OVERCOMING QOS CHALLENGES IN A FULL AUTOMOTIVE ETHERNET ARCHITECTURE

Xiaoting LI,	Ampere
Josetxo VILLANUEVA,	Ampere
Xiaojie GUO,	RTaW
Jörn MIGGE,	RTaW

AMPERE

RTaW
RealTime-at-Work

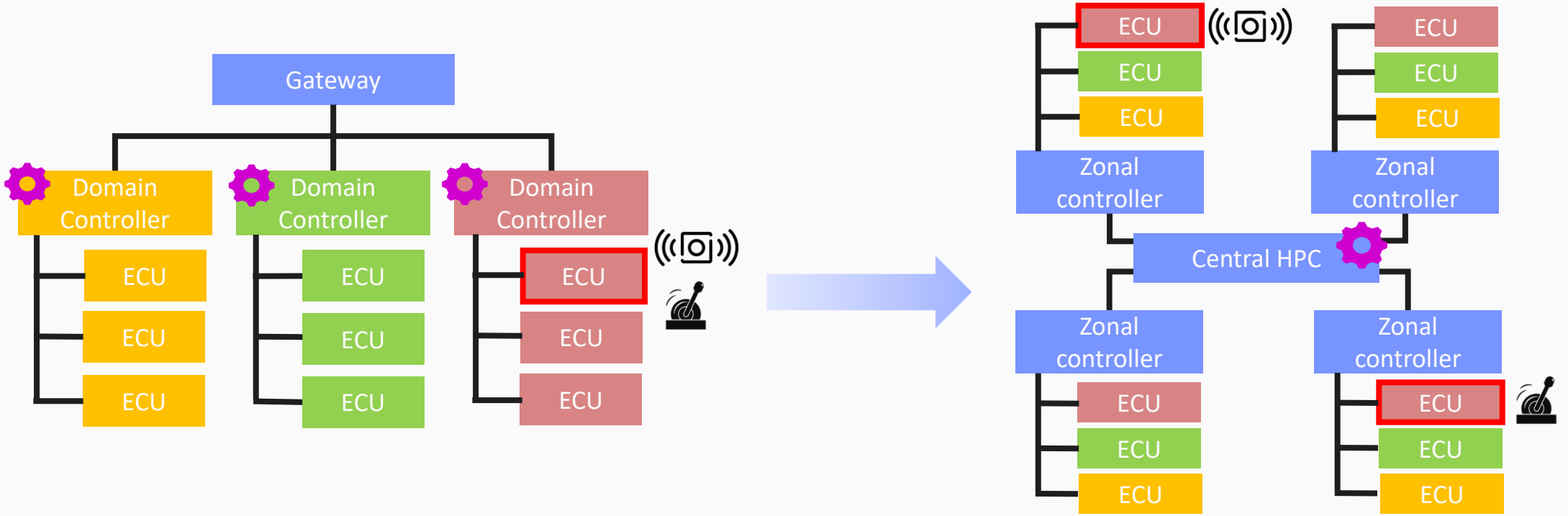
IEEE ETHERNET & IP @ AUTOMOTIVE TECHNOLOGY DAY

OCTOBER 15, 2025 | TOULOUSE, FRANCE

PROBLEM STATEMENT
MULTIPROTOCOL ARCHITECTURE

XIAOTING LI
JOSETXO VILLANUEVA

QOS CHALLENGES IN A MULTI-PROTOCOL EE ARCHITECTURE
PROBLEM STATEMENT & MOTIVATION

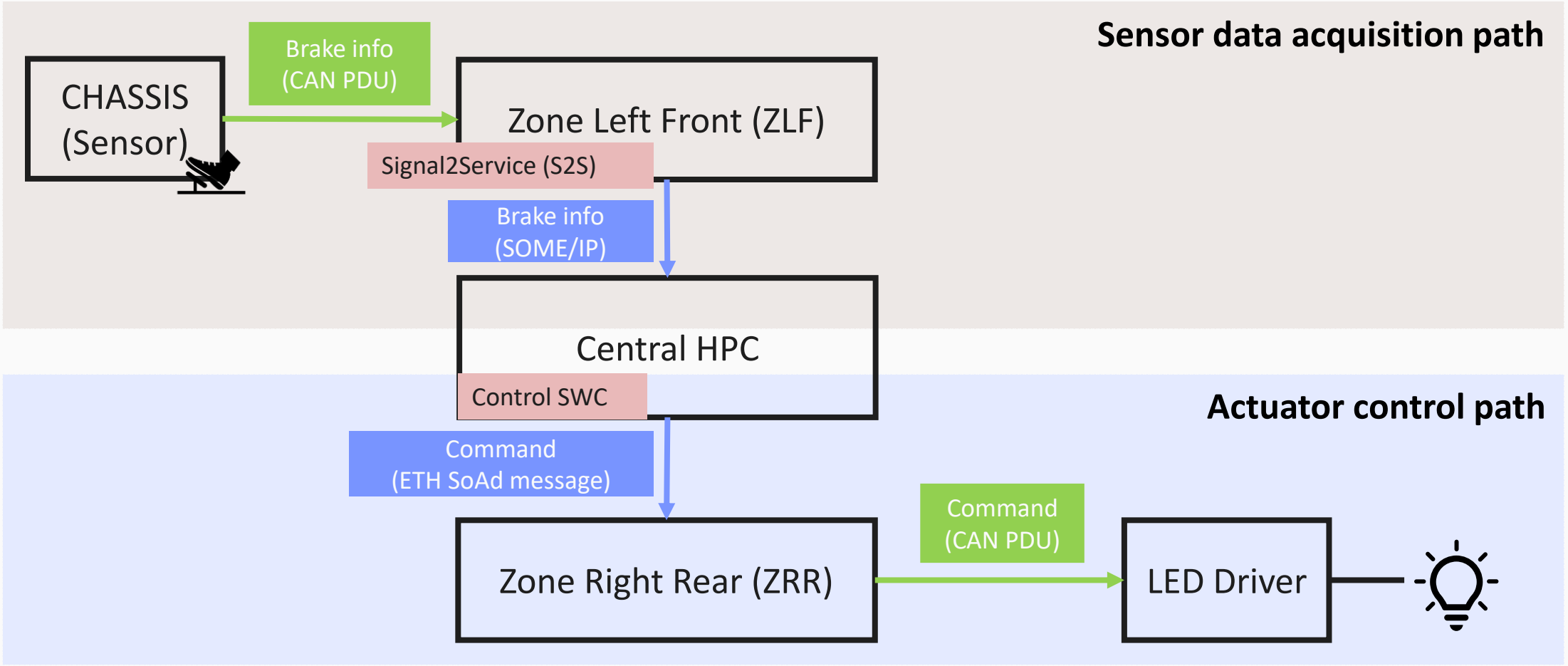


- **Automotive EE architecture today: multiple communication protocols co-exist**
- **End-to-End Latency challenge**:
  - Domain **EE architecture**: Sensor and actuator are either locally managed by the same ECU, or on CAN networks with bounded latency.
  - **Zonal EE architecture**: Sensor and actuator can be separated by Ethernet backbone. Additional protocols (SOME/IP) and handlings (S2S: Signal2Service/Service2Signal) can introduce extra latency.

