Course Summary

Raj Rajkumar

Lecture #24

Final Exam

- Final Exam on Sunday, December 17, 2017
  - 8:30 - 11:30 AM
  - Baker Hall A51

COURSE SUMMARY

Goals of the Course

- Understand the scientific principles and concepts behind embedded real- time systems, and
- Obtain hands-on experience in programming embedded real-time systems (in the form of smartphones)
  - Understand the "big ideas" in embedded real-time systems.
  - Obtain direct hands-on experience.
  - Understand basic real-time resource management theory.
  - Understand the basics of embedded real-time system application concepts such as signal processing and feedback control.

Introduction to Embedded Systems

- What is an embedded system?
  - It is a device that needs to accomplish a particular set of tasks. Over time though, the distinction between an embedded system and a computer is becoming fuzzier but is still distinct in many cases.
- Attributes:
  - Reactive
  - Real-Time
- Typical Constraints:
  - Small Size, Low Weight
  - Low Battery, Harsh Environment
  - Safety-critical operation
  - Extreme Cost Sensitivity
  - Timing Constraints: Hard deadlines, Soft deadlines.

OS Basics

- Process Management:
  - Process States
  - Process Control Black
  - Threads *vs* Processes
  - Process creation: fork, exec
- RTOS: includes both logical and temporal correctness
- Resource Availability: Abundant, Insufficient, Sufficient but Scarce
- Tasks: Periodic, Sporadic and Aperiodic
- Scheduling: Static and Dynamic
- **R**ate-**M**onotonic **A**nalysis

Review Question

How many times will the following program print “Hello World!”?

```
#include <sys/types.h>
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>

int n = <some positive integer>;

int main() {
   int i;

   for (i = 0; i < n; i++) {
         fork();
         fork();
         printf("Hello World!\n");
   }
   exit(0);
}
```


Android

- From Google
- Uses Java
- Linux Kernel 2.6
- Dalvik Virtual Machine
- Provides Application Framework
- Application-level Power Management done through “wakelocks”
- Provides debugging interface through ADB

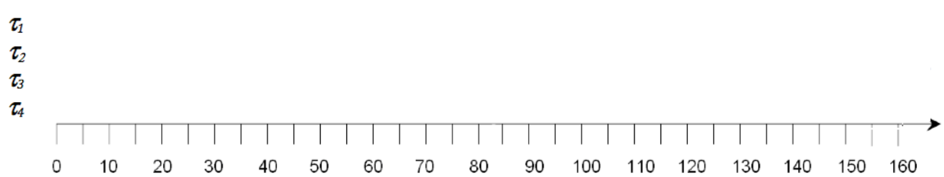
Uniprocessor Scheduling

- **Earliest Deadline First**:
  - Dynamic priority, Higher Priority to task with earlier deadline
  - Task-set is schedulable if U ≤ 1
  - If deadlines are shorter than periods, necessary condition for schedulability under EDF is an open problem.
  - If U > 1, which task will miss its deadline is unpredictable.
- **Rate-Monotonic Scheduling**:
  - Higher priority to task with shorter period
  - RMS leads to a feasible schedule if $U \leq n(2^{1/n} - 1)$
  - The above bound is sufficient but not necessary
- **Critical Instant**: The instant or relative phasing at which a task arrival encounters its worst-case response time.
- **Critical Zone**: The duration between the critical instant of a task instance and its completion.

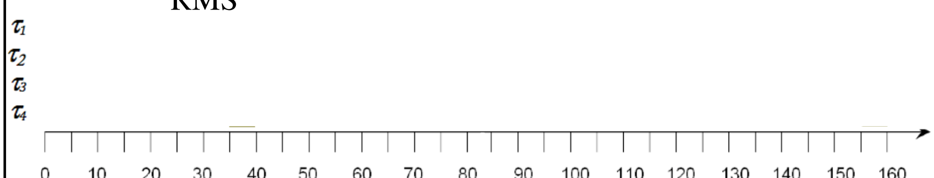
Review Question

Consider the following task-set with the following τ = (C, T=D) parameters: {(5, 10), (5, 20), (5, 40), (5, 80)}. Draw the time-lines for this task-set at least until t=120. Assume that all tasks are released at t=0. When deadlines or priorities are equal, break ties on a FCFS (first-come-first-served) basis.

