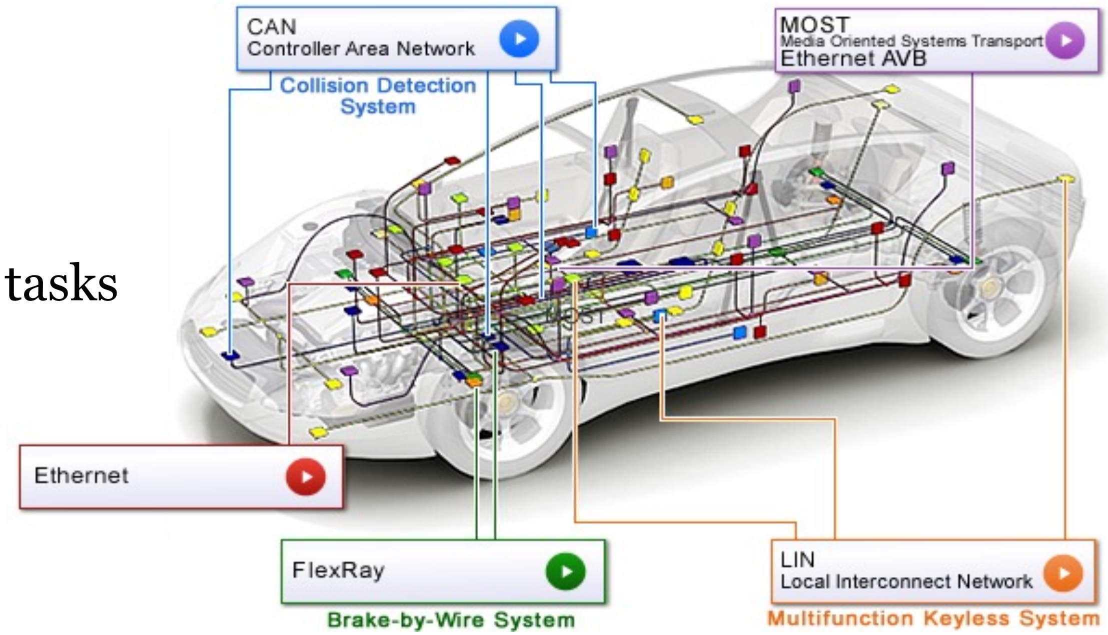
Lectures 6

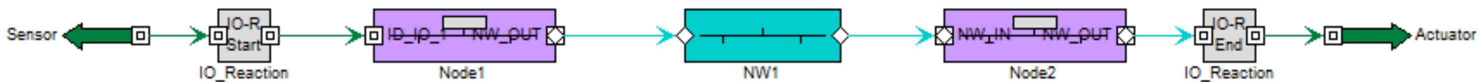
# Schedulability Analysis
## Part 2

*Saad Mubeen*

# Outline




- Response-time analysis (RTA) of tasks
  - Basic RTA
  - RTA with blocking
  - RTA with jitter
  - RTA with Offsets
- RTA of Controller Area Network (CAN) messages
- Practical limitations in CAN Controllers and their effect on RTA for CAN
- Holistic RTA (for distributed systems)
- Data-propagation delay analysis
  - Single node systems
  - Distributed systems (End-to-end data-propagation delay analysis)




Figures courtesy of: http://www.cvel.clemson.edu/
http:// www.renesas.eu

# Controller Area Network (CAN): facts and motivation

- Modern premium cars
  - 70-100 ECUs
  - 5 or more networks
- Controller Area Network (CAN)
  - Speed up to 1 Mbit/s
  - ISO 11898 (1-3)
  - According to CiA, more than 2 billion CAN controllers have been sold
    - 80% have been used in the automotive domain
- Modern heavy trucks
  - 45 ECUs
  - 20 CAN buses
  - over 6000 MSGs
- Higher-level protocols for CAN
  - CANopen, MilCAN, HCAN, AUTOSAR

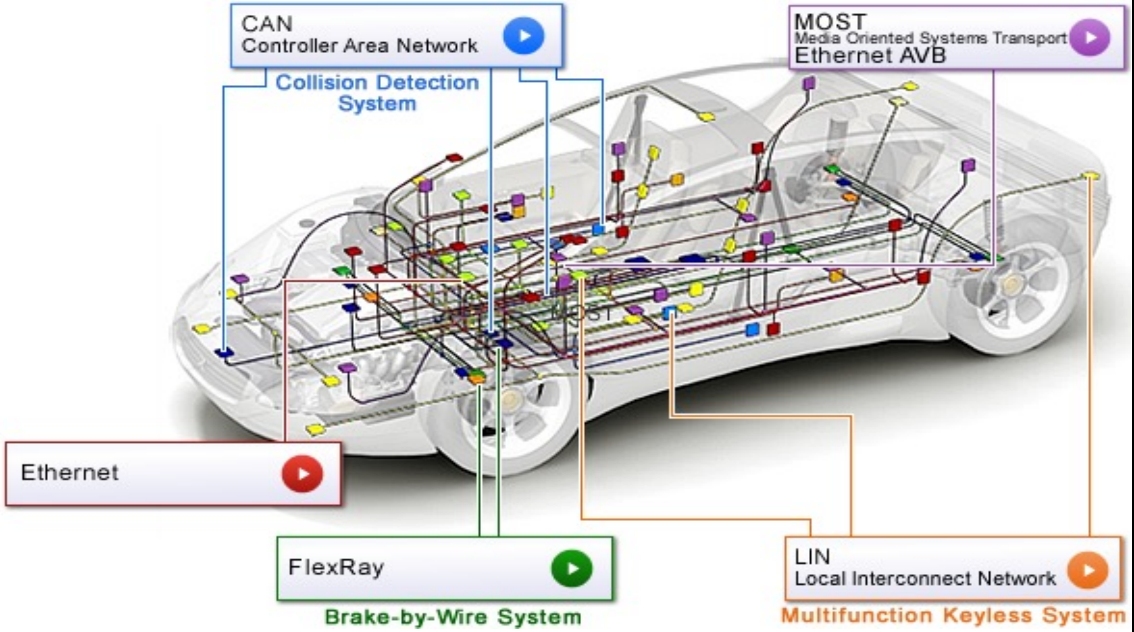



Figure courtesy of www.renesas.eu

Figure courtesy of www.timmo-2-use.org


RTA for CAN

## Response Time of CAN message

$R_i$ = Worst-case response time of a CAN message "*i*"

$R_i$ = Worst-case transmission time of "*i*"

+ Blocking time due to lower priority messages

+ Interference due to higher priority messages

## What about $C_i$?

- **In RTA for tasks, $C_i$ is the "execution time"**
  - For a message, $C_i$ is the **transmission time**

- **All nodes use the same baudrate**
  - Time to send one bit = $\tau_{\mathrm{bit}}$ (Tau-bit)

- **$S_i$ = the maximum size of message payload**
  - $S_i$ is 0…8 bytes

| SOF (Start Of Frame) | Identifier | RTR (Remote Trans-mission Request) | Control | Data | CRC (Cyclic Redun-dancy Check) | CRC Delimiter | ACK Acknow-ledg | ACK Delimiter | EOF (End Of Frame) | IFS (Inter Frame Space) |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 bit | 11 bits | 1 bit | 6 bits | 0-8 bytes | 15 bits | 1 bit | 1 bit | 1 bit | 7 bits | min 3 bits |

- **+ 47 control bits**

## What about $C_i$ (cont)

- **Other source of variability**
  - Bit-stuffing
  - (i.e. adding extra 1 or 0 to avoid six consecutive equals)
- **34 of 47 control-bits + payload are subjected to bit-stuffing**

| SOF (Start Of Frame) | Identifier | RTR (Remote Trans-mission Request) | Contro l | Data | CRC (Cyclic Redun-dancy Check) | CRC Delimiter | ACK Acknow-ledg | ACK Delimit er | EOF (End Of Frame) | IFS (Inter Frame Space) |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 bit | 11 bits | 1 bit | 6 bits | 0-8 bytes | 15 bits | 1 bit | 1 bit | 1 bit | 7 bits | min 3 bits |

- **$34+8*S_i$ bits are subjected to bit-stuffing**
  - Maximum of $\lfloor(34+8S_i -1)/4\rfloor$ stuff-bits

- $C_i = \left( 47 + 8S_i + \left\lfloor(34+8S_i -1)/4\right\rfloor \right)\tau_{bit}$

## What about $B_i$?

- **Blocking = bounded interference from *lower* priority messages**
  - For CAN: $B_i$ is transmission time from 1 lower priority message
  - $B_i = \forall_{j \in lp(i)} \max(C_j)$

- **Safe approximation of blocking**
  - $B_i = \forall_j \max(C_j)$
  - or even safer/simpler: $B_i = 135\ \tau_{bit}$

$$C_i = \left( 47 + 8S_i + \left\lfloor (34+8S_i-1)/4 \right\rfloor \right) \tau_{bit}$$

Put $S_i = 8$

## What about $B_i$?

- **Blocking = bounded interference from *lower* priority messages**
  - For CAN: $B_i$ is transmission time from 1 lower priority message
  - $B_i = \forall_{j \in lp(i)} \max(C_j)$

- **Safe approximation of blocking**
  - $B_i = \forall_j \max(C_j)$
  - or even safer/simpler: $B_i = 135\ \tau_{bit}$

- **Why approximate?**
  - Exact $B_i = \forall_{j \in lp(i)} \max(C_j)$
  - Risk of "unknown" low priority messages
    - E.g. third party equipment
    - Incomplete model for analysis
  - Also: Analysis becomes significantly more complex:
    - Controller Area Network (CAN) schedulability analysis: Refuted, revisited and revised. Robert I. Davis, Alan Burns, Reinder J. Bril, Johan J. Lukkien. J. of Real-Time Systems 2007

## Asynchronous nodes and signal propagation

- **CAN is global priority queue**
  - But: Nodes are not synchronized

- **Maximum delay to observe common event**
  - Bounded by signal propagation delay
  - Remember: Arbitration works since delay is “small enough”
  - i.e. delay is never bigger than $\tau_{\text{bit}}$

- **Signal propagation gives “uncertainty” in observation of arrivals at remote nodes**
  - This uncertainty is called “Jitter”
  - Jitter increases the time-interval during which a high-priority message can interfere
  - This increase is bounded by $\tau_{\text{bit}}$

## Putting in all together I

$$= B_i + \Sigma_{j \in hp(i)} \lceil ( \quad ) / T_j \rceil C_j$$

Jitter:

- Increases the time-interval during which a high-priority message can interfere
- This increase is bounded by $\tau_{bit}$

$${}_i = B_i + \Sigma_{j \in hp(i)} \lceil ( \quad {}_i + \tau_{bit} ) / T_j \rceil C_j$$

## Putting in all together II

$$= B_i + \Sigma_{j \in hp(i)} \lceil ( \quad + \tau_{bit} )/T_j \rceil C_j$$

- **Non-preemtive transmission**
  - "Window of interference" ($w_i$) does NOT include $C_i$
  - I.e. hp(i) messages cannot interfere once transmission has started

$$W^{n+1}_i = B_i + \Sigma_{j \in hp(i)} \lceil (w^n_i + \tau_{bit} )/T_j \rceil C_j$$

- **Instead $C_i$ is added to the "window"**
  - $C_i$ comes after the window of interference

$$R_i = C_i + w_i$$

# RTA for CAN messages with Jitter

Response time analysis must be updated with a release jitter term:

$$J_i$$

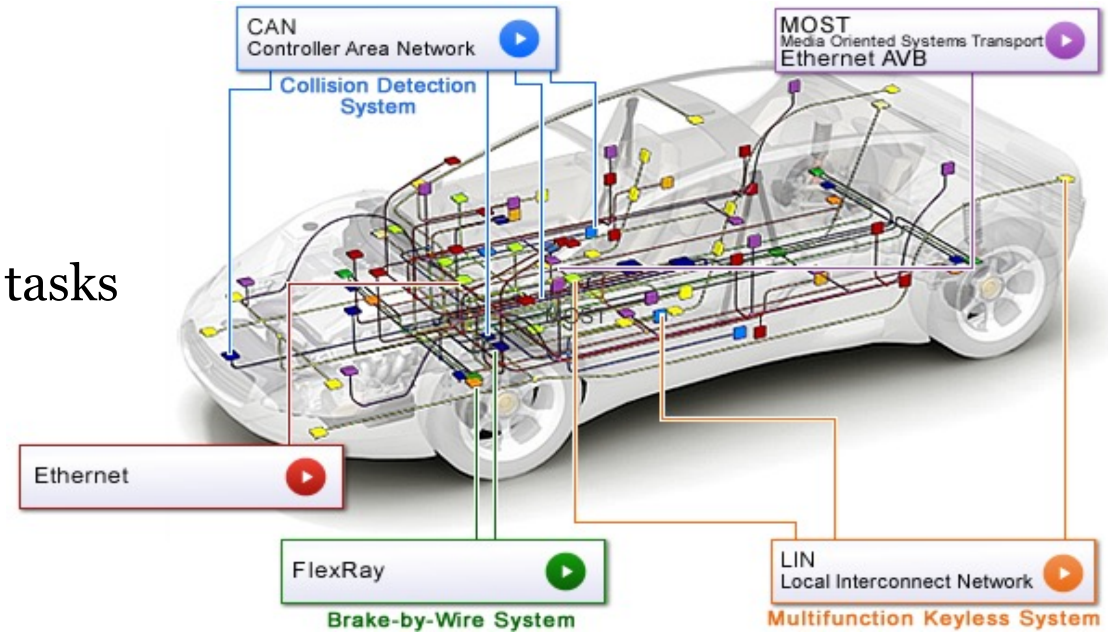
New equation:

$$w_i^{n+1} = B_i + \sum_{\forall j \in hp(i)} \left\lceil \frac{w_i^n + J_j + \tau_{bit}}{T_j} \right\rceil C_j$$

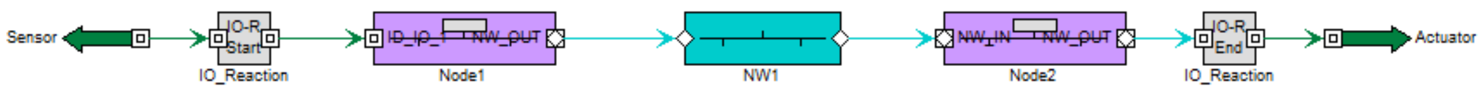
$$R_i = C_i + J_i + w_i$$

MÄLARDALEN UNIVERSITY SWEDEN

# Outline




- Response-time analysis (RTA) of tasks
  - Basic RTA
  - RTA with blocking
  - RTA with jitter
  - RTA with Offsets
- RTA of Controller Area Network (CAN) messages
- Practical limitations in CAN Controllers and their effect on RTA for CAN
- Holistic RTA (for distributed systems)
- Data-propagation delay analysis
  - Single node systems
  - Distributed systems (End-to-end data-propagation delay analysis)




