EE445M/EE360L.6 Embedded and Real-Time Systems/ Real-Time Operating Systems

Lecture 5:

Real-Time Scheduling, Priority Scheduler

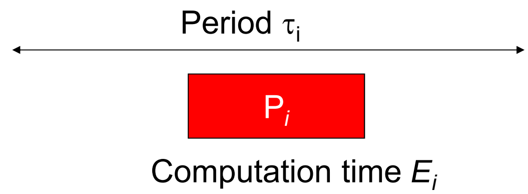
Real-Time Scheduling

- Tasks have deadlines
  - Some tasks are more important than others
  - In order to do something first, something else must be second
  - Priority scheduler
- Reactivity
  - When to run the scheduler?
    - Periodically, systick and sleep
    - On **`OS_Wait`**
    - On **`OS_Signal`**
    - On **`OS_Sleep, OS_Kill`**

Reference Book, Chapter 5

Real-Time Scheduling Model

- E_i is execution time of process i
- Deadline τ_i is period of process i



- Response time r_i
  - Time from arrival until finish of task
- Lateness l_i
  - $r_i - l_i$

Scheduling Metrics

- How do we evaluate a scheduling policy?
  - Ability to satisfy all deadlines
    - Minimize maximum lateness
  - CPU utilization $\sum_i E_i / \tau_i$
    - Percentage of time devoted to useful work
  - Scheduling overhead
    - Time required to make scheduling decision

Scheduling Algorithms

- Rate monotonic scheduling (RMS), static
  - Assign priority based on how frequent task is run
  - Lower *period* (more frequent) are higher priority
- Earliest deadline first (EDF), dynamic
  - Assign priority based on closest deadline
- Least slack-time first (LST), dynamic
  - Slack = (time to deadline)-(work left to do)
- …

Lecture 5 J. Valvano, A. Gerstlauer EE445M/EE380L.6 5

Scheduling Analysis

- Rate monotonic scheduling theorem
  - All n tasks are periodic
    - Priority based on period τ_i
    - Maximum execution time E_i
  - No synchronization between tasks (independent)
  - Execute highest priority task first
  - Guarantee deadlines if processor utilization:

$$\sum \frac{E_i}{\tau_i} \leq n\left(2^{1/n} - 1\right) \leq \ln(2) \approx 69\%$$

Lecture 5 J. Valvano, A. Gerstlauer EE445M/EE380L.6 6

Rate Monotonic Analysis (RMA)

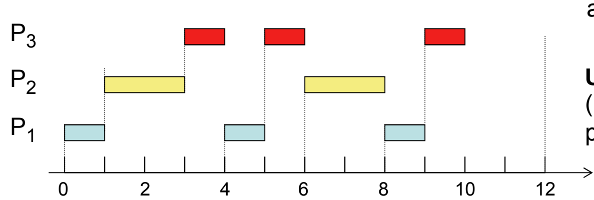
- Optimal (fixed) priority assignment
  - Shortest-period process gets highest priority
    - priority based preemption can be used…
  - Priority inversely proportional to period
  - Break ties arbitrarily
- No fixed-priority scheme does better.
  - RMS provides the highest worst case CPU utilization while ensuring that all processes meet their deadlines

RMS Example 1

Process P_i	Execution Time E_i	Period τ_i
P_1	1	**4**
P_2	2	**6**
P_3	3	**12**

Static priority: P1 >> P2 >> P3

Critical instant
all tasks arrive at same time



Unrolled schedule
(least common multiple of process periods)

RMS Example 2

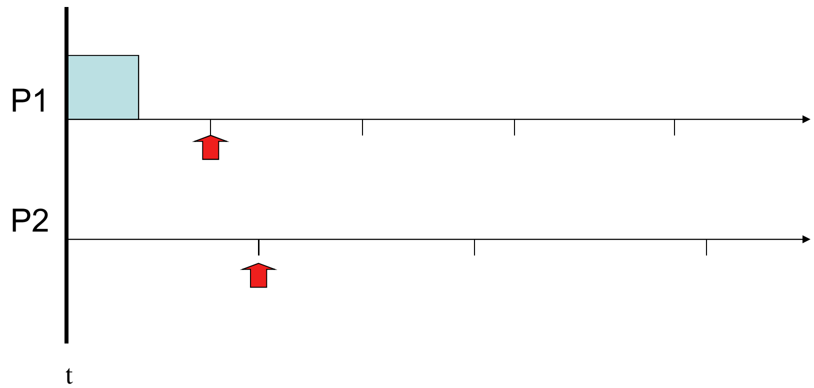
Process P_i	Execution Time E_i	Period τ_i
P_1	1	**4**
P_2	6	**8**

Is this task set schedulable?? If yes, give the CPU utilization.