EDF



RMS



Response Time Test

Consider the task set{C, T=D} = {(20, 100), (90, 150), (60, 300)}.

$$a_{k+1}^{i} = C_i + \sum_{j=1}^{i-1} \left\lceil \frac{a_k^i}{T_j} \right\rceil C_j \qquad \text{where } a_0^i = \sum_{j=1}^{i} C_j$$

Test terminates when $a_{k+1}^{i} = a_k^{i}$

For task τ_1

$$a_0^1 = 20$$

$$a_0^2 = C_1 + C_2 = 20 + 90 = 110$$

For task τ_2

$$a_1^2 = C_2 + \left\lceil \frac{110}{100} \right\rceil * 20 = 130$$

$$a_2^2 = 90 + \left\lceil \frac{130}{100} \right\rceil * 20 = 130$$

Response Time test

For task τ_3

$$a_0^3 = C_1 + C_2 + C_3 = 90 + 60 + 20 = 170$$

$$a_1^3 = C_3 + \left\lceil \frac{170}{100} \right\rceil * 20 + \left\lceil \frac{170}{150} \right\rceil * 90 = 280$$

$$a_2^3 = 60 + \left\lceil \frac{280}{100} \right\rceil * 20 + \left\lceil \frac{280}{150} \right\rceil * 90 = 300$$

$$a_3^3 = 60 + \left\lceil \frac{300}{100} \right\rceil * 20 + \left\lceil \frac{300}{150} \right\rceil * 90 = 300$$

Real Time Synchronization

- **Priority Inversion**: When lower-priority tasks cause higher-priority tasks to wait.
- Critical Section: The duration of a task using a shared resource.
  - Can potentially lead to unbounded delay of a task execution.
- Sources: Synchronization and mutual exclusion, Non-preemptible regions of code.
- **Basic Priority Inheritance Protocol**
  - Let the lower priority task use the highest priority of the higher priority task that it blocks.
  - There will be no deadlocks IF there are no nested locks, or application level deadlock avoidance scheme.
- **Priority Ceiling Protocol**
  - A priority ceiling is assigned to each mutex, which is equal to the highest priority task that may use this mutex.
  - A task can lock its mutex if and only if its priority is higher than the priority ceilings of all mutexes locked by other tasks.
  - If a task is blocked by a lower priority task, the lower priority task inherits its priority.
- **Highest Locker's Priority Protocol**
  - Execute critical section immediately at priority = priority ceiling of critical section

Review Question

- Consider the task set τ_1= (40, 100, 100), τ_2= (40, 200, 200), τ_3 = (150, 600, 600), which specifies the (C,T,D) triplet for each task τ_i. These tasks access one or more critical sections guarded by 3 mutexes M_1, M_2 or M_3.
  - Within its execution time, τ_1 accesses a critical section guarded by M_1 for 15 ms; and after normal execution, it enters a critical section guarded by M_2 for 18 ms.
  - During its execution, τ_2 accesses a critical section guarded by M_2 for 13 ms, and after normal execution, it enters a critical section guarded by M_3 for 10 ms.
  - During its execution, τ_3 accesses a critical section guarded by M_1 for 20 ms and after normal execution, it enters a critical section guarded by M_3 for 14 ms.
- Blocking time when using:
  - PIP? PCP? Highest locker? Non-preemption protocol?

POSIX pthreads

- POSIX is a standard set of interfaces for operating systems, standardized by the IEEE.
- Pthreads is the set of POSIX interfaces for threads
- Support interfaces for
  - Threads management, thread attributes
  - Mutex management, mutex attributes
  - Condition variable management, condition variable attributes
- Real-time scheduling policies to support basic priority inheritance and priority ceiling protocols are included.

RTOS Options

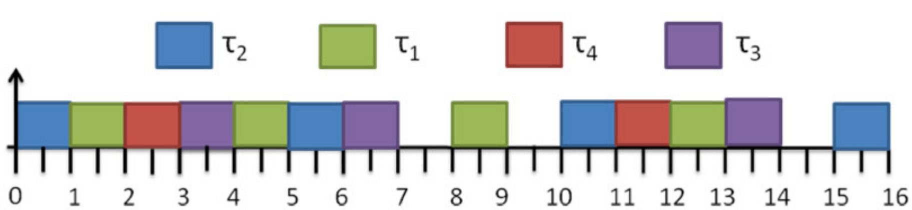
- **OS Approaches limiting Real-time and Non-Real-time Task interactions**
  - Compliant Kernel Approach: 100% Linux API, Any application can run on real-time kernel
  - Thin Kernel Approach: RT-Linux or RTAI
- **OS Approached that integrate Real-time and Non Real-time tasks**
  - Core Kernel Approach: Make the kernel suitable for real-time while changes to the kernel are localized. Allows the use of most if not all existing Linux primitives, applications and tools.
  - Resource Kernel Approach : Provides applications Timely, Guaranteed, and Enforces access to System Resources. Applications need to specify their resource demands only.

