

Period Optimization and Priority Assignment for Cause Effect Chains

Jian-Jia Chen and Mario Günzel

Department of Computer Science, Chair 12,
TU Dortmund University, Germany

8th of July, 2025



partially supported by the German Federal Ministry of Education and Research (BMBF) and by the European Research Council (ERC).

Real-Time Systems

Cause-Effect Chains

Open Problem(s)

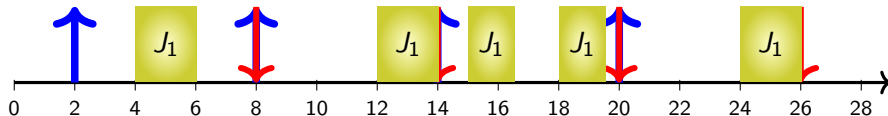
Real-Time Systems

Cause-Effect Chains

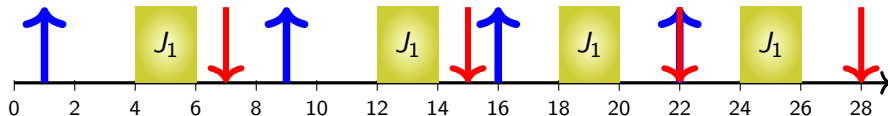
Open Problem(s)

Examples of Recurrent Task Models

Periodic task: $(O_i, C_i, T_i, D_i) = (2, 2, 6, 6)$



Sporadic task: $(C_i, T_i, D_i) = (2, 6, 6)$



Real-Time Scheduling Problems

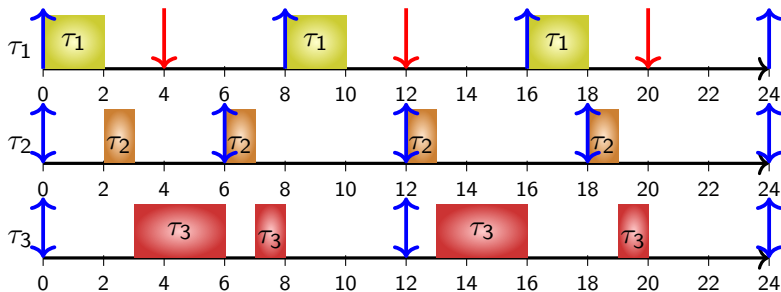
Two separate but co-related problems are studied in real-time systems:

- ① how to *design scheduling policies* to feasibly schedule the tasks, referred to as the *scheduler design* problem as mentioned above already, and
- ② how to *validate* the schedulability of a scheduling algorithm, referred to as the *schedulability test* problem.

Deadline-Monotonic (DM) Scheduling

Priority Definition: A task with a smaller **relative deadline** has higher priority, in which ties are broken arbitrarily.

Example Schedule: $\tau_1 = (2, 8, 4)$, $\tau_2 = (1, 6, 6)$, $\tau_3 = (4, 12, 12)$. $[(C_i, T_i, D_i)]$



Time Demand Analysis (TDA)

Proposed by Joseph in 1986 and Lehoczky, Sha, and Ding [RTSS 1989]:

Theorem (TDA)

The minimum positive Δ such that

$$\Delta = C_k + \sum_{\tau_i \in hp(\tau_k)} \left\lceil \frac{\Delta}{T_i} \right\rceil C_i$$

is a safe upper bound on R_k if it is no more than T_k .

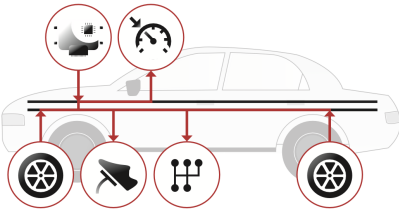
Real-Time Systems

Cause-Effect Chains

Open Problem(s)