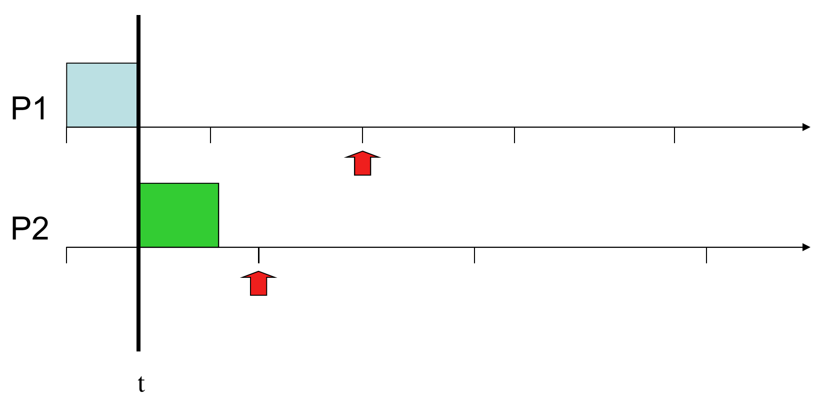
Earliest-Deadline-First (EDF)

- *Dynamic* priority scheduling scheme
  - Process closest to its deadline has highest priority
- EDF is optimal
  - EDF can use 100% of CPU for worst case
- Expensive to implement
  - On each OS event, recompute priorities and resort tasks

