

“Scheduling Algorithms for
Multiprogramming in a
Hard-Real-Time Environment”
Authors: Liu & Layland

Research Seminar Fall 2001

Venkita Subramonian

venkita@cs.wustl.edu

www.cs.wustl.edu/~venkita

Agenda

- Real-time systems
- Scheduling
- Rate monotonic scheduling
- Deadline driven scheduling
- Hybrid scheduling

Real time Systems

- Guaranteed execution of tasks
- Required to complete tasks in a timely manner
- Examples – digital control, real-time command and control, signal processing
- Often run “under the covers”

Hard vs Soft real-time

- Hard real-time
 - Guaranteed response time
 - Aircraft safety controls
 - Missile systems
- Soft real-time
 - Execution may not be guaranteed
 - Statistical distribution of response time acceptable
 - Online transaction systems, telephone switches, electronic games

Hard Real-time systems

- Need to work correctly and responsively
- More Important – Need to be able to demonstrate the timeliness of the system using *validation* techniques
- Need **proof** that the time constraints are met

Terminology

- **Deadline**
 - Time at which task should complete execution
- **Overflow**
 - Unfulfilled request
- **Response time**
 - Time to fulfill the request. (elapsed time)
- **Run Time**
 - time taken without interruption
 - vs Response time

Terminology (contd)

- Critical Instant
 - Instant at which request will have the largest response time
- Critical time zone
 - Time interval between a critical instant and the deadline for the task
- Feasible priority assignment
 - All tasks are able to run to completion before their deadlines

Scheduling

- Static
 - Schedule determined *a priori*
- Dynamic
 - Schedule determined at run-time
- Priority Driven
 - Each task assigned a priority
- Fixed priority assignment
 - Priorities assigned statically
- Preemptive
 - Higher priority task preempts lower priority task

Notations

- Task – J_i
- Request Period – T_i
- Run Time – C_i
- Priority of Task J_i – $P(J_i)$

RM Assumptions

- A1. Requests for all tasks are periodic
 - constant interval between requests

- A2. Deadline consists of run-ability constraints only
 - deadline for a task same as end of interval between requests for the task

RM Assumptions (contd)

A3. Tasks are independent

- Not dependent on initiation or completion of other tasks

A4. Run-time for task is constant

- time taken by a processor to execute the task is a constant
- does not vary over time

RM Assumptions (contd)