Deadline-Monotonic Scheduling

- Assign fixed priority based on D (and not T)
  - Shorter the relative deadline, higher the priority.
- When D=T, RMS and DMS are one and the same.
- When D > T, neither RMS nor DMS is the optimal fixed-priority scheduler.
- DMS is optimal among fixed-priority preemptive schedulers for periodic real-time tasks when D <= T.
- RMS is optimal among fixed-priority preemptive schedulers for periodic real-time tasks when D = T.
- The general set of principles for analyzing fixed-priority preemptive scheduling policies is called RMA (rate-monotonic analysis).

Review Question

Task	C_i	T_i	D_i	Priority
τ_1	1	4	4	
τ_2	1	5	3	
τ_3	1	6	6	
τ_4	1	10	5	



Carnegie Mellon 18-648: Embedded Real-Time Systems Electrical & Computer ENGINEERING

Periodic Servers

- **Polling Server**:
  - Simple
  - Commonly used
  - WCRT can be long.
- **Deferrable Server**:
  - Improves response time
  - Cannot be generalized to multiple instances at different priority levels.
- **Sporadic Server**:
  - Improves upon the deferrable server and is very generalizable
  - Higher run-time complexity

Review Question

Draw the execution timeline for the tasks $\tau_{1:}$ (C = 2, D = T = 8), τ_2: (2,10) and a deferrable server task τ_s: (2,6) till t=24 where:

- *All the tasks are scheduled using RMS.*
- *The arrival and computation times of aperiodic tasks associated with the server are:*

Arrival Time	Computation Time
5	2
9	1
12	2
16	1

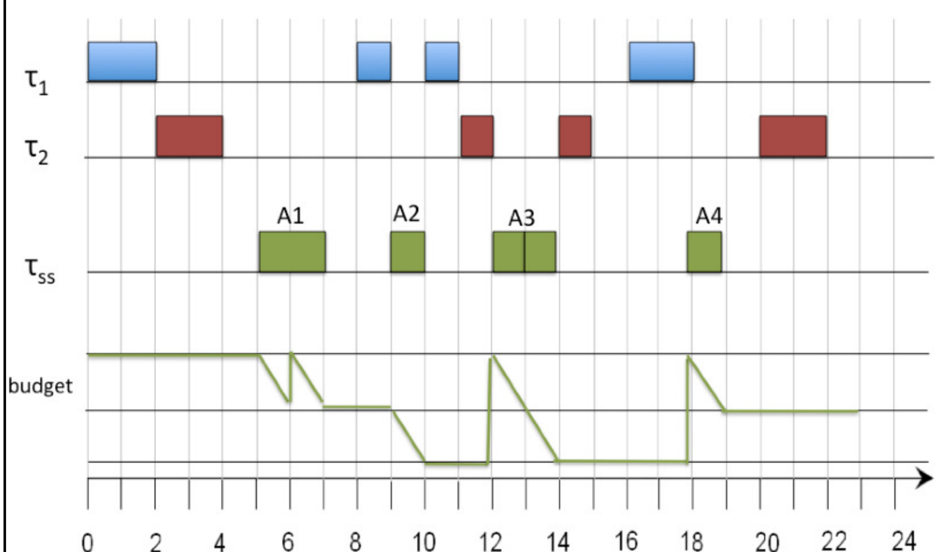
Review Question

Draw the execution timeline for the tasks τ_1: (C = 2, D = T = 8), τ_2: (2, 10) and a <u>sporadic server</u> task τ_s(2,6) till t=24 where:

- *All the tasks are scheduled using RMS.*
- *The arrival and computation times of the aperiodic task are given below:*

