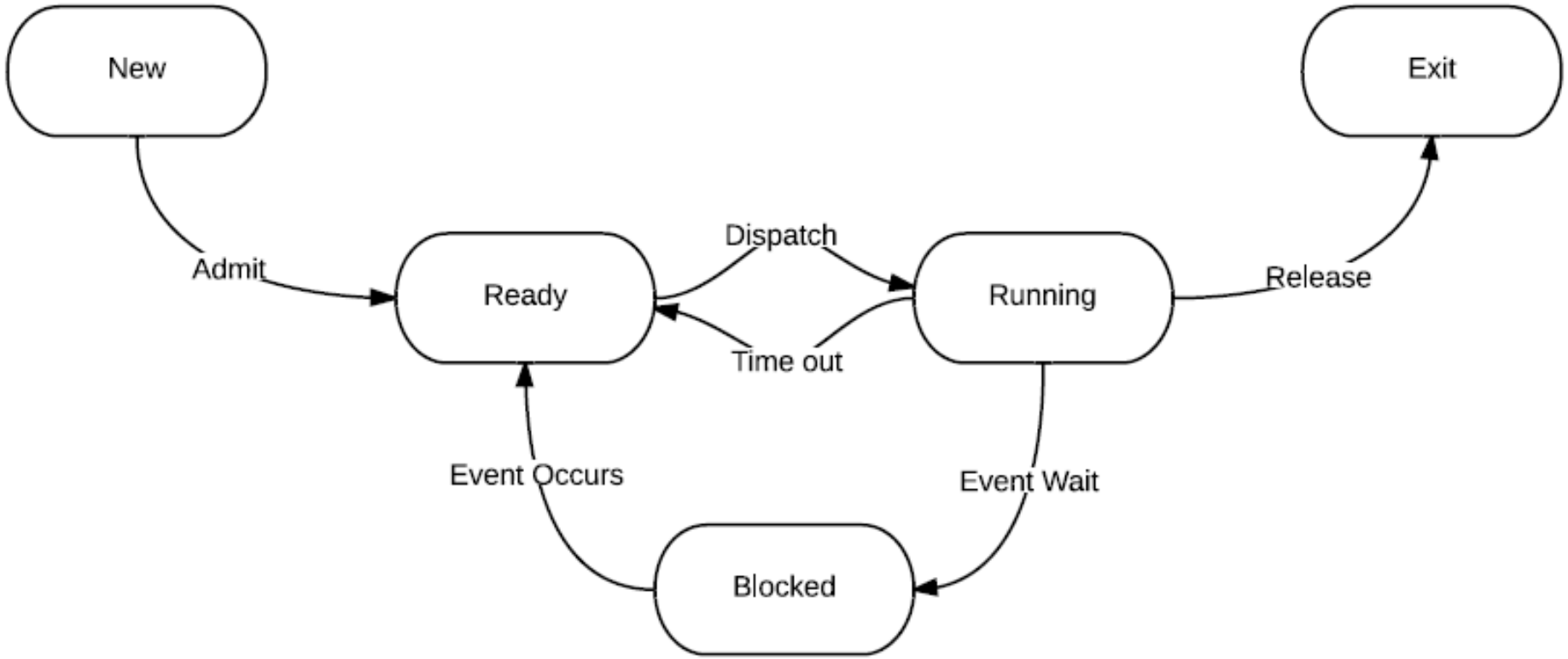
Scheduling

Ravi Suppiah
Lecturer, NUS SoC

State Transition Diagram



Scheduling Algorithm

NUS National University of Singapore

- A real-time scheduling algorithm consists of a set of rules (policy) for allocating tasks to the processor(s)
- The goal is to ensure that all the tasks meet their respective deadlines
- We will consider single processor and multiple tasks

ARM

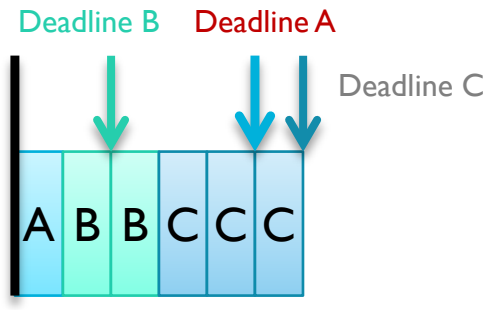
Example

- The system has three tasks:

Task	Execution Time	Deadline
A	10ms	50ms
B	20ms	20ms
C	30ms	60ms

- Scheduling Algorithm 1: A →B →C (no preemptions)
  - Worst-case behavior: All tasks ready at time 0

B will miss deadline



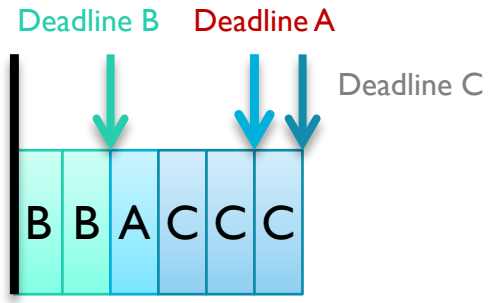
Example (contd.)

- The system has three tasks:

Task	Execution Time	Deadline
A	10ms	50ms
B	20ms	20ms
C	30ms	60ms

- Scheduling Algorithm 2: B →A →C (no preemptions)
  - Worst-case behavior: All tasks ready at time 0

All tasks will meet deadline

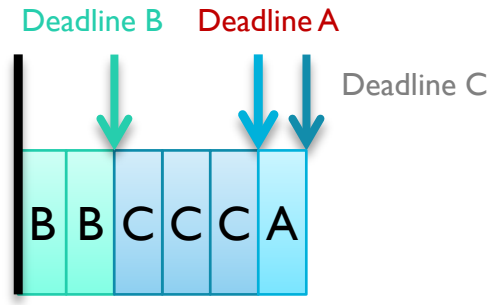


Example (contd.)

- The system has three tasks:

Task	Execution Time	Deadline
A	10ms	50ms
B	20ms	20ms
C	30ms	60ms

- Scheduling Algorithm 3: B → C →A (no preemptions)
  - Worst-case behavior: All tasks ready at time 0

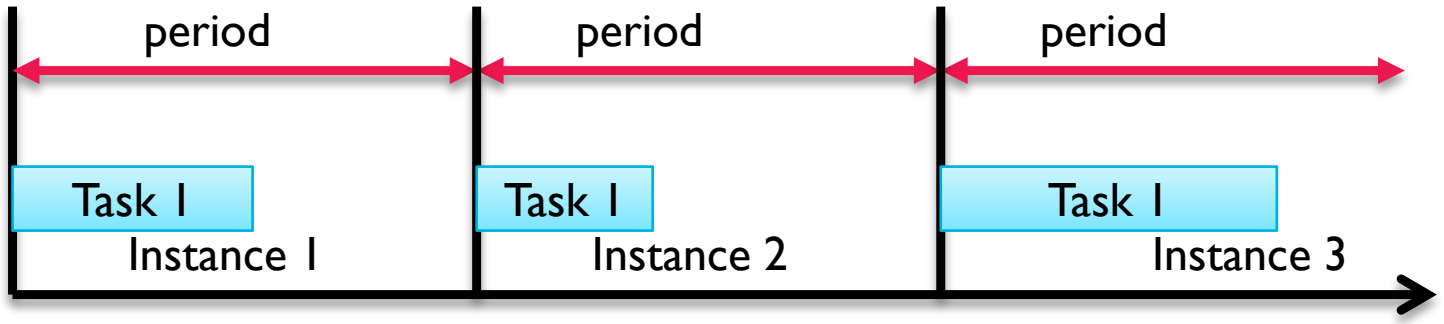


A will miss deadline

Basic Real-Time Task Model

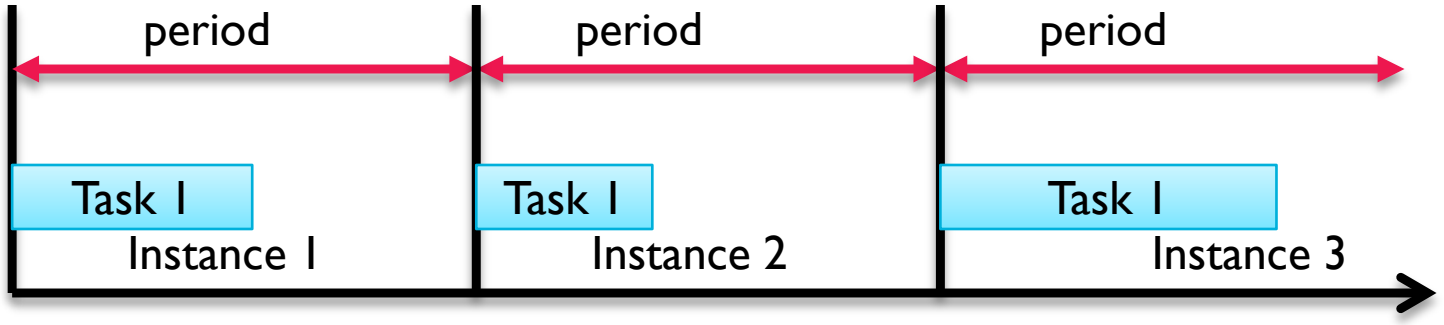
- Concurrent program consists of fixed number (N) of tasks

- Tasks are periodic with known periods (P)
  - Periods do not change with time
  - A task can be seen as an infinite sequence of **instances**
  - One task instance is **released** (becomes ready) at the beginning of each period



