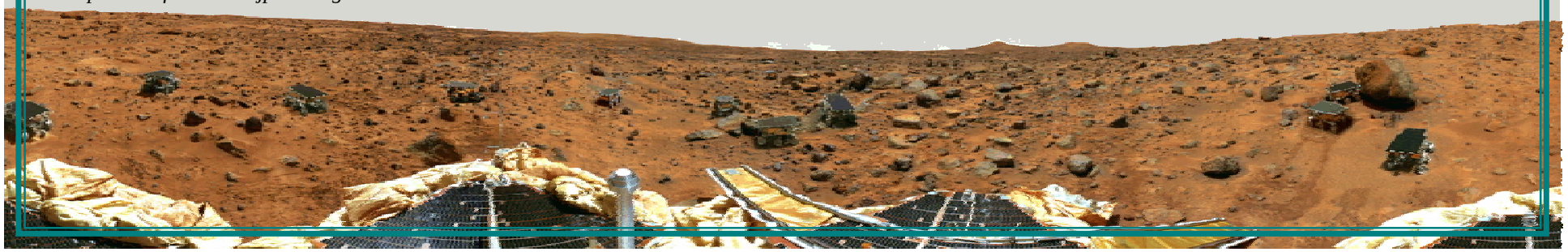
Rate Monotonic Analysis

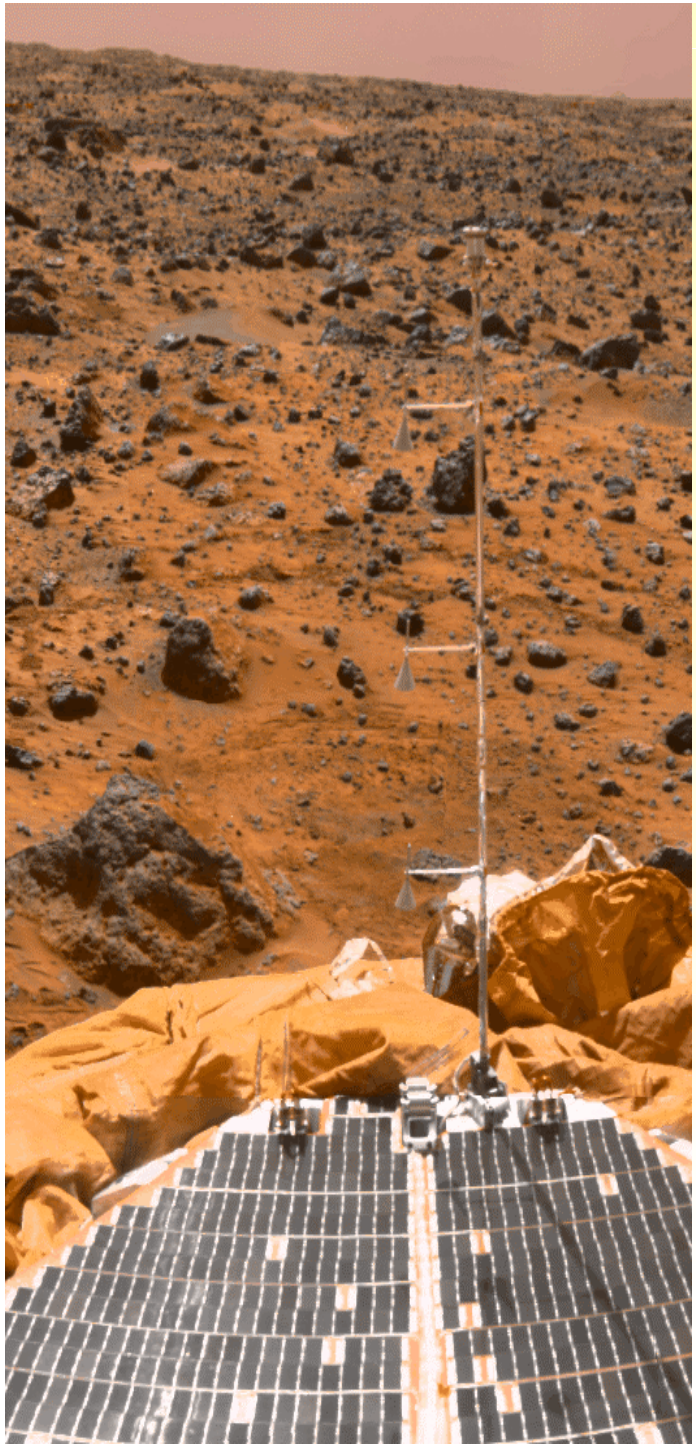
A Presentation by
Nate Forman, Motorola Inc.