Arrival Time	Computation Time
5	2
9	1
12	2
16	1

Review Question



Power Management

- Power is proportional to V^2f
- Scaling Techniques:
  - DFS (dynamic frequency scaling)
  - DVS (dynamic voltage scaling)
  - DVFS (dynamic voltage and frequency scaling)
- Static Voltage Scaling Algorithm
  - **Sys-Clock**: Optimal system-wide clock frequency assignment
  - **PM-Clock**: Task-specific clock frequency assignment
    - Longer the period, lower the clock frequency
- Static Voltage Scaling with EDF
- Cycle-conserving EDF
- Harmonized scheduling
  - **Rate-Harmonized Scheduling**
  - **Energy-Saving Rate-Harmonized Scheduling**

Review Question

You are given two tasks τ_1 and τ_2 defined as $\tau_1 = (C = 3, T = 6, D = 5)$ and $\tau_2 = (1, 30, 30)$. Assume that the maximum frequency you can run the processor is f_{max}.

What is the Sys-Clock frequency?
Ans:

What are the clock-frequency assignments under PM-Clock?
Ans:

0.6
Task 1: 0.6 Task 2: 0.2

Review Question

Consider the following taskset scheduled on a micro-controller with a sleep duration C_{sleep} of 8.

$\tau_1 = \{C_1 = 4, T_1 = 16\}$

$\tau_2 = \{C_2 = 2, T_2 = 27\}$

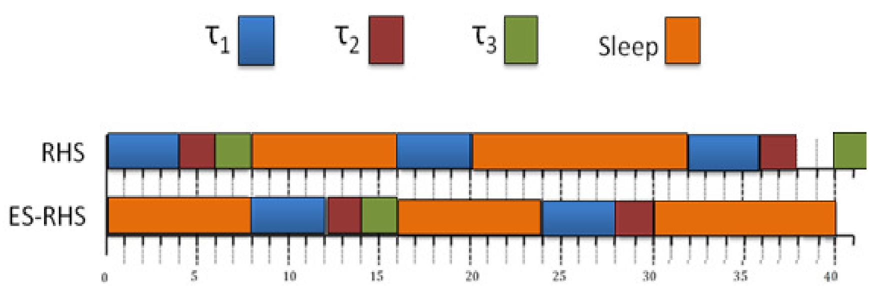
$\tau_3 = \{C_3 = 2, T_3 = 40\}$

Simulate the timeline up to 40 for the given tasks under

RHS with a harmonizing period of 8

ES-RHS with a harmonizing period of 16

Answer to Review Question



Multi-processor Scheduling

- *Global scheduling* and *partitioned scheduling* are two approaches to multiprocessor scheduling.
- Timing behavior under task scheduling strategies can be brittle. Small changes can have big (unexpected) consequences.
- Bin-Packing Heuristics for partitioned scheduling.
  - The problem is known to be NP-complete.
  - Very efficient near-optimal heuristics exist.
  - Problems with bin-packing?
- Task Splitting to circumvent 50% bound
  - Highest-priority task splitting
  - Period transformation
  - Task splitting for harmonics

Review Question

Consider the following task-set with 6 tasks (Assume $D_i=T_i$). Allocate the tasks to processors when tasks cannot be split. Mission: Minimize bin count.

