

Introduction to Intelligent Vehicles

[4. System Design]

Chung-Wei Lin

cwlin@csie.ntu.edu.tw

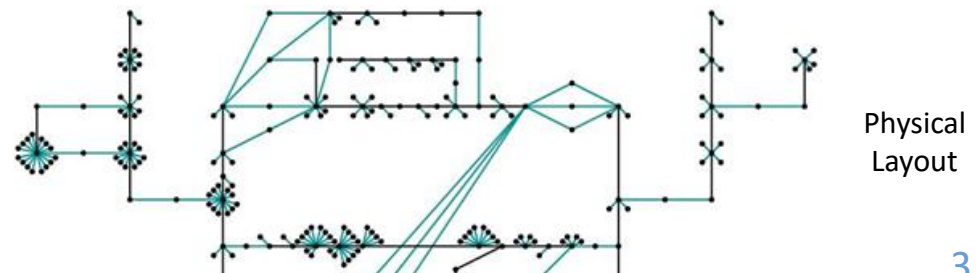
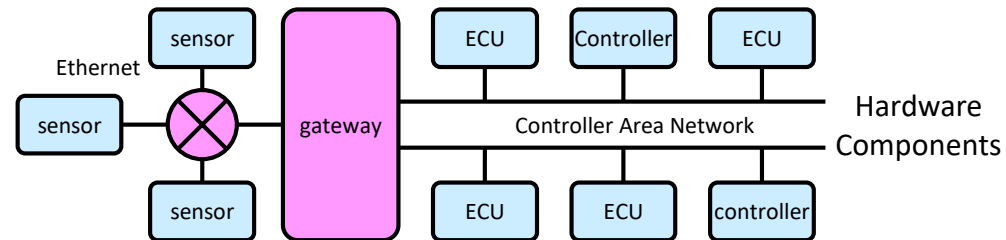
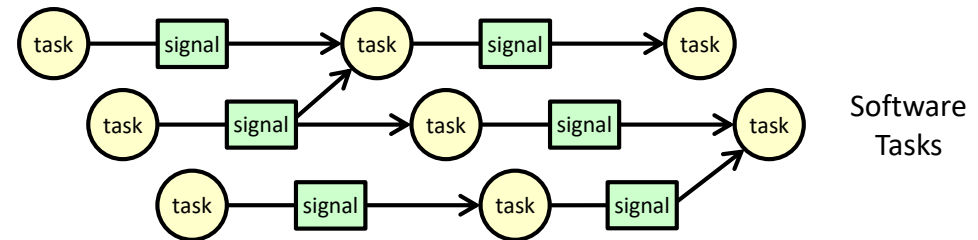
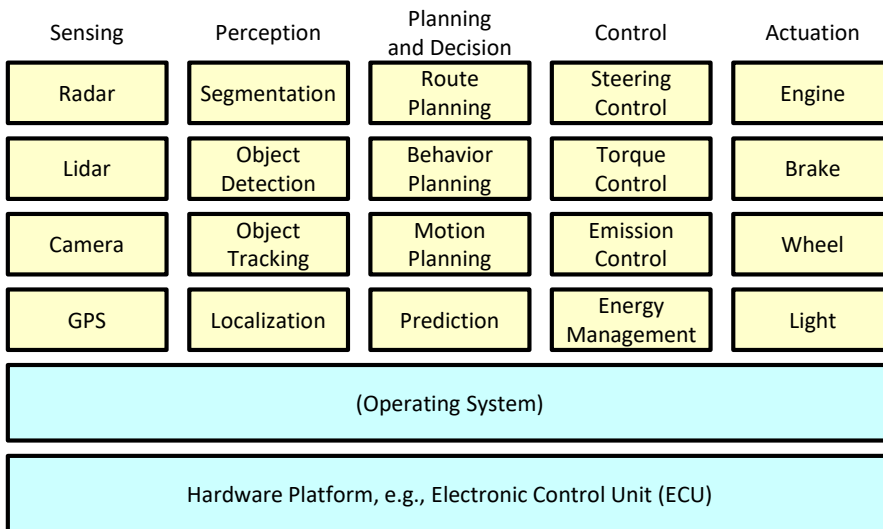
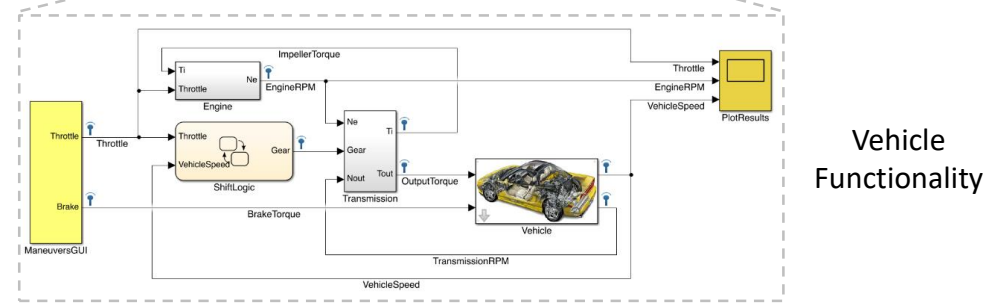
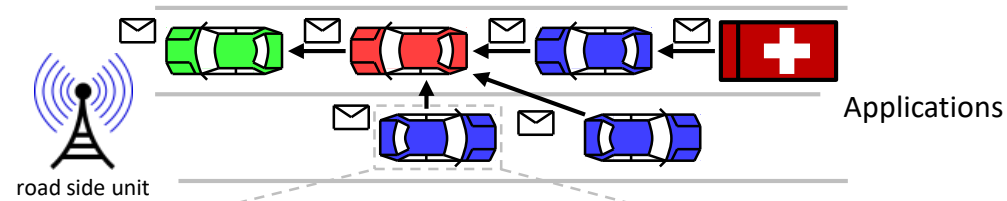
CSIE Department