For Dr. Brian Evans'
Embedded Systems Class
Fall, 1999

Executive Software Engineering Program
University of Texas at Austin

Mars pictures from mars.jpl.nasa.gov

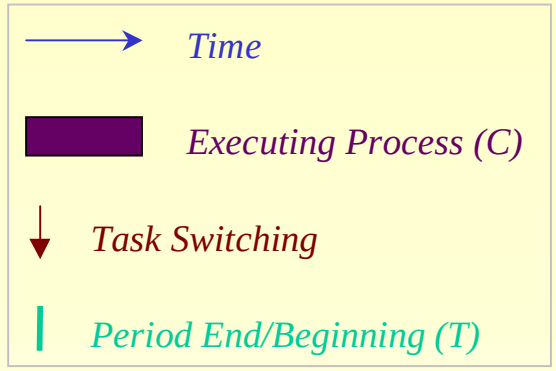


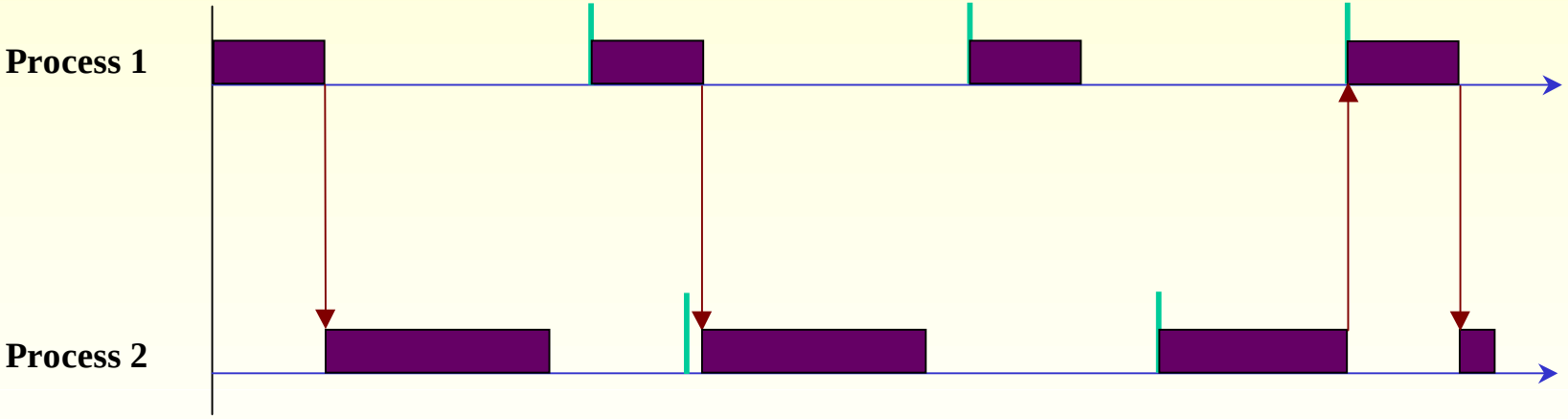


Agenda

- Basic Rate Monotonic Theory
- Schedulability Tests
- Extensions
- RMADriver Application
- Mars Lander Discussion

Task Modelling for RMA





Rate Monotonic Rules

- Instantaneous task switching
- Tasks account for all execution time
- Task interactions are not allowed
- Tasks become ready at the beginning of their period and relinquish the CPU when execution is complete
- The end of a task's period is its deadline
- Priority is assigned inversely by length of period
- Lower priority tasks never execute when a higher priority task is ready

Utilization Bound (UB) Test

- Asymptotic processor utilization bound for n rate monotonic processes:

$$U(n) = n\,(2^{1/n} - 1)$$

- Processor utilization for n rate monotonic processes:

$$C_1/T_1 + \ldots + C_n/T_n$$

- Schedulability:

$$(C_1/T_1 + \ldots + C_n/T_n) < U(n) \Rightarrow \text{schedulable}$$

$$U(n) < (C_1/T_1 + \ldots + C_n/T_n) < 1 \Rightarrow \text{inconclusive}$$

$$1 < (C_1/T_1 + \ldots + C_n/T_n) \Rightarrow \text{unschedulable}$$

Response Time (RT) Test

A more precise test to apply when the UB test is inconclusive.

For a set of independent, periodic tasks,
if each task meets its first deadline, with
worst-case task phasing, the deadline
will always be met.

Apply the following calculation recursively until a least fixed point is reached:

$$a_{n+1} = C_i + \sum_{j=1}^{i-1} \left\lceil \frac{a_n}{T_j} \right\rceil C_j \qquad a_0 = \sum_{j=1}^{i} C_j$$

Each task j is a task with higher priority than the current task. The a_0 calculation includes the current task. If the value of a_n for the least fixed point is less than the task's deadline, the task is schedulable

Extensions

- Nonzero task switching times
- Preperiod task deadlines
- Aperiodic task handling with sporadic servers
- Interrupt handling for top-priority tasks
- Blocking from shared resources

UML Class Diagram