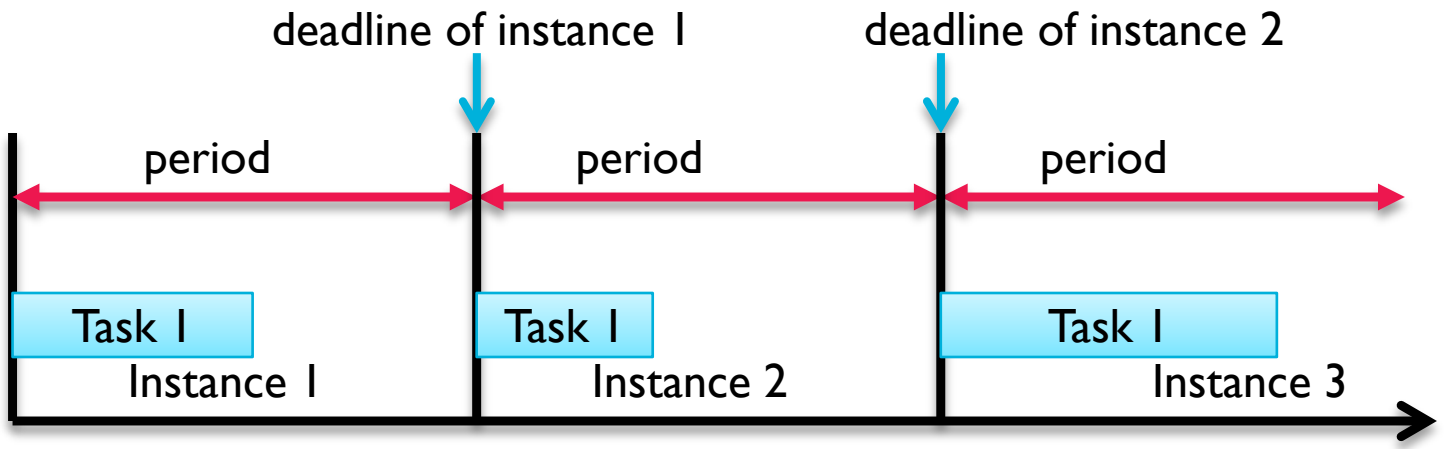
Basic Real-Time Task Model (contd.)

- Each task has a fixed worst-case execution time WCET (C) that can be computed offline
  - Each task instance may take a different amount of time to execute
  - WCET: Upper limit of execution time for any instance of a task
  - Example:
  - Task is a sorting program; different execution time depending on the input to be sorted for different instances
  - Execution time of task instances 10s, 20s, 30s; WCET of task is 30s



Basic Real-Time Task Model (contd.)

- Timing constraints are expressed as deadline (D)
  - Maximum time allowed between a task instance being released (beginning of period) and its completion time
- Clearly C ≤ D and D ≤ P; therefore C ≤ D ≤ P
- We assume deadline of each task is equal to period (D = P)
- All tasks have hard deadline, i.e., deadline must be satisfied for all task instances



Basic Real-Time Task Model (contd.)

NUS National University of Singapore

Additional Assumptions:

(1) Tasks are completely independent of each other

- Whether a task is ready or not does not depend on other tasks

(2) System overhead for scheduling and context switch time are negligible

ARM

WCET versus WCRT

NUS National University of Singapore

- WCET (C): Worst-case execution time of a task is the maximum time required to complete the execution of any instance of the task without any interference from other activities

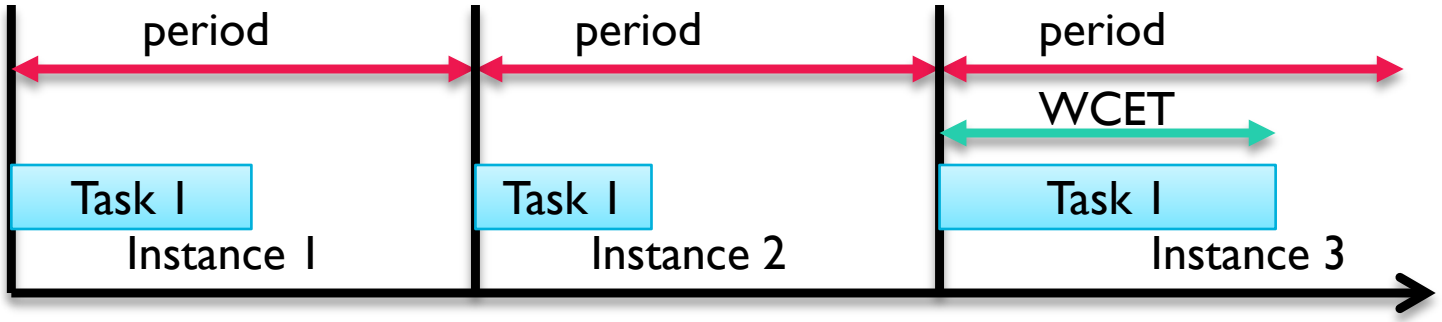
- WCRT (R): Worst-case response time of a task is the maximum elapsed time between the release of a task instance and its completion

- Clearly, $C \leq R$

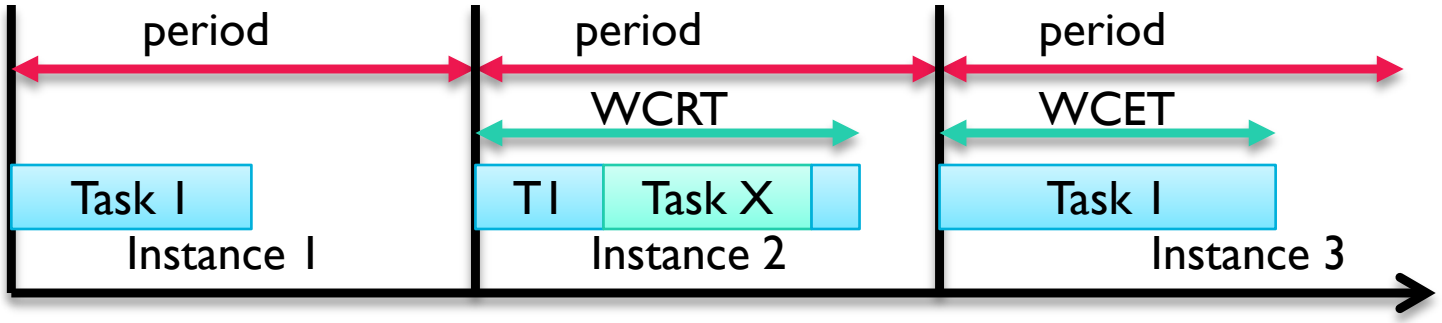
- To meet deadline, we need $R \leq D$

ARM

WCET versus WCRT (contd.)

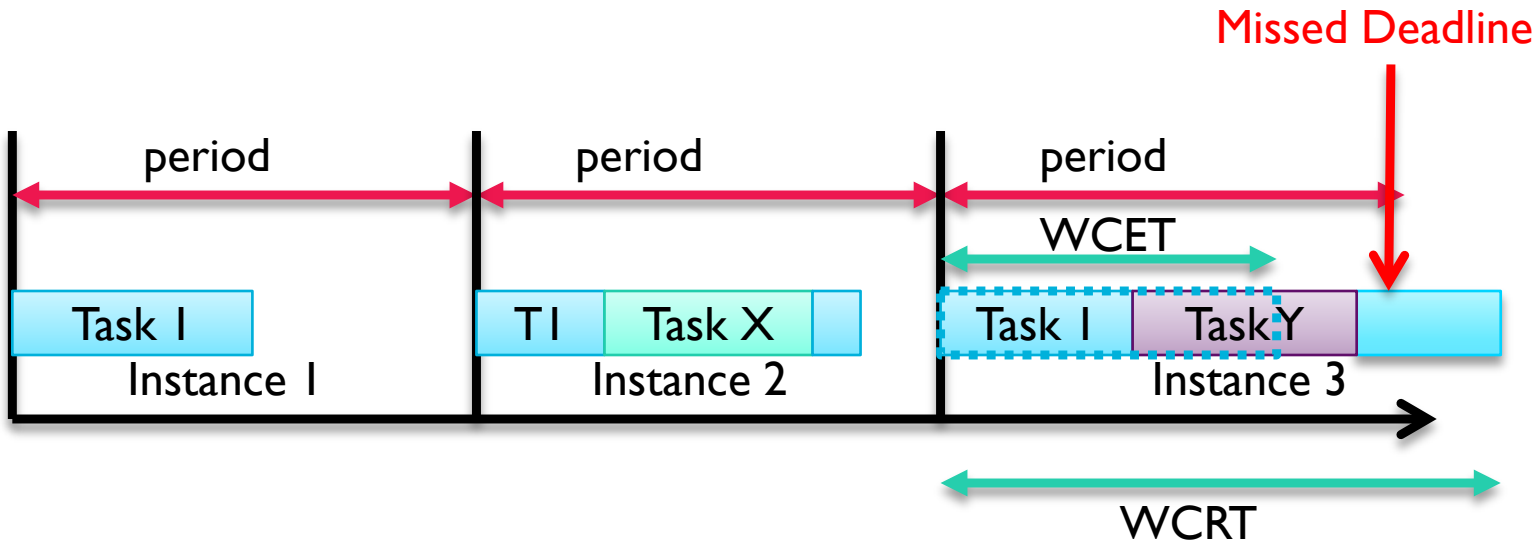


No interference



Interference from Task X for Task 1 (instance 2)

WCET versus WCRT (Contd.)