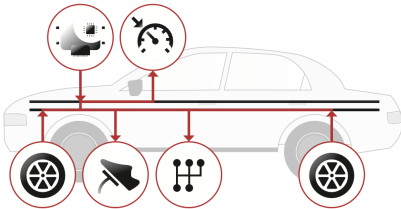
Cause-Effect Chains

- Automotive Systems: Tasks work together to perform functionality



Cause-Effect Chains

- Automotive Systems: Tasks work together to perform functionality



- Example: Automatic break system

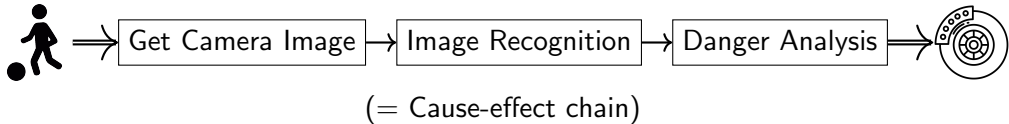
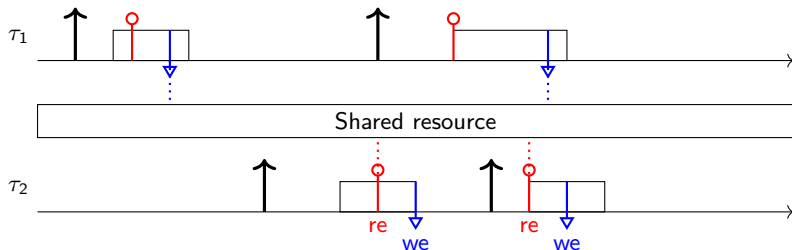


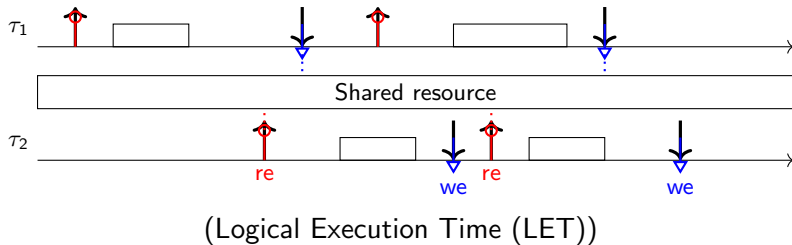
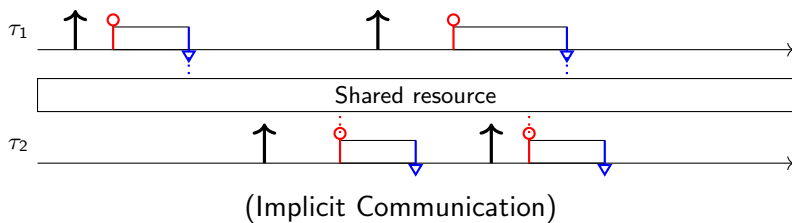
image: Flaticon.com

How Do Tasks Communicate?

- Communication modeled through shared resource
- Read-events (re) and write-events (we)



Typical Communication Policies



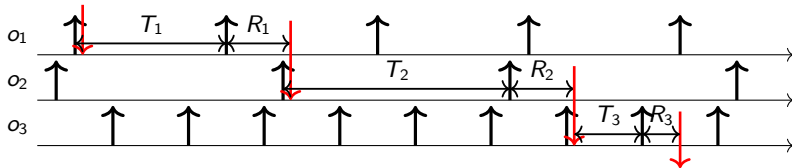
End-to-End Latency

- How long does the data flow of one path take?

Assumptions:

- Sources are activated by external events, while sinks activate actuators
- Objects/Tasks are periodically activated

Analytical upper bound: $\sum_i T_i + R_i$



First-to-First / Maximum Reaction Time (MRT)

is the longest time interval starting from the occurrence of an external cause to the earliest time where this external cause is fully processed.

Responsiveness: how long does it take?

First-to-First / Maximum Reaction Time (MRT)

is the longest time interval starting from the occurrence of an external cause to the earliest time where this external cause is fully processed.

Responsiveness: how long does it take?

Last-to-Last / Maximum Data Age (MDA)

is the length of the longest time interval starting with sampling a value to the last point in time where an actuation is based on this sampled value.

Data Freshness: how old is the input data of the actuation?

Maximum Reaction Time (MRT) / First-to-First (Responsiveness)

How long does it take until an external activity is fully processed?

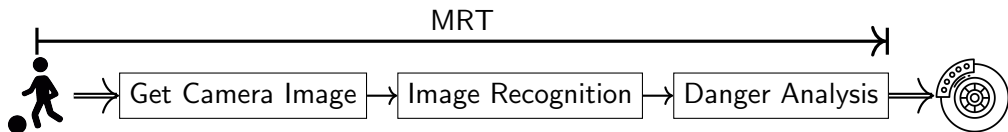
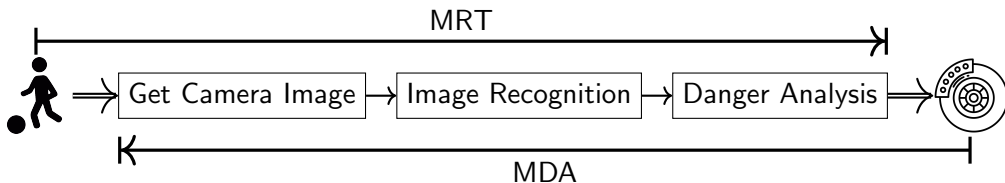


image: Flaticon.com

Maximum Reaction Time (MRT) / First-to-First (Responsiveness)

How long does it take until an external activity is fully processed?



