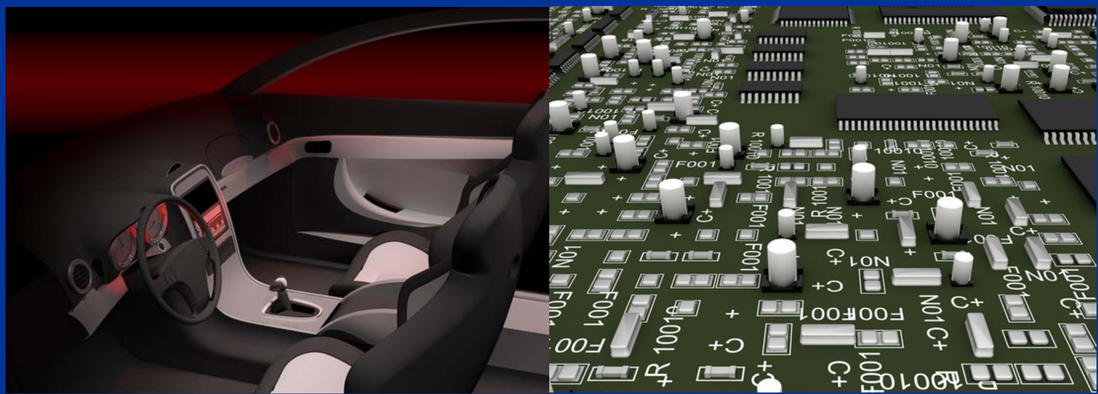
Automotive embedded systems: some research challenges

Nicolas Navet
nicolas.navet@realtimeatwork.com



ETR'2009, Paris
03/09/2009

Better technical solutions for real-time systems

Electronics is the driving force of innovation

- 90% of new functions use software
- Electronics: 40% of total costs
- Huge complexity! 70 ECUs, 2000 signals, 6 networks, multi-layered run-time environment (AUTOSAR), multi-source software, multi-core CPUs, etc

Strong costs, safety, reliability, time-to-market, reusability, legal constraints !

Many issues in the design of E/E systems are not strictly technical!

Eg. Key issues in architecture development at Volvo in paper ref[2]

- Lack of background in E/E at management level (often mechanical background)
- Influence of E/E architecture wrt to business value? Lacks long term strategy
- Lack of clear strategy between in-house and externalized developments
- Technical parameters are regarded as less important than cost for supplier / components selection

Key issues in architecture development at Volvo in paper ref[2] (2/2)

- How to share architecture/sub-systems between several brands/models with different constraints/objectives?
- Sub-optimal solutions for each component / function
- Architectural decisions often:
  - are made on experience / gut feeling (poor tool support)
  - Lacks well-accepted process

Where to tackle the problem from a technical point of view? (see ref[3])

- **Design** : model functional and non-functional features ⇨ software components, MDD, etc
- **Validation / Analysis** : dependability, (end-to-end) response time, memory consumption (e.g. buffers), deadlocks, etc
- **Synthesis** : remove unused features, mapping of components to runtime objects (ECU/Tasks), setting runtime parameters (priorities, offsets, etc)
- **Runtime mechanisms** : OS, protocols, drivers, NM, diagnostics, etc

Validation is a key activity!
Personal view on the developments

Mostly ahead of us!

Talk part 2

Talk part 1

« correctness by construct » and optimal synthesis

Probabilistic analysis system level

« Worst-case » deterministic analysis system level

Probabilistic analysis (sub-system)

« Worst-case » deterministic analysis (sub-system)

« Smart » monitoring tools

Simulation tools (co-simulation, HIL)

1994 1997 2009

INRIA

RTaW RealTime-at-Work

Part 1 - probabilistic framework for schedulability analysis: illustration on the aperiodic traffic on the bus

(joint work with PSA Peugeot-Citroën
see paper ref[5])

Probabilistic analysis is needed

- Systems are not designed for the worst-case (provided it is rare enough!)
- Reliability/Safety are naturally expressed and assessed in terms of probability (e.g. < 10^-9 per hour)
- Deterministic assumptions are sometimes unrealistic or too pessimistic, e.g.:
  - Worst-Case Execution Time on modern platforms,
  - Aperiodic activities: ISR, frame reception,
  - ...
- Faults/errors are not deterministic (and better modeled probabilistically)

Accounting for the aperiodic traffic

- Transmission patterns can hardly be characterized: purely aperiodic, mixed periodic/aperiodic, etc
- Aperiodic frames do jeopardize RT constraints
- Few approaches in the litterature:
  - deterministic approaches, such as sporadic, generally lead to unusable results (e.g., $\rho>1$)
  - Average case probabilistic approach not suited to dependability-constrained systems
  - Probabilistic approaches with safety adjustable level, see paper ref[6] and ref[7]

INRIA RTaW RealTime-at-Work

Approach advocated here

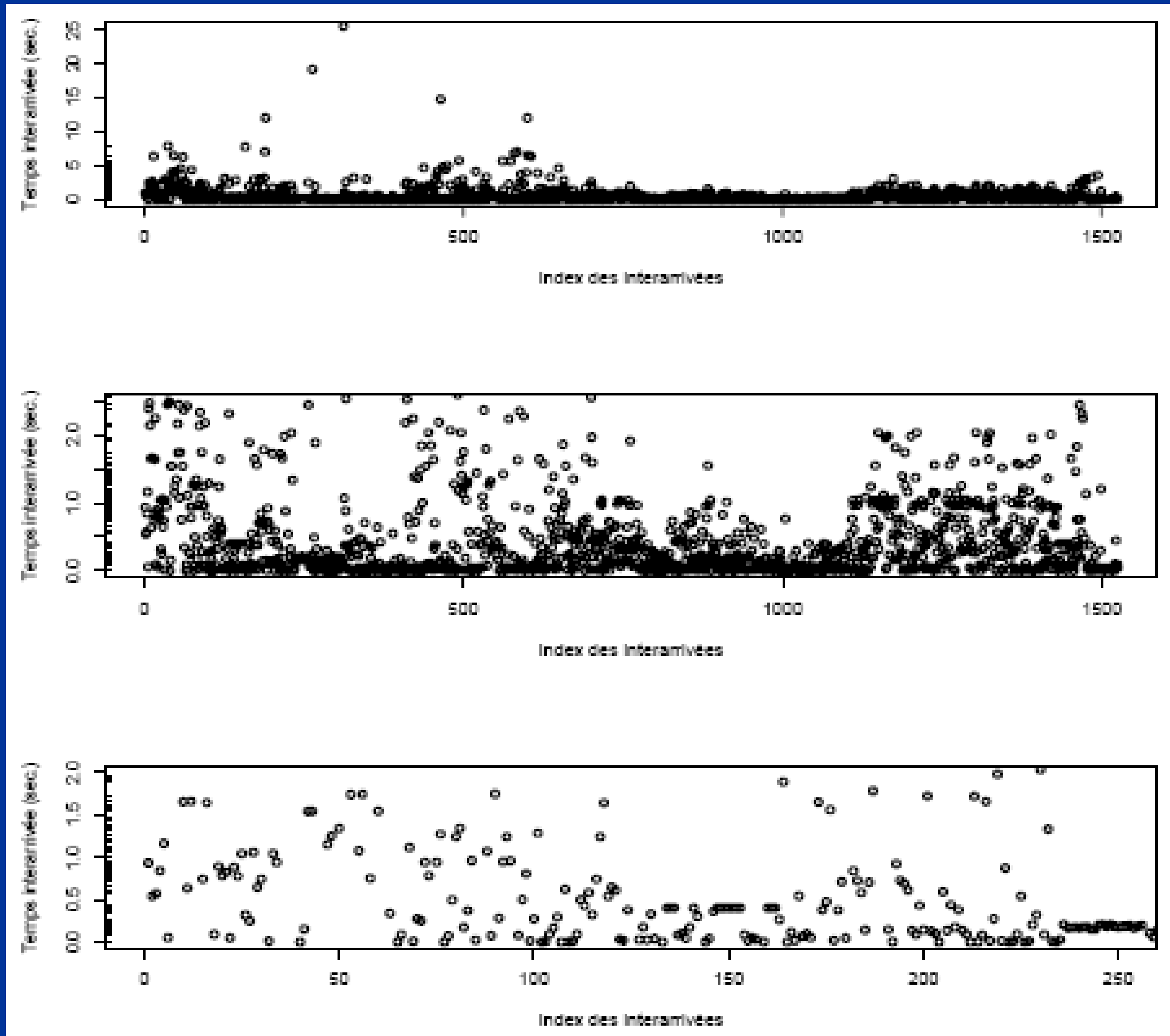
1) Measurements / data cleaning
2) Modeling aperiodic traffic arrival process
3) Deriving aperiodic Work Arrival Process (WAF)
4) Integrating aperiodic WAF into schedulability analysis

Data trace analysis

y: aperiodic interarrival times – x: index of interarrivals

ZOOM +

Approximate because what is seen on the bus is not the actual arrival process at ECU level! can be handled

