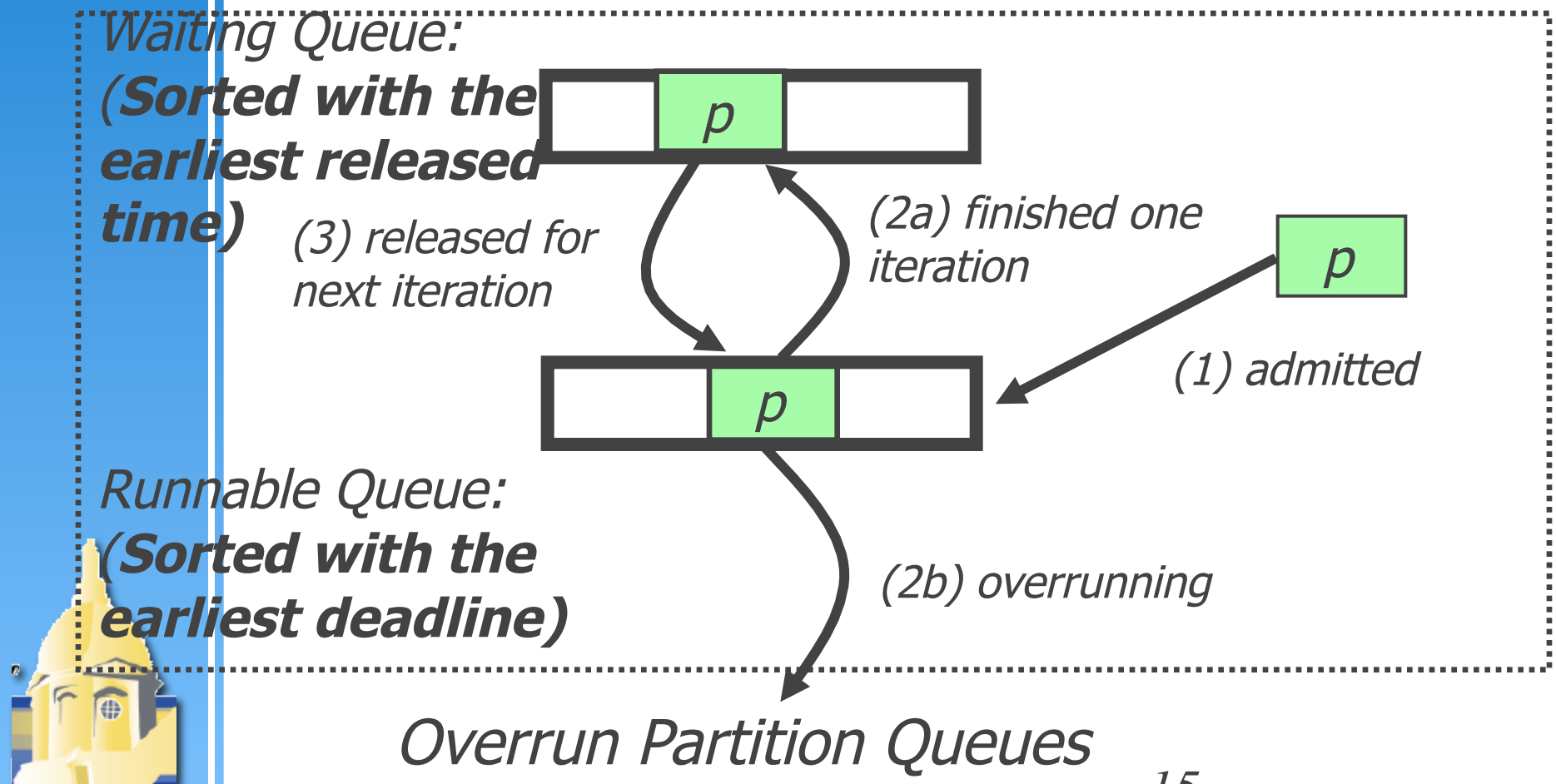
- Compute average processor usage = 50ms
- Compute peak processor usage = 62ms.
- Max Burst = 12ms