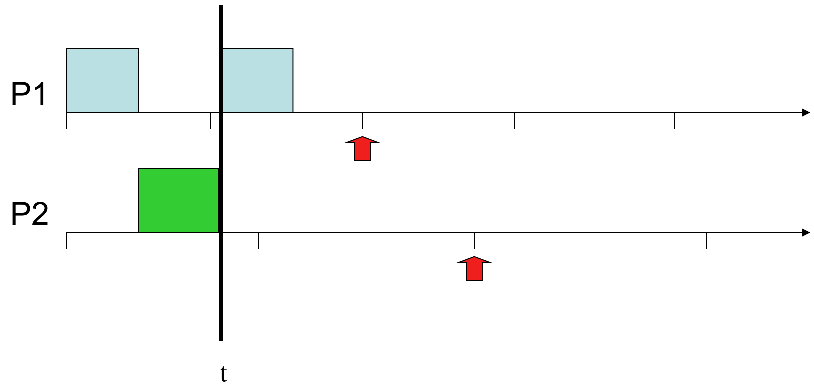
EDF Example



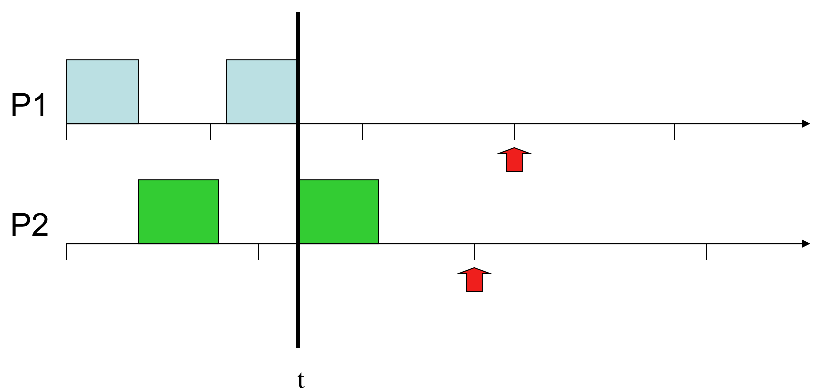
EDF Example



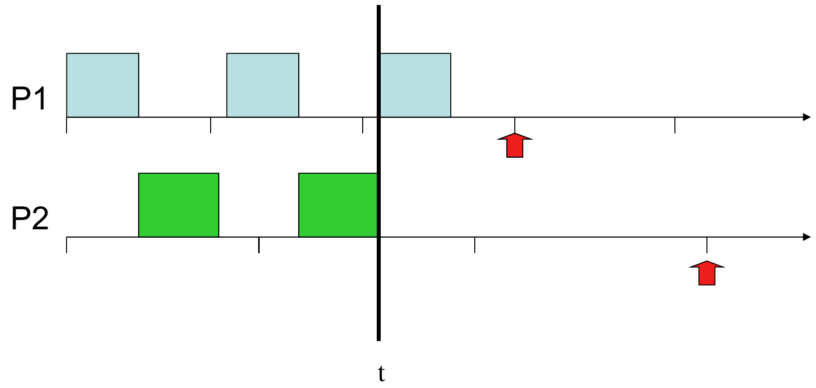
EDF Example



EDF Example

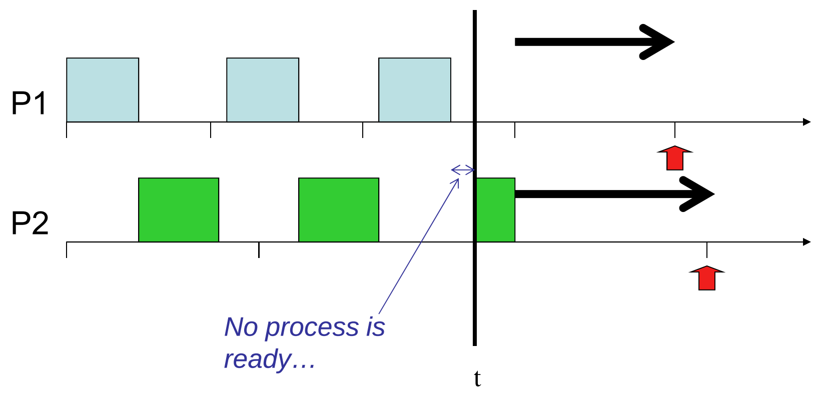


EDF Example



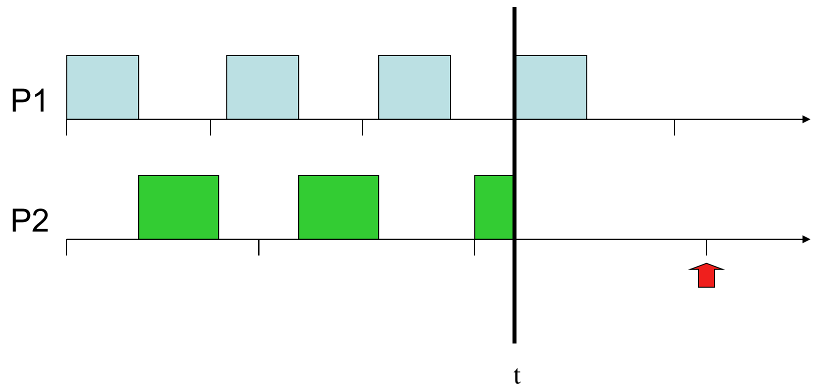
Source: M. Jacome, UT Austin

EDF Example



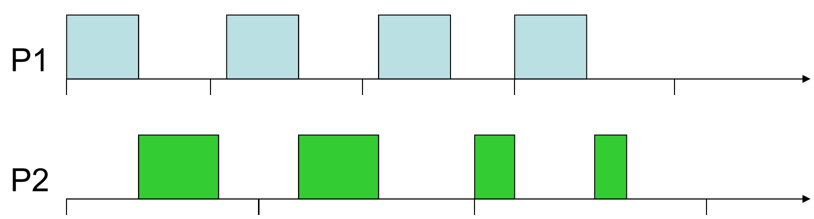
Source: M. Jacome, UT Austin

EDF Example



Lecture 5 J. Valvano, A. Gerstlauer EE445M/EE380L.6 Source: M. Jacome, UT Austin 17