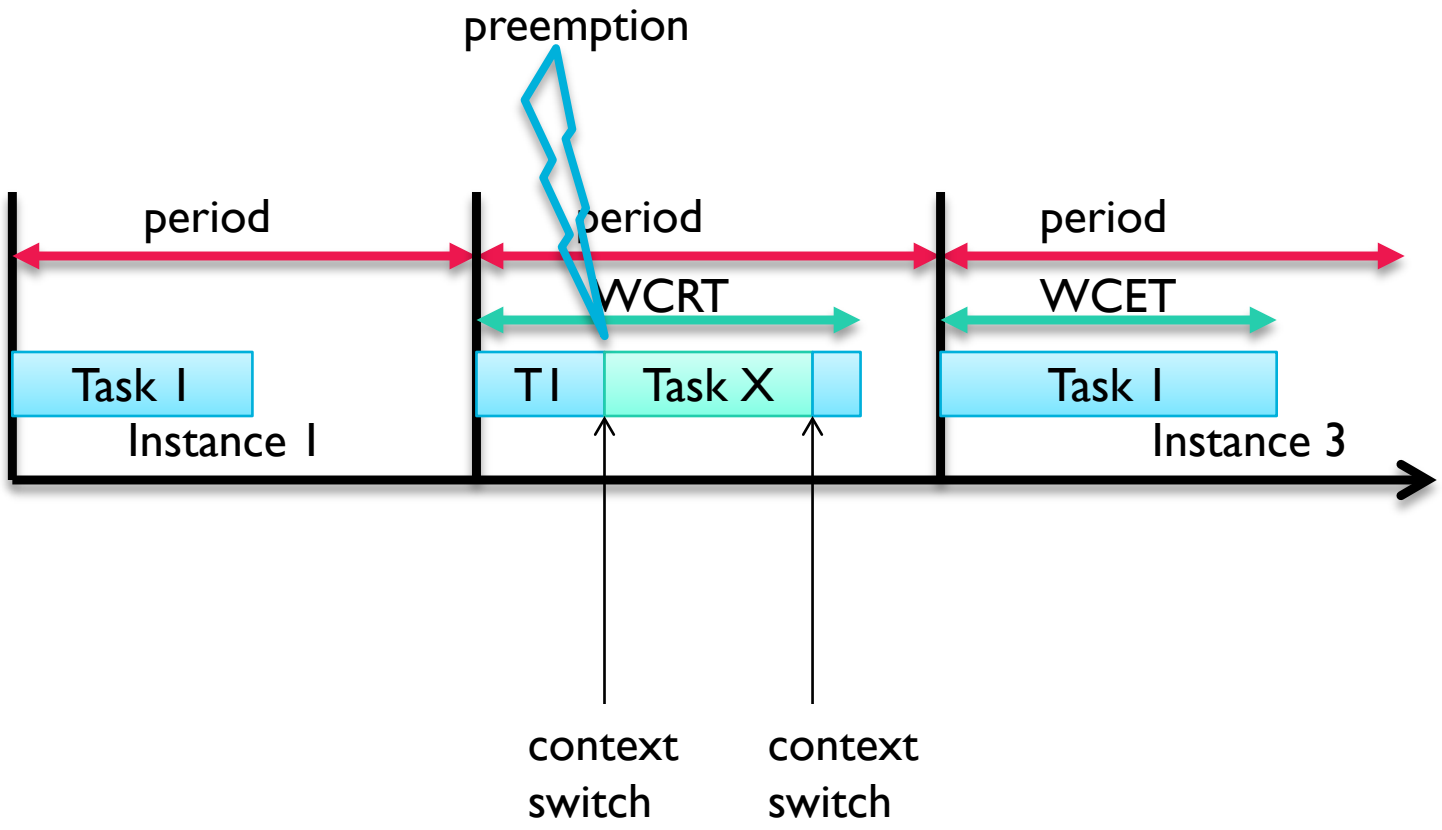


Interferences

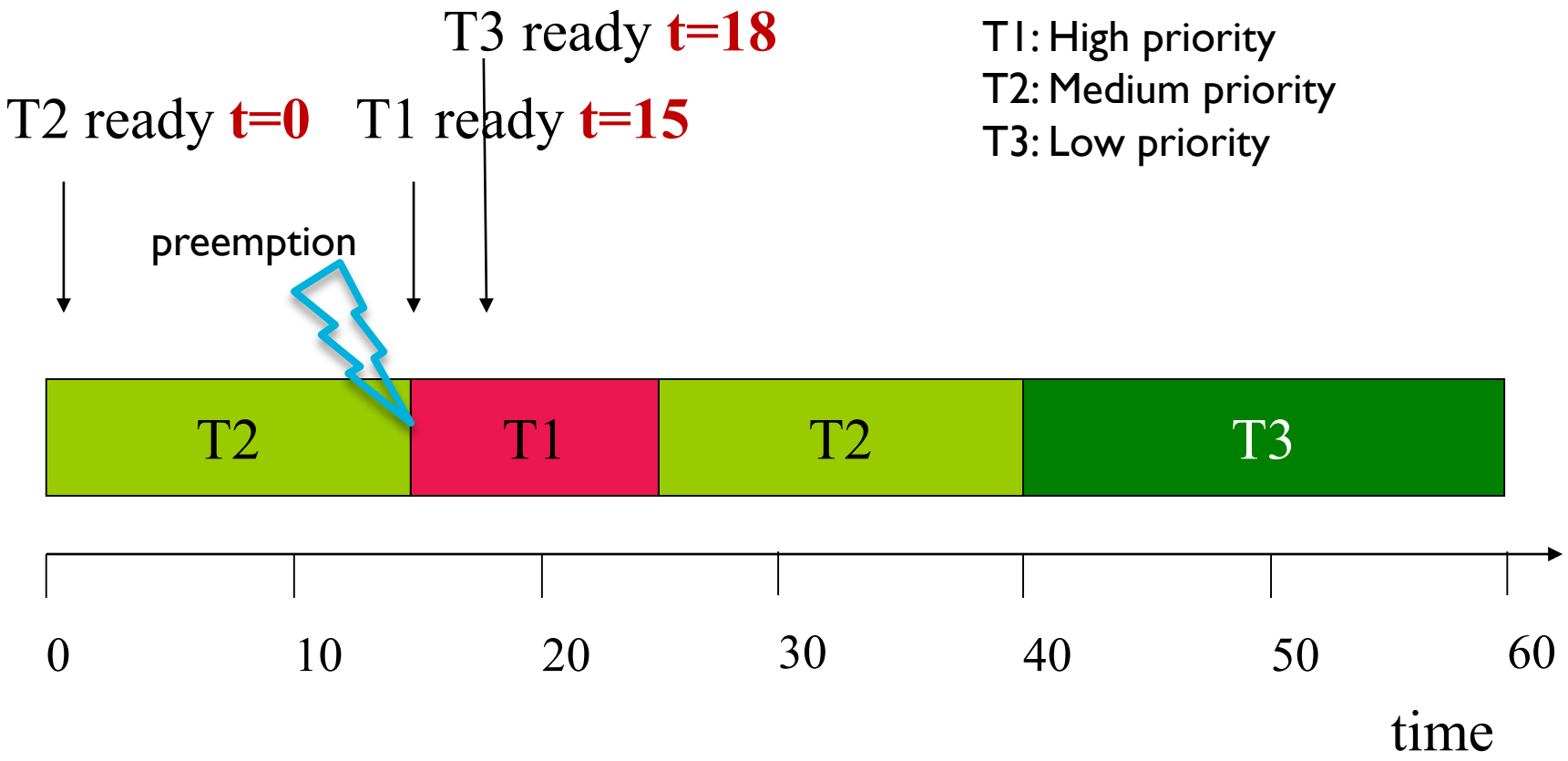
1. From Task X for Task I (instance 2)
2. From Task Y for Task I (instance 3)

Preemptive Priority-Driven Scheduling

- The policy in which a lower-priority task that is currently executing gets context switched out once a higher-priority task become ready



Preemptive Priority-Driven Scheduling



T1: High priority
T2: Medium priority
T3: Low priority

Real-Time Scheduling Policy

NUS National University of Singapore

- Priority-driven preemptive scheduling
- How do we assign priorities to the tasks?
  - **Fixed priority:** Priority does not change with time
  - **Dynamic priority**: Priority may change with time

ARM

Rate Monotonic Scheduling (RMS)

NUS National University of Singapore

- RMS (Liu and Layland 1973)
  - widely-used, analyzable scheduling policy
- Fixed-priority assignments in periodic tasks
- Priority is inversely proportional to task period
  - Shorter the task period, higher the priority
  - Does not depend on WCET
- Intuition: Processes requiring frequent attention (smaller period) should receive higher priority