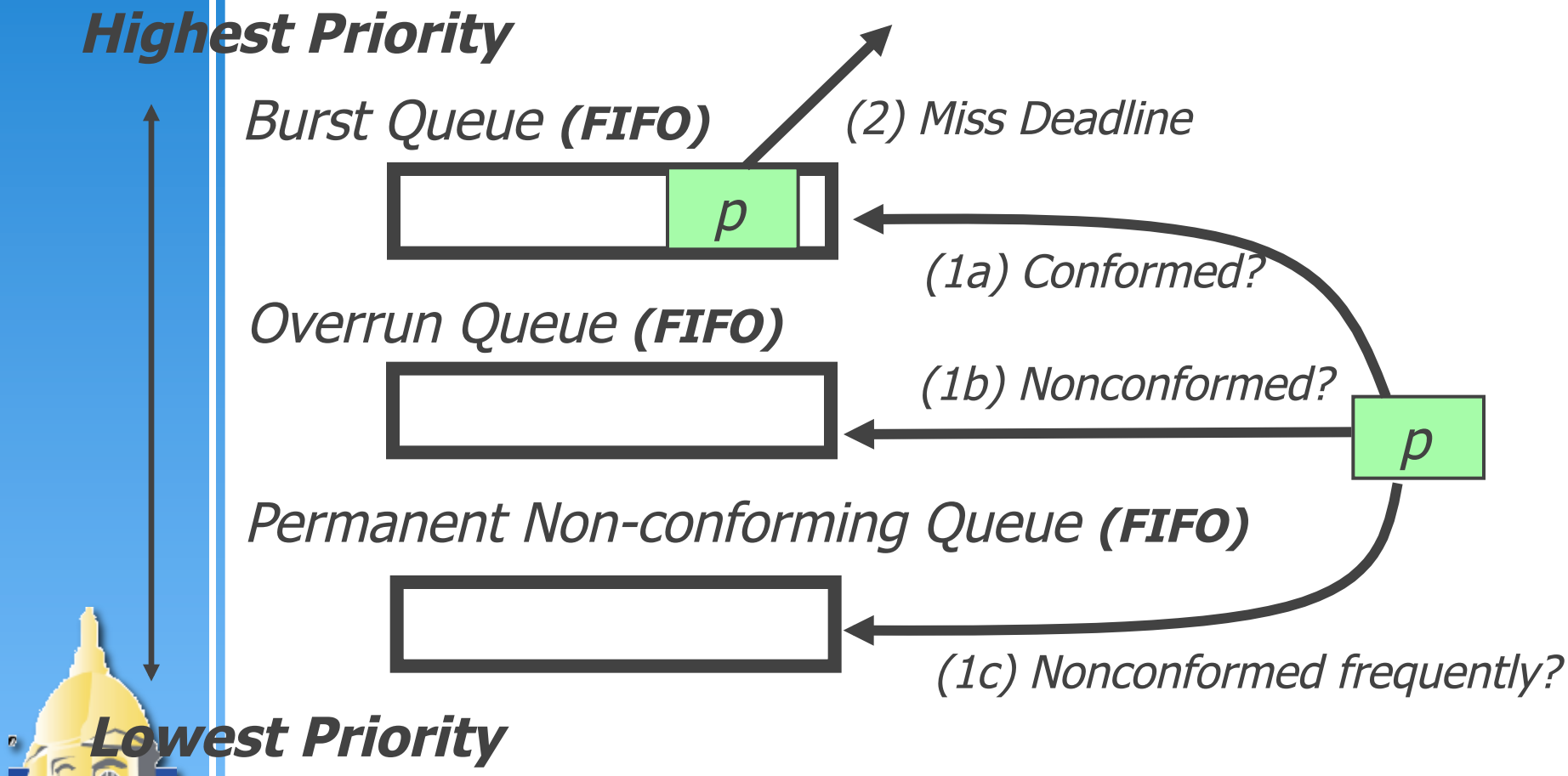
Partition Scheduling



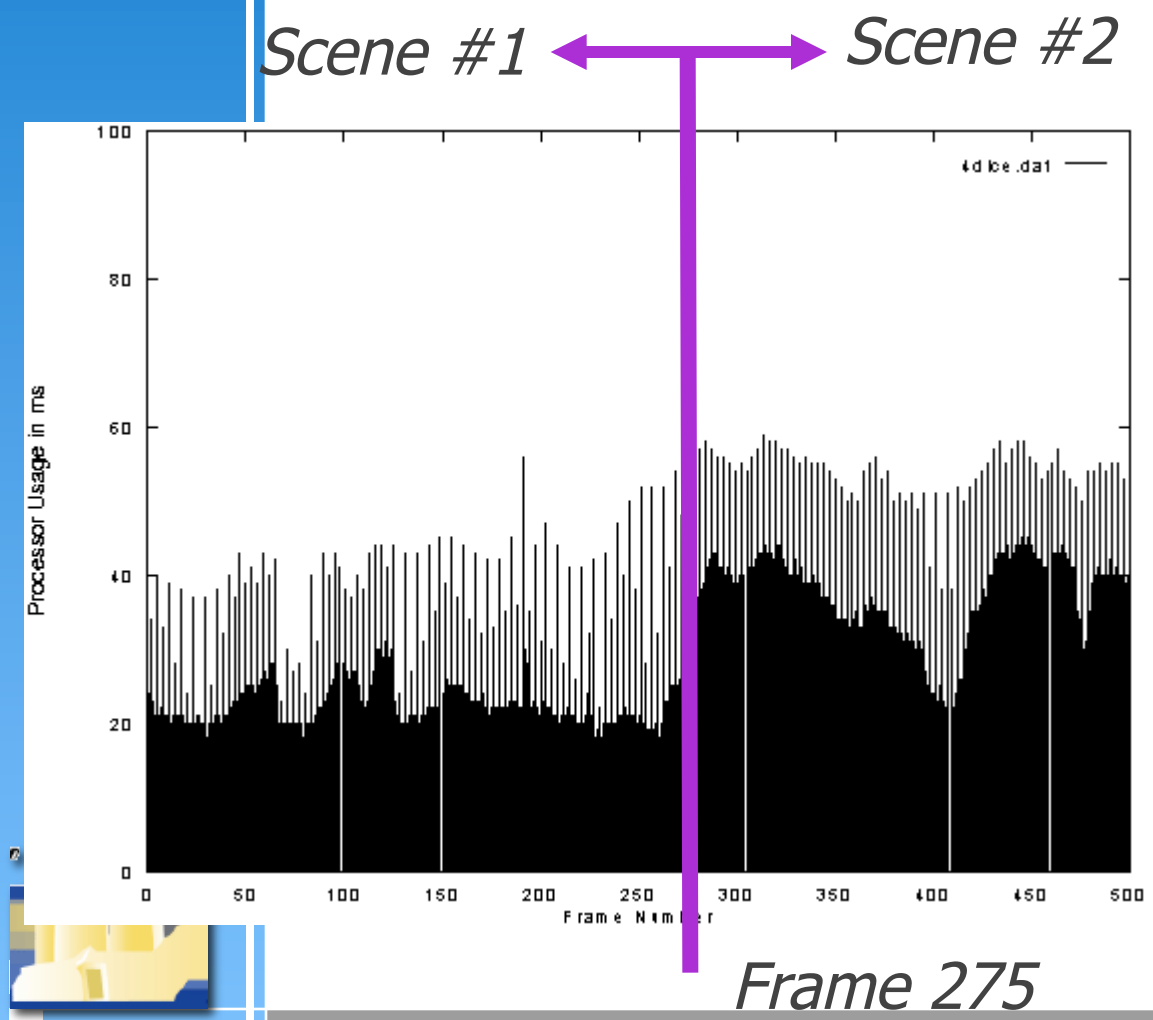
RT Partition Scheduler



Overrun Partition Scheduler

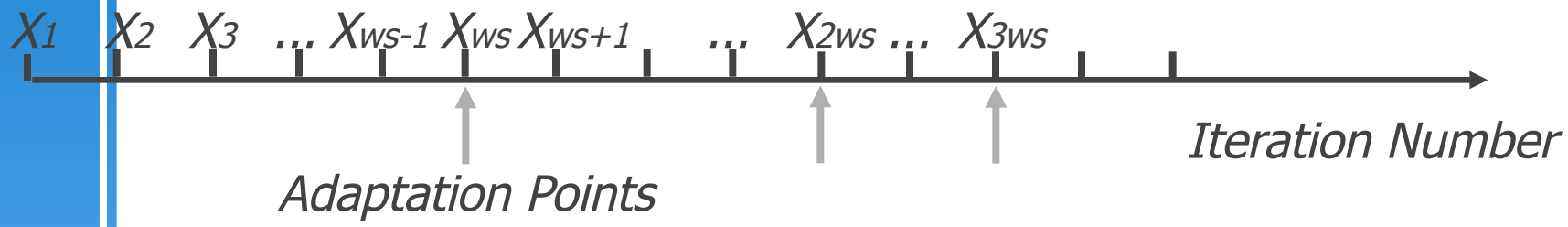


Adaptation



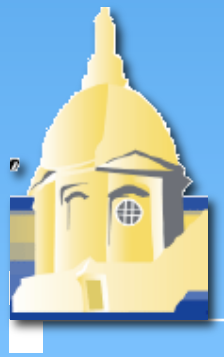
- Adjust Reservation.
- 3 Strategies