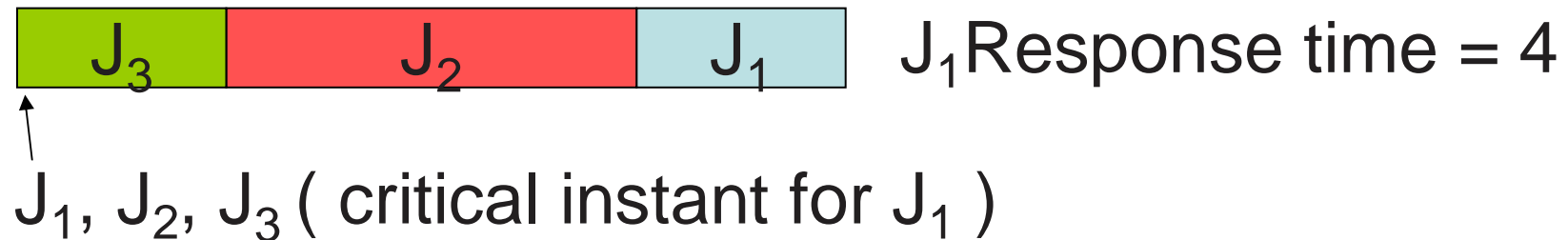
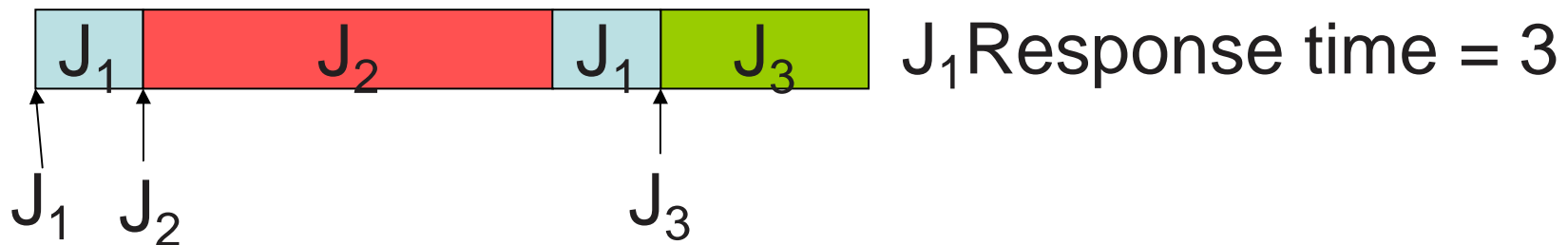
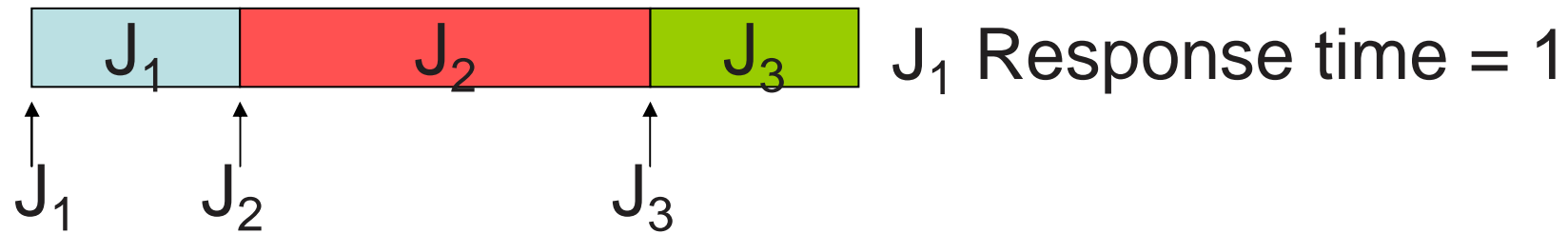
- A5. Non-periodic tasks are special
- initialization/failure-recovery routines
 - do not have hard critical deadlines
 - displace periodic tasks

Maximum response time – an example

Tasks	C_i
J_1	1
J_2	2
J_3	1

$$P(J_1) < P(J_2) < P(J_3)$$

Maximum response time – an example (contd)



A critical instant for any task occurs whenever the task is requested simultaneously with requests for all higher priority tasks

[Theorem 1 - Liu and Layland]

Priority assignment

- Given $T_1 < T_2$
 - $P(J_1) > P(J_2)$
 - $\text{floor}(T_2/T_1) * C_1 + C_2 \leq T_2$
 - $P(J_2) > P(J_1)$
 - $C_1 + C_2 \leq T_1$
- $C_1 + C_2 \leq T_1$ implies
 - $\text{floor}(T_2/T_1) * C_1 + \text{floor}(T_2/T_1) * C_2 \leq \text{floor}(T_2/T_1) * T_1$
 - $\text{floor}(T_2/T_1) * C_1 + \text{floor}(T_2/T_1) * C_2 \leq T_2$
 - $\text{floor}(T_2/T_1) * C_1 + C_2 \leq T_2$
- Any assignment feasible with $P(J_2) > P(J_1)$ is also feasible with $P(J_2) < P(J_1)$

RM Priority Assignment

- Rate monotonic priority assignment
 - Assign higher priority to tasks with higher request rate
 - Optimum assignment
 - higher processor utilization

If a feasible [fixed] priority assignment exists for some task set, the rate-monotonic priority assignment is feasible for that task set