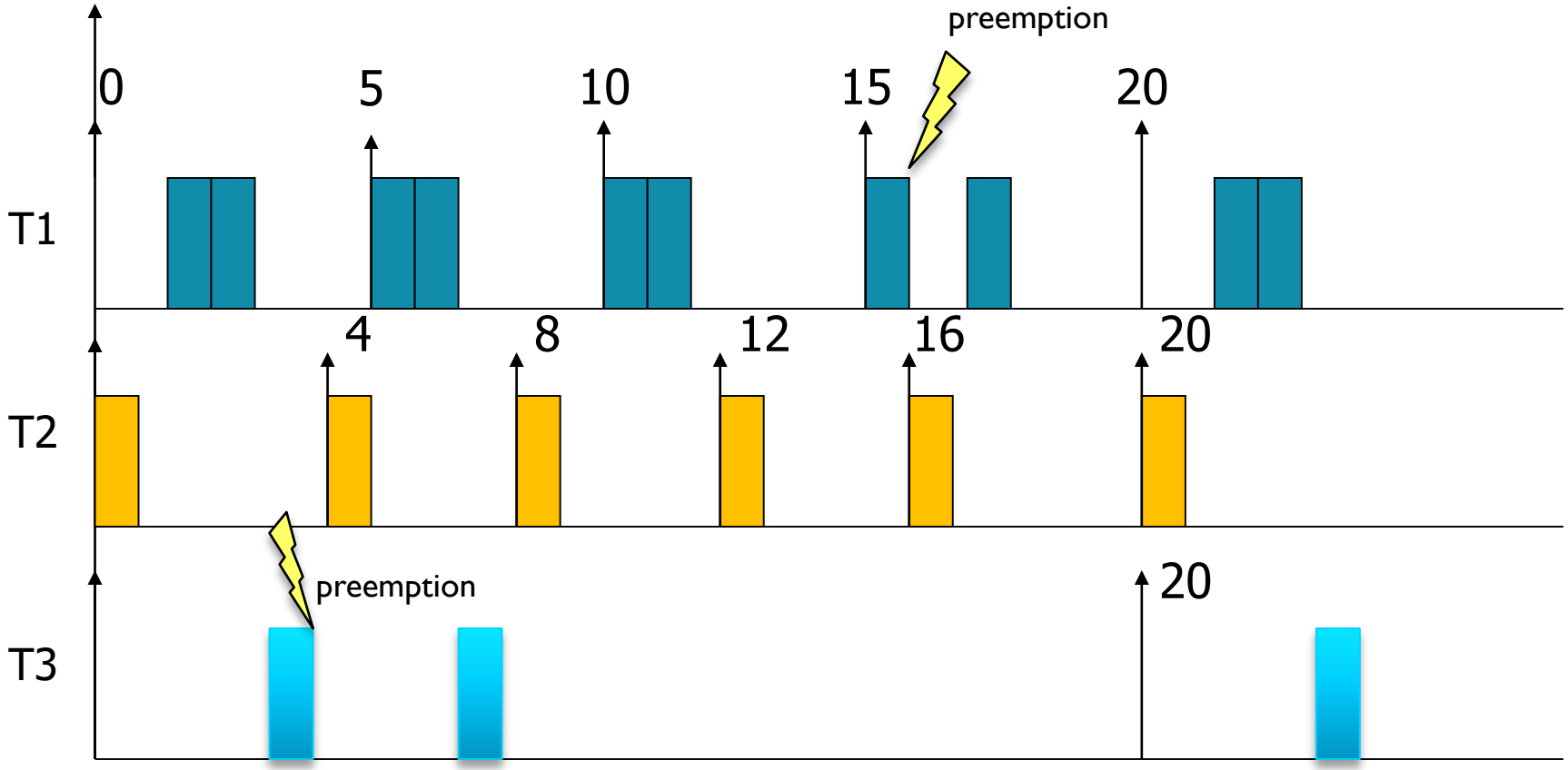
ARM

RMS Example

T1: c=2, p=d=5
T2: c=1, p=d=4
T3: c=2, p=d=20

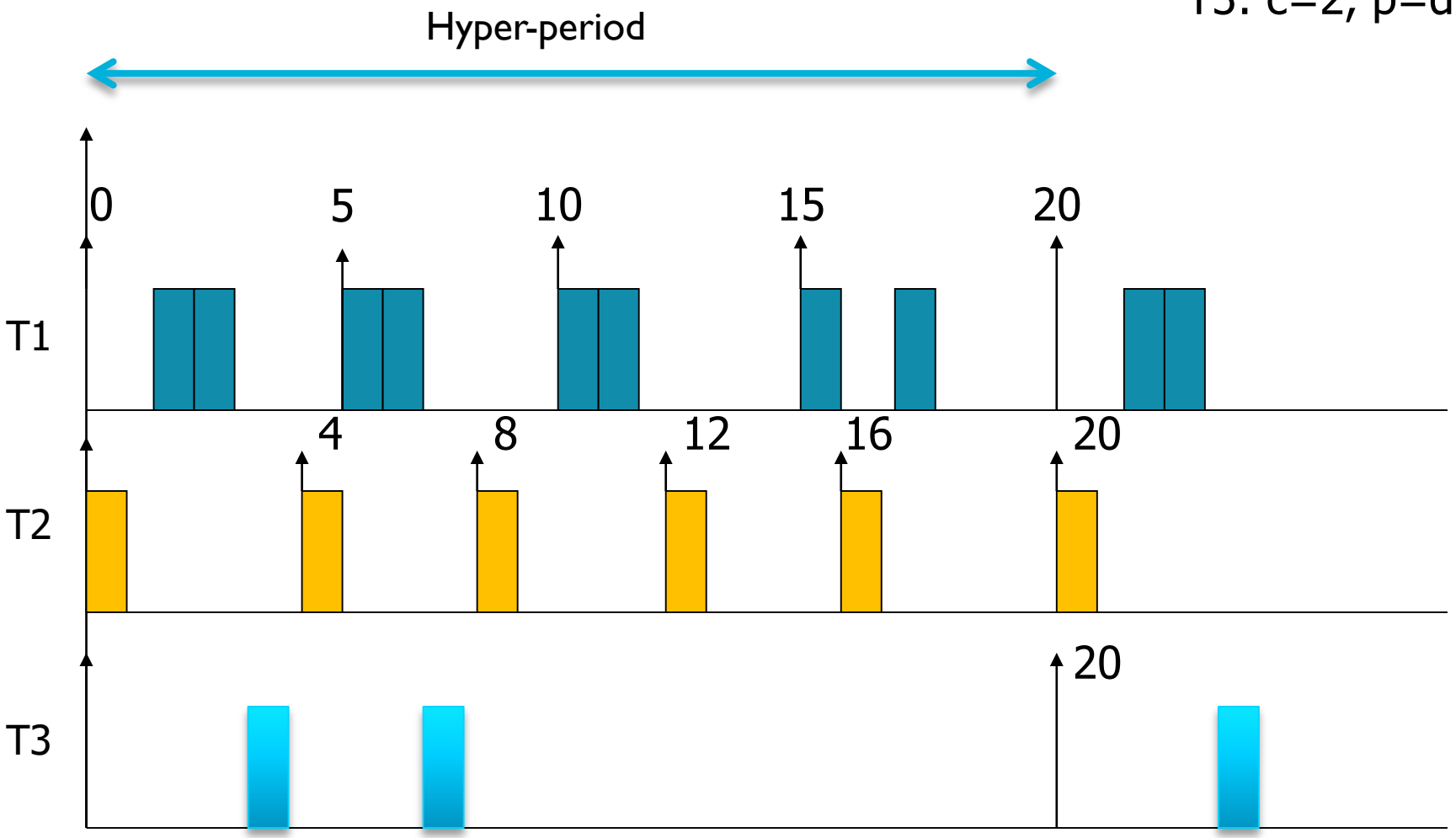
Priority of Tasks: T2 > T1 > T3

All Tasks are Ready at t=0



Hyper-Period

T1: c=2, p=d=5
T2: c=1, p=d=4
T3: c=2, p=d=20



Hyper-Period

- Least Common Multiple (LCM) of the task periods

- Example:
  - Task periods 4s, 5s, 10s
  - LCM(4, 5, 10) = 20
  - Hyper-period = LCM = 20s

- The schedule repeats itself every hyper-period

- If you simulate the behavior of the schedule over one hyper-period and no deadline is missed, then the task set is schedulable

Schedulability Analysis and Optimality

NUS National University of Singapore

- Feasible schedule: if the schedule can meet all the timing constraints

- Optimal scheduling algorithm: A scheduling algorithm that produces a feasible schedule for a task set if one exists

ARM

RMS: Optimal Fixed-Priority Scheduling

NUS National University of Singapore

- If a task set is schedulable by any arbitrary but fixed-priority assignment, then it is schedulable by RMS as well
  - Given a task set with N tasks, there exist different fixed-priority assignments
  - Number of possible priority assignments is equal to the number of permutations of N tasks N!

High	Medium	Low
T1	T2	T3
T1	T3	T2
T2	T1	T3
T2	T3	T1
T3	T1	T2
T3	T2	T1

ARM

RMS: Optimal Fixed-Priority Scheduling

NUS National University of Singapore

- Assume highlighted priority assignments are feasible
- Then priority assignment according to RMS will be either of the two highlighted priority assignments
- We do not need to exhaustively try out all the priority assignments because RMS is guaranteed to pick the right one

High	Medium	Low
T1	T2	T3
T1	T3	T2
T2	T1	T3
T2	T3	T1
T3	T1	T2
T3	T2	T1

ARM

Schedulability Analysis

NUS National University of Singapore

- Schedulability analysis is the method of
  - analyzing the system and predicting its worst-case behavior with respect to timings when a scheduling algorithm is applied
  - validating that the worst-case behavior meets the timing constraints imposed on the system at design time

ARM

Schedulability of a task set

- Given a task set, how do we know if the task set is schedulable?

- Utilization of a periodic task set is the fraction of processor time spent in the execution of the task set

$$U = \sum_i \frac{c_i}{p_i}$$

- Example:
  - U = 2/5 + 1/4 + 2/20 = 0.75

Task	WCET	Period
T1	2	5
T2	1	4
T3	2	20

Necessary but not sufficient condition

NUS National University of Singapore

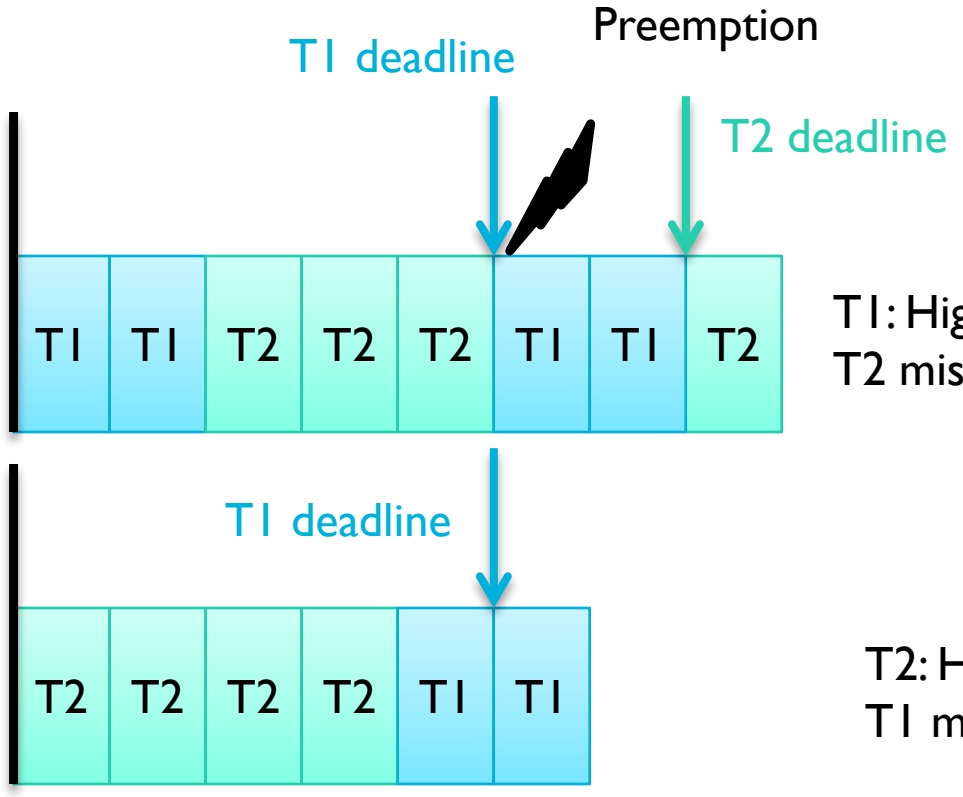
- $U \leq 1$ is a necessary condition for any task set to have a feasible schedule
  - Necessary condition means that the condition **must be satisfied** for a feasible schedule to exist
  - If $U \leq 1$ condition is not satisfied, the task set cannot possibly have a feasible schedule no matter what
- $U \leq 1$ is NOT a sufficient condition for a task set to have a feasible schedule under fixed priority
  - Sufficient condition means that if the condition is satisfied, then a feasible schedule **must exist**
  - Even if $U \leq 1$ condition is satisfied, a task set still may not have a feasible schedule with fixed priority