  - Exponential Average
  - Range
  - Statistical

Exponential Average Adaptation Strategy



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

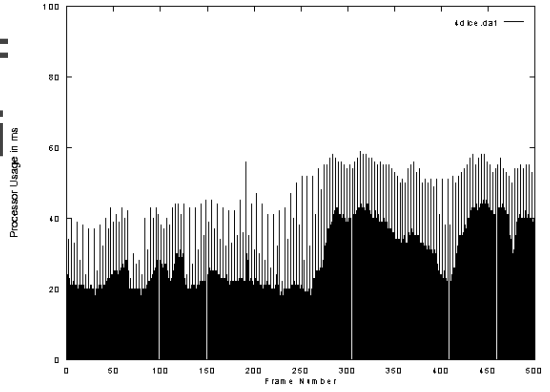
$$X_i = (1-\alpha)X_{i-1} + \alpha\,\overline{X}_i$$



Experiment Setup

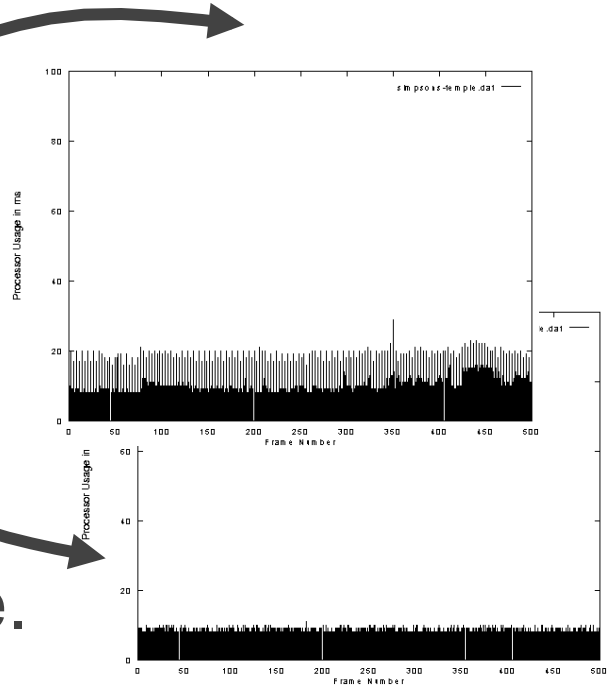
- Run 8 TS processes and 5 SRT processes concurrently.
- $TS_{1\text{-}6}$: Computational intensive programs.
- $TS_{7\text{-}8}$: Compilation programs.
- SRT_1: A MPEG player at 10 FPS.
  - Probing (PVPT class, P=100ms,
  - SPT=28ms, PPT=40ms, BT=
  - Adaptation Strategy: (Statisti

  f = 20%, ws = 20).

