Introduction to Intelligent Vehicles
[2. Timing Analysis I]

Chung-Wei Lin

cwlin@csie.ntu.edu.tw

CSIE Department

National Taiwan University

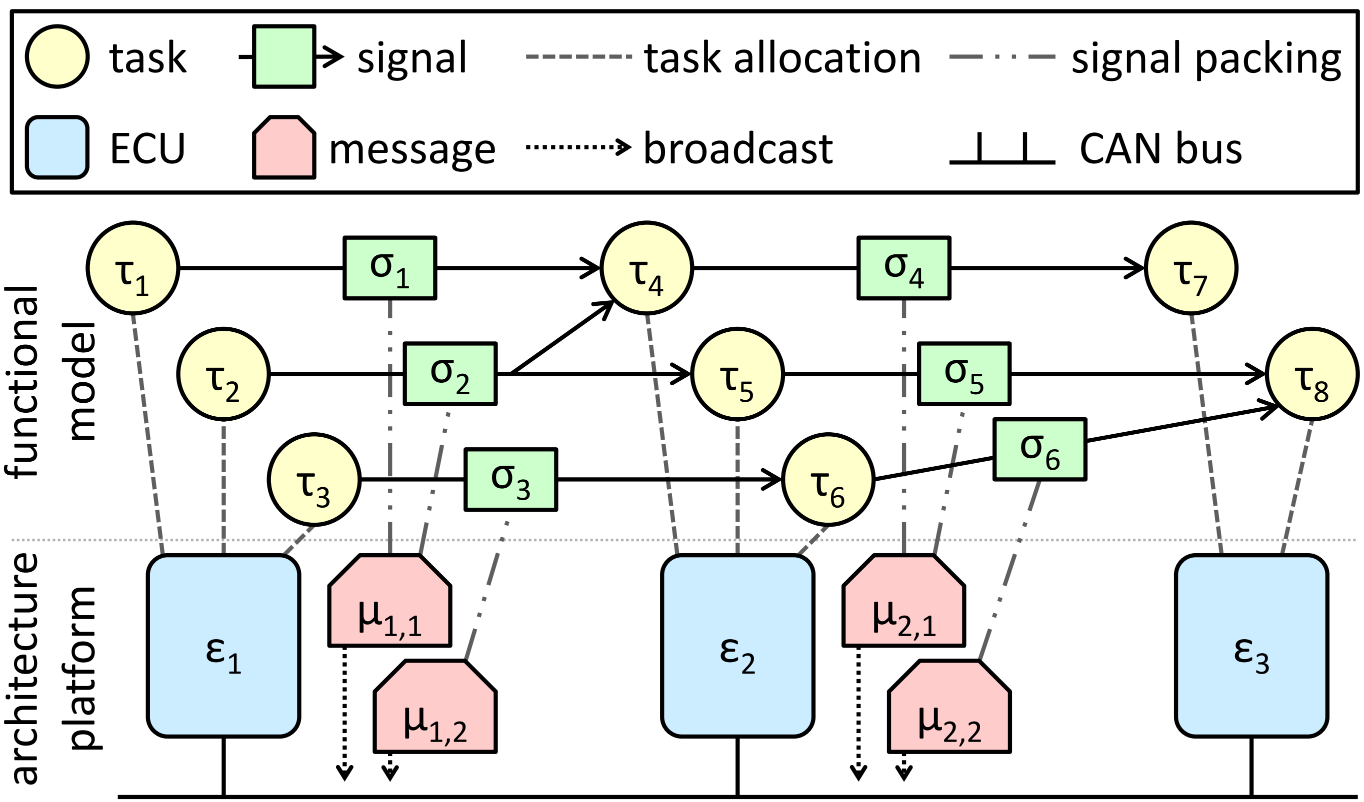
We would like to thank Prof. Marco Di Natali (Scuola Superiore Sant'Anna) for contributing the lecture material

Timing Analysis

- What is timing analysis?
- Why is timing analysis needed?
- When is timing analysis done?
- Where is timing analysis done?
- Who performs timing analysis?
- How to perform timing analysis?

Example of Timing Analysis

- What is timing analysis? / Why is timing analysis needed?
- When / Where is timing analysis done?
- Who / How to perform(s) timing analysis?



Outline

- **Introduction to Controller Area Network (CAN)**
- Timing Analysis of Controller Area Network (CAN)
- Generalization to Software Tasks

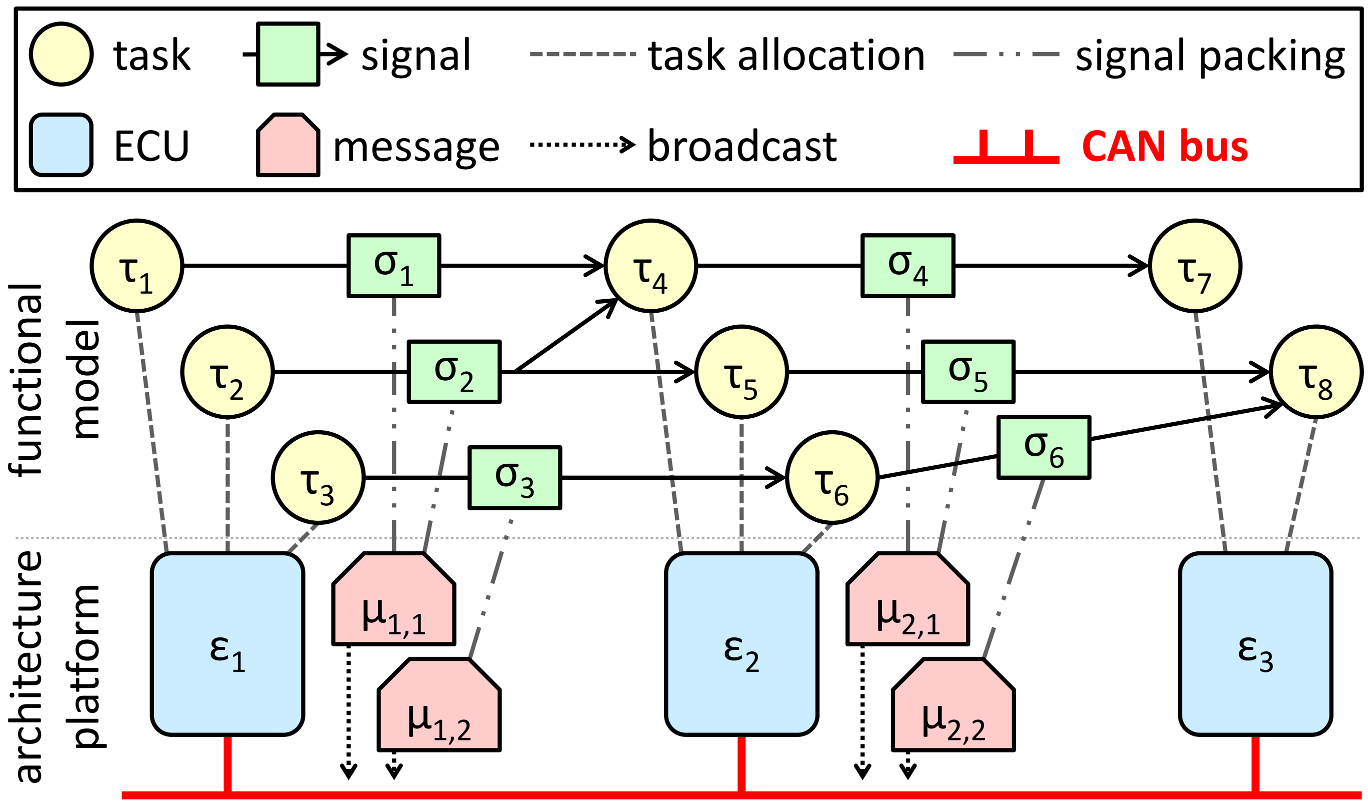
Basic Information of CAN

- Standard
  - http://esd.cs.ucr.edu/webres/can20.pdf
- Serial data bus developed by Bosch in the 80s
  - Support for broadcast and multicast communication
  - Low cost
  - Deterministic resolution of the contention
  - Priority-based arbitration
  - Automotive standard but used also in automation, factory control, avionics, and medical devices
  - Simple two-wire connection
  - Speed up to 1Mb/s
  - Error detection and signaling

Why CAN?