ARM

Schedulability of a task set under fixed priority

- A task set may or may not be schedulable under fixed priority even if U≤ 1
- Example:
  - $U = \frac{34}{35} \leq 1$

Task	WCET	Period
T1	2	5
T2	4	7



T1: High priority
T2 missed deadline

T2: High priority
T1 missed deadline

Schedulability of a task set under fixed priority

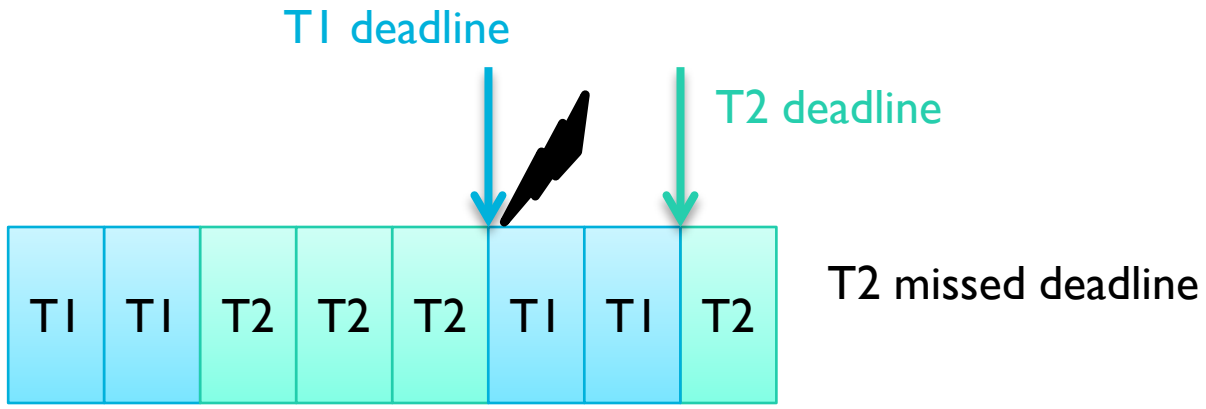
NUS National University of Singapore

- A task set may or may not be schedulable under fixed priority even if U≤ 1
- Use the optimality of RMS to check the schedulability of a task set under fixed priority
  - Step 1: Assign priorities according to RMS
  - Step 2: Simulate the schedule for hyper-period
  - Step 3: If no deadline is missed during hyper-period, task set is schedulable; otherwise task set is not schedulable with fixed priority

ARM

Example

- T1: c1 = 2 p1 = 5 : High Priority according to RMS
- T2: c2 = 4 p2 = 7 : Low Priority according to RMS
- RMS schedule is shown below
- As T2 missed deadline within hyper-period, task set is not schedulable using fixed priority



Utilization Bound: Sufficient but not necessary condition

- Checking the schedule for the entire hyper-period may be computationally expensive for large hyper-period
- Sufficient Condition (Utilization Bound): A task set with n tasks is schedulable with fixed priorities if

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

- The utilization bound is NOT a necessary condition
- A task set may still be schedulable even if the utilization bound is not satisfied

Utilization Bound Values: Sufficient but not necessary condition

- Sufficient Condition (Utilization Bound): A task set with n tasks is schedulable with fixed priorities if

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

B(1)=1.0	B(4)=0.756	B(7)=0.728
B(2)=0.828	B(5)=0.743	B(8)=0.724
B(3)=0.779	B(6)=0.734	U(∞)=0.693

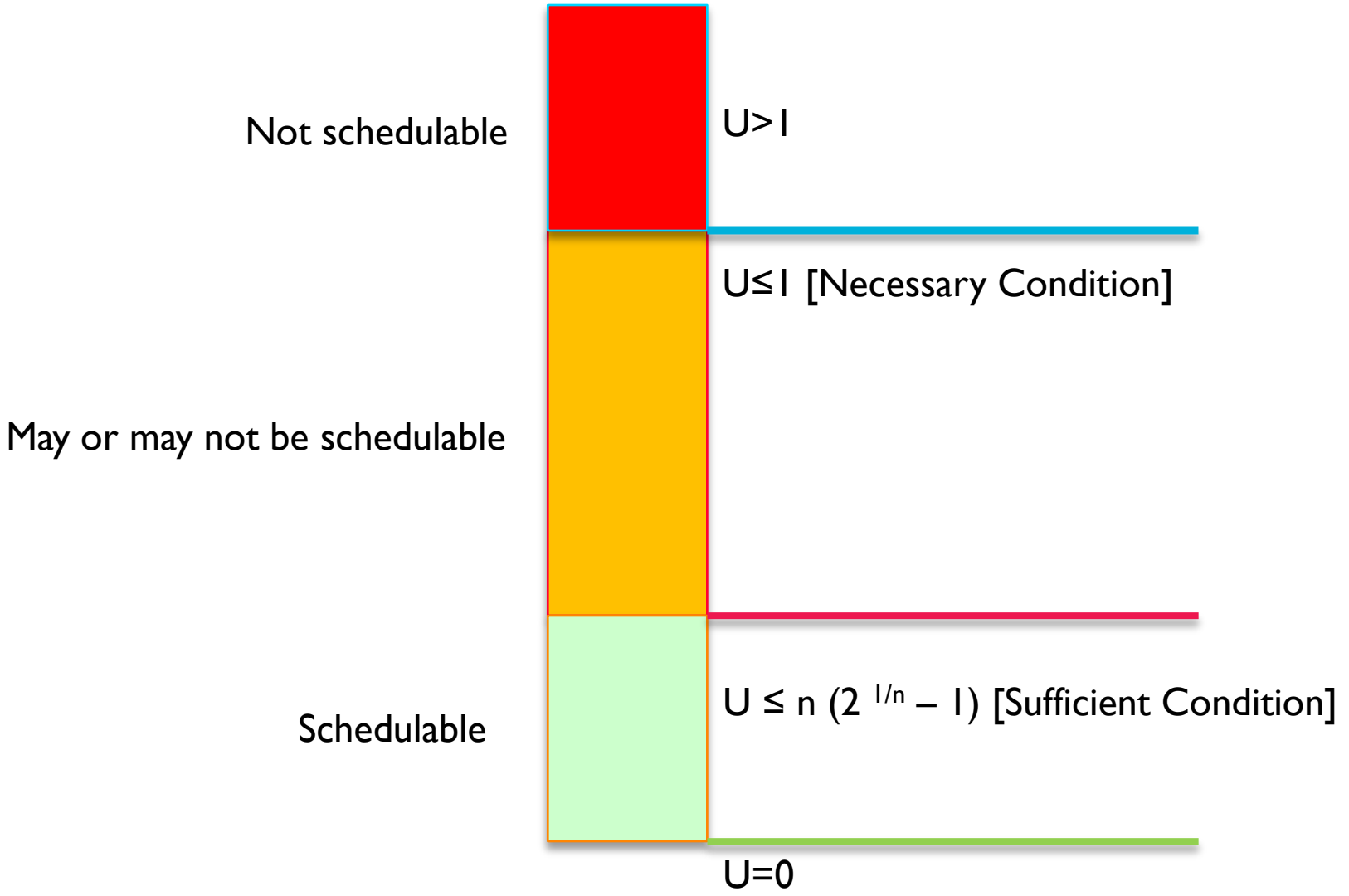
Example

- $U \leq n\,(2^{1/n} - 1)$ is not necessary

B(1)=1.0	B(4)=0.756	B(7)=0.728
B(2)=0.828	B(5)=0.743	B(8)=0.724
B(3)=0.779	B(6)=0.734	U(∞)=0.693

- T1: c1 = 1 P1 = 2
- T2: c2 = 1 P2 = 3
- U = 0.8333> 0.828
- But the task set is still RMS schedulable

Schedulability Analysis: What we have so far



Rate Monotonic Analysis (RMA)

- Schedulability analysis for task sets that meet the necessary condition but not the sufficient condition

Utilization bound for an example

- T1: c1 = 40 p1 = 100 [Priority High]
- T2: c2 = 40 p2 = 150 [Priority Medium]
- T3: c3 = 100 p3 = 350 [Priority Low]

Recall B(1) = 1.0, B(2) = 0.828, B(3) = 0.779

U(T1,T2,T3) = 0.952 > B(3) Does not meet sufficient condition

U(T1) = 40/100 = 0.4 ≤ B(1) * start with highest priority *
pass utilization bound test;T1 is schedulable

U(T1,T2) = 40/100 + 40/150 = 0.667 ≤ B(2) * next highest priority *
pass utilization bound test;T2 is schedulable