Figures courtesy of: http://www.cvel.clemson.edu/
http:// www.renesas.eu

# Response Time Analysis for Controller Area Network (CAN): Practical Limitations

*Saad Mubeen*

# Practical limitations in CAN Controllers and their effect on RTA for CAN

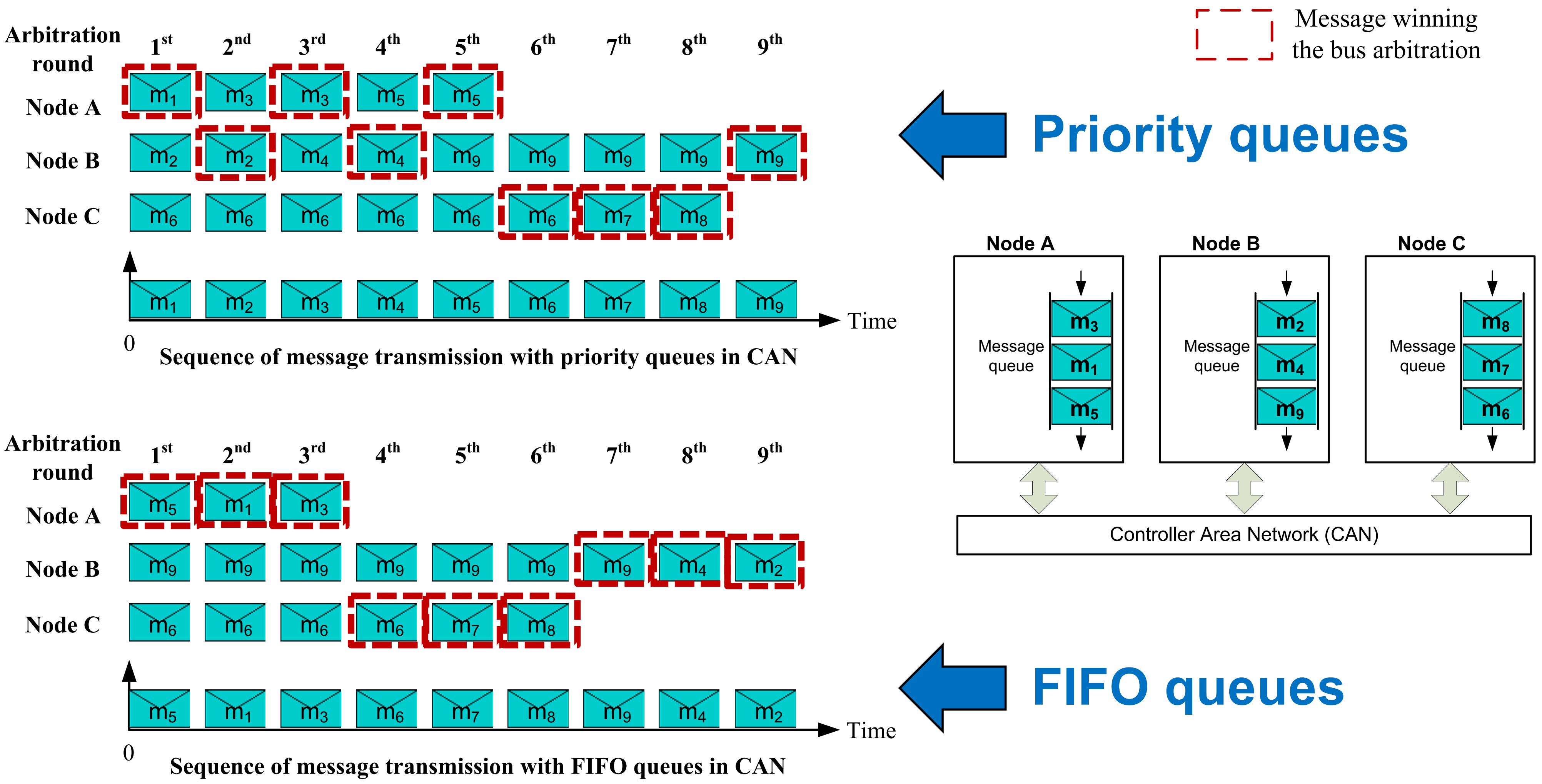
The classical RTA for CAN (that we studied in this lecture) assumes that if there are more than one messages in a node that are ready to be sent then the highest priority message will be sent first.

However, this assumption may be violated due to various types of queuing polices implemented by the CAN device drivers and communications stacks, internal organization and hardware limitations in the CAN controllers.

In addition to these limitations, mixed transmission patterns supported by higher-level protocols can have significant impact on the timing behavior of CAN messages.

- **Queuing policies used in the CAN controllers**
  - Priority
  - First In First Out (FIFO)
- **Buffer limitations in the CAN controllers**
  - Non-abortable transmit buffers
  - Abortable transmit buffers
- **Mixed messages implemented by higher-level protocols**

# Practical limitations in CAN Controllers and their effect on RTA for CAN

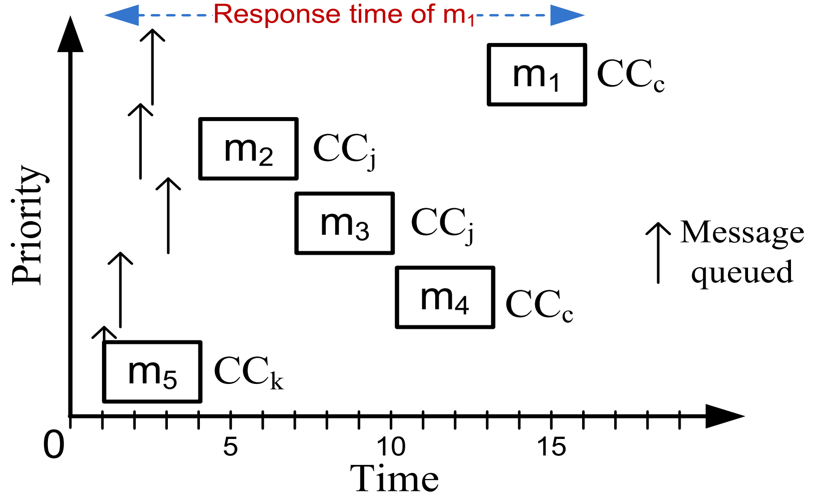



Some CAN controllers implement FIFO queues. In FIFO queues, priorities are often not respected. There may be long buffering delays. This may results in longer response times of messages especially those with higher priorities. RTA for CAN should consider these issues.

# Practical limitations in CAN Controllers and their effect on RTA for CAN

## Buffer limitations in the CAN controllers

- **Non-abortable transmit buffers**

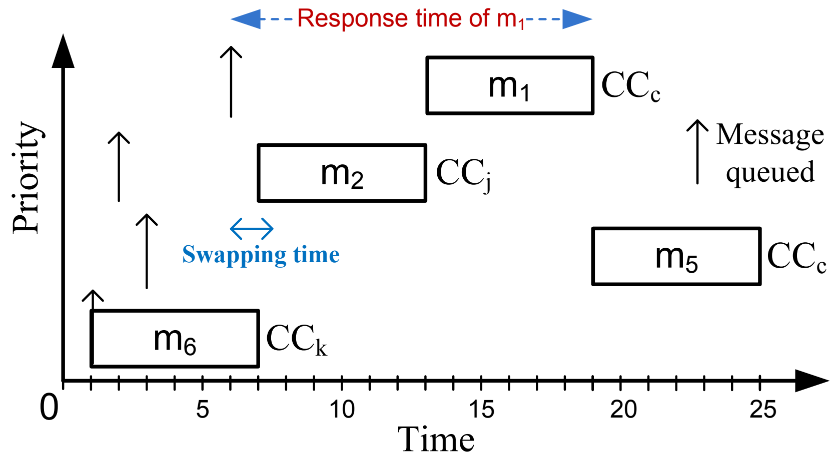



- When the highest prio msg $m_1$ in node $CC_c$ is queued, all trasnsmit buffers in $CC_c$ are occupoed by lower priority messages.
- The CAN controllers do not support transmission abort requests, hence $m_1$ has to wait in the ready queue before the highest priority message in $CC_c$ (say $m_4$) is transmitted thereby vacating a space for $m_1$ in the transmit buffers.
- However, before starting its transmission, $m_4$ can be interefered by higher priority messages from other nodes whose priorities are smaller than $m_1$. This results in an extra delay for $m_1$ other than the normal blocking (e.g., due to $m_5$) and interference from higher priority messages (if any).
- This extra delay should be considered in the worst-case response time of $m_1$.

# Practical limitations in CAN Controllers and their effect on RTA for CAN

**Buffer limitations in the CAN controllers**

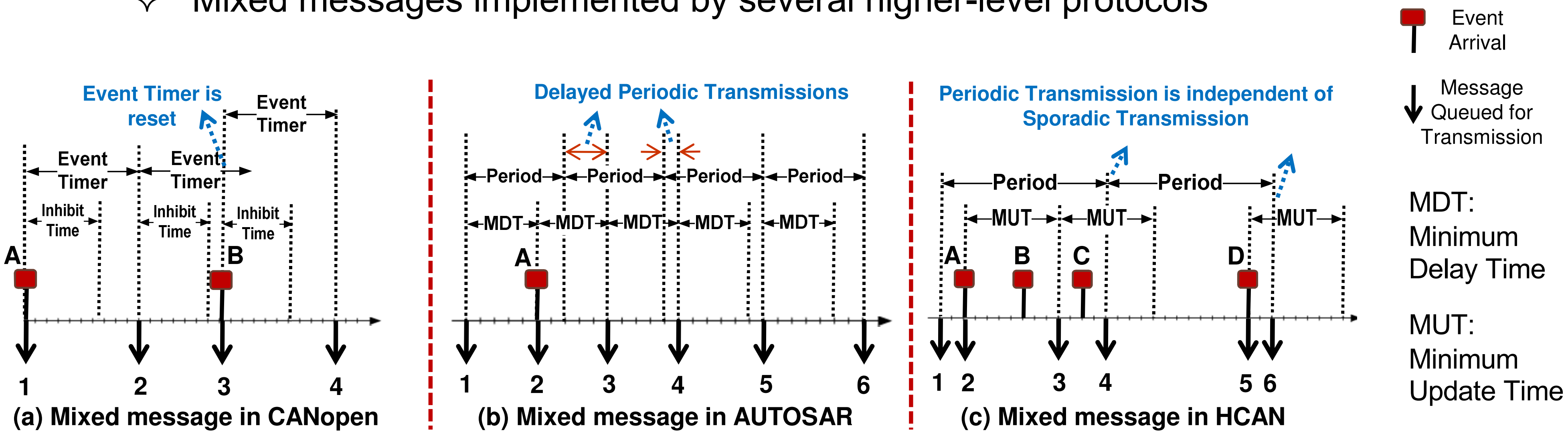
- **Abortable transmit buffers**




- When the highest prio msg $m_1$ in node $CC_c$ is queued, all trasnsmit buffers in $CC_c$ are occupoed by lower priority messages.
- The CAN controllers support transmission abort requests, hence the highest priority message in the transmit buffers of $CC_c$ is swapped with $m_1$.
- If the swapping time is long, it is possible that a lower priority msg (say $m_2$) belonging to another node $CC_j$ may win the bus arbitration and start to transmit.
- Since CAN uses non-preemptive sheduling, $m_2$ cannot be preempted. This results in an extra delay for $m_1$ other than the normal blocking (e.g., due to $m_5$) and interference from higher priority messages (if any).
- This extra delay should be considered in the worst-case response time of $m_1$.

# Practical limitations in CAN Controllers and their effect on RTA for CAN

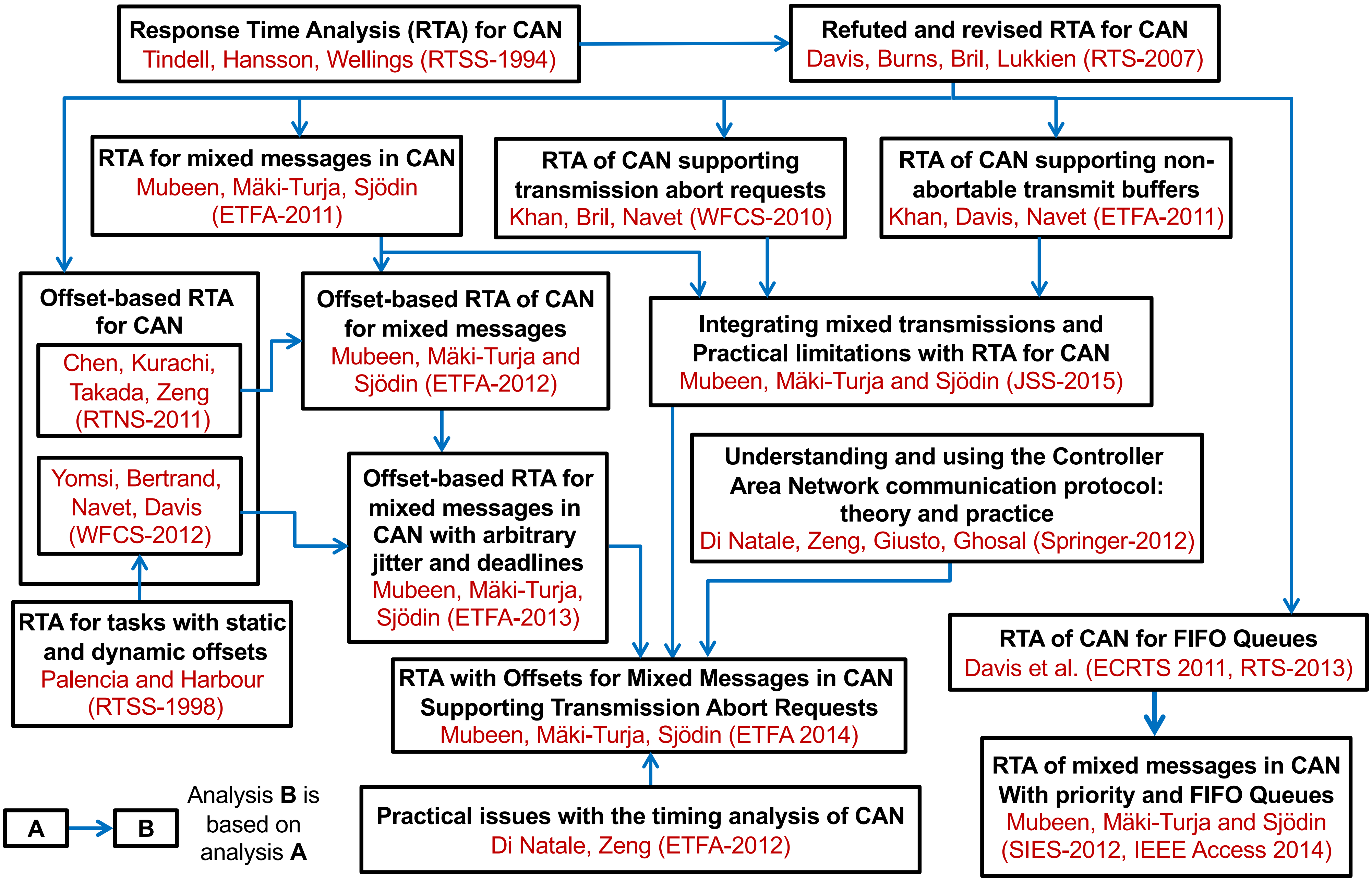
- **Mixed messages are both periodic and sporadic**
  - ✧ Mixed messages implemented by several higher-level protocols



