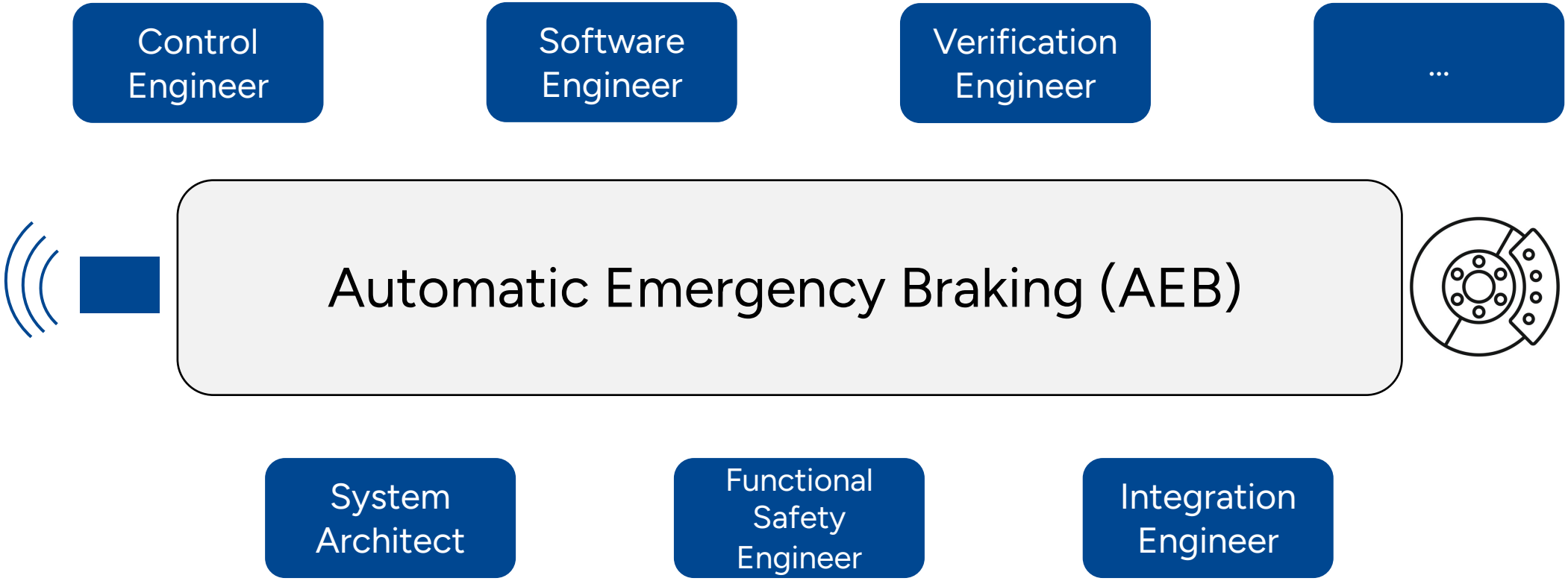
Challenges in the design of cause-effect chains: How to tune the end-to-end latency?

Matthias Becker

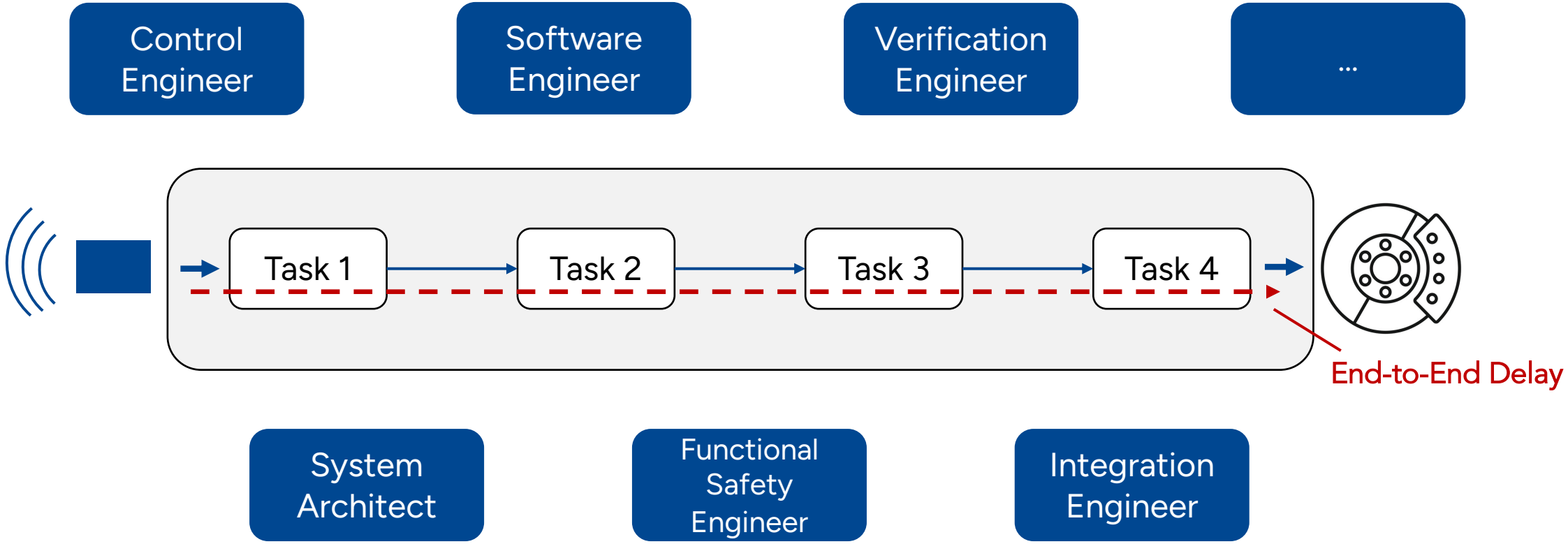
15th International Real-Time Systems Open Problems Seminar
Lund, Sweden 2026

Introduction



- Functionality is often described as a chain of communicating tasks.
- A ***cause-effect chain*** is a sequence of communicating tasks $E = (\tau_1 \rightarrow \cdots \rightarrow \tau_n)$

Introduction



- Functionality is often described as a chain of communicating tasks.
- A ***cause-effect chain*** is a sequence of communicating tasks $E = (\tau_1 \rightarrow \cdots \rightarrow \tau_n)$

Introduction

Control Engineer

Software Engineer

Verification Engineer

...

Where during the design process can we affect the end-to-end latency?

What options do we have at the different stages to affect the end-to-end latency?

Are there good mechanisms to affect the end-to-end latency already at early design phases, without detailed information of the final system?

System Architect

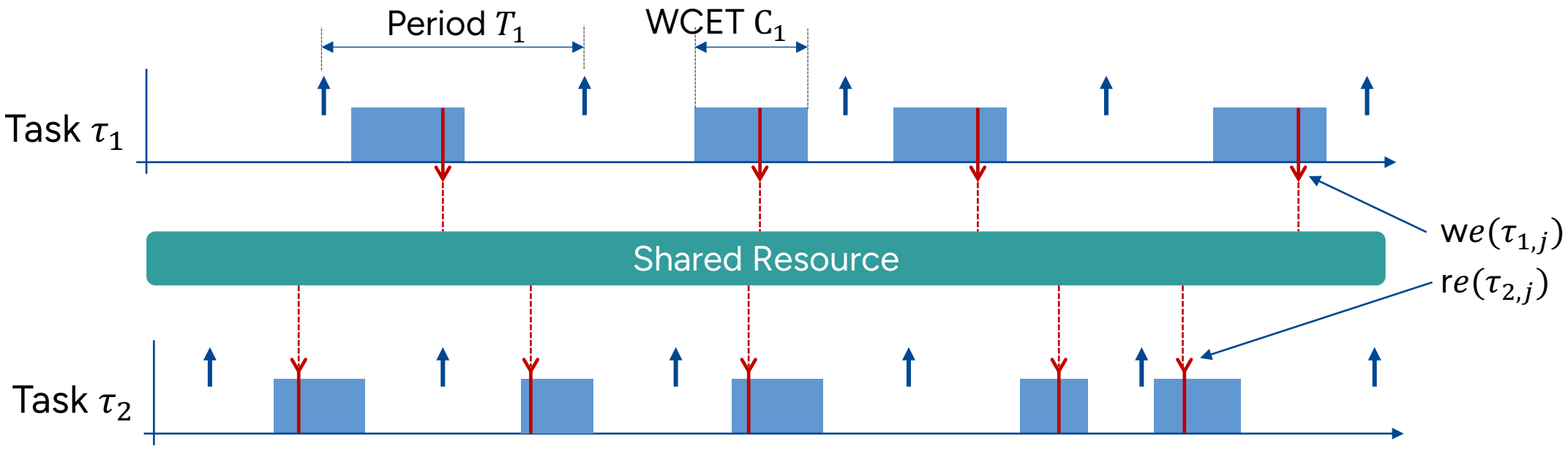
Functional Safety Engineer

Integration Engineer

- Functionality is often described as a chain of communicating tasks.
- A ***cause-effect chain*** is a sequence of communicating tasks $E = (\tau_1 \rightarrow \cdots \rightarrow \tau_n)$

Tasks and Communication

- Periodic tasks that communicate via shared resources
- A task sends data by *writing* to the shared resource → Write event of a job: $we(\tau_{i,j})$
- A task receives data by *reading* from the shared resource → Read event of a job: $re(\tau_{i,j})$

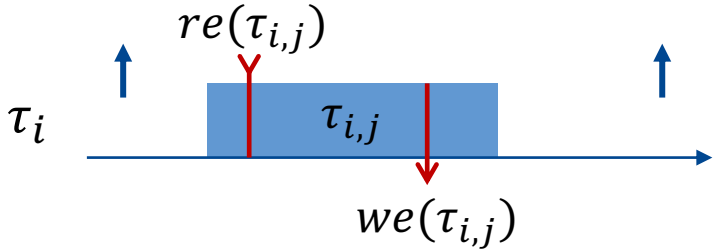


Typically, a shared resource is called a label

Communication Semantics

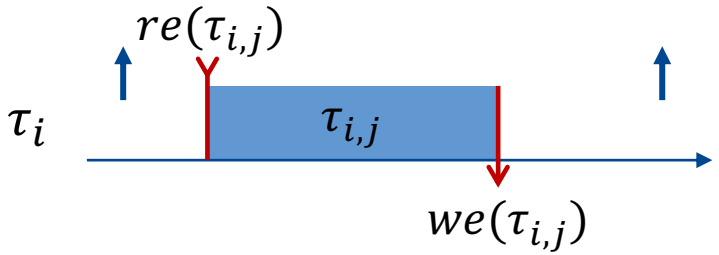
Explicit Communication

- Read- and write-event can happen anytime.
- Read must happen before write.



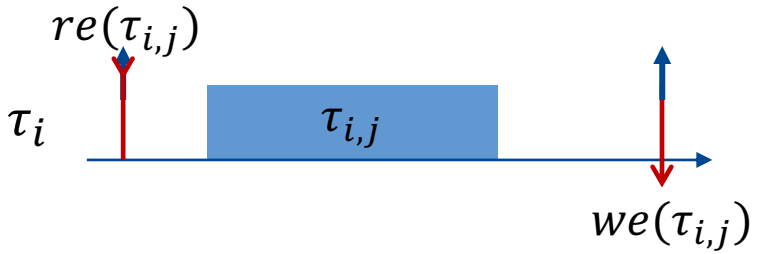
Implicit Communication

- Read-event happens at the job's start time.
- Write-event happens at the job's finish time.
- A task creates a local copy of the data.