    ➔ Need to guarantee End-to-End latency for real-time applications.
  - **Example**: Brake ➔ Stop lamps

QOS CHALLENGES IN A MULTI-PROTOCOL EE ARCHITECTURE USE CASE ANALYSIS

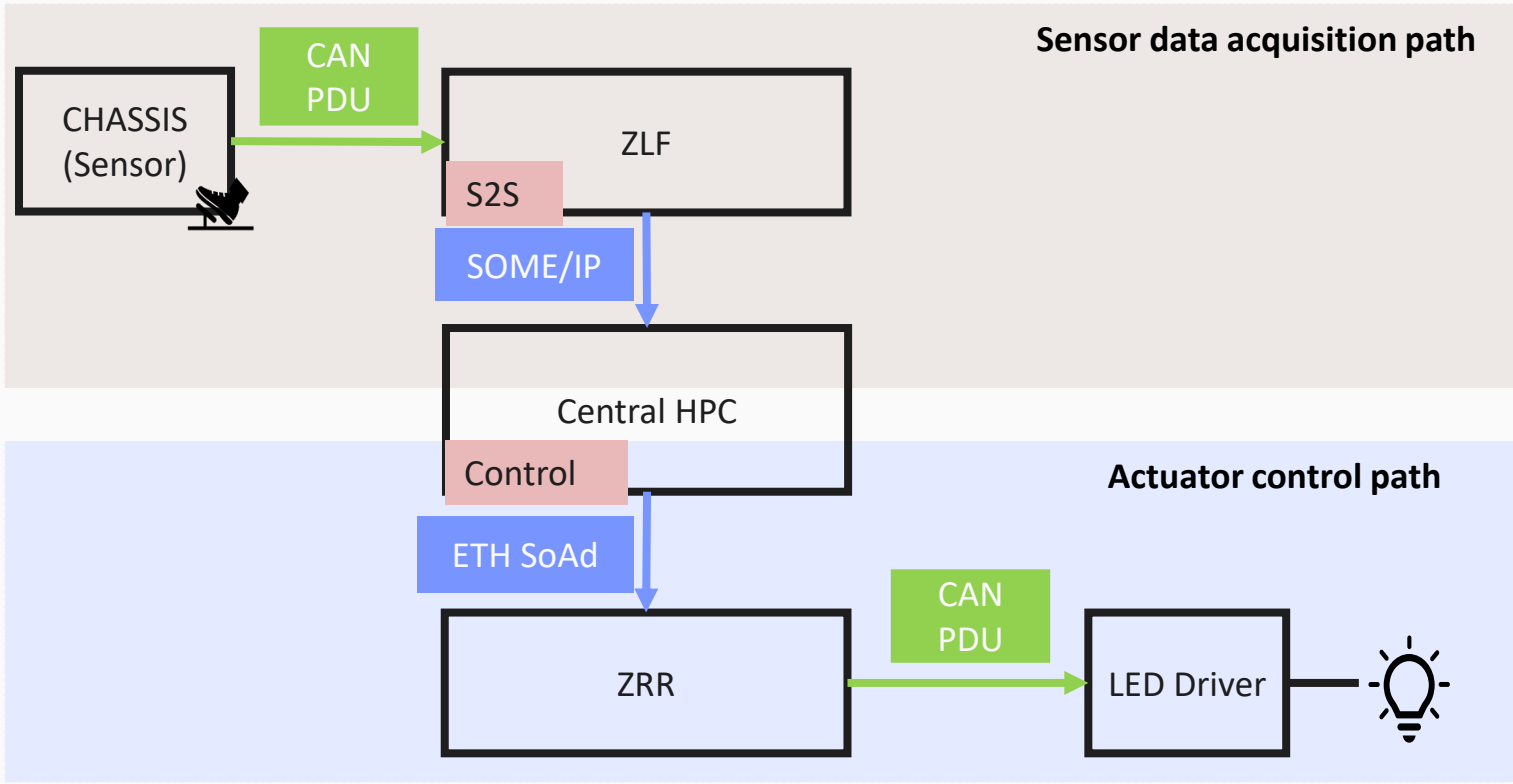
- **Use Case**: Brake ➔ Stop lamps ON
  - **End-to-End constraint**: 100ms
  - **Sensor data acquisition:** Brake info
  - **Actuator control:** Stop lamps ON command



END-TO-END LATENCY ANALYSIS: USE CASE STOP LAMPS

WORST-CASE ANALYSIS

- **Traffic Model**:
  - Brake info:
    - cyclically sent every 10ms.
    - CAN message + SOME/IP service
  - Stop lamps control command:
    - cyclically sent every 10ms.
    - CAN message + ETH SoAd message
- **Worst-Case (WC) End-to-End analysis considers**:
  - SW handling latency: 70ms ➡ 51%
  - COM stack latency: 61ms ➡ 44%
  - ETH network access time: 5ms ➡ 4%
  - CAN bus access time: 2ms ➡ <1%



ECU	CHASSIS			ZLF				Central HPC				ZRR			LED Driver		Total
Path	App SW	CAN Tx Com	CAN access	CAN Rx Com	App (S2S)	ETH Tx Com	ETH access	ETH Rx Com	App SW	ETH Tx Com	ETH access	ETH Rx Com	CAN Tx Com	CAN access	CAN Rx Com	App SW	
WC latency	10ms	10ms	1ms	10ms	10ms	10ms + 5ms	2.5ms	10ms	10ms	10ms	2.5ms	10ms	1ms	1ms	5ms	10ms	118ms

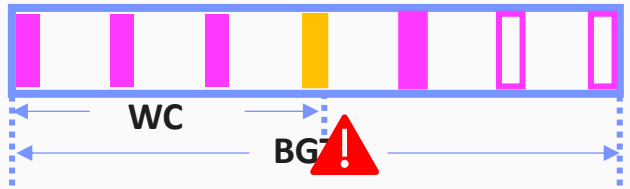
Assumptions:
- CAN Rx by IT, ETH Rx by polling
- Cross-core communication for Central HPC and ZC
- CBS (Credit-Based Shaping) is implemented

Example

END-TO-END LATENCY ANALYSIS: USE CASE STOP LAMPS BUDGET ANALYSIS

- **Worst-Case End-to-End latency break-down**:
  - SW handling latency: 70ms ➡ 51%
  - COM stack latency: 61ms ➡ 44%
  - ETH network access time: 5ms ➡ 4% (dependency on message set)
  - CAN bus access time: 2ms ➡ <1% (dependency on message set)

- **Latency Budget (BGT) based analysis allows reservation for future new Use Cases**
  ➔ Scalable network architecture (SDV)



ECU	CHASSIS			ZLF				Central HPC				ZRR			LED Driver		Totol
Path	App SW	CAN Tx Com	CAN access	CAN Rx Com	App (S2S)	ETH Tx Com	ETH access	ETH Rx Com	App SW	ETH Tx Com	ETH access	ETH Rx Com	CAN Tx Com	CAN access	CAN Rx Com	App SW	
WC latency	10ms	10ms	1ms	10ms	10ms	10ms + 5ms	2.5ms	10ms	10ms	10ms	2.5ms	10ms	1ms	1ms	5ms	10ms	**118ms**

Example

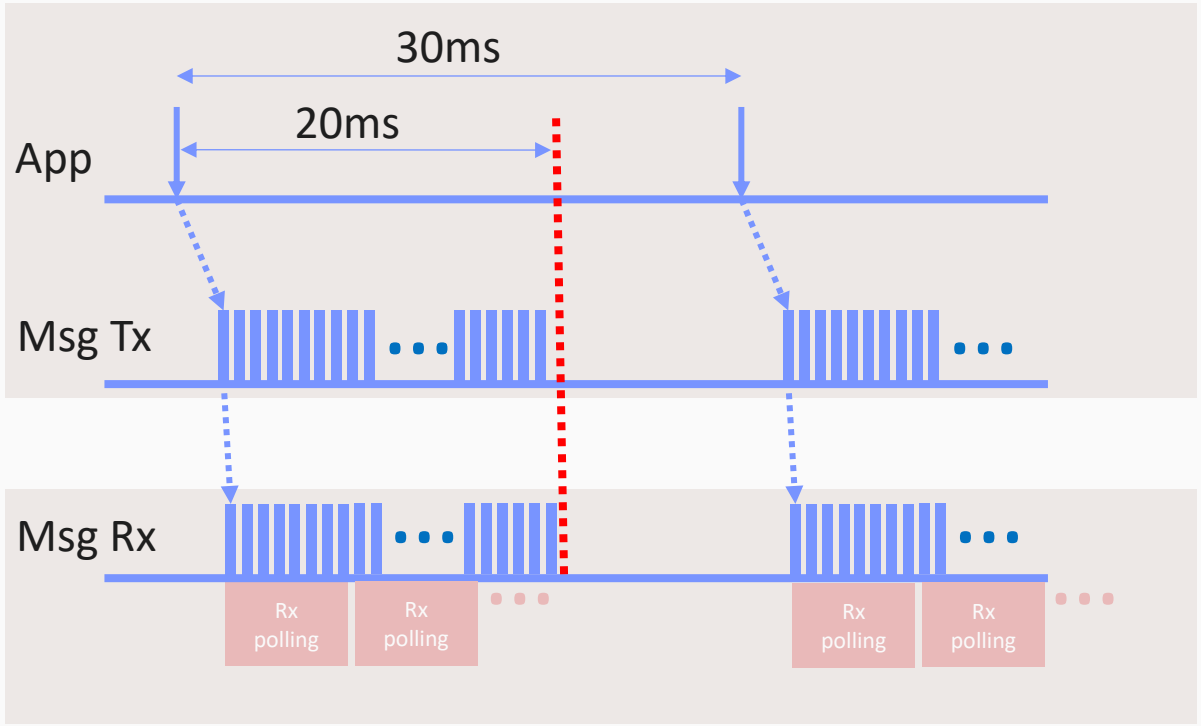
Assumptions:
- CAN Rx by IT, ETH Rx by polling
- Cross-core communication for Central HPC and ZC
- CBS (Credit-Based Shaping) is implemented