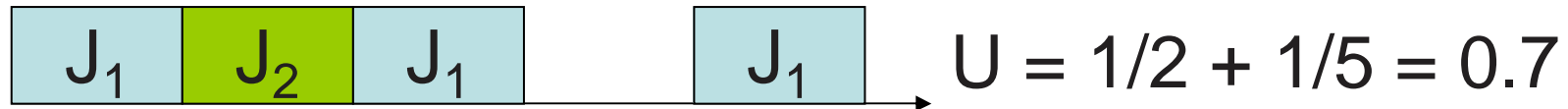
[Theorem 2 - Liu and Layland]

Processor Utilization

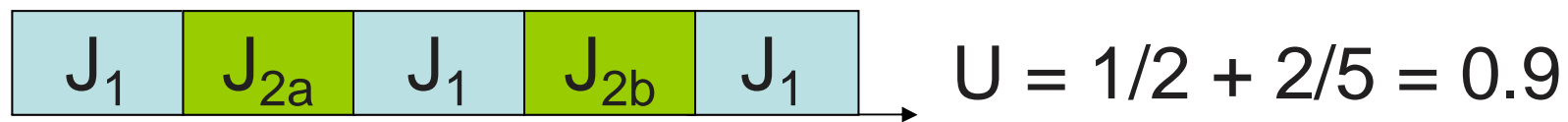
- Utilization factor
 - $U \geq (C_i/T_i), i = 1, 2, \dots, m$
- Full utilization
 - $U = 1.0$
- Maximum Feasible Utilization
 - If priority assignment feasible
 - Any increase in run-time of any task would make the priority assignment infeasible
- Least Upper Bound on Utilization factor
 - For a given fixed priority assignment, the minimum of the utilization factors over all set of tasks that fully utilize the processor

Utilization - Example

$$T_1 = 2, C_1 = 1, T_2 = 5, C_2 = 1, P(J_1) > P(J_2)$$



$$T_1 = 2, C_1 = 1, T_2 = 5, C_2 = 2, P(J_1) > P(J_2)$$



J_1, J_2

$$T_1 = 2, C_1 = 1, T_2 = 5, C_2 = 1, P(J_2) > P(J_1)$$



$$T_1 = 2, C_1 = 1, T_2 = 5, C_2 = 2, P(J_1) > P(J_2)$$



$$U = 1/2 + 2/5 = 0.9$$

For a set of two tasks with fixed priority assignment, the least upper bound to the processor utilization factor is $U = 2(2^{1/2}-1)$

$$U_{min} \approx 0.83$$

[Theorm 3 - Liu and Layland]

Proof

- Given $T_2 > T_1$
- Case 1:
$$C_1 \leq T_2 - T_1 * \text{floor}(T_2/T_1)$$
$$\max C_2 = T_2 - C_1 * \text{ceiling}(T_2/T_1)$$
- Case 2:
$$C_1 \geq T_2 - T_1 * \text{floor}(T_2/T_1)$$
$$\max C_2 = T_1 * \text{floor}(T_2/T_1) - C_1 * \text{floor}(T_2/T_1)$$
- Minimum U occurs when
$$C_1 = T_2 - T_1 * \text{floor}(T_2/T_1)$$

For a set of m tasks with fixed priority, and the restriction that the ratio between any two request periods is less than 2, the least upper bound to the processor utilization factor is

$$U = m(2^{1/m} - 1)$$

For large m , $U \approx \ln 2$

[Theorem 4 - Liu and Layland]

Proof

- “*ratio between any two request periods is less than 2*”

$$\text{floor}(T_2/T_1) = 1 \text{ and } \text{ceiling}(T_2/T_1) = 2$$

Minimum U occurs when

$$C_1 = T_2 - T_1 * \text{floor}(T_2/T_1)$$

$$C_1 = T_2 - T_1$$

Similarly

$$C_2 = T_3 - T_2$$

$$C_3 = T_4 - T_3$$

.....

$$C_{m-1} = T_m - T_{m-1}$$

For a set of m tasks with fixed priority order, the least upper bound to processor utilization is

$$U = m(2^{1/m} - 1)$$

[Theorem 5 - Liu and Layland]