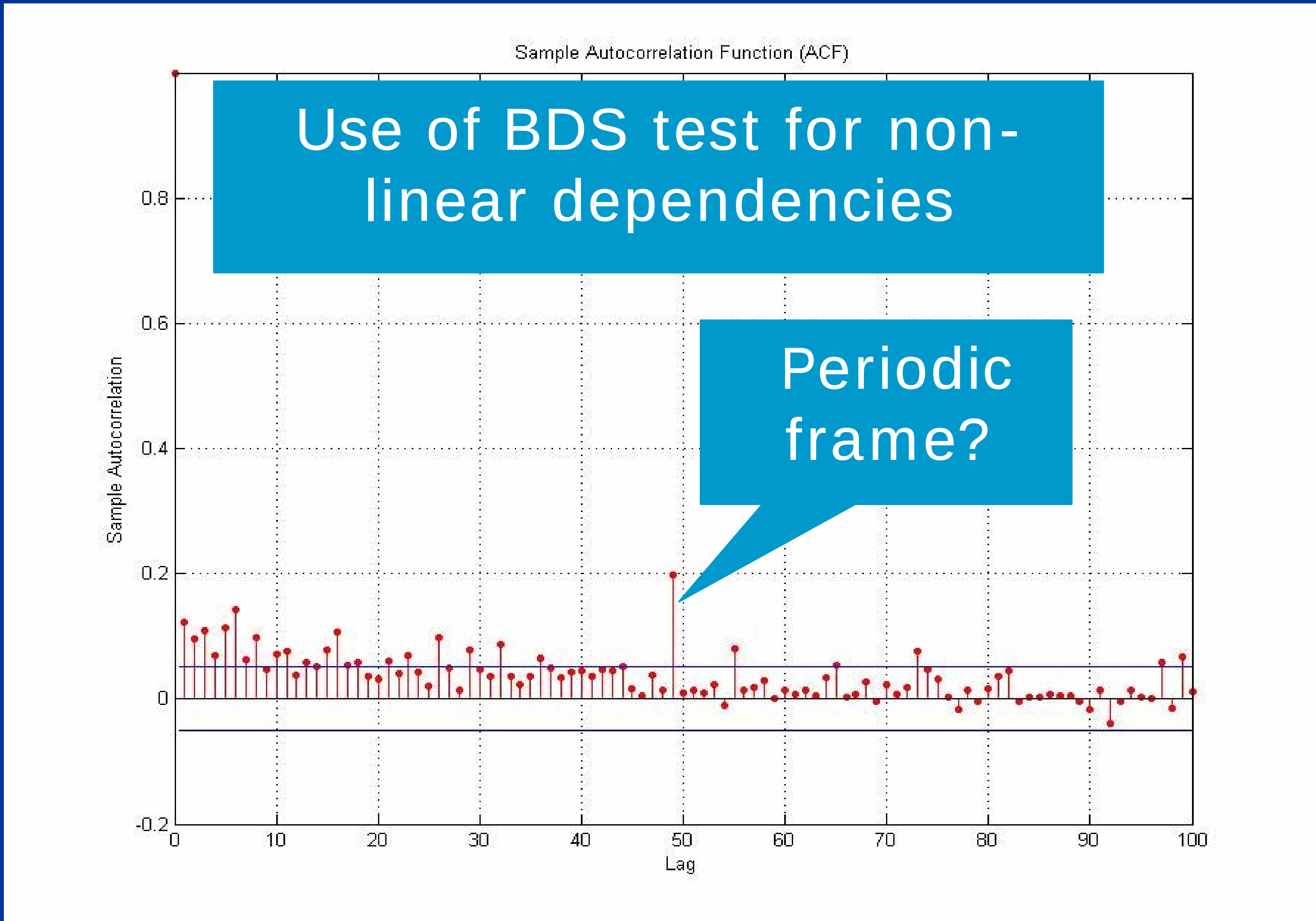


x : [0-1500]
y : [0-25]

x : [0-1500]
y : [0-2.5]

x : [0-250]
y : [0-2]

Question: are interarrival times i.i.d. ?



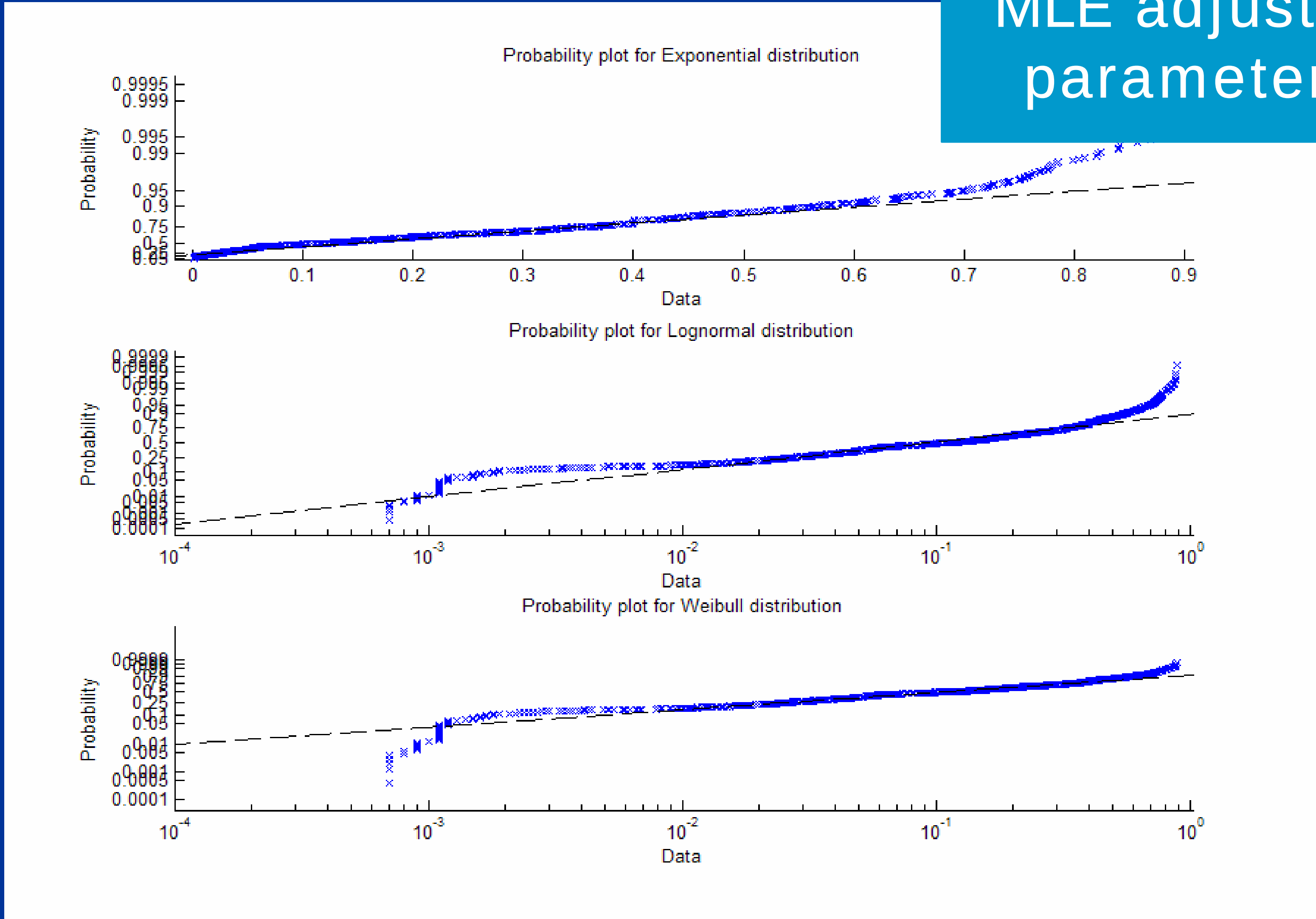
Use of BDS test for non-linear dependencies

Periodic frame?

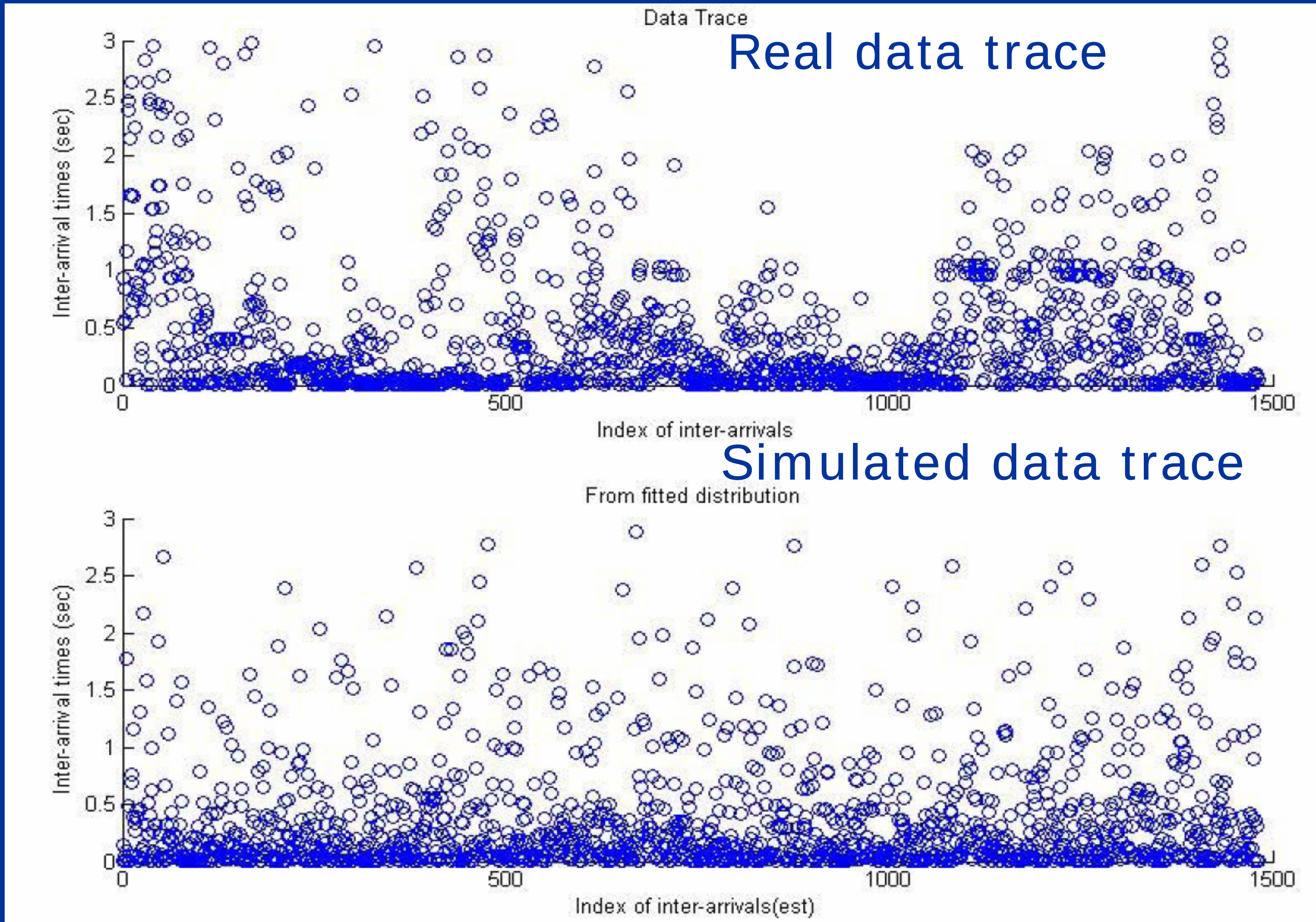
Distribution fitting for aperiodic interarrival : 3 candidates here