DESIGN RULE FOR NETWORK LATENCY BUDGET VALUE

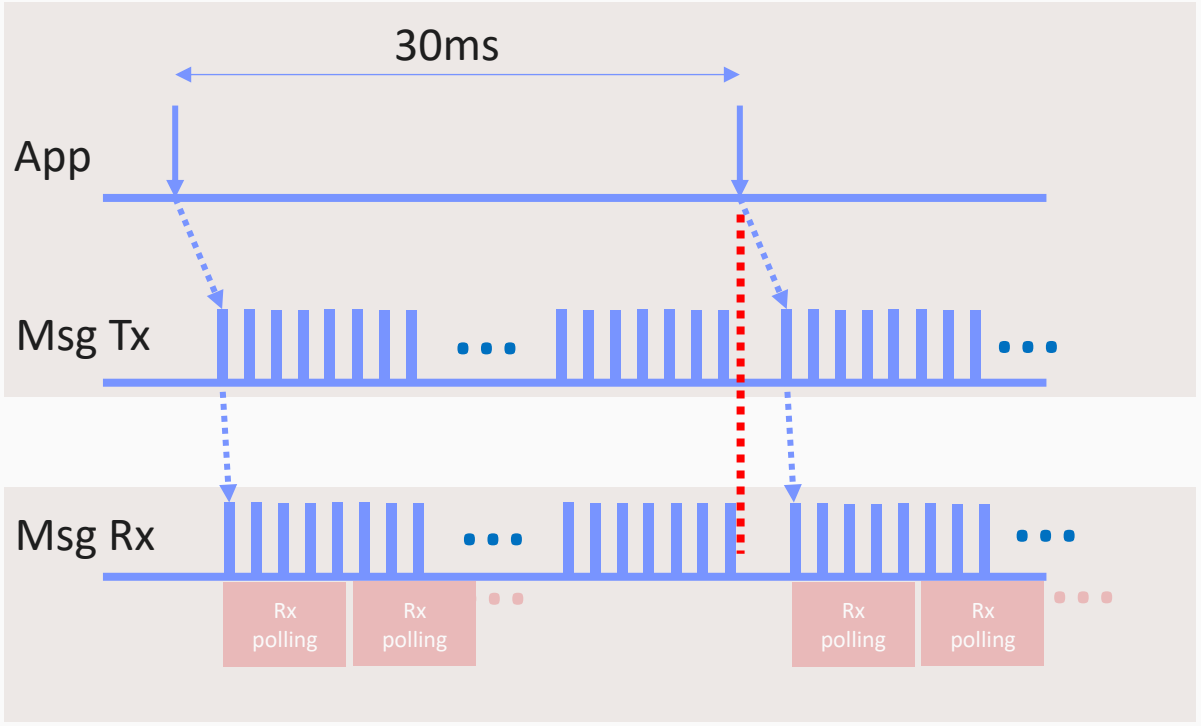
The definition of network latency budget value is OEM specific, but shall take into account:

- Buffer Usage
- Latency constraint
- Example: App sends 30k bytes every 30ms ➔ Each Ethernet frame takes 1k byte in payload.

- **Case 1**: Latency budget: **20ms**
  - Tx Req: 30 frames within **20ms**
  - CBS config: **13Mbps**
  - Rx buffer: **8 frames/5ms**

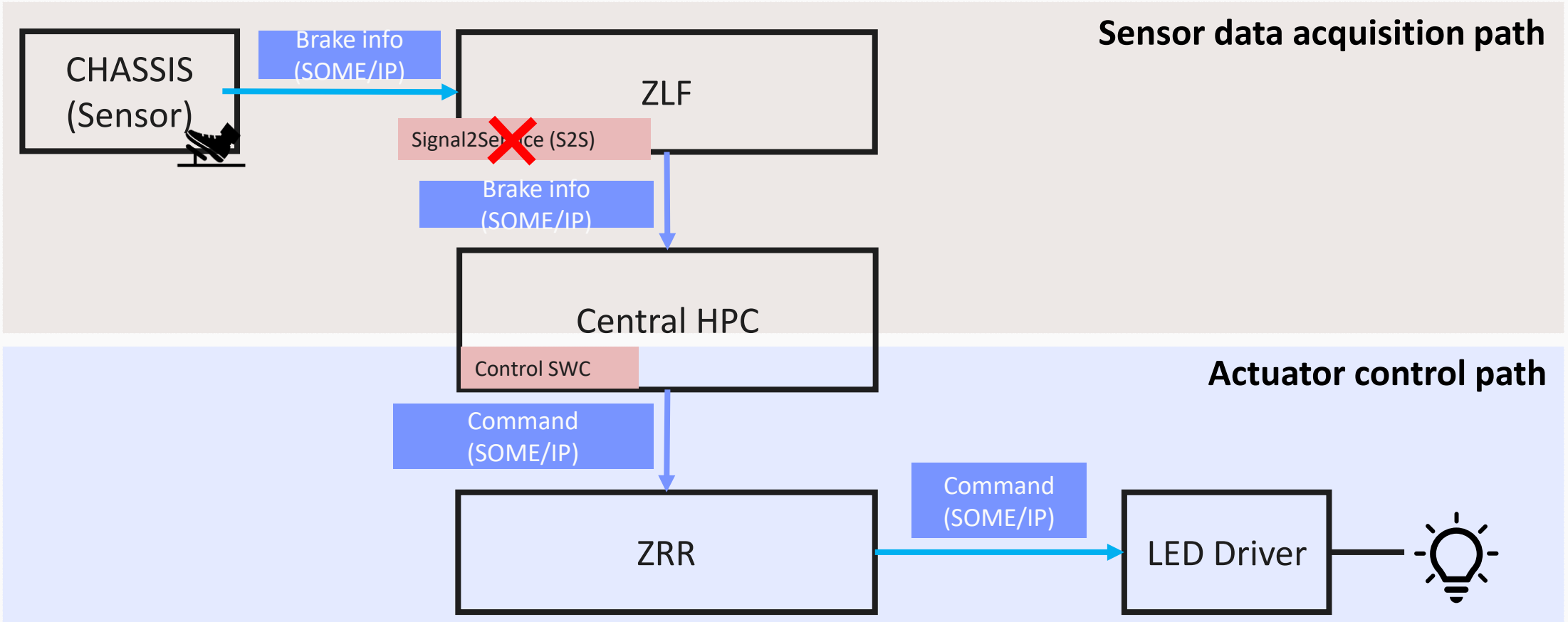


- **Case 2**: Latency budget: **30ms**
  - Tx Req: 30 frames within **30ms**
  - CBS config: **9Mbps**
  - Rx buffer: **5 frames/5ms**