EDF Example



Lecture 5 J. Valvano, A. Gerstlauer EE445M/EE380L.6 Source: M. Jacome, UT Austin 18

Priority Scheduling

- Execute highest priority first
  - Two tasks at same priority?
- Assign a dollar cost for delays
  - Minimize cost
  - Minimize latency on real-time tasks
  - Minimize maximum lateness (relative to deadline)

Priority Scheduler

- Assigns each thread a priority number
  - Reduce latency (response time) by giving high priority
  - Static (creation) or dynamic (runtime)
  - Performance measures (utilization, latency/lateness)
- Blocking semaphores and not spinlock semaphores
- Strictly run the ready task with highest priority at all times
  - Priority 2 is run only if no priority 1 are ready
  - Priority 3 only if no priority 1 or priority 2 are ready
  - If all have the same priority, use a round-robin system
- On a busy system, low priority threads may never be run
  - Problem: Starvation
  - Solution: Aging

How to find Highest Priority

- Search all for highest priority ready thread
  - Skip if blocked
  - Skip if sleeping
  - Linear search speed (number of threads)
- Sorted list by priority
  - Chain/unchain as ready/blocked
- Priority bit table (uCOS-II and uCOS-III)
  - See **OSUnMapTbl** in os_core.c
  - See **OS_Sched** (line 1606) Software\uCOS-II\Source
  - See **CPU_CntLeadZeros** in cpu_a.asm

Software\uC-CPU\Cortex-M3\RealView

Lecture 5 J. Valvano, A. Gerstlauer EE445M/EE380L.6 21

Adaptive Priority- Aging

- Solution to starvation
- Real and temporary priorities in TCB
- Priority scheduler uses temporary priority
- Increase temporary priority periodically
  - If a thread is not running
- Reset temporary back to real when runs

Lecture 5 J. Valvano, A. Gerstlauer EE445M/EE380L.6 22

Exponential Queue

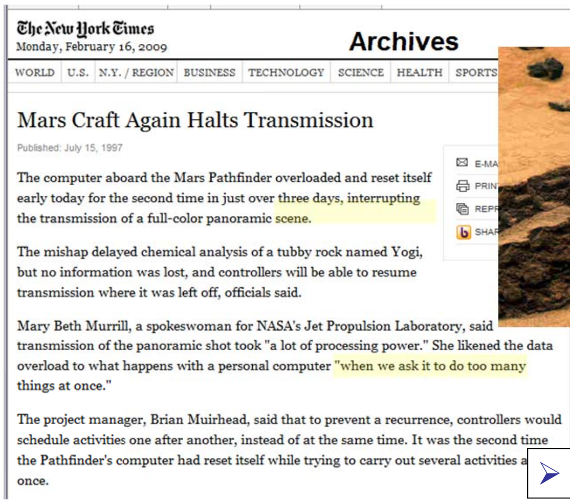
- Exponential comes from doubling/halving
  1. Round robin with variable timeslices
     - Time slices 8,4,2,1 ms
  2. Priority with variable priority/timeslices
     - Time slices 8,4,2,1 ms
     - Priorities 0,1,2,3

Final exam 2006, Q5

I/O Centric Scheduler

- Automatically adjusts priority
  - Exponential queue
- High priority to I/O bound threads
  - I/O needs low latency
  - Every time it issues an input or output,
    - Increase priority by one, shorten time slice
- Low priority to CPU bound threads
  - Every time it runs to completion
    - Decrease priority by one, lengthen time slice

Scheduling Anomalies


The New York Times
Monday, February 16, 2009
Archives
WORLD | U.S. | N.Y. / REGION | BUSINESS | TECHNOLOGY | SCIENCE | HEALTH | SPORTS

Mars Craft Again Halts Transmission

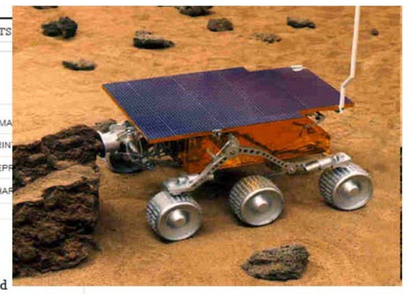
Published: July 15, 1997

The computer aboard the Mars Pathfinder overloaded and reset itself early today for the second time in just over three days, interrupting the transmission of a full-color panoramic scene.