MLE adjusted parameters

Kolmo. Smi. and χ2 tests to confirm visual impression



Captured data trace VS random trace generated with MLE-fitted Weibull



Deriving the aperiodic WAF

- S(t) : aperiodic WAF
- X(t) : stochastic process which counts the number of aperiodic frames in time interval t
- "smallest" S(t) such that the probability of X(t) being larger than or equal to S(t) is lower than a threshold α

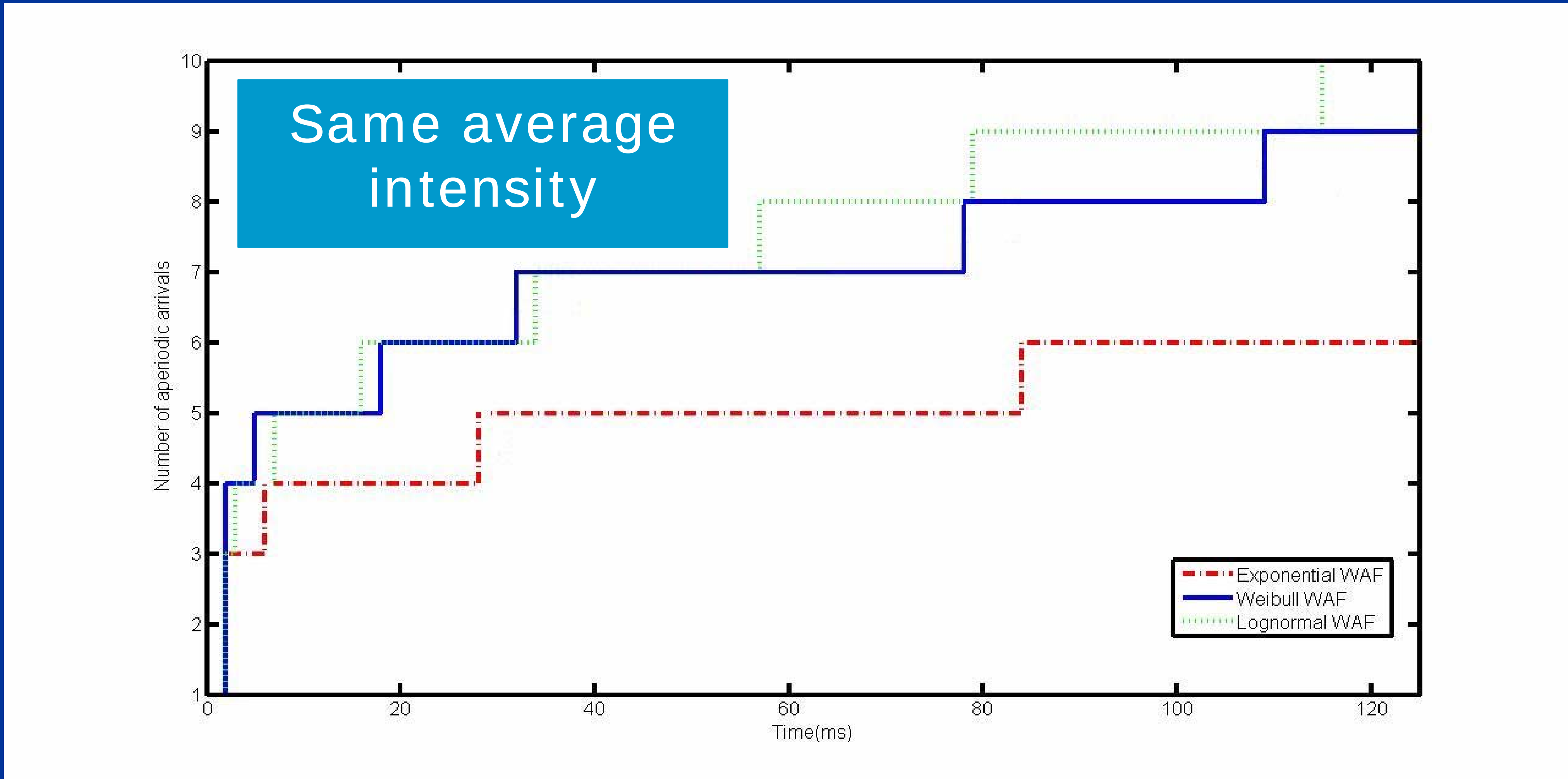
$$\hat{S}(t) = \min\{S(t) \mid \underline{Pr[X(t) \geq S(t)]} \leq \alpha\}$$

By simulation, numerical approximation or analysis (simplest cases such as exp.)

Design choice: e.g., 10^{-9}

INRIA RTaW RealTime-at-Work

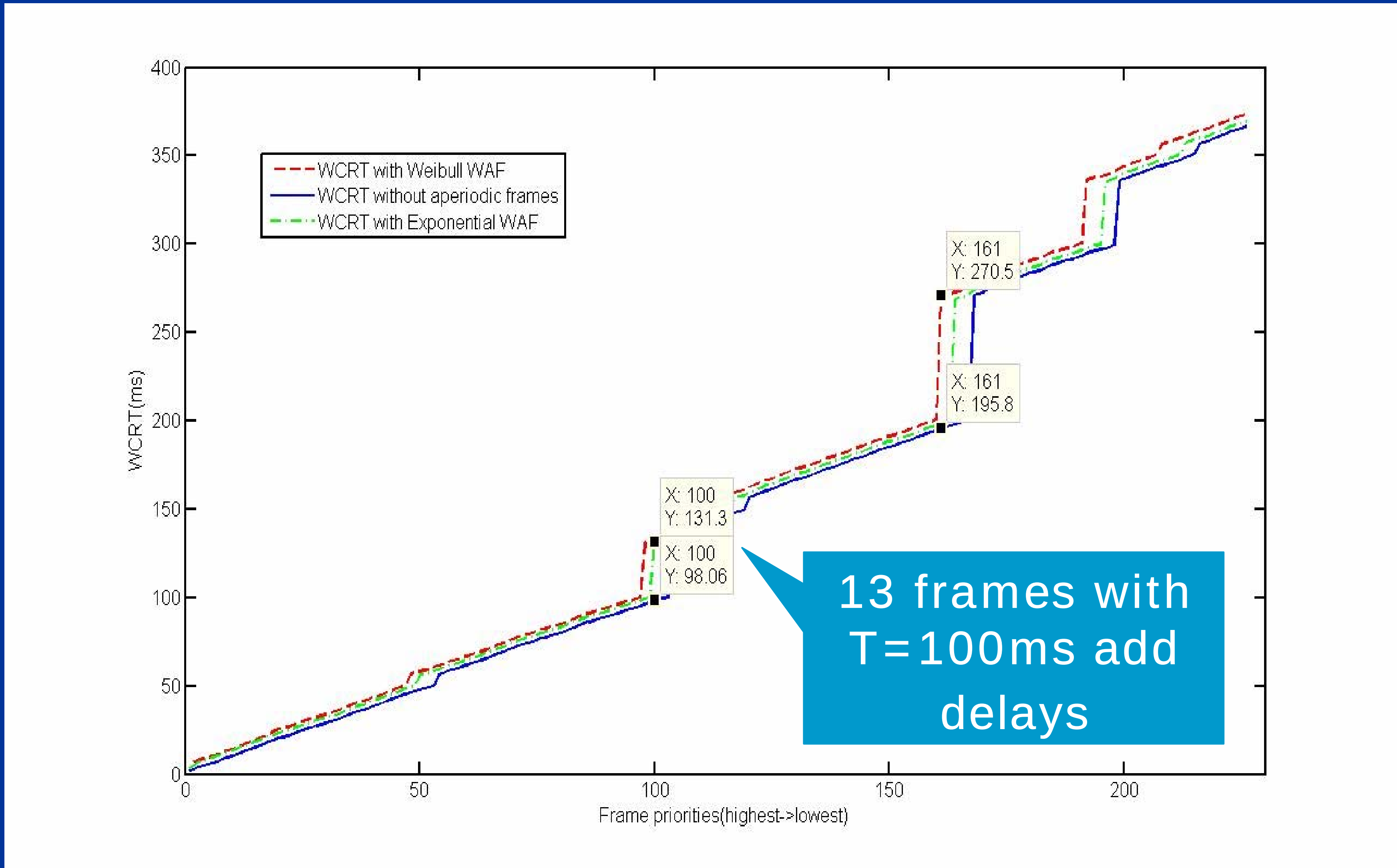
Aperiodic WAF depends on the underlying interarrival distribution



Case-study on a typical body network

- Body network benchmark generated using GPL-licensed Netcarbench
- Characteristics:
  - 125kbps, 16 ECUs, 105 CAN frames with deadlines equal to periods and 1 to 8 bytes of data.
  - Total periodic load is equal to 35.47%
- NETCAR-Analyzer for WCRT computation
- 3% aperiodic traffic
- 7 byte aperiodic frames
- $\alpha = 10^{-4}$

Worst-case response times with/out aperiodic traffic (3%)