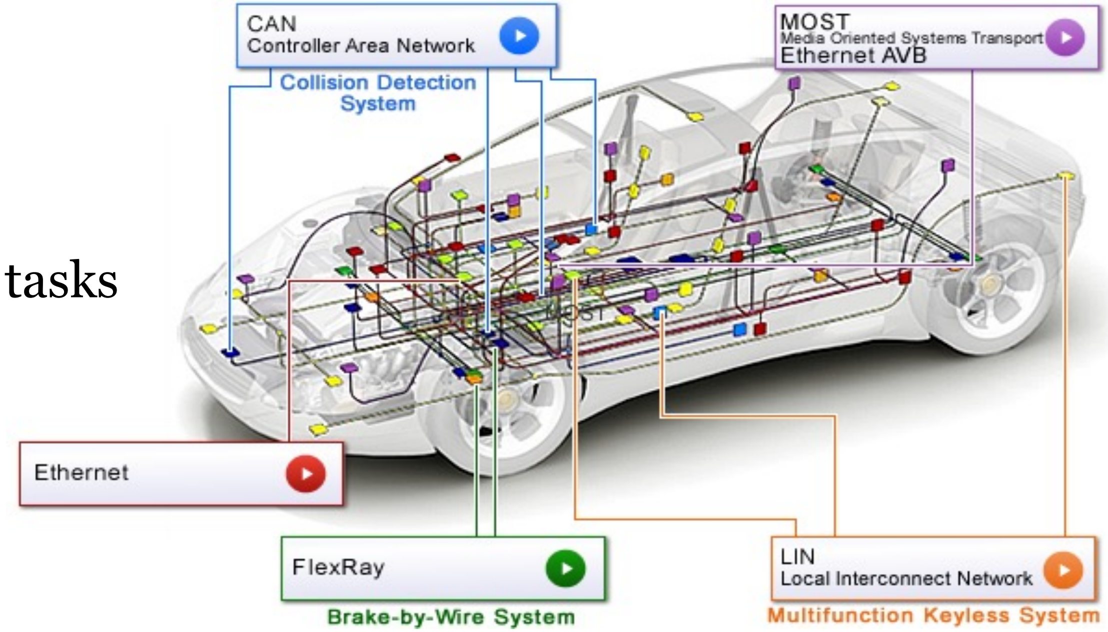

- The implementations of mixed messages in (a) and (b) are supported by the classical RTA for CAN
  - ✧ This is because the worst-case periodicity is bounded by Inhibit time and MDT in the implementations (a) and (b) respectively

- However, some implementations of mixed messages such as implementation (c) are not supported by the classical RTA for CAN
  - ✧ This is because the worst-case periodicity is neither bounded by period nor by minimum inter-arrival time (e.g. MUT in (c))

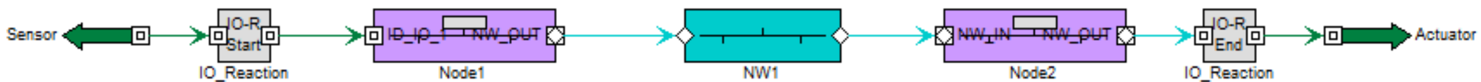
# Some recent extensions of RTA for CAN



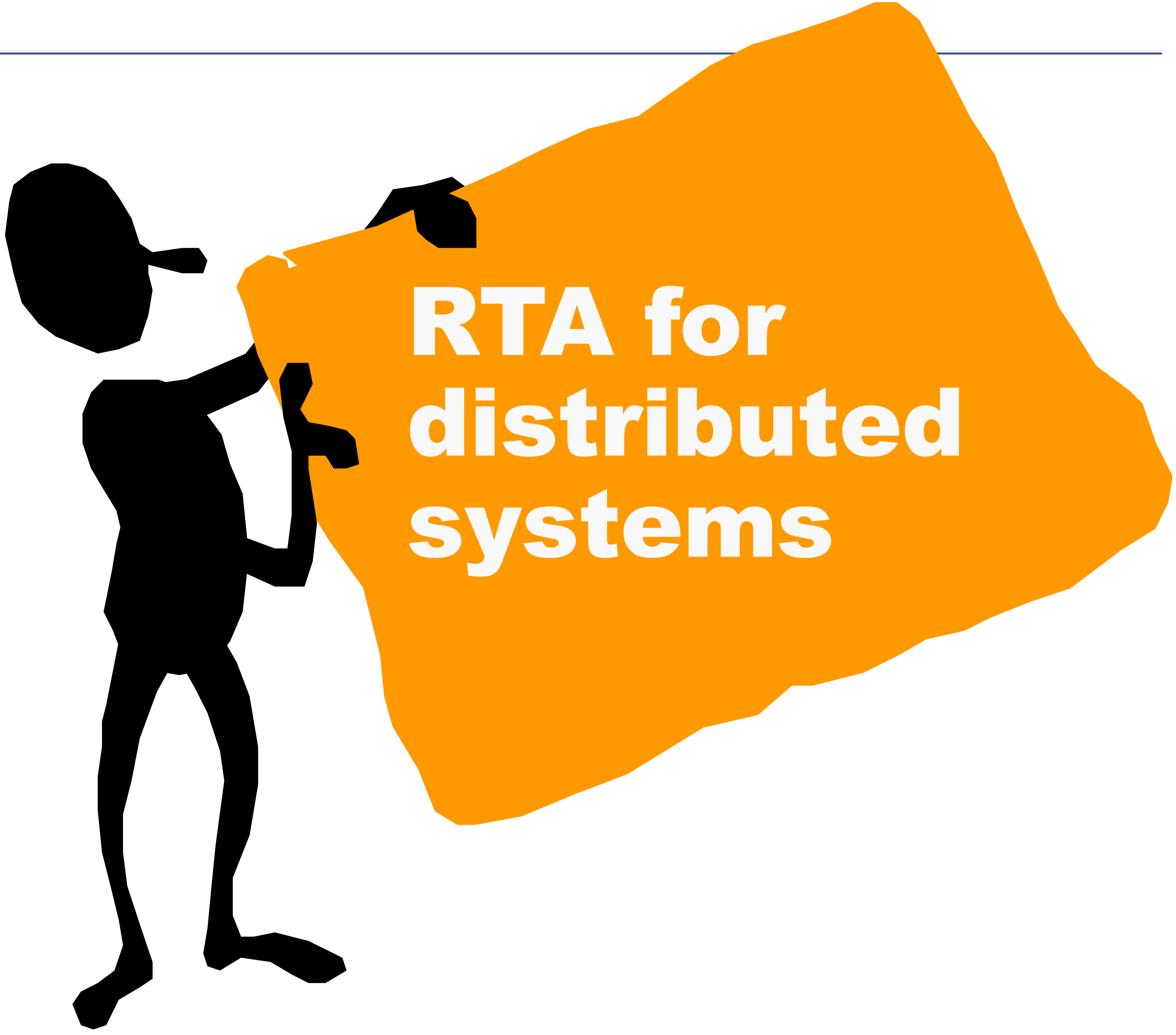
# Outline



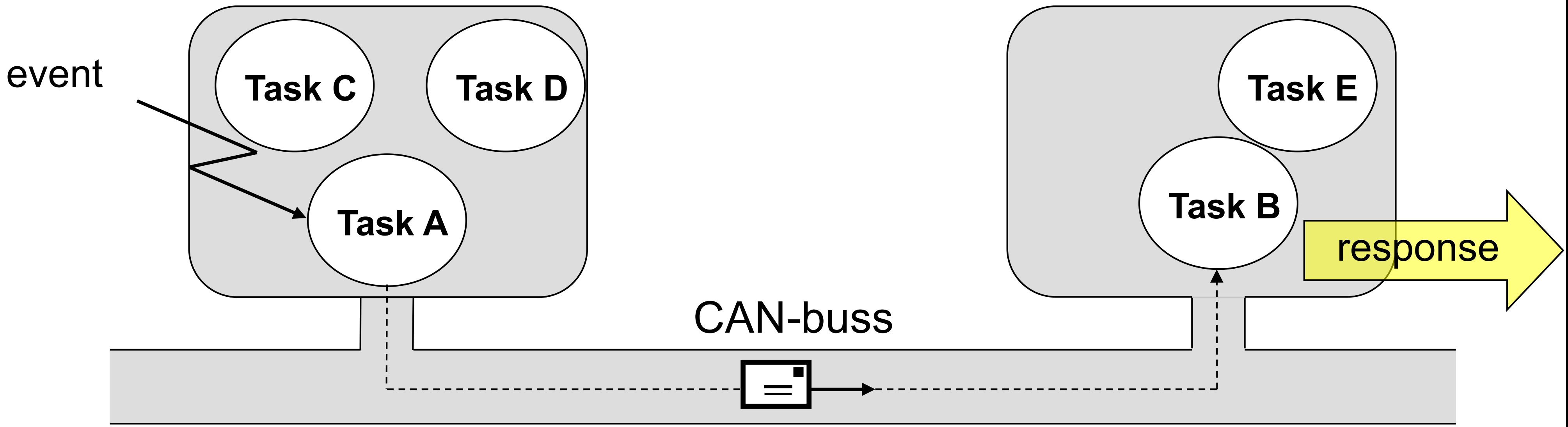

- Response-time analysis (RTA) of tasks
  - Basic RTA
  - RTA with blocking
  - RTA with jitter
  - RTA with Offsets
- RTA of Controller Area Network (CAN) messages
- Practical limitations in CAN Controllers and their effect on RTA for CAN
- Holistic RTA (for distributed systems)
- Data-propagation delay analysis
  - Single node systems
  - Distributed systems (End-to-end data-propagation delay analysis)

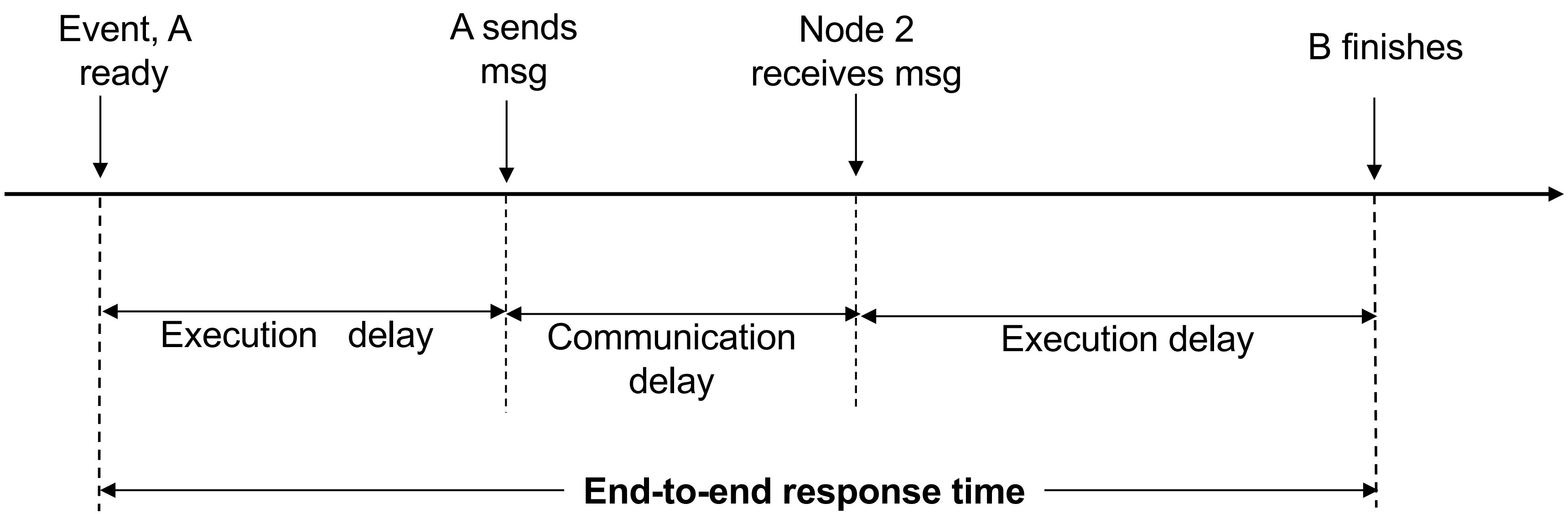



Figures courtesy of: http://www.cvel.clemson.edu/
http:// www.renesas.eu

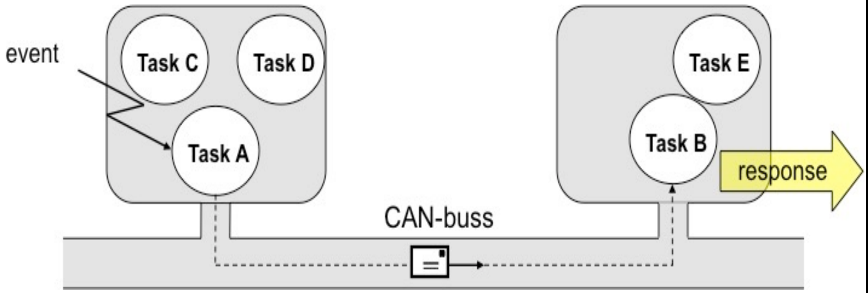

RTA for distributed systems

# End-to-end deadline



# Distributed transaction – response times




**Execution delay of A (= $R_A$) – Caused by other tasks on node 1**

- Use response time analysis to calculate $R_A$

**Message release jitter ($J_{m1}$) – Caused by variations in A's execution**

- Message jitter = difference between $R_A^{max}$ and $R_A^{min}$

**Communication delay (= $R_{m1}$) – Caused by other messages on the bus**

- Use jitter inherited from A ($J_{m1}$) and apply response time analysis for CAN to calculate $R_{m1}$

