2IMN20 - Real-Time Systems

The end-to-end timing analysis (putting all together)

Lecturer: Dr. Mitra Nasri

Assistant professor

IRIS Cluster

m.nasri@tue.nl

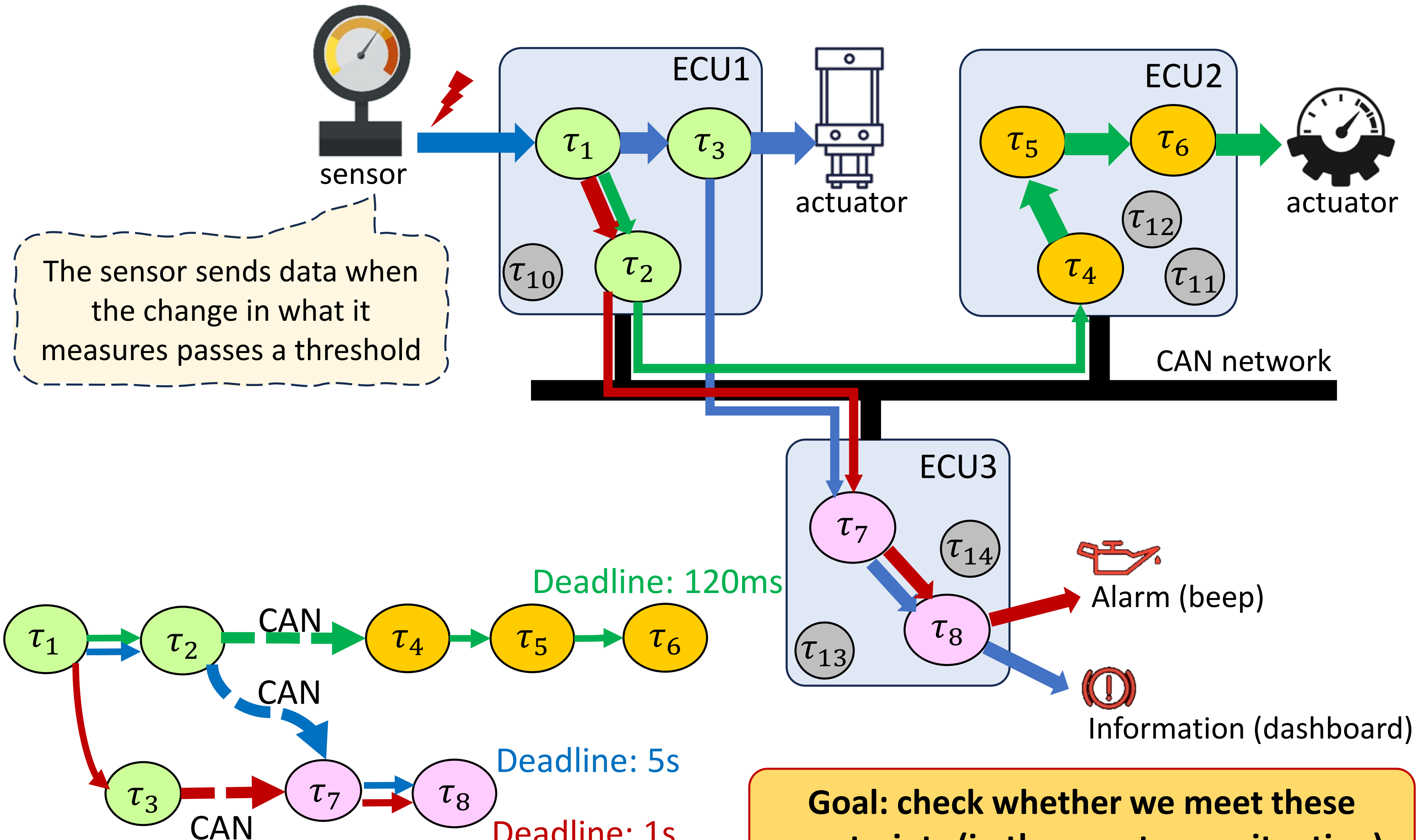
Agenda of the topic

- **Event chains (cause-effect chains)**
  - Response-time analysis (single-core setup)
  - Response-time analysis (distributed setup)

- **Multi-rate task chains**
  - Common end-to-end timing constraints
  - Analyzing timing constraints

Event chains (cause-effect chains)

Event chains (cause-effect chains)



Goal: check whether we meet these constraints (in the worst-case situation)

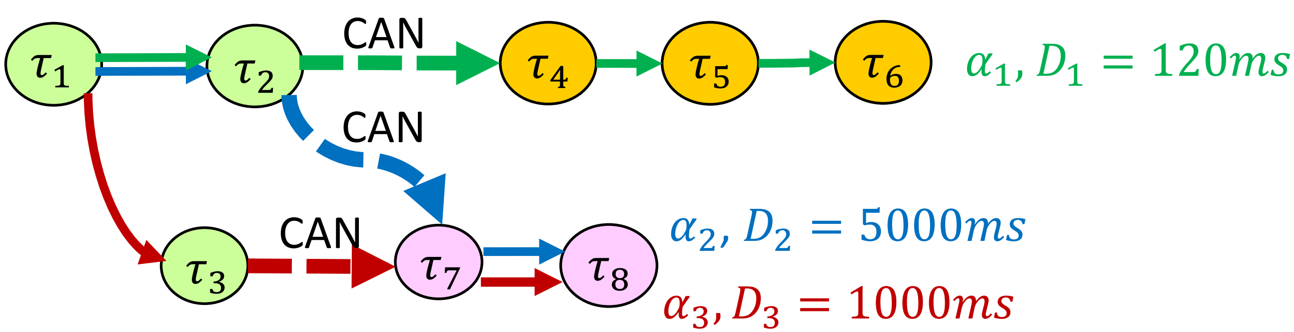
ECU is a computing node in a car

Event chains: workload model

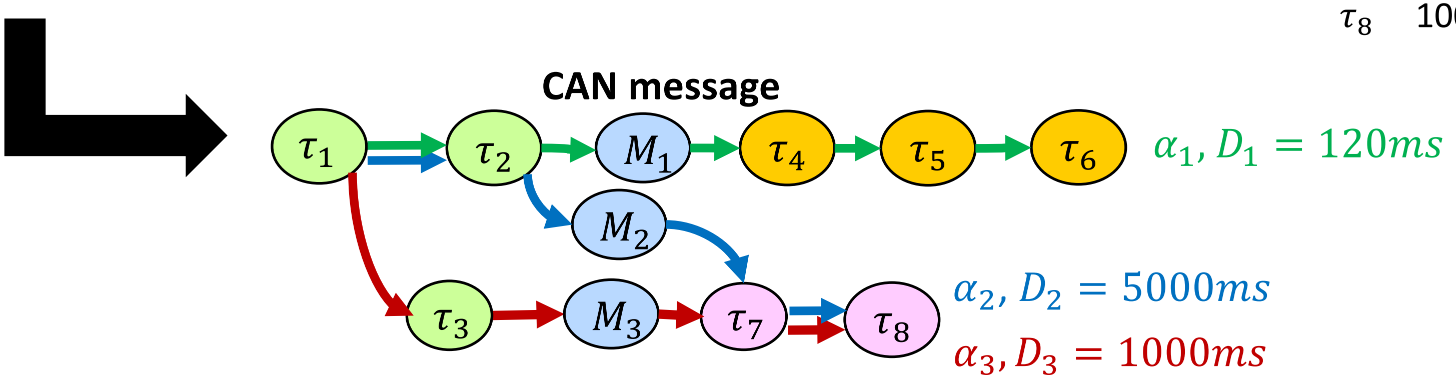
The **task graph** (dependency between task executions) is a Directed Acyclic Graph (DAG).

Time-constrained chains are denoted by $\alpha = \{\alpha_1, \alpha_2, \ldots, \alpha_k\}$. Each chain α_i has a deadline D_i.

If a task sends a message on the network, we need the priority and transmission time of that message and have to add that to the graph as well.



τ_i	C_i	P_i
τ_1	5	90
τ_2	10	80
τ_3	3	78
τ_4	10	91
τ_5	20	90
τ_6	5	80
τ_7	10	10
τ_8	100	20



Event chains: workload model

Task τ_1 (the root) is activated sporadically with a minimum inter-arrival time T.

This should be obtained after analyzing how frequently the event can be generated in the worst case.

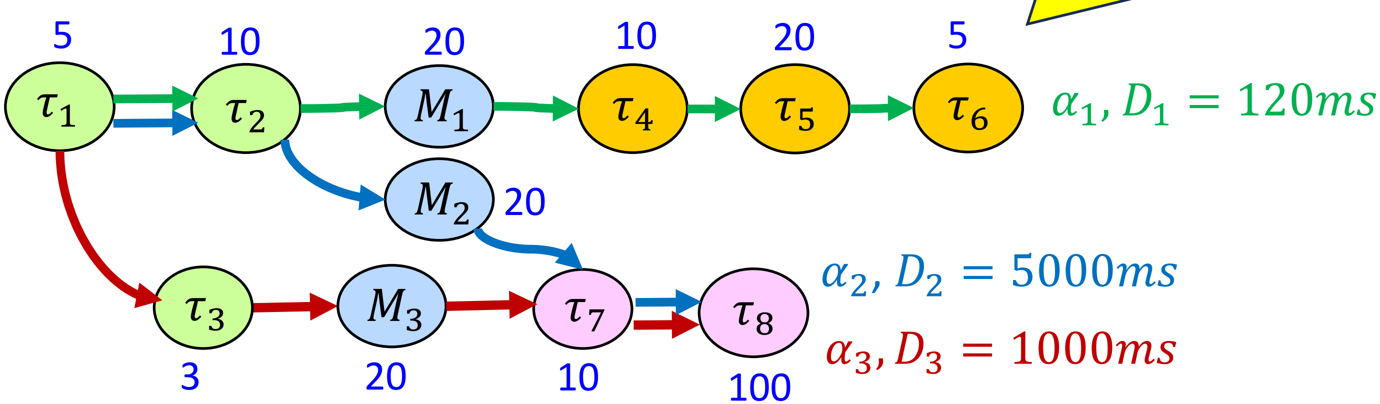
Other tasks are **released** only when their **predecessor** is **completed**. Therefore, they are also sporadic (with the same minimum inter-arrival time T), but with a **release jitter $\boldsymbol{\sigma_i}$** which **depends on the chain.**

The worst-case execution time (WCET) of each task τ_i is denoted by C_i.

Priority of a task τ_i is denoted by P_i (larger values indicate higher priorities).

Do you think will meet our deadlines? Why?

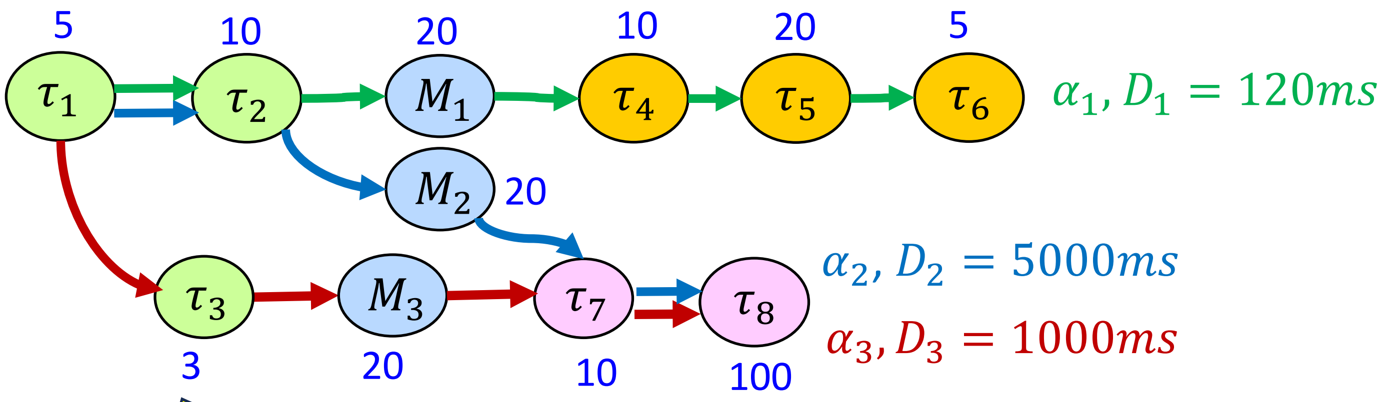
$T = 2000ms$



τ_i	C_i	P_i
τ_1	5	90
τ_2	10	80
τ_3	3	78
τ_4	10	91
τ_5	20	90
τ_6	5	80
τ_7	10	20
τ_8	100	10
M_1	20	High
M_2	20	Medium
M_3	20	Low
…		

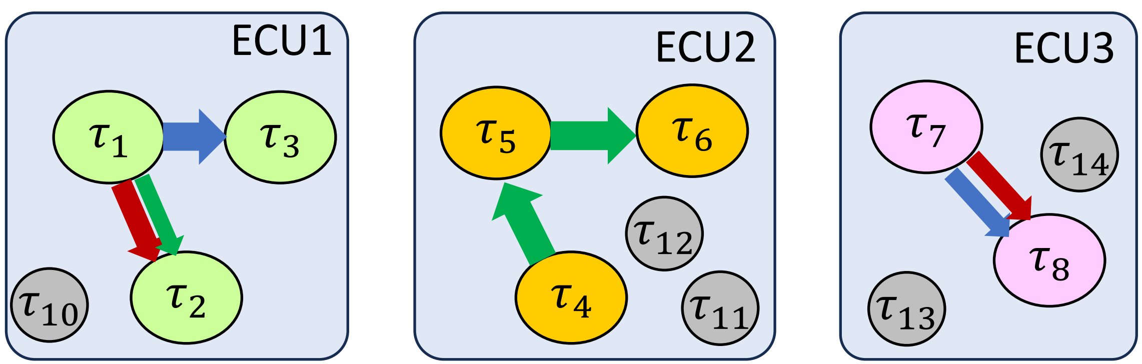
Event chains: A closer look

τ_i	C_i	P_i
$\boldsymbol{\tau_1}$	**5**	**90**
$\boldsymbol{\tau_2}$	**10**	**80**
$\boldsymbol{\tau_3}$	**3**	**78**
τ_4	10	91
τ_5	20	90
τ_6	5	80
τ_7	10	20
τ_8	100	10
M_1	20	High
M_2	20	Medium
M_3	20	Low



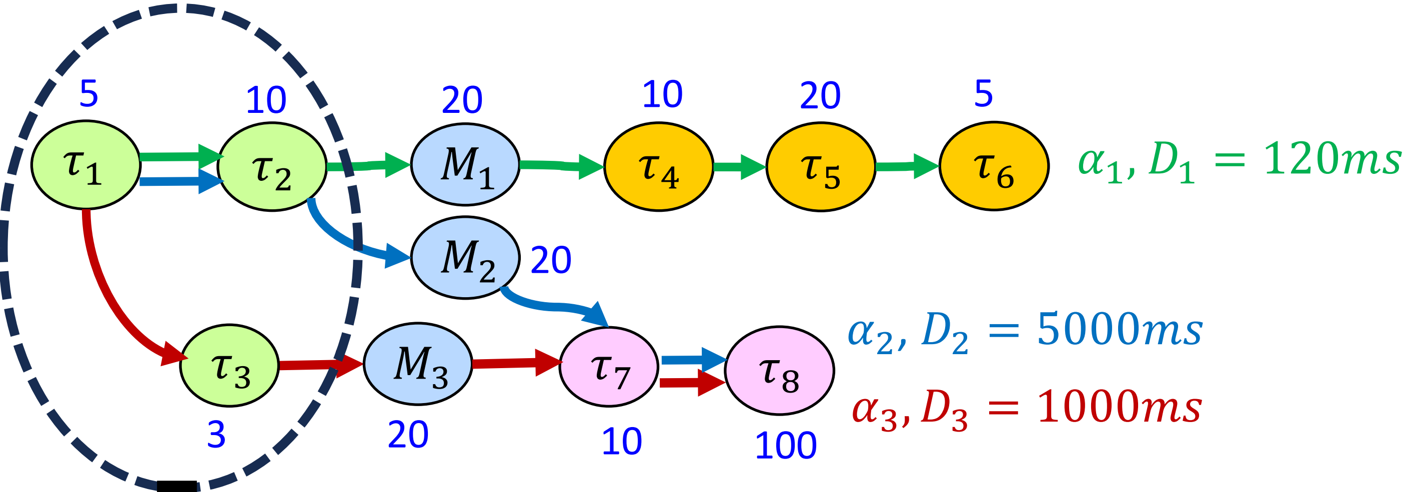
What else do we need before we can find the response-time of each task and each chain?