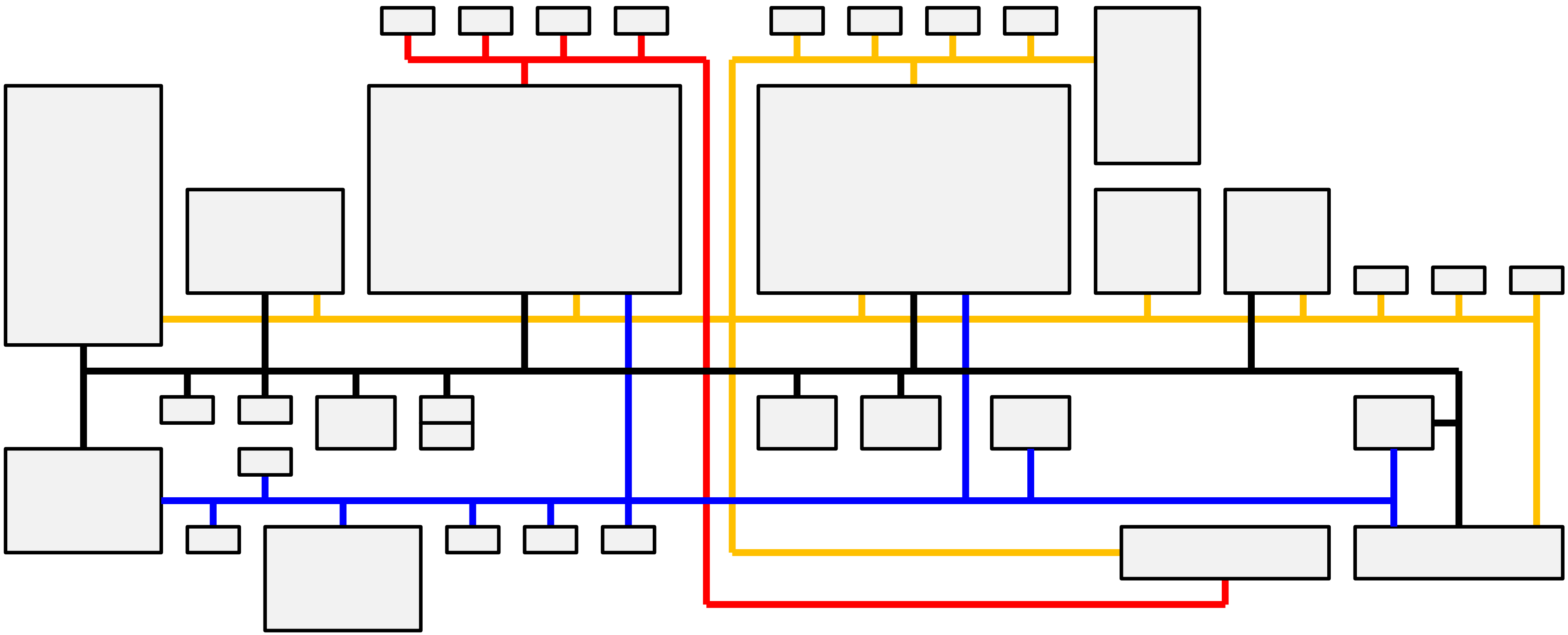
- Why is "old, slow, small-payload" CAN still used?
  - Cheap
  - Simple
  - Guarantee
  - Deterministic
  - Used for long time and still usable
  - More reasons?
- Can we replace it by Ethernet?

CAN in Architecture Model (1/2)



CAN in Architecture Model (2/2)

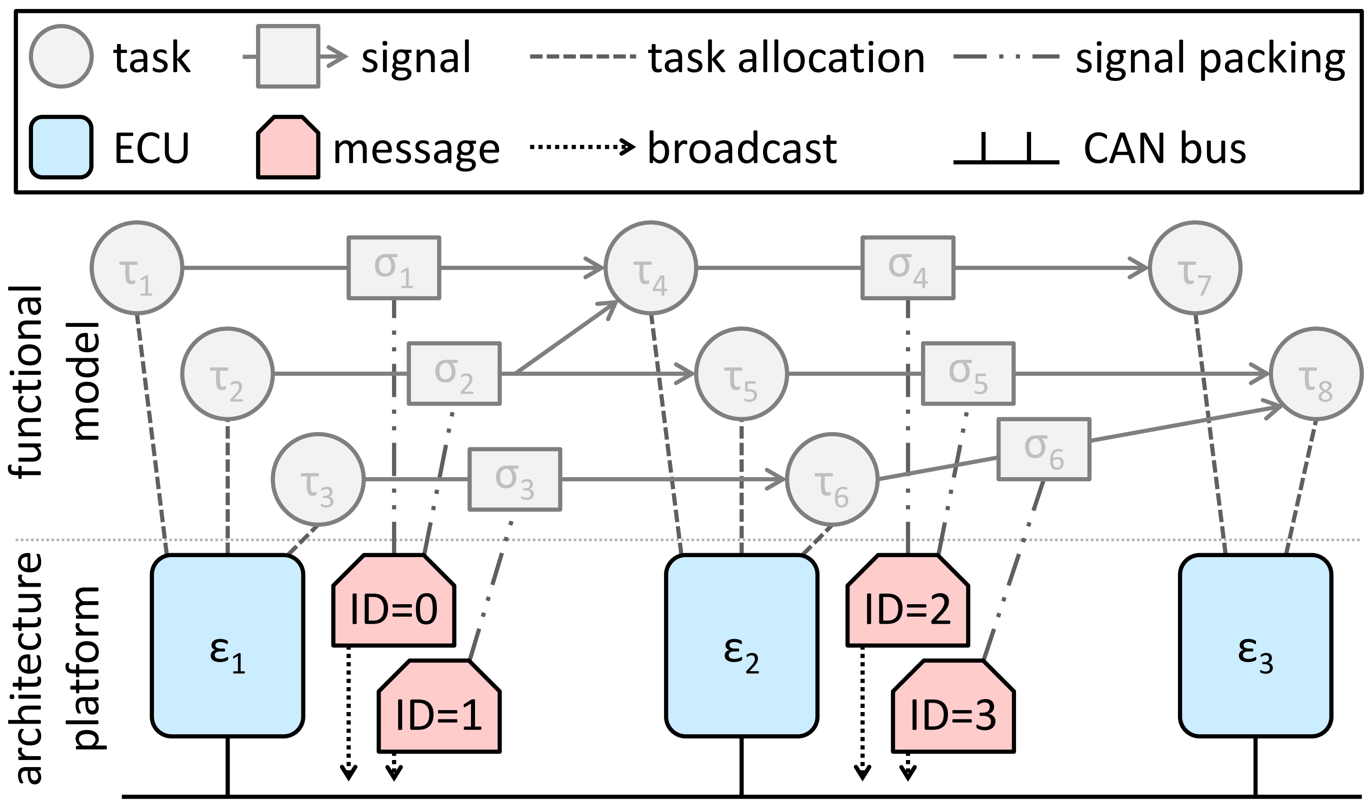
- All the color lines are CAN buses



More Information of CAN

- CAN can be regarded as a Media Access Control (MAC)-layer protocol
- CAN does not require node (or system) configuration information (e.g., address)
  - Flexibility: a node can be added at any time
  - Message delivery: the content is identified by an IDENTIFIER field defining the message content
  - Multicast: all messages are received by all nodes that can filter messages based on their IDs
  - Data consistency: a message is accepted by all nodes or by no node

Example of CAN Multicast



I am the sender of Msg 0 and Msg 1; I will not process Msg 2 and Msg 3

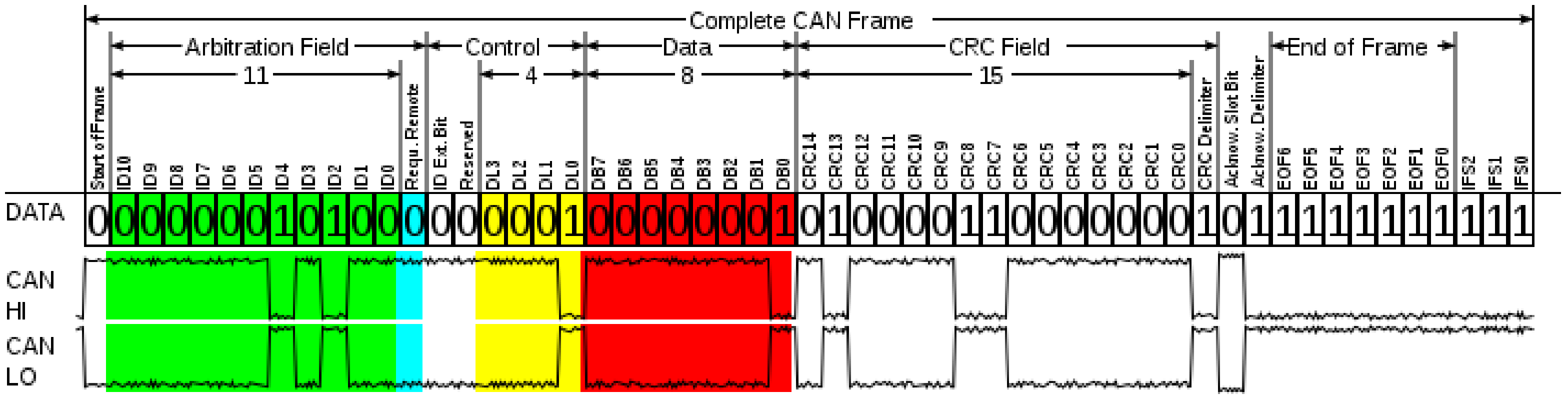
I will process Msg 0 and Msg 1; I am the sender of Msg 2 and Msg 3