**Release jitter for B ($J_B$) – Caused by variations in m1's transmission**

- Release jitter for B = difference between $\mathbf{R_{m1}^{max}}$ and $\mathbf{R_{m1}^{min}}$
- $R_{m1}^{min} = 47\tau_{bit}$ (no other messages on the bus)

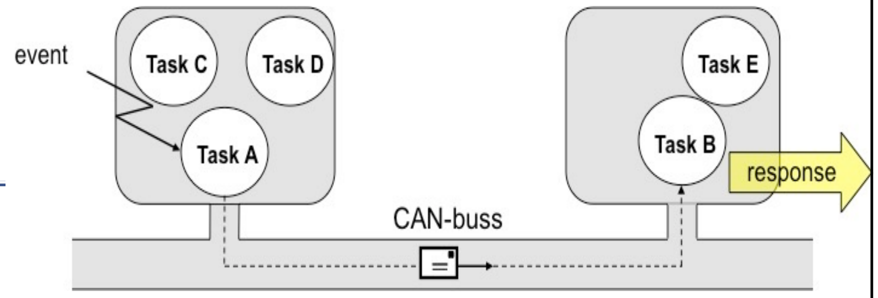
$$C_i = \left(47 + 8S_i + \left\lfloor (34+8S_i-1)/4 \right\rfloor\right)\tau_{bit}$$

Put $S_i = 0$

**Execution delay of B (= $R_B$) – Caused by other tasks on node 2**

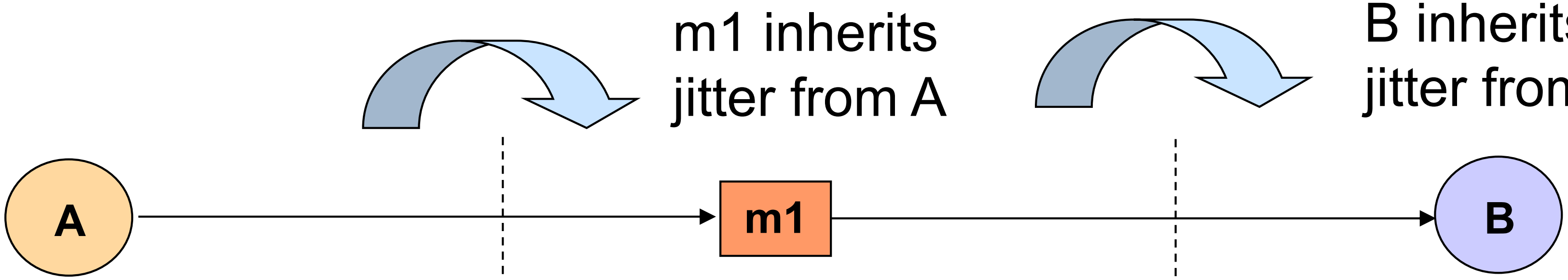
- Use jitter inherited form m1 ($\mathbf{J_B}$) and calculate response time for **B**

# Distributed transaction – Response times




m1 inherits jitter from A

B inherits jitter from m1




$$w_A^{n+1} = C_A + B_A + \sum_{\forall j \in hp(A)} \left\lceil \frac{w_A^n + J_j}{T_j} \right\rceil C_j$$

$$R_A = w_A + J_A$$

$$J_{m1} = R_A^{\max} - R_A^{\min}$$

$$w_{m1} = B_{m1} + \sum_{\forall j \in hp(m1)} \left\lceil \frac{w_{m1} + J_j + \tau_{bit}}{T_j} \right\rceil C_j$$

$$R_{m1} = J_{m1} + w_{m1} + C_{m1}$$

$$J_B = R_{m1}^{\max} - R_{m1}^{\min}$$

$$w_B^{n+1} = C_B + B_B + \sum_{\forall j \in hp(B)} \left\lceil \frac{w_B^n + J_j}{T_j} \right\rceil C_j$$

$$R_B = w_B + J_B$$

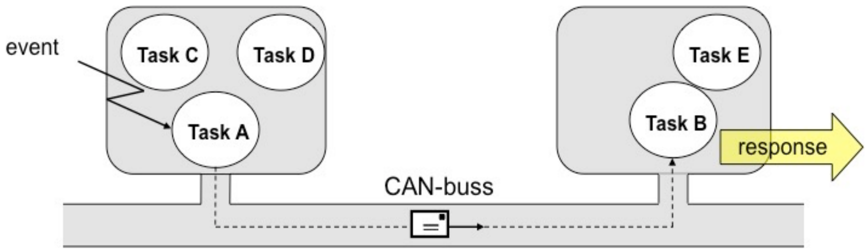
0 = Safe approximation

47 $\tau_{bit}$ = Safe approximation

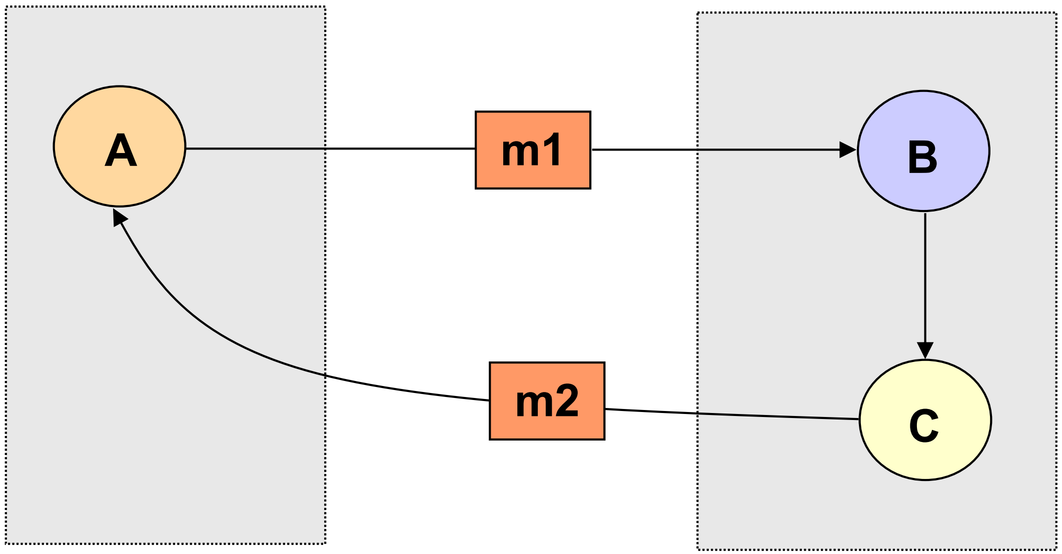
# Holistic schedulability analysis

**We learned how to handle jitter, as long this inheritance goes in one direction**

- Eg., from A via CAN-bus to B, as in previous example



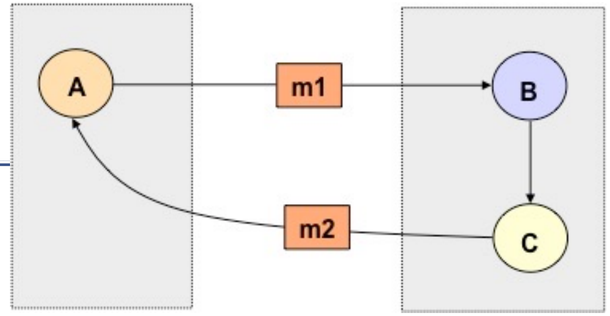

**But what happens if we have communication in both directions?**




1. B inherits jitter from A (via m1)
2. C inherits jitter from B
3. A inherits jitter from C (via m2)

**Holistic schedulability problem!**

# Holistic scheduling

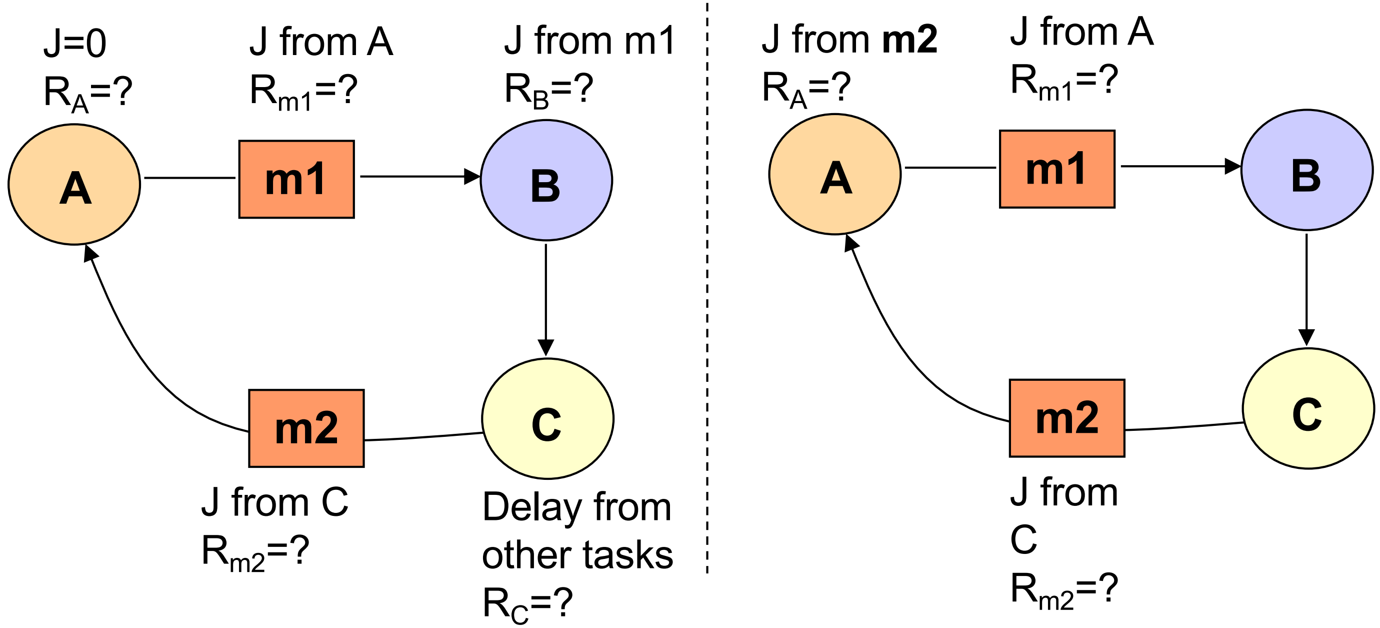



## One more iteration level

1. Initially we assume jitter=0
2. We iterate as before on each node and bus until we reach a local fix point
3. A fix point on a node gives information about jitter-inheritance
4. We must continue until jitter gets unchanged (**global fix point**)

**Example:**

Iteration 0 | Iteration 1 | . . . | Iteration n

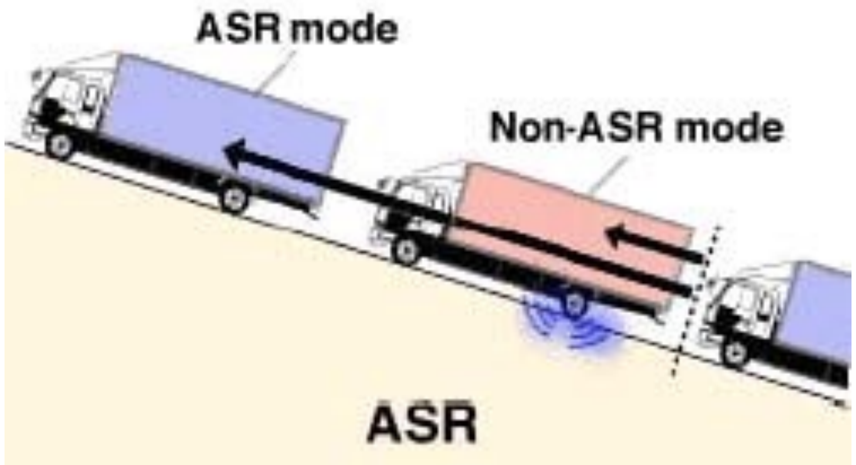



Iterate until all response times are stabilised

# Example holistic scheduling – ASR system

**Anti-Slip Regulation System for vehicles**

- For easier starting, accelerating and climbing on wet surfaces
- Spinnig of the wheels detected by wheels sensors and engine rotation adjusted to mantain the optimum speed



# Example holistic scheduling – ASR system

**System consist of five nodes and a CAN bus**

- 4 wheel nodes (*wheel1, wheel2, wheel3, wheel4*)
- 1 node for the central unit (*central*)

**Tasks on each *wheel* node**

- **S** – sampling of rotation speed of a wheel (time-trigged)
- **B** – brakes a wheel (event-trigged)
- $\mathbf{OS_{wheel}}$ – system task (time-trigged)