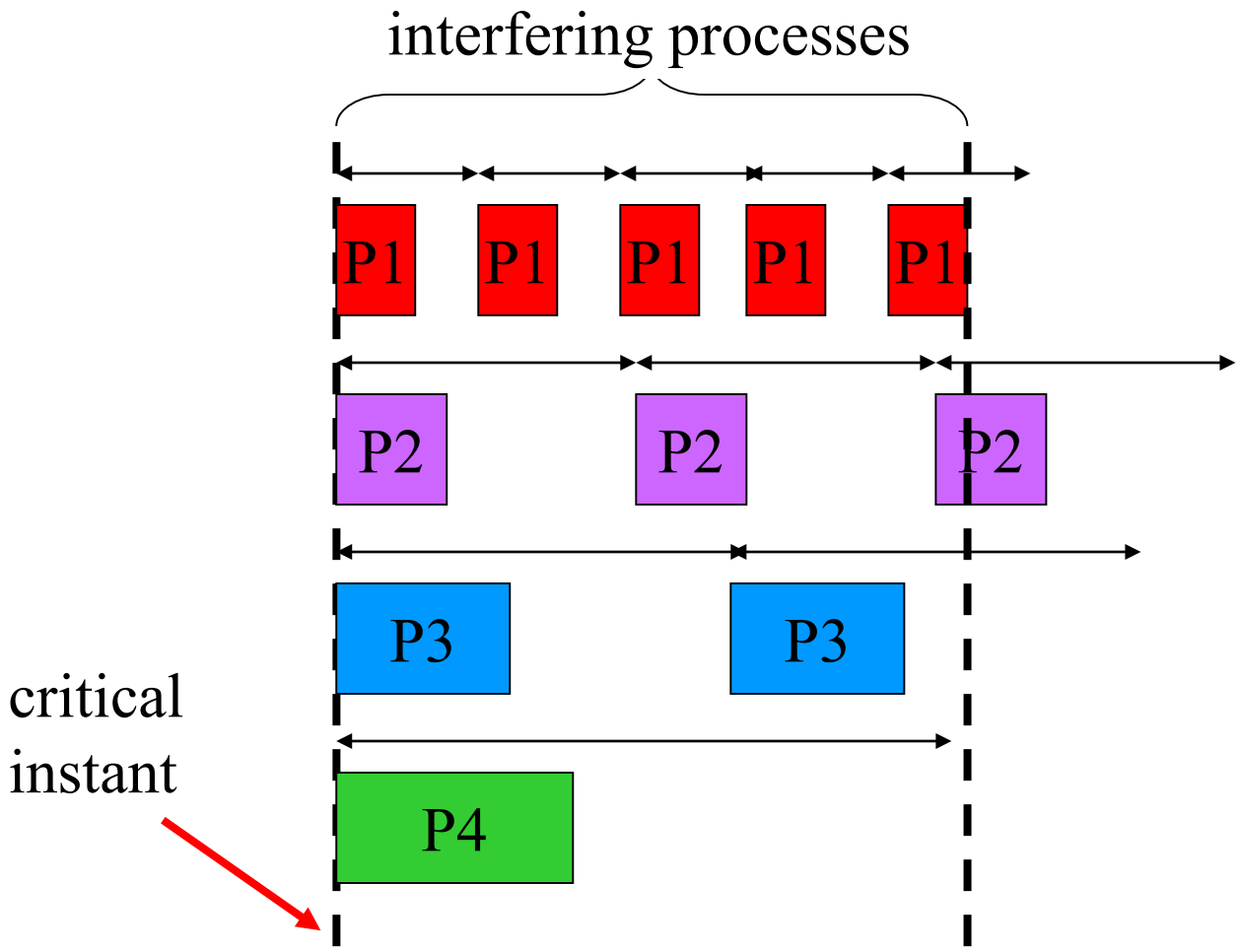
U(T1,T2,T3) = 0.952 > B(3) * next highest priority *
fail utilization test but may still be schedulable

Critical Instant

- Critical Instant: The instant at which the release of the task will produce the largest response time

- Theorem: A critical instance for any task occurs whenever it is release simultaneously with the release of all higher-priority tasks

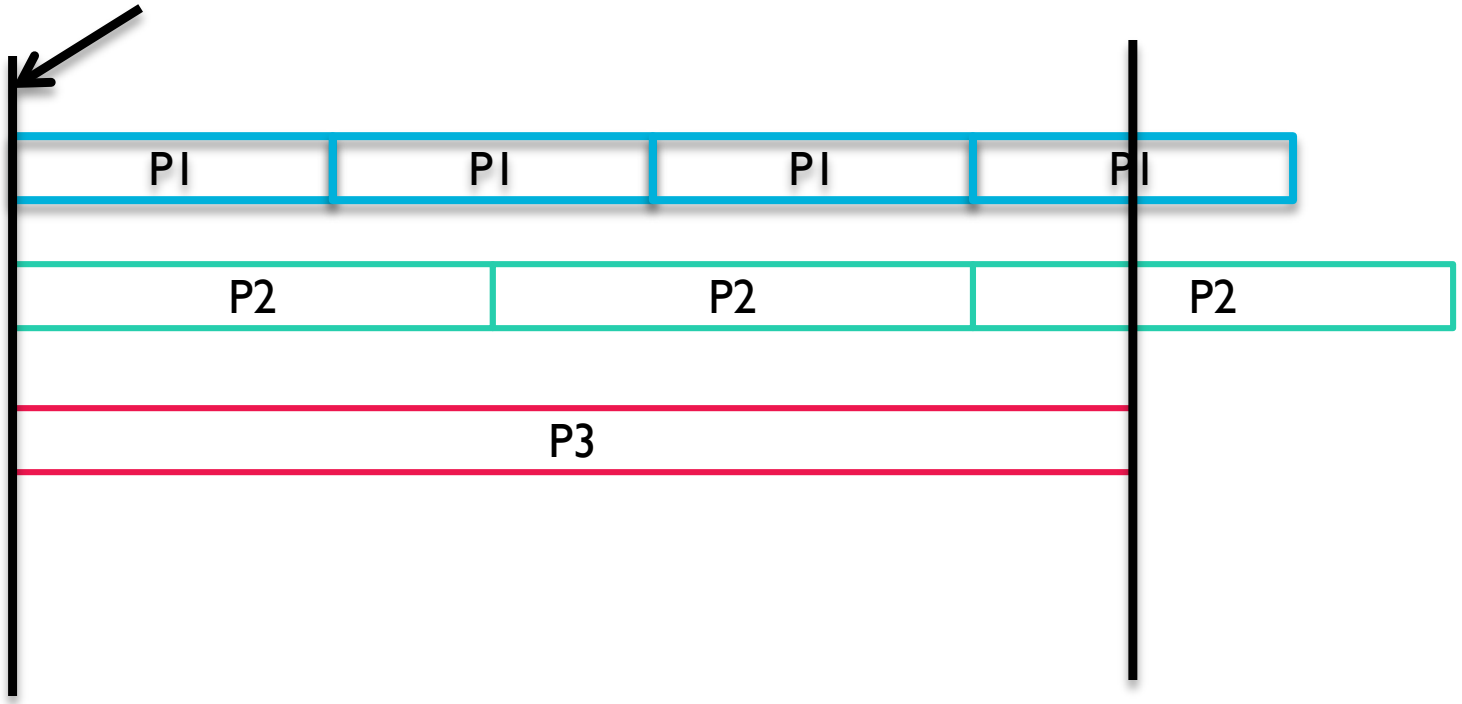
Critical instant



The same example

- T1: c1 = 40 p1 = 100 [Priority High]
- T2: c2 = 40 p2 = 150 [Priority Medium]
- T3: c3 = 100 p3 = 350 [Priority Low]

Critical Instant for T3



Critical instance analysis

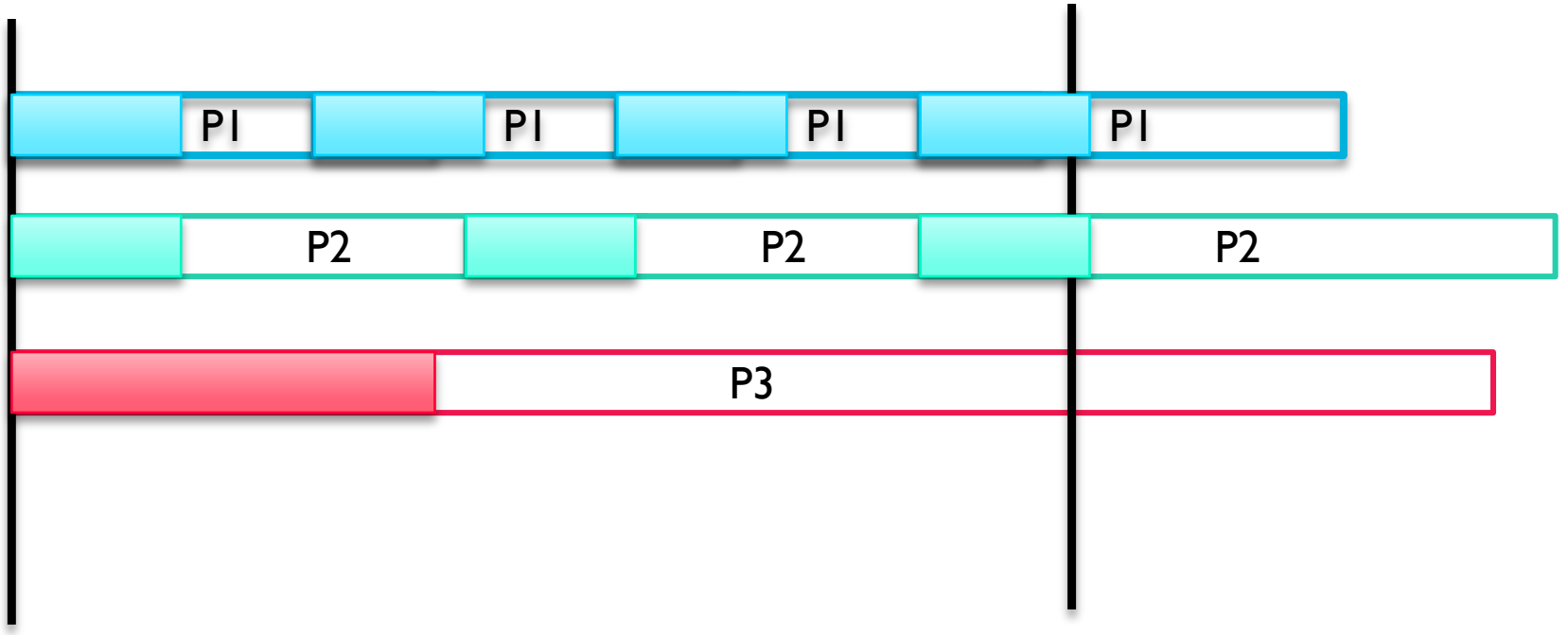
NUS National University of Singapore

- Worst-case response time (WCRT) R of a task T at its critical instance must be less than its deadline D (= P)
- Consider all tasks that are higher priority of task T
- For each higher priority task, find out the maximum number of its task instances that may release (interfere) within the deadline D of T
- Sum up computational demands from all higher priority tasks; this is the interference (I)
- For T to be schedulable ($R = C + I \leq D$)

