Periodic Task Scheduling

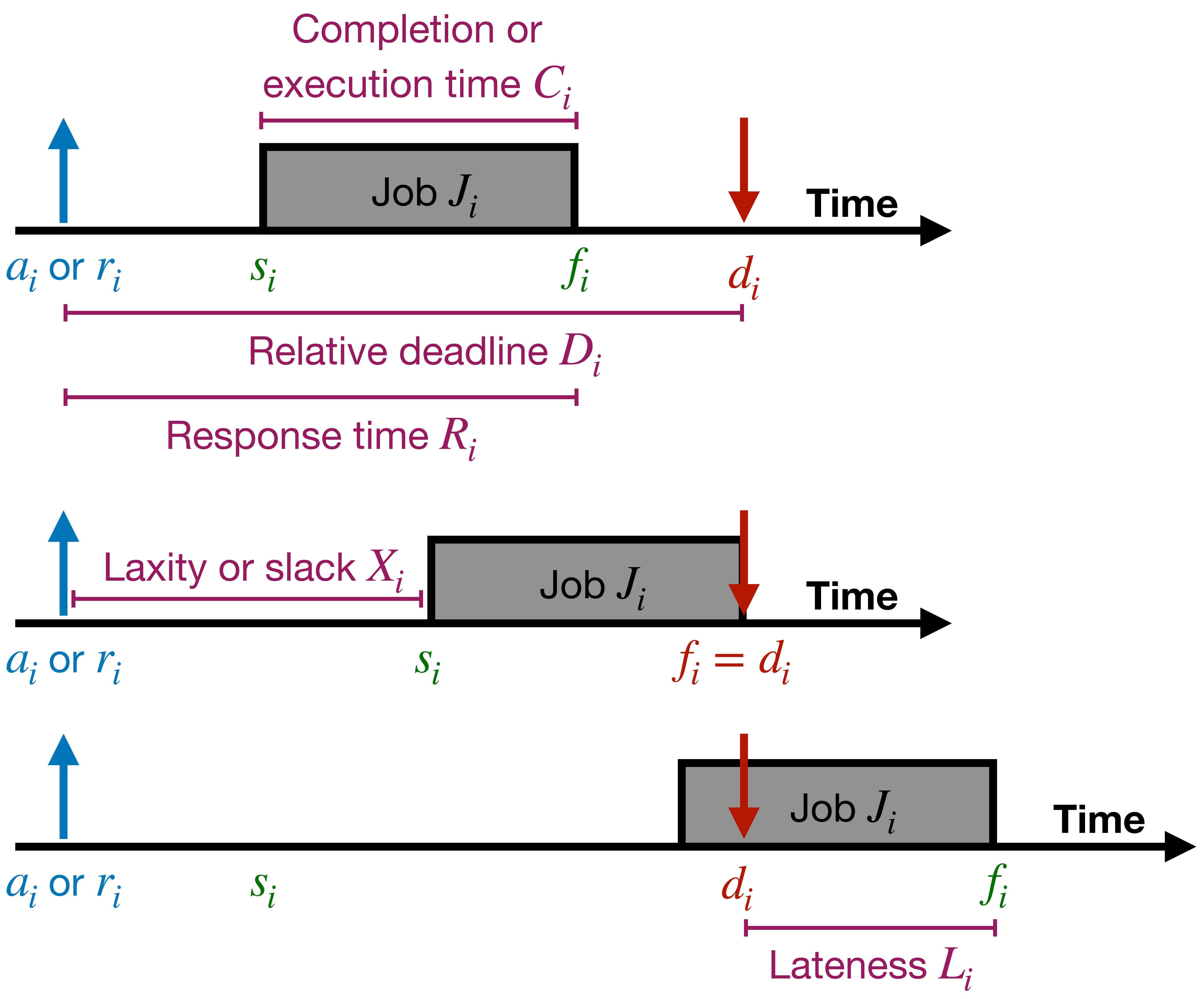
CPEN 432 Real-Time System Design

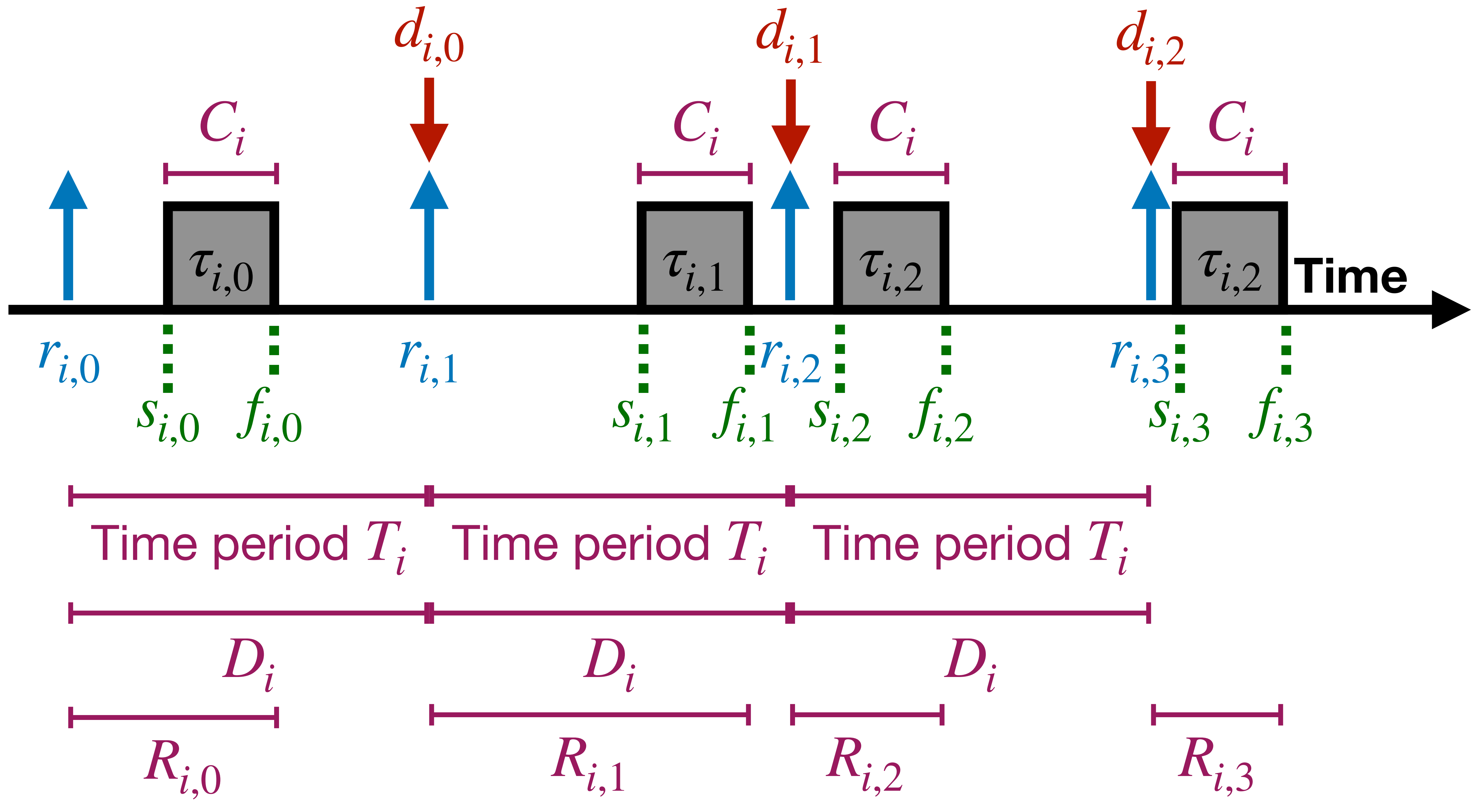
Arpan Gujarati
University of British Columbia

Assignment 1

- Deadline is **11:59 PM, 7 February, 2022**

Recap: Aperiodic Job vs. Periodic Task





Some property Z

- response time
- slack
- lateness
- etc.