I will not process Msg 0 and Msg 1; I will process Msg 2 and Msg 3

Frame Types of CAN

- ❑ **DATA FRAME**
  - ➢ Carries regular data
- ❑ REMOTE FRAME
  - ➢ Used to request the transmission of a DATA FRAME with the same ID
- ❑ ERROR FRAME
  - ➢ Transmitted by any unit detecting a bus error
- ❑ OVERLOAD FRAME
  - ➢ Used to force a time interval in between frame transmissions

Base Data Frame of CAN



Field name	Length (Bits)	Purpose
Start-of-frame	1	Denotes the start of frame transmission
Identifier	11	A (unique) identifier which also represents the message priority
Remote transmission request (RTR)	1	Must be dominant (0) for data frames and recessive (1) for remote request frames
Identifier extension bit (IDE)	1	Must be dominant (0) for base frame format with 11-bit identifiers
Reserved bits (r0)	1	Reserved bit. Must be dominant (0), but accepted as either dominant or recessive.
Data length code (DLC)	4	Number of bytes of data (0–8 bytes)
Data field	0--64	Data to be transmitted (length in bytes dictated by DLC field)
CRC	15	Cyclic redundancy check
CRC delimiter	1	Must be recessive (1)
ACK slot	1	Transmitter sends recessive (1) and any receiver can assert a dominant (0)
ACK delimiter	1	Must be recessive (1)
End-of-frame (EOF)	7	Must be recessive (1)

https://en.wikipedia.org/wiki/CAN_bus

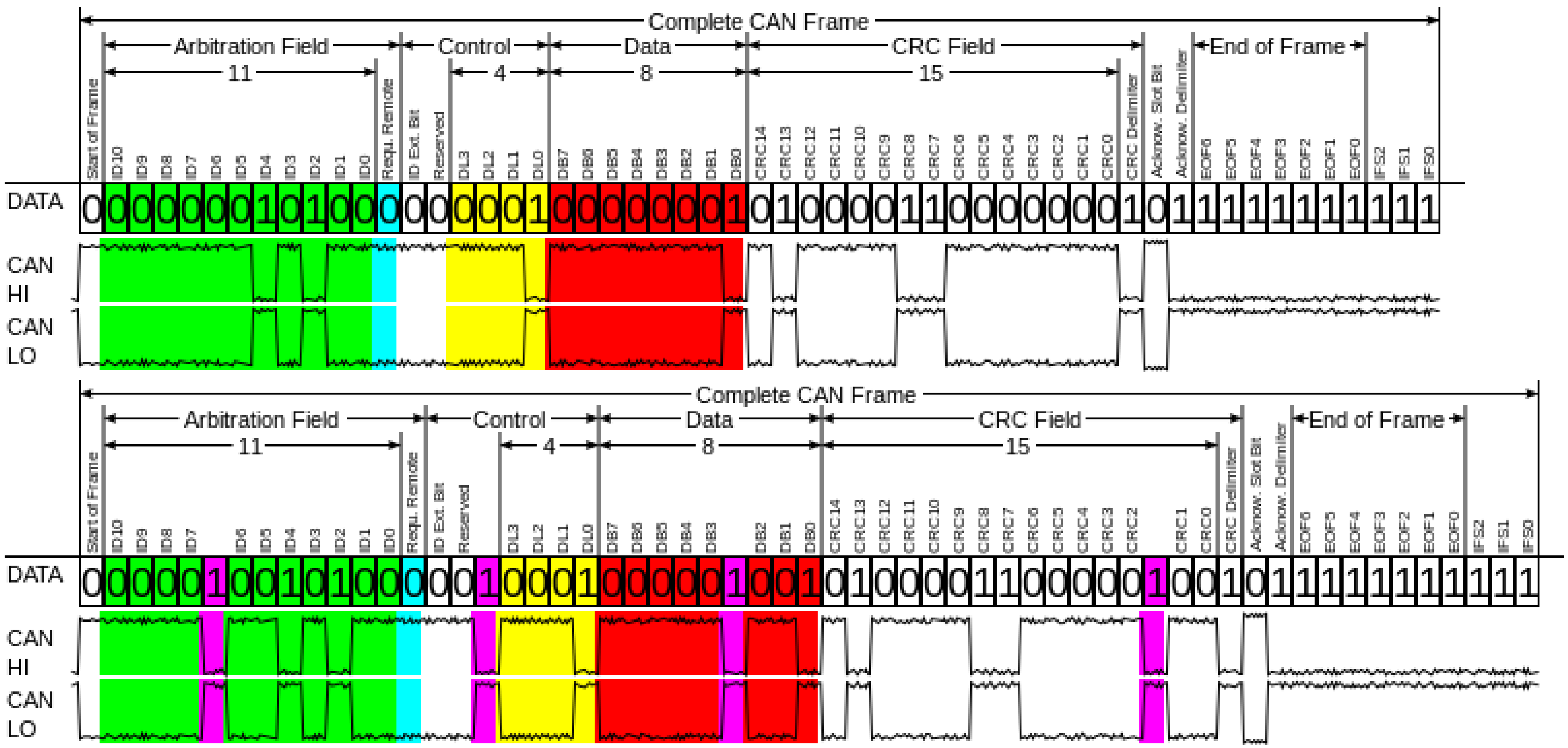
Extended Data Frame of CAN

Field name	Length (Bits)	Purpose
Start-of-frame	1	Denotes the start of frame transmission
Identifier A	11	First part of the (unique) identifier which also represents the message priority
Substitute remote request (SRR)	1	Must be recessive (1)
Identifier extension bit (IDE)	1	Must be recessive (1) for extended frame format with 29-bit identifiers
Identifier B	18	Second part of the (unique) identifier which also represents the message priority
Remote transmission request (RTR)	1	Must be dominant (0) for data frames and recessive (1) for remote request frames
Reserved bits (r1, r0)	2	Reserved bits which must be set dominant (0), but accepted as either dominant or recessive
Data length code (DLC)	4	Number of bytes of data (0–8 bytes)
Data field	0--64	Data to be transmitted (length in bytes dictated by DLC field)
CRC	15	Cyclic redundancy check
CRC delimiter	1	Must be recessive (1)
ACK slot	1	Transmitter sends recessive (1) and any receiver can assert a dominant (0)
ACK delimiter	1	Must be recessive (1)
End-of-frame (EOF)	7	Must be recessive (1)

https://en.wikipedia.org/wiki/CAN_bus

Bit Stuffing of CAN

- Any sequence of 5 bits of the same type requires the addition of an opposite type bit by the transmitter (and removal from the receiver)