Activation pattern (period or minimum inter-arrival time), **WCET**, and **priority** of all other tasks and other messages on the network!



CAN network

Event chains: A closer look



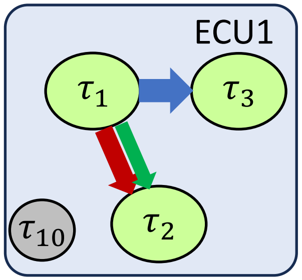
τ_i	C_i	P_i
τ_1	5	90
τ_2	10	80
τ_3	3	78
τ_4	10	91
τ_5	20	90
τ_6	5	80
τ_7	10	20
τ_8	100	10
M_1	20	High
M_2	20	Medium
M_3	20	Low
τ_{10}	8	86 $(T_{10} = 21)$

The assumptions

In each ECU, tasks are scheduled on the same core using partitioned preemptive fixed-priority scheduling

Assume that T is larger than the worst-case response time of the longest chain in the event chain.

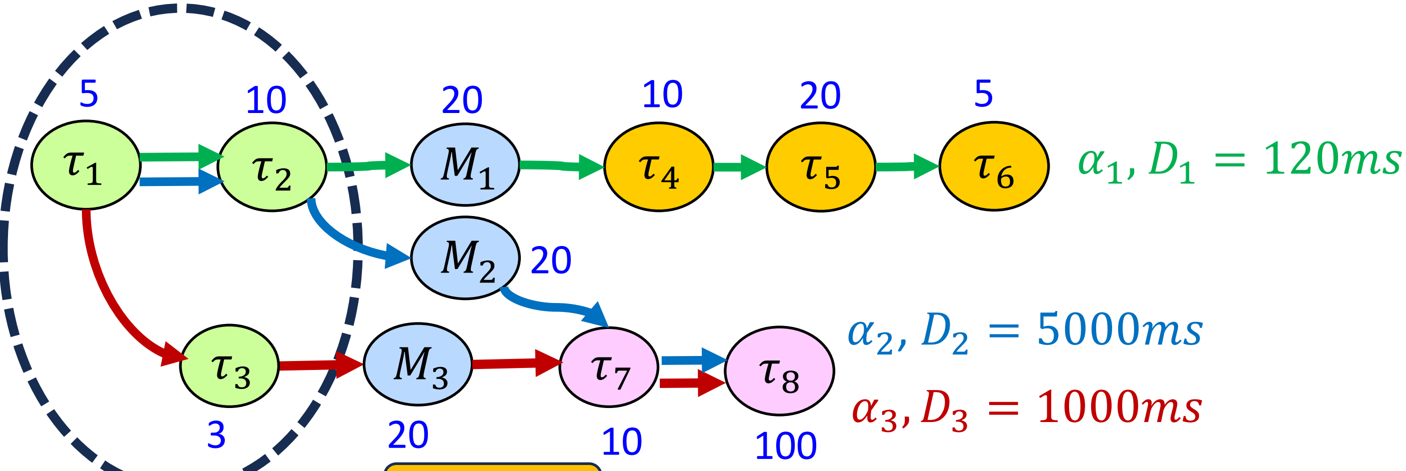
Note that there is no precedence constraint between τ_3 and τ_2, but $P_3 < P_2$.



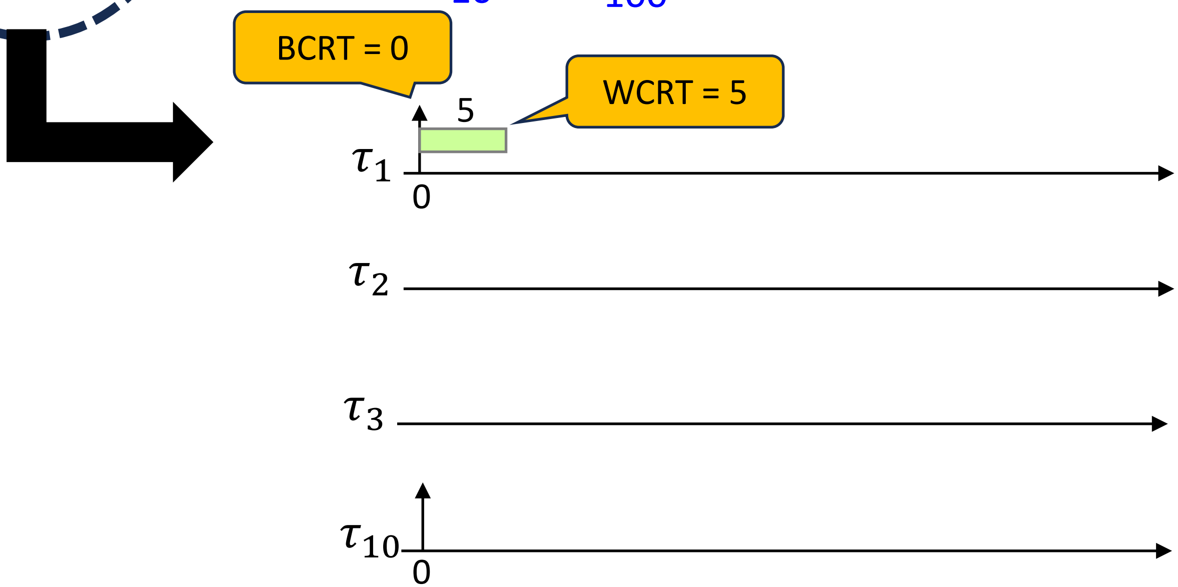
$P_{10} = 86$

Event chains: A closer look

τ_i	C_i	P_i	σ_i
τ_1	5	90	0
τ_2	10	80	5
τ_3	3	78	5
τ_{10}	8	86	$T_{10} = 21$



Assume that the BCET of each task is 0.

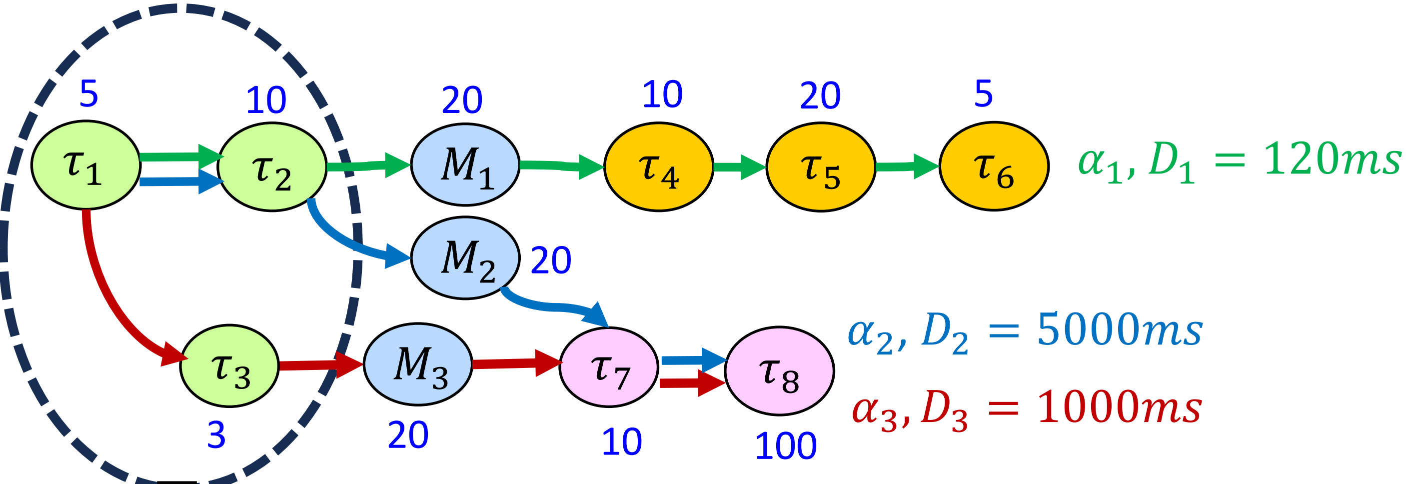


Why can I not just schedule these tasks to see what is the response time?

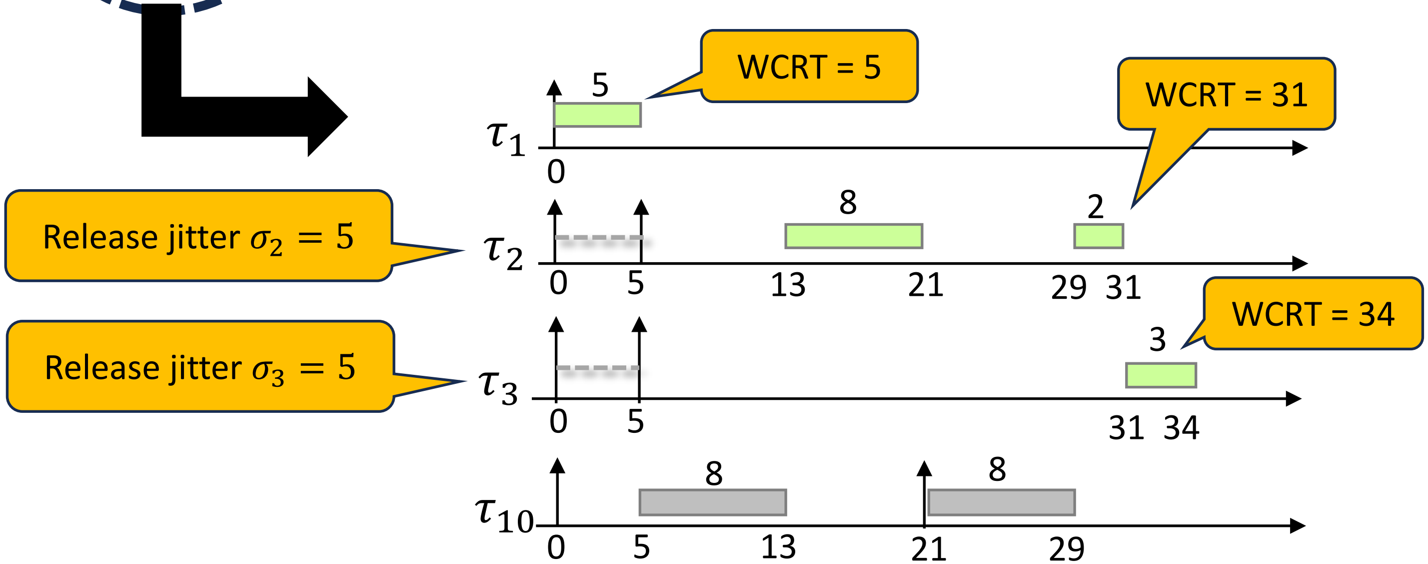
Because the execution time variation of τ_1 becomes like a release jitter for τ_2.

Event chains: A closer look

τ_i	C_i	P_i	σ_i
τ_1	5	90	0
τ_2	10	80	5
τ_3	3	78	5
τ_{10}	8	86	$T_{10} = 21$



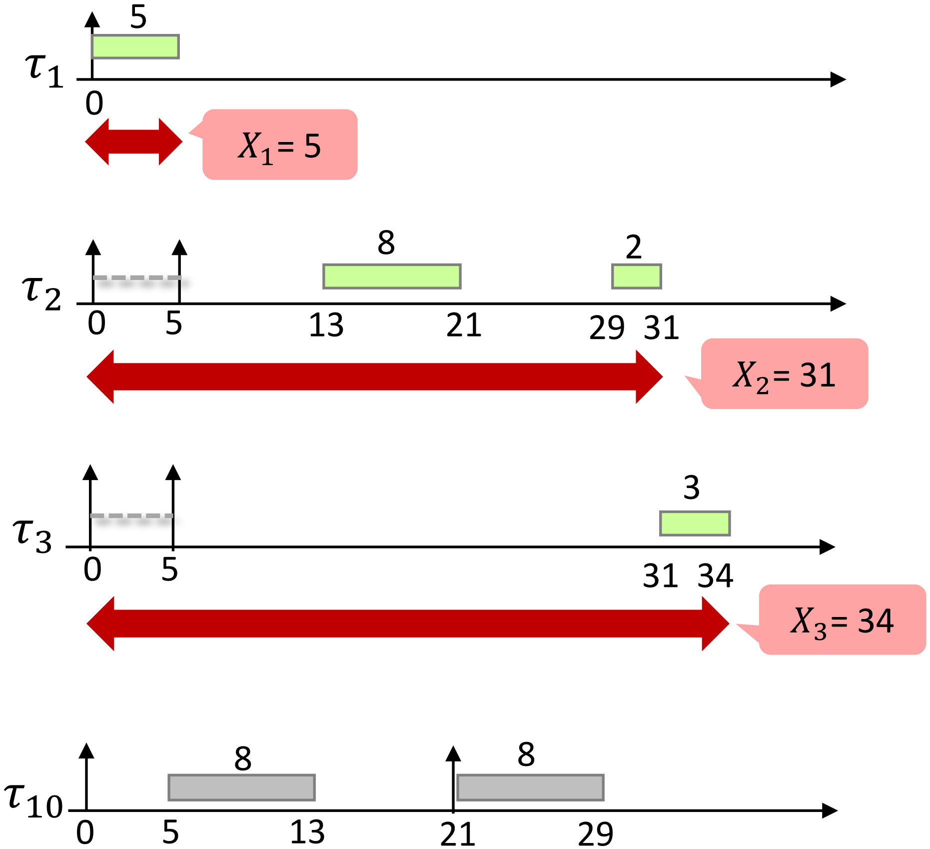
Assume that the BCET of each task is 0.



τ_3 had to wait for τ_2, because $P_3 < P_2$.

Release jitter of a task τ_i in a chain α_x, is the difference between the best-case and worst-case response time of the task before τ_i on the chain α_x

Event chains: A closer look



$hp(\tau_i)$ is the set of tasks with a higher priority than τ_i that are assigned to the same core as τ_i.

Worst-case response-time (WCRT) of task τ_i under FP scheduling is

$$R_i = \sigma_i + X_i$$

where X_i is the **worst-case delay of the task** (when it is interfered the most by other tasks on the same core):

$$X_i^{(0)} = C_i$$

$$X_i^{(k)} = C_i + \sum_{j \in hp(\tau_i)} \left\lceil \frac{X_i^{(k-1)} + \sigma_j}{T_j} \right\rceil \cdot C_j$$

Stop if $X_i^{(k)} = X_i^{(k-1)}$.

X_i is the final answer of the above fixed-point iteration.

Event chains: A closer look

$$X_2^{(0)} = 10$$

$$X_2^{(1)} = 10 + \sum_{j \in hp(\tau_2) = \{\tau_1, \tau_{10}\}} \left\lceil \frac{X_2^{(0)} + \sigma_j}{T_j} \right\rceil \cdot C_j$$

$$= 10 + \left\lceil \frac{10+0}{2000} \right\rceil \cdot 5 + \left\lceil \frac{10+0}{21} \right\rceil \cdot 8 = 23$$

$$X_2^{(2)} = 10 + \left\lceil \frac{23+0}{2000} \right\rceil \cdot 5 + \left\lceil \frac{23+0}{21} \right\rceil \cdot 8 = 31$$

$$X_2^{(3)} = 10 + \left\lceil \frac{31+0}{2000} \right\rceil \cdot 5 + \left\lceil \frac{31+0}{21} \right\rceil \cdot 8 = 31$$

$$X_2 = 31$$

τ_i	C_i	P_i	σ_i	period
τ_1	5	90	0	2000
τ_2	10	80	5	2000
τ_3	3	78	5	2000
τ_{10}	8	86	0	21

$hp(\tau_i)$ is the set of tasks with a higher priority than τ_i that are assigned to the same core as τ_i.