END-TO-END LATENCY ANALYSIS: USE CASE STOP LAMP MOVE TO FULL ETHERNET EE ARCHITECTURE

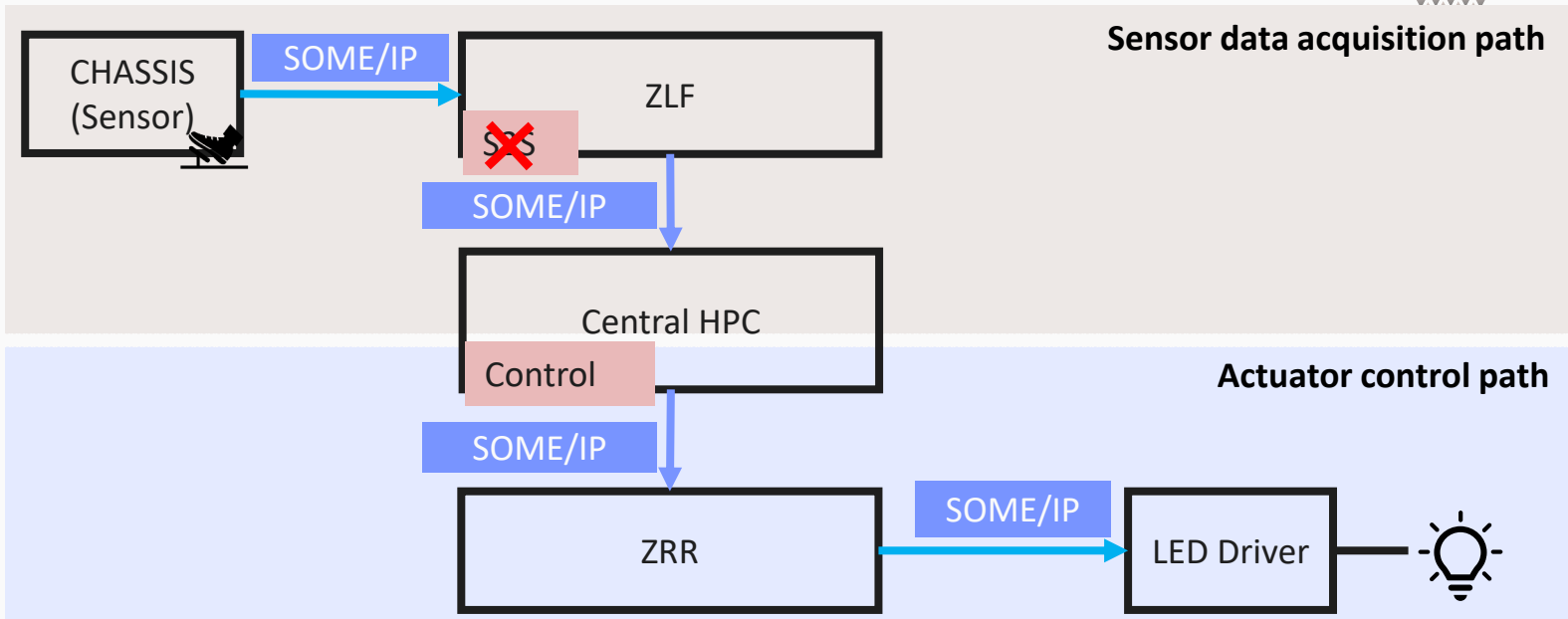
- **Use Case**: Brake ➔ Stop lamps ON
  - **End-to-End constraint**: 100ms
  - **Sensor data acquisition:** Brake info
  - **Actuator control:** Stop lamps ON command



END-TO-END LATENCY ANALYSIS: USE CASE STOP LAMPS MOVE TO FULL ETHERNET EE ARCHITECTURE

Unified backbone & Simplified software stack

Challenge: Ensuring deterministic latency, especially with PLCA delays



ECU	CHASSIS			ZLF	Central HPC				ZRR	LED Driver		Totol
Path	App SW	**ETH** Tx Com	**ETH** access **(T1S)**	ETH Switching	ETH Rx Com	App SW	ETH Tx Com	ETH access	ETH Swtching **(T1S)**	**ETH** Rx Com	App SW	
WC latency	10ms	10ms+ 5ms	?	2.5ms	10ms	10ms	10ms+5 ms	2.5ms	?	5ms	10ms	?
BGT latency	10ms	10ms + 5ms	5ms	5ms	10ms	10ms	10ms + 5ms	5ms	5ms	5ms	10ms	95ms

Assumptions:
- ETH Rx by polling
- Cross-core communication for Central HPC and ZC
- CBS (Credit-Based Shaping) is implemented