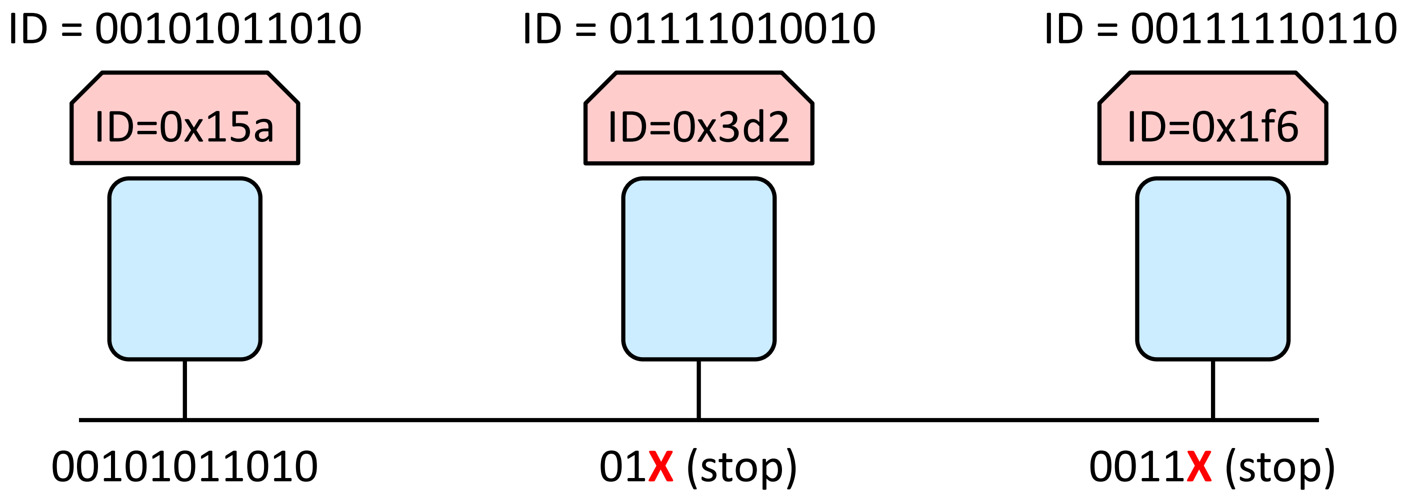
https://en.wikipedia.org/wiki/CAN_bus

Data Efficiency of CAN

- Worst-case frame length of a base data frame
  - 64 bits: data field / 44 bits: other fields / 3 bits: inter-frame spacing
  - 24 bits: bit stuffing
    - 64 bit (in the data field) and 34 bits (in the other fields) are subject to stuffing
    - floor ((64 + 34 - 1) / 4) = 24
    - Why divided by 4, not 5? Why subtracted by 1? Why floor function?
      - 11111000011110000111 1…
  - Total 135 bits
- Best-case data efficiency (without bit stuffing)
  - 64 / (64 + 44 + 3) = 0.577
- Worst-case data efficiency (with bit stuffing)
  - 1 / (8 + 44 + 3 + 10) = 0.015
    - 10 is from "floor ((8 + 34 - 1) / 4) = 10"

Arbitration of CAN

- All nodes are synchronized on the Start-of-Frame bit
  - i.e., a message needs to wait until the bus is idle before it can be entered into arbitration
- The bus behaves as a wired-AND
  - "0" wins the arbitration over "1"



- Is this non-preemptive or preemptive?

Bit Rate of CAN

- The type of arbitration implies that the bit time is at least twice the propagation latency on the bus
- This defines a relation between the maximum bus length and the transmission speed

Bus Length	Bit Rate
25m	1 Mbit/s
50m	800 kbit/s
100m	**500 kbit/s**
250m	250 kbit/s
500m	125 kbit/s
1000m	50 kbit/s
2500m	20 kbit/s
5000m	10 kbit/s

Outline

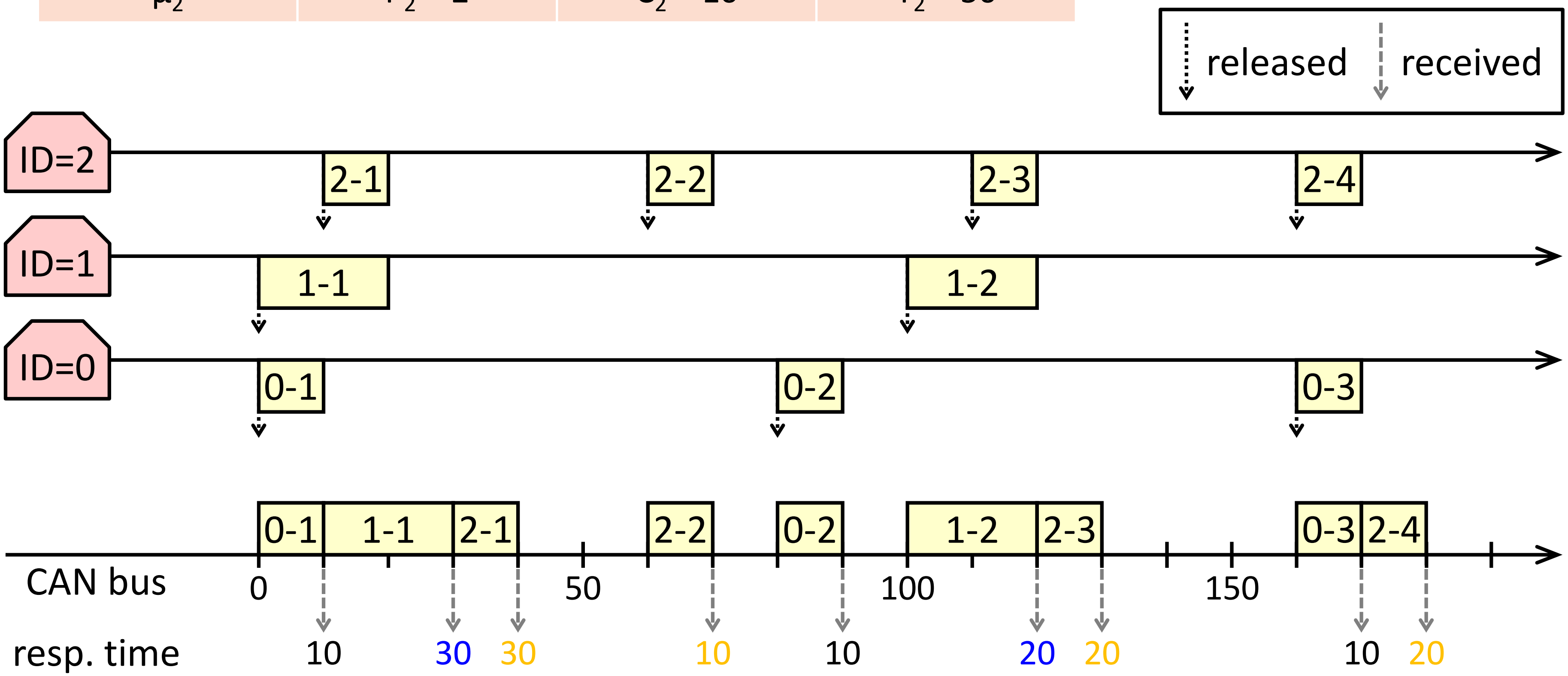
- ❑ Introduction to Controller Area Network (CAN)
- ❑ **Timing Analysis of Controller Area Network (CAN)**
- ❑ Generalization to Software Tasks

Problem Formulation

- Given a set of **periodic** messages: $\mu_0, \mu_1, \ldots, \mu_{n-1}$
  - **Unique** ID (=priority): $P_0, P_1, \ldots, P_{n-1}$
    - $P_i < P_j$ if and only if μ_i has a higher priority than μ_j
  - Transmission Time: $C_0, C_1, \ldots, C_{n-1}$
    - Number of bits divided bit rate of the CAN bus
  - Period: $T_0, T_1, \ldots, T_{n-1}$
- Compute the worst-case response time R_i for each message μ_i
  - Worst-case response time: the longest time from being released (ready to be transmitted) to being received (transmitted completely)
  - Note that
    - We assume that there is no jitter (the time from initiating to being released)
    - Each message has many instances
    - All messages are not synchronized (no fixed alignment --- check later slides)
- Can we underestimate or overestimate R_i?