ARM

Critical Instance Analysis for the example

- T1: c1 = 40 p1 = 100 [Priority High]
- T2: c2 = 40 p2 = 150 [Priority Medium]
- T3: c3 = 100 p3 = 350 [Priority Low]



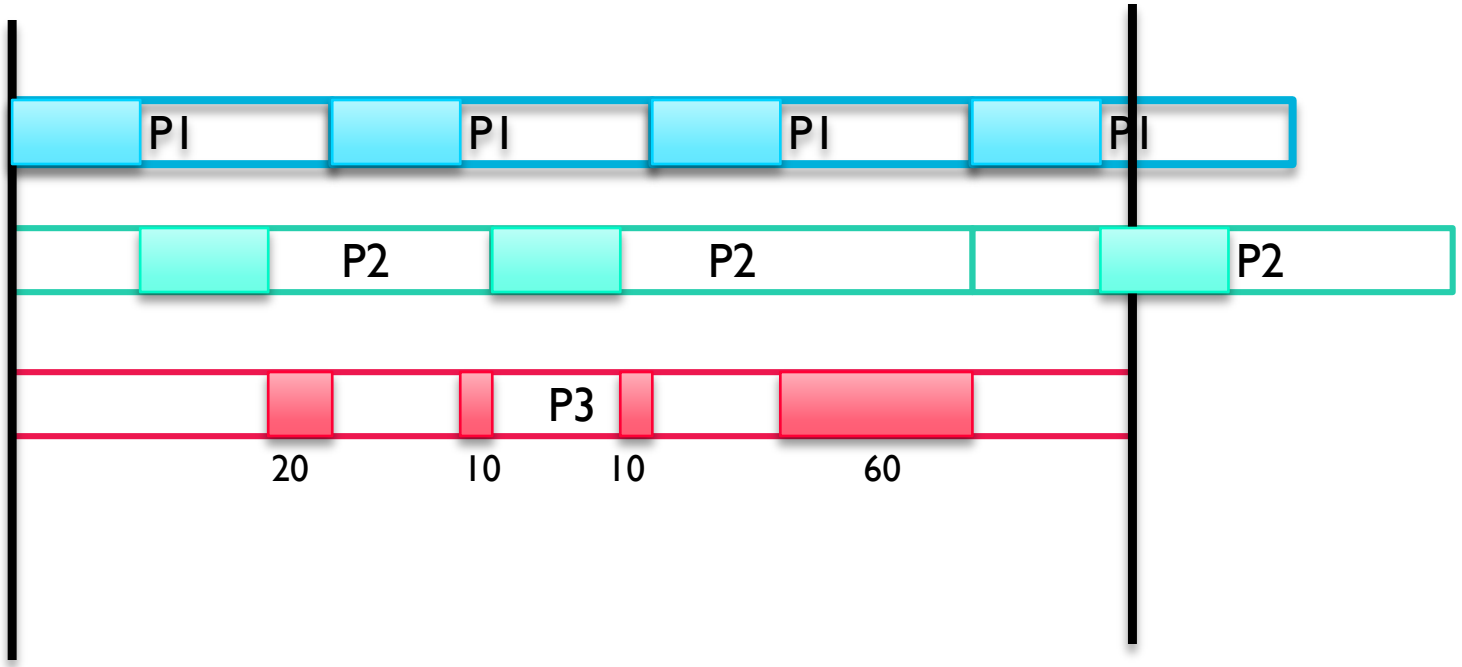
Critical Instance Analysis

- We are interested in response time computation of T3 at its critical instance
- T1 and T2 are higher priority than T3
- T1 (P1 = 100) can release 4 instances within T3's deadline (P3 = 350)
- T2 (P2 = 150) can release 3 instances within T3's deadline
- Total interference due to T1 and T2 = I = 40 x 4 + 40 x 3 = 280
- WCRT R3 of T3 = I + C3 = 280 + 100 = 380 > P3
- So T3 is not schedulable?

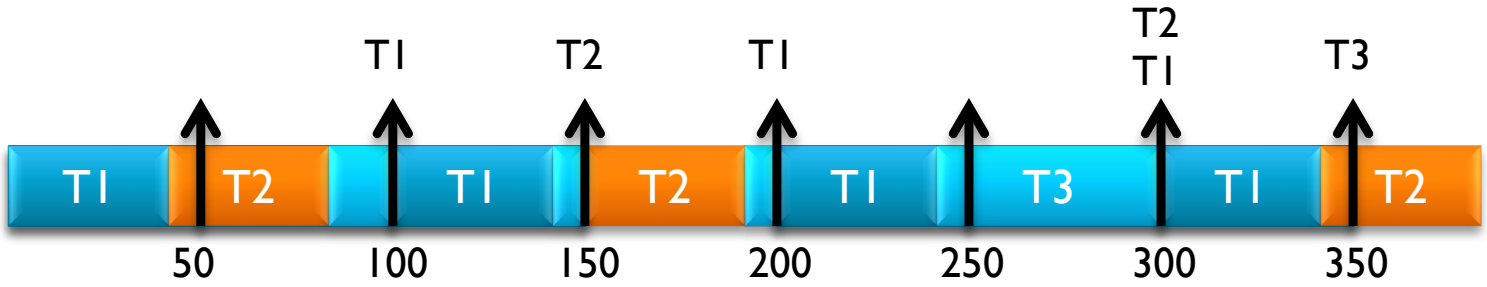
Task	WCET	Period
T1	40	100
T2	40	150
T3	100	350

Valid Schedule for T3

- T1: c1 = 40 p1 = 100 [Priority High]
- T2: c2 = 40 p2 = 150 [Priority Medium]
- T3: c3 = 100 p3 = 350 [Priority Low]



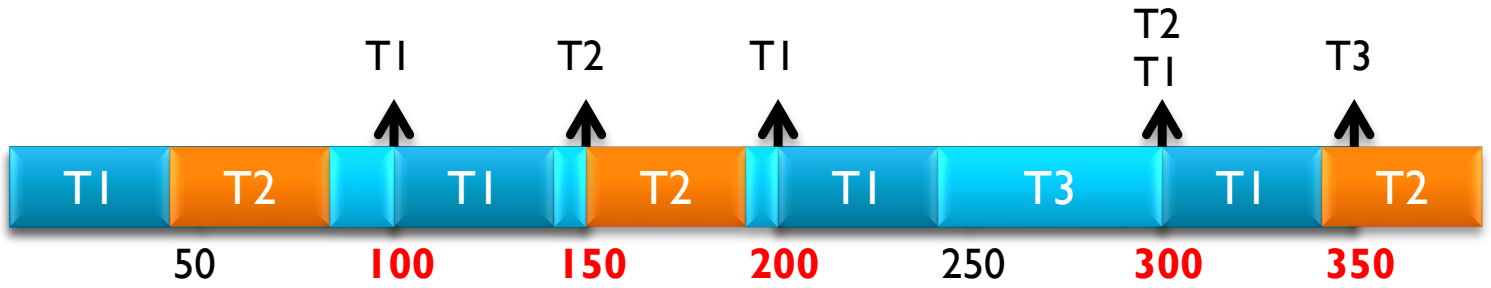
Problem with WCRT computation



Task	WCET	Period
T1	40	100
T2	40	150
T3	100	350

- T3 actually managed to complete execution by 300 cycle --- much before its deadline of 350
- But as we only check at 350, we need to introduce one additional task instance of both T1 and T2 making it look like we will miss deadline for T3
- To solve this issue, we need to check at more points

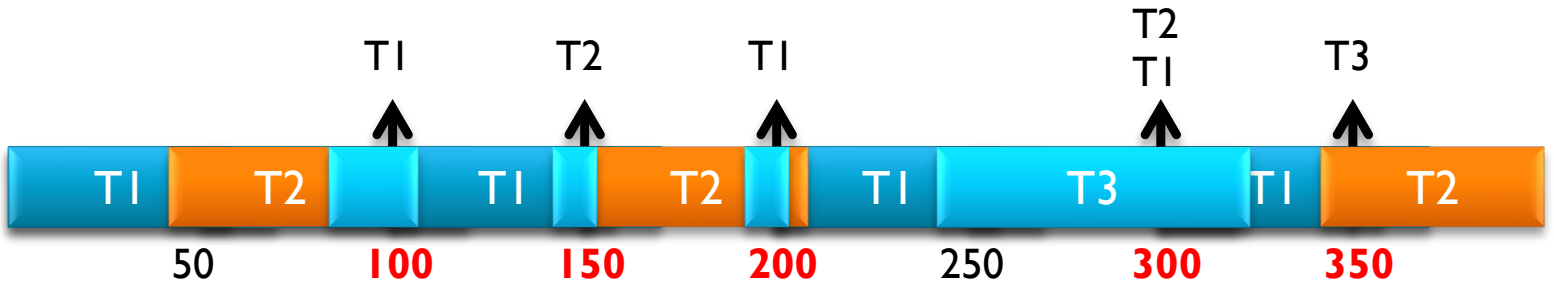
Which time instances to check?



- We need to check at 100, 150, 200, 300, 350
- That is, just before any new task instance is released, we check if total demand ≤ capacity

Task	WCET	Period
T1	40	100
T2	40	150
T3	100	350

Which time instances to check?