τ_1: $\{C_1 = 4, T_1 = 10\}$

τ_2: $\{C_2 = 6, T_2 = 14\}$

τ_3: $\{C_3 = 4, T_3 = 16\}$

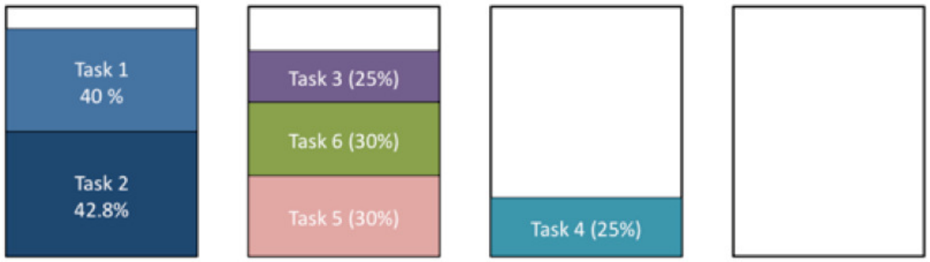
τ_4: $\{C_4 = 4, T_4 = 16\}$

τ_5: $\{C_5 = 3, T_5 = 10\}$

τ_6: $\{C_6 = 3, T_6 = 10\}$

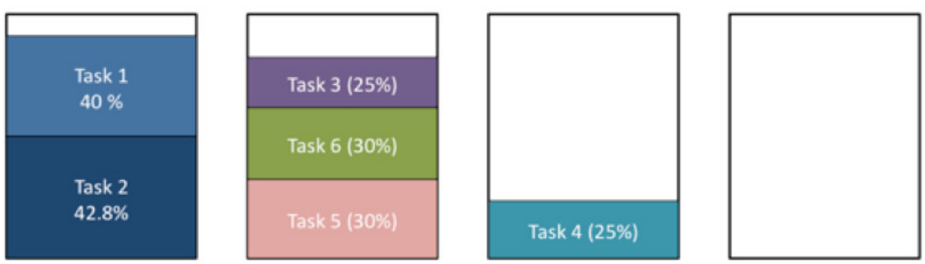
Review Question

First Fit Decreasing



Review Question

Best-Fit Decreasing



Review Question

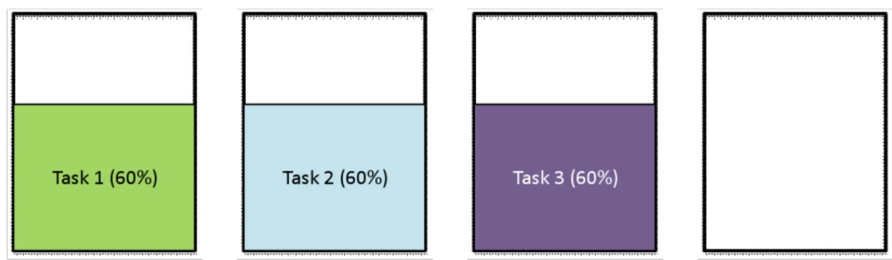
Consider the following task-set with 3 tasks (Assume D_i=T_i).

$\tau_1 = \{C_1 = 3, T_2 = 5\}$

$\tau_2 = \{C_2 = 6, T_2 = 10\}$

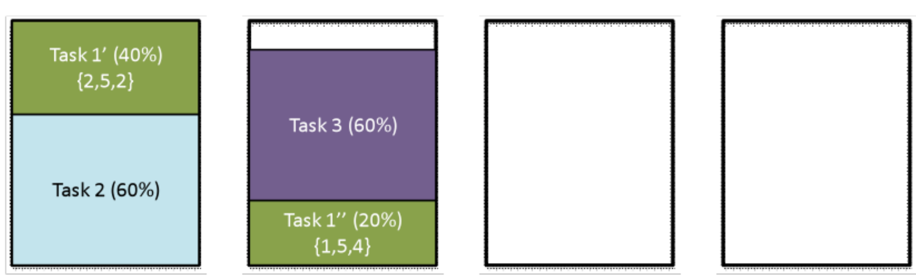
$\tau_3 = \{C_3 = 12, T_3 = 20\}$

BFD without task-splitting:



Review Question

BFD with task-splitting



Real-Time Communications

- Timely delivery may be deemed more desirable than reliable delivery.
- There are many causes of delay:
  - Queuing delay at sender, receiver, network and the propagation delay.
- Throughput and delay depend on the capacity of the link.
- The requirements of throughput and delay are application dependent.
- The buffering depends on the jitter.
- **Controller Area Network** (**CAN**):
  - Connections wired together as a logical AND function.
  - Distributed Priority-based Arbitration.
- **FlexRay**
  - Delivers deterministic, fault-tolerant and high-speed bus system.

Feedback Control

- Feedback control
  - Stability
  - Instability
  - Marginal stability
- Feedback controllers
  - Proportional control
  - Proportional + derivative control
  - Proportional + derivative + integral control

Signal Processing

- Source of deterministic errors and random noises
- Basics of signal spectrum
  - Nyquist sampling
    - the sampling rate must be at least two times the bandwidth, the highest frequency in the signal, to avoid aliasing. This is known as the **Nyquist Rate**.
  - Fourier transform
- Basic filters
  - If there is no aliasing, then the signal can be recovered perfectly *in theory* using ideal low-pass filters.

Final Exam

- An expanded, sometimes more in-depth, version of the 5 quizzes to date.
  - True/False
  - Short questions
  - Long questions with multiple sub-parts
- Bring a calculator, pencil and eraser.
- Do *not* get stuck on any question.
  - Skip to later/easier ones and return to tougher ones later
- Time limits will be strictly enforced.

BEST WISHES