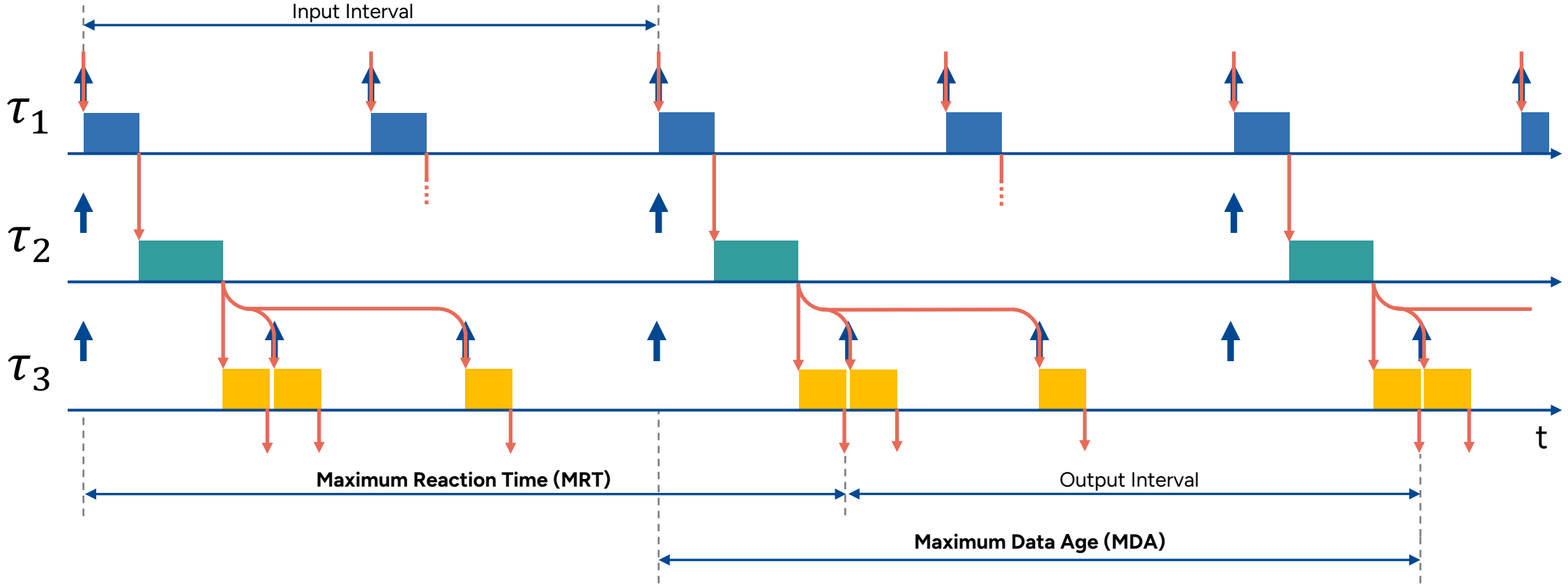
Logical Evecution Time

- Read- and write-event at task-specific offset.
- Typically, read-event at release and write-event at deadline.
- A task creates a local copy of the data.



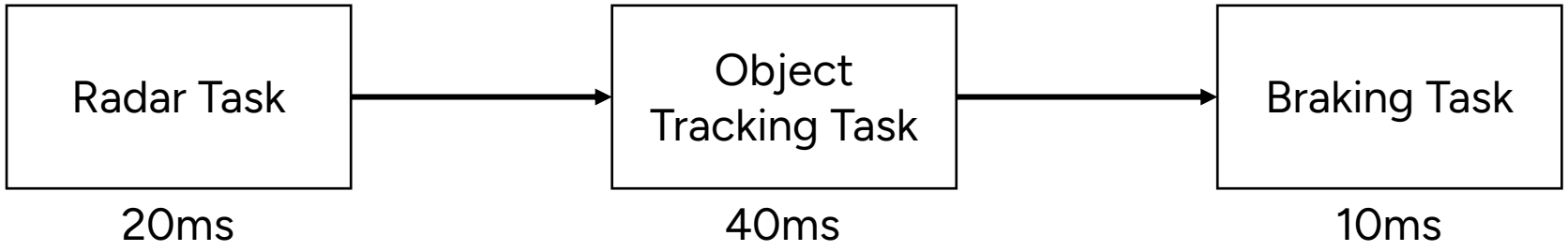
End-to-End Latency Metrics

- Assumption for the figure: read-event at start time, write-event at finish time (implicit com.)
  $E = (\tau_1 \rightarrow \tau_2 \rightarrow \tau_3)$



Example: Automatic Emergency Braking (AEB)

The vehicle must brake <= 100ms after an obstacle is detected.



(this is simplified!!!)

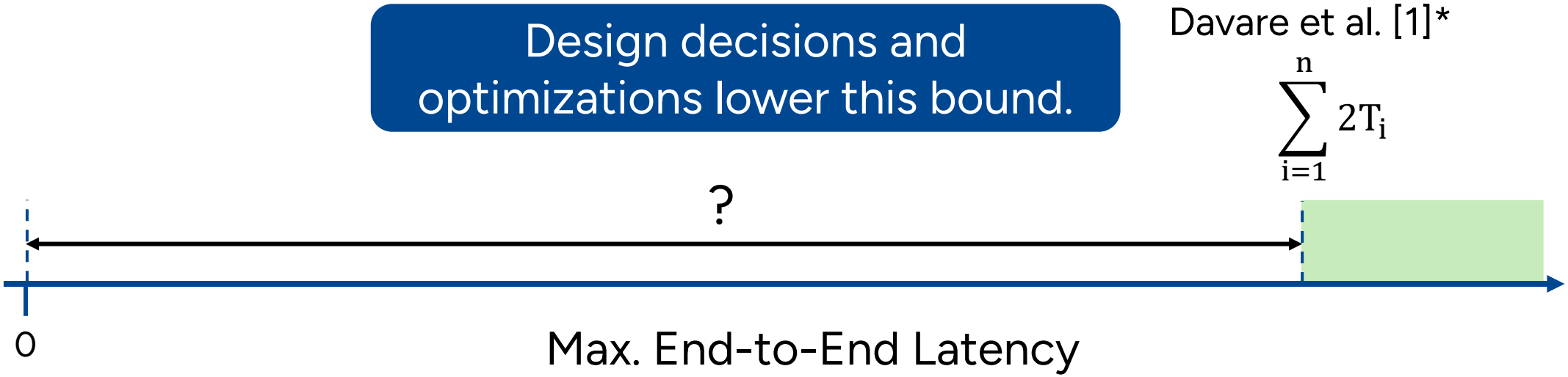
The Design Challenge

- The end-to-end latency is dependent on:
  - Periods selected
  - Task to ECU and core allocation
  - Task WCET
  - Scheduling algorithms used
  - Communication semantics (LET, implicit, etc.)
  - Network communication between ECUs
  - …
- As a consequence, the final maximum end-to-end latency of the system is known only late during system design

What can we say about the maximum end-to-end latency based on the task periods and execution time estimates alone?

Maximum End-to-End Latency in Early Design Phases

- **Assumption:** Early design phase, only task periods and execution time estimates known (tasks have implicit deadlines and BCET=WCET)



* The bound in [1] considers $R_i + T_i$, I assume here that we only know that the task meets its implicit deadline, hence $2T_i$.

[1] A. Davare et al. *Period Optimization for Hard Real-time Distributed Automotive Systems* (DAC 2007)

Optimizing End-to-End Latency

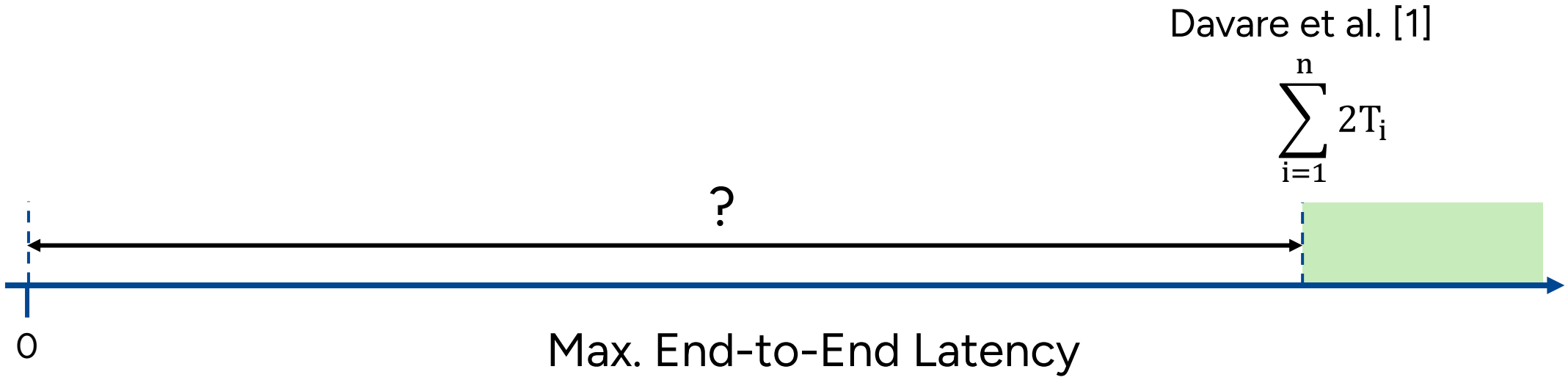
- Change the phase of the chain's task
- Assign priorities to the chain's task
- Use schedulers designed to minimize end-to-end latency
- Specify job-level dependencies
- etc.

Many "knobs" to tune. All with the same goal.

What is the minimum achievable maximum end-to-end latency L^*, based on the chain structure itself?

Maximum End-to-End Latency in Early Design Phases

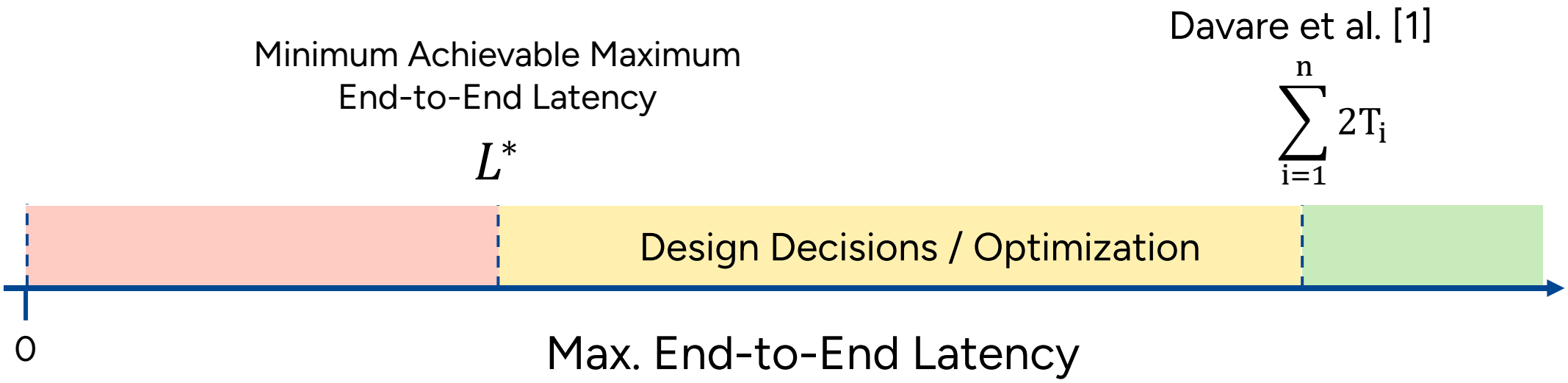
- **Assumption:** Early design phase, only task periods and execution time estimates known (tasks have implicit deadlines)



[1] A. Davare et al. *Period Optimization for Hard Real-time Distributed Automotive Systems* (DAC 2007)

Maximum End-to-End Latency in Early Design Phases

- **Assumption:** Early design phase, only task periods and execution time estimates known (tasks have implicit deadlines)



[1] A. Davare et al. *Period Optimization for Hard Real-time Distributed Automotive Systems* (DAC 2007)

Why this can be useful

- Knowing the minimum achievable maximum end-to-end latency allows to design optimization strategies that tune parameters available in the specific design for the execution pattern leading to L^*
  - Fixed priority scheduling → priorities
  - Task phase
  - Deadlines
  - Job-level dependencies
- Knowing which combination of optimization strategies can be applied for a specific setting can decouple the overall optimization strategie from the concrete realization for the concrete setting

LET Chains with Semi-Harmonic Periods

Assumptions:

$$\frac{\max_{\tau' \in \Gamma} T_{\tau'}}{T_\tau} \in \mathbb{N} \text{ for all } \tau \in \mathbb{T}$$

The period of every task evenly divides the largest task period of the chain.