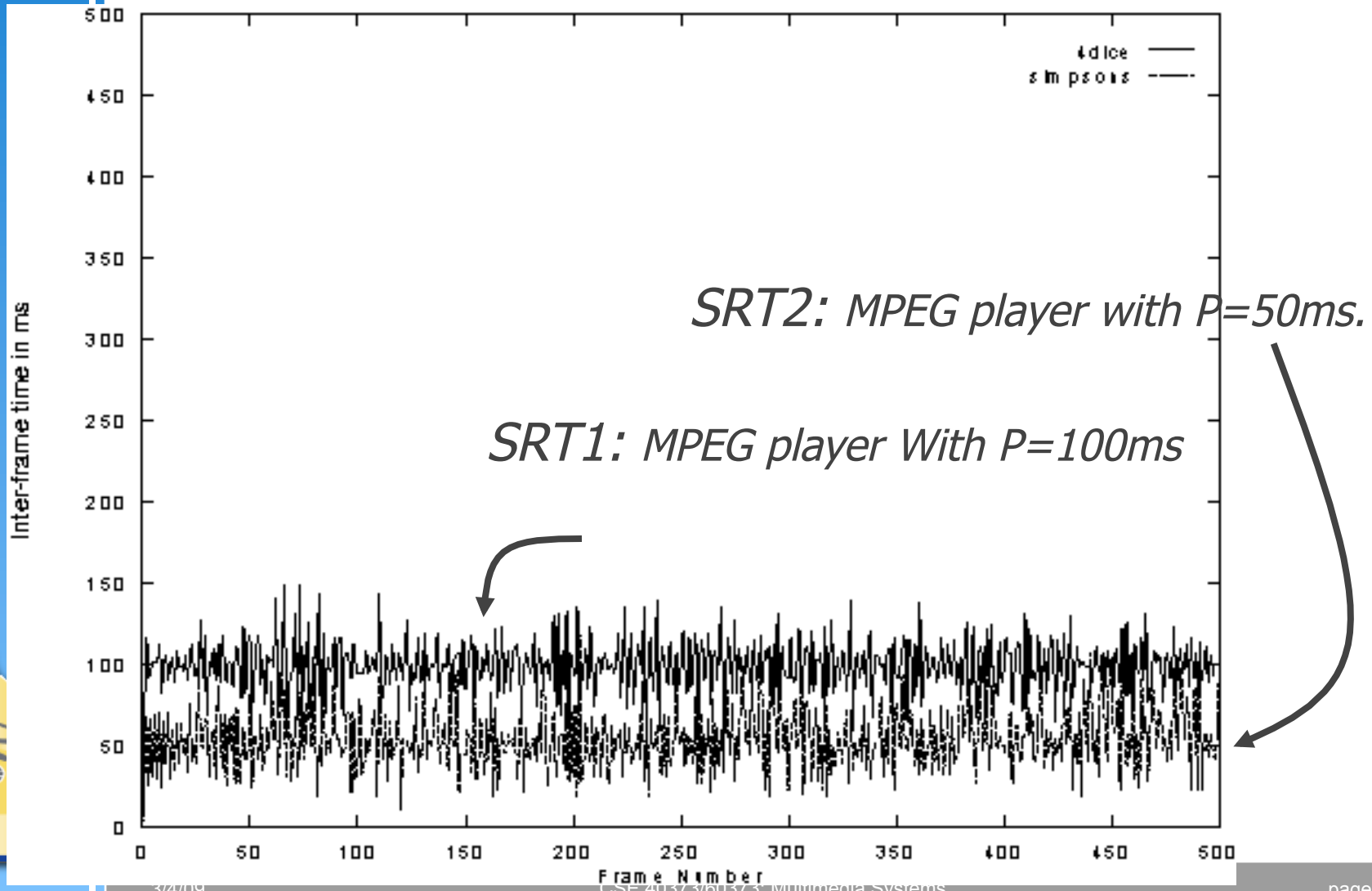


Experimental Setup (Cont.)