Worst-case response-time (WCRT) of task τ_i under FP scheduling is

$$R_i = \sigma_i + X_i$$

where X_i is the **worst-case delay of the task** (when it is interfered the most by other tasks on the same core):

$$X_i^{(0)} = C_i$$

$$X_i^{(k)} = C_i + \sum_{j \in hp(\tau_i)} \left\lceil \frac{X_i^{(k-1)} + \sigma_j}{T_j} \right\rceil \cdot C_j$$

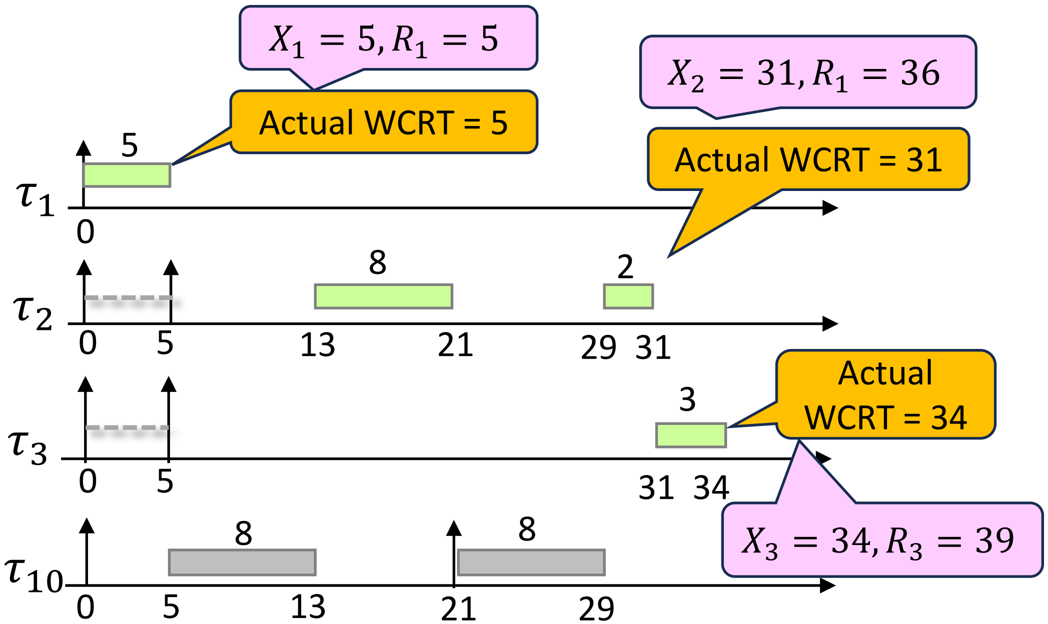
Stop if $X_i^{(k)} = X_i^{(k-1)}$.

X_i is the final answer of the above fixed-point iteration.

Event chains: A closer look + a note

Note: these equations are pessimistic. The analysis is sufficient but not exact.

Because it accounts for the interference by the higher-priority **predecessors** and **successors** of a task more than once.



$$R_i = \sigma_i + X_i$$
$$X_i^{(0)} = C_i$$
$$X_i^{(k)} = C_i + \sum_{j \in hp(\tau_i)} \left\lceil \frac{X_i^{(k-1)} + \sigma_j}{T_j} \right\rceil \cdot C_j$$

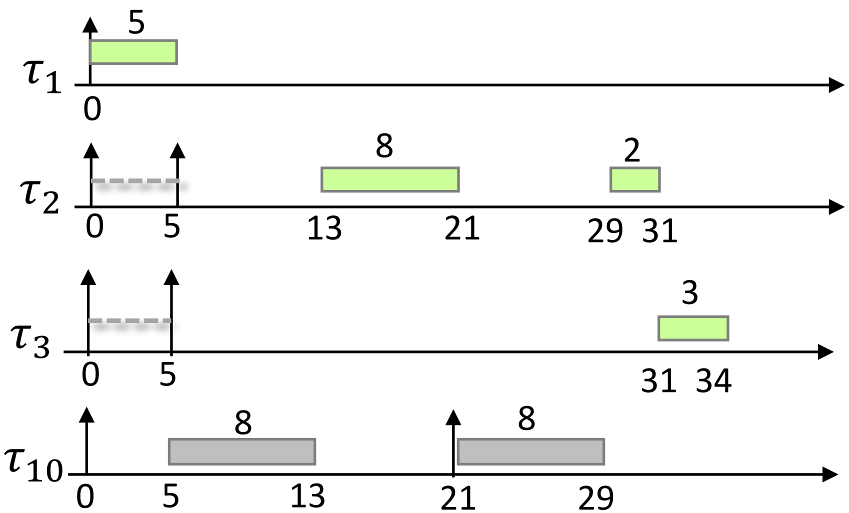
$hp(\tau_i)$ is the set of tasks with a higher priority than τ_i that are assigned to the same core as τ_i.

There are some recent more accurate analyses in the state of the art, but they are very complex, so we omitted them in the course

Event chains: A closer look + a note

What happens if we do not include the high-priority predecessors of τ_i in $hp(\tau_i)$?

τ_i	C_i	P_i	σ_i	period
τ_1	5	90	0	2000
τ_2	10	80	5	2000
τ_3	3	78	5	2000
τ_{10}	8	86	0	21



$$R_i = \sigma_i + X_i$$
$$X_i^{(0)} = C_i$$
$$X_i^{(k)} = C_i + \sum_{j \in hp(\tau_i)} \left\lceil \frac{X_i^{(k-1)} + \sigma_j}{T_j} \right\rceil \cdot C_j$$

$$X_2^{(0)} = 10$$
$$X_2^{(1)} = 10 + \sum_{j \in hp(\tau_2) = \{\tau_{10}\}} \left\lceil \frac{X_2^{(0)} + \sigma_j}{T_j} \right\rceil \cdot C_j$$
$$= 10 + \left\lceil \frac{10+0}{21} \right\rceil \cdot 8 = 18$$

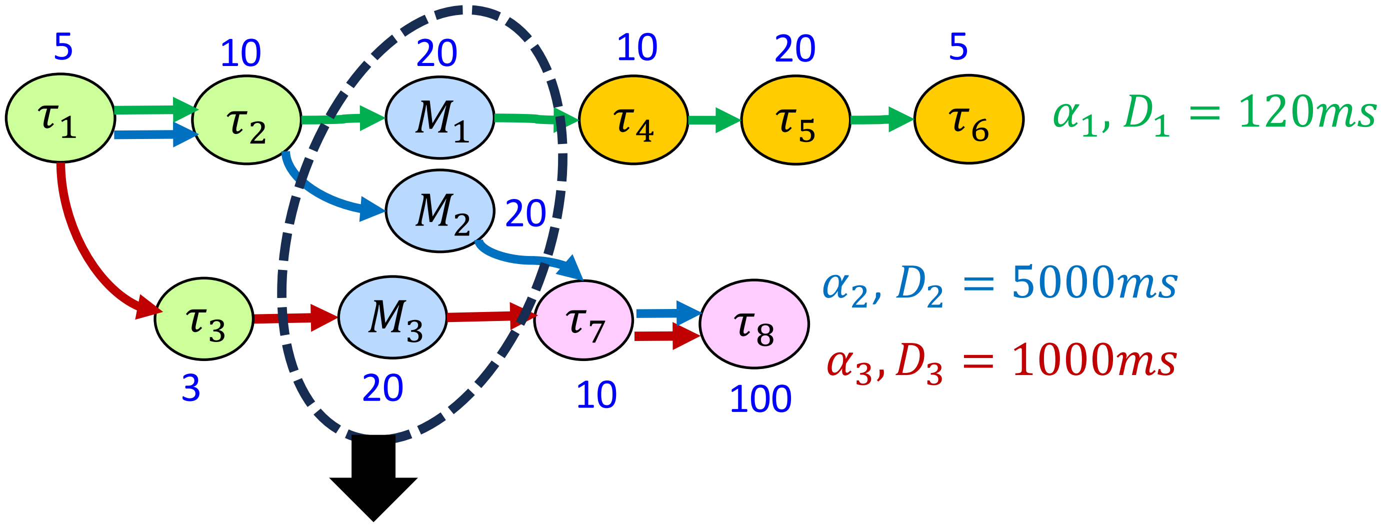
$$X_2^{(2)} = 10 + \left\lceil \frac{18+0}{21} \right\rceil \cdot 8 = 18$$

$$X_2 = 18$$
$$R_2 = 18 + 5 = 23$$

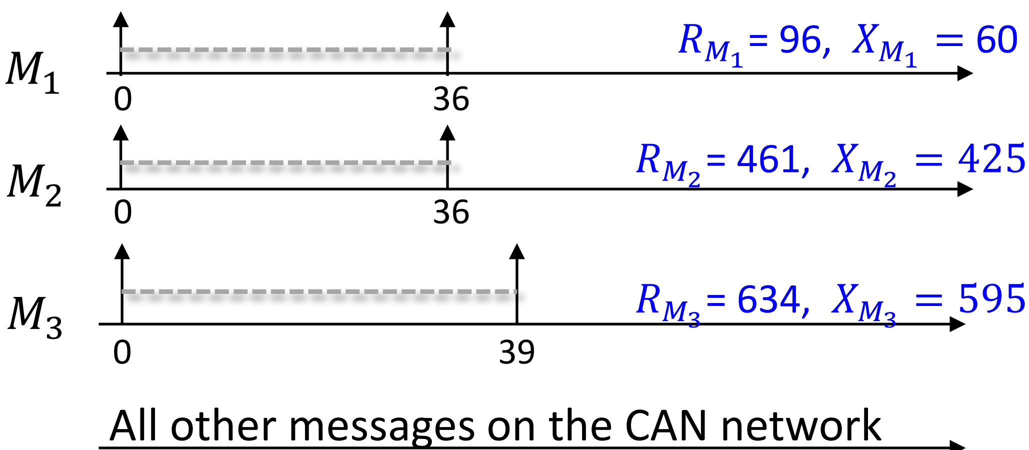
Definitely wrong because the worst-case response-time of τ_2 can be 31!

Event chains: A closer look

τ_i	C_i	P_i	σ_i
M_1	20	High	31
M_2	20	Medium	31
M_3	20	Low	34



τ_i	R_i
τ_1	5
τ_2	36
τ_3	39

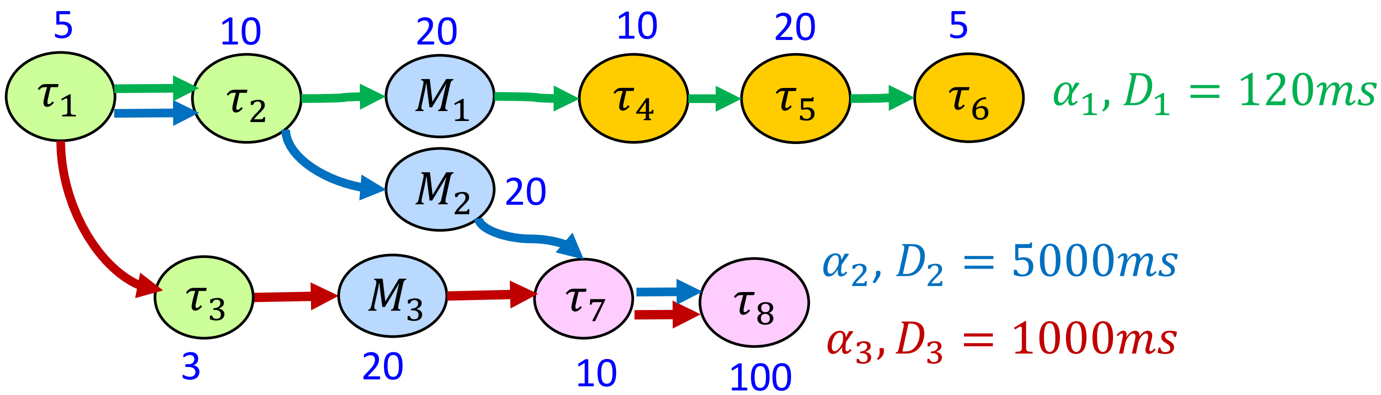


Use the worst-case response-time equations for the CAN.

(don't forget to include **release jitter** and the **blocking factor** in the equations)

Note that priority of messages in a chain might be lower than some other messages not in the chain.

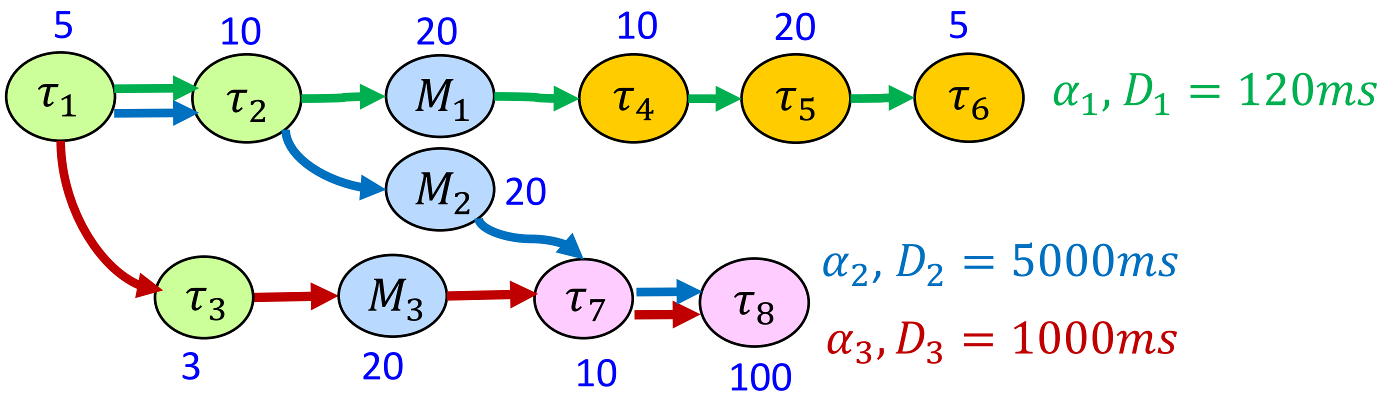
Event chains: A closer look



Worst-case response time of a chain α_i is

$$WCRT\,(\alpha_i) = \sigma_i + \sum_{\tau_j \text{ is the } j^{th} \text{ task of } \alpha_i} X_j$$

Event chains: A closer look



τ_i	R_i	X_i
τ_1	5	5
τ_2	36	31
τ_3	39	34
M_1	96	60
M_2	461	425
M_3	634	595
τ_4	106	10
τ_5	126	30
τ_6	131	35
$\tau_7(\alpha_2)$	479	18
$\tau_8(\alpha_2)$	619	140
$\tau_7(\alpha_3)$	652	18
$\tau_8(\alpha_3)$	792	140

Worst-case response time of a chain α_i is

$$WCRT\ (\alpha_i) = \sigma_i + \sum_{\tau_j \text{ is the } j^{th} \text{ task of } \alpha_i} X_j$$

$$WCRT\ (\alpha_1) = 0 + 5 + 31 + 60 + 10 + 30 + 35 = 171$$ ✗ **Deadline miss**

$$WCRT\ (\alpha_2) = 0 + 5 + 31 + 425 + 18 + 140 = 619$$

$$WCRT\ (\alpha_3) = 0 + 5 + 34 + 595 + 18 + 140 = 792$$

Event chains: more details (check out at home)

$X_4^{(0)} = 10$

$$X_4^{(1)} = 10 + \sum_{j \in hp(\tau_4)=\{\,\}} \left\lceil \frac{X_4^{(0)} + \sigma_j}{T_j} \right\rceil \cdot C_j$$

$$= 10$$

$X_4 = 10$

$R_4 = 96 + 10 = 106 \quad \rightarrow \quad \sigma_5 = 106$

$X_5^{(0)} = 20$

$$X_4^{(1)} = 20 + \sum_{j \in hp(\tau_5)=\{\,\tau_4\}} \left\lceil \frac{X_5^{(0)} + \sigma_j}{T_j} \right\rceil \cdot C_j$$

$$= 20 + \left\lceil \frac{20+96}{2000} \right\rceil \cdot 10 = 30$$

$$X_5^{(2)} = 20 + \left\lceil \frac{30 + 96}{2000} \right\rceil \cdot 10 = 30$$

$X_5 = 30$

$R_5 = 106 + 30 = 136 \quad \rightarrow \quad \sigma_6 = 136$

τ_i	C_i	P_i	σ_i	period
τ_4	10	91	96	2000
τ_5	20	90	106	2000
τ_6	5	80	136	2000
τ_{11}	100	10	0	500
τ_{12}	12	8	0	30

$hp(\tau_i)$ is the set of tasks with a higher priority than τ_i that are assigned to the same core as τ_i.