- Implicit Deadlines
- Read event is located at the task release
- Write event is located at the task deadline

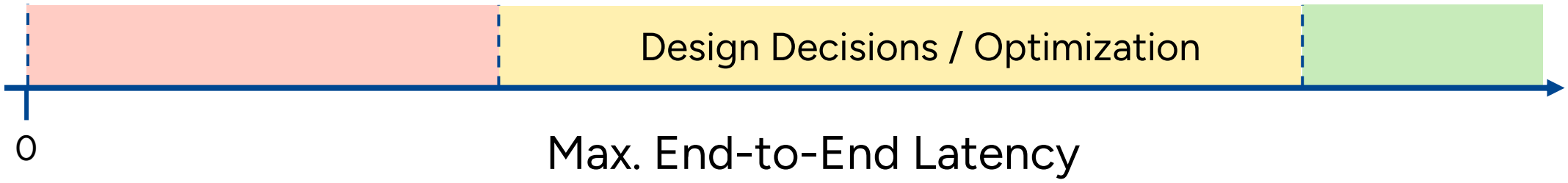
Thus, only the task phase can be optimized!

Günzel et al. [3]

$$\max_{i=1,\ldots,n} T_i + \sum_{i=1}^{n} T_i$$

Hamann et al. [2]

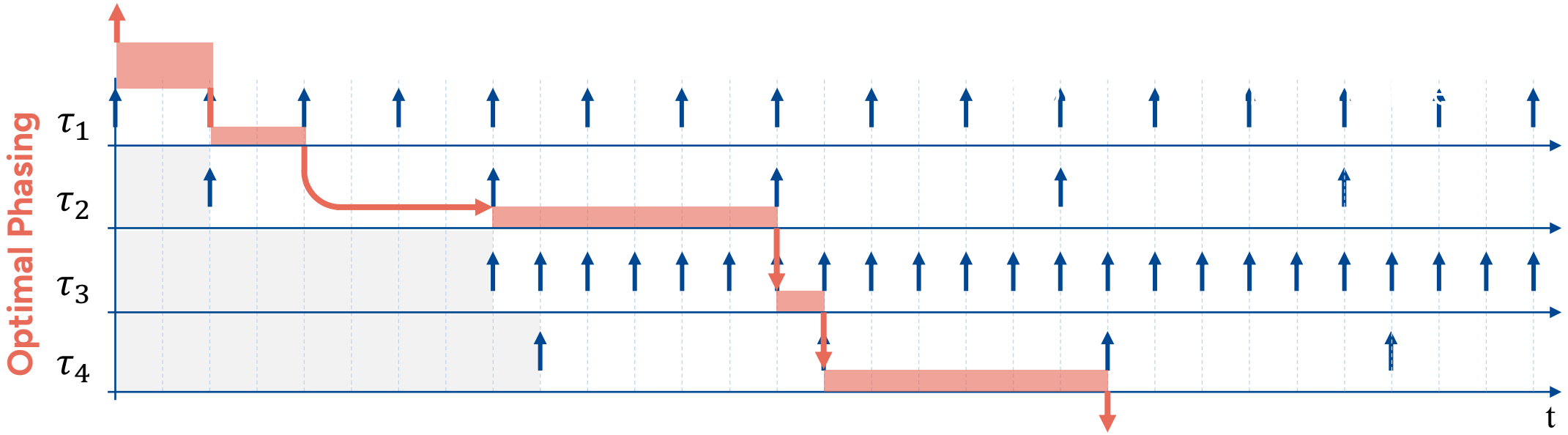
$$\sum_{i=1}^{n} 2T_i$$



[2] Hamann et al. “Communication centric design in complex automotive embedded systems.” ECRTS, 2017.
[3] Günzel and Becker. “Optimal Task Phasing for End-To-End Latency in Harmonic and Semi-Harmonic Automotive Systems.” RTAS, 2025.

Optimal Offsets for Semi-Harmonic LET Chains

- Align read and write events → $\phi_i^{MH} = \sum_{k=i}^{i-1} T_k$ for all $\tau_i \in \mathrm{E}$

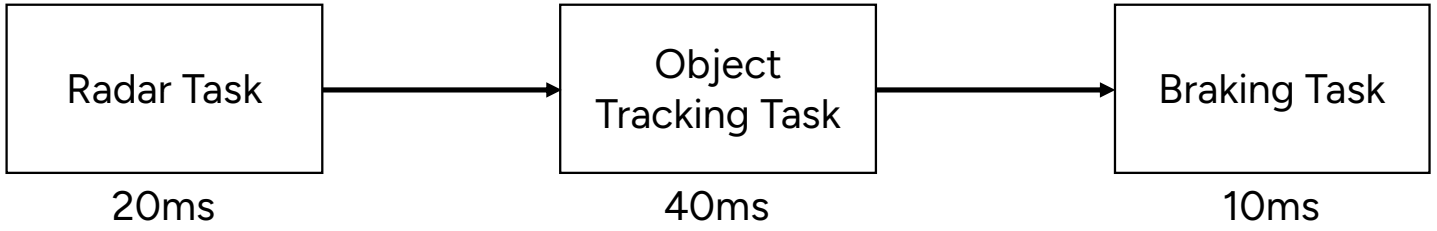


[3] Günzel and Becker. "Optimal Task Phasing for End-To-End Latency in Harmonic and Semi-Harmonic Automotive Systems." RTAS, 2025.

What is the best we can do for implicit communication?

The goal is to find the minimum achievable maximum end-to-end latency with implicit communication, based on the chain structure (i.e., periods and execution time estimates). Applicable at early design phases.

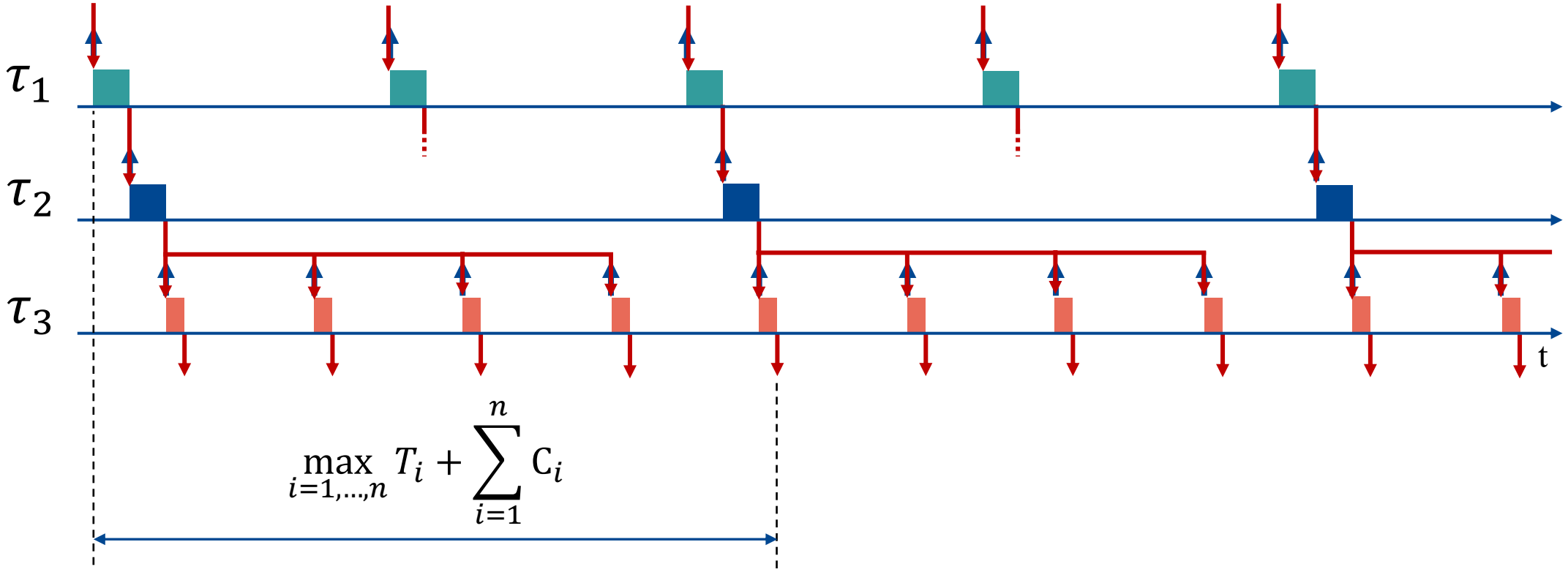
- The AEB example:



Note that the periods here are harmonic

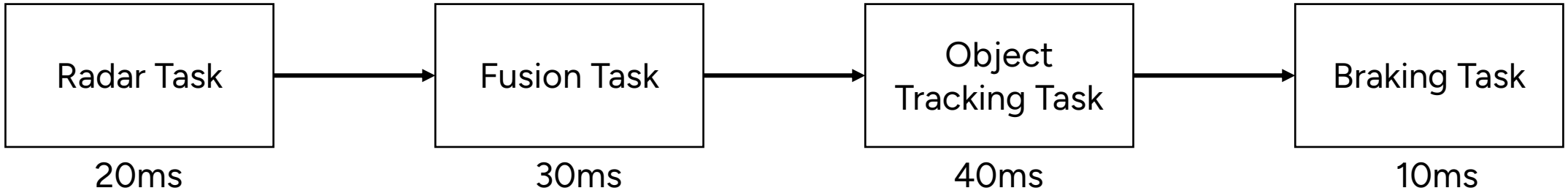
Implicit Communication and Harmonic Periods

- Same idea: Align read and write events, starting from jobs of the task with largest period
- Consecutive jobs of the task with largest period must have the same distance

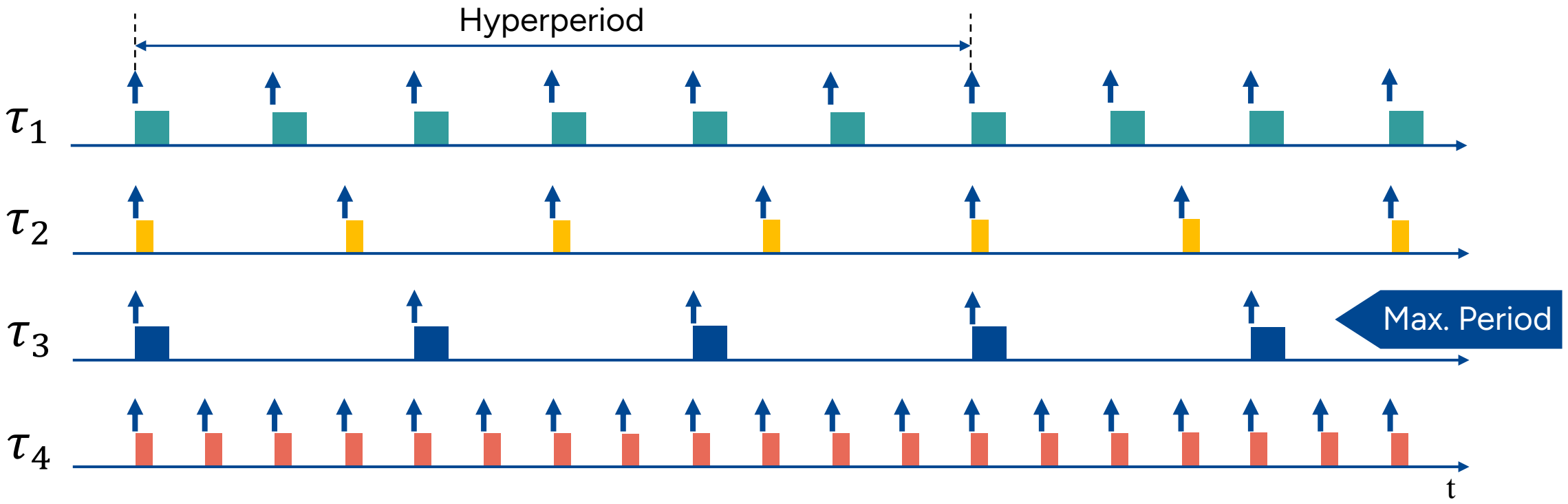


From Harmonic to General Case?