Example

RTaW
RealTime-at-Work

NETWORK DESIGN
100BASE-T1 + T1S-PLCA

JORN MIGGE
XIAOJIE GUO

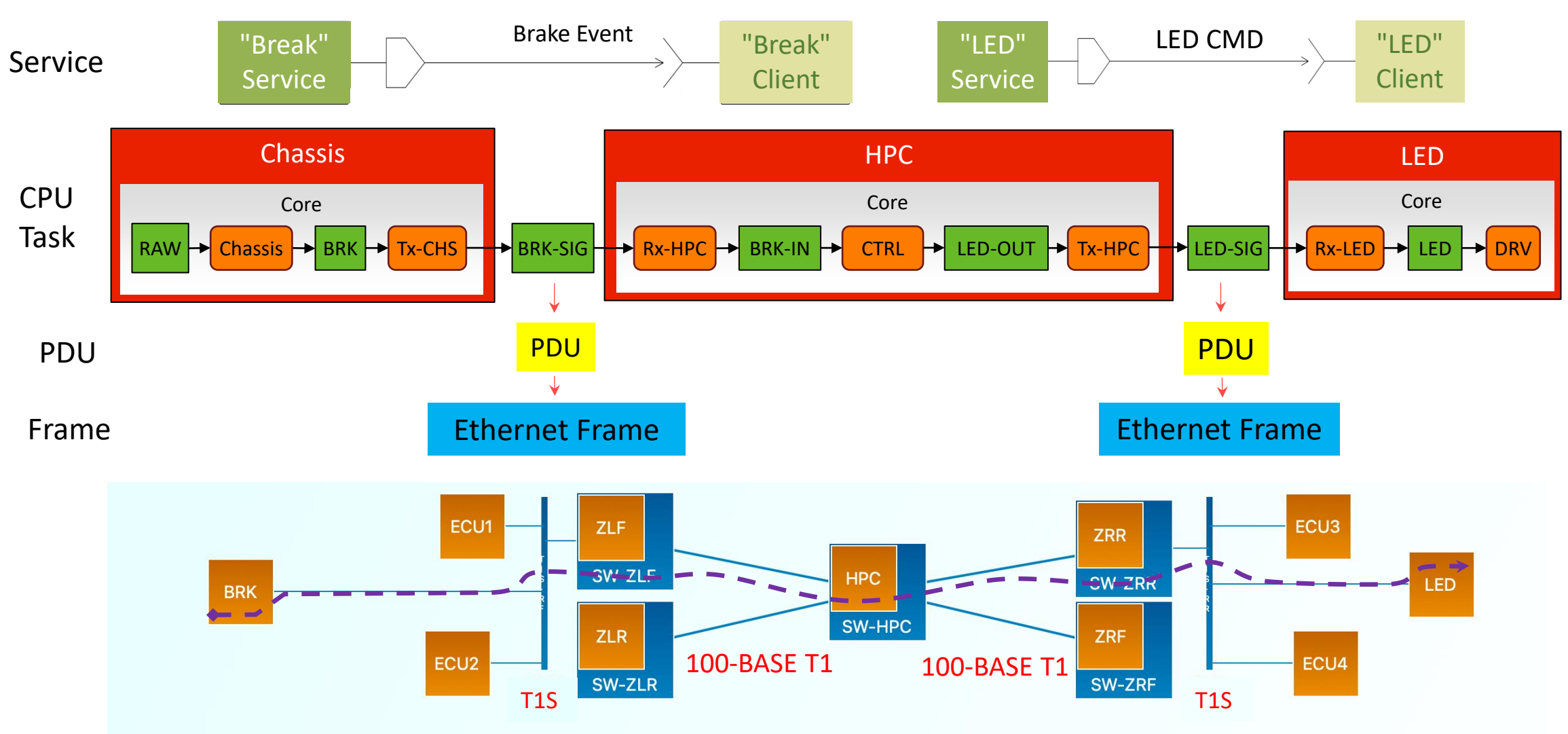
End-To-End Timing Chain

RTaW RealTime-at-Work

Application

Traffic

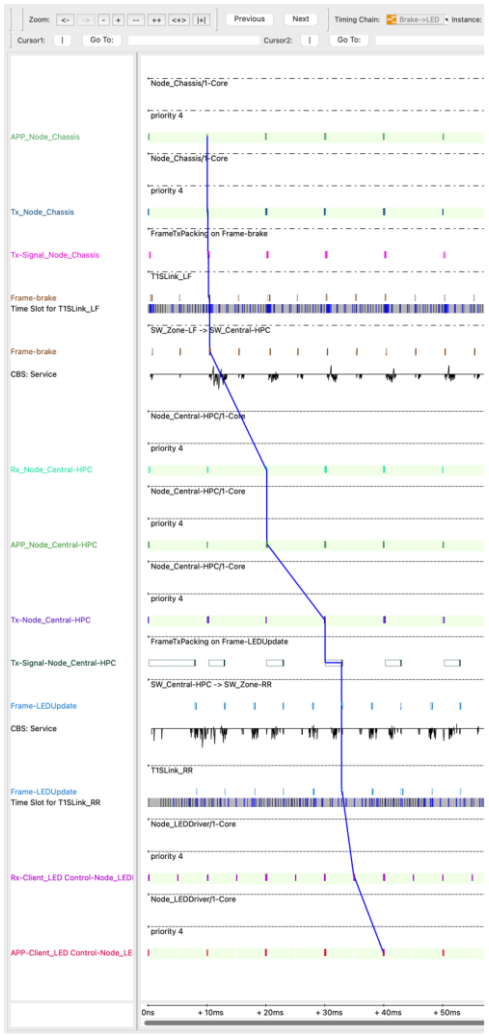
Hardware



RTaW-Pegase screenshots

End-to-End Latency Breakdown

		Segment	Min	Average	Max	Bound / Window
Chassis		Waiting for 'APP_Node_Chassis'				10.000 ms
		APP_Node_Chassis	0.100 ms	0.148 ms	0.201 ms	0.201 ms
		'APP-Signal_Node_Chassis' waiting	0.000 ms	4.084 ms	9.802 ms	10.000 ms
		Tx_Node_Chassis	0.100 ms	0.152 ms	0.201 ms	0.201 ms
		'Tx-Signal_Node_Chassis' waiting for	0.084 ms	0.133 ms	0.185 ms	5.000 ms
T1S		Node_Chassis-> SW_Zone-LF	0.107 ms	0.228 ms	0.878 ms	1.063 ms
Zone LF → HPC		SW_Zone-LF-> SW_Central-HPC	0.010 ms	0.155 ms	2.634 ms	2.726 ms
Central HPC		'Tx-Signal_Node_Chassis' waiting	0.018 ms	5.126 ms	9.473 ms	10.003 ms
		Rx_Node_Central-HPC	0.099 ms	0.198 ms	0.300 ms	0.301 ms
		'Rx-Signal-Node_Central-HPC' waiting	0.000 ms	4.870 ms	9.800 ms	10.000 ms
		APP_Node_Central-HPC	0.099 ms	0.201 ms	0.300 ms	0.301 ms
		'APP-Signal-Node_Central-HPC' waiti	0.000 ms	5.674 ms	9.800 ms	10.000 ms
		Tx-Node_Central-HPC	0.099 ms	0.200 ms	0.300 ms	0.301 ms
		'Tx-Signal-Node_Central-HPC' waiting	2.606 ms	2.707 ms	7.807 ms	5.000 ms
HPC → Zone RR		SW_Central-HPC-> SW_Zone-RR	0.010 ms	0.099 ms	0.509 ms	1.791 ms
T1S		SW_Zone-RR-> Node_LEDDriver	0.107 ms	0.140 ms	0.494 ms	0.893 ms
LED Driver		'Tx-Signal-Node_Central-HPC' waiting	0.089 ms	2.632 ms	4.951 ms	5.002 ms
		Rx-Client_LED Control-Node_LEDDriv	0.100 ms	0.125 ms	0.201 ms	0.201 ms
		'Rx-Signal-Node_LEDDriver' waiting	0.000 ms	5.308 ms	9.801 ms	10.000 ms
		APP-Client_LED Control-Node_LEDDr	0.100 ms	0.150 ms	0.201 ms	0.201 ms



Brake Event -> LED CMD
- Constraints: 100 ms

Time Budget verification:
- Worst-Case Analysis
- WC latency: 83,184 ms

Ethernet Segments:
- Network Calculus
  - T1S
  - T1+CBS

⇒ Varying magnitudes of sub-delay

RTaW-Pegase screenshots

End-to-End Latency Breakdown

RTaW RealTime-at-Work

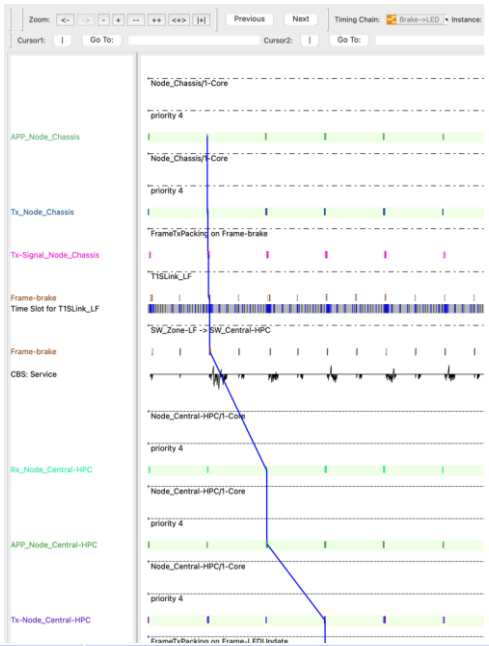
Chassis

T1S

Zone LF → HPC

Central HPC

	Segment	Min	Average	Max	Bound / Window
	Waiting for 'APP_Node_Chassis'				10.000 ms
	APP_Node_Chassis	0.100 ms	0.148 ms	0.201 ms	0.201 ms
	'APP-Signal_Node_Chassis' waiting	0.000 ms	4.084 ms	9.802 ms	10.000 ms
	Tx_Node_Chassis	0.100 ms	0.152 ms	0.201 ms	0.201 ms
	'Tx-Signal_Node_Chassis' waiting for	0.084 ms	0.133 ms	0.185 ms	5.000 ms
	Node_Chassis-> SW_Zone-LF	0.107 ms	0.228 ms	0.878 ms	1.063 ms
	SW_Zone-LF-> SW_Central-HPC	0.010 ms	0.155 ms	2.634 ms	2.726 ms
	'Tx-Signal_Node_Chassis' waiting	0.018 ms	5.126 ms	9.473 ms	10.003 ms
	Rx_Node_Central-HPC	0.099 ms	0.198 ms	0.300 ms	0.301 ms
	'Rx-Signal-Node_Central-HPC' waiting	0.000 ms	4.870 ms	9.800 ms	10.000 ms
	APP_Node_Central-HPC	0.099 ms	0.201 ms	0.300 ms	0.301 ms
	'APP-Signal-Node_Central-HPC' waiti	0.000 ms	5.674 ms	9.800 ms	10.000 ms



Brake Event -> LED CMD
- Constraints: 100 ms

Time Budget verification:
- Worst-Case Analysis
- WC latency: 83,184 ms

Ethernet Segments:
- Network Calculus
  - T1S
  - T1+CBS

ECU	CHASSIS			ZLF	Central HPC				ZRR	LED Driver		Total
Path	App SW	**ETH** Tx Com	**ETH** access **(T1S)**	ETH Switching	ETH Rx Com	App SW	ETH Tx Com	ETH access	ETH **(T1S)**	**ETH** Rx Com	App SW	
WC latency	10ms	10ms+ 5ms	**1.06ms**	**2.72ms**	10ms	10ms	10ms + 5ms	**1.79ms**	**0.89ms**	5ms	10ms	**82ms**
BGT latency	10ms	10ms + 5ms	5ms	5ms	10ms	10ms	10ms + 5ms	5ms	5ms	5ms	10ms	**95ms**

But how much traffic can be supported?

RTaW-Pegase screenshots

Network Delays: Future-Proof Design

RTaW RealTime-at-Work

Optimal design when traffic increases:

1. If a **less time critical** frame is **added**, the impact on existing, more time critical frames, should be as low as possible.
2. If a **more time critical** frame is **added**, it should be possible to limit the interference from existing less time critical frames.