$X_6^{(0)} = 5$

$$X_6^{(1)} = 5 + \sum_{j \in hp(\tau_6)=\{\,\tau_4,\tau_5\}} \left\lceil \frac{X_6^{(0)} + \sigma_j}{T_j} \right\rceil \cdot C_j$$

$$= 5 + \left\lceil \frac{5+96}{2000} \right\rceil \cdot 10 + \left\lceil \frac{5+106}{2000} \right\rceil \cdot 20 = 35$$

$$X_5^{(2)} = 20 + \left\lceil \frac{35 + 96}{2000} \right\rceil \cdot 10 + \left\lceil \frac{35 + 106}{2000} \right\rceil \cdot 20 = 35$$

$X_5 = 35$

$R_5 = 136 + 35 = 171$

Event chains: more details (check out at home)

$$X_7^{(0)} = 10$$

$$X_7^{(1)} = 10 + \sum_{j \in hp(\tau_7)=\{\tau_{14}\}} \left\lceil \frac{X_7^{(0)} + \sigma_j}{T_j} \right\rceil \cdot C_j$$

$$= 10 + \left\lceil \frac{10}{50} \right\rceil \cdot 8 = 18$$

$$X_7^{(2)} = 10 + \left\lceil \frac{18}{50} \right\rceil \cdot 8 = 18$$

$$X_7 = 18$$

$$R_7 = 18 + 461 = 479 \quad \rightarrow \quad \sigma_8 = 479$$

τ_i	C_i	P_i	σ_i	period
τ_7	10	20	461	2000
τ_8	100	10		2000
τ_{13}	6	15	0	200
τ_{14}	8	30	0	50

$hp(\tau_i)$ is the set of tasks with a higher priority than τ_i that are assigned to the same core as τ_i.

$$X_8^{(0)} = 100$$

$$X_8^{(1)} = 100 + \sum_{j \in hp(\tau_8)=\{\tau_7, \tau_{13}, \tau_{14}\}} \left\lceil \frac{X_i^{(k-1)} + \sigma_j}{T_j} \right\rceil \cdot C_j$$

$$= 100 + \left\lceil \frac{10+461}{2000} \right\rceil \cdot 10 + \left\lceil \frac{10}{200} \right\rceil \cdot 6 + \left\lceil \frac{10}{50} \right\rceil \cdot 8 = 124$$

$$X_8^{(2)} = 100 + \left\lceil \frac{124 + 461}{2000} \right\rceil 10 + \left\lceil \frac{124}{200} \right\rceil 6 + \left\lceil \frac{124}{50} \right\rceil 8 = 140$$

$$X_8^{(3)} = 100 + \left\lceil \frac{140 + 461}{2000} \right\rceil 10 + \left\lceil \frac{140}{200} \right\rceil 6 + \left\lceil \frac{140}{50} \right\rceil 8 = 140$$

$$X_8 = 140$$

$$R_8 = 479 + 140 = 619$$

Summary of the response-time analysis of event chains

Obtain the workload model. It includes **arrival pattern, execution time, and priority** of **every task or message in the system** that runs on the **same ECU or network**.

For each chain α_x

Obtain the BCRT and WCRT and the best-case and worst-case delay of the tasks on the chain in the order they appear in the chain.

- The difference between the BCRT and WCRT of the i^{th} task in the chain is the **release jitter** of the $(i+1)^{th}$ task in the chain.
- Tasks may have different release jitter when considering different chains. It may change their final response time.
- When using response-time equations, don't forget to **consider all higher-priority tasks on the same core** (and do not include those that are not on the same core)

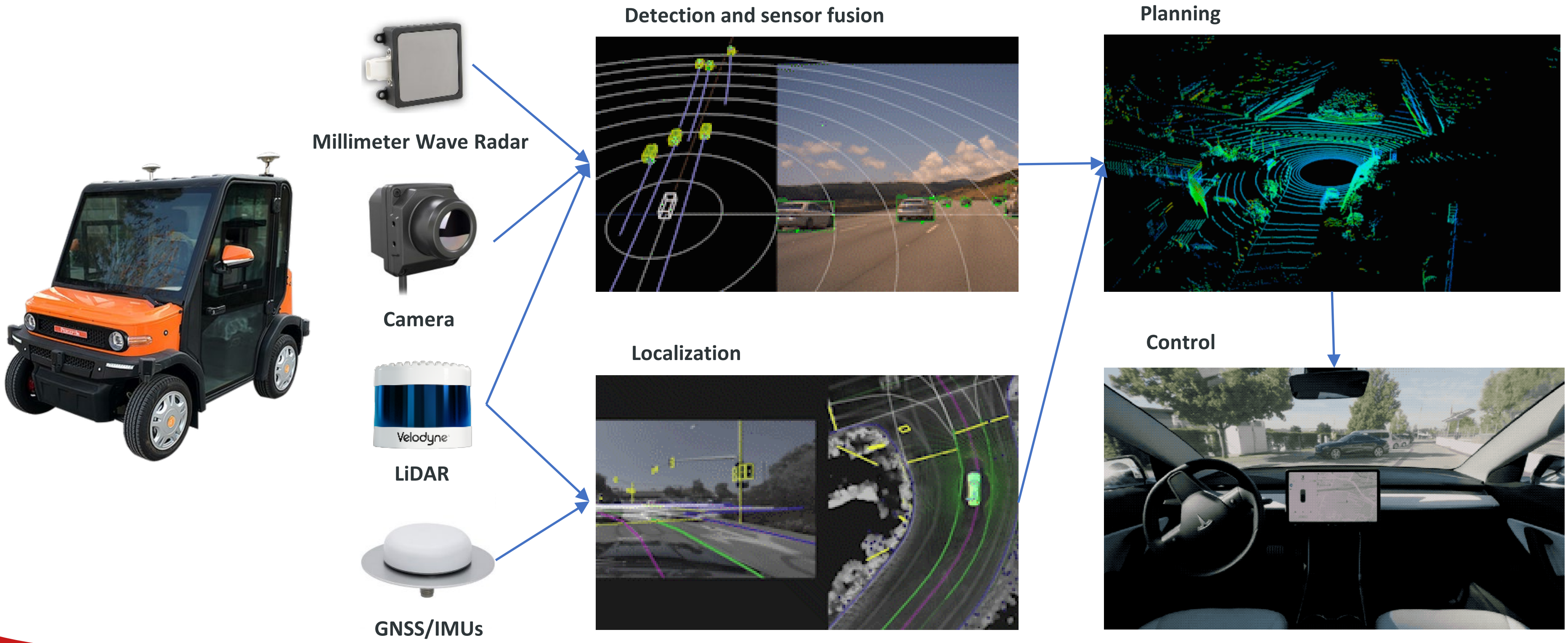
Agenda of the topic

Part a: end-to-end timing analysis

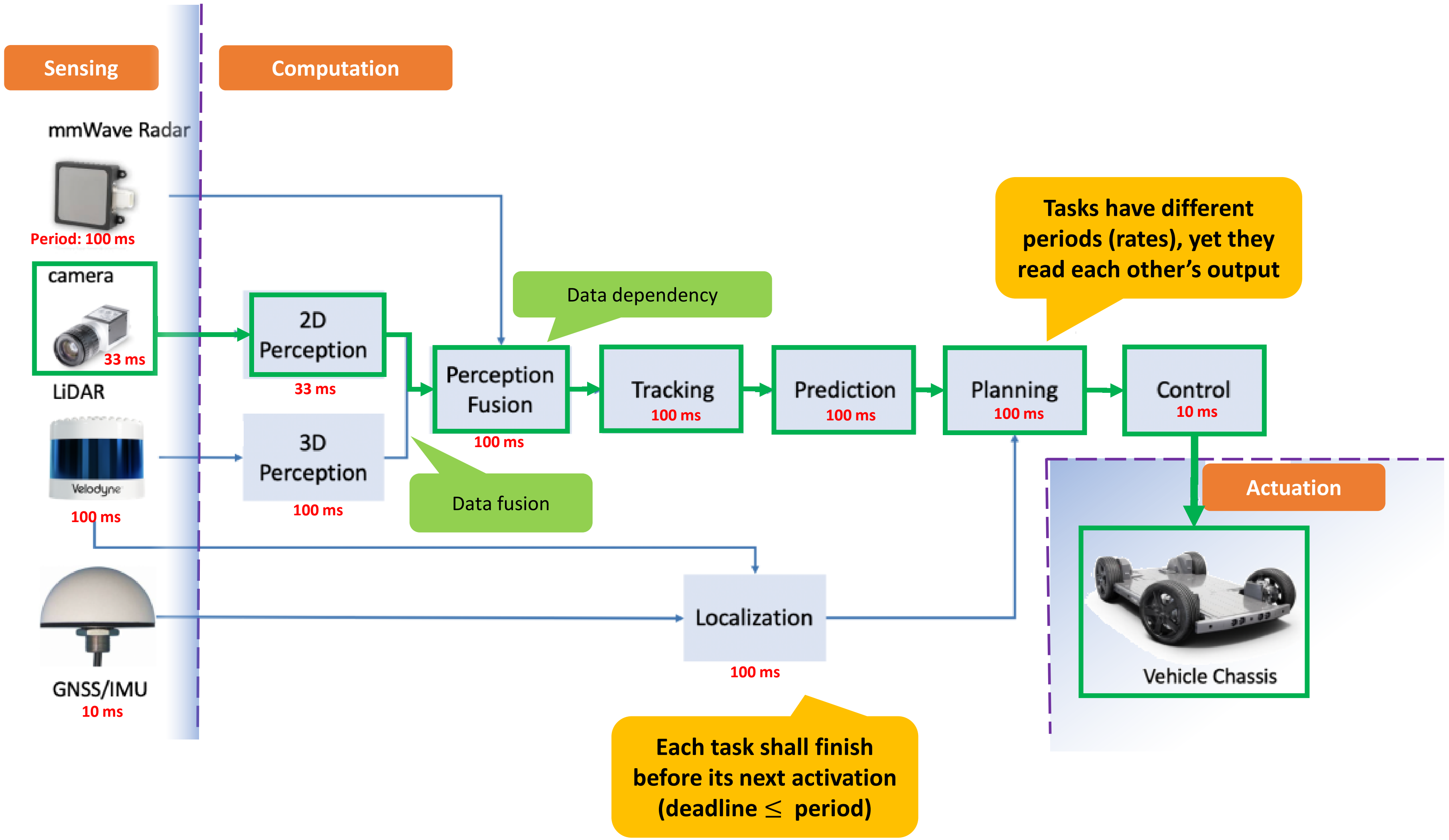
1. Event chains (cause-effect chains)
    - Response-time analysis (single-core setup)
    - Response-time analysis (distributed setup)

2. **Multi-rate task graphs**
    - Common end-to-end timing constraints
    - Analyzing end-to-end response-time

Time-triggered task chains (multi-rate periodic tasks)



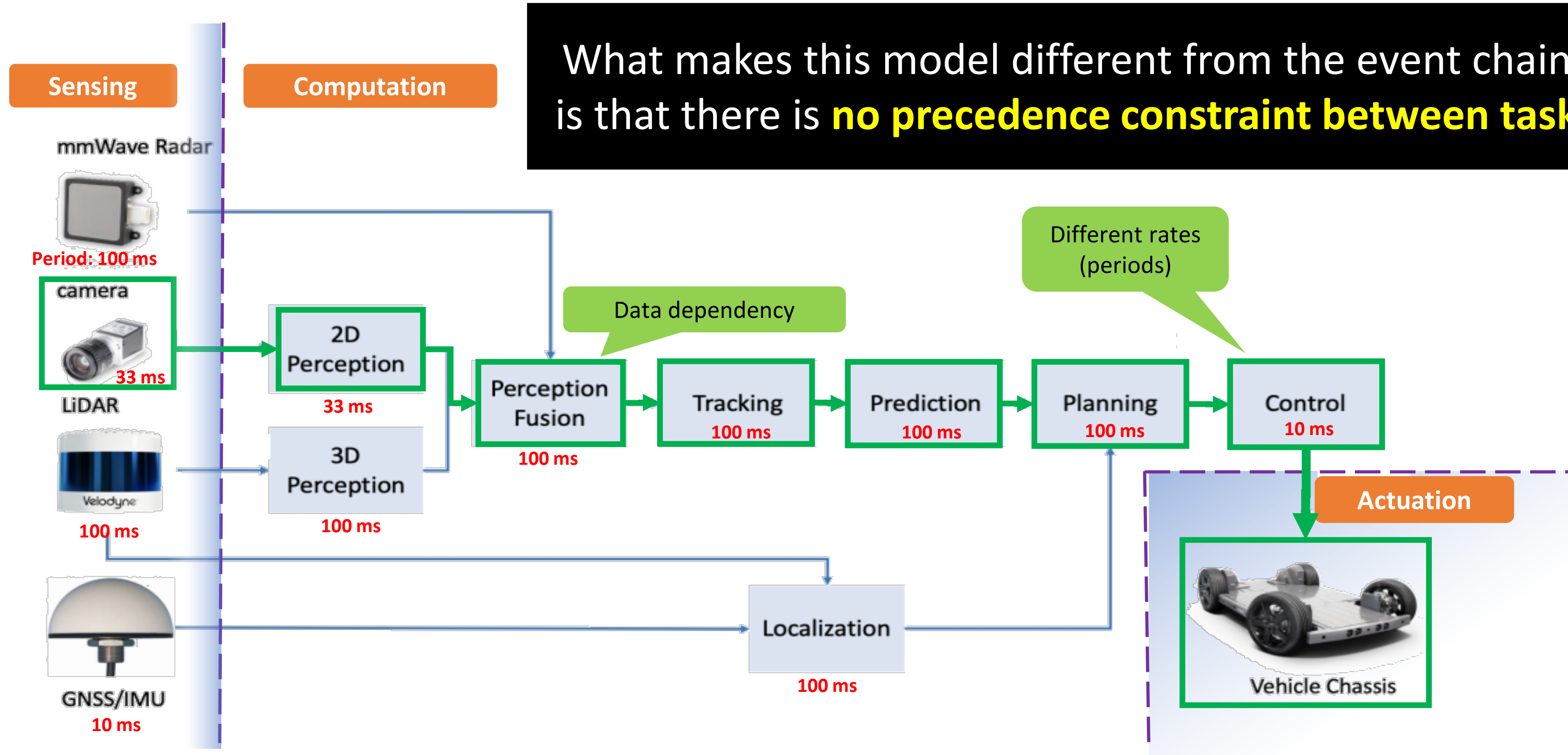
Data-dependent multi-rate tasks



S. Liu, B. Yu, N. Guan, Z. Dong, and B. Akesson. 2021. RTSS 2021 Industry Session. http://2021.rtss.org/industry-session/

Multi-rate task chain

What makes this model different from the event chains is that there is **no precedence constraint between tasks**



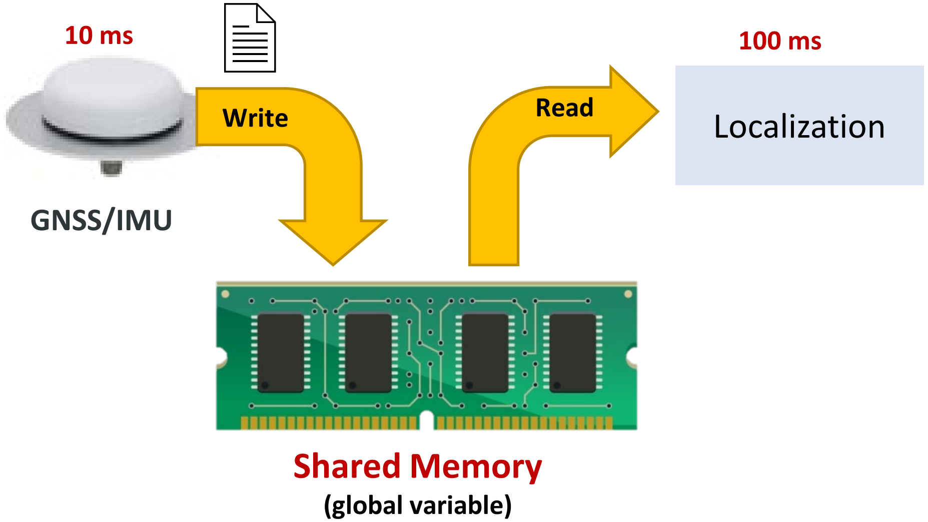
Task chain:

A sequence of tasks in which every two consecutive tasks are data-dependent.