- Does this still work for the case with non-harmonic periods?
- Extend the AEB example with a fusion task → The hyperperiod becomes 120ms

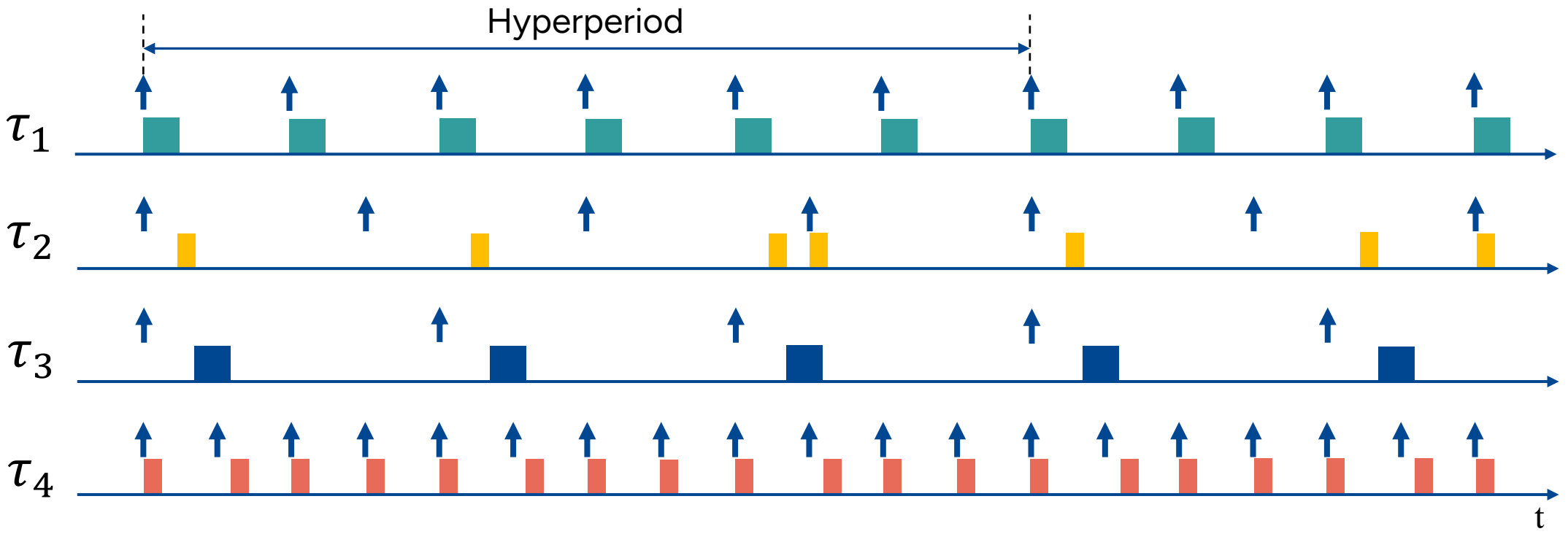


Implicit Communication



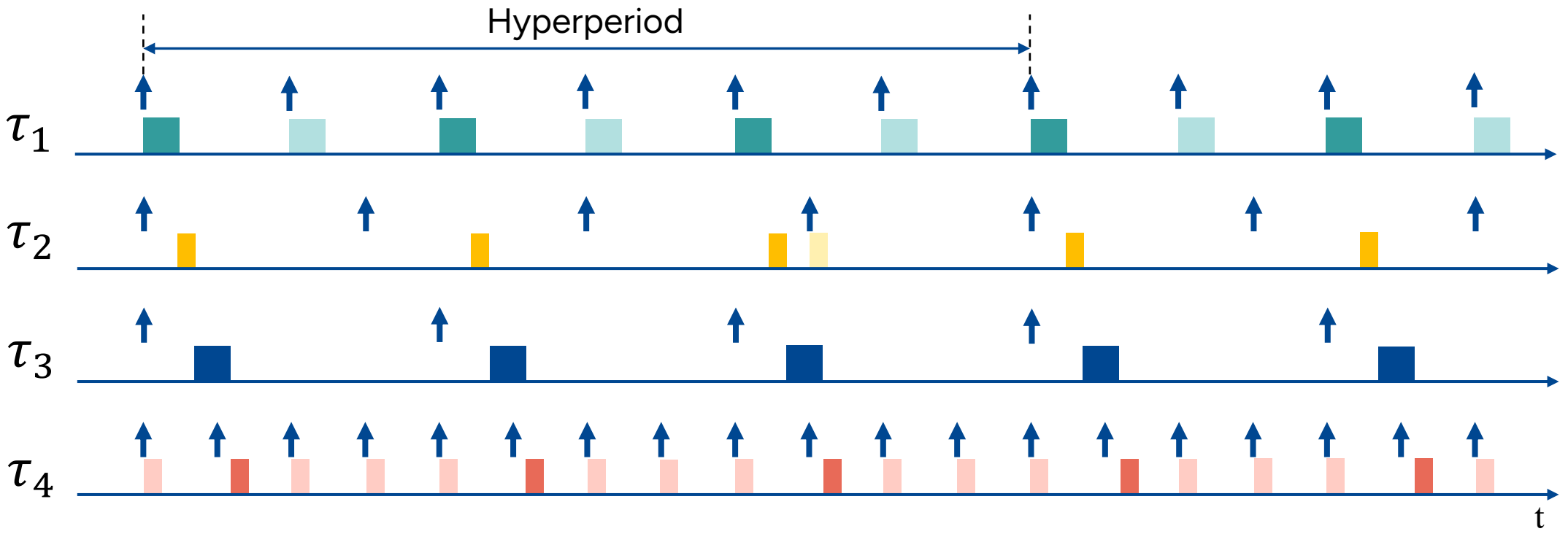
- For each job of the task with max. period, there is at least one job of any other task
- We can construct the data propagation up/down from the jobs of the task with max. period

Implicit Communication



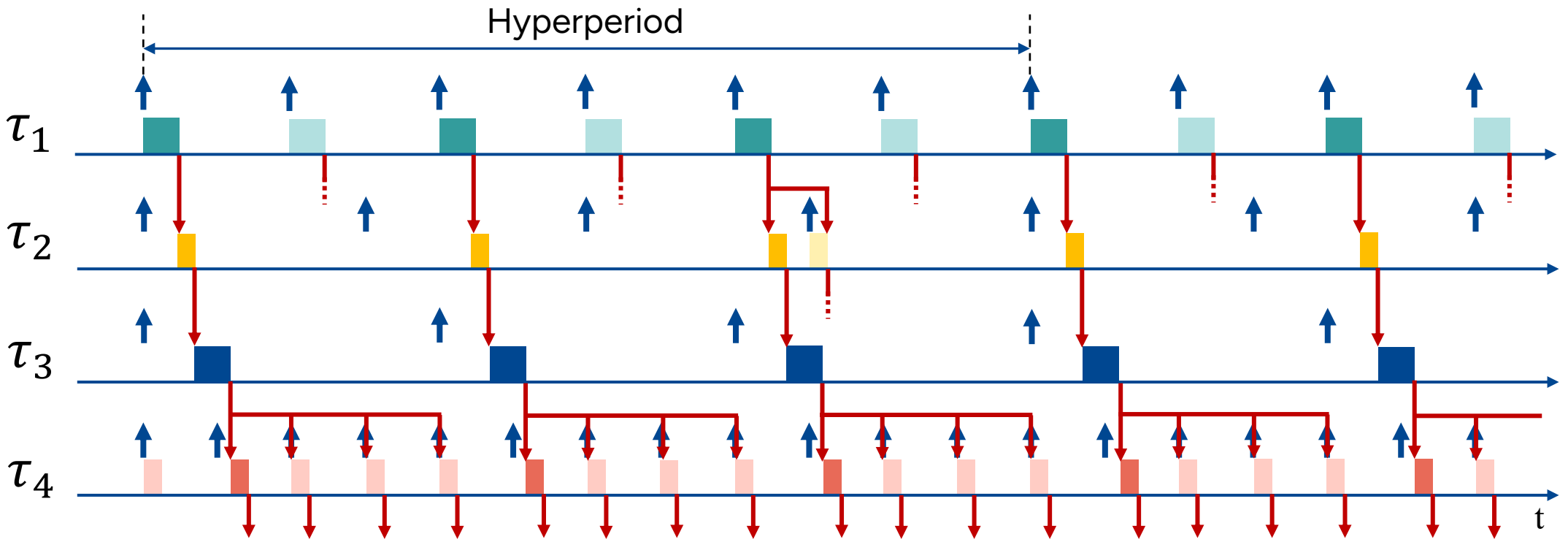
- For each job of the task with max. period, there is at least one job of any other task
- We can construct the data propagation up/down from the jobs of the task with max. period
- The exact position of all remaining jobs does not affect the end-to-end latency

Implicit Communication

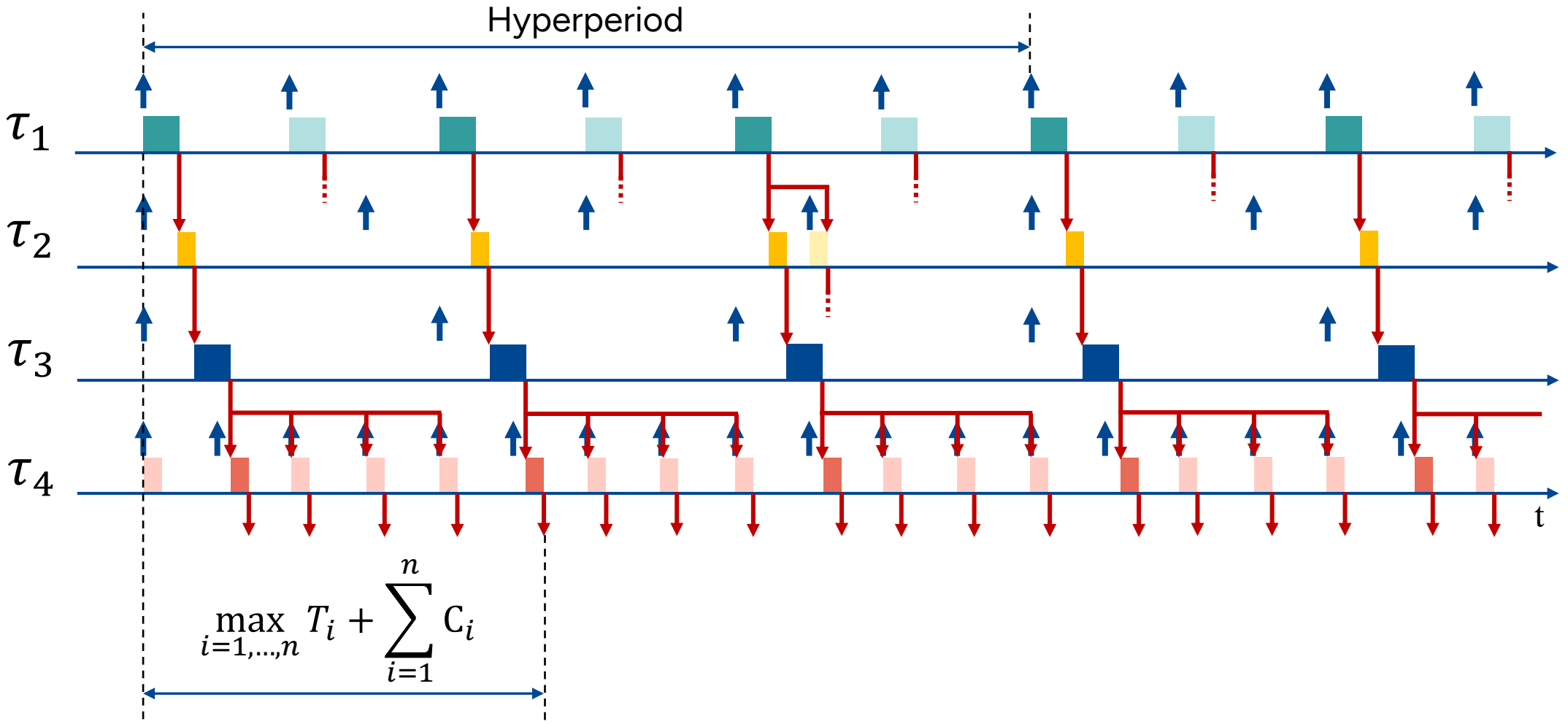


- For each job of the task with max. period, there is at least one job of any other task
- We can construct the data propagation up/down from the jobs of the task with max. period
- The exact position of all remaining jobs does not affect the end-to-end latency