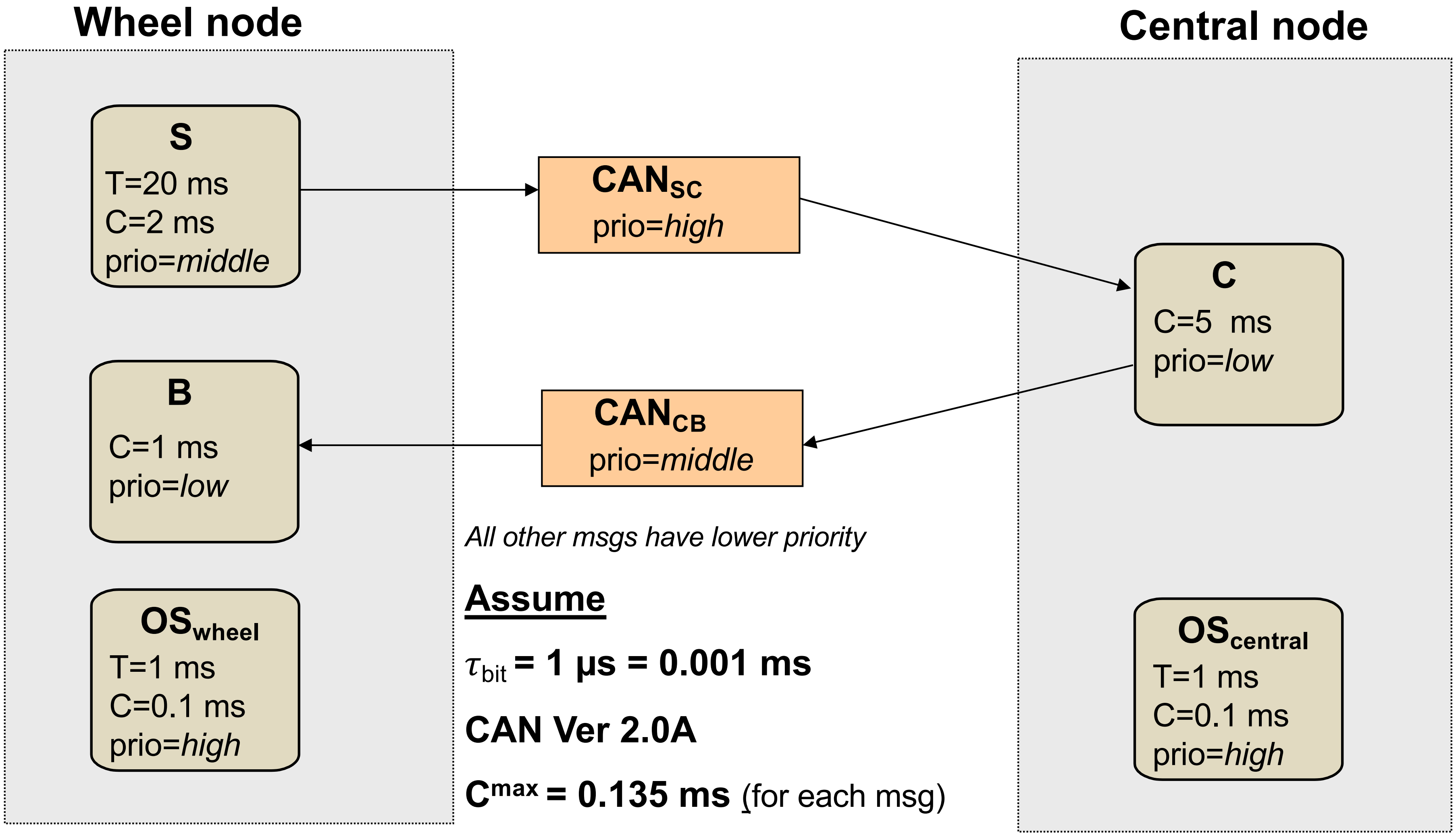
**Tasks on the node *central***

- **C** – calculates brake power based on the rotation speed (event-trigged)
- $\mathbf{OS_{central}}$ – system task (time-trigged)

**Communication between *wheel* and *central*:**

1. Sample rotation speed of the *wheel*
2. Send speed data to *central* via CAN-bus
3. Calculate brake power on node *central*
4. Send result to *wheel*

# Example holistic scheduling – ASR system




*All other msgs have lower priority*

**<u>Assume</u>**

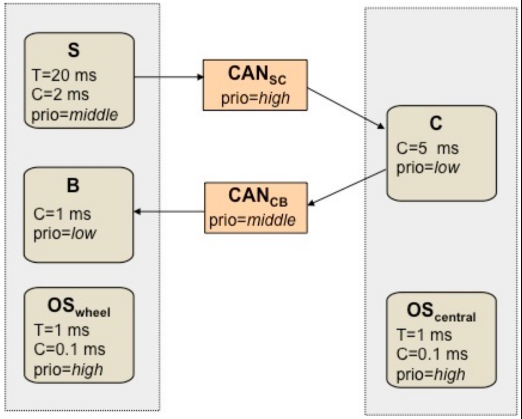
$\tau_{bit}$ **= 1 μs = 0.001 ms**

**CAN Ver 2.0A**

$C^{max}$ **= 0.135 ms** (for each msg)

Will the distributed transaction **S** → **CAN$_{SC}$** → **C**→ **CAN$_{CB}$** → **B** meet its end-to-end deadline of **18** ms?

# Example holistic scheduling – ASR system




**We start by calculating response time of S:**

- No blocking (no common resources): $B_S=0$
- High priority tasks on the same node: $hp(S) = \{OS_{wheel}\}$

$$w_{S_V}^0 = C_S = 2$$

$$w_S^1 = C_S + B_S + \left\lceil \frac{w_S^0 + J_{OSwheel}}{T_{OSwheel}} \right\rceil * C_{OSwheel} = 2+0+\left\lceil \frac{2+0}{1} \right\rceil * 0.1 = 2.2$$

$$w_S^2 = 2+0+\left\lceil \frac{2.2+0}{1} \right\rceil * 0.1 = 2.3$$

$$w_S^3 = 2+0+\left\lceil \frac{2.3+0}{1} \right\rceil * 0.1 = 2.3 \qquad w_S^3 = w_S^2 \rightarrow w_S = 2.3$$

- Incomming jitter in the first itteration step is equal to zero. Response time is:

$$R_S = J_S + w_S = 0 + 2.3 = 2.3ms$$

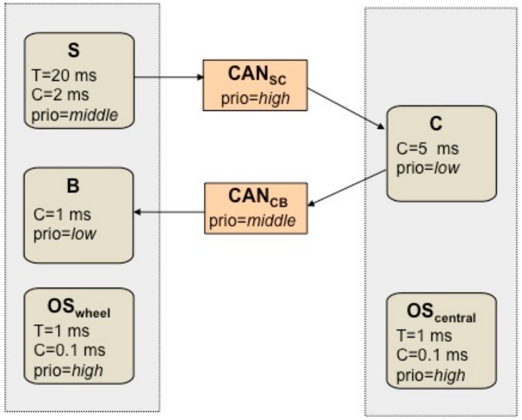
# Example holistic scheduling – ASR system

**We continue by calculating response time for $CAN_{SC}$:**

- Low prio messages: *$lp(CAN_{SC})=\{CAN_{CB}$, all other$\}$*. This gives blocking factor:

$$B_{CAN_{SC}} = \max_{\forall k \in lp(CAN_{SC})} C_k = \max(CAN_{CB}, all\_other) = \max(0.135, 0.135, \ldots) = 0.135ms$$

- No high prio messages: *$hp(CAN_{SC}) = \{\}$*




$$w_{CAN_{SC}} = B_{CAN_{SC}} + 0 = 0.135ms$$

- Incomming jitter is equal to response time of the task that sends the message, i.e., task *S*:

$$J_{CAN_{SC}} = R_S = 2.3ms$$

(Note! We assume $R^{min}$=0, analysis is pessimistic but safe )

- Finally, we get the response time for *$CAN_{SC}$* :

$$R_{CAN_{SC}} = J_{CAN_{SC}} + w_{CAN_{SC}} + C_{CAN_{SC}} = 2.3 + 0.135 + 0.135 = 2.57ms$$

# Example holistic scheduling – ASR system

**Next step is to calculate response time for C (that is activated by $CAN_{SC}$):**

- No blocking: dvs $B_C$=0
- hp(C) = {$OS_{central}$}

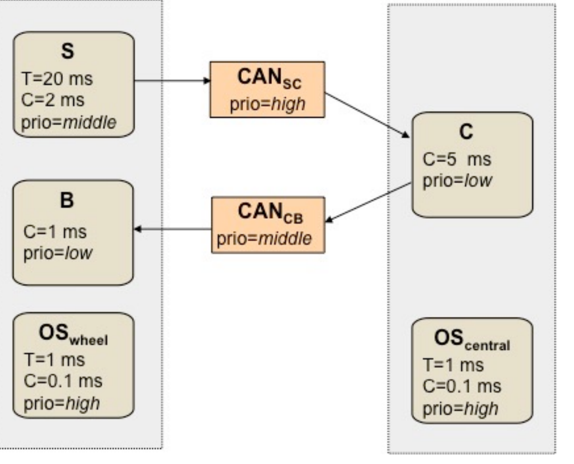
$$w_C^0 = C_C = 5$$

$$w_C^1 = C_C + B_C + \left\lceil \frac{w_C^0 + J_{OScentral}}{T_{OScentral}} \right\rceil * C_{OScentral} = 5 + 0 + \left\lceil \frac{5+0}{1} \right\rceil * 0.1 = 5.5$$

$$w_C^2 = 5 + 0 + \left\lceil \frac{5.5+0}{1} \right\rceil * 0.1 = 5.6$$

$$w_C^3 = 5 + 0 + \left\lceil \frac{5.6+0}{1} \right\rceil * 0.1 = 5.6$$

$$w_C^3 = w_C^2 \rightarrow w_C = 5.6$$




- The jitter is:

$$J_C = R_{CAN_{SC}} = 2.57ms$$

- Response time is:

$$R_C = J_C + w_C = 2.57 + 5.6 = 8.17ms$$

# Example holistic scheduling – ASR system

**Because we are interested in entire transaction S→$CAN_{SC}$→C→$CAN_{CB}$→B we continue by calculating the response time of $CAN_{CB}$:**

- Low priority messages: *lp($CAN_{CB}$)={all except $CAN_{SC}$}*. Blocking factor is:

$$B_{CAN_{CB}} = \max_{\forall k \in lp(CAN_{CB})} C_k = 0.135ms$$

- One high priority message: *hp($CAN_{CB}$) = { $CAN_{SC}$ }*. Period time for $CAN_{SC}$ is equal to the period time of the task that sends it, i.e., task S, T= 20ms. This gives:

$$w^0_{CAN_{CB}} = 0.135ms$$

$$w^1_{CAN_{CB}} = B_{CAN_{CB}} + \left\lceil \frac{w^0_{CAN_{CB}} + J_{CAN_{SC}} + \tau_{bit}}{T_{CAN_{SC}}} \right\rceil * C_{CAN_{SC}} = 0.135 + \left\lceil \frac{0 + 2.3 + 0.001}{20} \right\rceil * 0.135 = 0.27$$

$$w^2_{CAN_{CB}} = 0.135 + \left\lceil \frac{1.135 + 2.3 + 0.001}{20} \right\rceil * 0.135 = 0.27 = w^1_{CAN_{CB}} \rightarrow w_{CAN_{CB}} = 0.27ms$$

- Incoming jitter is equal to response time for the task that is sending the message, i.e., task *C*:

$$J_{CAN_{CB}} = R_C = 8.17ms$$

- Finally, we get response time for *$CAN_{CB}$* as:

$$R_{CAN_{CB}} = J_{CAN_{CB}} + w_{CAN_{CB}} + C_{CAN_{CB}} = 8.17 + 0.27 + 0.135 = 8.575ms$$

## Example holistic scheduling – ASR system

**We calculate the response time for the last part of the chain, task B:**

- $B_B=0$
- hp(B) = {S,$OS_{wheel}$}

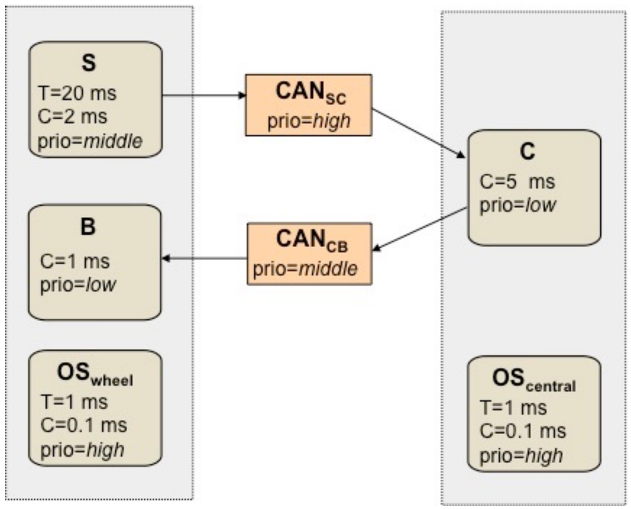
$$w_B^0 = C_B = 1$$

$$w_B^1 = C_B + B_B + \left\lceil \frac{w_B^0 + J_{OSwheel}}{T_{OSwheel}} \right\rceil * C_{OSwheel} + \left\lceil \frac{w_B^0 + J_S}{T_S} \right\rceil * C_S = 3.1$$

$$w_B^2 = 1 + 0 + \left\lceil \frac{3.1+0}{1} \right\rceil * 0.1 + \left\lceil \frac{3.1+0}{20} \right\rceil 2 = 1 + 0 + 0.4 + 2 = 3.4$$

$$w_B^3 = 1 + 0 + \left\lceil \frac{3.4+0}{1} \right\rceil * 0.1 + \left\lceil \frac{3.4+0}{20} \right\rceil 2 = 1 + 0 + 0.4 + 2 = 3.4$$

$$w_B^3 = w_B^2 \rightarrow w_B = 3.4ms$$




Incomming jitter depends on $CAN_{CB}$:

$$J_B = R_{CAN_{CB}} = 8.575ms$$

Response time is:

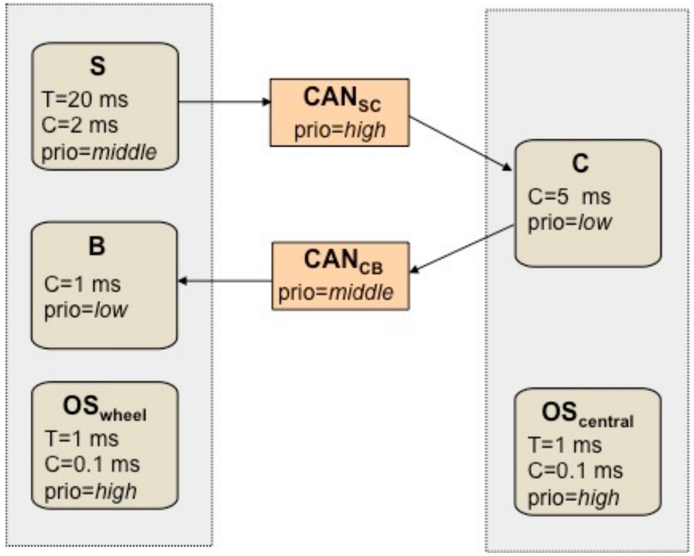
$$R_B = J_B + w_B = 8.575 + 3.4 = 11.975ms$$