$$\overline{Z}_{\underline{i},\overline{k}}$$

Corresponding to the k^{th} activation of task τ_i

Corresponding to task τ_i

Task response time
$R_i = \max_k (R_{i,k})$

Recap: Assumptions

A1: All jobs of τ_i are regularly activated at a **constant frequency** of $1/T_i$

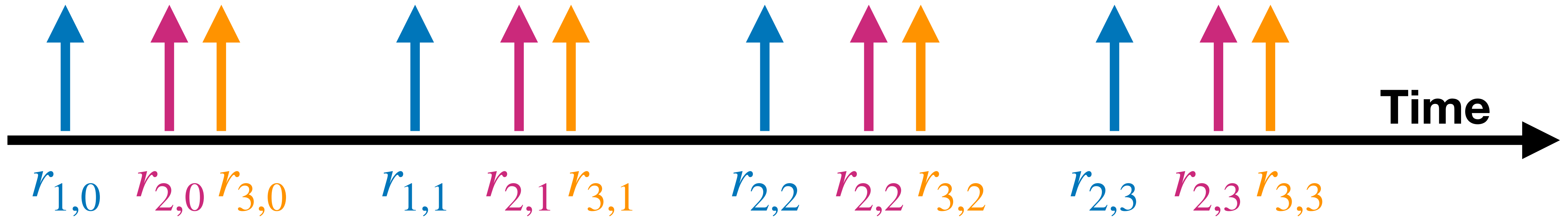
A2: All jobs of τ_i have the **same worst-case execution time** C_i

A3. All jobs of τ_i have the **same relative deadline** $D_i = T_i$

A4. All tasks in τ are **independent** (no dependencies, no shared resources)

Recap: Assumptions

- The tasks **need not be released synchronously**
  - E.g., it is possible that $r_{1,0} \neq r_{2,0} \neq \ldots \neq r_{n,0}$



- The tasks can be **preempted** in between

Rate Monotonic Scheduling

Recap: Overview

- RM is a **fixed-priority** scheduling algorithm
  - Each task is assigned a priority beforehand
- RM assigns priorities based on **task frequency**
  - Higher frequency (smaller time period) $\implies$ Higher priority
- Famous result by Liu and Layland [1973]
  - RM is **optimal** among all fixed-priority algorithms
    - i.e., no fixed-priority algorithm can schedule a task set that cannot be scheduled by RM
    - i.e., if any fixed-priority algorithm can schedule a task set, RM can also schedule the task set

RM Schedulability Test

RM Schedulability Test

- Processor utilization factor
  - Fraction of processor time spent executing tasks in $\tau = \{\tau_1, \tau_2, \ldots, \tau_n\}$

$$\boxed{U = \sum_{i=1}^{n} \frac{C_i}{T_i}}$$

RM Schedulability Test

$$U = \sum_{i=1}^{n} \frac{C_i}{T_i}$$

- Processor utilization factor
  - Fraction of processor time spent executing tasks in $\tau = \{\tau_1, \tau_2, \ldots, \tau_n\}$
- By simply checking the utilization, can we say if RM can schedule it?
  - I.e., can we find U_{ub} such that
    - if $U \leq U_{ub}$, irrespective of the task parameters, τ is schedulable by R