National Taiwan University

System Design

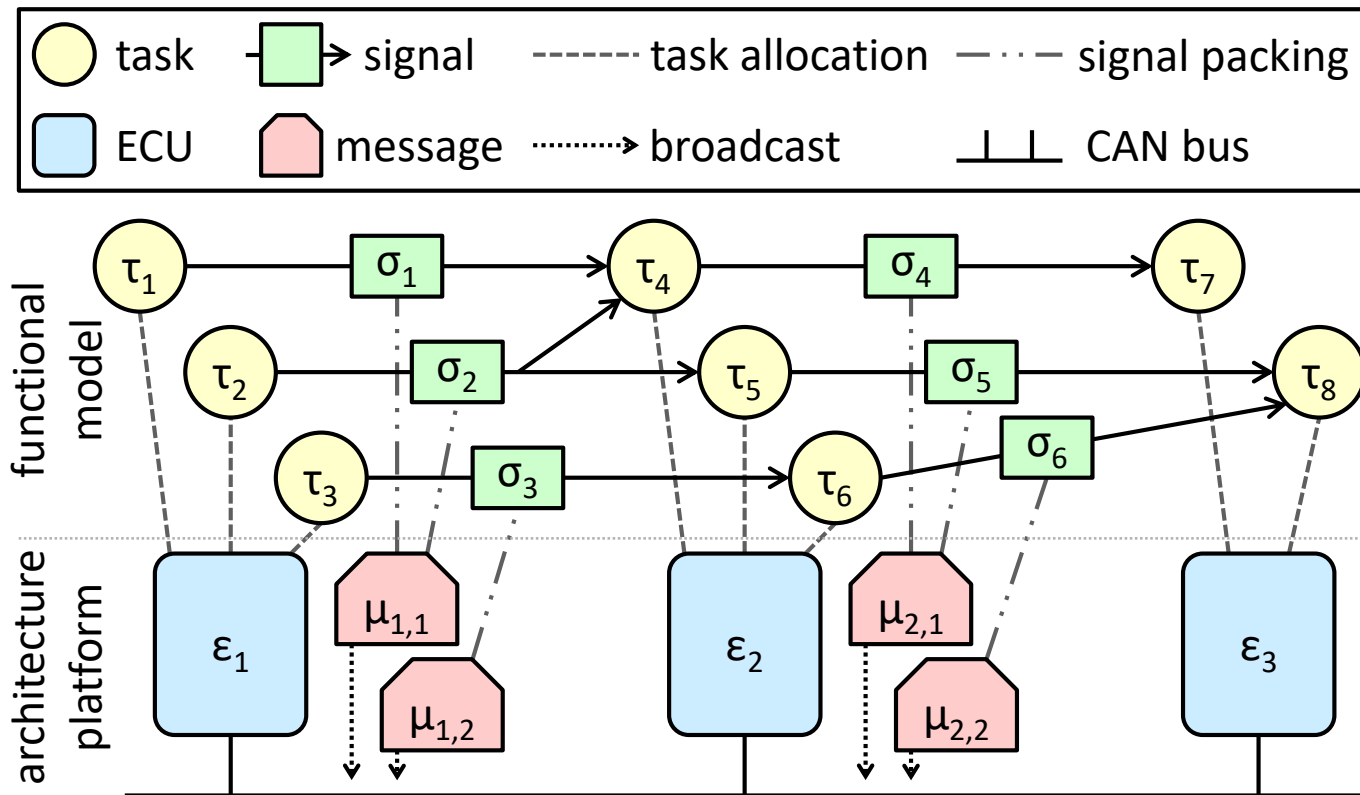
- ❑ What is system design?
- ❑ Why is system design needed?
- ❑ When is system design done?
- ❑ Where is system design done?
- ❑ Who performs system design?
- ❑ How to perform system design?

Layered Design



Example (Piece) of System Design

- ❑ What is system design? / Why is system design needed?
- ❑ When / Where is system design done?
- ❑ Who / How to perform(s) system design?



Cores of Model-Based Design

❑ Modeling

- Work (design and analyze) upon models rather than real systems

❑ Design

- Optimize some objectives (performance, robustness, security, etc.)
- Satisfy some constraints

❑ Analysis

- Check if there is any flaw or how good the designs are?

❑ (Implementation)

- Make it real

Outline

- ❑ Mixed Integer Linear Programming (MILP)
- ❑ Simulated Annealing
- ❑ Mapping Problem
 - It is just a piece of system design

Linear Programming

□ Linear Programming (LP)

➤ Maximize

- $2x_1 - x_2$

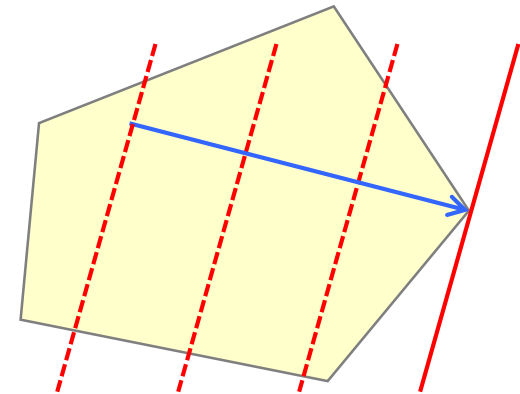
➤ Subject to

- $x_1 - x_2 \leq 1$

- $2x_2 \leq 3$

- $x_1, x_2 \geq 0$

- x_1, x_2 are real numbers



□ Canonical form

$$\min \quad \mathbf{c}^T \mathbf{x}$$

$$\text{subject to} \quad \mathbf{Ax} \leq \mathbf{b}, \mathbf{x} \geq \mathbf{0}, \mathbf{x} \text{ in } \mathbb{R}$$

□ Simplex algorithm

➤ https://en.wikipedia.org/wiki/Simplex_algorithm

Integer (Linear) Programming (ILP)

□ Integer (Linear) Programming (ILP)