- Often from sensing to actuation
- Often contains tasks that together fulfill **a certain functionality**
- **Does not force** the **data-producer** to execute before **data-consumer**

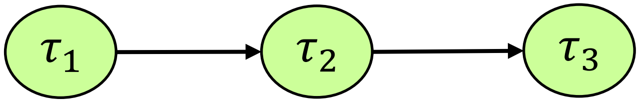
Multi-rate task chain

- Tasks are executed **independently** and **produce their output at their own rate**
- **Data consumer tasks** use the **most recent data** produced by their predecessor task (**data producer**)
- Examples: **publisher-subscriber model in ROS [1]**, **read-execute-write in AUTOSAR [2]**
- Provide flexibility in terms of system design, scheduling, and sharing of information [3]

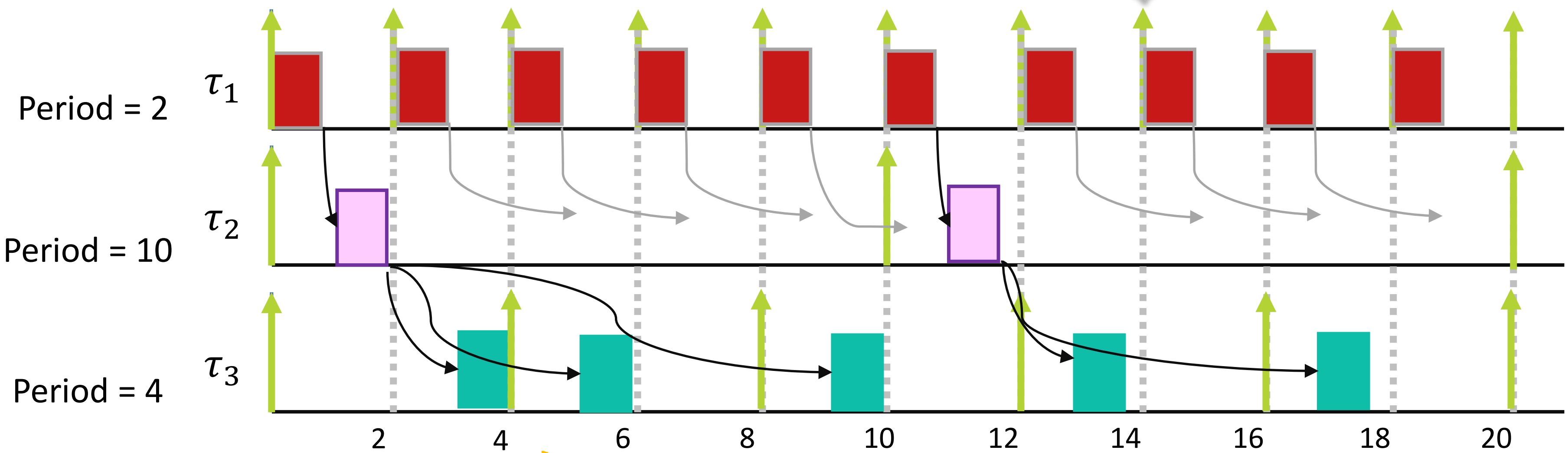


[1] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, A. Y Ng, et al. “ROS: an open-source Robot Operating System”. In ICRA workshop on open-source software, 2019.
[2] AUTOSAR - specification of timing extensions. https://www.autosar.org/
[3] H. Choi, M. Karimi and H. Kim, "Chain-Based Fixed-Priority Scheduling of Loosely-Dependent Tasks," In International Conference on Computer Design (ICCD), 2020.

Multi-rate task graphs



Oversampling happens if $T_1 < T_2$.

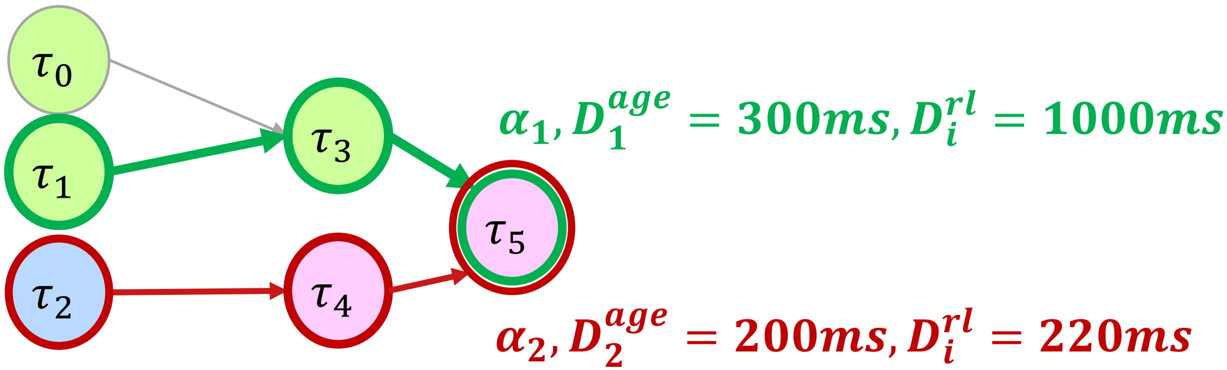


Under-sampling happens if $T_2 > T_3$.

T_i is the period of task τ_i

Assumptions

Workload model: task graph



- Tasks are periodic without release jitter.
- ECUs (computing nodes) are synchronized.
- There is no enforcement of precedence constraint.
- BCET and WCET of the tasks are known.
- Tasks are partitioned on different cores.
- Scheduling policy is preemptive fixed-priority scheduling.
- For each chain α_i, the maximum tolerable **data age** is given by D_i^{age}, and maximum tolerable **reaction latency** is given by D_i^{rl}.
- Some data might need to be shared on the network. In that case, they are sent via periodic messages (same period as the data producer) on the network.

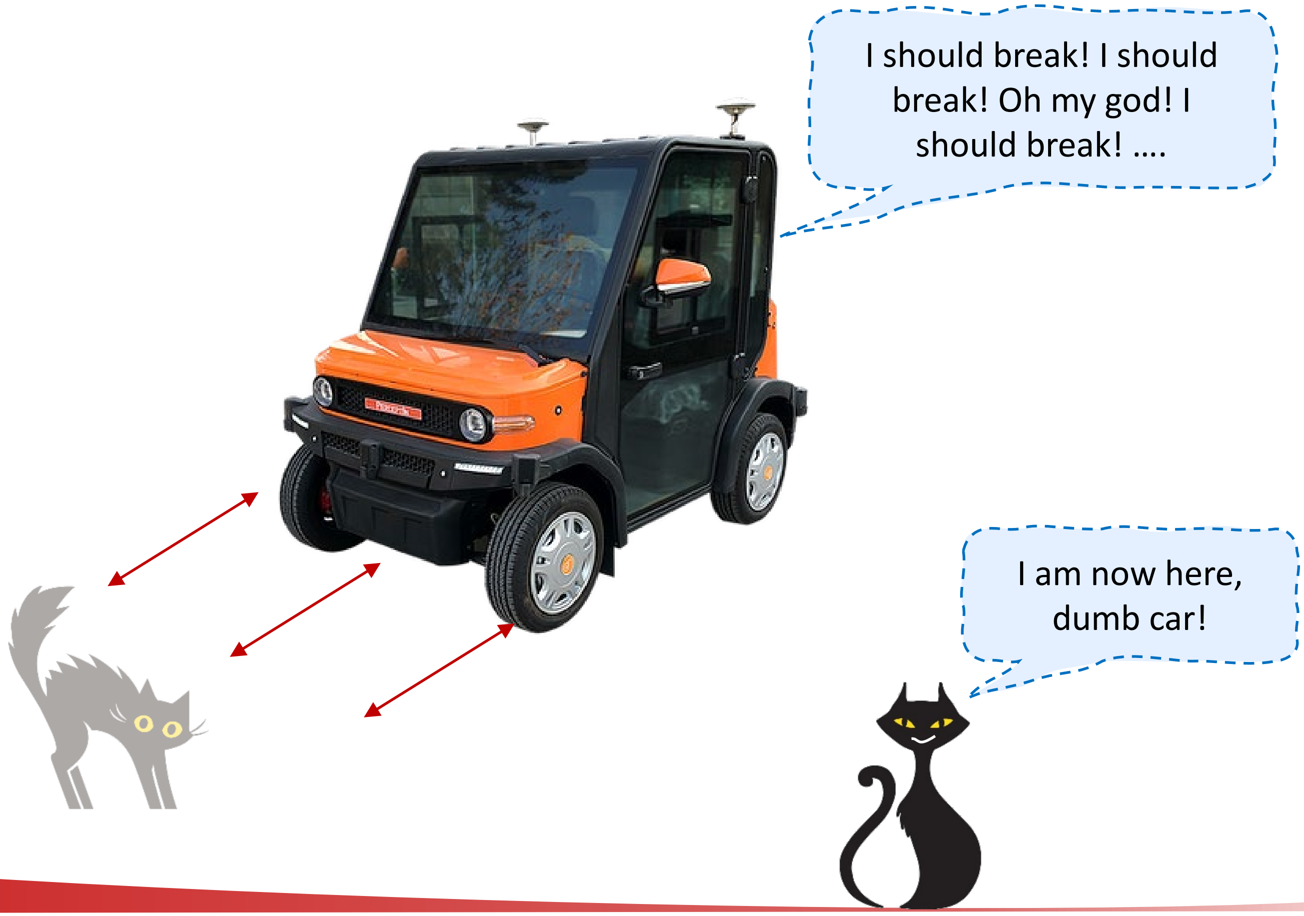
Types of end-to-end latency constraints

Data age | **Reaction latency**

Data age (data freshness)

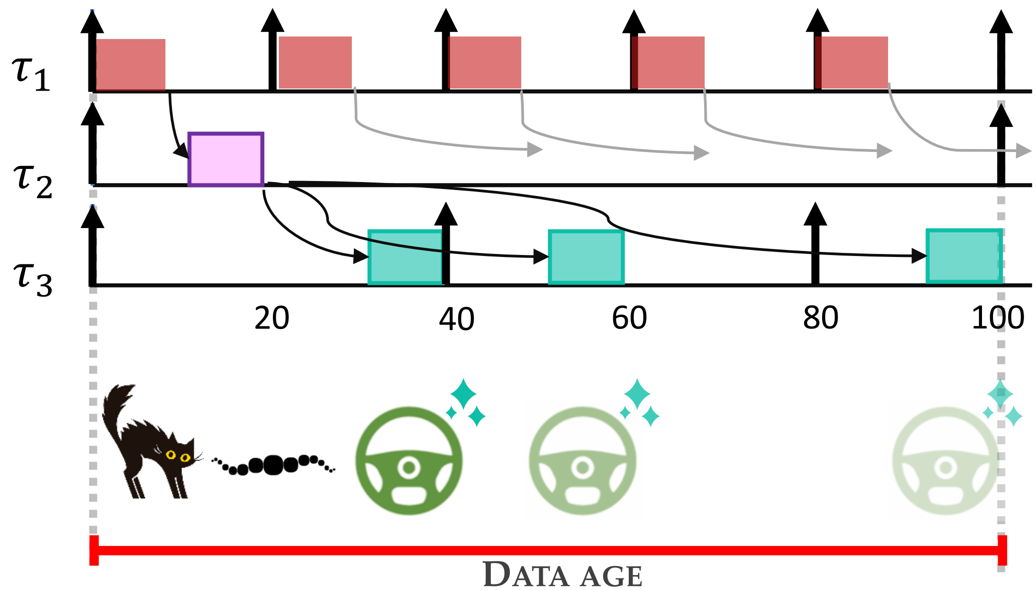
We want data age to be small.

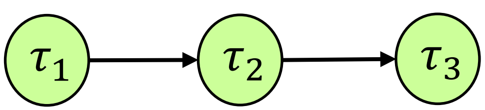
Data age is the maximum amount of time during which an **[old] input data** still affects the output of the system.



Data age (data freshness)

Data age is the maximum amount of time during which an **[old] input data** still affects the output of the system.

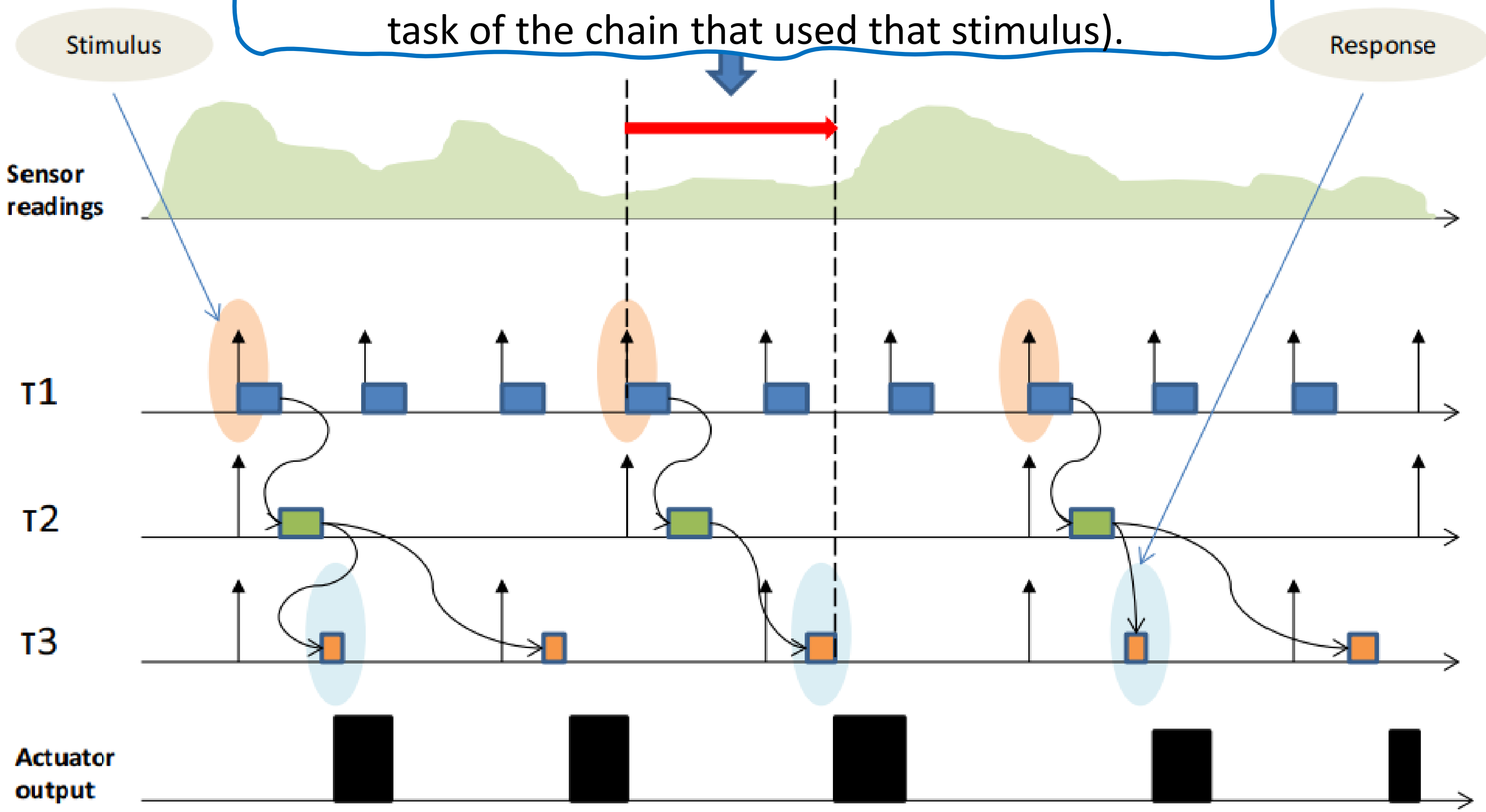




We assume the data is ready at the sensor (therefore, we calculate the data age from the arrival time of the first task in the chain).