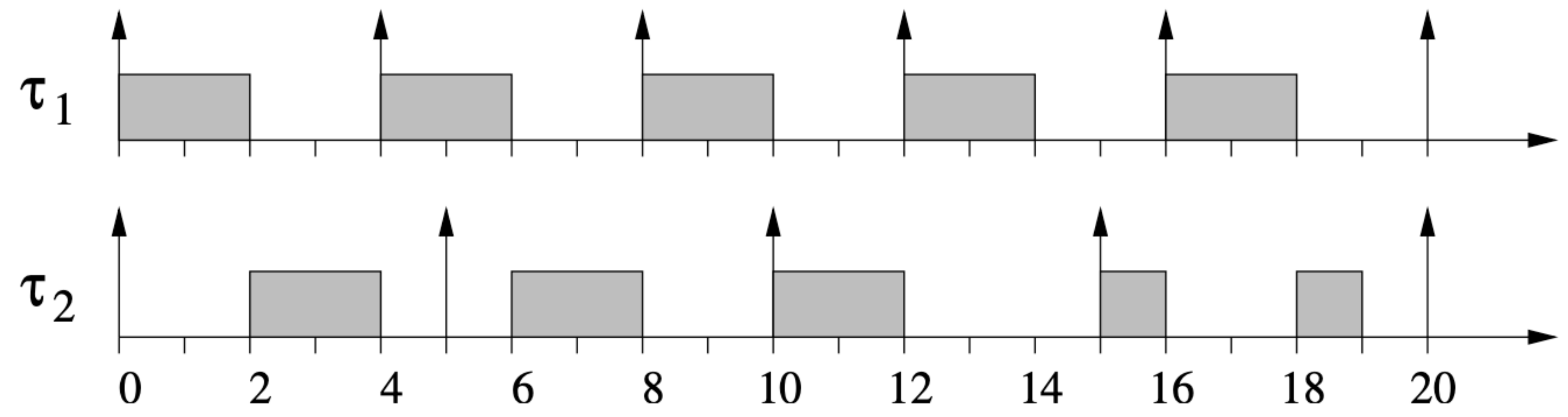
RM Schedulability Test

RM Schedulability Test

- Example
    - $U_{ub} = 1.0?$

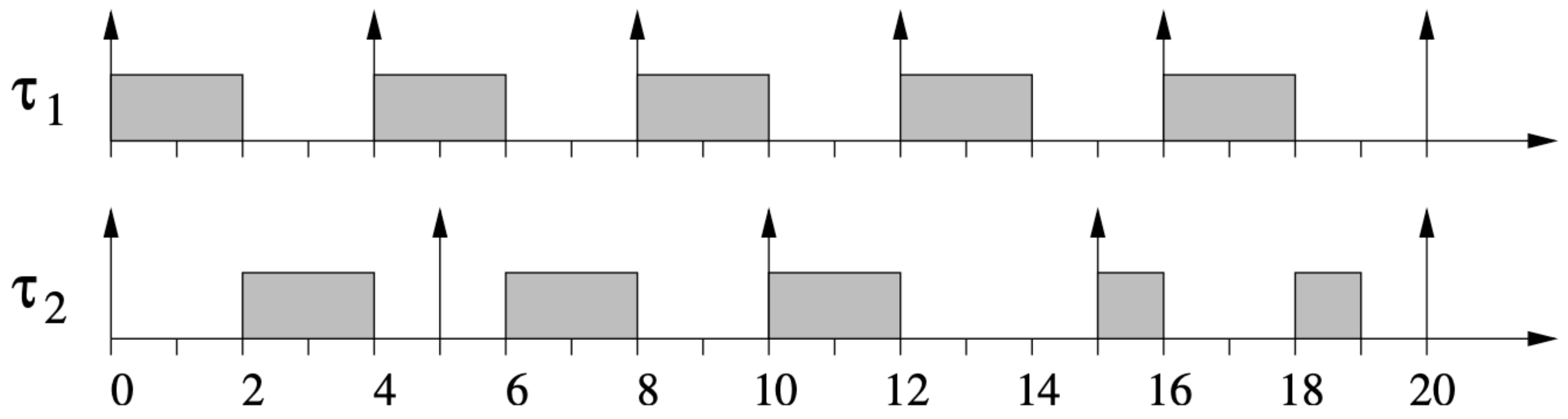
RM Schedulability Test

- Example
  - $U_{ub} = 1.0$?