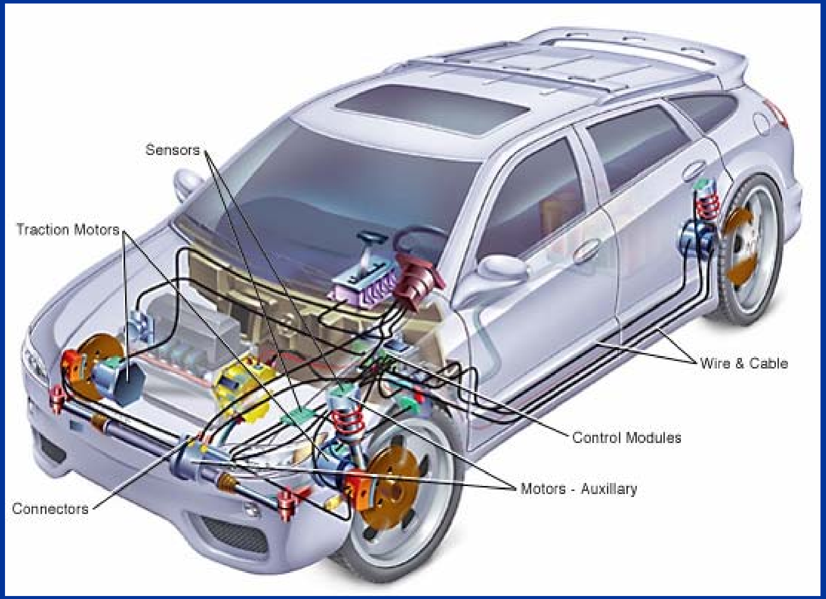
13 frames with T=100ms add delays

Conclusion - part 1

- Chosen dependability requirements are met while pessimism kept to minimum:
  - Practical approach
  - Real data are required
  - Can be extended to the non i.i.d. case (not needed here)
- What is needed now is a system level approach that
  - Can handle arbitrary activation processes
  - goes beyond the i.i.d. case (for dependability)

Part 2 – optimized synthesis, the case of frame scheduling on CAN

(see paper ref[8])

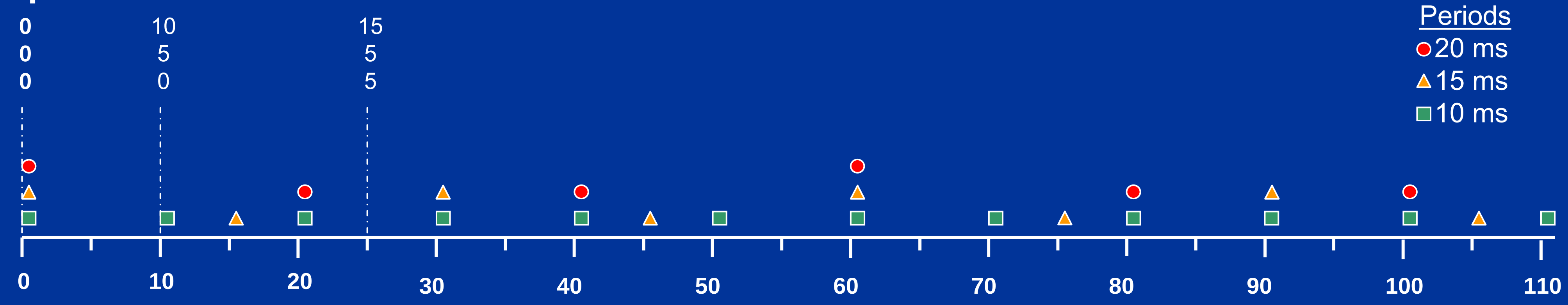


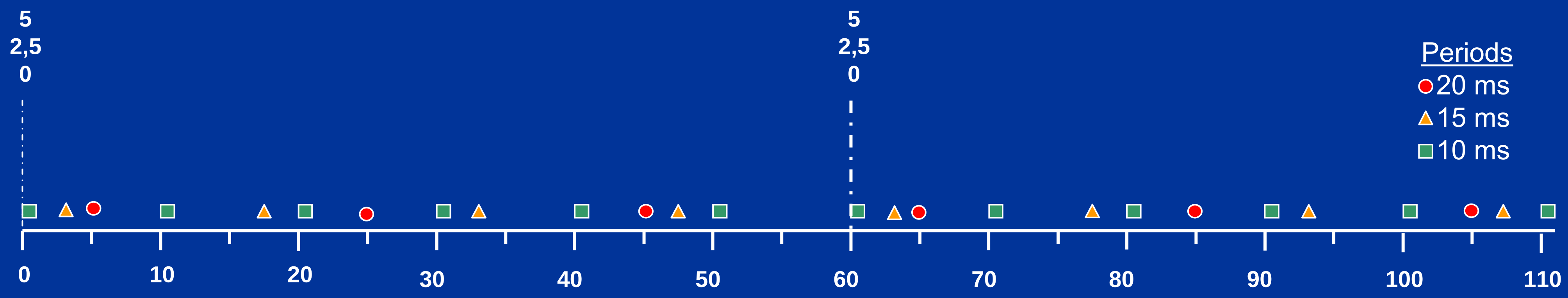
Optimizing the use of resources is becoming an industrial requirement

- Reasons for optimizing (i.e., less hardware):
  - Complexity of the architectures (protocols, wiring, ECUs, gateways, etc)
  - Hardware costs / weight, room, fuel consumption, etc
  - Need for incremental design
  - Industrial risk and time to master new technologies
  - Performances (sometimes):
    - Ex1: A 60% loaded CAN network may be more efficient that two 35% networks interconnected by a gateway
    - Ex2: cost of communication in distributed functions
  - Limits of current technologies (CPU frequency w/o fan),
  - Etc ...

Scheduling frames with offsets ?!

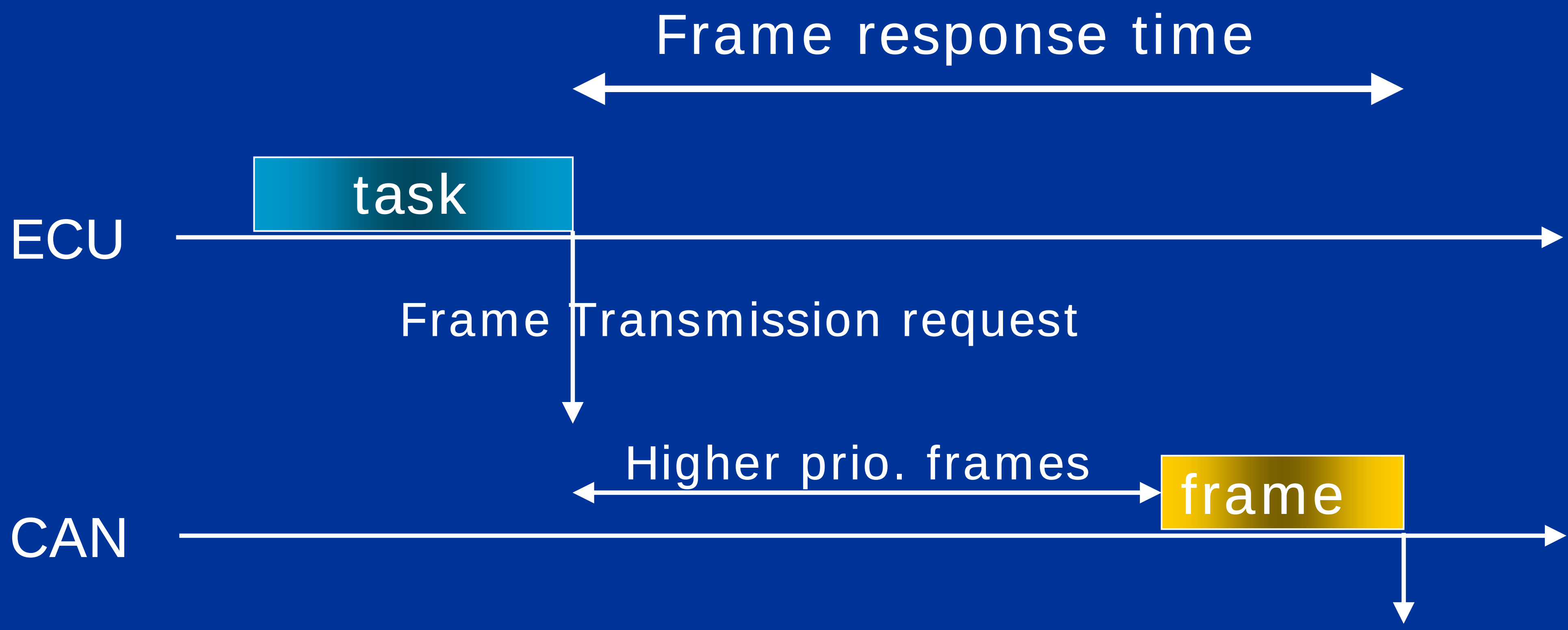
Principle: desynchronize transmissions to avoid load peaks





Algorithms to decide offsets are based on arithmetical properties of the periods and size of the frame

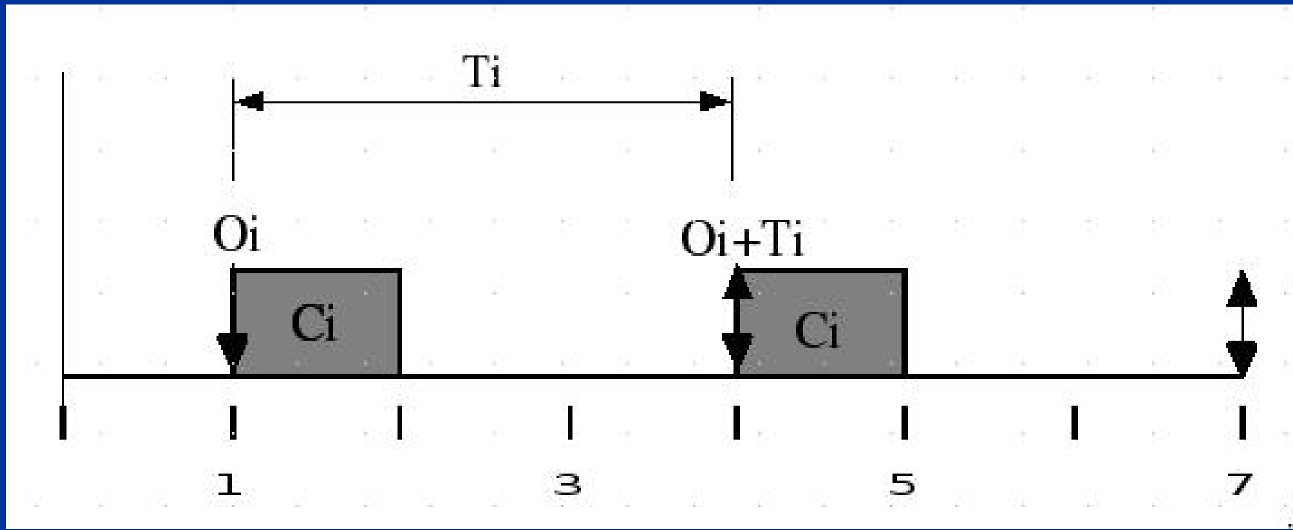
System model (1/2)