Implicit Communication



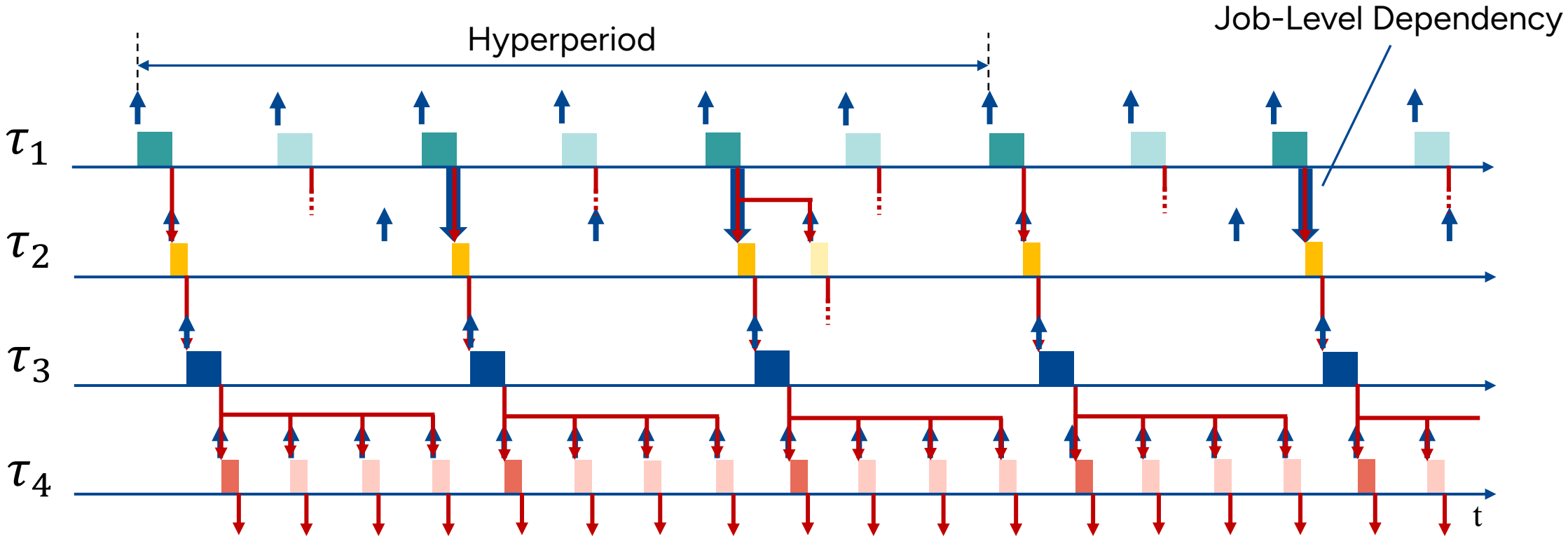
Implicit Communication



Observations

- The minimum achievable maximum end-to-end latency based on the chain structure alone consists of two parts:
  - The propagation through all tasks
  - The maximum period of any task in the chain
- The constructed schedule does not consider schedulability, but can be a good reference for the chain characteristics at early design phases (i.e. what is the limit) and may be a good starting point to design optimization strategies
- A single optimization strategy likely will not achieve L^*
  - How close can we get by a single optimization strategy?

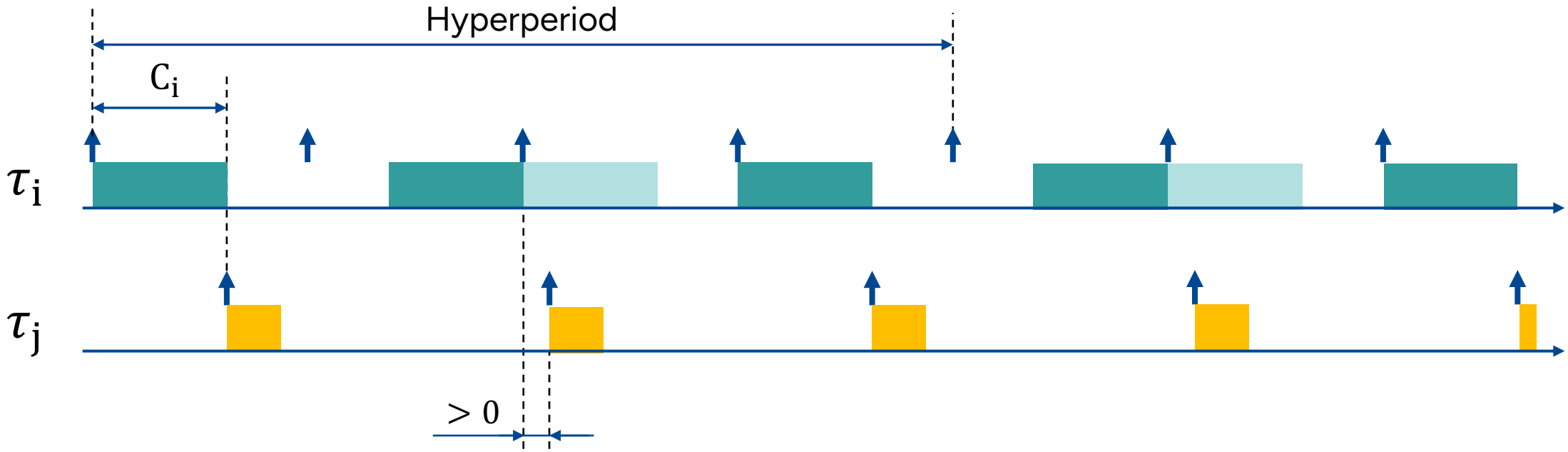
Implicit Communication – Fixed Priority Scheduling



- Task phases set to align read and write events of first jobs
- Priority assignment, decreasing along the chain
- Job-level dependencies to enforce cases where periods are not harmonic
- Two cores available to schedule the tasks

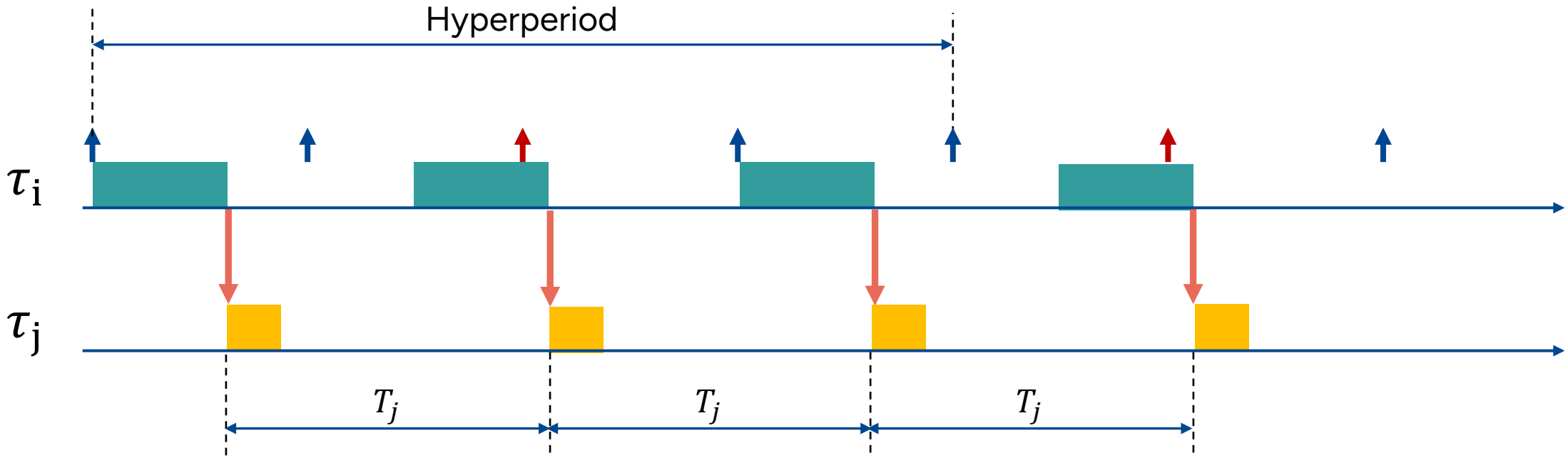
Does this always work?

- No, consider... $E = (\cdots \rightarrow \tau_i \rightarrow \tau_j \rightarrow \cdots)$ $T_j = \max_{i=1,\ldots,n} T_i = 6$ $Ti = 4$ $C_i = 2{,}5$ $C_j = 1$



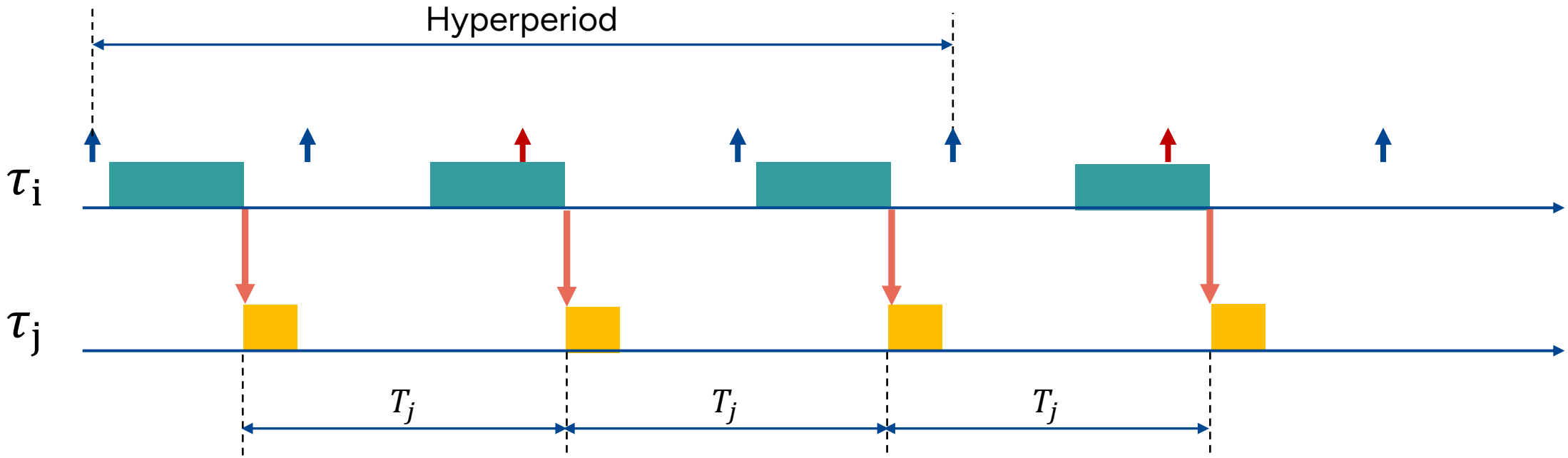
Does this always work?

- No, consider... $E = (\cdots \rightarrow \tau_{\mathrm{i}} \rightarrow \tau_{\mathrm{j}} \rightarrow \cdots)$ $\mathrm{T_j} = \max_{i=1,\ldots,n} T_i = 6$ $Ti = 4$ $C_i = 2{,}5$ $C_{\mathrm{j}} = 1$



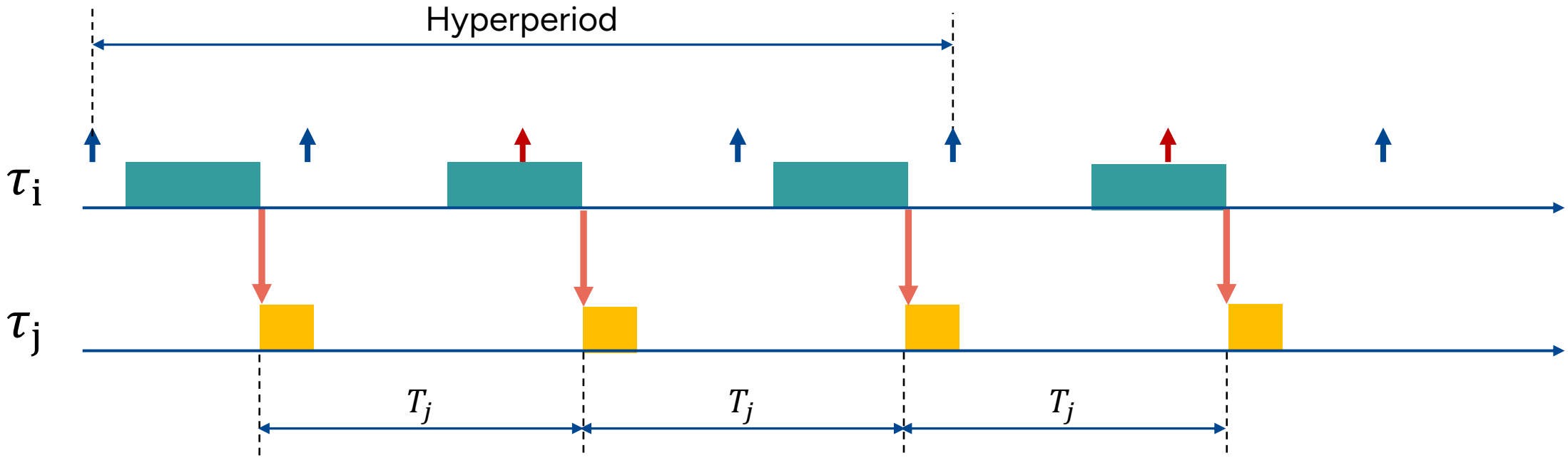
Does this always work?