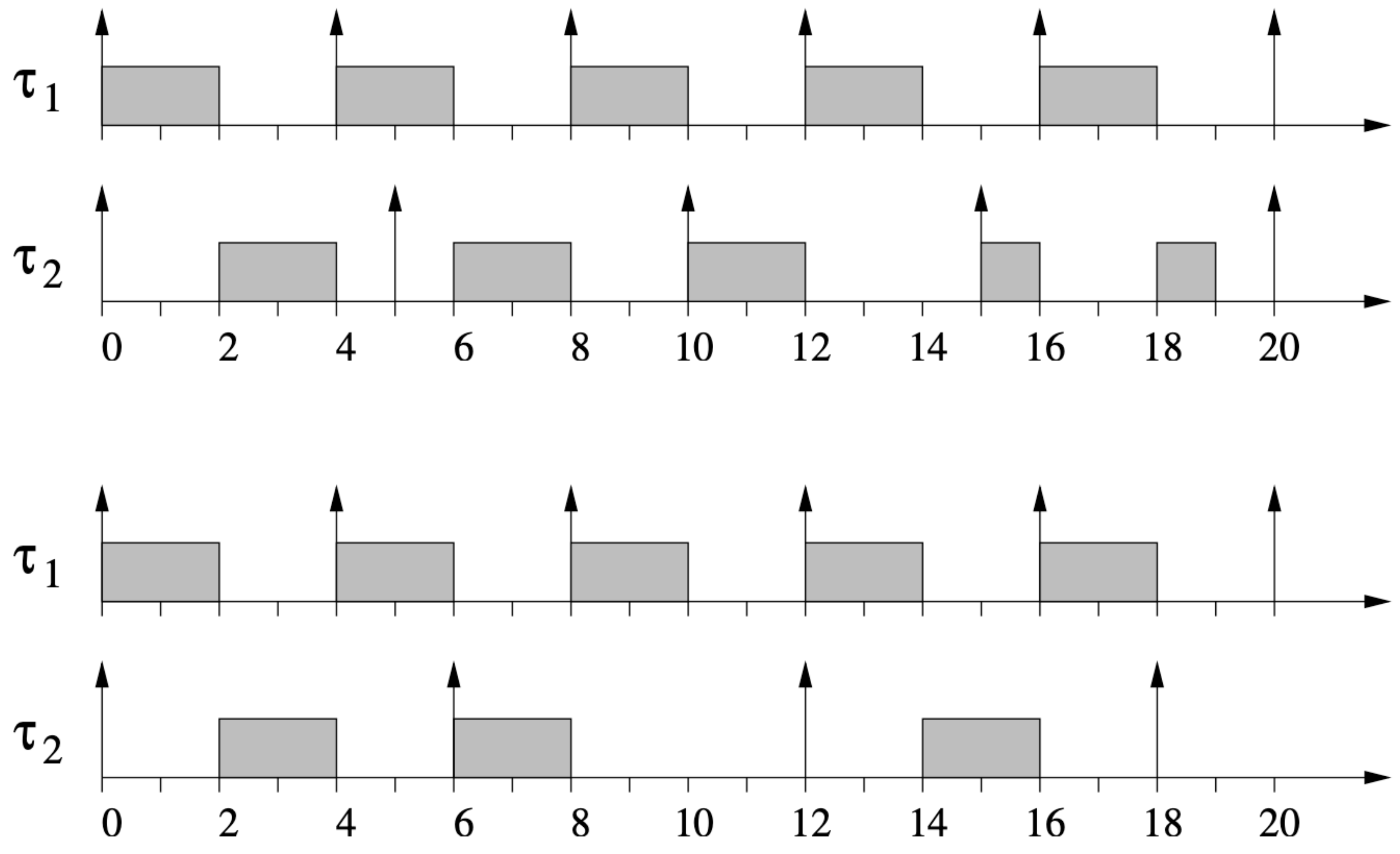
RM Schedulability Test

- Example
  - $U_{ub} = 1.0?$
  - $U_{ub} = 0.9?$



RM Schedulability Test

- Example
  - $U_{ub} = 1.0?$
  - $U_{ub} = 0.9?$



RM Utilization Bound Derivation [1/n]

RM Utilization Bound Derivation [1/n]

- For simplicity
  - Let $\tau = \{\tau_1, \tau_2\}$ such that $T_1 < T_2$

RM Utilization Bound Derivation [1/n]

- For simplicity
  - Let $\tau = \{\tau_1, \tau_2\}$ such that $T_1 < T_2$
- Only two fixed-priority assignments possible
  - RM: τ_1 is assigned the higher priority
  - ~~Algorithm A: τ_2 is assigned the higher priority~~ (we only care about RM!)