# Example holistic scheduling – ASR system

**We are done with the first iteration. Do we need to continue iterating?**

**No, since higher priority tasks to B has not changed their jitter**

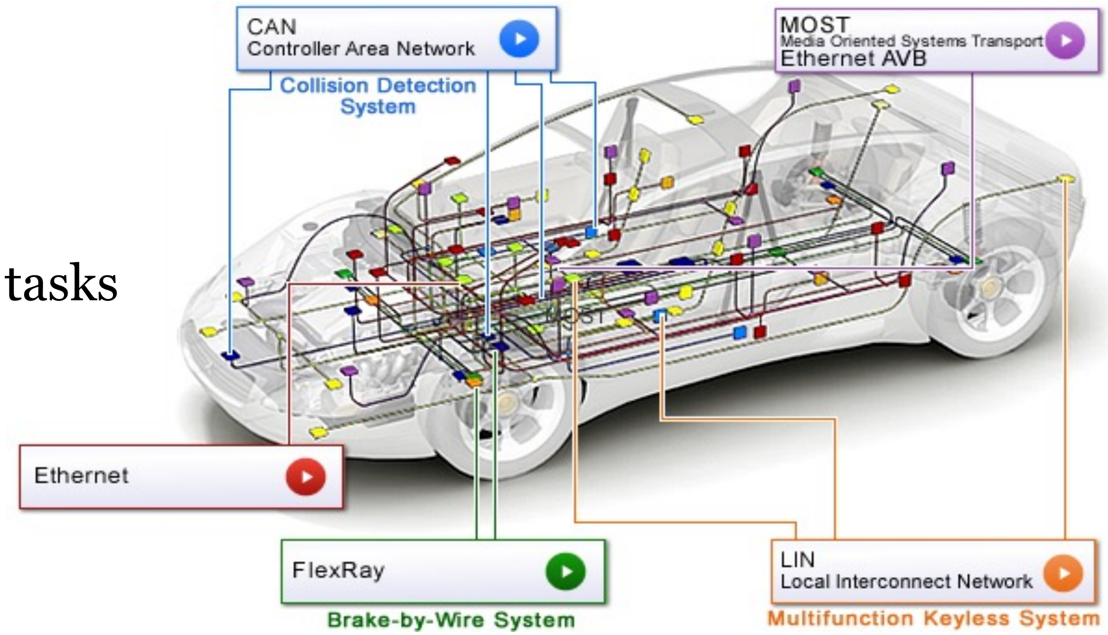
**In other words, nothing will change if we iterate once again**



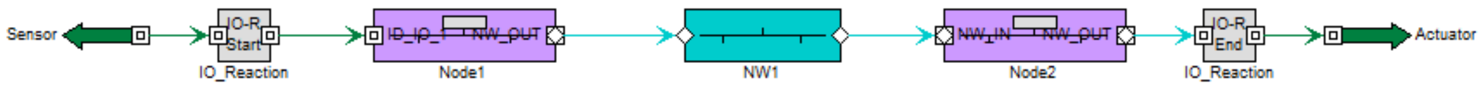

**Answer: Transaction's response time is 11.975 ms is less that the deadline 18ms, so it will meet its timing requirement.**

MÄLARDALEN UNIVERSITY SWEDEN

# Outline

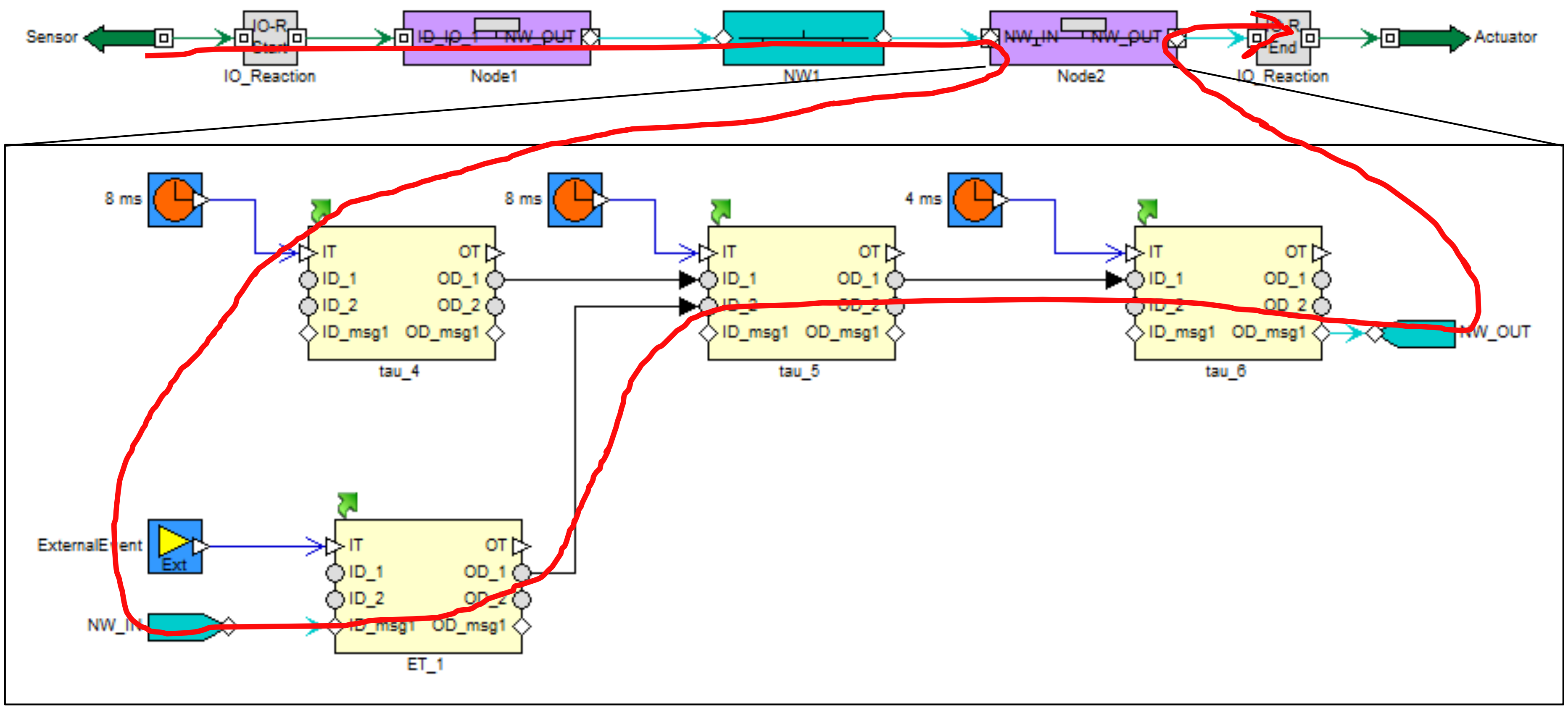



- Response-time analysis (RTA) of tasks
  - Basic RTA
  - RTA with blocking
  - RTA with jitter
  - RTA with Offsets
- RTA of Controller Area Network (CAN) messages
- Practical limitations in CAN Controllers and their effect on RTA for CAN
- Holistic RTA (for distributed systems)
- Data-propagation delay analysis
  - Single node systems
  - Distributed systems (End-to-end data-propagation delay analysis)

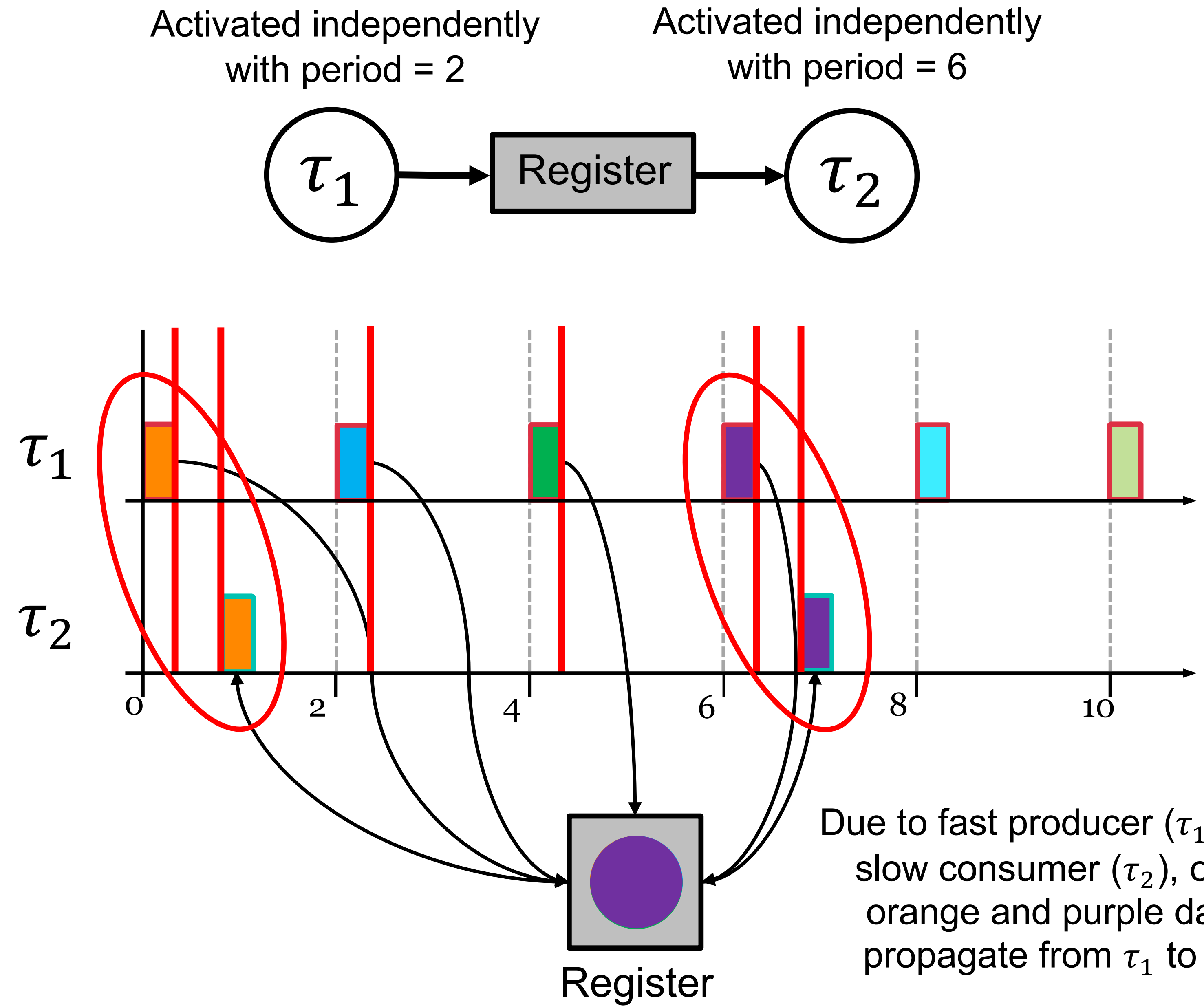



Figures courtesy of: http://www.cvel.clemson.edu/
http:// www.renesas.eu

# End-to-end data propagation delay analysis



# Register-based Communication

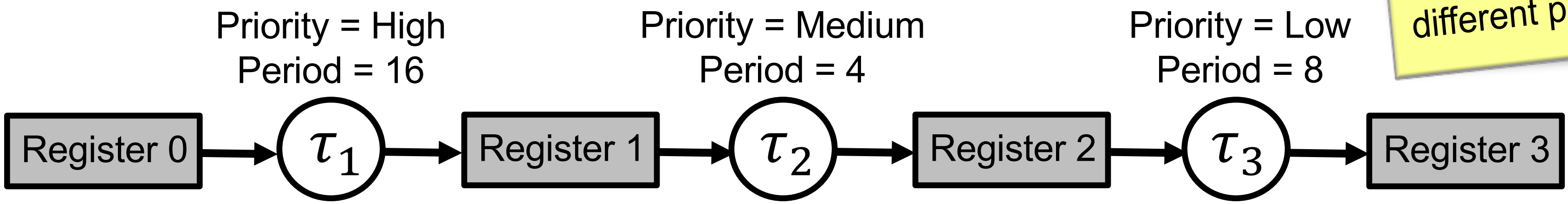



Due to fast producer ($\tau_1$) and slow consumer ($\tau_2$), only orange and purple data propagate from $\tau_1$ to $\tau_2$.

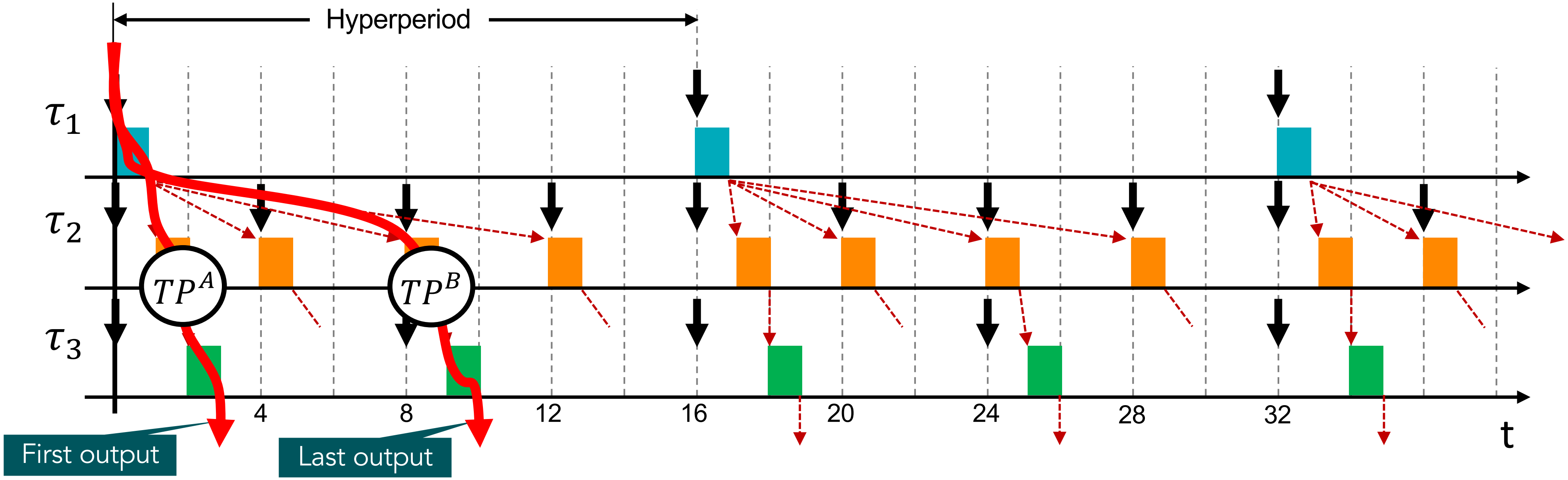
Figure courtesy of Matthias Becker, MDH

# Data Propagation Delays

This is a multi-rate chain as the tasks are activated with different periods.