- No, consider... $E = (\cdots \rightarrow \tau_{\mathrm{i}} \rightarrow \tau_{\mathrm{j}} \rightarrow \cdots)$ $\mathrm{T_j} = \max_{i=1,\ldots,n} T_i = 6$ $Ti = 4$ $C_i = 2{,}5$ $C_{\mathrm{j}} = 1$



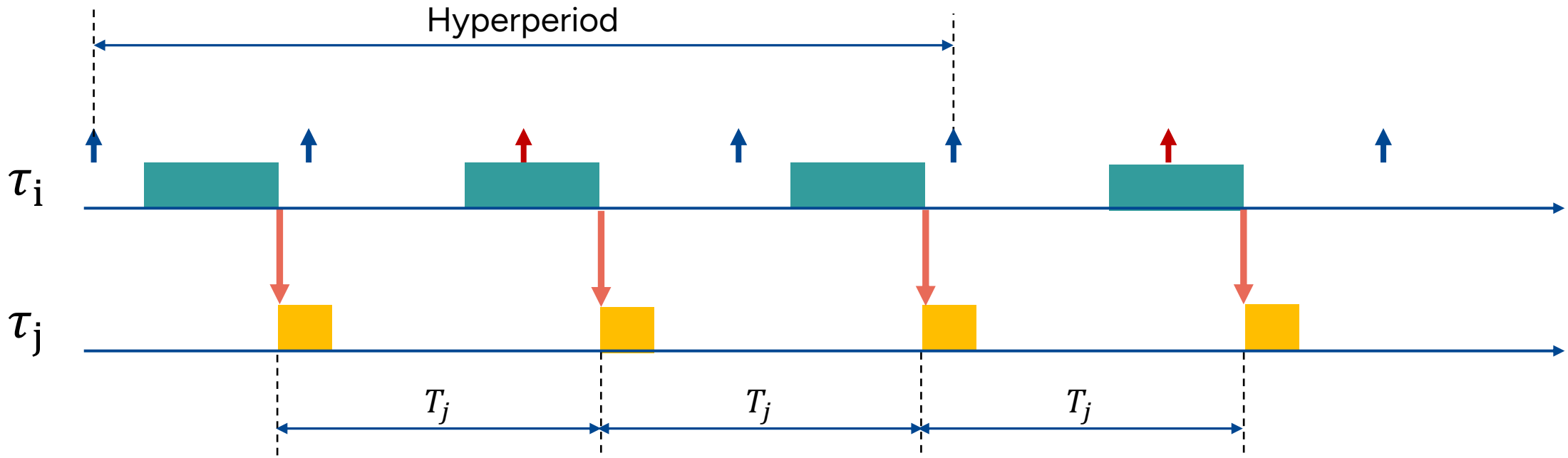
Does this always work?

- No, consider... $E = (\cdots \rightarrow \tau_{\mathrm{i}} \rightarrow \tau_{\mathrm{j}} \rightarrow \cdots)$ $\mathrm{T_j} = \max_{i=1,\ldots,n} T_i = 6$ $Ti = 4$ $C_i = 2{,}5$ $C_{\mathrm{j}} = 1$



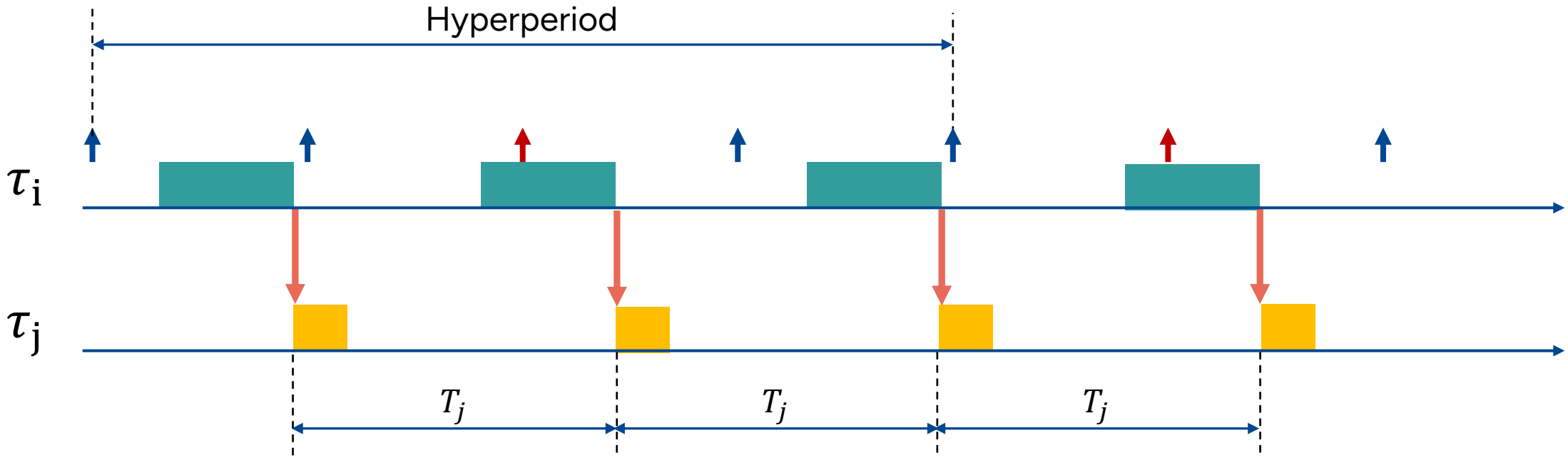
Does this always work?

- No, consider... $E = (\cdots \rightarrow \tau_{\mathrm{i}} \rightarrow \tau_{\mathrm{j}} \rightarrow \cdots)$ $\mathrm{T_j} = \max_{i=1,\ldots,n} T_i = 6$ $Ti = 4$ $C_i = 2{,}5$ $C_{\mathrm{j}} = 1$



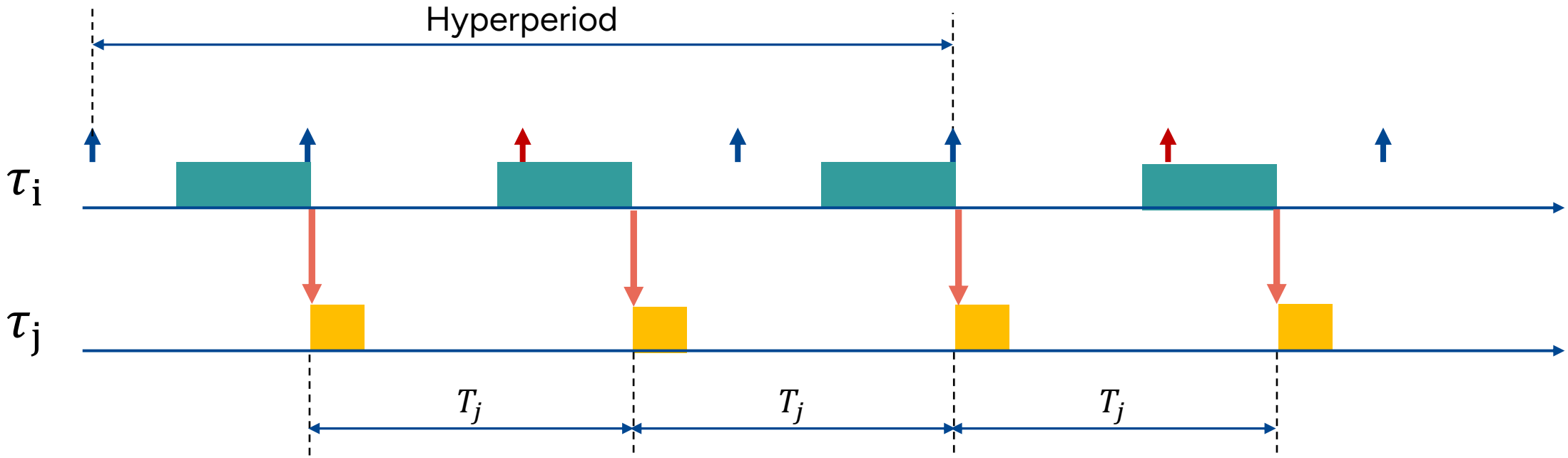
Does this always work?

- No, consider... $E = (\cdots \rightarrow \tau_{\mathrm{i}} \rightarrow \tau_{\mathrm{j}} \rightarrow \cdots)$ $\mathrm{T_j} = \max_{i=1,\ldots,n} T_i = 6$ $Ti = 4$ $C_i = 2{,}5$ $C_{\mathrm{j}} = 1$



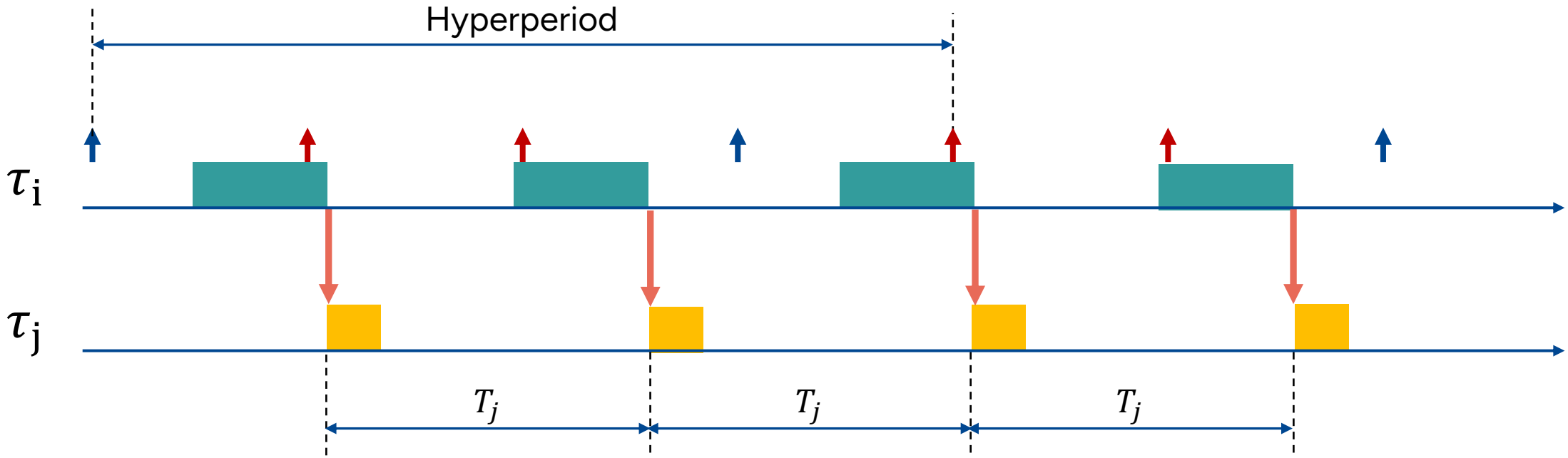
Does this always work?