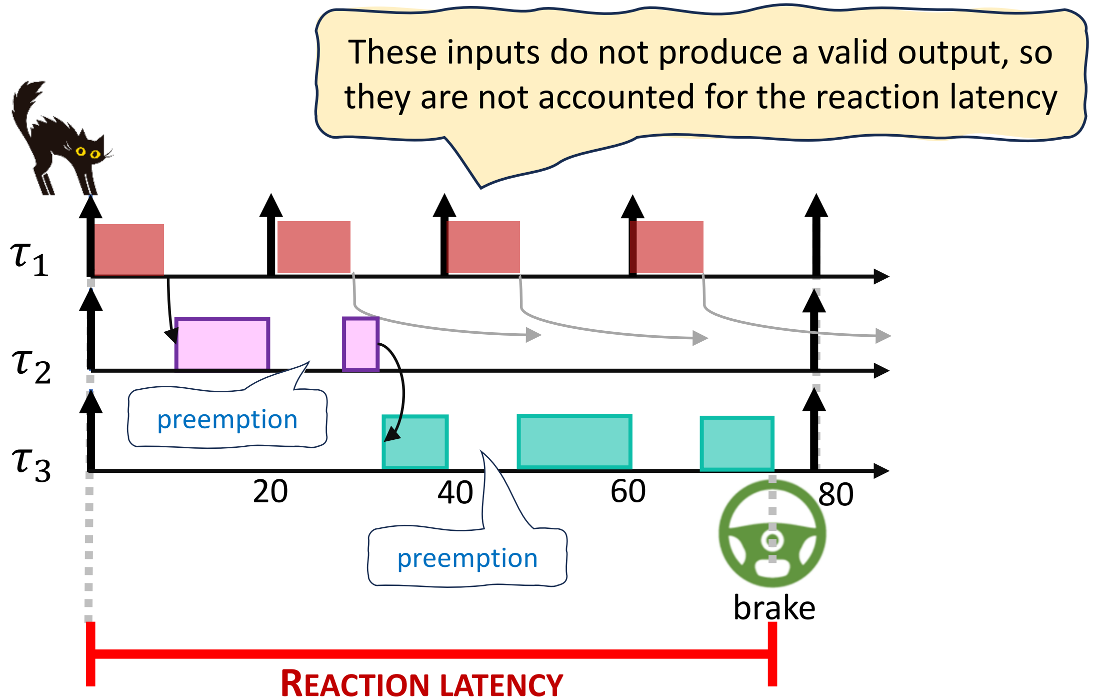
Reaction latency

Reaction latency is the largest delay between a **stimulus and its first response** (produced by the last task of the chain that used that stimulus).

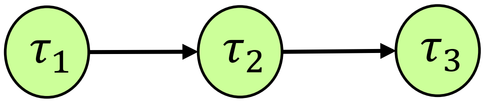


Jakaria Abdullah, Gaoyang Dai, and Wang Yi, Worst-Case Cause-Effect Reaction Latency in Systems with Non-Blocking Communication, 2019.

Reaction latency: an example

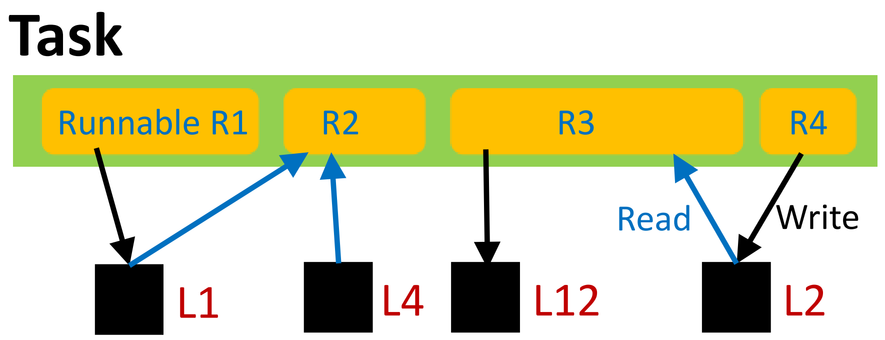


We assume the data is ready at the sensor (therefore, we calculate the reaction latency from the arrival time of the first task in the chain).



When to read or write?

Let's have a look at AUTOSAR (a standard widely used in automotive)



- Runnables are the smallest executable units.
- They are grouped into tasks.

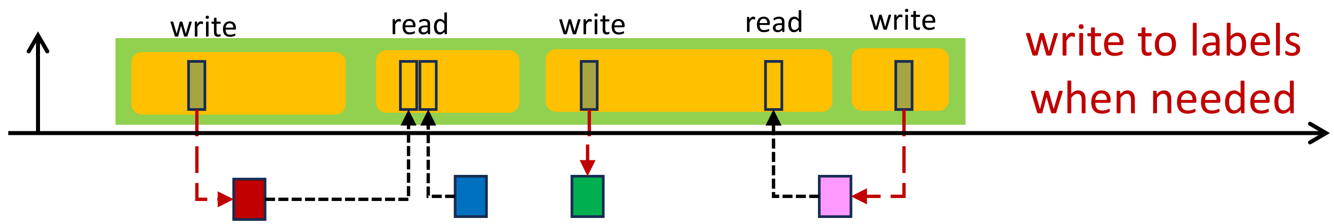
Runnables communicate with each other via reading or writing on "labels"

Each label contains a certain type of data, for example, car speed, engine's angle, amount of fuel in the tank, …

Real world automotive benchmark for free (2015)
Authors: Simon Kramer, Dirk Ziegenbein, Arne Hamann
From: Robert Bosch GmbH, Renningen, Germany
https://www.ecrts.org/forum/viewtopic3edd-2.html?f=20&t=23&sid=d74079af129d5480a5ac4fd1778eecc1

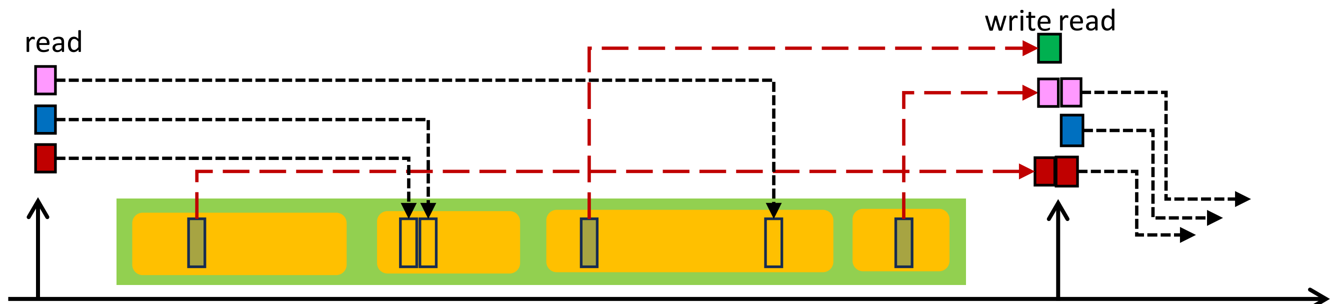
When to read or write?

Explicit access model



Read from and write to labels when needed

Logical Execution Time (LET) a.k.a. implicit access model



Read the labels at the arrival time

Update the labels just before the next arrival, and then read again

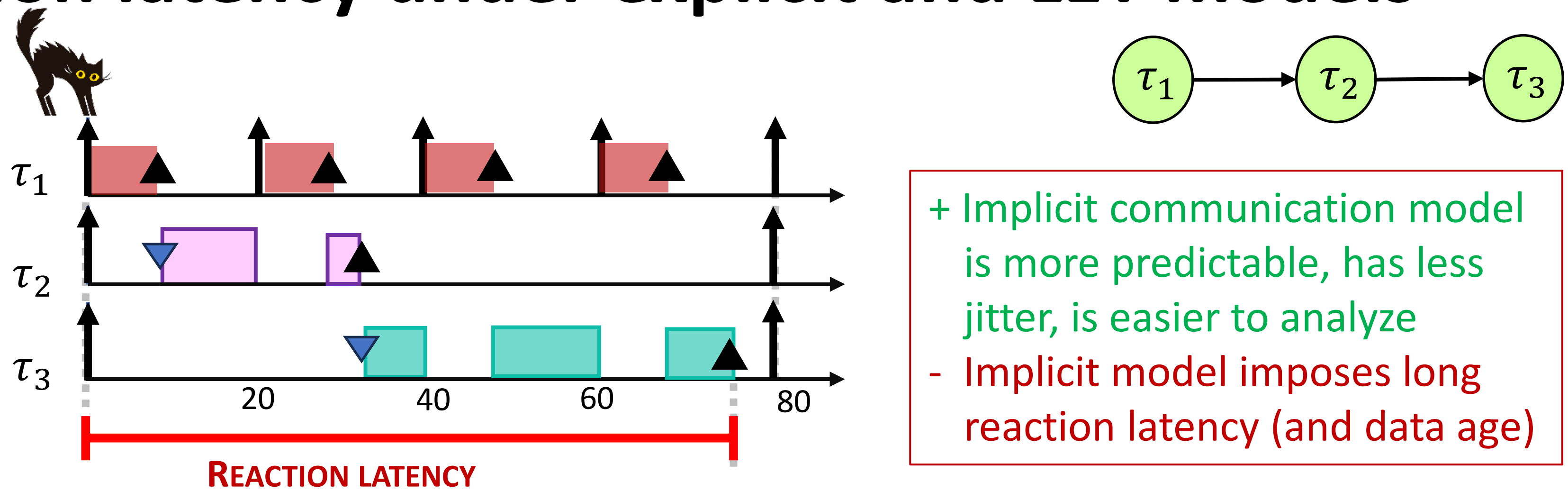
Implicit model has no I/O jitter and has a fixed sampling delay, so it facilitates design of control systems.

Deriving the bound on the blocking time caused by accessing shared resources is simpler in the implicit model.

Real world automotive benchmark for free (2015), Authors: Simon Kramer, Dirk Ziegenbein, Arne Hamann
From: Robert Bosch GmbH, Renningen, Germany
https://www.ecrts.org/forum/viewtopic3edd-2.html?f=20&t=23&sid=d74079af129d5480a5ac4fd1778eecc1

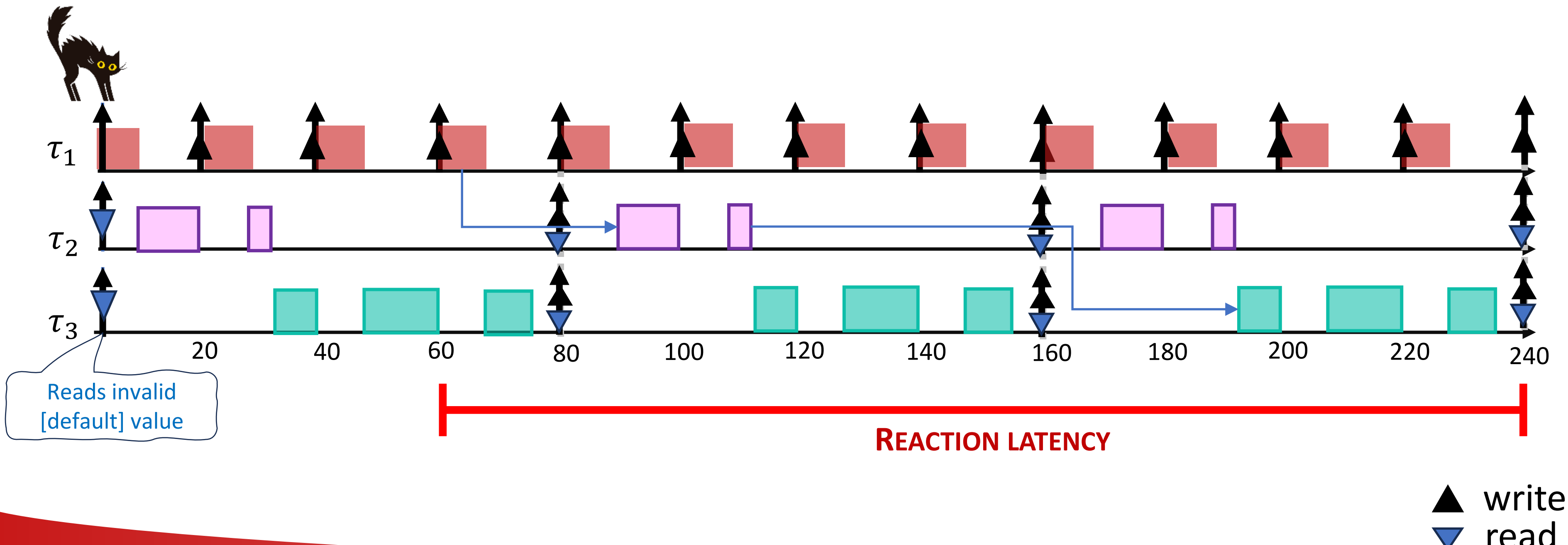
Reaction latency under explicit and LET models

Explicit



- + Implicit communication model is more predictable, has less jitter, is easier to analyze
- - Implicit model imposes long reaction latency (and data age)

LET



Simplifying assumption

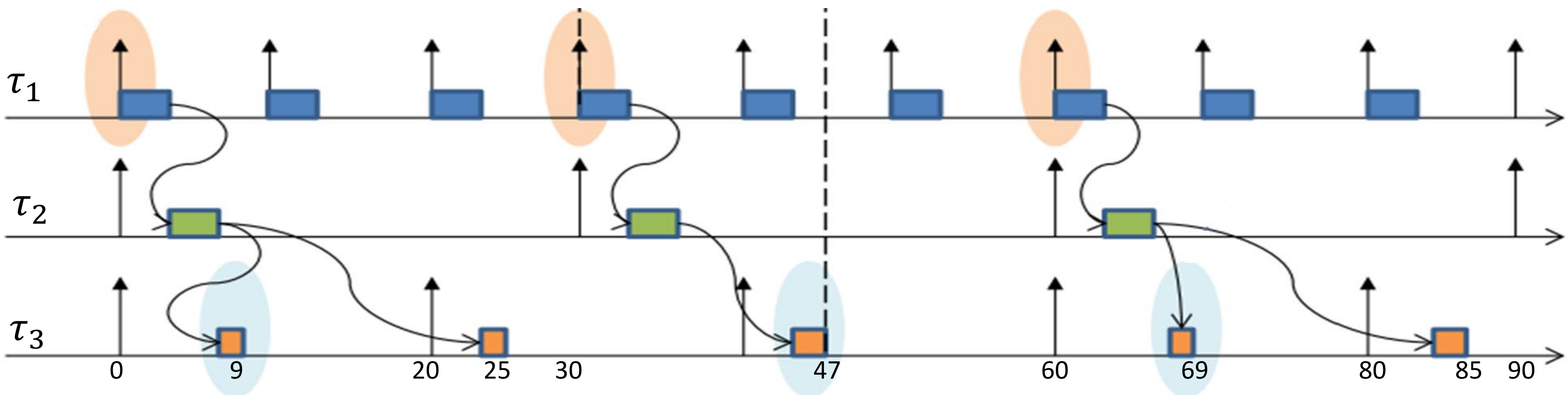
In this part of the lecture (to analyze data age and reaction latency), assume that tasks always execute for the worst-case execution time (BCET = WCET)

The literature provides solutions for cases where the execution times vary, but that's not covered by the course in the analysis of reaction latency and data age.

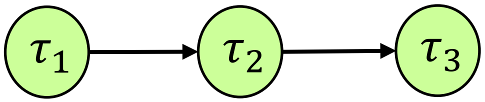
Jakaria Abdullah, Gaoyang Dai, and Wang Yi, Worst-Case Cause-Effect Reaction Latency in Systems with Non-Blocking Communication, 2019.

Calculating reaction latency

1. Find all **complete chain instances** (that start from a stimulus and end in a response)
2. Obtain the **reaction latency of each chain instance** (only until its first response, not other responses for the same stimulus)
3. Keep **track of the longest** response
4. Continue until **all** possible complete chain instances are found (may go beyond one hyperperiod)



Reaction latency = $\max\{\mathbf{9-0, 47-30, 69-60}\} = \mathbf{17}$



Jakaria Abdullah, Gaoyang Dai, and Wang Yi, Worst-Case Cause-Effect Reaction Latency in Systems with Non-Blocking Communication, 2019.