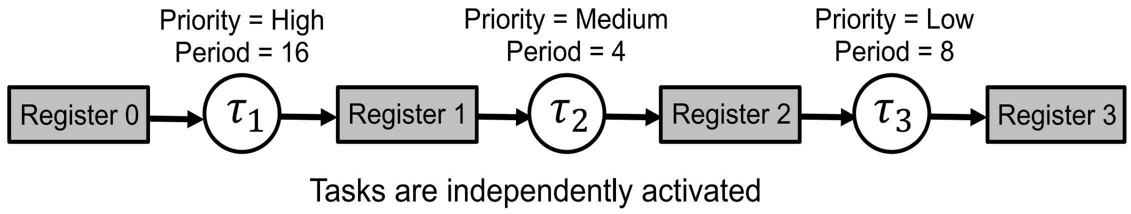

Tasks are independently activated

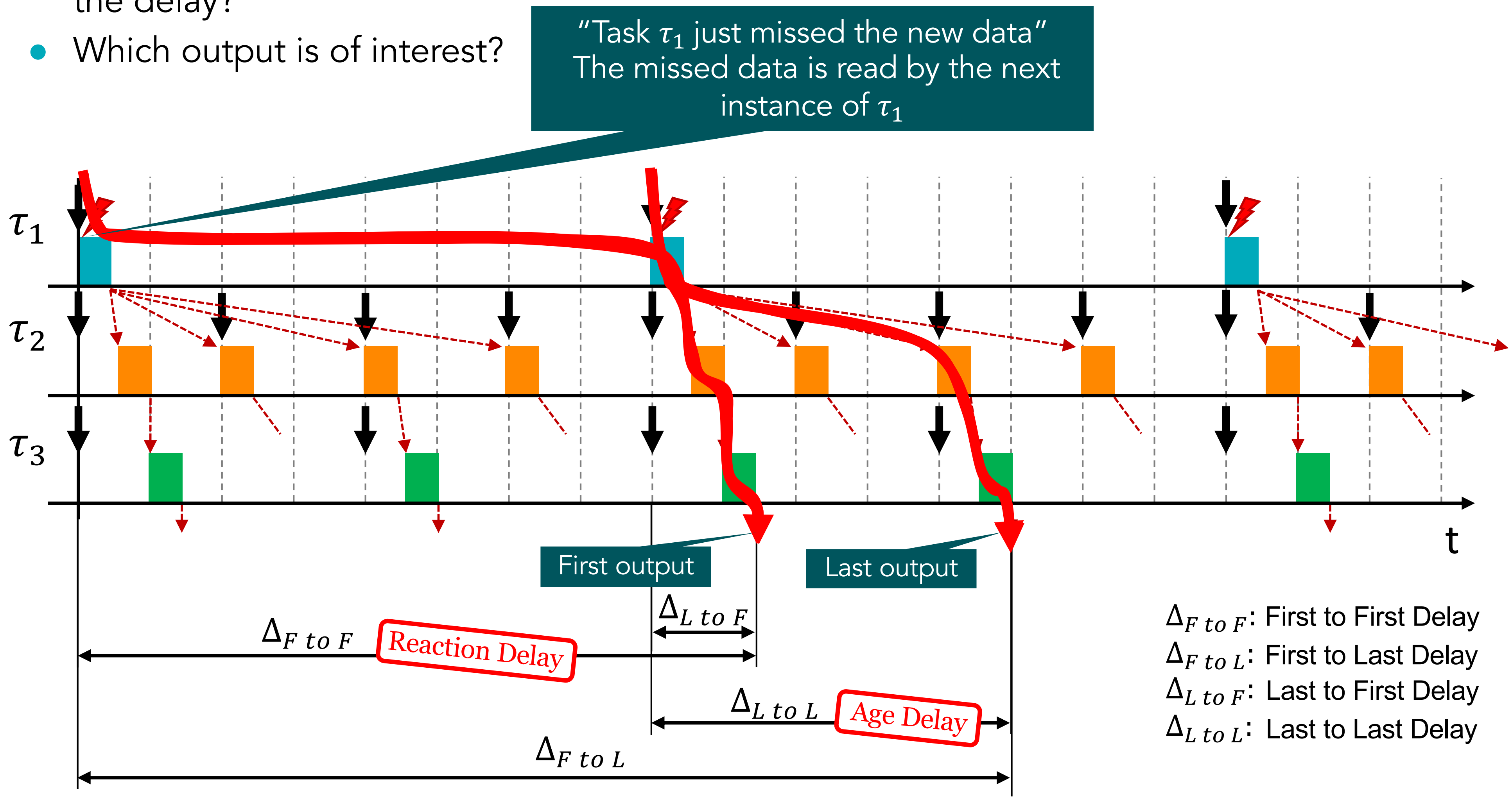



$TP^A$: Timed Path A
$TP^B$: Timed Path B

# Data Propagation Delays

- 4 different delays
  - What is the start event of the delay?
  - Which output is of interest?



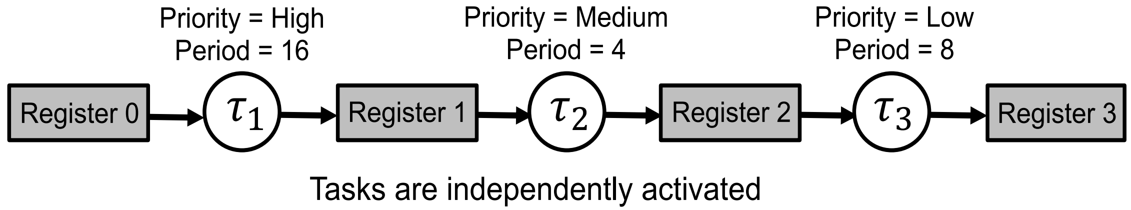



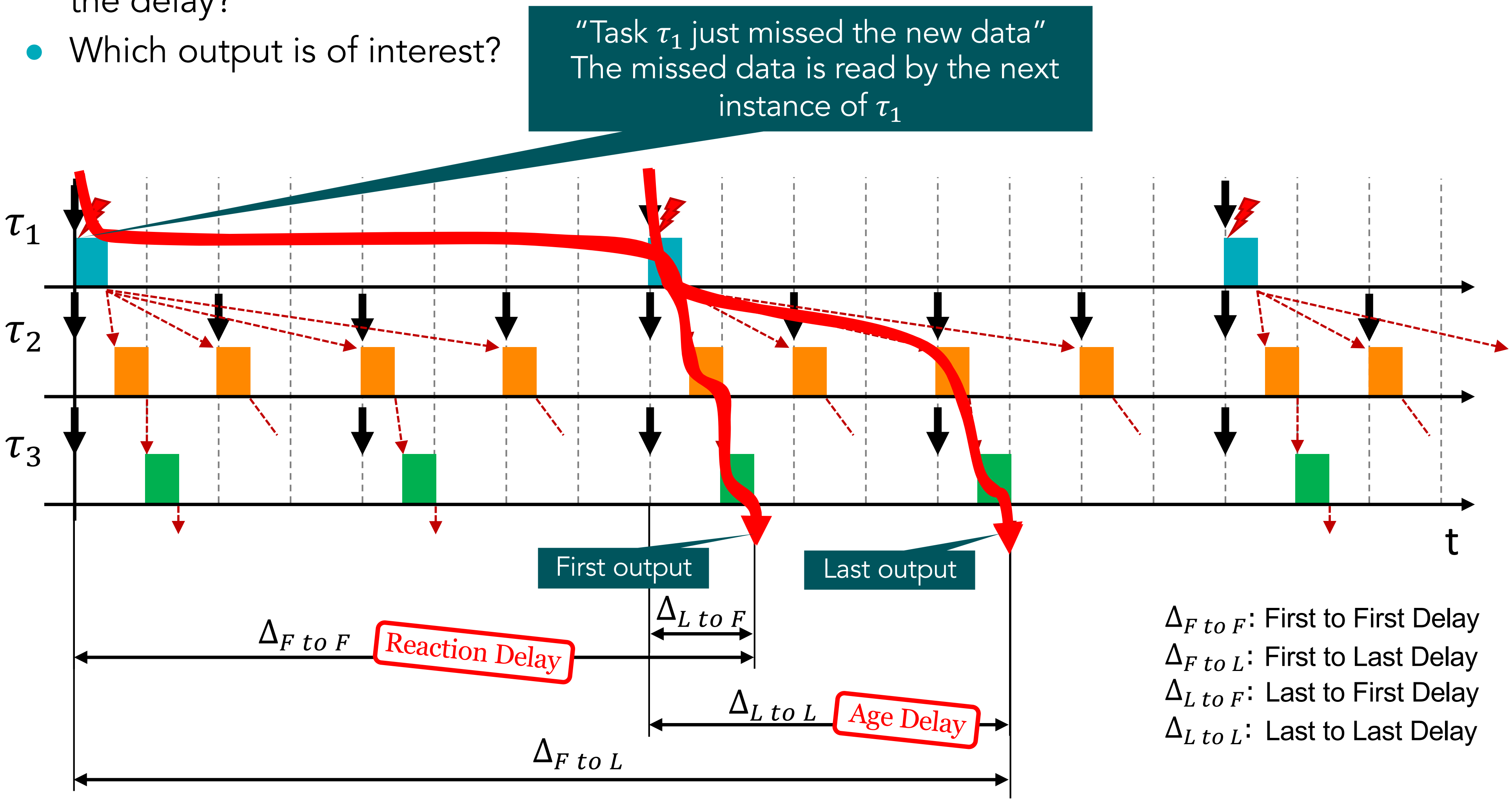

$\Delta_{F\ to\ F}$: First to First Delay
$\Delta_{F\ to\ L}$: First to Last Delay
$\Delta_{L\ to\ F}$: Last to First Delay
$\Delta_{L\ to\ L}$: Last to Last Delay

Figure courtesy of Matthias Becker, MDH

# Data Propagation Delays

- 4 different delays
  - What is the start event of the delay?
  - Which output is of interest?




$\Delta_{F\ to\ F}$: First to First Delay
$\Delta_{F\ to\ L}$: First to Last Delay
$\Delta_{L\ to\ F}$: Last to First Delay
$\Delta_{L\ to\ L}$: Last to Last Delay

# Data Propagation Delays: Reaction & Age

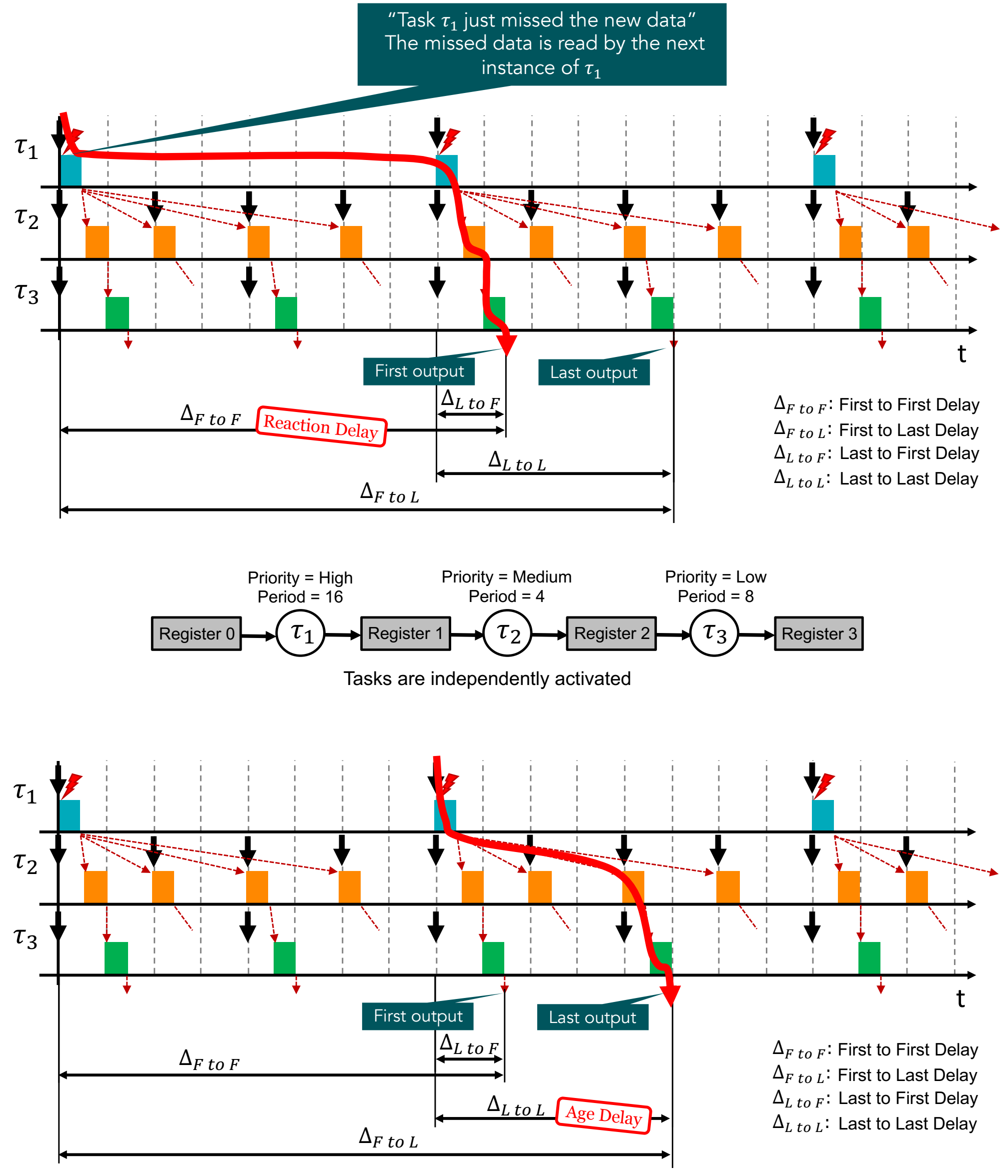
## Data Reaction delay

- Applications in the body electronics domain in vehicles
  - Button-to-action delay
  - E.g., electronic door lock in a car

## Data Age delay

- Applications in the control systems domain in vehicles
  - Maximum age of the data
  - E.g., control signals driving the actuators do not exceed a maximum age

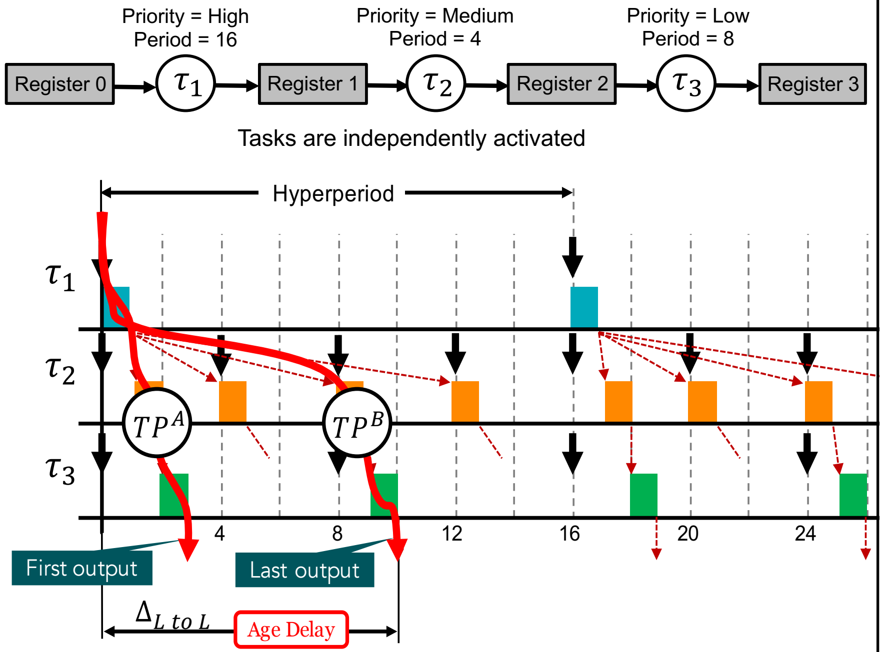
**Note:** "last-to-last" (Age) considers the delay between the last input of the chain (that is not overwritten) until the last output of the chain (even in case of duplicates) [1].