- No, consider... $E = (\cdots \rightarrow \tau_{\mathrm{i}} \rightarrow \tau_{\mathrm{j}} \rightarrow \cdots)$ $\mathrm{T_j} = \max_{i=1,\ldots,n} T_i = 6$ $Ti = 4$ $C_i = 2{,}5$ $C_{\mathrm{j}} = 1$



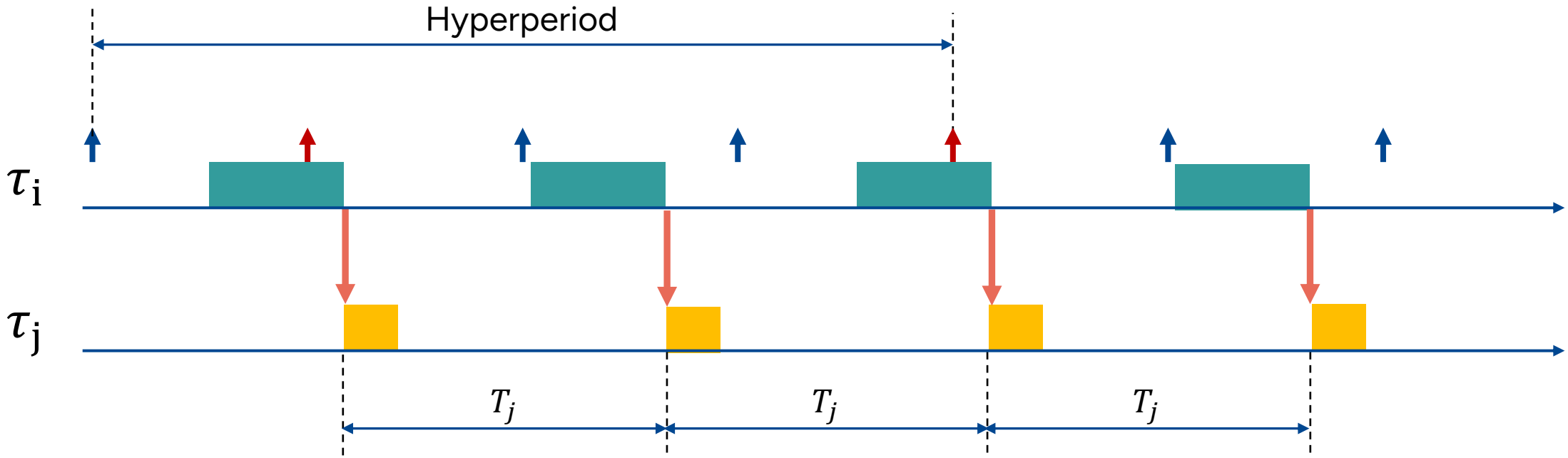
Does this always work?

- No, consider... $E = (\cdots \rightarrow \tau_\mathrm{i} \rightarrow \tau_\mathrm{j} \rightarrow \cdots)$ $\mathrm{T_j} = \max_{i=1,\ldots,n} T_i = 6$ $Ti = 4$ $C_i = 2{,}5$ $C_\mathrm{j} = 1$



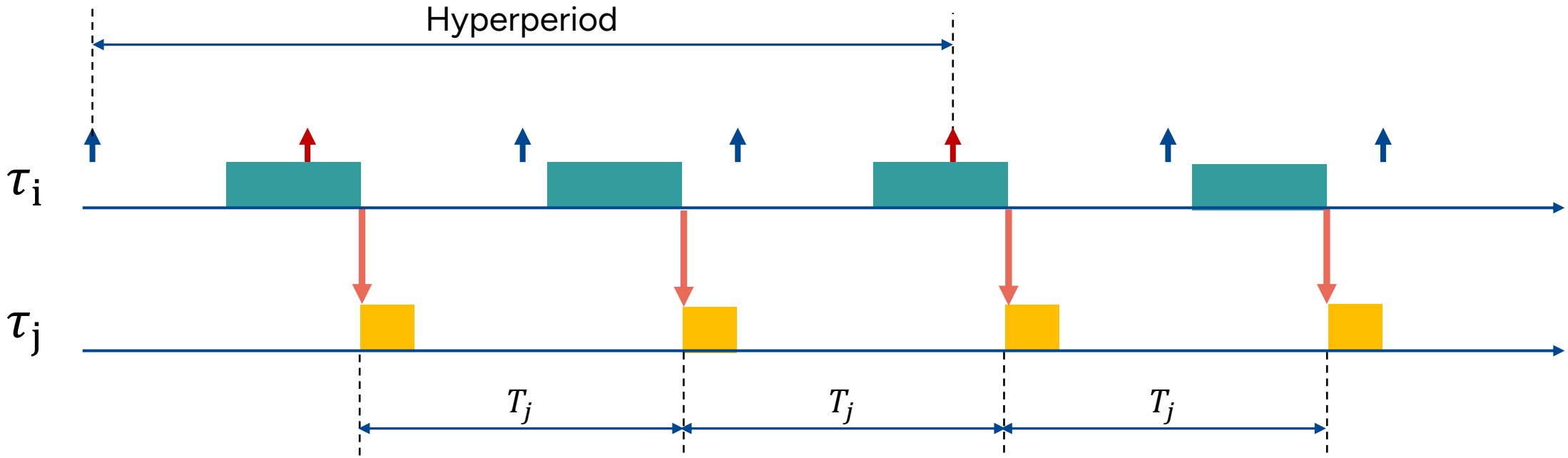
Does this always work?

- No, consider... $E = (\cdots \rightarrow \tau_{\mathrm{i}} \rightarrow \tau_{\mathrm{j}} \rightarrow \cdots)$ $\mathrm{T_j} = \max_{i=1,\ldots,n} T_i = 6$ $Ti = 4$ $C_i = 2{,}5$ $C_{\mathrm{j}} = 1$



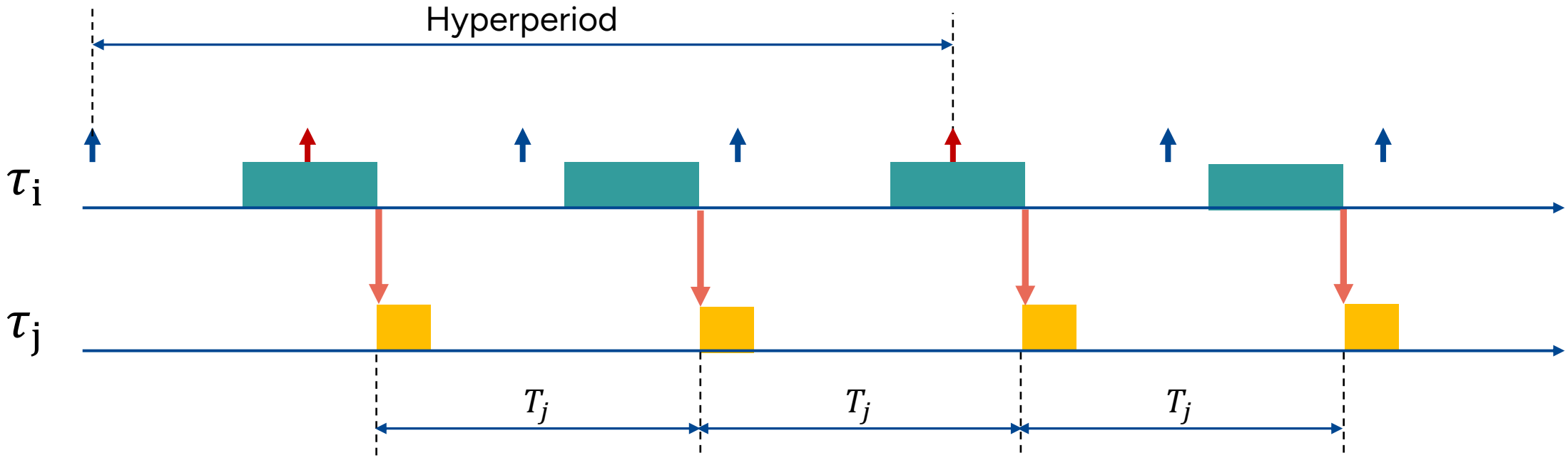
Does this always work?

- No, consider... $E = (\cdots \rightarrow \tau_{\mathrm{i}} \rightarrow \tau_{\mathrm{j}} \rightarrow \cdots)$ $\mathrm{T_j} = \max_{i=1,\ldots,n} T_i = 6$ $Ti = 4$ $C_i = 2{,}5$ $C_{\mathrm{j}} = 1$



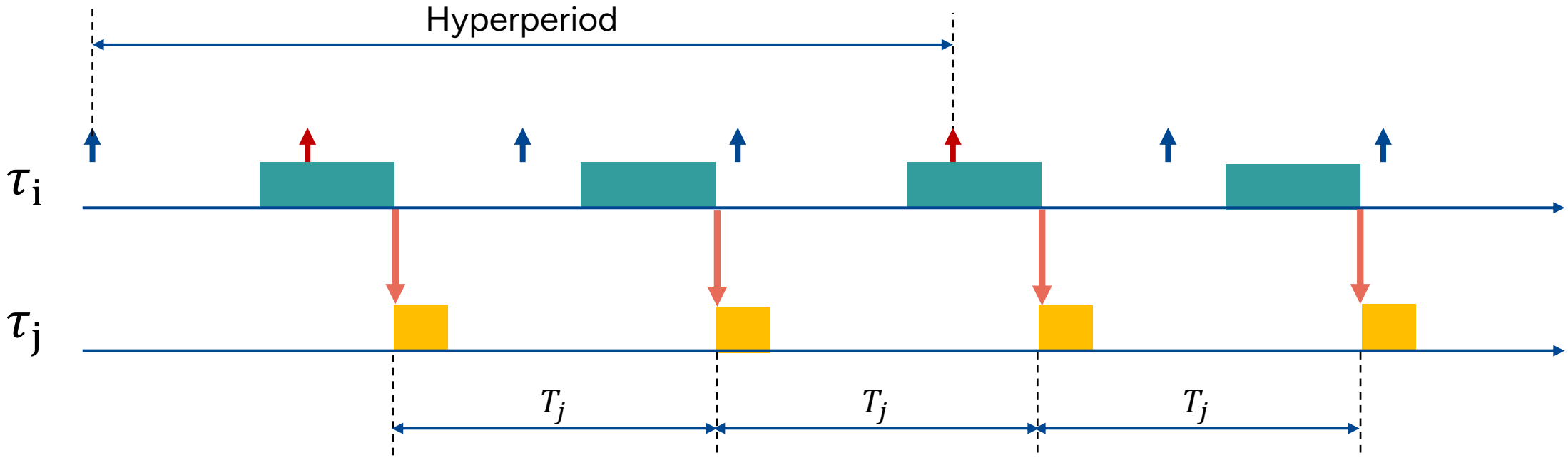
Does this always work?

- No, consider... $E = (\cdots \rightarrow \tau_{\mathrm{i}} \rightarrow \tau_{\mathrm{j}} \rightarrow \cdots)$ $\mathrm{T_j} = \max_{i=1,\ldots,n} T_i = 6$ $Ti = 4$ $C_i = 2{,}5$ $C_{\mathrm{j}} = 1$



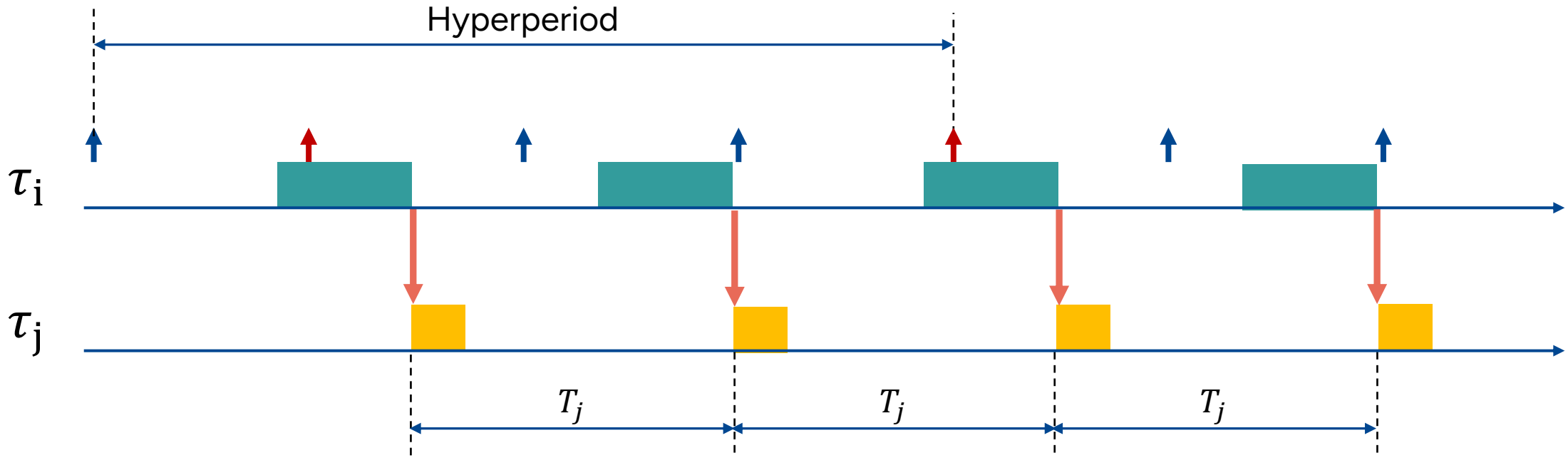
Does this always work?