Maximum Data Age (MDA) / Last-to-Last (Data Freshness)

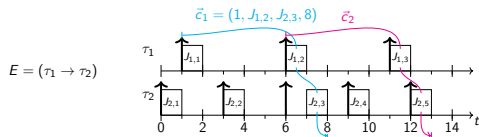
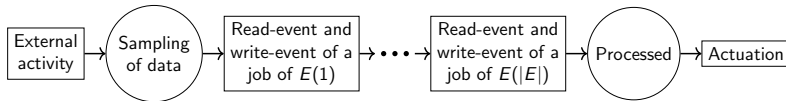
How old is the data used in an actuation?

image: Flaticon.com

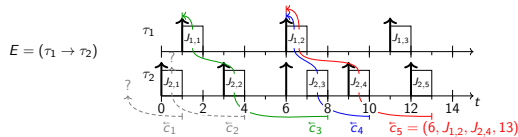
End-to-End Latency

Dürr et al. 2019, Günzel et al. 2021:

- MRT and MDA can be formalized by immediate forward/immediate backward augmented job chains $(z, J_1, \dots, J_{|E|}, z')$
- $MDA \leq MRT$; even: $MDA = MRT$ (Günzel et al. 2023)



$$MRT(E) := \sup \left\{ \ell(\vec{c}_m^E) \right\}$$



$$MDA(E) := \sup \left\{ \ell(\vec{c}_m^E) \right\}$$

Real-Time Systems

Cause-Effect Chains

Open Problem(s)

Problem Statement

Given:

- two cause-effect chains with $E_1 = (\tau_{1,1} \rightarrow \tau_{1,2} \rightarrow \cdots \rightarrow \tau_{1,n_1})$ and $E_2 = (\tau_{2,1} \rightarrow \tau_{2,2} \rightarrow \cdots \rightarrow \tau_{2,n_2})$
- end-to-end latency constraints of the cause-effect chains, which may differ

Problem Statement

Given:

- two cause-effect chains with $E_1 = (\tau_{1,1} \rightarrow \tau_{1,2} \rightarrow \cdots \rightarrow \tau_{1,n_1})$ and $E_2 = (\tau_{2,1} \rightarrow \tau_{2,2} \rightarrow \cdots \rightarrow \tau_{2,n_2})$
- end-to-end latency constraints of the cause-effect chains, which may differ

How to assign the periods, phases, and priorities of the tasks so that the end-to-end latency constraints are fulfilled?

Problem Statement

Given:

- two cause-effect chains with $E_1 = (\tau_{1,1} \rightarrow \tau_{1,2} \rightarrow \dots \rightarrow \tau_{1,n_1})$ and $E_2 = (\tau_{2,1} \rightarrow \tau_{2,2} \rightarrow \dots \rightarrow \tau_{2,n_2})$
- end-to-end latency constraints of the cause-effect chains, which may differ

How to assign the periods, phases, and priorities of the tasks so that the end-to-end latency constraints are fulfilled?

We assume a fixed-priority preemptive scheduler.

Naive Approach

Use typical bounds for optimization, such as:

- $\text{Lat}(E) \leq \sum_{i=1}^n T_{\tau_i} + R_{\tau_i}$ (Davare et al. 2007)
 - $\text{Lat}(E) \leq \sum_{i=1}^n T_{\tau_i} + R_{\tau_i} - \sum_{i=1}^{n-1} [P_i] \cdot \min(R_{\tau_i}, T_{\tau_{i+1}})$ (Dürr et al. 2019)
- Davare et al. 2007 used geometric programming to assign periods

Naive Approach

Use typical bounds for optimization, such as:

- $\text{Lat}(E) \leq \sum_{i=1}^n T_{\tau_i} + R_{\tau_i}$ (Davare et al. 2007)
- $\text{Lat}(E) \leq \sum_{i=1}^n T_{\tau_i} + R_{\tau_i} - \sum_{i=1}^{n-1} [P_i] \cdot \min(R_{\tau_i}, T_{\tau_{i+1}})$ (Dürr et al. 2019)

- Davare et al. 2007 used geometric programming to assign periods
- Drawback: When *phases* can be assigned, bounds become very pessimistic