Example #1: First Example

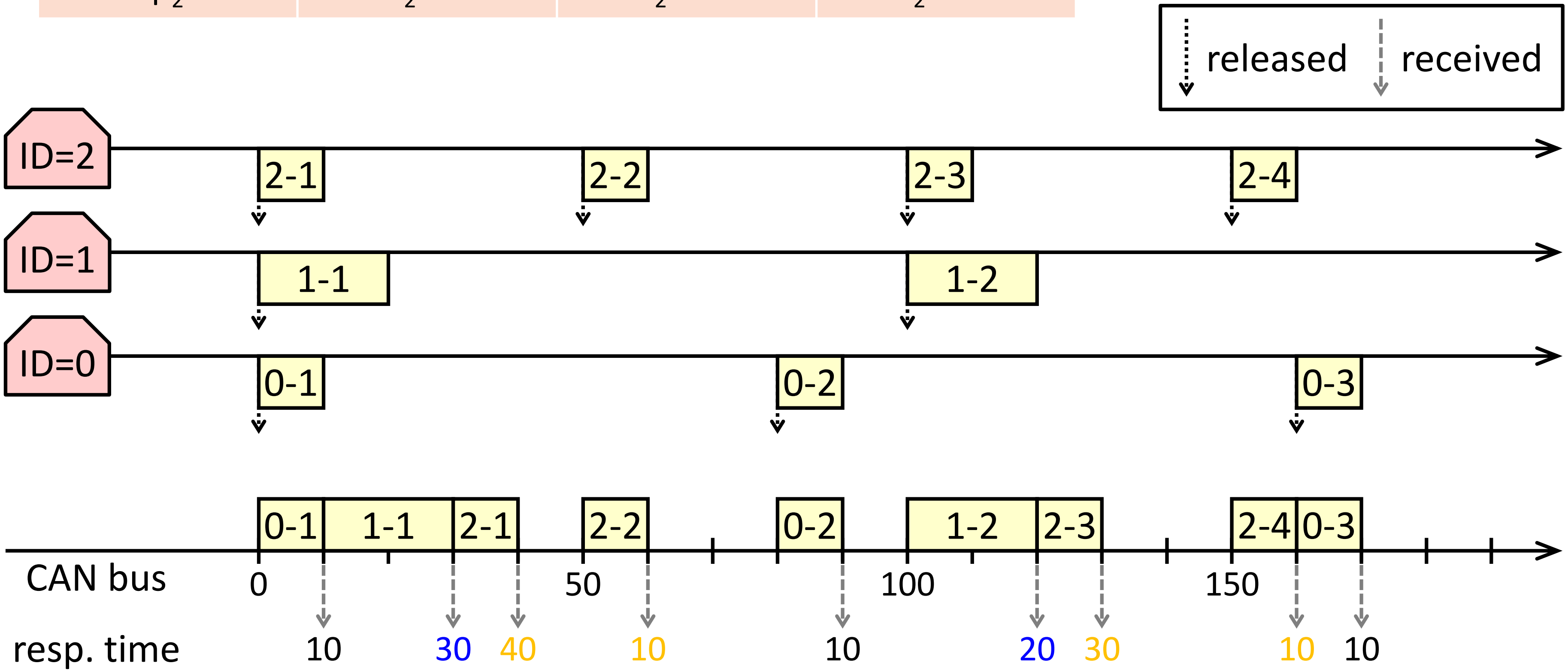
Message	Unique ID (Priority)	Transmission Time	Period
μ_0	$P_0 = 0$	$C_0 = 10$	$T_0 = 80$
μ_1	$P_1 = 1$	$C_1 = 20$	$T_1 = 100$
μ_2	$P_2 = 2$	$C_2 = 10$	$T_2 = 50$



$R_0 = 10?$ $R_1 = 30?$ $R_2 = 30?$

Example #2: "Alignment" Matters

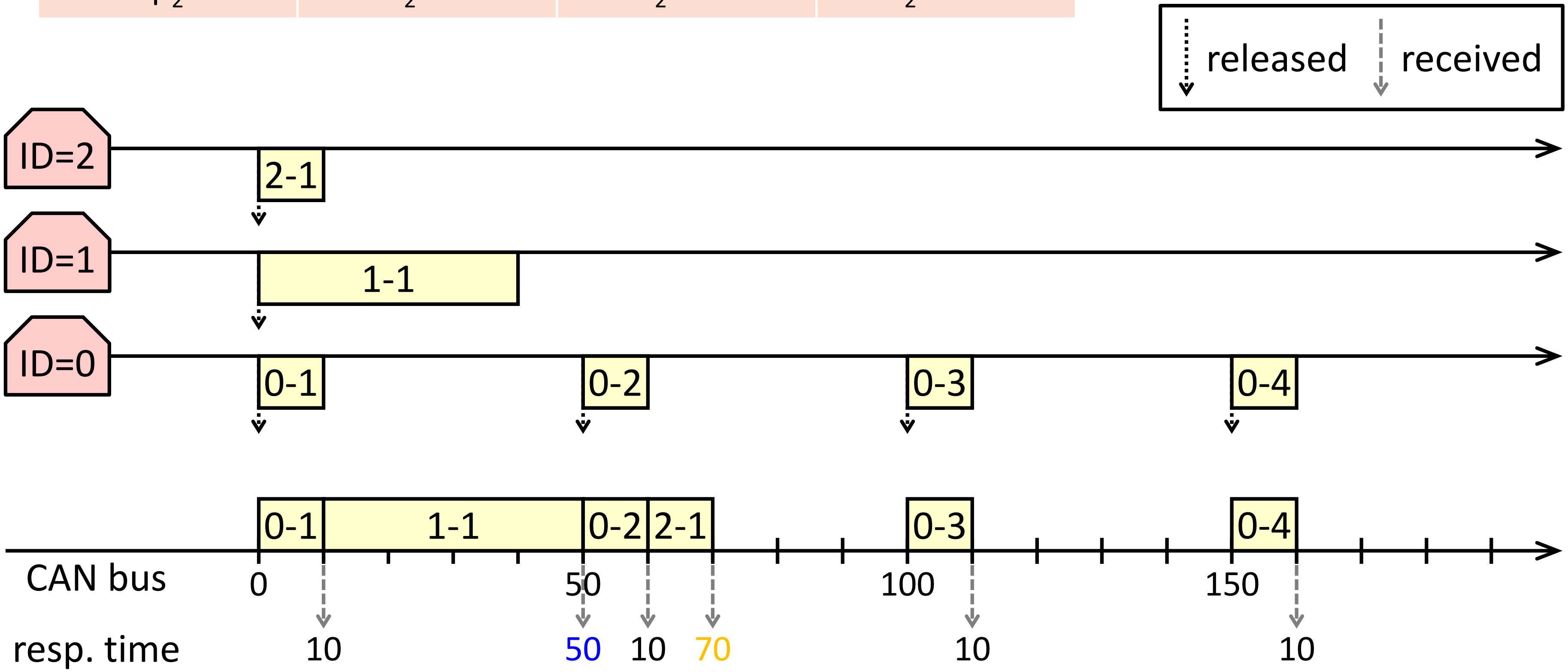
Message	Unique ID (Priority)	Transmission Time	Period
μ_0	$P_0 = 0$	$C_0 = 10$	$T_0 = 80$
μ_1	$P_1 = 1$	$C_1 = 20$	$T_1 = 100$
μ_2	$P_2 = 2$	$C_2 = 10$	$T_2 = 50$