➤ Maximize

- $2x_1 - x_2$

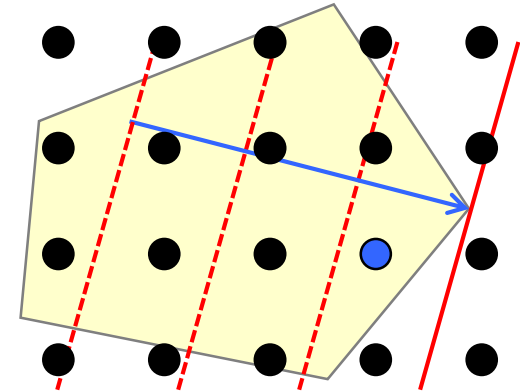
➤ Subject to

- $x_1 - x_2 \leq 1$

- $2x_2 \leq 3$

- $x_1, x_2 \geq 0$

- x_1, x_2 are integers



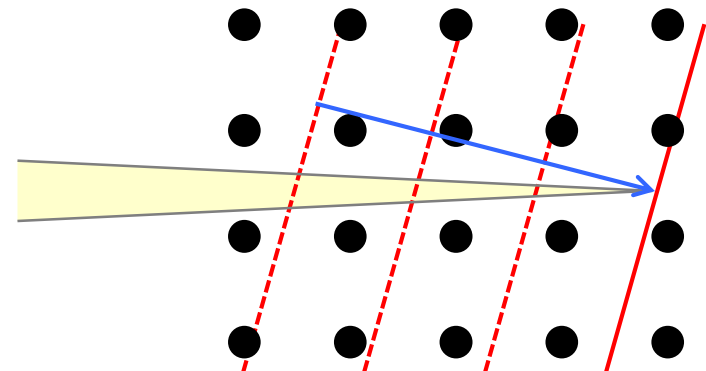
□ Canonical form

$$\max \quad \mathbf{c}^T \mathbf{x}$$

$$\text{subject to} \quad \mathbf{Ax} \leq \mathbf{b}, \mathbf{x} \geq \mathbf{0}, \mathbf{x} \in \mathbb{Z}$$

□ NP-complete

➤ Why is it hard?



Mixed Integer Linear Programming (MILP)

❑ Mixed Integer Linear Programming (MILP)

➤ Maximize

- $2x_1 - x_2$

➤ Subject to

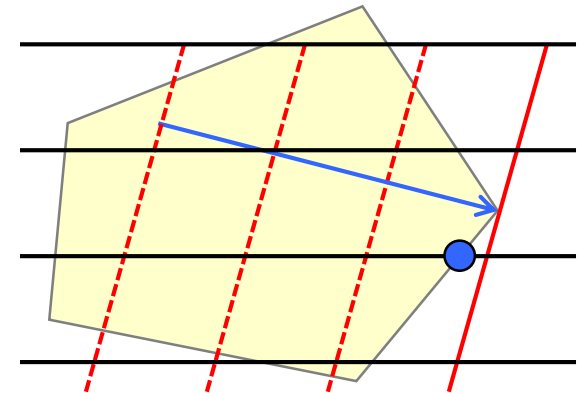
- $x_1 - x_2 \leq 1$

- $2x_2 \leq 3$

- $x_1, x_2 \geq 0$

- x_1 is real number

- x_2 is integer



❑ Canonical form

$$\max \quad \mathbf{c}^T \mathbf{x}$$

$$\text{subject to} \quad \mathbf{Ax} \leq \mathbf{b}, \mathbf{x} \geq \mathbf{0}, \text{ some } \mathbf{x} \text{ are integers}$$

❑ Many tools are available for solving LP, ILP, and MILP

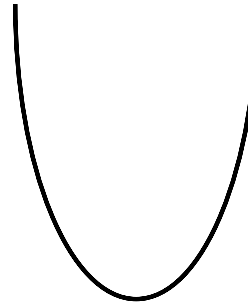
Quadratic Programming (QP)

□ Canonical form

$$\begin{array}{ll}\min & 1/2 \mathbf{x}^T \mathbf{Q} \mathbf{x} + \mathbf{c}^T \mathbf{x} \\ \text{subject to} & \mathbf{A} \mathbf{x} \leq \mathbf{b}, \mathbf{x} \geq \mathbf{0}\end{array}$$

- \mathbf{Q} is symmetric

➤ A special case of convex optimization



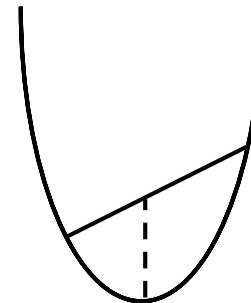
□ Convex optimization

$$\begin{array}{ll}\max & f(\mathbf{x}) \\ \text{subject to} & g_i(\mathbf{x}) \leq 0 \text{ for all } i\end{array}$$

- f, g_i are convex

➤ A function f is convex if

- For all x_1 and x_2 in the domain of f , t in $[0,1]$
 $f(t \cdot x_1 + (1-t) \cdot x_2) \leq t \cdot f(x_1) + (1-t) \cdot f(x_2)$



Practice

□ Bin packing problem [Wikipedia]

➤ Given

- A set of bins S_1, S_2, \dots with the same size V
- A set of n items with sizes a_1, a_2, \dots, a_n

➤ "Pack" all items into bins and minimize the number of used bins

□ LP? ILP? MILP? QP?

max $\mathbf{c}^T \mathbf{x}$

subject to $\mathbf{Ax} \leq \mathbf{b}, \mathbf{x} \geq \mathbf{0}, \mathbf{x} \in \mathbf{Z}$