- t=100: demand = 40 + 40 + 100 = 180 > 100
- t =150: demand = 2x40 + 40 + 100 = 220 > 150
- t =200: demand = 2x40 + 2x40 + 100 = 260 > 200
- t =300: demand = 3x40 + 2x40 + 100 = 300 ≤ 300

So T3 is schedulable at t=300

This is exactly what the next schedulability test does

Task	WCET	Period
T1	40	100
T2	40	150
T3	100	350

Necessary and Sufficient condition

$$w_i(t) = \sum_{k=1}^{i} c_k \times \left\lceil \frac{t}{p_k} \right\rceil$$

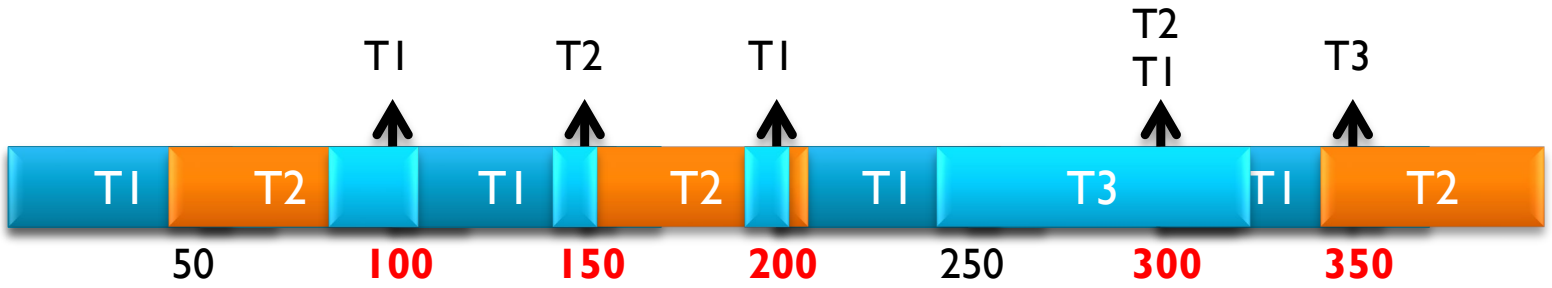
- Tasks $T_1, \ldots T_{(i-1)}$ be higher priority than task T_i in priority order
- The inequality $w_i(t) \leq t$ holds for any (one or more) time instant t where:

$$t = kp_j, \quad j = 1, \ldots, i, \quad k = 1, \ldots. \quad \left\lfloor \frac{p_i}{p_j} \right\rfloor$$

if and only if task T_i is RM-schedulable.

- Essentially we check inequality at release time of all higher priority task instances within the deadline of task T_i

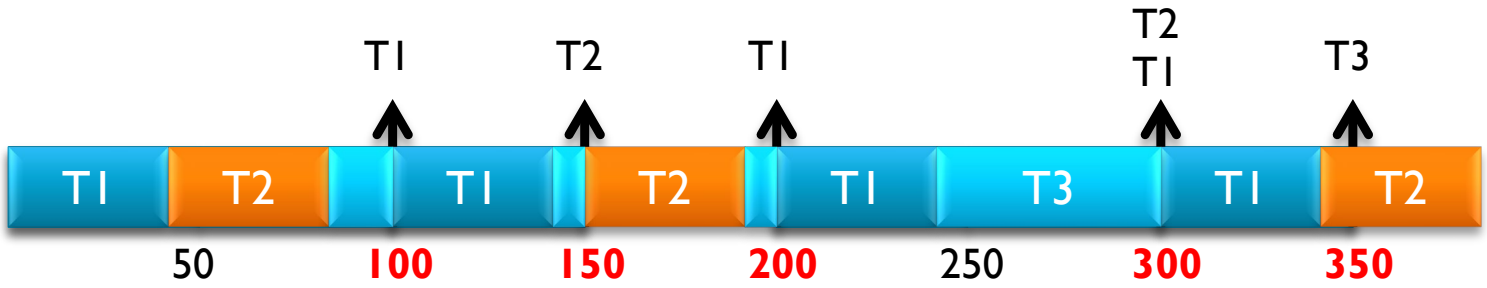
Application of the Condition



- i = 3: Higher priority tasks T_1 and T_2
- Time instances to consider: $t = kp_j$, $j = 1, \ldots, i$, $k = 1, \ldots \left\lfloor \frac{p_i}{p_j} \right\rfloor$
  - j = 1; $\left\lfloor \frac{p_i}{p_j} \right\rfloor = \left\lfloor \frac{350}{100} \right\rfloor = 3$ Thus k = 1, 2, 3 and t = 100, 200, 300
  - j = 2; $\left\lfloor \frac{p_i}{p_j} \right\rfloor = \left\lfloor \frac{350}{150} \right\rfloor = 2$ Thus k = 1, 2 and t = 150, 300
  - j = 3; $\left\lfloor \frac{p_i}{p_j} \right\rfloor = \left\lfloor \frac{350}{350} \right\rfloor = 1$ Thus k = 1 and t = 350

Task	WCET	Period
T1	40	100
T2	40	150
T3	100	350

Application of the Condition



- Consider t = 300

$$w_i(t) = \sum_{k=1}^{i} c_k \times \left\lceil \frac{t}{p_k} \right\rceil$$

$w_3(300) = 40 \times \left\lceil \frac{300}{100} \right\rceil + 40 \times \left\lceil \frac{300}{150} \right\rceil + 100 \times \left\lceil \frac{300}{350} \right\rceil$

$= 40 \times 3 + 40 \times 2 + 100 = 300$

$w_3(300) <= 300$

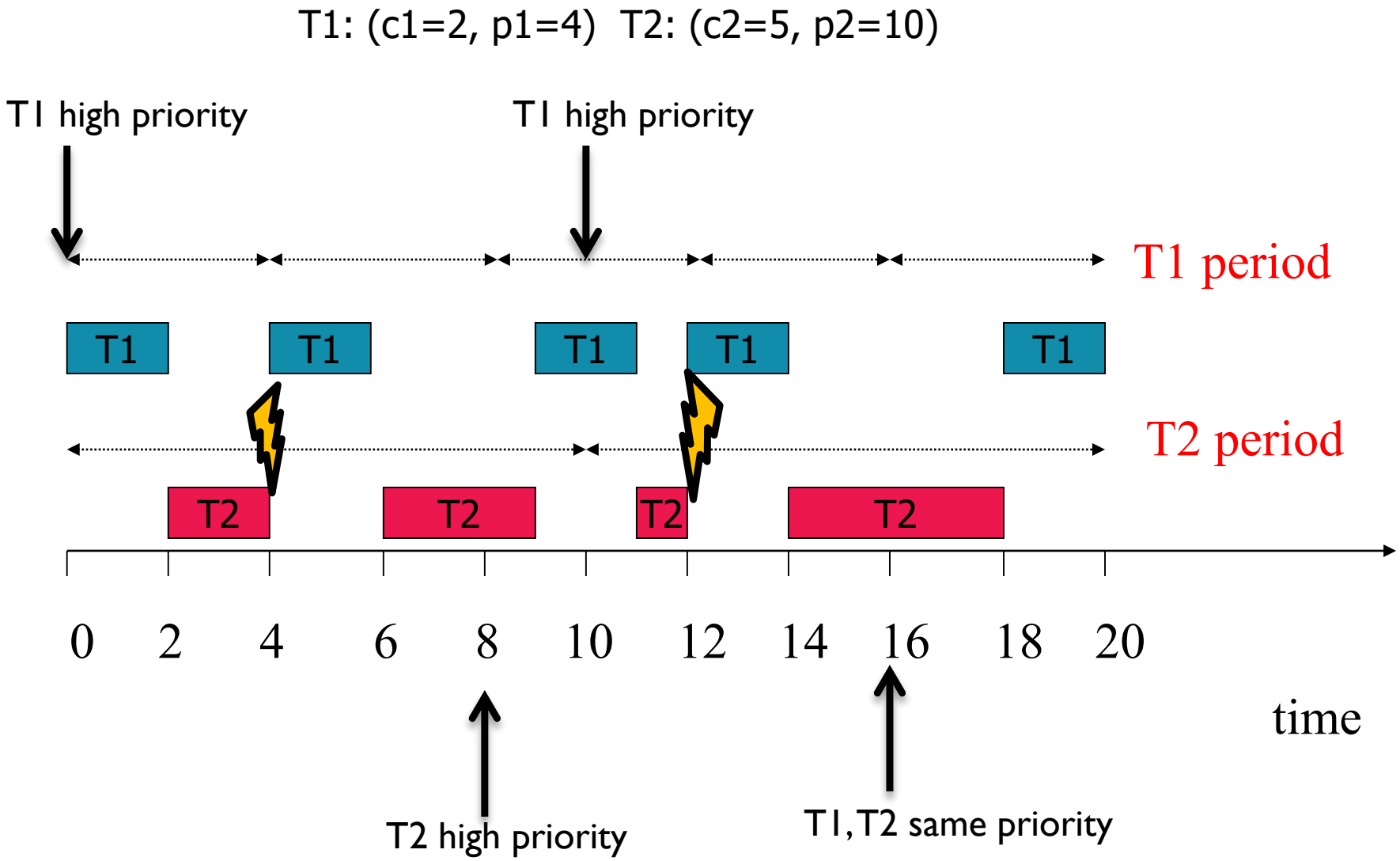
Earliest Deadline First (EDF) Scheduling

NUS National University of Singapore

- Dynamic priority scheduling scheme
- At any instant, the task with the earliest deadline has the highest priority
- Priority of a task depends on the current deadline of the active task instances
- Task priority may be updated only when any new task instance is released

ARM

EDF example


T1: (c1=2, p1=4) T2: (c2=5, p2=10)
T1 high priority
T1 high priority
T1 period
T1
T1
T1
T1
T1
T2 period
T2
T2
T2
T2
0
2
4
6
8
10
12
14
16
18
20
time
T2 high priority
T1, T2 same priority

EDF Optimality

- EDF is an optimal scheduling policy
  - If a feasible schedule exists using dynamic priorities, then EDF will produce a feasible schedule
- EDF can always produce a feasible schedule if $U \leq 1$
- Scheduling with dynamic priority is feasible if and only if $U \leq 1$ (necessary and sufficient)

The End!

NUS National University of Singapore

- In Summary
  - Real-time scheduling policy is essential for any real-time system with multi-tasking
  - Static priority scheduling policy: RMS
  - Dynamic priority scheduling policy: EDF

ARM