$R_0 = 10?$ $R_1 = 30?$ $R_2 = 40?$

Example #3: Losers Need to Wait

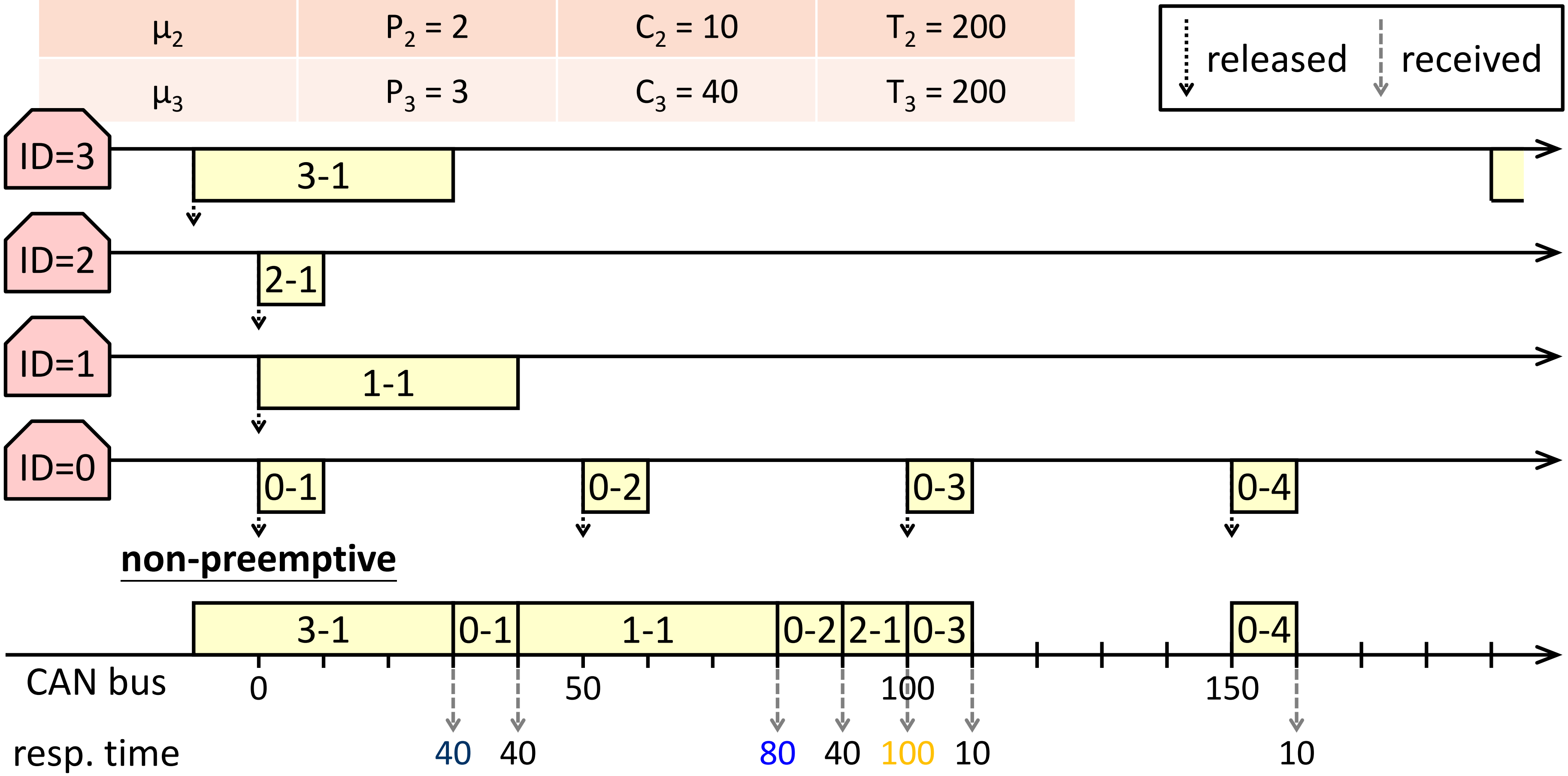
Message	Unique ID (Priority)	Transmission Time	Period
μ_0	$P_0 = 0$	$C_0 = 10$	$T_0 = 50$
μ_1	$P_1 = 1$	$C_1 = 40$	$T_1 = 200$
μ_2	$P_2 = 2$	$C_2 = 10$	$T_2 = 200$



$R_0 = 10$? $R_1 = 50$? $R_2 = 70$?

Example #4: Non-Preemption

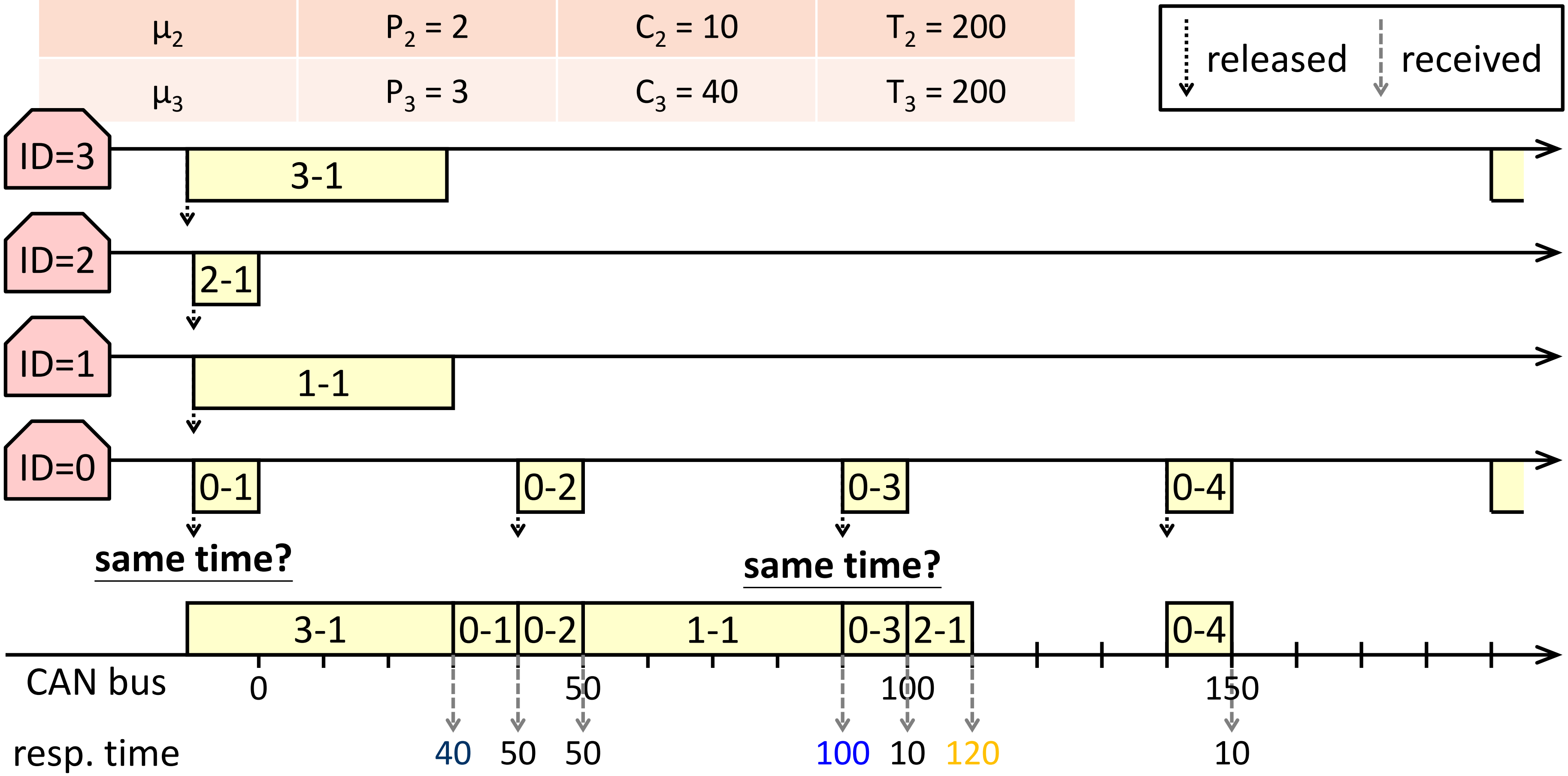
Message	Priority	Trans. Time	Period
μ_0	$P_0 = 0$	$C_0 = 10$	$T_0 = 50$
μ_1	$P_1 = 1$	$C_1 = 40$	$T_1 = 200$
μ_2	$P_2 = 2$	$C_2 = 10$	$T_2 = 200$
μ_3	$P_3 = 3$	$C_3 = 40$	$T_3 = 200$



$R_0 = 40$? $R_1 = 80$? $R_2 = 100$? $R_3 = 40$?

Example #5: Even Worse

Message	Priority	Trans. Time	Period
μ_0	$P_0 = 0$	$C_0 = 10$	$T_0 = 50$
μ_1	$P_1 = 1$	$C_1 = 40$	$T_1 = 200$
μ_2	$P_2 = 2$	$C_2 = 10$	$T_2 = 200$
μ_3	$P_3 = 3$	$C_3 = 40$	$T_3 = 200$