min $y_1 + y_2 + \dots + y_n$

subject to $x_{i1} + x_{i2} + \dots + x_{in} = 1$ for all i

$a_1 x_{1j} + a_2 x_{2j} + \dots + a_n x_{nj} \leq Vy_j$ for all j

$x_{ij} = 0$ or $1 \rightarrow 1$: if and only if item i is packed in to bin j

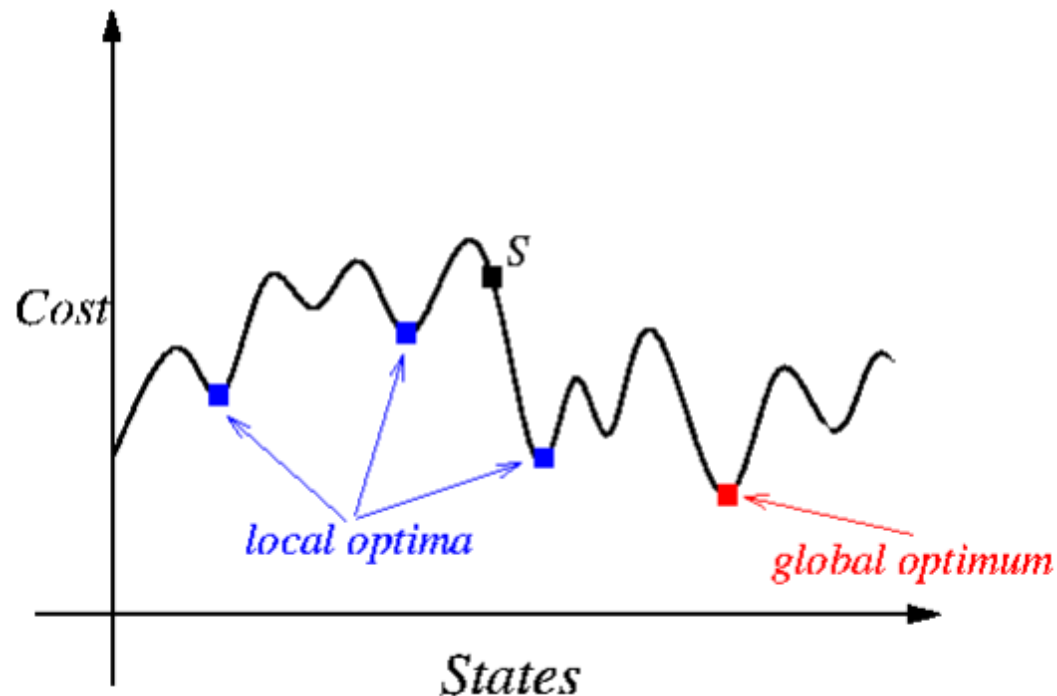
$y_j = 0$ or $1 \rightarrow 1$: if and only if bin j is used

Outline

- ❑ Mixed Integer Linear Programming (MILP)
- ❑ **Simulated Annealing**
- ❑ Mapping Problem
 - It is just a piece of system design

Simulated Annealing: Background

- Kirkpatrick, Gelatt, and Vecchi, "Optimization by simulated annealing," Science, May 1983.



Simulated Annealing: Basics

- ❑ Non-zero probability for "up-hill" moves

- ❑ Probability depends on

- Magnitude of the "up-hill" movement
- Total search time

- ❑ $\text{Prob}(S \rightarrow S') = \min(1, e^{-\Delta C/T})$

- Equivalence

- $\text{Prob}(S \rightarrow S') = 1$ if $\Delta C \leq 0$ ("down-hill" move)
- $\text{Prob}(S \rightarrow S') = e^{-\Delta C/T}$ if $\Delta C > 0$ ("up-hill" move)

- $\Delta C = \text{cost}(S') - \text{cost}(S)$

- "Smaller" is better

- T = temperature

- T will cool down gradually

Simulated Annealing: Algorithm

- ❑ Get an initial solution S
- ❑ Get an initial temperature $T > 0$
- ❑ $S^* = S$
- ❑ While "not yet frozen" (i.e., T is large enough)
 - (Iterate many times)
 - Pick a random neighbor S' of S
 - $\Delta C = \text{cost}(S') - \text{cost}(S)$
 - If $\text{cost}(S') < \text{cost}(S^*)$, then $S^* = S'$
 - If $\Delta C \leq 0$, then $S = S'$
 - If $\Delta C > 0$, then $S = S'$ with probability $e^{-\Delta C/T}$
 - $T = rT$ (where $r < 1$)
- ❑ Return S^*

Simulated Annealing

❑ Basic ingredients

- Solution space
 - What are the feasible solutions?
- Neighborhood structure
 - How to find a neighboring solution from the current one?
- Cost function
 - How to evaluate the quality of a solution?
- Annealing schedule
 - How to conduct the search process to find a desired solution?

❑ Philosophy

Practice

❑ Bin packing problem [Wikipedia]

➤ Given

- A set of bins S_1, S_2, \dots with the same size V
- A set of n items with sizes a_1, a_2, \dots, a_n

➤ "Pack" all items into bins and minimize the number of used bins

❑ Basic ingredients

➤ Solution space

- What are the feasible solutions?

➤ Neighborhood structure

- How to find a neighboring solution from the current one?

➤ Cost function

- How to evaluate the quality of a solution?

➤ Annealing schedule

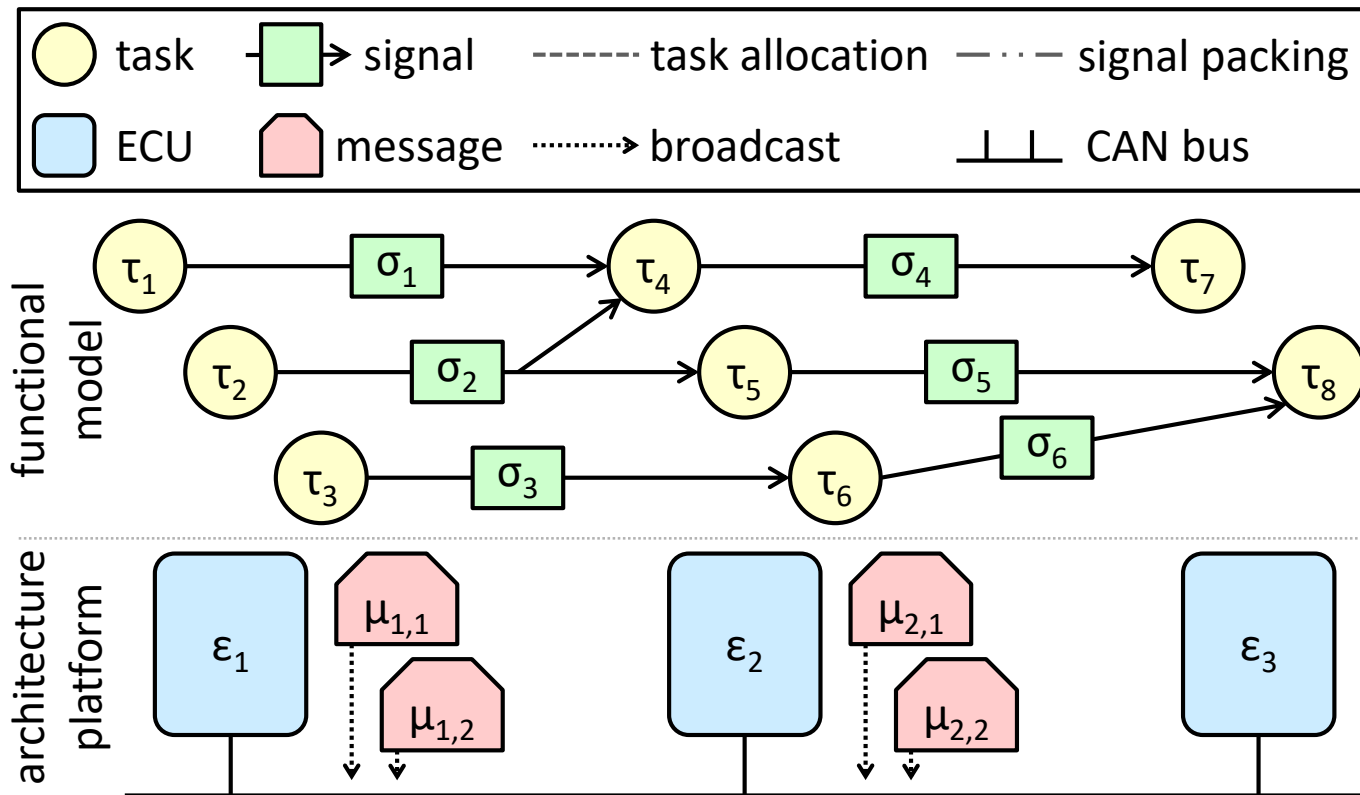
- How to conduct the search process to find a desired solution?