- Performance metric: worst-case response time

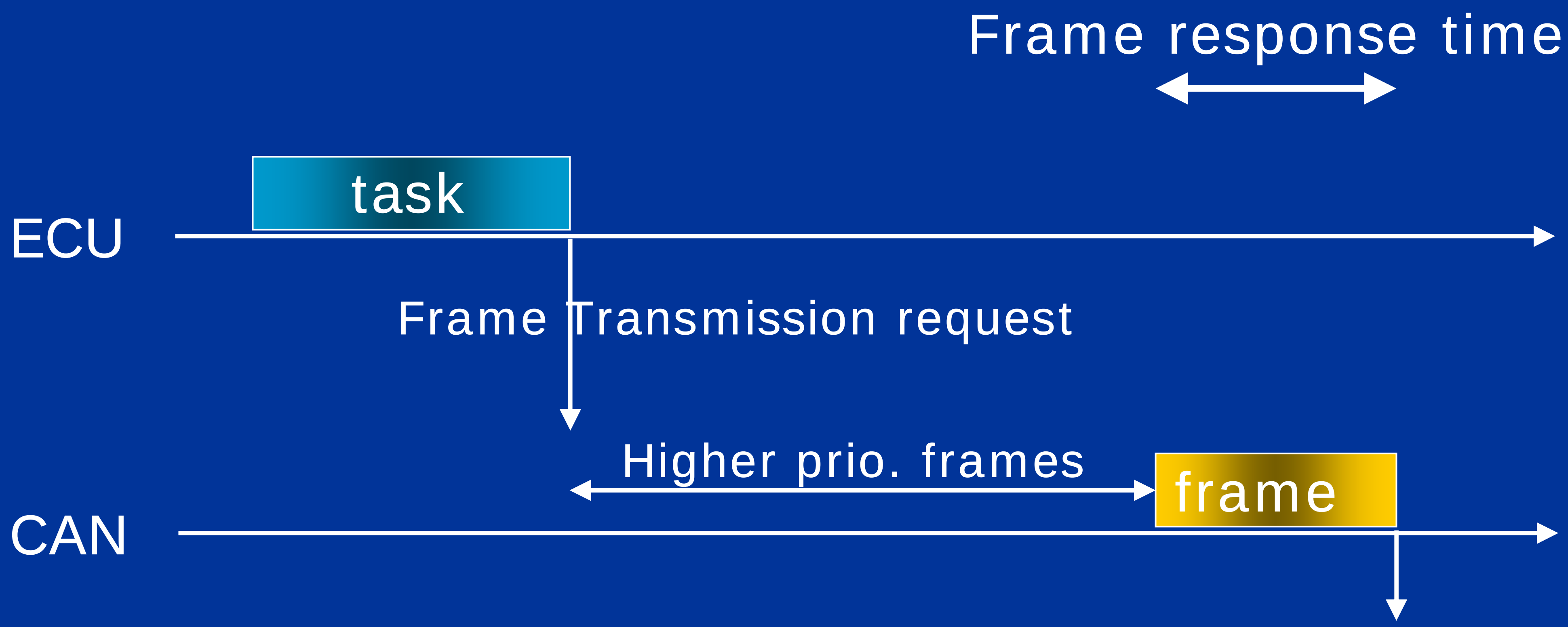
System model (2/2)

- The offset of a message stream is the time at which the transmission request of the first frame is issued



- Complexity: best choosing the offsets is exponential in the task periods → approximate solutions
- Middleware task imposes a certain granularity
- Without ECU synchronisation, offsets are local to ECUs

But task scheduling has to be adapted...



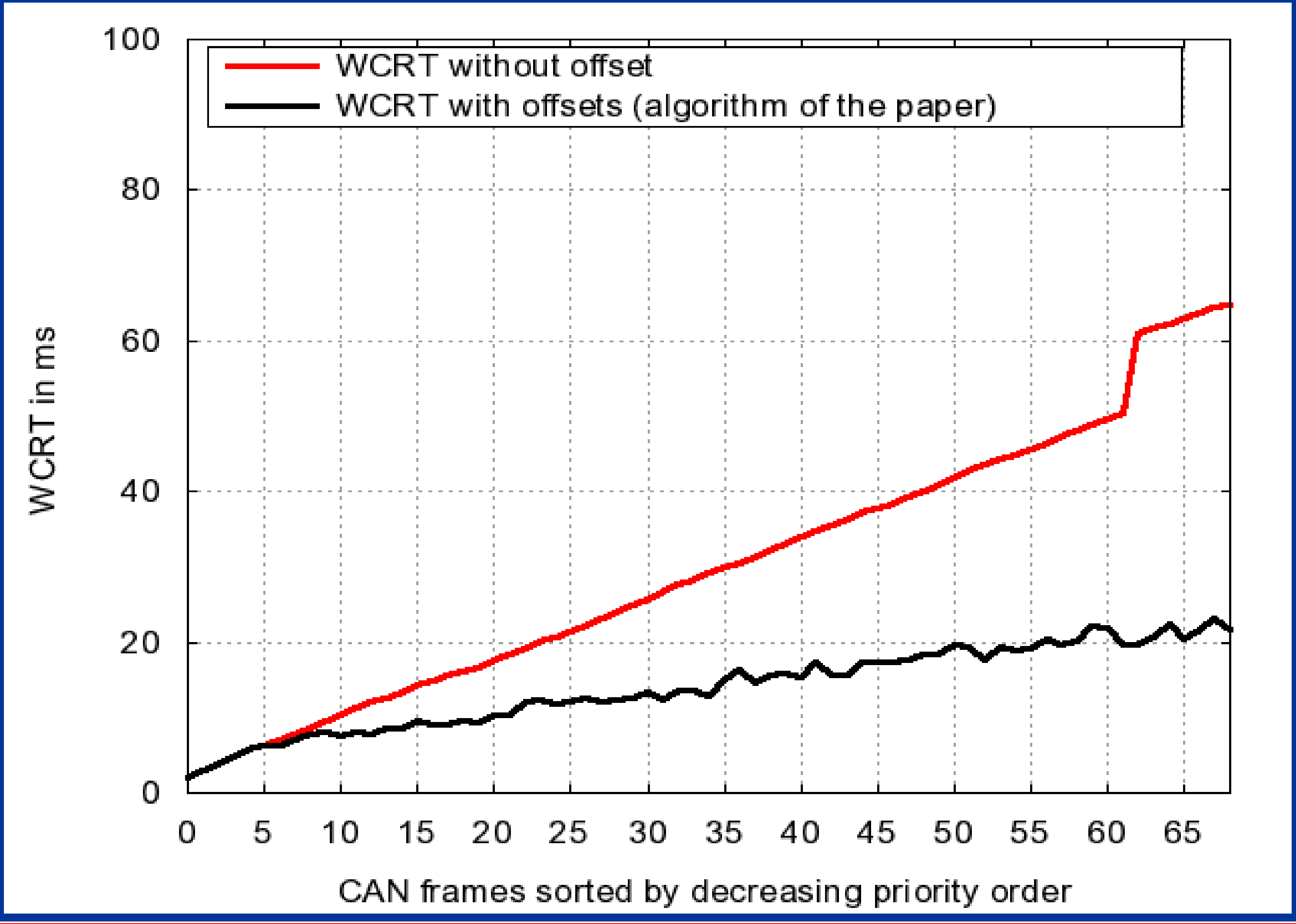
- In addition, avoiding consecutive frame constructions on an ECU allows to reduce latency

Simple offsets Algorithm (1/3)

- Ideas:
  - assign offsets in the order of the transmission frequencies
  - release of the first frame is as far as possible from adjacent frames
  - identify “least loaded interval”
- Ex: $f_1=(T_1=10)$, $f_2=(T_2=20)$, $f_3(T_3=20)$

Time	0	2	4	6	8	10	12	14	16	18
Frame			$f_{1,1}$		$f_{2,1}$			$f_{1,2}$		$f_{3,1}$

