Announcement

You have learned to program concurrent tasks for real time applications. When you have many real time tasks.

- How do you know if the deadline of all the tasks can be met?
- What happens if you modified a task, can the rest of task still meet their deadlines.
- ...

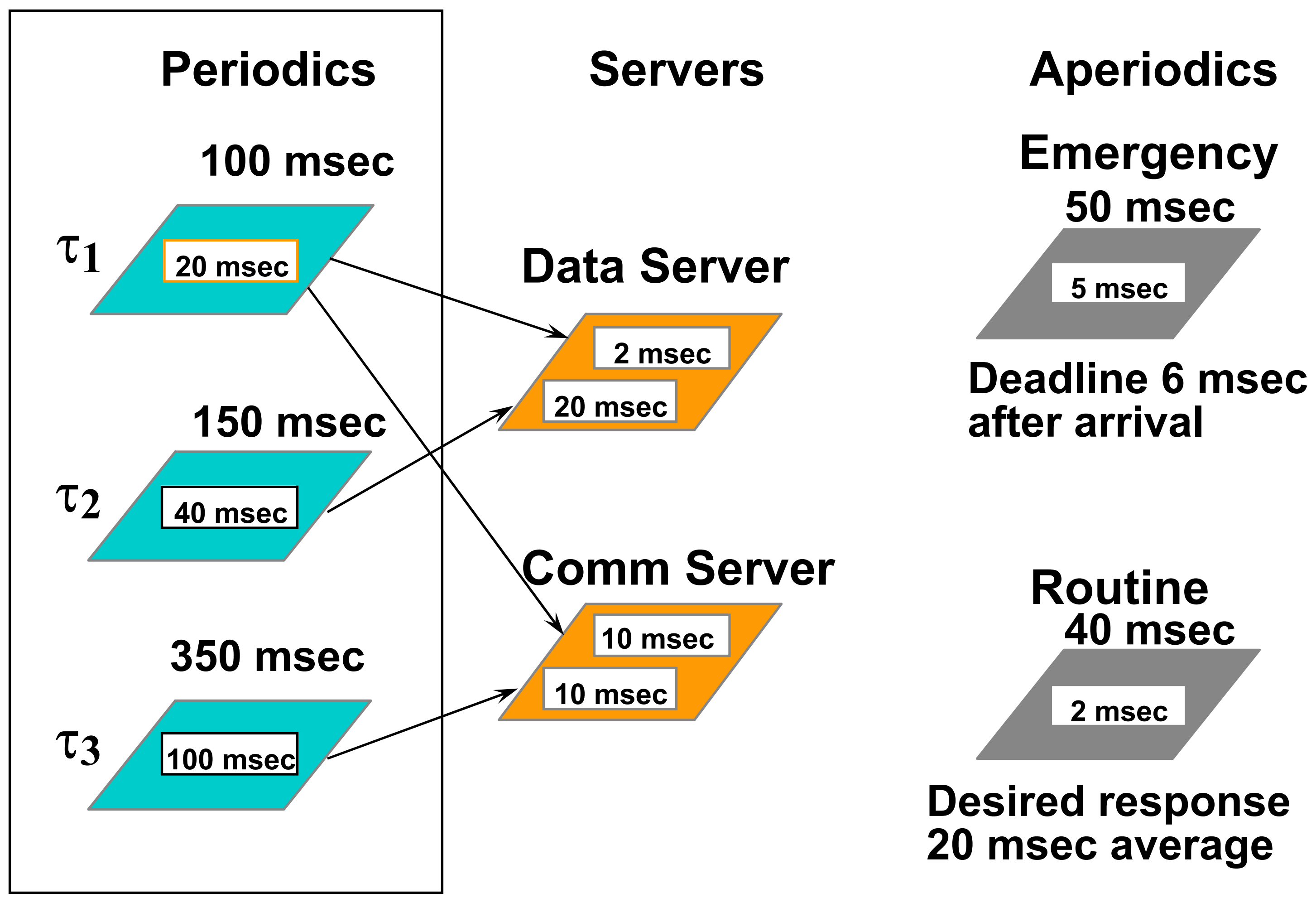
To answer these questions, we need to learn the basics of real time scheduling theory.