Outline

- ❑ Mixed Integer Linear Programming (MILP)
- ❑ Simulated Annealing
- ❑ **Mapping Problem**
 - It is just a piece of system design

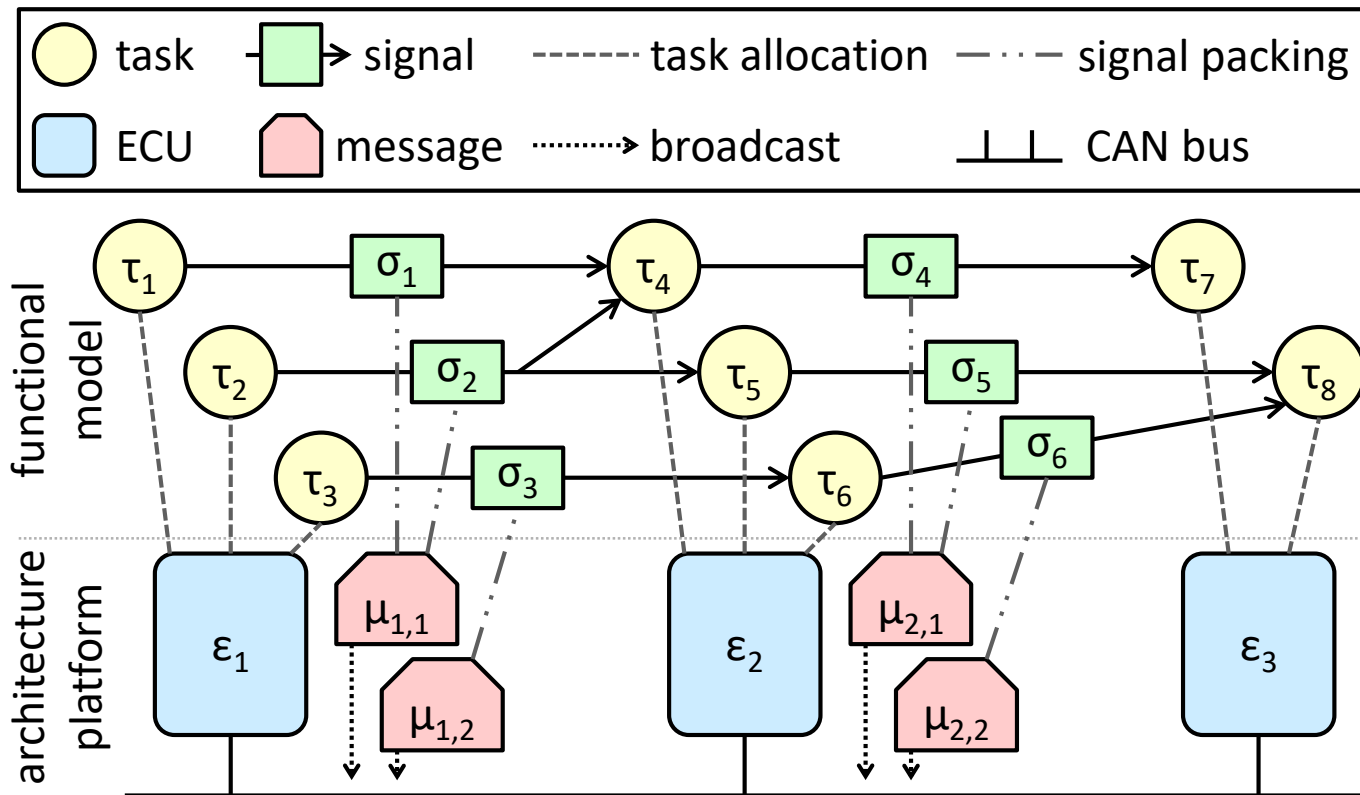
Mapping from Software to Hardware

- ❑ Software (functional model): task graph
- ❑ Hardware (architectural platform): distributed Electronic Control Units (ECUs) connected by a network



Problem Formulation

- Decide task allocation, signal packing, and priority assignments (tasks on ECUs and messages on CAN bus)
- Satisfy timing constraints for tasks, signals, and paths



Task Allocation

□ Indices and variables

- i, j : index of a task
- k : index of an ECU
- $a_{i,k}$: $[0,1]$ task i is allocated onto ECU k
- $s_{i,j}$: $[0,1]$ tasks i and j are allocated onto the same ECU

□ Constraints

- $\sum_k a_{i,k} = 1$ for all i
- $a_{i,k} + a_{j,k} + s_{i,j} \neq 2$ for all i, j, k

□ Questions

- What are the meanings of the constraints?
- Why do we need $s_{i,j}$?
- Is it possible that all tasks are allocated onto one ECU?