Offsets Algorithm applied on a typical body network



65 ms

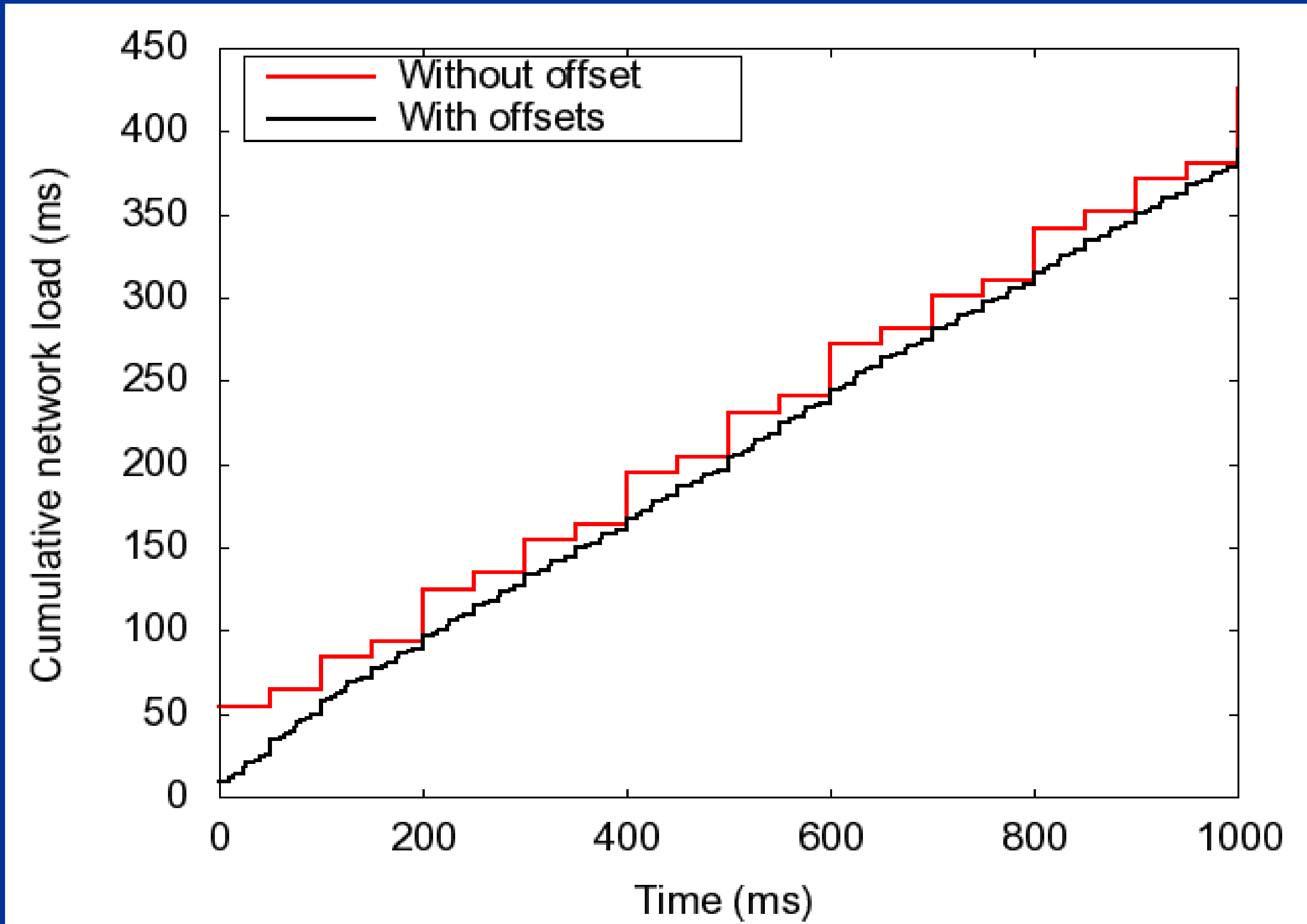
21 ms

Offsets Algorithm – industrial needs

- Low complexity and efficient as is but further improvements possible:
  - add frame(s) / ECU(s) to an existing design
  - user defined criteria : optimize last 10 frames, a specific frame,
  - take priorities on the bus into account
  - optimization algorithms: tabu search, hill climbing, genetic algorithms
  - ...

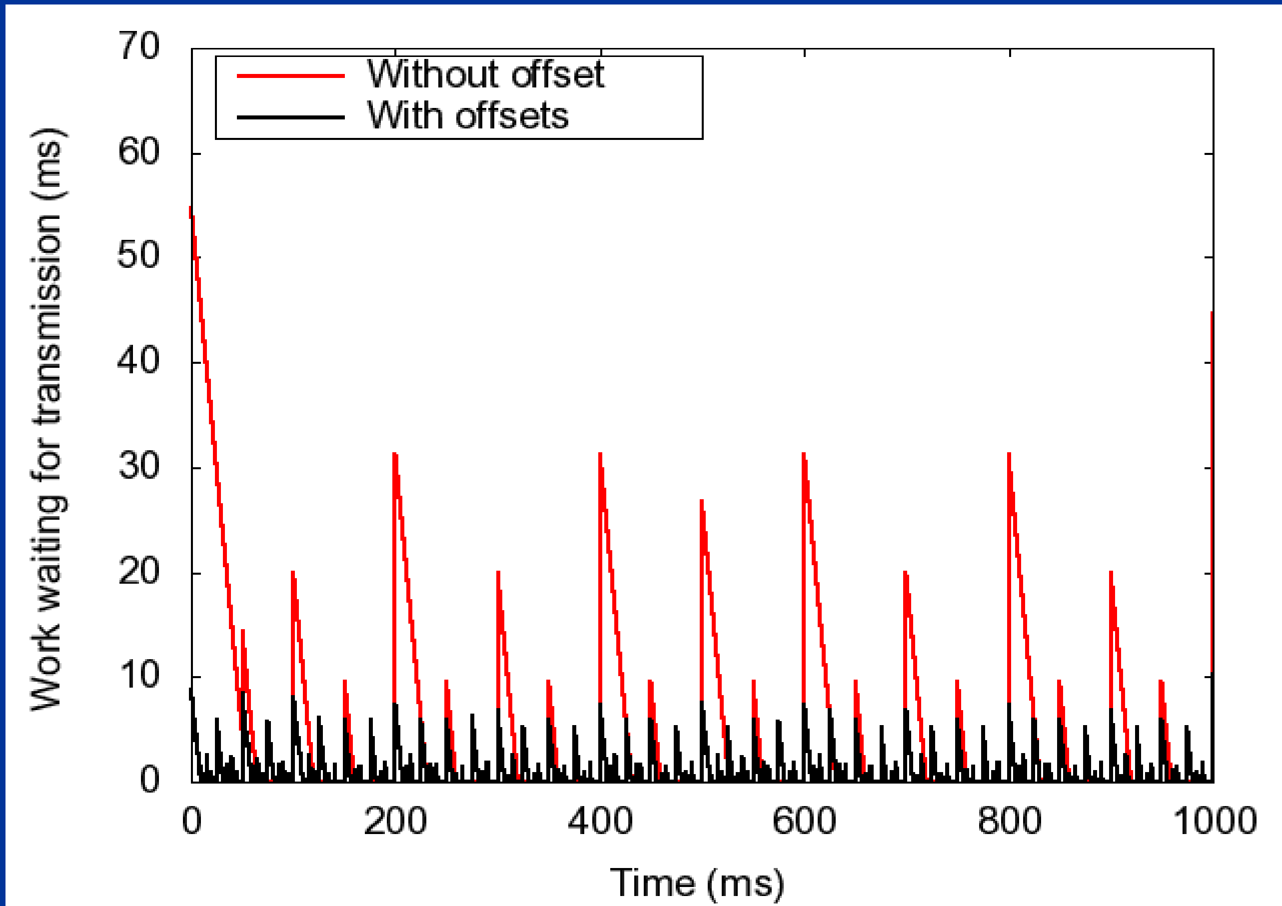
Efficiency of offsets : some insight (1/2)

Work = time to transmit the CAN frames sent by the stations



➢ Almost a straight line, suggests that the algorithm is near-optimal

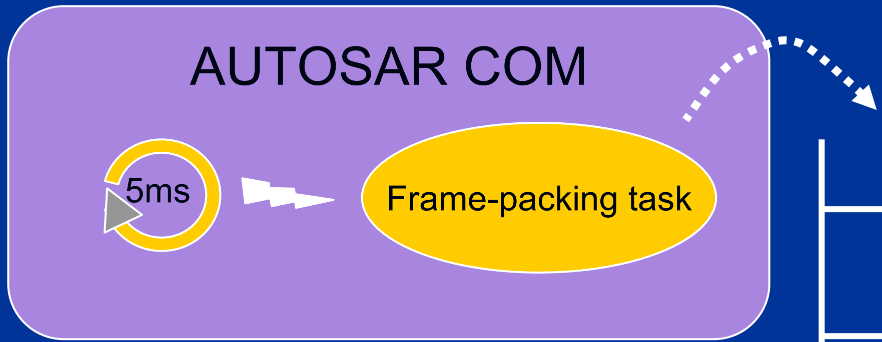
Efficiency of offsets : some insight (2/2)



➢ A larger workload waiting for transmission implies larger response times for the low priority frames ..

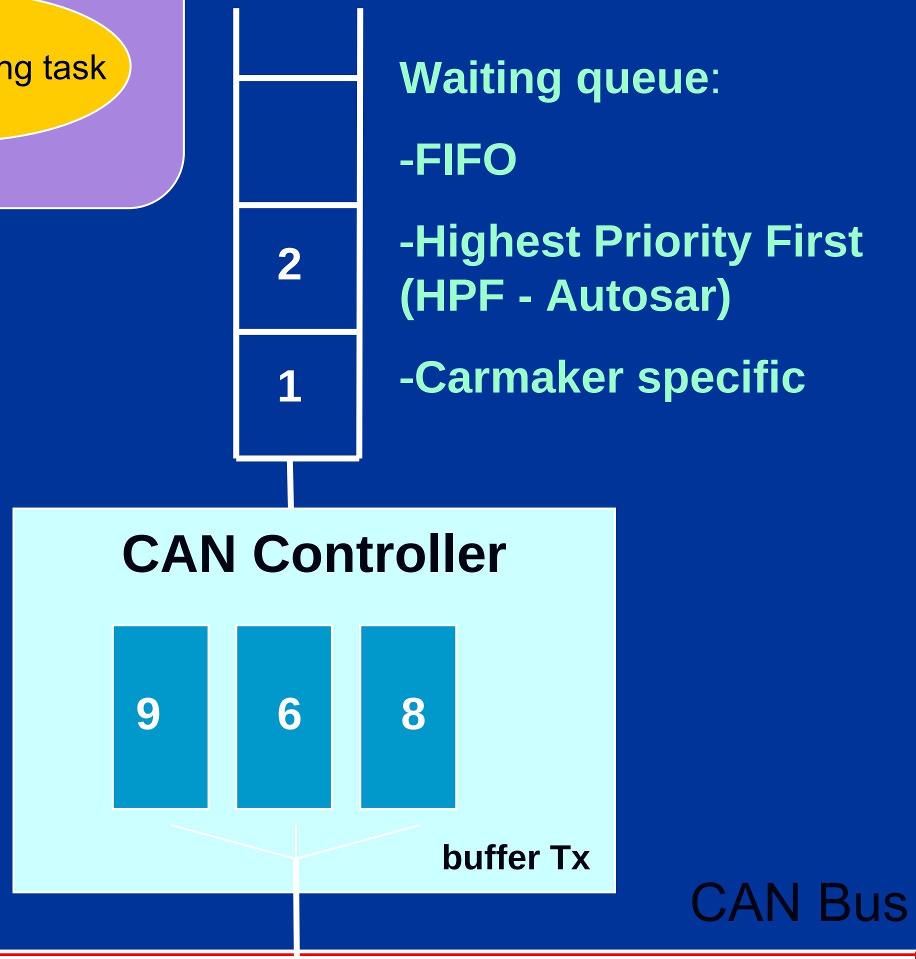
Computing worst-case response times with offsets