RM Utilization Bound Derivation [1/n]

- For simplicity
  - Let $\tau = \{\tau_1, \tau_2\}$ such that $T_1 < T_2$
- Only two fixed-priority assignments possible
  - RM: τ_1 is assigned the higher priority
  - ~~Algorithm A: τ_2 is assigned the higher priority~~ (we only care about RM!)
- Recall the **critical instant** theorem
  - It suffices to check for a task's schedulability when it is **released simultaneously with all higher-priority tasks**

RM Utilization Bound Derivation [1/n]

- For simplicity
  - Let $\tau = \{\tau_1, \tau_2\}$ such that $T_1 < T_2$
- Only two fixed-priority assignments possible
  - RM: τ_1 is assigned the higher priority
  - ~~Algorithm A: τ_2 is assigned the higher priority~~ (we only care about RM!)
- Recall the **critical instant** theorem
  - It suffices to check for a task's schedulability when it is **released simultaneously with all higher-priority tasks**
- Proof sketch
  - Step 1: Given T_1, T_2, and C_1, find the maximum value for C_2 such that RM can schedule τ
    - This gives us $U_{ub} = f(T_1, T_2, C_1)$, such that for any C_2, task set utilization $U \leq U_{ub}$ guarantees that τ is schedulable using RM

RM Utilization Bound Derivation [1/n]

- For simplicity
  - Let $\tau = \{\tau_1, \tau_2\}$ such that $T_1 < T_2$
- Only two fixed-priority assignments possible
  - RM: τ_1 is assigned the higher priority
  - ~~Algorithm A: τ_2 is assigned the higher priority~~ (we only care about RM!)
- Recall the **critical instant** theorem
  - It suffices to check for a task's schedulability when it is **released simultaneously with all higher-priority tasks**
- Proof sketch
  - Step 1: Given T_1, T_2, and C_1, find the maximum value for C_2 such that RM can schedule τ
    - This gives us $U_{ub} = f(T_1, T_2, C_1)$, such that for any C_2, task set utilization $U \leq U_{ub}$ guarantees that τ is schedulable using RM
  - Step 2: Minimize U_{ub} with respect to C_1
    - This gives us $U'_{ub} = g(T_1, T_2)$, such that for any C_1 and C_2, task set utilization $U \leq U'_{ub}$ guarantees that τ is schedulable using RM