Task Priority Assignment

□ Indices and variables

- i, j, j' : index of a task
- $p_{i,j}$: $[0,1]$ task i has a higher priority than another task j

□ Constraints

- $p_{i,j} + p_{j,i} = 1$ for all $i, j, i \neq j$
- $p_{i,j} + p_{j,j'} - 1 \leq p_{i,j'}$ for all $i, j, j', i \neq j, i \neq j', j \neq j'$

□ Questions

- What are the meanings of the constraints?
- Why do we use binary variables $p_{i,j}$, not integer $p_i = 1, 2, 3, \dots$?
 - $p_i \neq p_j$
 - $p_i < p_j$ or $p_j < p_i$
 - $p_{i,j} + p_{j,i} = 1$ and $(p_i - p_j) < (1 - p_{i,j}) M$ and $(p_j - p_i) < p_{i,j} M$
 - M : a large constant

Signal Packing

□ Indices, constant parameters, and variables

- i, j : index of a task; k : index of an ECU; l : index of a message
- $T_{i,j}$: the period of the signal from task i to task j
- $T_{k,l}$: the period of ECU k 's message l
- $a_{i,k}$: $[0,1]$ task i is allocated onto ECU k
- $t_{i,j,k,l}$: $[0,1]$ the signal from task i to task j is packed into ECU k 's message l
- $v_{k,l}$: $[0,1]$ ECU k 's message l is used

□ Constraints

- $\sum_l t_{i,j,k,l} = a_{i,k} (1 - a_{j,k})$ for all i, j, k
- $t_{i,j,k,l} \leq v_{k,l}$ for all i, j, k, l
- $t_{i,j,k,l} T_{k,l} \leq T_{i,j}$; $t_{i,j,k,l} T_{i,j} \leq T_{k,l}$ for all i, j, k, l

□ Questions

- What are the meanings of the constraints?
- Why do we need $v_{k,l}$?

Message Priority Assignment

- ❑ Similar to task priority assignment

Timing Constraints

❑ Task: response time \leq period (given)

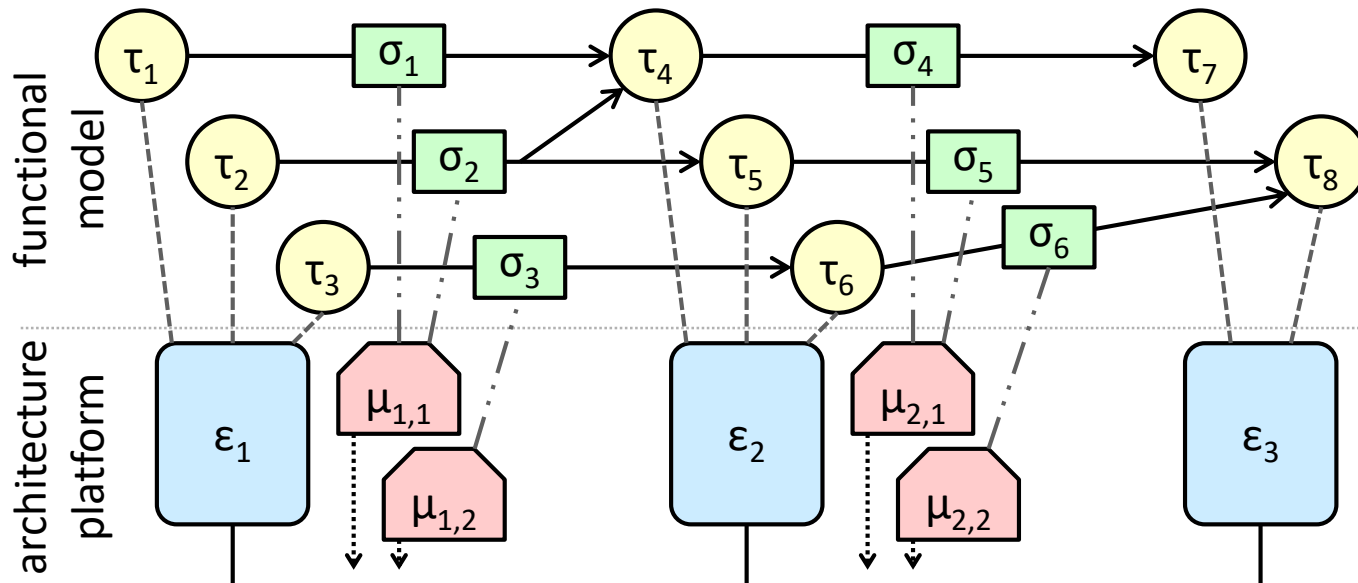
➤ Response time? $r_i = c_i + \sum_{p_j < p_i} \left\lceil \frac{r_i}{T_j} \right\rceil C_j$

❑ Message and signal: response time \leq period (given)

➤ Response time? $q_i = b_i + \sum_{p_j < p_i} \left\lceil \frac{q_i + \tau}{T_j} \right\rceil c_j$ and $r_i = q_i + c_i$

❑ Path: latency \leq deadline (given)

➤ Latency?



Remaining Questions

□ Variables

- $s_{i,j}$: $[0,1]$ tasks i and j are allocated onto the same ECU
- $v_{k,l}$: $[0,1]$ ECU k 's message l is used

□ Questions

- Task allocation
 - Why do we need $s_{i,j}$?
 - $s_{i,j} \cdot p_{i,j}$ in response time computation
 - Is it possible that all tasks are allocated onto one ECU?
 - We have timing constraints
- Signal packing
 - Why do we need $v_{k,l}$?
 - $v_{k,l} \cdot (\text{message size})$ in response time computation

Solved by MILP: Linearization

❑ Inequality of three binary variables: $\alpha + \beta + \gamma \neq 2$

➤ $\alpha + \beta + \gamma \neq 2 \iff \alpha + \beta - \gamma \leq 1; \alpha - \beta + \gamma \leq 1; -\alpha + \beta + \gamma \leq 1$

❑ Ceiling function: $\text{ceil}(f)$

➤ Replace $\text{ceil}(f)$ by an integer x

➤ $\text{ceil}(f) = x \iff 0 \leq x - f < 1$

❑ Multiplication of two binary variables: $\alpha \cdot \beta$

➤ Replace $\alpha \cdot \beta$ by a binary variable γ

➤ $\alpha \cdot \beta = \gamma \iff \alpha + \beta - 1 \leq \gamma; \gamma \leq \alpha; \gamma \leq \beta$