Example

Assign: phase 0, all task same period T , priorities in order of chain

$$\Rightarrow \text{Lat}(E) = T + \sum_{i=1}^n C_i$$

Two Independent Chains (No Shared Tasks)

- WLOG: $L_1 \leq L_2$

Procedure

- Assign tasks in E_1 higher priority than those in E_2 (each following the respective order)
 - Set one period T to tasks in E_1 , and one period $\hat{T}$ to tasks in E_2
-
- $\text{Lat}(E_1) = T + \sum_{i=1}^{n_1} C_{1,\tau_i}$
 - $\text{Lat}(E_2) = \hat{T} + R_{2,n_2}$

Two Independent Chains (No Shared Tasks)

- WLOG: $L_1 \leq L_2$

Procedure

- Assign tasks in E_1 higher priority than those in E_2 (each following the respective order)
 - Set one period T to tasks in E_1 , and one period $\hat{T}$ to tasks in E_2
- $\text{Lat}(E_1) = T + \sum_{i=1}^{n_1} C_{1,\tau_i}$
 - $\text{Lat}(E_2) = \hat{T} + R_{2,n_2}$

How to set T ?

Two Independent Chains (No Shared Tasks)

- WLOG: $L_1 \leq L_2$

Procedure

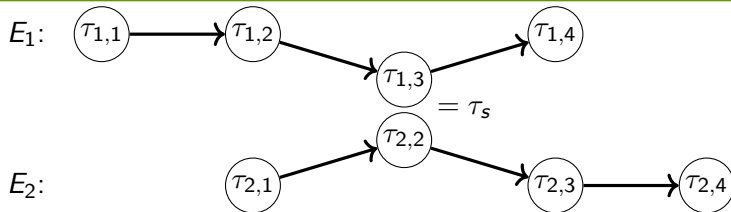
- Assign tasks in E_1 higher priority than those in E_2 (each following the respective order)
 - Set one period T to tasks in E_1 , and one period $\hat{T}$ to tasks in E_2
- $\text{Lat}(E_1) = T + \sum_{i=1}^{n_1} C_{1,\tau_i}$
 - $\text{Lat}(E_2) = \hat{T} + R_{2,n_2}$

How to set T ?

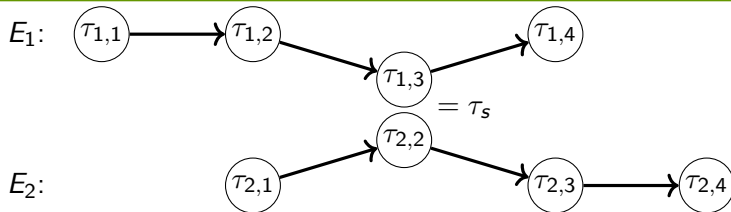
Conjecture

Setting T to $L_1 - \sum_{i=1}^{n_1} C_{1,\tau_i}$ is optimal.

Two Chains with Shared Tasks



Two Chains with Shared Tasks



Problems Transferring this approach

For shared task τ_s

- Want to assign τ_s multiple periods T and $\hat{T}$
- Want to give τ_s multiple priorities

$\Rightarrow$ Optimal assignment becomes *much more* complex

Two Chains with Shared Tasks

Heuristic 1

Try out different possibilities for shared tasks

Heuristic 1

Try out different possibilities for shared tasks

- Does not scale well: 4^S options with S shared tasks
- Unclear whether one of these options is actually the optimal

Two Chains with Shared Tasks

Heuristic 1

Try out different possibilities for shared tasks

- Does not scale well: 4^S options with S shared tasks
- Unclear whether one of these options is actually the optimal

Heuristic 2

Decompose the chain and optimize individually (e.g., using Cutting Theorem):

$$E_1 = E_1^1 \rightarrow \tau_s \rightarrow E_1^2 \text{ and } E_2 = E_2^1 \rightarrow \tau_s \rightarrow E_2^2$$

Two Chains with Shared Tasks

Heuristic 1

Try out different possibilities for shared tasks

- Does not scale well: 4^S options with S shared tasks
- Unclear whether one of these options is actually the optimal

Heuristic 2

Decompose the chain and optimize individually (e.g., using Cutting Theorem):

$$E_1 = E_1^1 \rightarrow \tau_s \rightarrow E_1^2 \text{ and } E_2 = E_2^1 \rightarrow \tau_s \rightarrow E_2^2$$

- How to distribute L_1 and L_2 among the new chains?
- What is the pessimism introduced through Cutting?
- Worst case: Cutting at every task $\Rightarrow$ (Davare et al. 2007)?