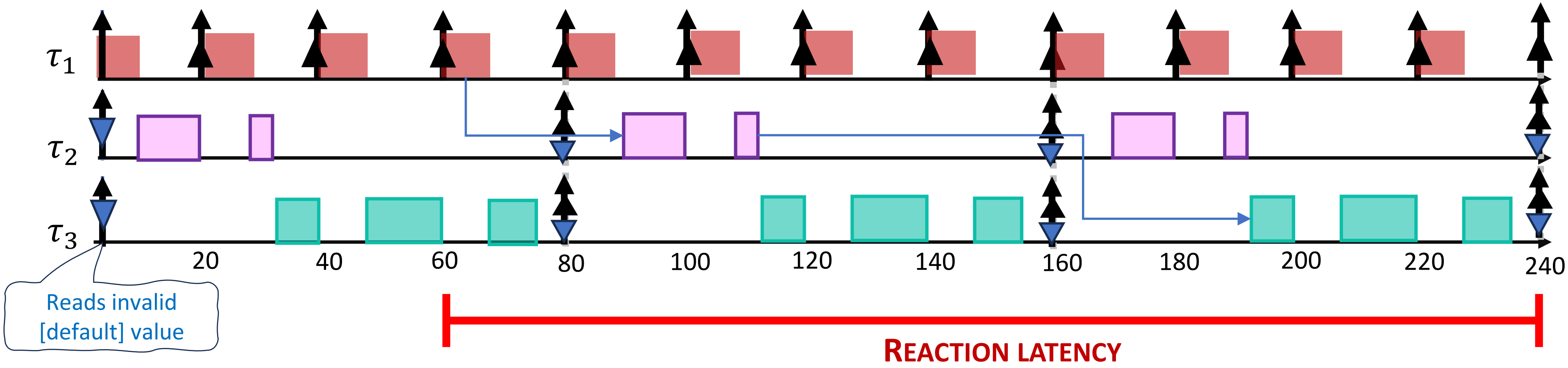
The observation window

Under the assumption that tasks do not have release jitter and all ECUs are time synchronized, **the largest reaction latency and data age** of a task on the chain α_k appears during the interval $[0, OW(\alpha_k)]$

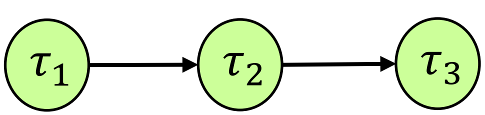
$$OW(\alpha_k) = \sum_{\tau_i \in \alpha_k} 2 \times T_i$$

Matthias Becker, Dakshina Dasari, Saad Mubeen, Moris Behnam, and Thomas Nolte. 2016. Synthesizing job-level dependencies for automotive multi-rate effect chains. In Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA). Pp. 159–169.

Reaction latency in the LET example

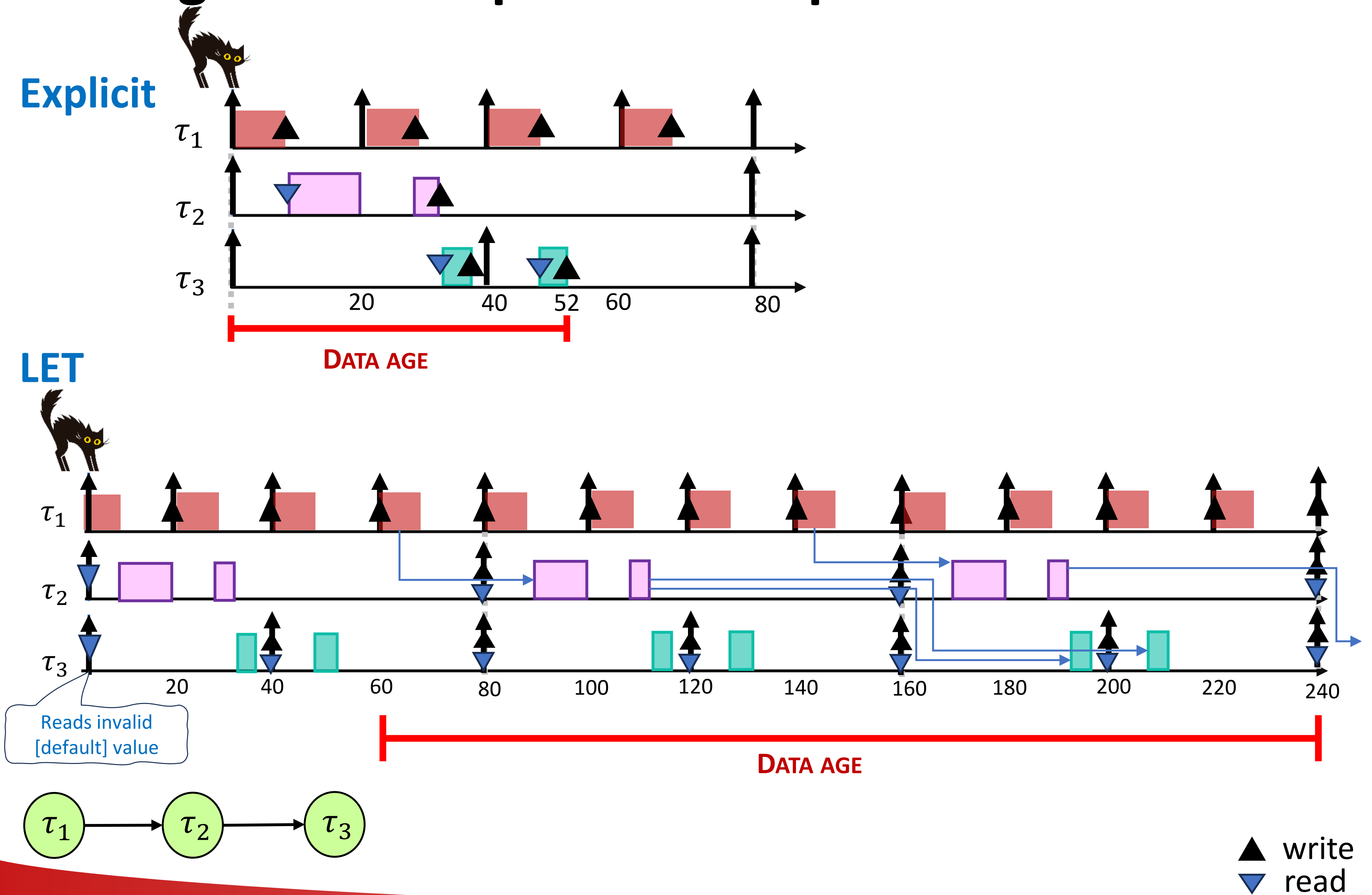


Reaction latency = $\max\{240 - 60\} = 180$



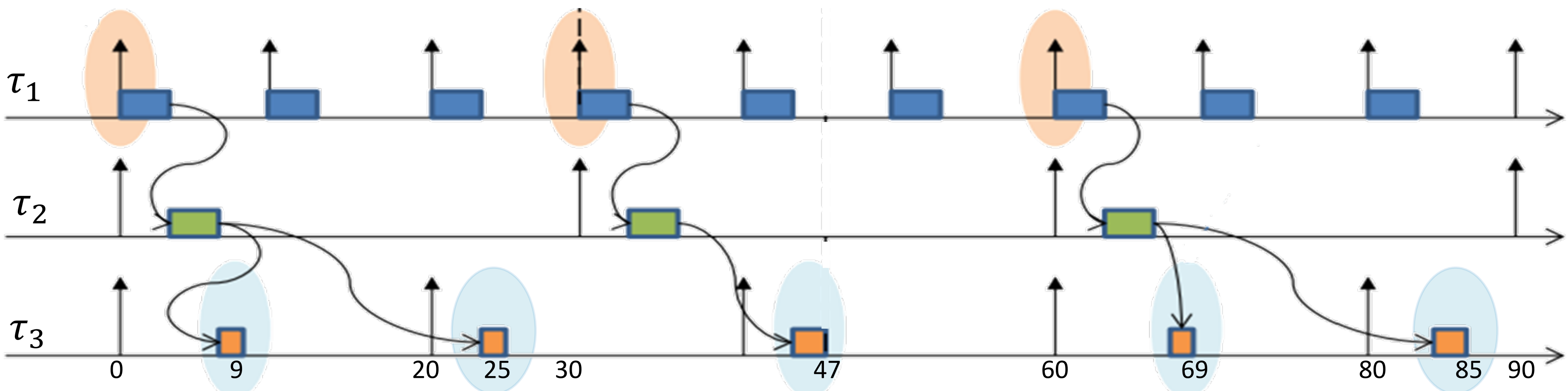
▲ write
▼ read

Data age under implicit and explicit models

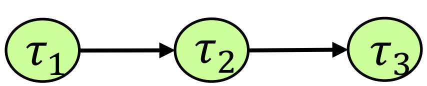


Data age under implicit and explicit models

1. Find all **complete chain instances** (that start from a stimulus and end in a response)
2. Obtain the **data age of each chain instance** (**until the last response made by the chain -– do not stop at the first response**)
3. Keep **track of the longest** response
4. Continue until **all** possible complete chain instances are found (may go beyond one hyperperiod)



Data age $= \max\{\mathbf{9-0, 25-0, 47-30, 69-60, 85-60}\}$
$= \mathbf{25}$

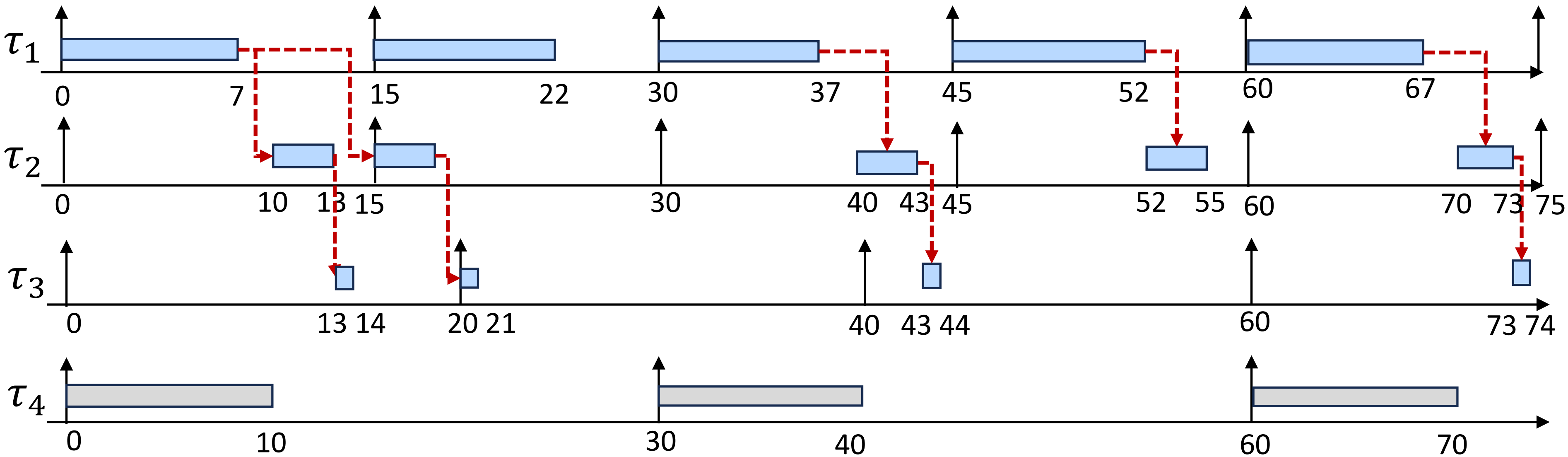


Jakaria Abdullah, Gaoyang Dai, and Wang Yi, Worst-Case Cause-Effect Reaction Latency in Systems with Non-Blocking Communication, 2019.

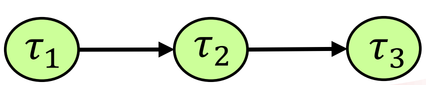
Impact of execution time variation

Shorter execution times (than WCET) may result in longer **reaction latency**

τ_i	C_i	P_i	$core$	period
τ_1	7	90	Core1	15
τ_2	3	80	**Core2**	15
τ_3	1	70	**Core2**	20
τ_4	10	90	Core2	30



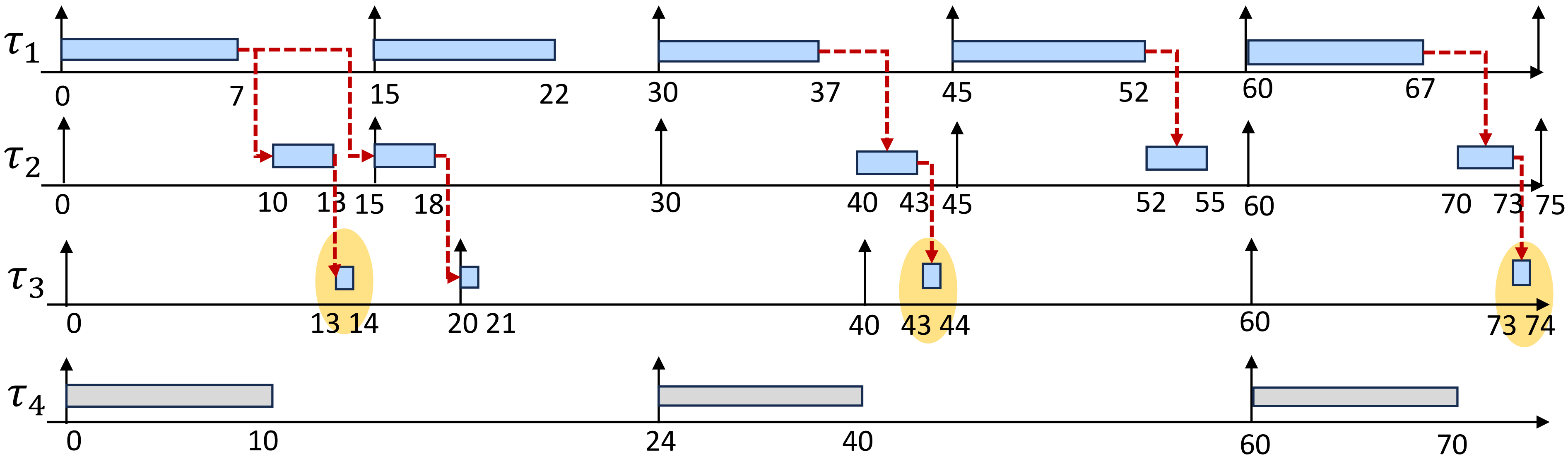
Note: a response is calculated from the arrival time of the root task until the time the data produced by the last task is written onto the memory.



Impact of execution time variation

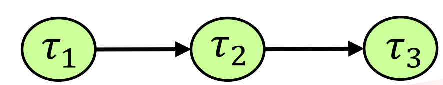
Shorter execution times (than WCET) may result in longer reaction latency

τ_i	C_i	P_i	$core$	period
τ_1	7	90	Core1	15
τ_2	3	80	**Core2**	15
τ_3	1	70	**Core2**	20
τ_4	10	90	Core2	30



Reaction latency = $\max\{\mathbf{14-0, 44-30, 74-60, \ldots}\} = \mathbf{14}$

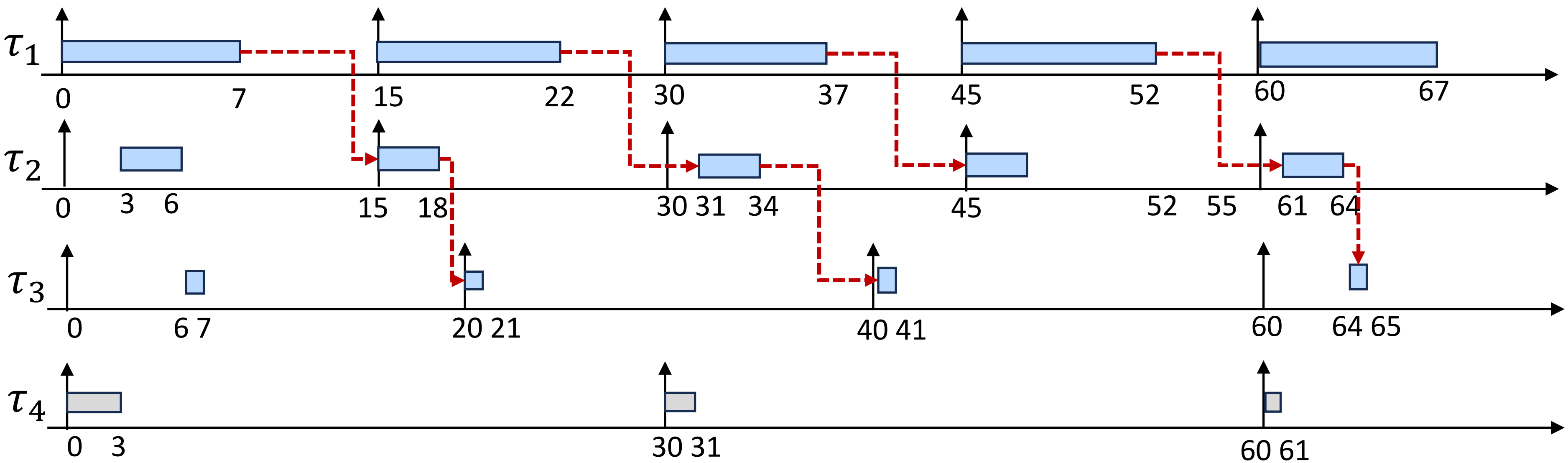
Note: a response is calculated from the arrival time of the root task until the time the data produced by the last task is written onto the memory.