The mishap delayed chemical analysis of a tubby rock named Yogi, but no information was lost, and controllers will be able to resume transmission where it was left off, officials said.

Mary Beth Murrill, a spokeswoman for NASA's Jet Propulsion Laboratory, said transmission of the panoramic shot took "a lot of processing power." She likened the data overload to what happens with a personal computer "when we ask it to do too many things at once."

The project manager, Brian Muirhead, said that to prevent a recurrence, controllers would schedule activities one after another, instead of at the same time. It was the second time the Pathfinder's computer had reset itself while trying to carry out several activities a[t] once.


Courtesy NASA/JPL-Caltech

- **Priority inversion**

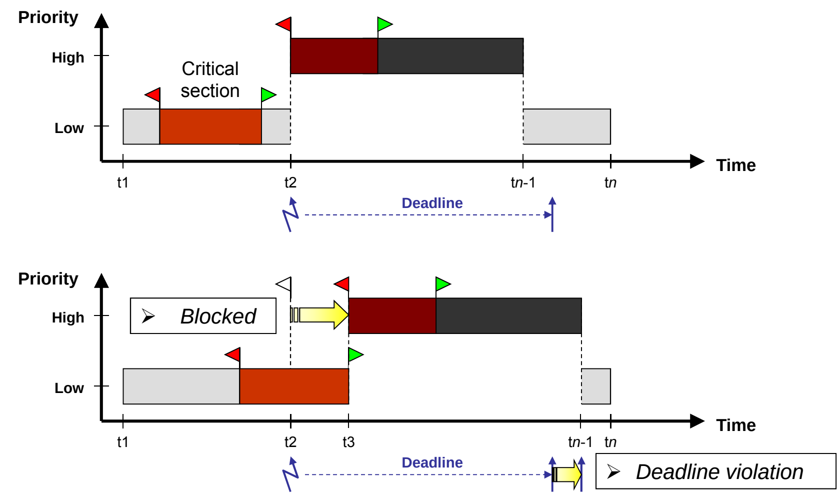
Lecture 5 J. Valvano, A. Gerstlauer EE445M/EE380L.6 25

Priority Inversion

- Low-priority process keeps high-priority process from running.
  - Low-priority process grabs resource (semaphore)
  - High-priority device needs resource (semaphore), but can't get it until low-priority process is done.
- Can cause deadlock

Lecture 5 J. Valvano, A. Gerstlauer EE445M/EE380L.6 26

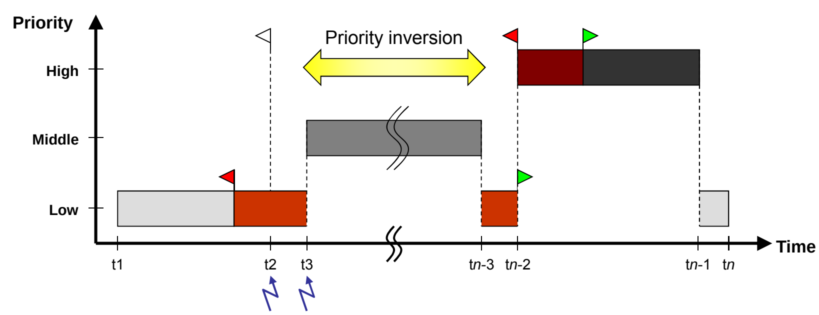
Priority-Based Scheduling



Lecture 5 J. Valvano, A. Gerstlauer EE445M/EE380L.6 27

Priority Inversion

- Low-priority process blocking high-priority
  - Starvation of high priority processes