100% Utilization

- Least upper bound with RM priority assignment $\leq 100\%$
- How to make least upper bound for utilization = 100%
 - T_m is a multiple of T_i for $i=1,2,3,\dots,m-1$
(RM with harmonic periods)
 - Dynamic priority assignment

Deadline driven scheduling

- Task with earliest deadline given highest priority
- Dynamic priority assignment
- Optimum – if a set of tasks can be scheduled by some priority assignment, it can be scheduled using deadline driven priority assignment

When the deadline driven scheduling algorithm is used to schedule a set of tasks on a processor, there is no processor idle time prior to an overflow

[Theorem 6 - Liu and Layland]

For a given set of m tasks, the deadline driven scheduling algorithm is feasible if and only if

$$(C_1/T_1) + (C_2/T_2) + \dots + (C_m/T_m) \leq 1$$

[Theorem 7 - Liu and Layland]

Proof

- Schedule infeasible if total demand for computation time exceeds the available processor time.

- Necessity proof

$$T = T_1 \cdot T_2 \cdot T_3 \cdot \dots \cdot T_m$$

$$(T/T_1)C_1 + (T/T_2)C_2 + \dots + (T/T_m)C_m > T$$

$$(C_1/T_1) + (C_2/T_2) + \dots + (C_m/T_m) > 1$$

Proof (contd)

- Sufficiency proof

Assume necessity condition is satisfied and yet the scheduling is not feasible

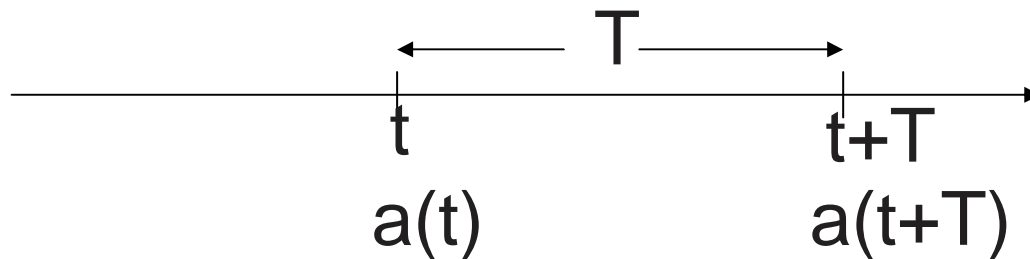
There exists $T \mid 0 \leq T \leq T_1.T_2.T_3.....T_m$, where an overflow occurs

$$\text{floor}(T/T_1).C_1 + \text{floor}(T/T_2).C_2 + \dots + \text{floor}(T/T_m).C_m > T$$

$$(T/T_1).C_1 + (T/T_2).C_2 + \dots + (T/T_m).C_m > T$$

Mixed Scheduling Algorithm

- Combination of RM and deadline driven scheduling
- Shortest tasks 1..k are scheduled using RM and k+1..m are scheduled using deadline driven algorithm
- Processor availability function



Mixed Scheduling (contd)

- Tasks 1..k scheduled thru RM, k+1..m scheduled thru deadline driven schedule, $a_k(t)$ is the availability function w.r.t tasks k+1..m
- $a_k(t)$ is a non-decreasing function of t
- $a(t)$ is sublinear if
$$a(T) \leq a(t+T) - a(t)$$

If a set of tasks are scheduled by the deadline driven scheduling algorithm on a processor whose availability function is sublinear, then there is no processor idle time prior to an overflow

[Theorem 8 - Liu and Layland]

A necessary and sufficient condition for the feasibility of the deadline driven scheduling algorithm w.r.t a processor with availability function $a_k(t)$ is

$$\text{floor}(t/T_{k+1}) \cdot C_{k+1} + \text{floor}(t/T_{k+2}) \cdot C_{k+2} + \dots + \text{floor}(t/T_m) \cdot C_m \leq a_k(t)$$

[Theorem 9 - Liu and Layland]

Summary

- Rate monotonic scheduling
- Deadline driven scheduling
- Mixed scheduling