Reference: Sha, L., and Goodenough, J. B., “Real-Time Scheduling Theory and Ada”*, IEEE Computers*, April 1990. We will put a stack in the lab for you to pick up.

Measure of Merit

	Real time systems	*Non-real time systems*
performance	Schedulable Utilization (schedulability)	*throughput*
responsiveness	Worst case response time of each tasks	*Average response time*
overload	*Stability (getting critical tasks done)*	*Fairness*

A Sample Problem



What is GRMS for?

As the hardware becomes much faster and the RTOS market is gaining wide spread usage, the key required skills are no longer assembly programs executing on bare custom hardware.

The skills in demand now become the development of large and sophisticated multi-threaded real time applications.

GRMS is a practical theory that will enable you to design and analyze of multi-thread real time applications.

GRMS in The Real World

"A major payoff...System designers can use this theory to predict whether task deadlines will be met long before the costly implementation phase of a project begins. It also eases the process of making modifications to application software." DoD *1991 Software Technology Strategy*. pp. 8-15.

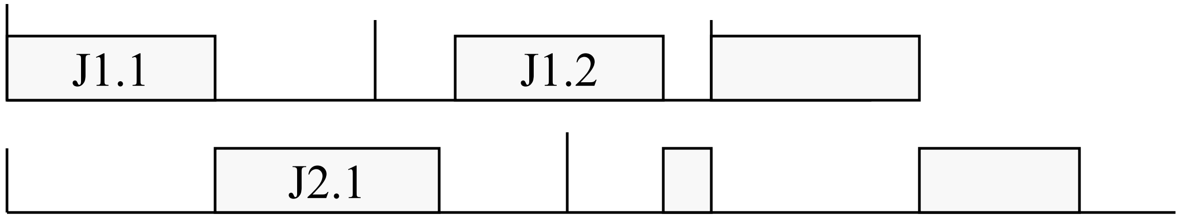
"Through the development of [Generalized] Rate Monotonic Scheduling, we now have a system that will allow [Space Station] Freedom's computers to budget their time, to choose between a variety of tasks, and decide not only which one to do first but how much time to spend in the process."
--- Aaron Cohen, former deputy administrator of NASA, "Charting The Future: Challenges and Promises Ahead of Space Exploration", October, 28, 1992, p. 3.

It also provides the theoretical foundation to solve MARS Pathfinders reset problem while it was on MARS. http://catless.ncl.ac.uk/Risks/19.49.html.

GRMS is supported by almost all the commercially available RTOS

Dynamic vs "Static" Priority Scheduling in Theory

- An instance of a task is called a job. "Static" priority assigns a (base) priority to all the jobs in a task. Dynamic priority scheduling adjust priorities in each task job by job.



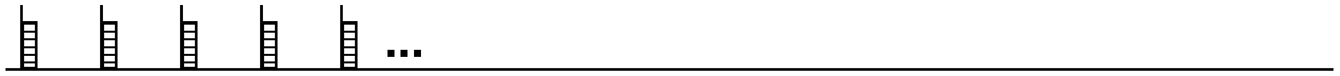
- <u>what type of scheduling algorithm is used to schedule these two tasks? (answer EDF)</u>

- <u>An</u> optimal dynamic scheduling algorithm is the earlier deadline first (EDF) algorithm. Jobs closer to deadlines will have high priority. With independent periodic tasks, all tasks will meet their deadlines as long as the processor utilization is less than 1.

- <u>An</u> optimal "static" scheduling algorithm is the rate monotonic scheduling (RMS) algorithm. For a periodic task, the higher the rate (frequency) the higher the priority.

Periodic Tasks

- A task τ_i is said to be periodic if its inter-arrival time (period), T_i, is a constant.