More concretely:

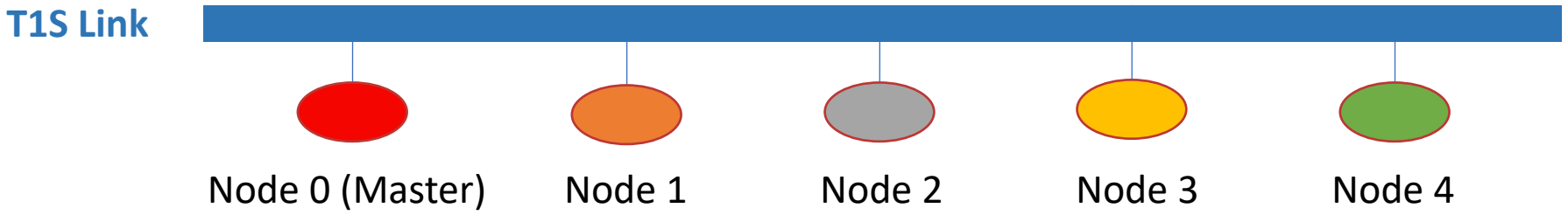
- A new **100 ms frame** should **"not really impact"** an existing **5 ms frame**
- A new **5 ms frame** should "**not really be impacted"** by an existing **100 ms frame**

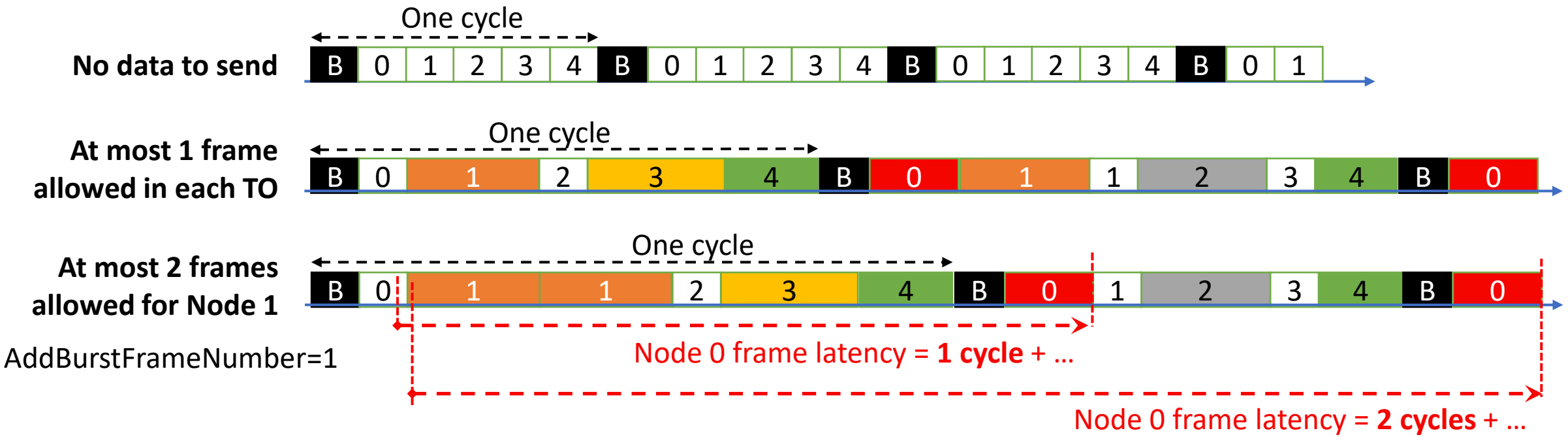
Depends on the "features" of the scheduling mechanism

- CAN: IDs play the role of priorities -> very efficient
- And T1S?

Note: the question is not if T1S is better or worse than CAN, but if in the context of an all-Ethernet topology, we can find good solutions with T1S.

Recap: T1S/PLCA Mechanism Overview

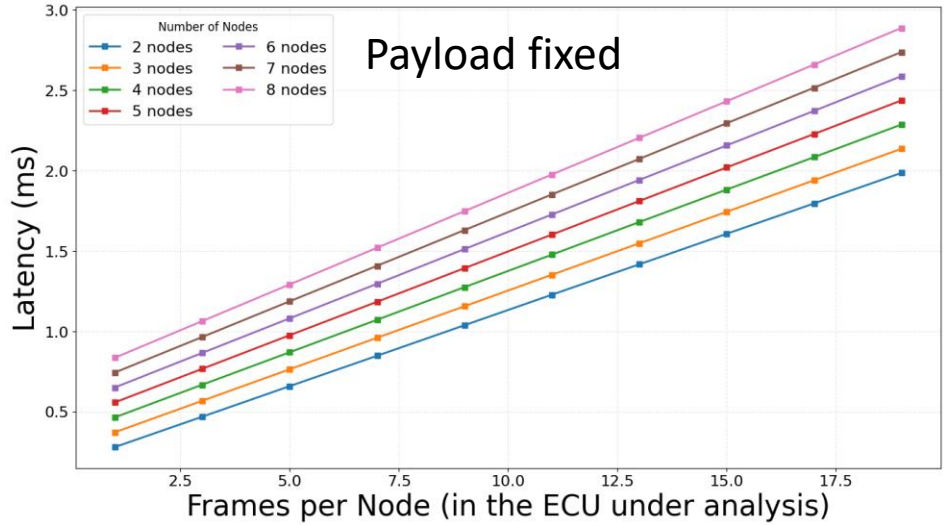


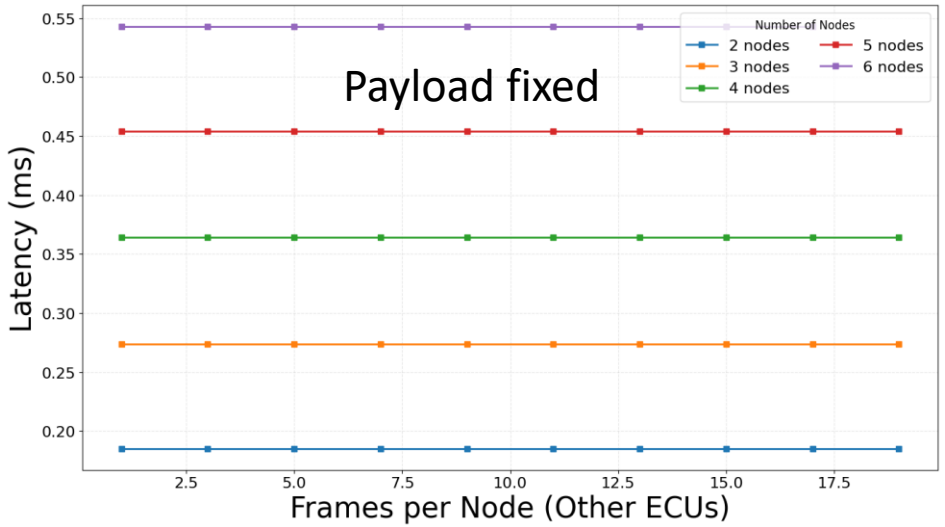


TO: Transmission Opportunity

Worst-Case Delay ≤ Max T1S cycle length * (number of previously queued frames)

Exploring How Configuration Choices Affect T1S Latency

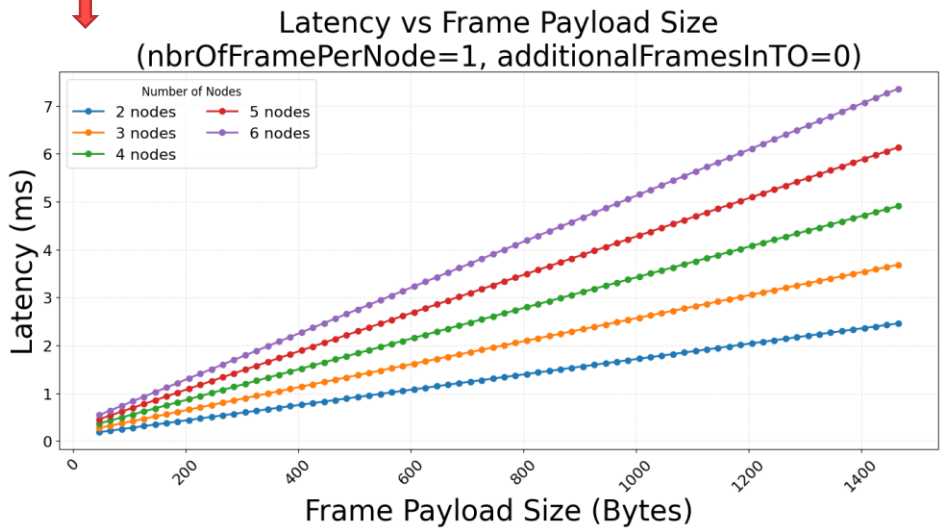




✓For every frame added to **the same ECU**, the impact is an entire T1S cycle

✓If an additional station is added, the T1S cycle becomes longer

✓If frames are added to **other ECUs**, impact only due to larger frames sizes



Exploring How Configuration Choices Affect T1S Latency

AddBurstFrameNumber > 0 (more than 1 frame per transmission opportunity)