[1] N. Feiertag, K. Richter, J. Nordlander, J. Jonsson, A Compositional Framework for End-to-End Path Delay Calculation of Automotive Systems under Different Path Semantics, Workshop on Compositional Theory and Technology for Real-Time Embedded Systems (CRTS), 2008.

# Assumptions, Definitions & Notations




**Assumptions (for simplified analysis)**

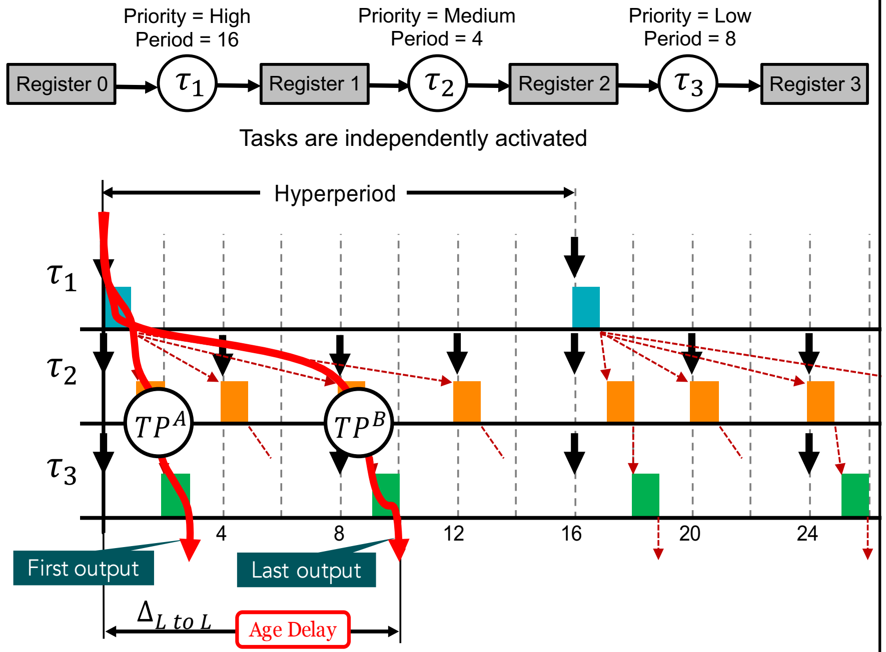
- No offsets
- Within a task chain, the priority of any task is not higher than the priority of its predecessor task

Note that the analysis in [1] is not restricted by these assumptions.

These assumptions are made to simplify the analysis for the purpose of understanding.

[1] N. Feiertag, K. Richter, J. Nordlander, J. Jonsson, A Compositional Framework for End-to-End Path Delay Calculation of Automotive Systems under Different Path Semantics, Workshop on Compositional Theory and Technology for Real-Time Embedded Systems (CRTS), 2008.

# Assumptions, Definitions & Notations




- **Timed Path (TP)** is a sequence of task instances along a task chain.

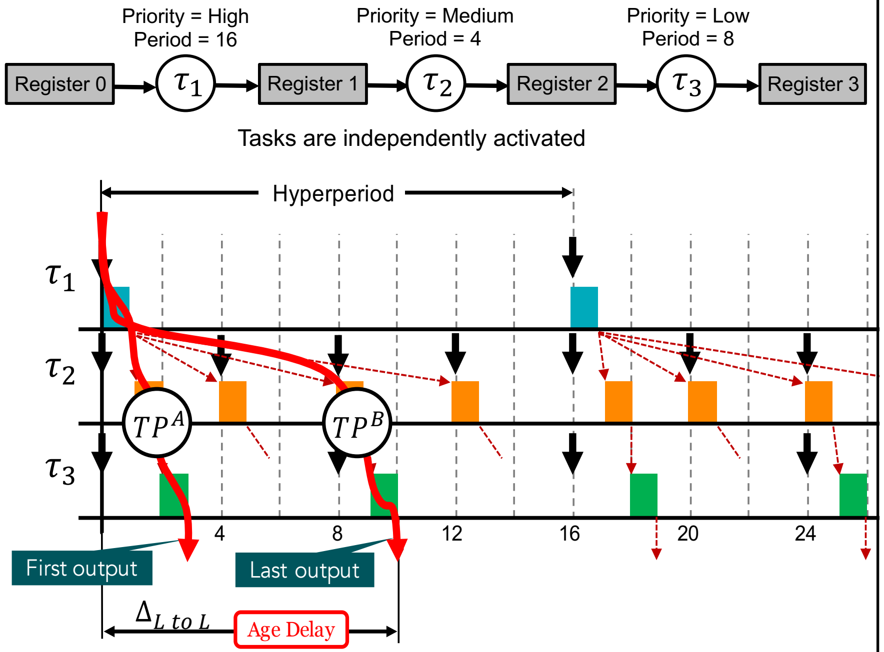
- A **reachable timed path** consists of the sequence of task instances along the chain through which the data can actually propagate from the first task to the last task in the chain.

- $TP^A$ and $TP^B$ in the figure are two reachable timed paths.
  - $TP^A$ = ($\tau_1(1)$, $\tau_2(1)$, $\tau_3(1)$), i.e., timed path from the 1st instance of $\tau_1$ to the 1st instance of $\tau_2$ to the 1st instance of $\tau_3$
  - $TP^B$ =

Note that 'n' in $\tau_k(n)$ represents the $n^{th}$ instance of task $\tau_k$. The first instance of $\tau_k$ is represented by $\tau_k(1)$.

- Path: ($\tau_1(1)$, $\tau_2(2)$, $\tau_3(2)$)

# Assumptions, Definitions & Notations



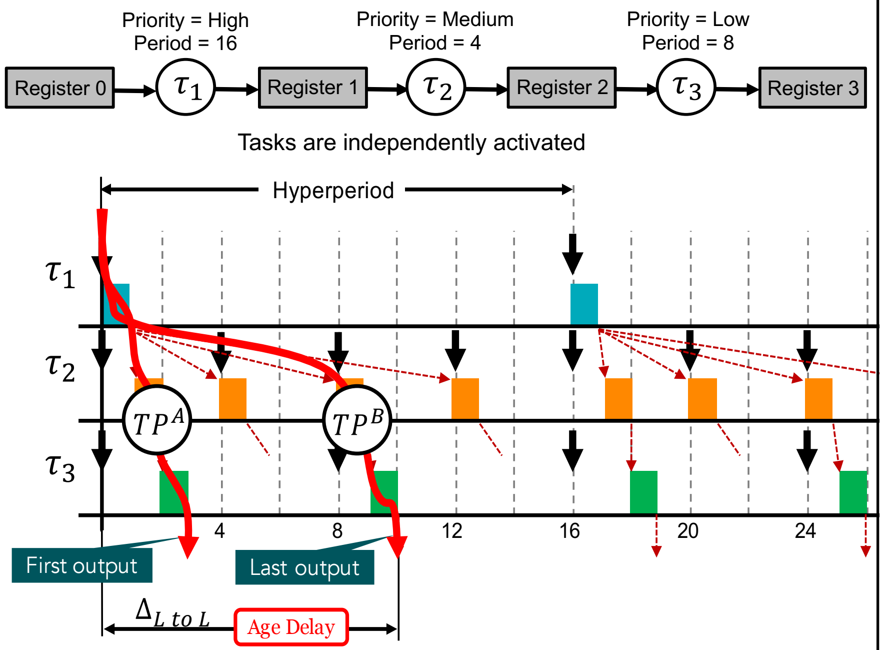

- Let delay in the time path $TP^A$ be represented by Delay($TP^A$)

- $\boldsymbol{\alpha}_k$(n) = Activation time of n$^{th}$ instance of task $\tau_k$
  - $\boldsymbol{\alpha}_1(1) = 0$, $\boldsymbol{\alpha}_2(1) = 0$, $\boldsymbol{\alpha}_3(1) = 0$

- $\boldsymbol{\alpha}_k(TP^B)$ is the activation time of the instance of the k$^{th}$ task which is in path $TP^B$
  - $\boldsymbol{\alpha}_1\,(TP^B)$ =
  - $\boldsymbol{\alpha}_2\,(TP^B)$ =
  - $\boldsymbol{\alpha}_3\,(TP^B)$ =

- $R_k$(n) = Response time of n$^{th}$ instance of task $\tau_k$
  - $R_3(1) = 3$, because the first instance of $\tau_3$ experiences interference from the first instances of $\tau_1$ and $\tau_2$ (assuming no other interferences)
  - $R_3(2)$ =

# Simplified Calculations for the Age Delay




**Step 1**

Identify all reachable timed paths TPs that are initiated in the hyperperiod.

**Step 2**

Calculate the delay in each path separately

Delay($TP^M$) =

Where, $\boldsymbol{\alpha}_{\text{Last}}(TP^M)$ is the activation time of the instance of the Last task which is in path $TP^M$

$\mathrm{R}_{\text{Last}}(TP^M)$ is the response time of the instance of the Last task which is in path $TP^M$

$\boldsymbol{\alpha}_{\text{First}}(TP^M)$ is the activation time of the instance of the First task which is in path $TP^M$

**Step 3**

Age delay is equal to the maximum delay among all reachable timed paths

Age delay = Max{Delay($TP^1$), ..., Delay($TP^M$)}

**Example**

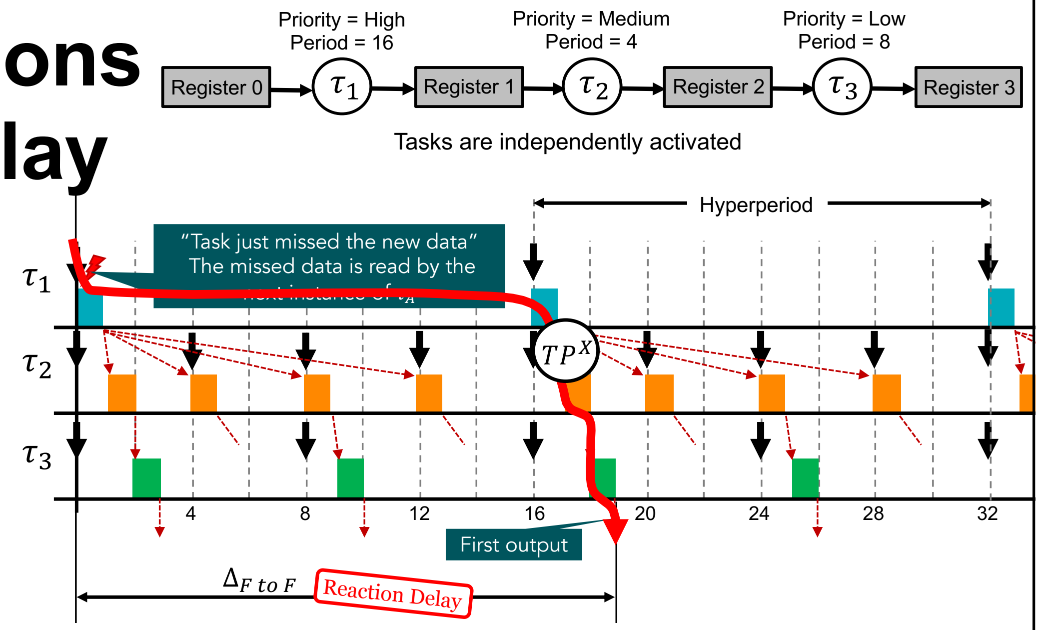
Delay($TP^A$) = $\boldsymbol{\alpha}_3(TP^A)$ + $\mathrm{R}_3(TP^A)$ - $\boldsymbol{\alpha}_1(TP^A)$ = $\boldsymbol{\alpha}_3(1)$ + $\mathrm{R}_3(1)$ - $\boldsymbol{\alpha}_1(1)$ =

Delay($TP^B$) = $\boldsymbol{\alpha}_3(TP^B)$ + $\mathrm{R}_3(TP^B)$ - $\boldsymbol{\alpha}_1(TP^B)$ = =

Age delay = Max{Delay($TP^A$), Delay($TP^B$)} = Max {3, 10} = 10

**Age delay = 10**

# Simplified Calculations for the Reaction Delay




**Step 1**

Identify all reachable timed paths in the hyperperiod that have **non-duplicate ("first") output** of the chain. Consider the effect of just missing the new data at the input (first task in the chain), e.g., $TP^X$

**Step 2**

Calculate the delay in each such path separately

$$\text{Reaction}(TP^M) = \boldsymbol{\alpha}_{\text{Last}}(TP^M) + \text{R}_{\text{Last}}(TP^M) - \boldsymbol{\alpha}_{\text{First}}(\text{Pred}(TP^M))$$

Where, $\boldsymbol{\alpha}_{\text{First}}(\text{Pred}(TP^M))$ is the activation time of the instance that is predecessor to the instance of the First task which is in path $TP^M$

**Step 3**

Reaction delay is equal to the maximum delay among all such timed paths

$$\text{Reaction delay} = \text{Max}\{\text{Delay}(TP^1), \ldots, \text{Delay}(TP^M)\}$$

**Example**

$\text{Delay}(TP^X) = \boldsymbol{\alpha}_3(TP^X) + \text{R}_3(TP^X) - \boldsymbol{\alpha}_1(\text{Pred}(TP^X))$ =

Reaction delay = $\text{Max}\{\text{Delay}(TP^X)\}$ = 19

**There is only one reachable TP in this example that fulfills the criteria in Step 1**

Reaction delay = 19

# Lecture 7: Highlight
# Group Assignment: End-to-end Data-propagation Delay Analysis of a Distributed Real-time System

- Make Groups. Each group should contain around 4-5 persons

- Give a cool name to your group

- We will solve an assignment in groups during the lecture

- You will write the name of of your group together with the analysis results on the board

- We will compare the results and discuss the solution(s)