Impact of execution time variation

Shorter execution times (than WCET) may result in longer reaction latency

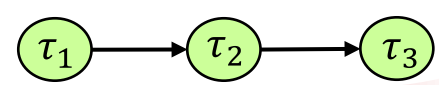
τ_i	C_i	P_i	$core$	period
τ_1	7	90	Core1	15
τ_2	3	80	**Core2**	15
τ_3	1	70	**Core2**	20
τ_4	10	90	Core2	30



Task τ_4 has execution time variation

Reaction latency $\geq \max\{\mathbf{21-0, 41-15, 65-45, \ldots}\} \geq \mathbf{21}$

Note: a response is calculated from the arrival time of the root task until the time the data produced by the last task is written onto the memory.



Impact of execution time variation: conclusion

Execution time variation does not impact the reaction latency and data age of the LET model

In the **explicit model**, if there is **execution time variation,** there might be **many scheduling scenarios** and therefore **our previous solution** reaction latency and data age (namely, finding all chains by drawing the schedule) **is not valid.**

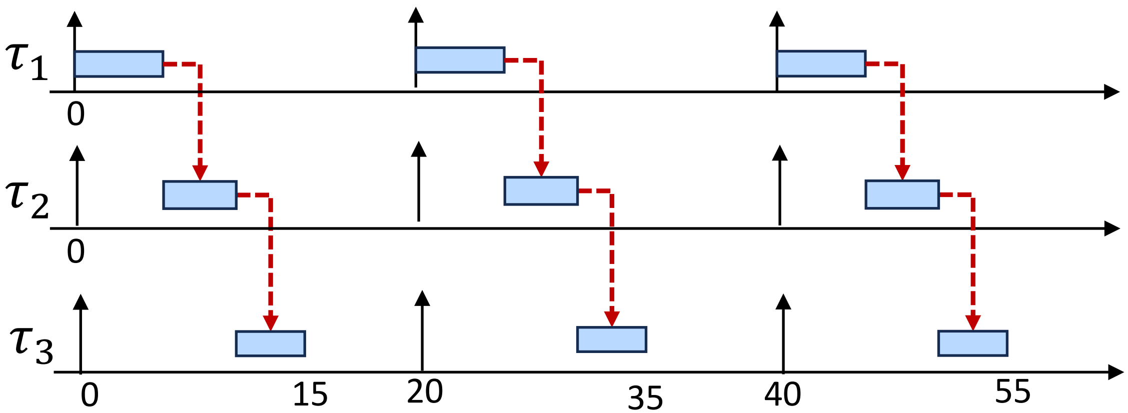
Solutions to this problem require working with uncertainty intervals:
Pourya Gohari, Mitra Nasri, Jeroen Voeten, "Data-Age Analysis for Multi-Rate Task Chains under Timing Uncertainty," the International Conference on Real-Time Networks and Systems (RTNS), 2022.

www.menti.com

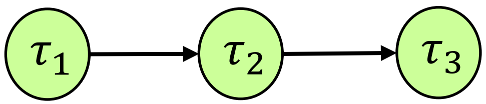
Quiz review

In the explicit model, Reaction Latency is larger than the period of the first task of a task chain.

Depends on the periods



Core 1: $\{\tau_1, \tau_2, \tau_3\}$



$T_1 = T_2 = T_3 = 20$
$C_1 = 5, C_2 = 5, C_3 = 5$

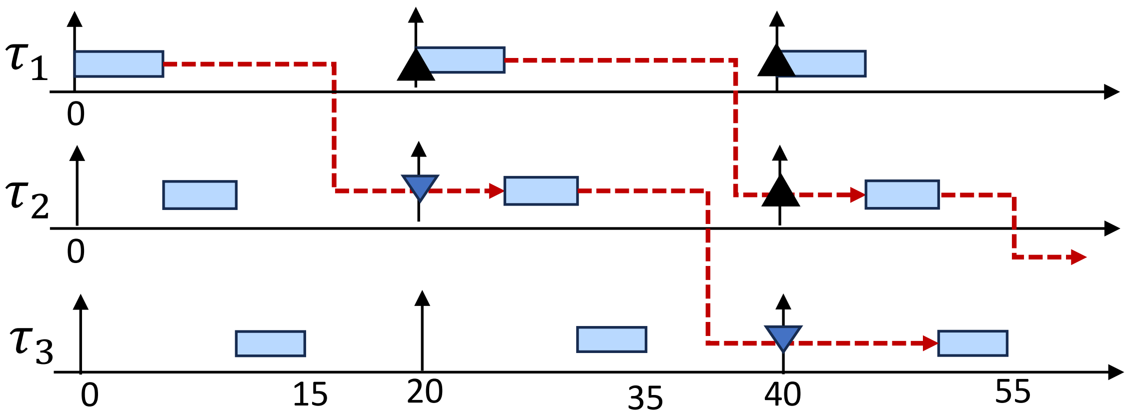
$\boldsymbol{P_1 > P_2 > P_3}$

Reaction latency = data age = 15

Quiz review

In the LET model, Reaction Latency is larger than the period of the first task of a task chain.

Always true



Core 1: $\{\tau_1, \tau_2, \tau_3\}$

$\tau_1 \rightarrow \tau_2 \rightarrow \tau_3$

$T_1 = T_2 = T_3 = 20$
$C_1 = 5, C_2 = 5, C_3 = 5$

$\boldsymbol{P_1 > P_2 > P_3}$

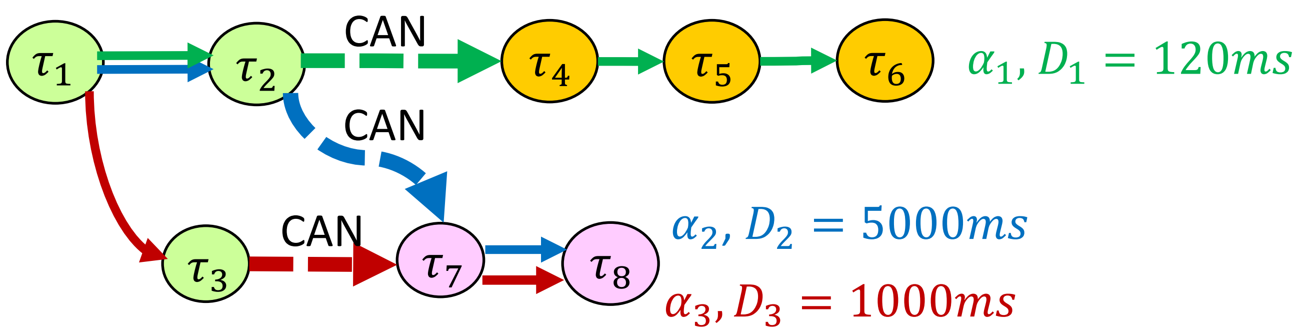
Reaction latency = data age = 55

▲ write
▼ read

Quiz review

In an event chain, predecessors and successors of a task can never preempt it even when they have a higher priority.

Always true



Quiz review

Xi is a lower bound and Ri is an upper bound on the WCRT of a task in a cause-effect chain.

$$R_i = \sigma_i + X_i$$
$$X_i^{(0)} = C_i$$
$$X_i^{(k)} = C_i + \sum_{j \in hp(\tau_i)} \left\lceil \frac{X_i^{(k-1)} + \sigma_j}{T_j} \right\rceil \cdot C_j$$

False

X_i can be larger than the true WCRT.

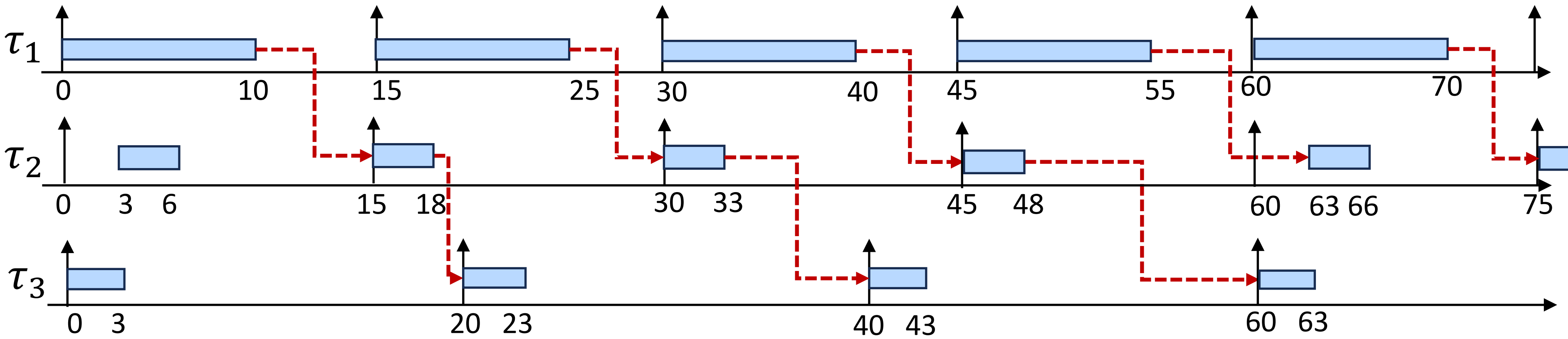
Calculating X_i requires knowing the **release jitter** of other higher-priority tasks (including those that might be on an event chain).

However, release jitter of those tasks comes from the WCRT of the tasks that were before them in the chain. Namely, it is derived from R_j values.

As mentioned before, there is no exact method to obtain tight bounds on R_j, therefore, the release jitter of the tasks on the chain is only an upper approximation. This means that X_i which is obtained using those release jitters is also an upper approximation of the worst-case delay, and hence, might be smaller than the actual WCRT.

Quiz review

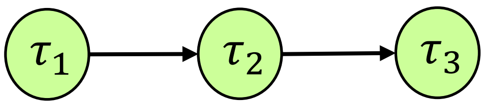
Which of these value can be a response for this multi-rate task chain?



Core 1: τ_1
Core 2: $\{\tau_2, \tau_3\}$

$P_1 > \boldsymbol{P_3} > \boldsymbol{P_2}$

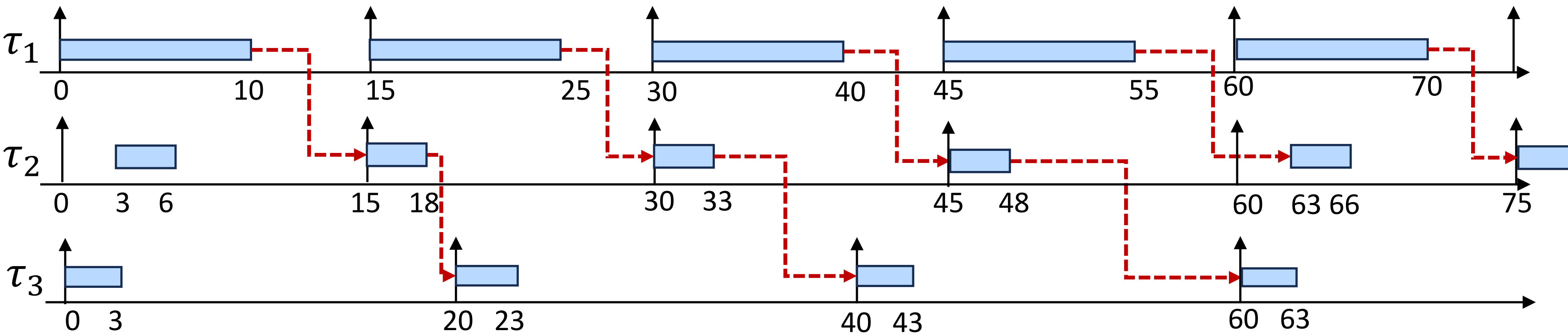
$T_1 = 15, T_2 = 15, T_3 = 20$
$C_1 = 10, C_2 = 3, C_3 = 3$



Responses: {23-0, 43-15, 63 – 30, ….} = {23, 28, 33}

Note: a response is calculated from the arrival time of the root task until the time the data produced by the last task is written onto the memory.

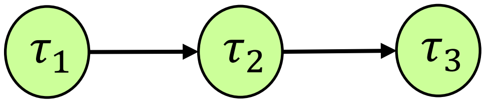
Quiz review: Reaction latency



Core 1: $\boldsymbol{\tau_1}$
Core 2: $\{\boldsymbol{\tau_2}, \boldsymbol{\tau_3}\}$

$P_1 > \boldsymbol{P_3} > \boldsymbol{P_2}$

$T_1 = 15, T_2 = 15, T_3 = 20$
$C_1 = 10, C_2 = 3, C_3 = 3$



Reaction latency = max $\{\mathbf{23 - 0, 43 - 15, 63 - 30,}$ …}
= max{23, 28, 33} = 33

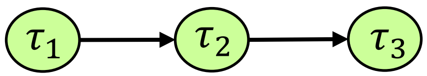
Note: a response is calculated from the arrival time of the root task until the time the data produced by the last task is written onto the memory.

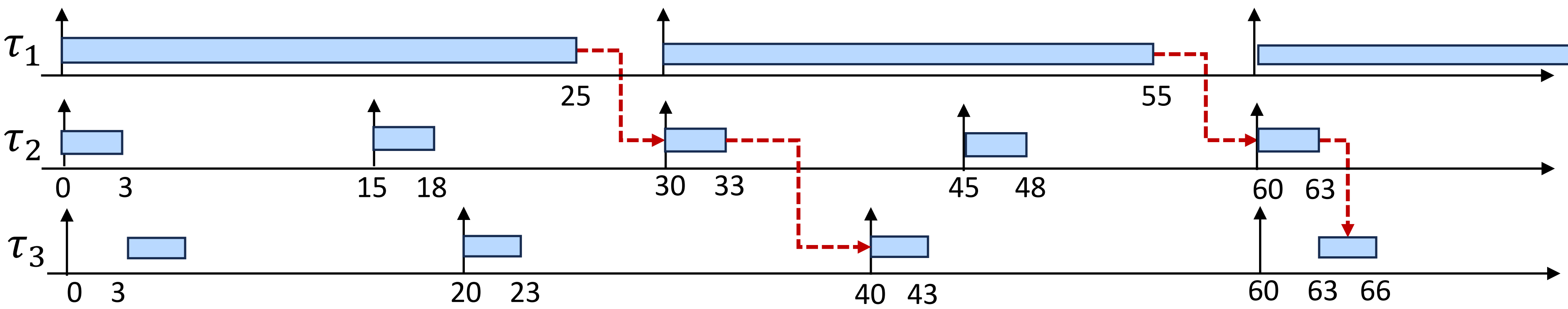
Could the execution time variation of any of these task change the reaction latency of the chain (assume the explicit model)?

Core 1: τ_1
Core 2: $\{\tau_2, \tau_3\}$

$P_1 > \boldsymbol{P_2} > \boldsymbol{P_3}$

$T_1 = 30, T_2 = 15, T_3 = 20$
$C_1 = 25, C_2 = 3, C_3 = 3$





Reaction latency = max $\{43 - 0, 66 - 30, \dots\}$ = max{43, 36, …} = 43

No

Any change in the execution times would still result in a response that starts from time 0 or 30, or 60, etc.
Changes in the execution times here can only decrease the reaction latency.

Note: a response is calculated from the arrival time of the root task until the time the data produced by the last task is written onto the memory.