- No, consider... $E = (\cdots \rightarrow \tau_{\mathrm{i}} \rightarrow \tau_{\mathrm{j}} \rightarrow \cdots)$ $\mathrm{T_j} = \max_{i=1,\ldots,n} T_i = 6$ $Ti = 4$ $C_i = 2{,}5$ $C_{\mathrm{j}} = 1$



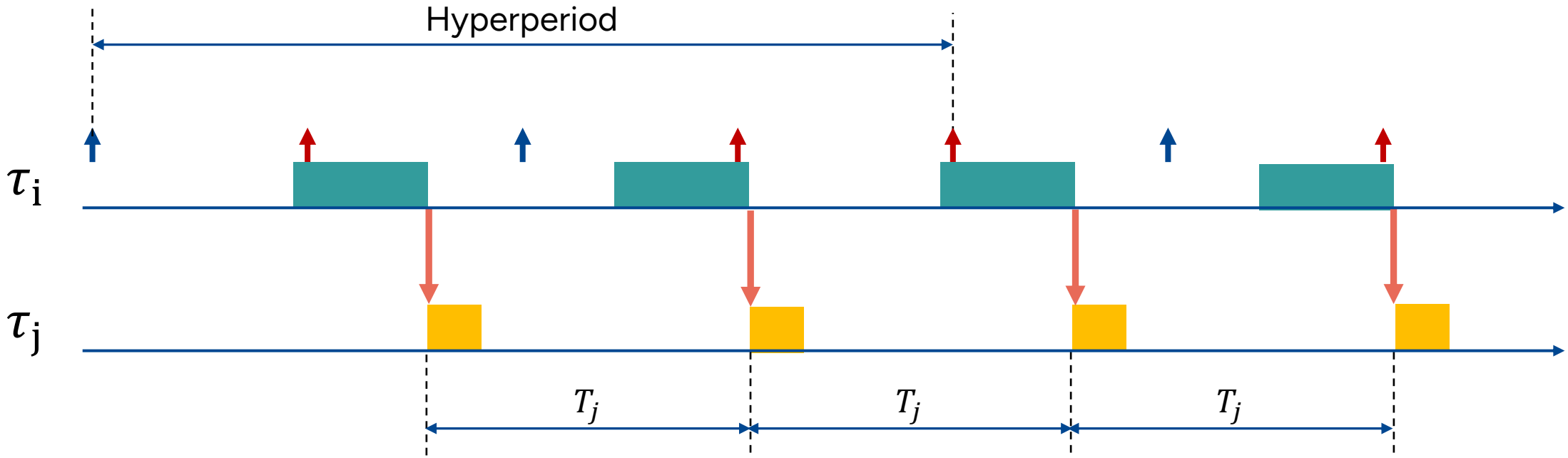
Does this always work?

- No, consider... $E = (\cdots \rightarrow \tau_{\mathrm{i}} \rightarrow \tau_{\mathrm{j}} \rightarrow \cdots)$ $\mathrm{T_j} = \max_{i=1,\ldots,n} T_i = 6$ $Ti = 4$ $C_i = 2{,}5$ $C_{\mathrm{j}} = 1$



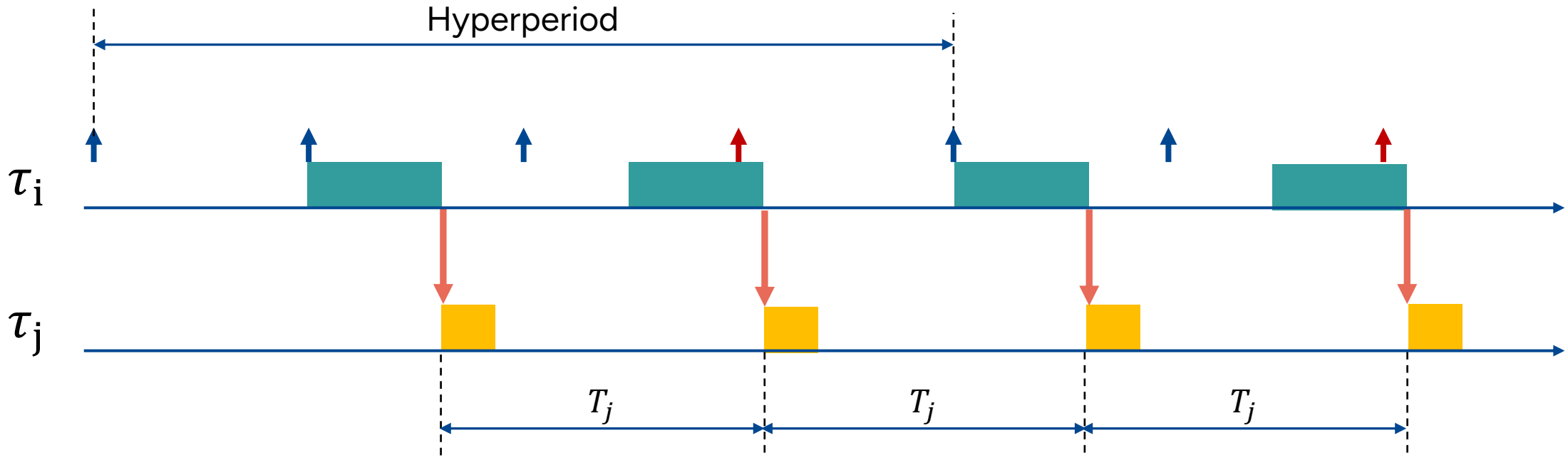
Does this always work?

- No, consider... $E = (\cdots \rightarrow \tau_{\mathrm{i}} \rightarrow \tau_{\mathrm{j}} \rightarrow \cdots)$ $\mathrm{T_j} = \max_{i=1,\ldots,n} T_i = 6$ $Ti = 4$ $C_i = 2{,}5$ $C_{\mathrm{j}} = 1$



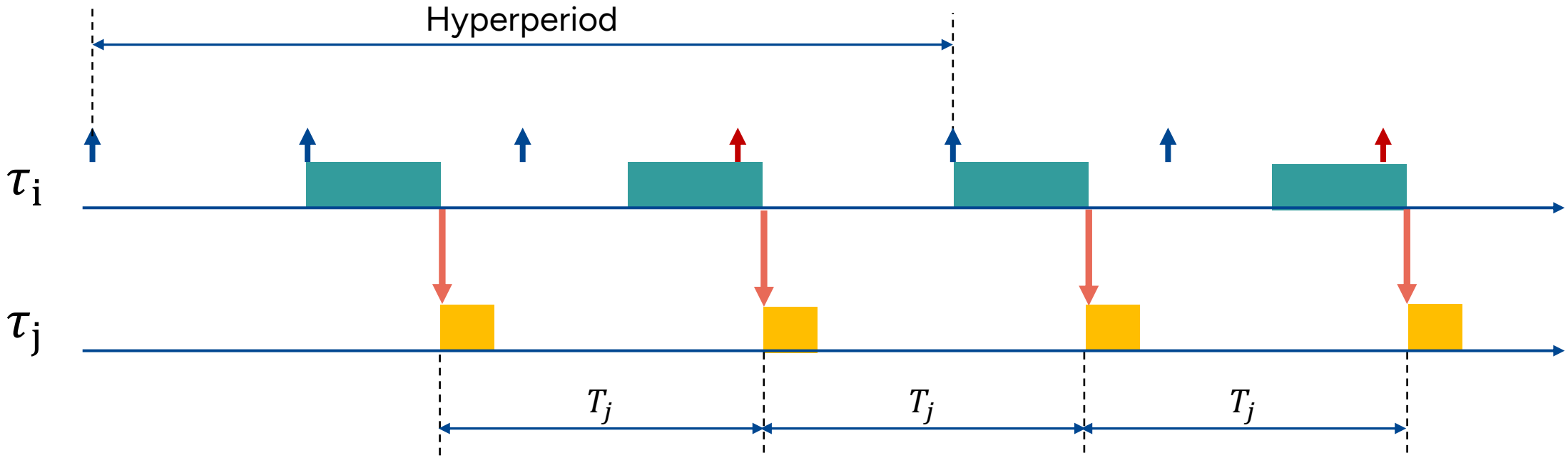
Does this always work?

- No, consider... $E = (\cdots \rightarrow \tau_{\mathrm{i}} \rightarrow \tau_{\mathrm{j}} \rightarrow \cdots)$ $\mathrm{T_j} = \max_{i=1,\ldots,n} T_i = 6$ $Ti = 4$ $C_i = 2{,}5$ $C_{\mathrm{j}} = 1$



Does this always work?

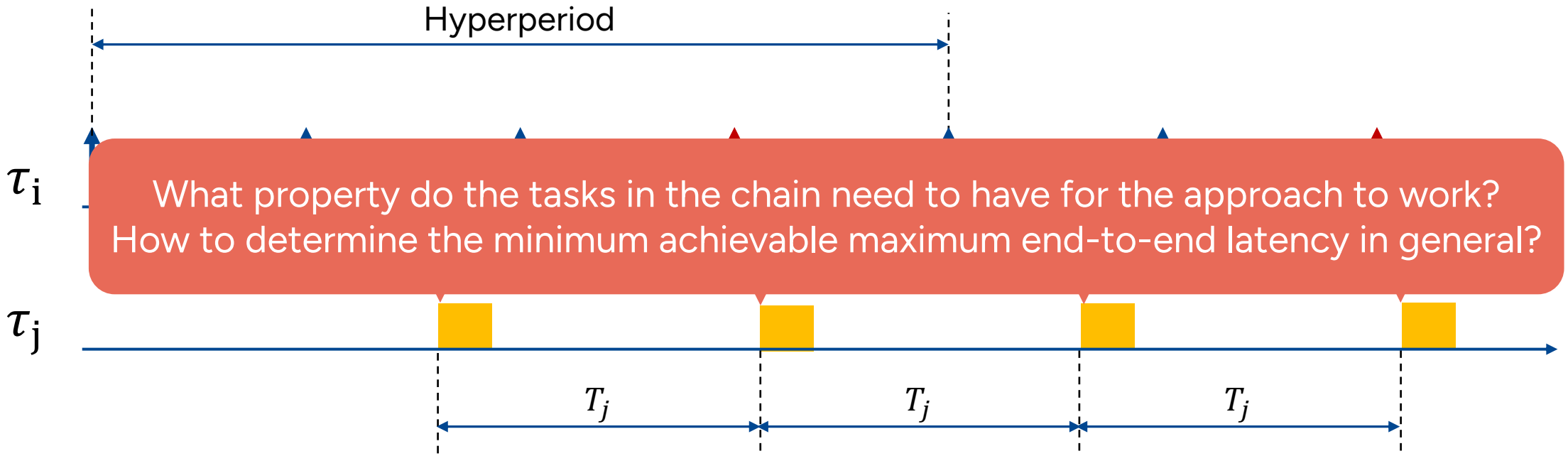
- No, consider... $E = (\cdots \rightarrow \tau_{\mathrm{i}} \rightarrow \tau_{\mathrm{j}} \rightarrow \cdots)$ $\mathrm{T_j} = \max_{i=1,\ldots,n} T_i = 6$ $Ti = 4$ $C_i = 2{,}5$ $C_{\mathrm{j}} = 1$



There is no phase assignment that allows aligning read and write events of the two tasks while the distance between consecutive jobs of τ_j is exactly T_j

Does this always work?

- No, consider... $E = (\cdots \rightarrow \tau_{\mathrm{i}} \rightarrow \tau_{\mathrm{j}} \rightarrow \cdots)$ $\mathrm{T_j} = \max_{i=1,\ldots,n} T_i = 6$ $Ti = 4$ $C_i = 2{,}5$ $C_{\mathrm{j}} = 1$



What property do the tasks in the chain need to have for the approach to work?
How to determine the minimum achievable maximum end-to-end latency in general?

There is no phase assignment that allows aligning read and write events of the two tasks while the distance between consecutive jobs of τ_j is exactly T_j

Conclusions and Future Work

- The minimum and maximum achievable maximum end-to-end latency of cause-effect chains based on only the chain structure can be a valuable insight at early design phases, indicating how difficult it will be to meet the constraint in the final system
- Not much focus on the minimum achievable maximum end-to-end latency
- Knowing the execution sequence leading to the mimimum achievable maximum end-to-end latency can help in designing optimization strategies

Challenges not discussed

- What happens when the system contains multiple chains and they all need to be optimized?
- What happens when tasks are part of multiple chains?

Thank you!
Questions?

Our work is supported by:

Vetenskapsrådet
Swedish Research Council