Computing frame worst-case response time with offsets



Requirements :

- handle 100+ frames
- very fast execution times
- ≠ waiting queue policy at the microcontroller level
- limited number of transmission buffers



WCRT : State of the art

- Scientific literature:
  - Complexity is exponential
  - No schedulability analysis with offsets in the distributed non-preemptive case
  - Offsets in the preemptive case : not suited for > 10-20 tasks
  - WCRT without offsets: infinite number of Tx buffers and no queue at the microcontroller level
- RTaW software: NETCAR-Analyzer

Performance evaluation :

- Experimental Setup
- WCRT of the frames wrt random offsets and lower bound
- WCRT reduction ratio for chassis and body networks
- Load increase : add new ECUs / add more traffic

Experimental Setup

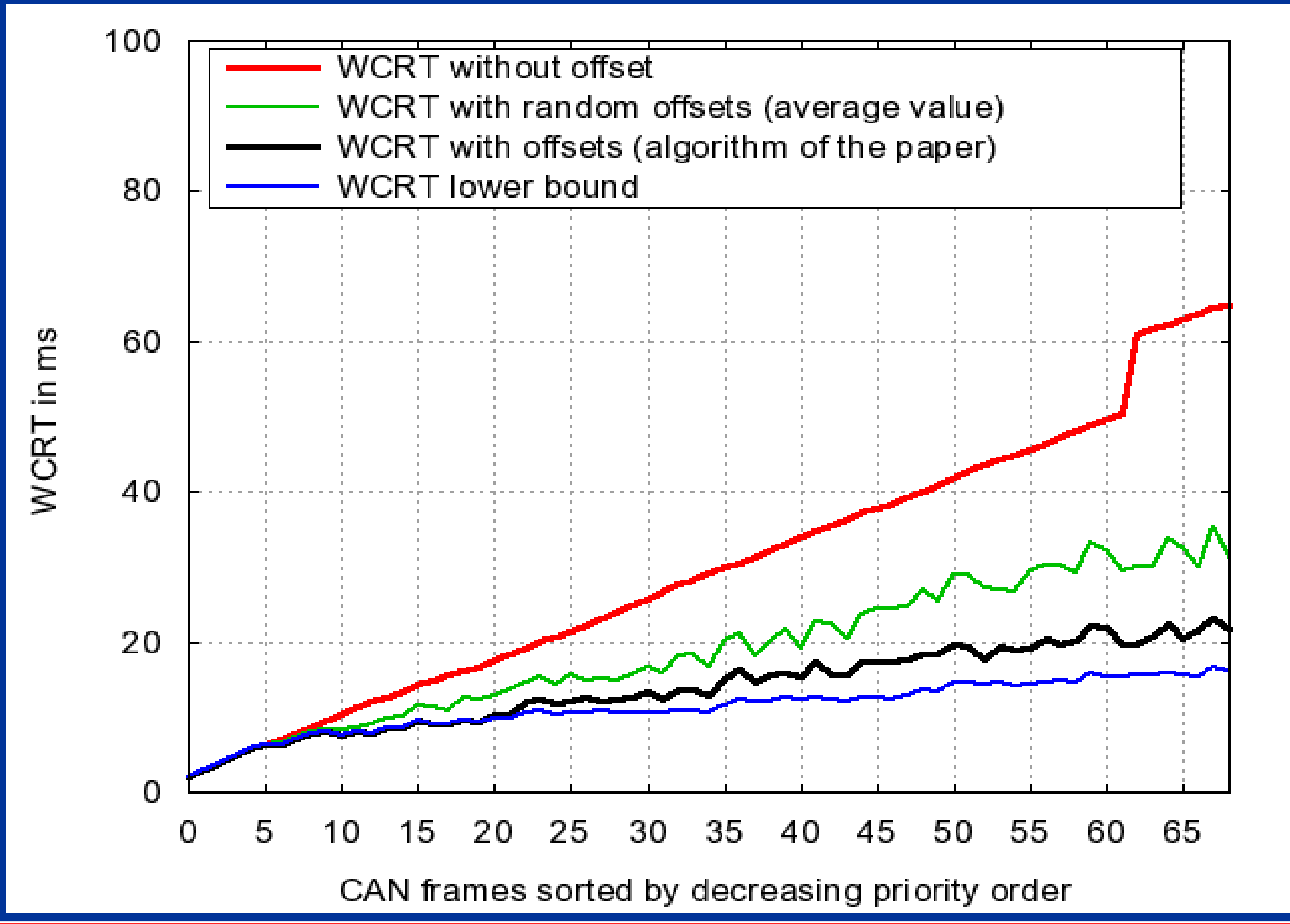
- Body and chassis networks

Network	#ECUs	#Messages	Bandwidth	Frame periods
Body	15-20	70	125Kbit/s	50ms-2s
Chassis	5-15	≈ 60	500Kbit/s	10ms-1s

With / without load concentration: one ECU generates 30% of the load

- Set of frames generated with NETCARBENCH

Offsets in practice : large response time improvements (1/2)