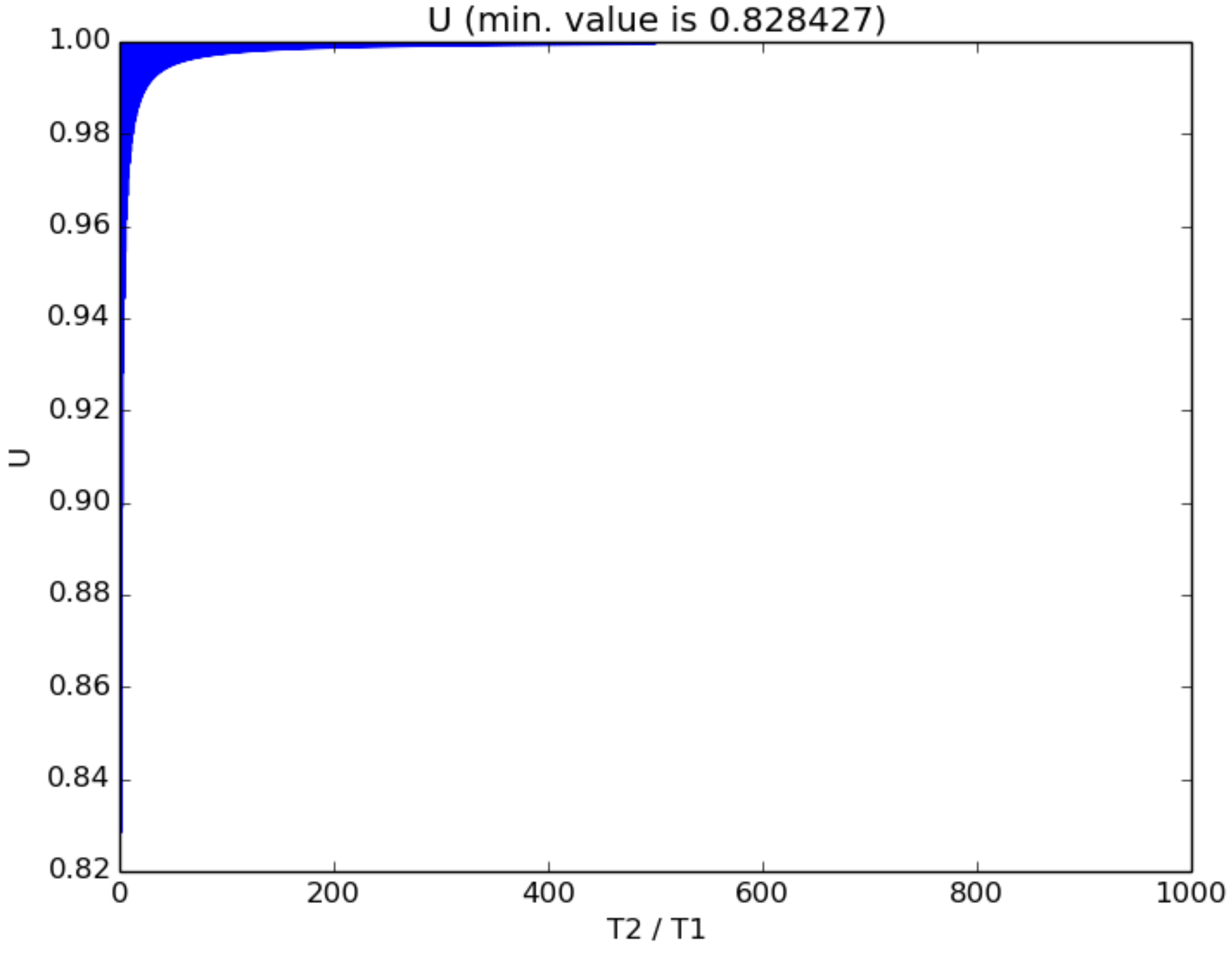
RM Utilization Bound Derivation [1/n]