- **reduces** the latency for the ECU's own frames, because they need to wait less T1S cycles
- **increases** the latency for other ECU's frames, because the T1S cycles become longer

→ helps only in particular cases, where few nodes have more critical frames than all others

Conclusions

- Latencies over T1S are determined by the
  - number of T1S cycles a frame must wait for its transmission opportunity
  - length of T1S cycles
- T1S mechanisms alone are not efficient for scheduling frames with different time criticalities, since all frame sent by a node suffer the same worst-case delay.

Topology Stress Test®(TST):
Overload Analysis on T1S - 10Mbit/s

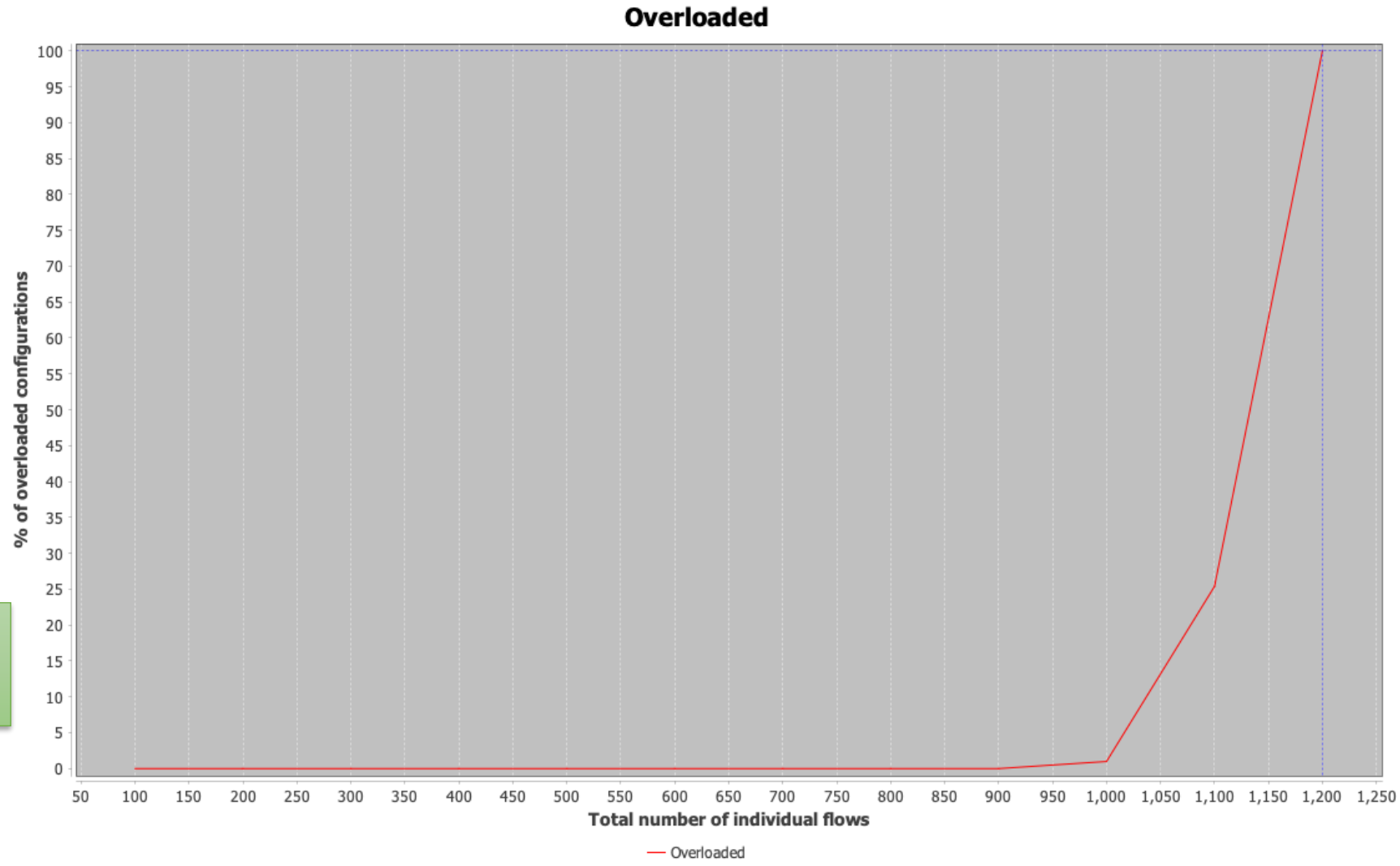
Frame generation characteristics:

- **Payload size 46 - 64 bytes**
- **Deadline = Period**

Period	Weight
5ms	8 %
10ms	14 %
20ms	26 %
50ms	26 %
100ms	26 %

T1S sustains up to ~1000 frames without overload in 98% of cases.

Note: both commit signals and beacon frames consume bandwidth in T1S



Topology Stress Test®(TST):
Deadline-Constrained Functional Scalability on TTS

RTaW RealTime-at-Work

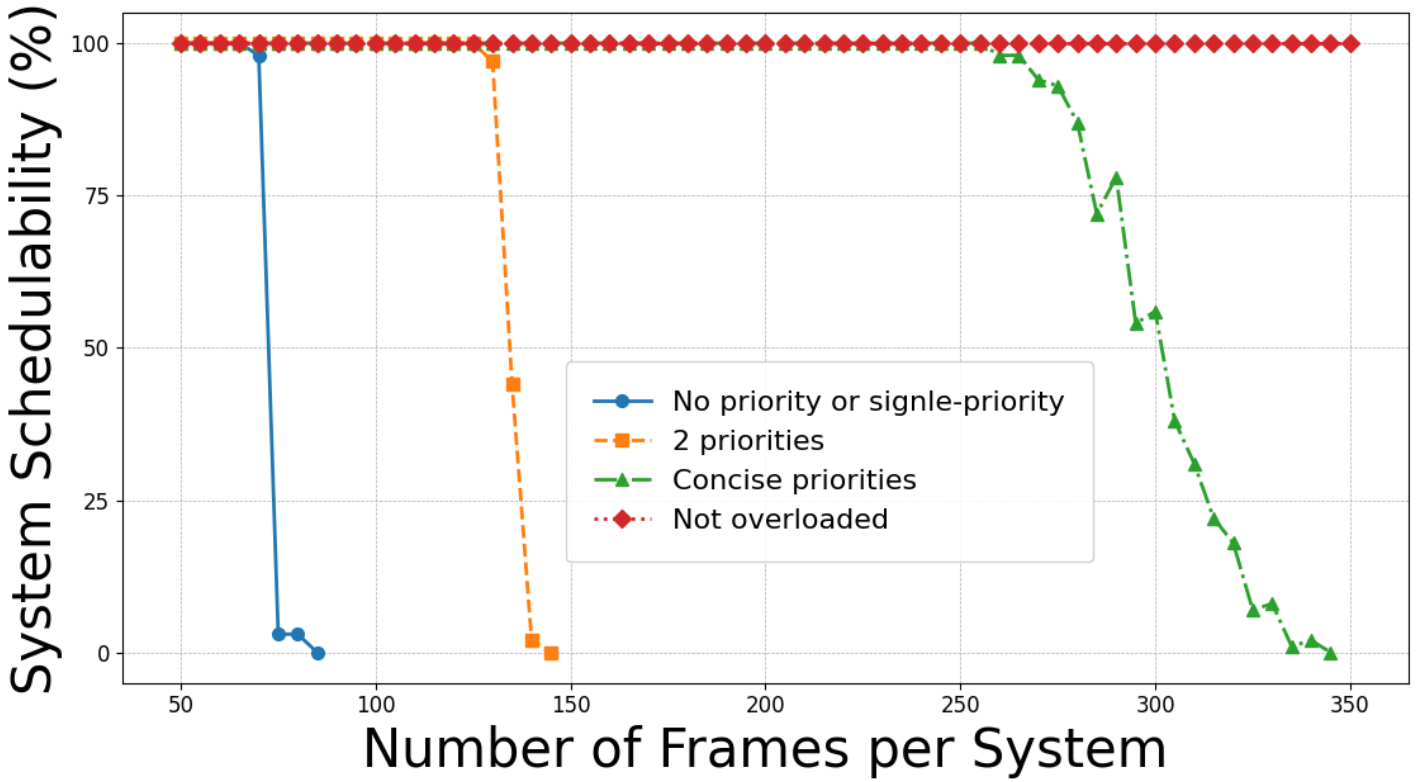
System capacity using priorities:

No priority/Single priority
70 frames

+ 85% with 2 priorities
130 frames

+ 278% with Concise Priorities®
up to 8 priorities: 265 frames

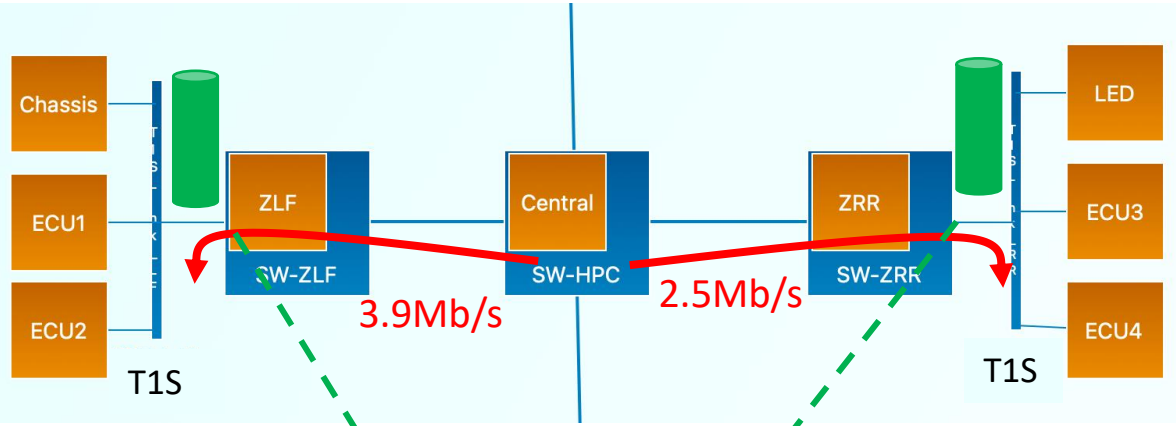
(optimal priorities assignment)



⇒ priorities are efficient for increasing schedulability

Edge Switch Port Memory: 100baseT1 → T1S

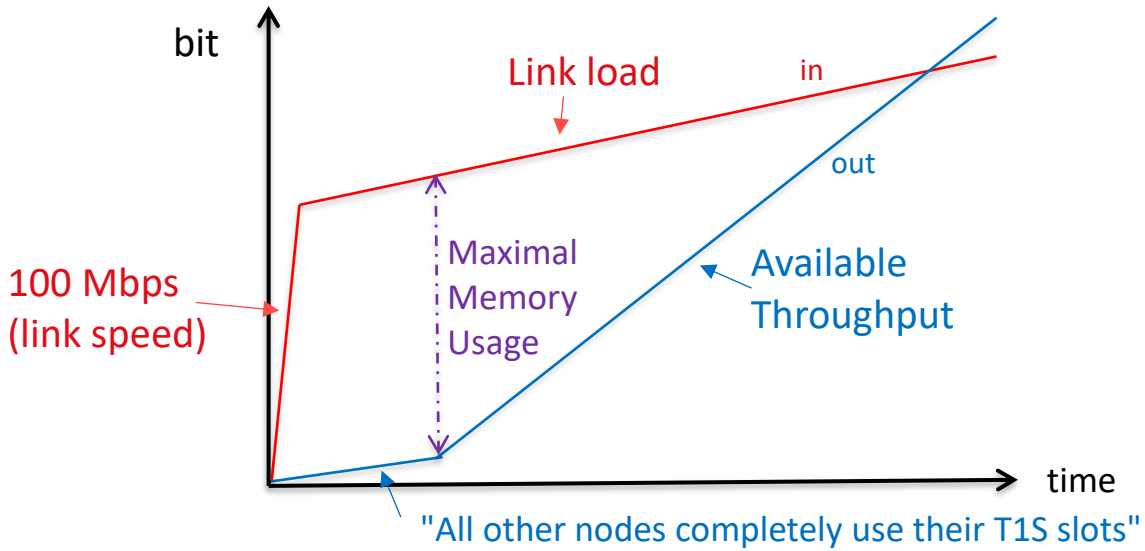
- Frame drops may occur in the edge switch towards T1S because of the necessity to store more frames due to the speed reduction

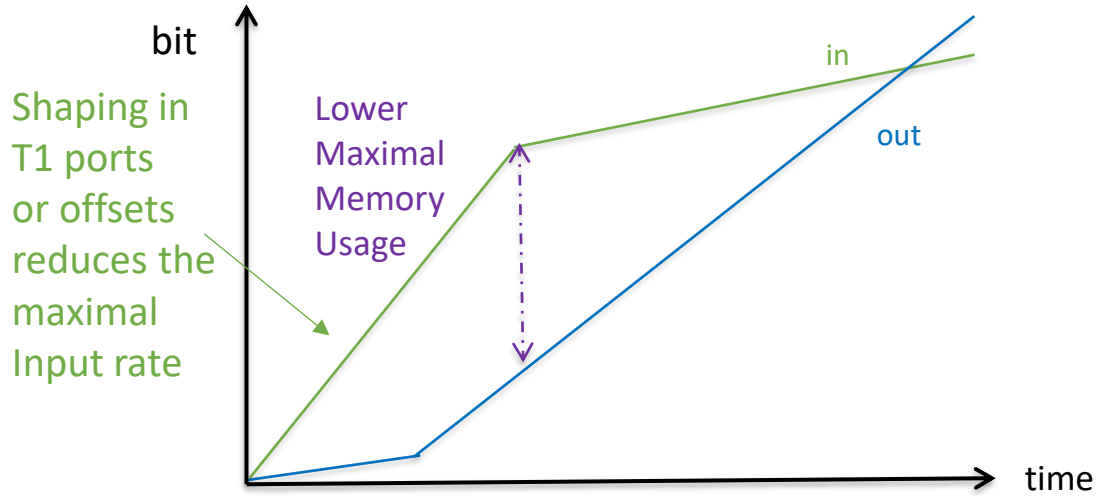


	Zone 1 (->T1sLink LF)	Zone 2 (-> T1SLink RR)
Without CBS	11134 bytes	6190 bytes
With CBS	**9346 bytes**	**4646 bytes**

Buffer size limit: 10k/port

Shaping in T1 reduces memory requirements in T1S port, but increases delays => trade-off must be found





Takeaways & Future Work

Takeaways

- **Latency:** a crucial challenge for **multi-protocol** Zonal EE architecture
- **Scalable latency analysis** needs a **budget-based** approach
- **10BASE-T1S:** avoid protocol gatewaying + gain resources as well as latency
- T1S+PLCA alone CANNOT separate time critical from less time critical traffic, **but traffic classes and priorities** allow to find solutions in an all-Ethernet context
- Shaping of backbone traffic in T1S ports may allow to reduce **memory** requirements in edge switch port towards T1S, but also increases **latencies** => **tradeoff** to be made
- The 10BASE-T1S topology and PLCA configuration: **important** impact on latency

➔ Shall be carefully addressed

Future Work

- Identify critical use case
- Investigation on 10BASE-T1S topology and PLCA configuration strategy
- Investigation of transmission offsets that spread out traffic bursts for reducing delays and memory requirements.

Thank you

Questions?

Contact: jorn.migge@realtimeatwork.com
C.T.O. RTaW