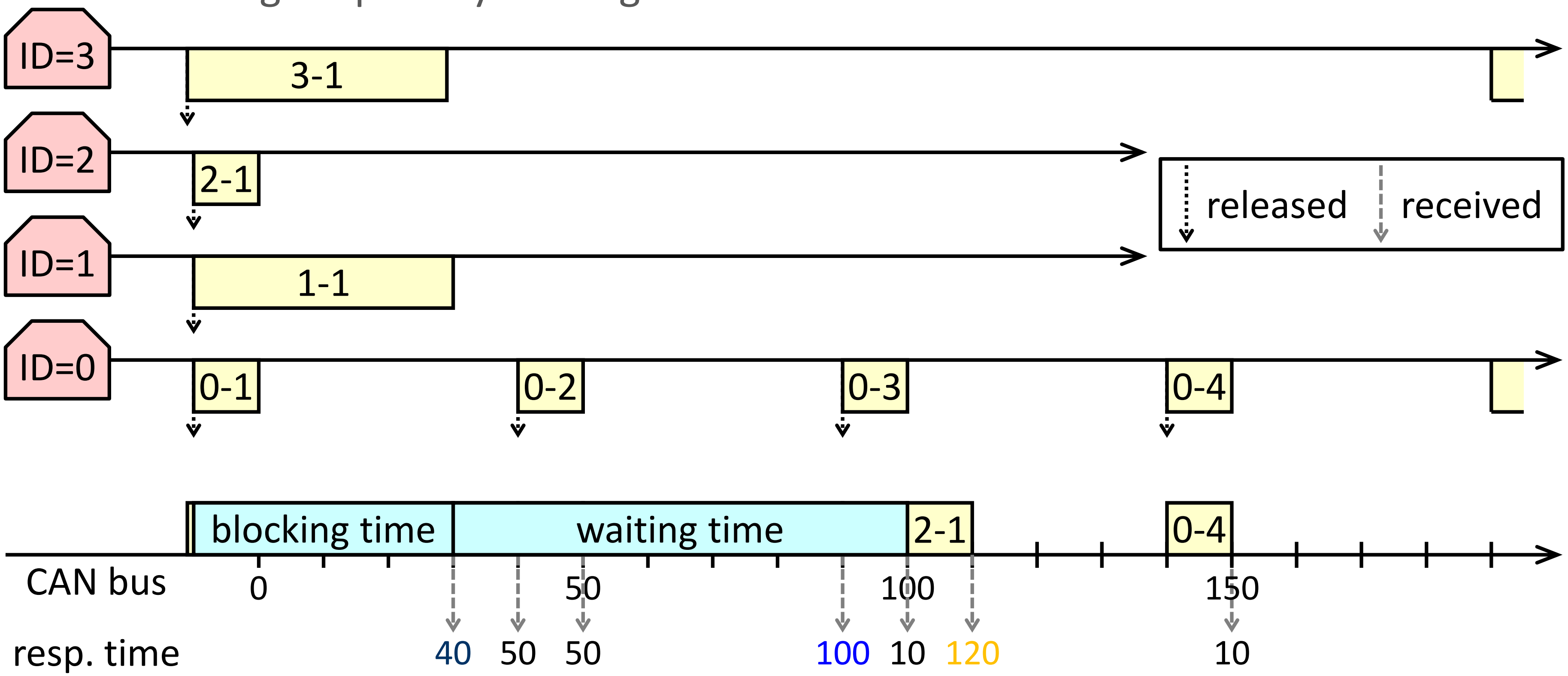
$R_0 = 50$ $R_1 = 100$ $R_2 = 120$ $R_3 = 40?$

Constraint

- The following analysis is applicable if
  - For each message, the computed worst-case response time does not exceed the period
  - i.e., for each μ_i, $R_i \le T_i$
- Technical reason
  - If $R_i > T_i$, the following math will not work
    - Will revisit this constraint again
- Practical reason
  - The information is usually not so useful after T_i
    - If a message is "vehicle speed", do you want use an old instance or an updated instance?

Worst-Case Scenario of μ_i

- Let's focus on the worst-case of a message μ_i
  - The longest lower or same priority message starts to be transmitted just before μ_i is released
  - All higher priority messages are released at the same time



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

Equation of R_i

- $$\underbrace{Q_i}_{(1)} = \underbrace{B_i}_{(2)} + \underbrace{\underbrace{\sum_{(\text{for all } j,\ P_j<P_i)}}_{(3)} \underbrace{\left\lceil \frac{Q_i+\tau}{T_j} \right\rceil}_{(4)} \underbrace{C_j}_{(5)}}_{(6)}$$
- $$R_i = Q_i + \underbrace{C_i}_{(7)}$$

(1): (2)+(6)

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

(3): index set of all higher priority messages

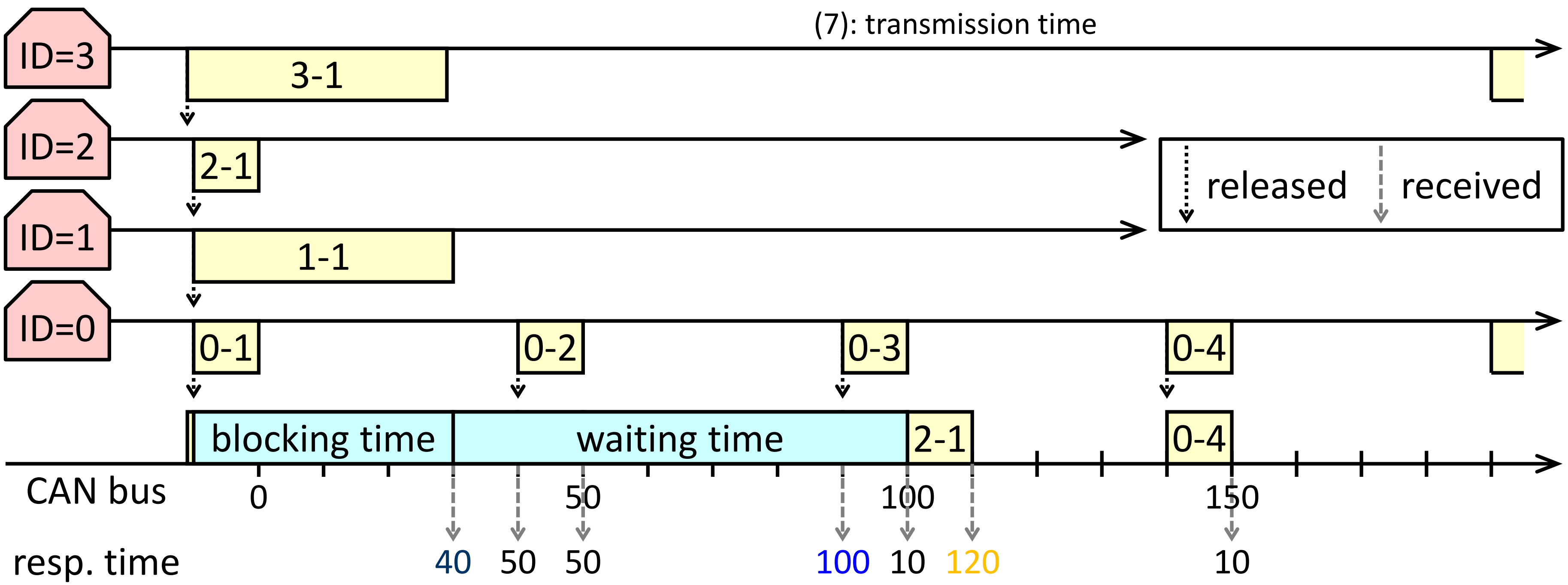
(4): max number of queued instances of message j within (1)

τ: transmission of one bit (Slide P30 for the reason)