❑ Multiplication of a binary variable α and a real variable x : $\alpha \cdot x$

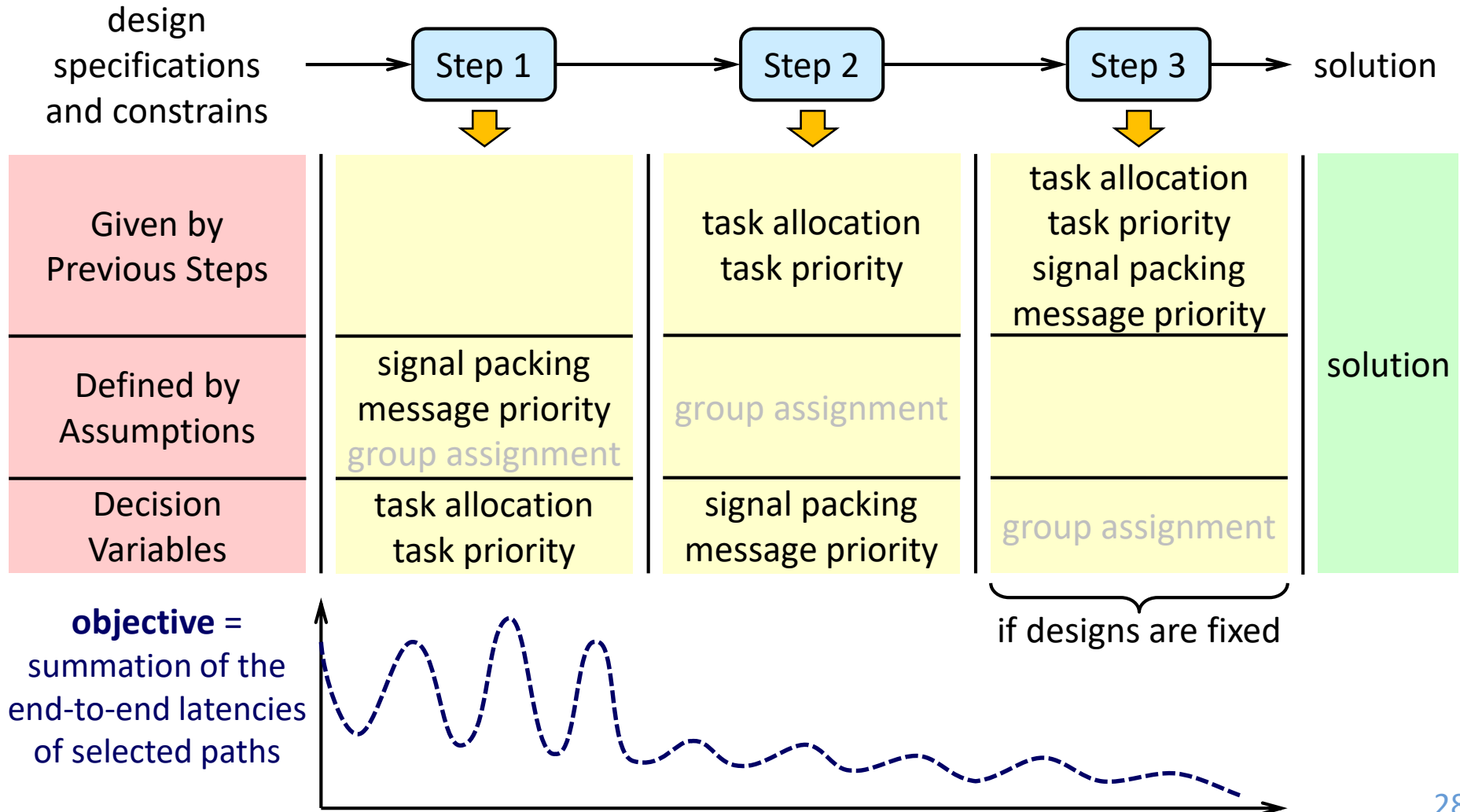
➤ Replace $\alpha \cdot x$ by a real variable y

➤ $\alpha \cdot x = y \iff 0 \leq y \leq x; x - M(1 - \alpha) \leq y \leq M\alpha$

- M : a large constant

Solved by MILP: Scalability Issue

❑ Let us ignore "group assignment" first



Solved by Simulated Annealing

❑ Basic ingredients

- Solution space
 - What are the feasible solutions?
- Neighborhood structure
 - How to find a neighboring solution from the current one?
- Cost function
 - How to evaluate the quality of a solution?
- Annealing schedule
 - How to conduct the search process to find a desired solution?

Implications

❑ Allocate tasks onto the same ECU

- A signal does not need to be transmitted if its source and target tasks are allocated onto the same ECU
- Two signals can be packed into the same message if their source tasks are allocated onto the same ECU