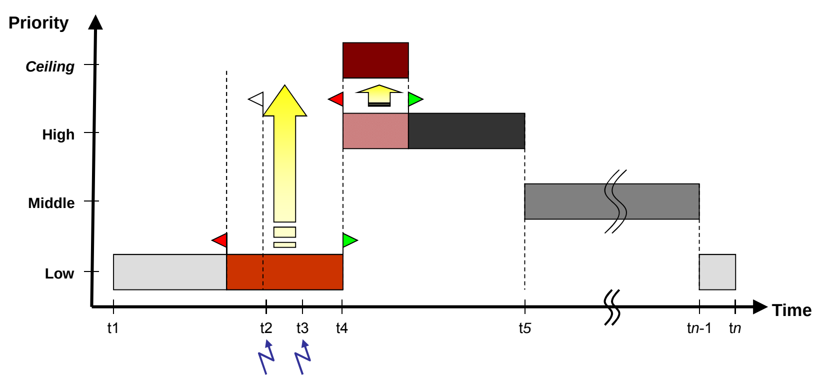
Lecture 5 J. Valvano, A. Gerstlauer EE445M/EE380L.6 28

Priority Inversion Solutions

- Avoid preemption in critical sections
  - Interrupt masking
  - Priority Ceiling Protocol (PCP)
  - Priority Inheritance Protocol (PIP)

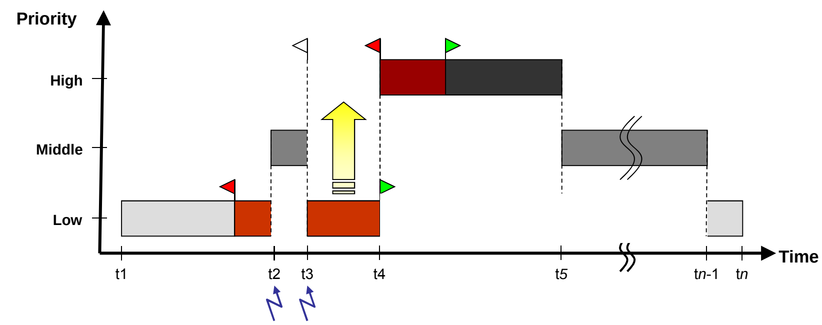
Priority Ceiling Protocol (PCP)

- Elevate priorities in critical sections
  - Assign priority ceilings to semaphore/mutex



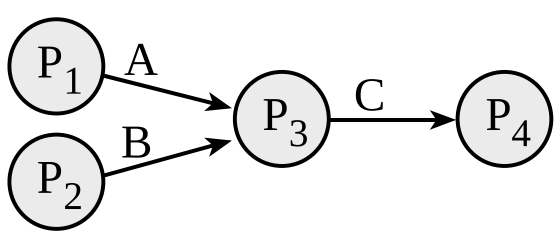
Priority Inheritance Protocol (PIP)

- Dynamically elevate only when needed
  - Raise priorities to level of requesting task



Kahn Process Network (KPN)

- Parallel programming model
  - Blocking read
  - Non-blocking writes (never full)
  - Tokens are data (no time stamp)



Kahn Process Network (KPN)

- Deterministic
  - Same inputs result in same outputs
  - Independent of scheduler
- Non-blocking writes (never full)
- Monotonic
  - Needs only partial inputs to proceed
  - Works in continuous time

Kahn Process Network (KPN)

```
void Process3(void){
long inA, inB, out;
  while(1){
    while(AFifo_Get(&inA)){};
    while(BFifo_Get(&inB)){};
    out = compute(inA,inB);
    CFifo_Put(out);
  }
}
```

```
void Process3(void){
long inA, inB, out;
  while(1){
    if(AFifo_Size()==0){
      while(BFifo_Get(&inB)){};
      while(AFifo_Get(&inA)){};
    } else{
      while(AFifo_Get(&inA)){};
      while(BFifo_Get(&inB)){};
    }
    out = compute(inA,inB);
    CFifo_Put(out);
  }
}
```

Kahn Process Network (KPN)

- Strictly bounded?
  - Prove it never fills (undecidable!)
  - Dependent on scheduler
- Termination
  - All processed blocked on input
- Scheduler
  - Needs only partial inputs to proceed
  - Works in real time

KPN Boundedness

- Try to find a mathematical proof
- Experimentally adjust FIFO size
  - Needs a realistic test environment
  - Profile/histogram DataAvailable for each FIFO
  - Leave the profile in delivered machine
- Dynamically adjust size with malloc/free
- Use blocking write (not a KPN anymore)
- Discard the data