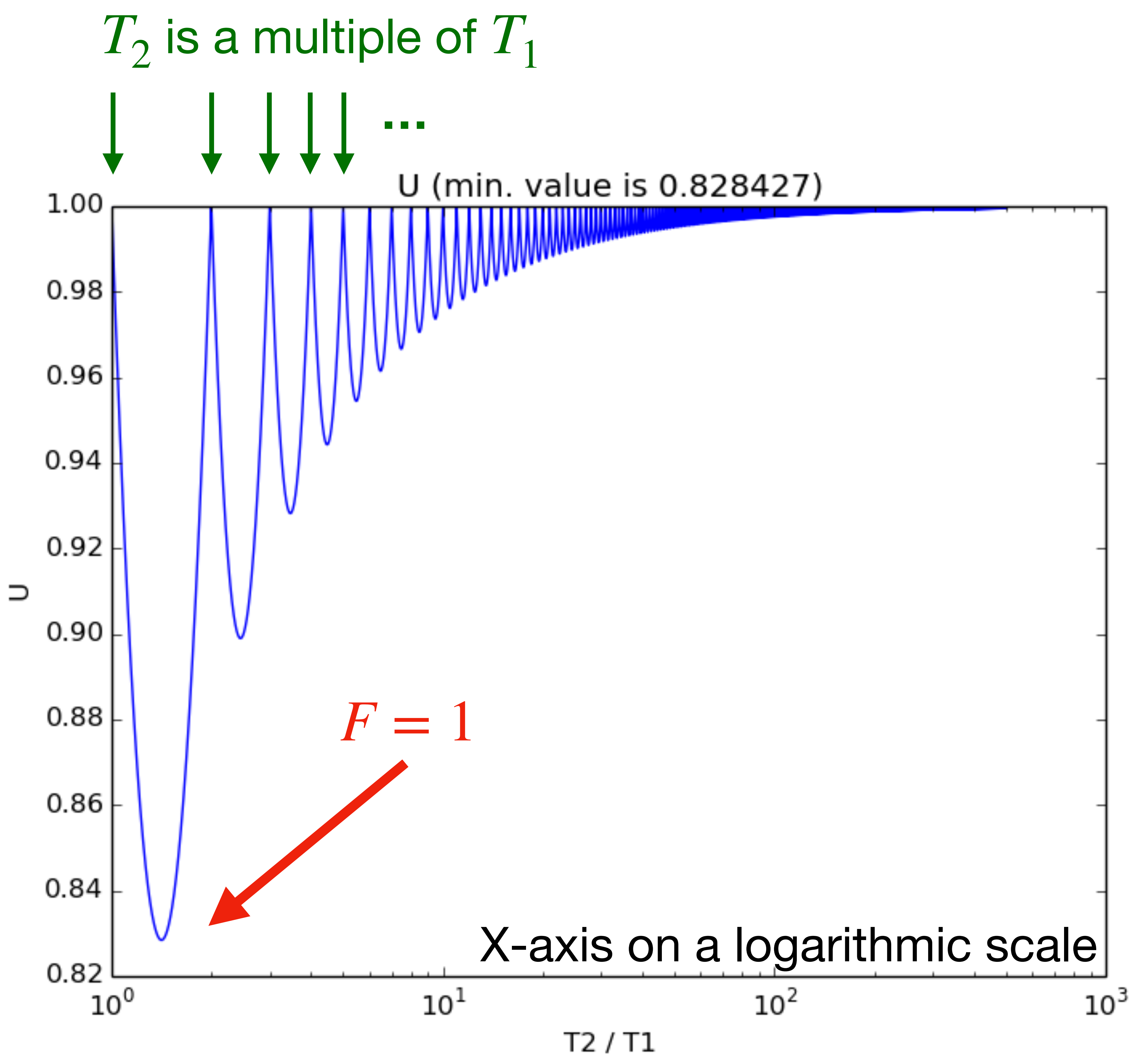
- For simplicity
  - Let $\tau = \{\tau_1, \tau_2\}$ such that $T_1 < T_2$
- Only two fixed-priority assignments possible
  - RM: τ_1 is assigned the higher priority
  - ~~Algorithm A: τ_2 is assigned the higher priority~~ (we only care about RM!)
- Recall the **critical instant** theorem
  - It suffices to check for a task's schedulability when it is **released simultaneously with all higher-priority tasks**
- Proof sketch
  - Step 1: Given T_1, T_2, and C_1, find the maximum value for C_2 such that RM can schedule τ
    - This gives us $U_{ub} = f(T_1, T_2, C_1)$, such that for any C_2, task set utilization $U \leq U_{ub}$ guarantees that τ is schedulable using RM
  - Step 2: Minimize U_{ub} with respect to C_1
    - This gives us $U'_{ub} = g(T_1, T_2)$, such that for any C_1 and C_2, task set utilization $U \leq U'_{ub}$ guarantees that τ is schedulable using RM
  - Step 3: Minimize U'_{ub} with respect to T1 and T2
    - This gives us U''_{ub} (constant), such that for any C_1, C_2, T_1, and T_2, task set utilization $U \leq U''_{ub}$ guarantees that τ is schedulable using RM

RM Utilization Bound Derivation [2/n]

RM Utilization Bound Derivation [3/n]

Equation 4.5 from the textbook: $U = \frac{T_1}{T_2}\left[F + \left(\frac{T_2}{T_1} - F\right)\left(\frac{T_2}{T_1} - F\right)\right]$





Earliest Deadline First

Example

Task ID	Time Period T	Computation Time C
1	5 ms	2 ms
2	7 ms	4 ms

RM

EDF

EDF Utilization Bound

- What?
- Intuition?

RM and EDF's Utilization Bounds

What if $D_i \leq T_i$?

Recap: Assumptions

A1: All jobs of τ_i are regularly activated at a **constant frequency** of $1/T_i$

A2: All jobs of τ_i have the **same worst-case execution time** C_i

A3. All jobs of τ_i have the **same relative deadline** $\cancel{D_i = T_i}$ $C_i \leq D_i \leq T_i$

A4. All tasks in τ are **independent** (no dependencies, no shared resources)

Is RM still optimal?