❑ However, not allocate too many tasks onto the same ECU

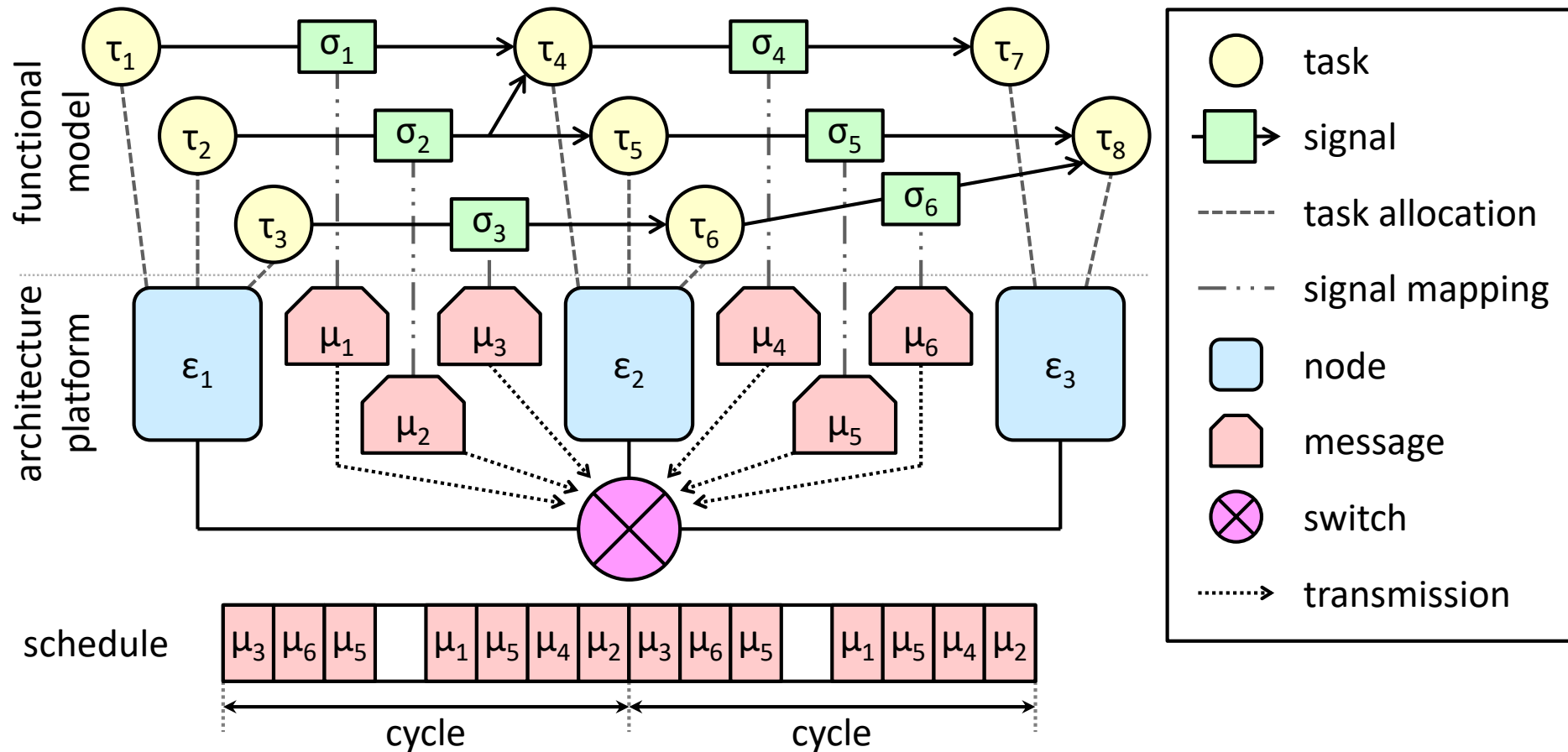
- Long task response time

❑ In most cases, pack signals into the same message

- Save headers!

Another Problem Formulation

Task allocation, task priority assignment, scheduling



TDMA Scheduling

- ❑ Optimal scheduling implied by the timing analysis
 - As early as possible for a synchronous message
 - As evenly as possible for an asynchronous message
- ❑ How to resolve conflicts between multiple messages?

Q&A

Revisit CAN Timing Analysis

$$\boxed{Q_i = B_i + \sum_{(\text{for all } j, P_j < P_i)} \left\lceil \frac{Q_i + \tau}{T_j} \right\rceil C_j}$$

(1)
(2)
(3)
(4)
(5)
(6)

$$\boxed{R_i = Q_i + C_i}$$

(7)

(1): (2)+(6)

(2): blocking time of the longest lower or same priority message

(3): index set of all higher priority messages

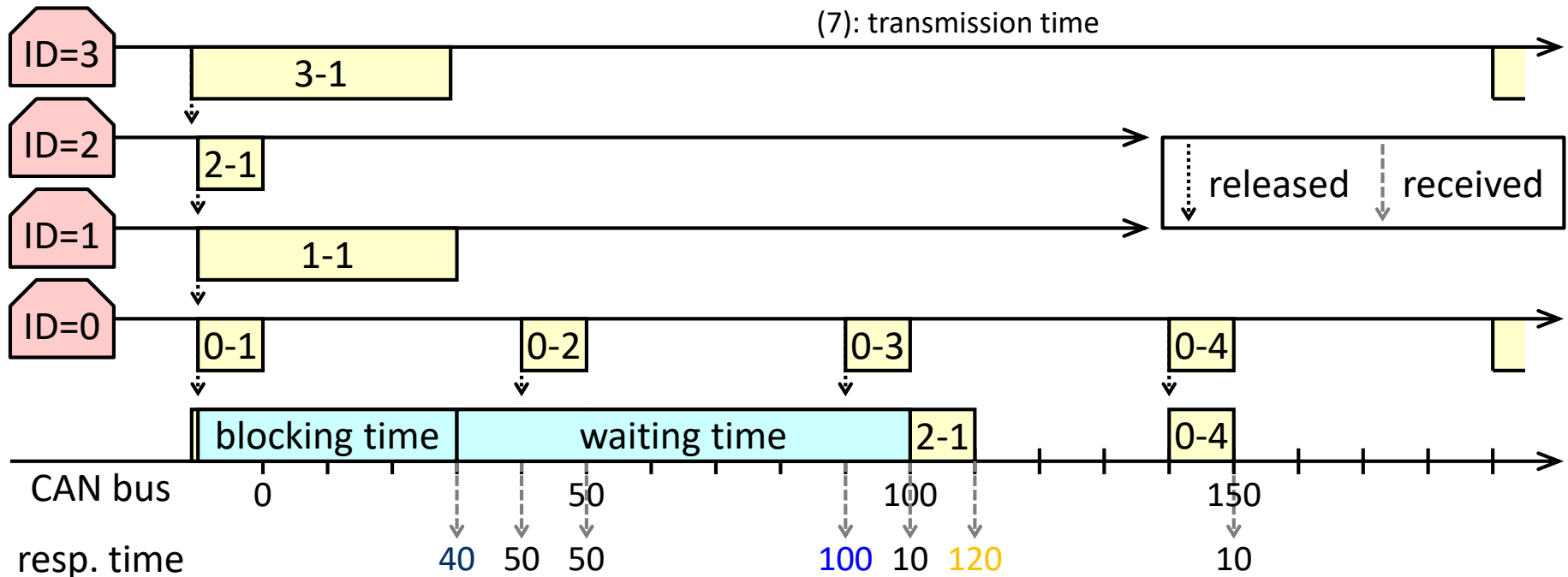
(4): max number of queued instances of message j within (1)

τ : transmission time of one bit

(5): transmission time of an instance of message j

(6): waiting time

(7): transmission time



$$R_0 = 50 \quad R_1 = 100 \quad R_2 = 120 \quad R_3 = 40?$$