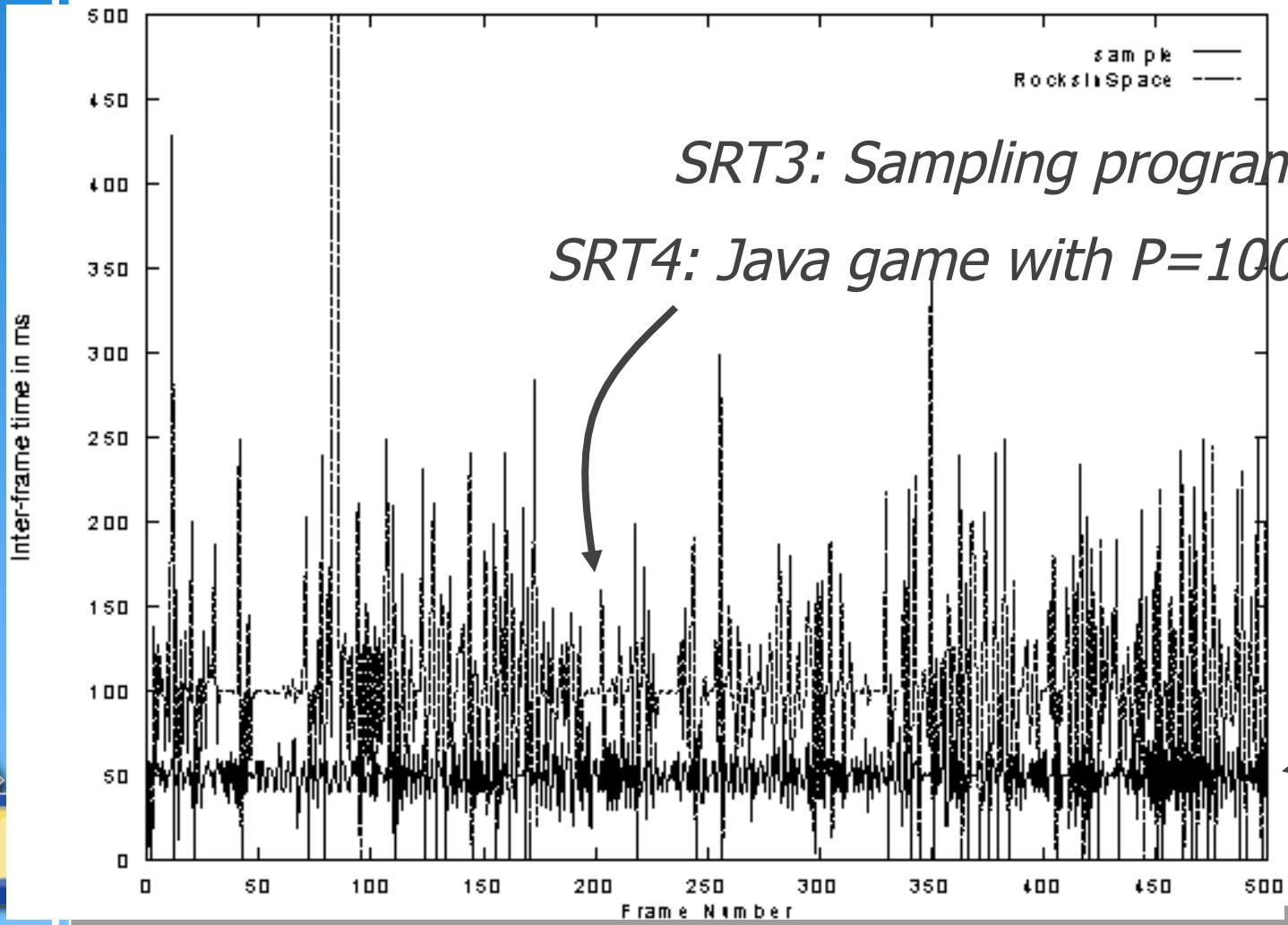
- SRT_2: A MPEG player at 20 FPS.
  - Probing (PVPT class, P=50ms, SPT=14ms, PPT=21ms, BT=6ms)
- SRT_3: A sampling program.
  - Probing(PCPT class, P=50ms, PPT=10ms)
- SRT_4: A Java RocksInSpace game.
  - (PCPT class, P=100ms, PPT=30ms).
- SRT_5: Misbehaving greedy program.
  - (PCPT class, P=500ms, PPT=10ms).



Experimental Result



Experimental Result (Cont.)



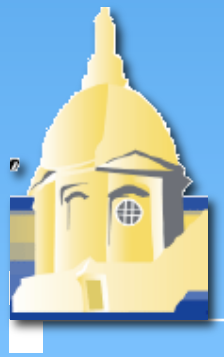
SRT3: Sampling program with P=50m

SRT4: Java game with P=100ms.

Conclusion (Over-provisioning Approach)

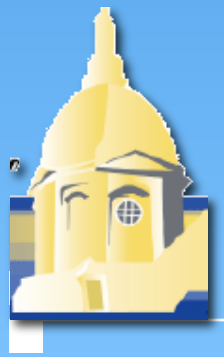
- Current Approach:
  - To achieve throughput - overprovision of resources (fast processors, large disks, fast I/O bus, high-speed networks), and exclusive usage of resources
  - To achieve timing - overprovision CPU, exclusive usage of resources, application-dependent scheduling

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



So far…

- Operating system scheduling needs to be aware of the temporal constraints for processing real time multimedia
  - Reservation based for performance guarantees and adaptive for flexibility and higher resource utilization