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

(6): waiting time

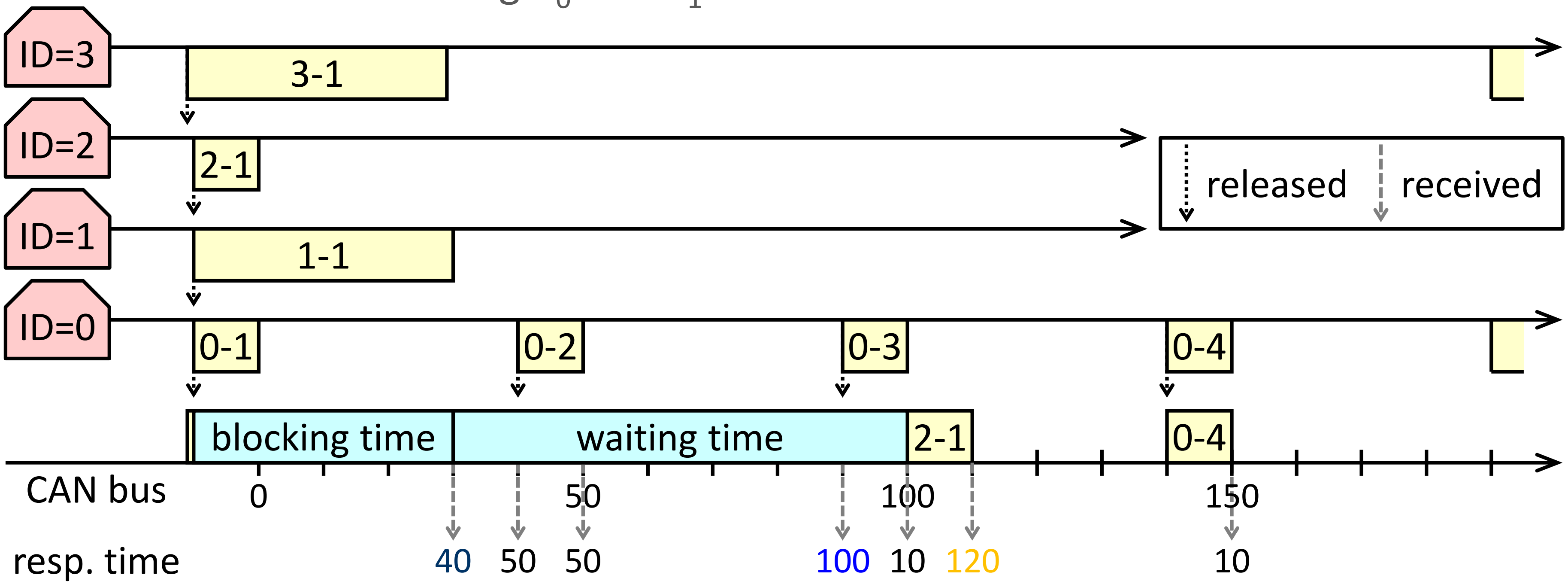
(7): transmission time



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

Checking R_2

- $Q_i = B_i + \sum_{(\text{for all } j,\ P_j<P_i)} \left\lceil \frac{Q_i + \tau}{T_j} \right\rceil C_j$
  - $110 = 40 + \lceil(110+\tau)/50\rceil\ 10 + \lceil(110+\tau)/200\rceil\ 40$
- $R_2 = Q_2 + C_2$
  - $120 = 110 + 10$
  - Exercises: checking R_0 and R_1



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

Computation of R_i

- $Q_i = B_i + \sum_{(\text{for all } j,\ P_j<P_i)} \left\lceil \frac{Q_i + \tau}{T_j} \right\rceil C_j$
  - Right-Hand-Side (RHS) is a monotonic non-decreasing function of Q_i
- $R_i = Q_i + C_i$
- Algorithm: for all i
  - Set $Q_i = B_i$
  - Iteration
    - Compute RHS
    - If RHS + $C_i > T_i$
      - Stop (i.e., break) → constraint violation (the system is not schedulable)
    - If Q_i == RHS
      - $R_i = Q_i + C_i$
      - Stop (i.e., break) → compute R_i successfully
    - Otherwise
      - Q_i = RHS
      - Repeat (i.e., continue)

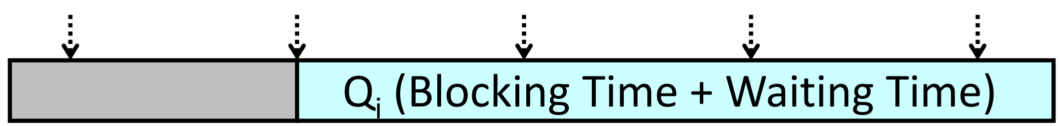
Message	Priority	Trans. Time	Period
μ_0	$P_0 = 0$	$C_0 = 10$	$T_0 = 50$
μ_1	$P_1 = 1$	$C_1 = 40$	$T_1 = 200$
μ_2	$P_2 = 2$	$C_2 = 10$	$T_2 = 200$
μ_3	$P_3 = 3$	$C_3 = 40$	$T_3 = 200$

- Exercises: computing R_0, R_1, R_2, and R_3

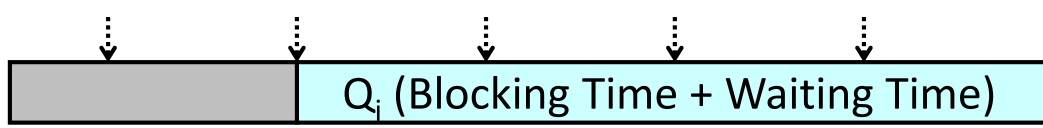
Revisit the Constraint

- $Q_i = B_i + \sum_{(\text{for all } j,\ P_j<P_i)} \left\lceil \frac{Q_i + \tau}{T_j} \right\rceil C_j$
  - Right-Hand-Side (RHS) is a monotonic non-decreasing function of Q_i
- $R_i = Q_i + C_i$

- Constraint: the analysis is applicable if
  - For each message, the computed worst-case response time does not exceed the period
  - i.e., for each μ_i, $R_i \leq T_i$
- Question: how many **<u>queued</u>** instances of message j within Q_i?
  - Example: $Q_i = 100$, $T_j = 30$
    - 4 or 5 or 6?



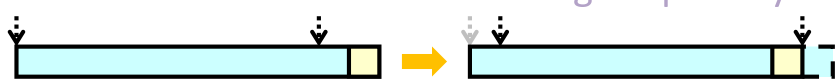
  - Example: $Q_i = 100$, $T_j = 25$
    - 4 or 5 or 6? Why τ?



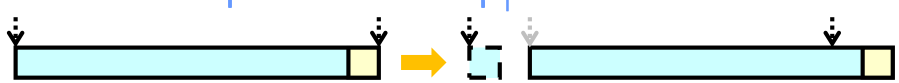
Backup: Critical Instant Theorem

- The worst-case response time R_i of a message μ_i occurs when it is released with all higher priority messages at the same time
  - It is a necessary but not sufficient condition
  - Case 1: a higher priority message is released later
    - It does not increase R_i as
      - μ_i is possible to be transmitted before the first instance of the higher priority message is released

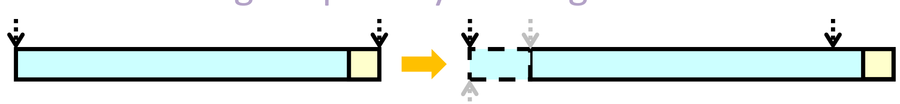
      - It is possible to have fewer instances of the higher priority message during the waiting time of μ_i



  - Case 2: a higher priority message is released earlier
    - Case 2-1: the CAN bus is idle at some point before μ_i is released
      - It will not become worse



    - Case 2-2: the CAN bus is always busy before μ_i is released
      - We can shift μ_i to be released with the higher priority message at the same time



Backup: Same Priority Message

- Why do we need to consider "the same priority message" (another instance of the same message) in B_i?
  - Given "the constraint", it seems to be unnecessary
    - However, we cannot prove a property ("the constraint") from assuming the property ("the constraint")
  - Note
    - The reference paper is a fix of another paper that people believe it is true for many years…

Last Example

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

Message	Priority	Trans. Time	Period
μ_0	$P_0 = 0$	$C_0 = 4$	$T_0 = 10$
μ_1	$P_1 = 1$	$C_1 = 4$	$T_1 = 13$
μ_2	$P_2 = 2$	$C_2 = 4$	$T_2 = 13$

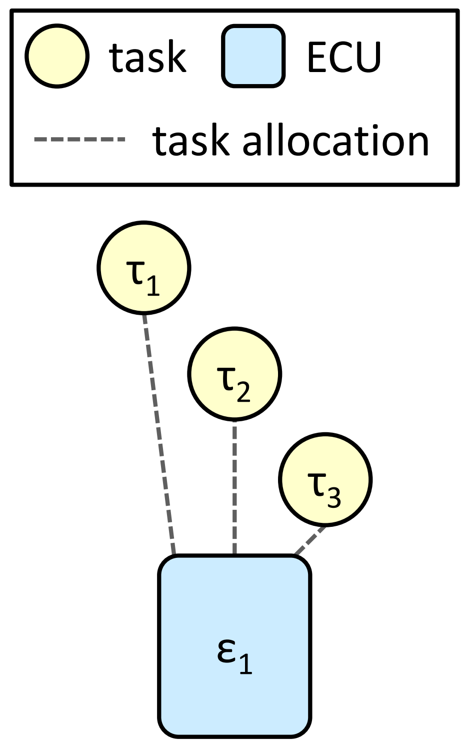
- If the B_i is replaced by maximum C_i with lower priority
  - The analysis will become schedulable

Outline

- ❑ Introduction to Controller Area Network (CAN)
- ❑ Timing Analysis of Controller Area Network (CAN)
- ❑ **Generalization to Software Tasks**

Software Tasks on ECU

- Similar scheduling can be applied to an Electronic Control Unit (ECU)
  - It is usually preemptive
- $R_i = C_i + \sum_{(\text{for all } j,\ P_j < P_i)} \lceil R_i / T_j \rceil C_j$



References

- R. I. Davis, A. Burns, R. J. Bril, and J. J. Lukkien, "Controller Area Network (CAN) schedulability analysis: Refuted, revisited and revised," in Real-Time Systems, vol. 35, no. 3, pp. 239--272, Apr. 2007.
- C. L. Liu and J. W. Layland, "Scheduling algorithms for multiprogramming in a hard-real-time environment," in Journal of ACM, vol. 20, no. 1, pp. 46--61, Jan. 1973.

Q&A