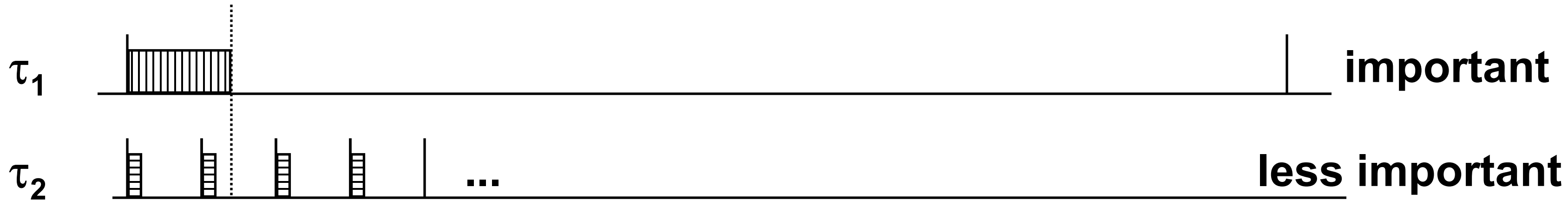


- Periodic tasks are common in real time systems because of the sampling actions.
- (Can you give some examples?)
- The utilization of task τ_i, is the ratio between its execution time C_i and its period T_i: $U_i = C_i / T_i$
- The default deadline of a task is the end of period.

Importance and Priority

- Task τ_1 : if it does not get done in time, the world will end.
- Task τ_2: if it does not get done in time, you may miss a sweet dream.
- Quiz: presume that the world is more important than your dream, should task τ_1 has a higher priority?

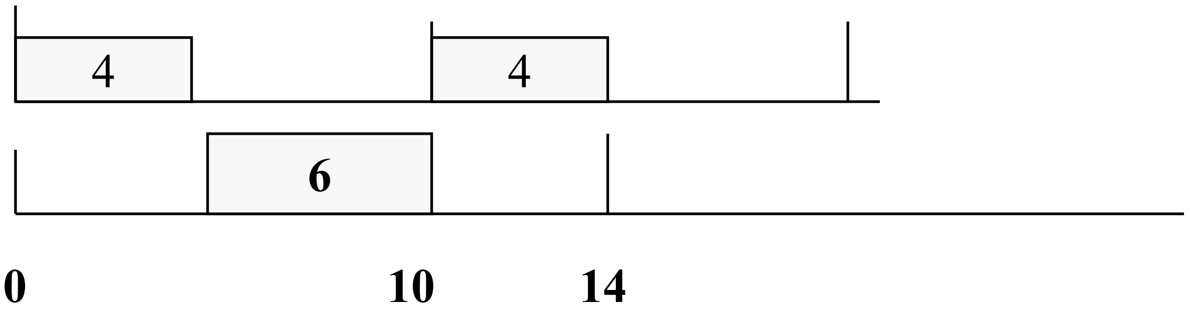
Why Ever Faster Hardware is Not Enough



- If priorities are assigned according to importance, there is <u>no lower bound of processor utilization, below which tasks deadlines can be guaranteed.</u> Why?

- $C_1/T_1 + C_2/T_2 = U$
- $U \rightarrow 0$, when $C_2 \rightarrow 0$ and $T_1 \rightarrow \infty$
- Task τ_2 will miss its deadline, as long as $C_1 > T_2$

Less Than 100% Utilization but not Schedulable



In this example, 2 tasks are scheduled under RMS, an optimal static priority method

4/10 + 6/14 = 0.83

note: if tasks periods are harmonic, e.g, 2, 4, 8, 16 etc, the utilization bound is 1

Schedulability: UB Test

- Utilization bound(UB) test[Liu73]: a set of n independent periodic tasks scheduled by the rate monotonic algorithm will always meet its deadlines, for all task phasing, if
- $$\frac{C_1}{T_1} + \dots + \frac{C_n}{T_n} \leq U(n) = n(2^{1/n} - 1)$$
- U(1) = 1.0 U(4) = 0.756 U(7) = 0.728
- U(2) = 0.828 U(5) = 0.743 U(8) = 0.724
- U(3) = 0.779 U(6) = 0.734 U(9) = 0.720

- For harmonic task sets, the utilization bound is U(n)=1.00 for all n. For large n, the bound converges to *ln 2* ~ 0.69.

- Conventions, task 1 has shorter period than task 2 and so on.

Sample Problem: Applying UB Test

	C	T	U
Task $\tau1$:	**20**	**100**	**0.200**
Task $\tau2$:	**40**	**150**	**0.267**
Task $\tau3$:	**100**	**350**	**0.286**

- Are all the tasks schedulable?
- What if we double the execution time of task $\tau1$?

Summary

- Today, we have introduced GRMS and the scheduling of periodic task.
- Next, we will take a more in depth look on the scheduling of periodic tasks
- We will then study synchronization protocols, the scheduling of aperiodic tasks and finally putting all them together.