65 ms

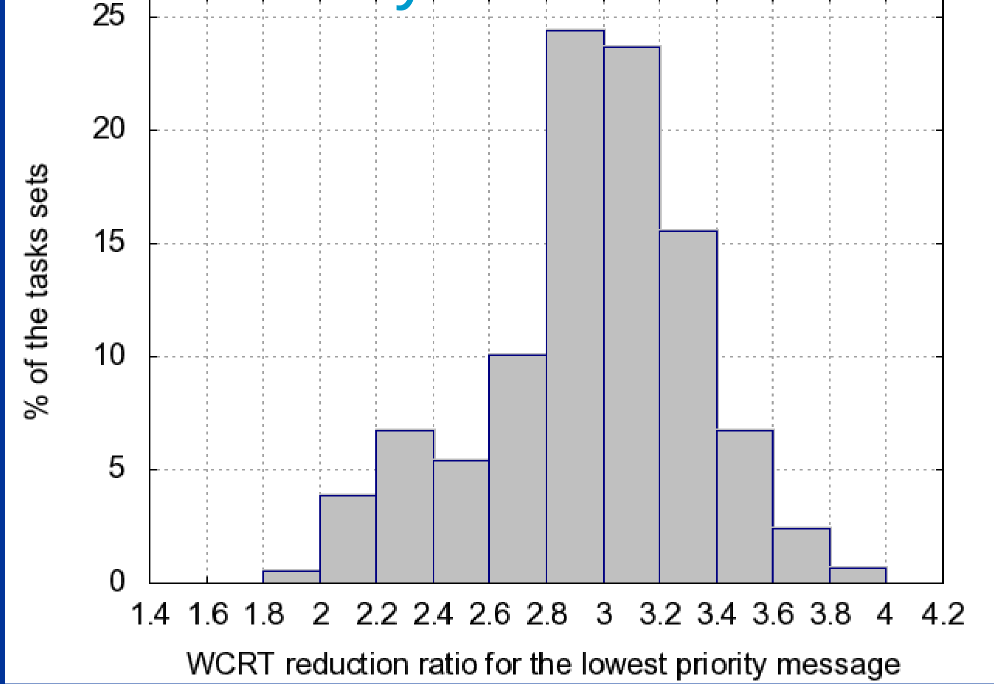
32

21

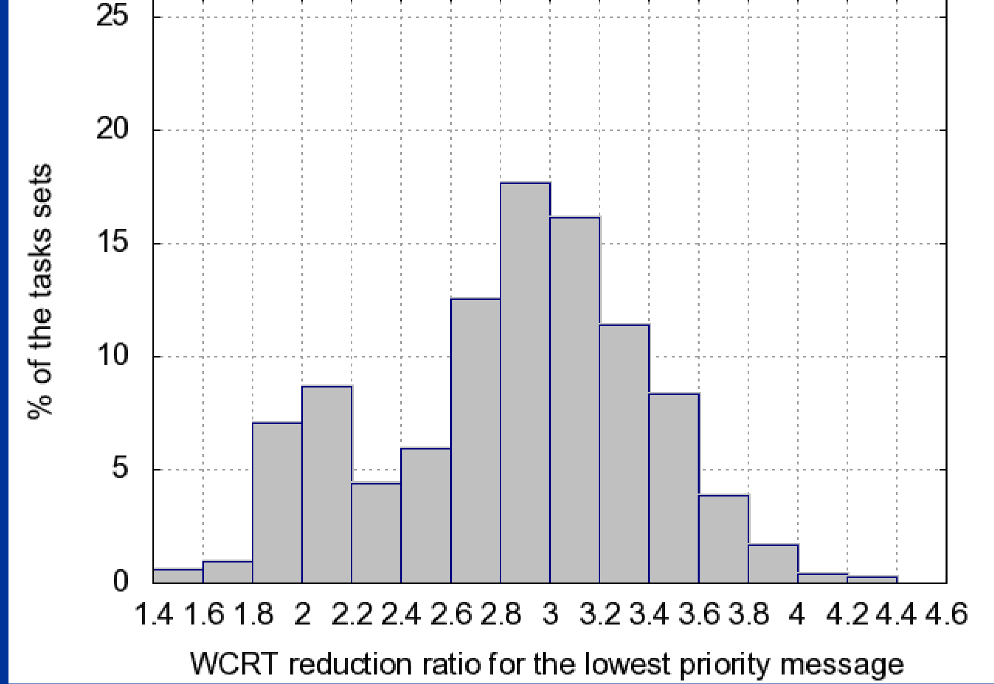
17

WCRT Reduction Ratio

Body Networks



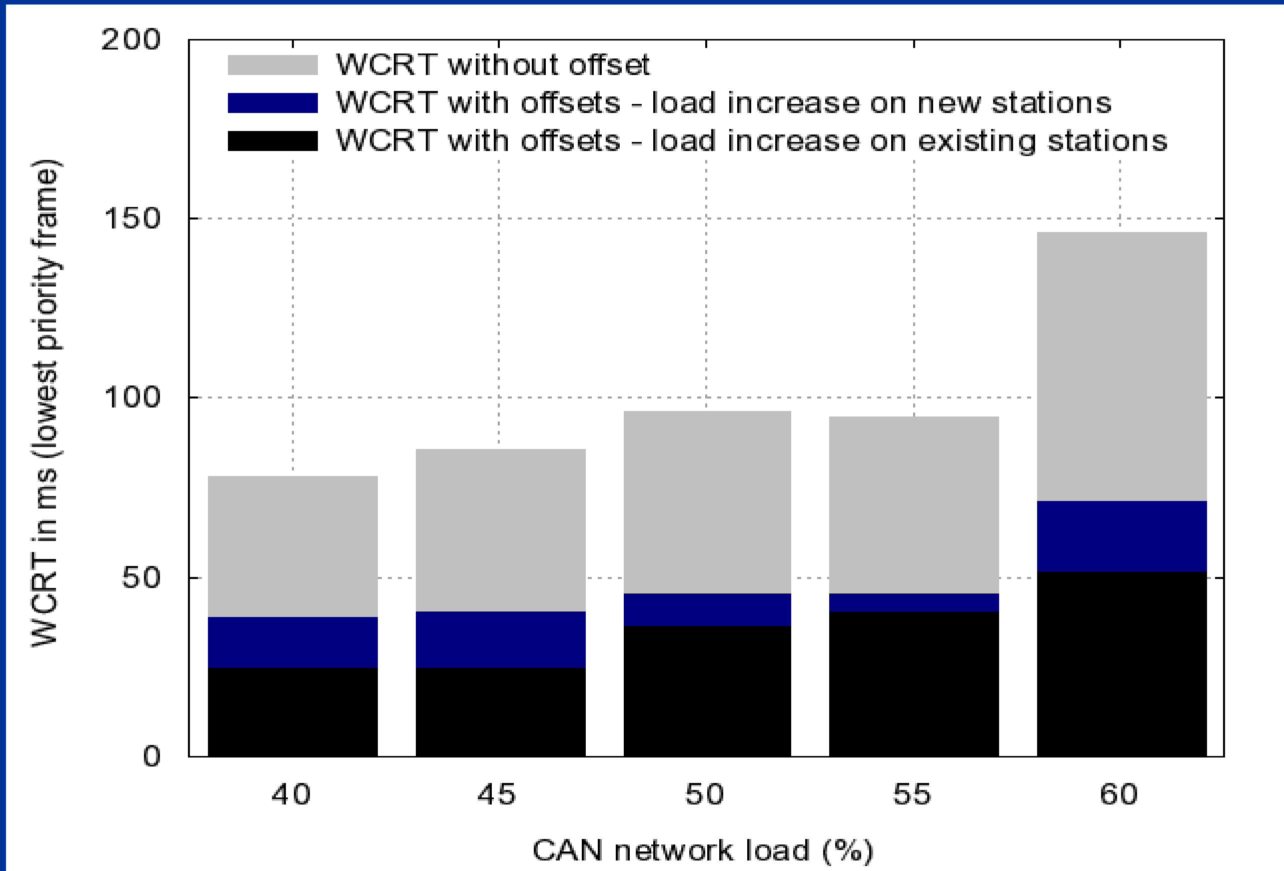
Chassis Networks



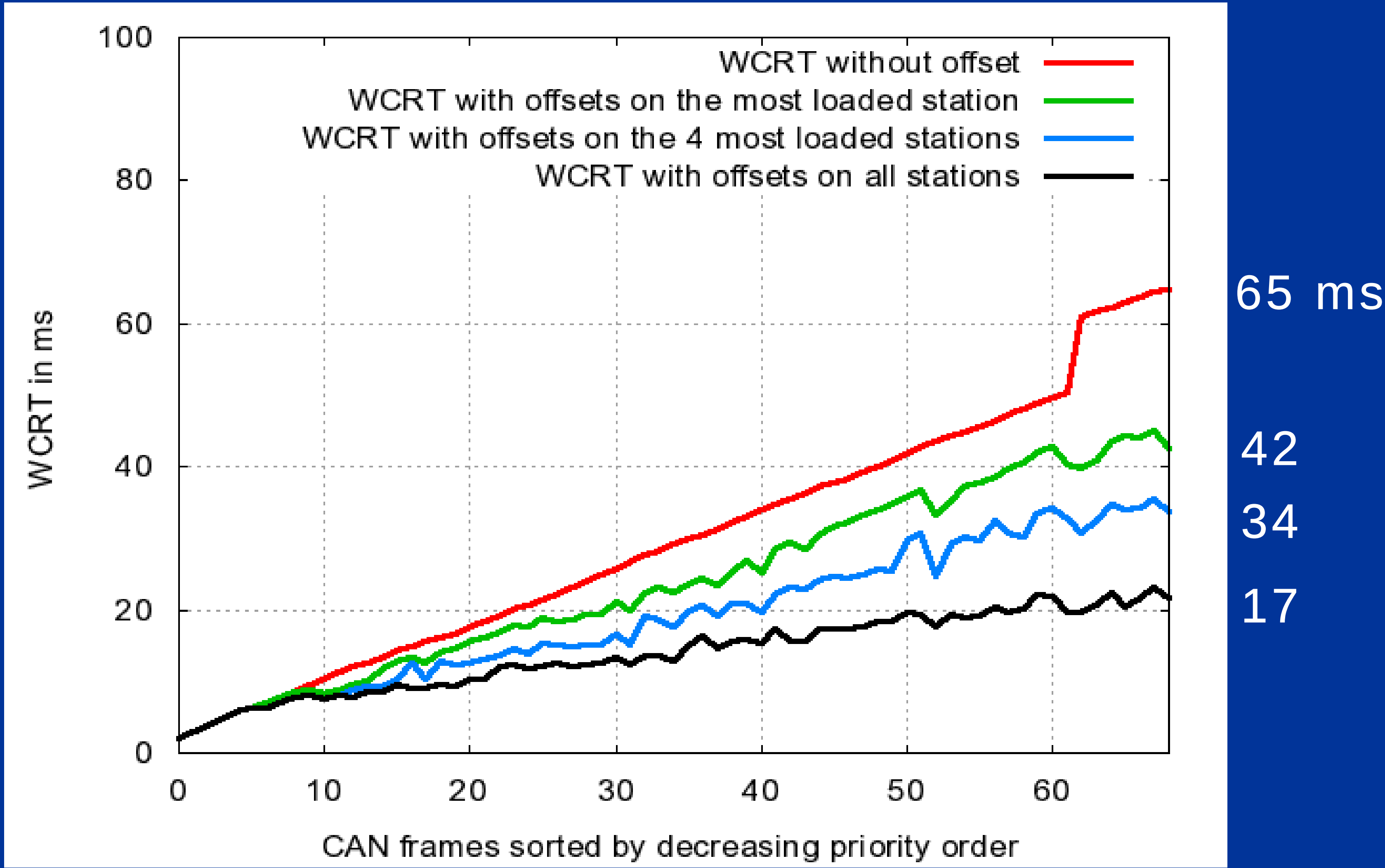
- Results are even better with loaded stations

Offsets allow higher network loads

- Typically: WCRT at 60% with offsets ≈ WCRT at 30% without offsets



Partial offset usage



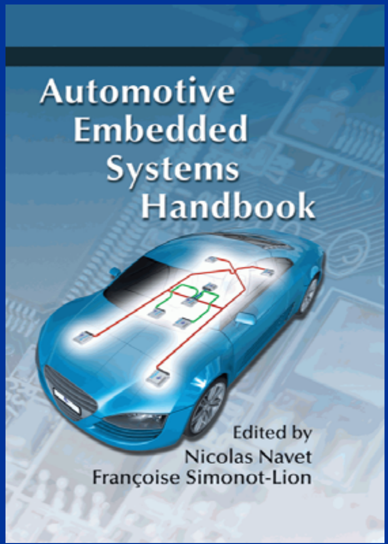
Conclusions on offsets

- Offsets provide an cost-effective short-term solution to postpone multiple CANs and FlexRay
- Tradeoff between Event and Time Triggered

- Further large improvements are possible by synchronizing the ECUs …

References

References (1/2)



Automotive Embedded Systems - General

[1] N. Navet, F. Simonot-Lion, editors, The Automotive Embedded Systems Handbook, Industrial Information Technology series, CRC Press / Taylor and Francis, ISBN 978-0849380266, December 2008.

[2] P. Wallin, Axelsson, A Case Study of Issues Related to Automotive E/E System Architecture Development, IEEE International Conference and Workshop on the Engineering of Computer Based Systems, 2008.

[3] H. Hansson, M. Nolin, T. Nolte, Beating the Automotive Code Complexity Challenge: Components, Models and Tools, National Workshop on High-Confidence Automotive Cyber-Physical Systems, 2008.

Dependability / probabilistic framework

[4] N. Navet, H. Perrault, “Mécanismes de protection dans AUTOSAR OS”, RTS Embedded Systems 2009 (RTS’09), Paris, April 2009.

[5] D. Khan, N. Navet, B. Bavoux, J. Migge, “Aperiodic Traffic in Response Time Analyses with Adjustable Safety Level“, IEEE ETFA2009, Mallorca, Spain, September 2009.

[6] N. Navet, Y-Q. Song, F. Simonot, “Worst-Case Deadline Failure Probability in Real-Time Applications Distributed over CAN (Controller Area Network)“, Journal of Systems Architecture, Elsevier Science, vol. 46, n°7, 2000.

[7] A. Burns, G. Bernat, I. Broster, A probabilistic framework for schedulability analysi, Third International Conference on Embedded Software (EMSOFT 2003), 2003.

Scheduling frame with offsets on CAN

[8] M. Grenier, L. Havet, N. Navet, “Pushing the limits of CAN - Scheduling frames with offsets provides a major performance boost“, Proc. of the 4th European Congress Embedded Real Time Software (ERTS 2008), Toulouse, France, 2008.

Questions / feedback ?

Please get in touch at:
nicolas.navet@realtimeatwork.com

http://www.realtimeatwork.com