Faculty of Sciences, Technology and Communication

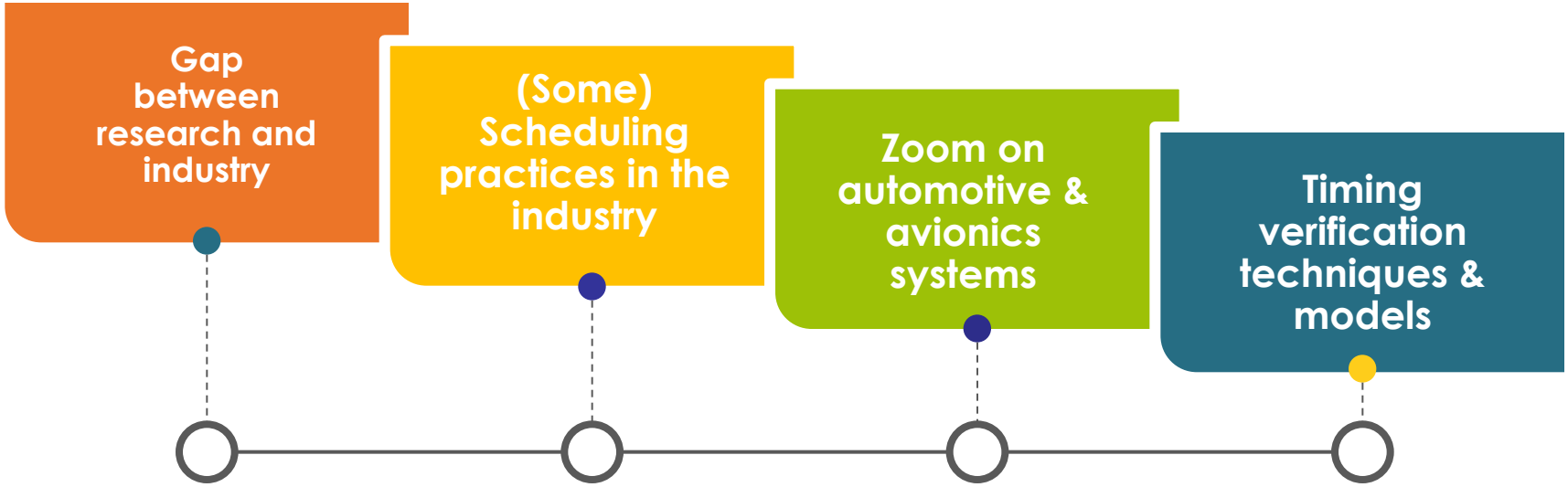
Industrial practices of real-time scheduling

Nicolas Navet
nicolas.navet@uni.lu

Colloquium “1972-2012: 40 years of research in real-time scheduling”, journées scientifiques de l’Université de Nantes, June 8, 2012.

uni.lu UNIVERSITÉ DU LUXEMBOURG

Outline


Gap between research and industry
(Some) Scheduling practices in the industry
Zoom on automotive & avionics systems
Timing verification techniques & models

There is a gap between research results and industrial practices

- ✓ “Support for any real-time scheduling algorithm or locking protocol developed with the last twenty years is practically non-existent in both commercial and open-source RTOS” - J. Andersson & Mollison in 2011 [1]
- ✓ Not the case in other fields as comp. architecture, graphics
- ✓ True for scheduling policies, resource sharing protocols, task activation(!), schedulability analysis - mono & multiprocessor, w/wo low-power constraint
- ✓ Time needed to actually apply RMA, TTA, CAN analysis …

Many reasons for that…

- ✓ End-users do not acknowledge they can benefit from state-of-the-art scheduling techniques
- ✓ Too much effort wrt short term benefits: learning theory & changing practices
- ✓ Research results not made easy to understand
- ✓ Models studied do not suit the needs, e.g.:
  - ✓ Task models: e.g, multiprocessor systems, I/O access, etc
  - ✓ Traffic models for networks: need for segmented message, aperiodic traffic, mixed transmission model, etc
  - ✓ Communication stack models : FIFO waiting queues, limited number of transmission buffers, delays in refilling buffers, limits of drivers, etc
- ✓ Tool support is weak or expensive, integration weak with OS and compilers

Specific reason: WCET (over)estimation

- ✓ Scheduling results (mainly) relies on WCET assumptions
- ✓ WCET estimations are conservative (typically +30%) with today's HW, how to reach >75% CPU load level ?

- ✓ (at least) 2 ways out of that:
  - More analyzable hardware
  - Scheduling framework relying on statistical measurements: both methodology and techniques needed

In some systems, gap is or should become narrower

- ✓ Systems subject to certification : e.g. AFDX networking
- ✓ When resource usage optimization is an industrial requirement: e.g. automotive Electronic Control Units & buses.
- ✓ Technology requires state-of-the-art techniques: many-core systems, 3D chips, low-power
- ✓ Model-Driven Development: hide the complexity from the users

(SOME) SCHEDULING PRACTICES IN INDUSTRY

Many RTS are simple enough to not need an OS (1/2)

```
for ( ;; ) {
if ( packet_received ) { // set by communication controller
  Process_data();
  Packet_received = 0; }
}
```

Polled loop

```
for ( ;; ) {
  task_1(); // functions are tasks here
  task_2(); // tasks can communicate through global variables
  …
  task_n();
  task_2();
}
```

static cyclic scheduling within the main function

Many RTS are simple enough to not need an OS (2/2)

```
void main() {
   init();
   while ( TRUE ) ; // wait for some interrupt to occur
}
void intr_1() // interrupt handler (IH) will execute task_1
{ save( context );
   task_1();
   restore( context );
}
void intr_2()
{ save( context );
   task_2();
   restore( context );
}
```

Interrupt driven systems

Types of scheduling

1. Time-triggered / static cyclic scheduling: Arinc653, TTP, FlexRay (static seg.)
2. Processor sharing: RR, GPS, WFQ
3. Priority driven:
   1. fixed priority scheduling: FPP, CAN
   2. dynamic priority scheduling: EDF

✓ Partitioned versus migrating algorithms in multiprocessor systems

Complex systems tend to use multi-layered / hierarchical scheduling solutions often static-cyclic + static priority

From federated to integrated architectures: complexity moves from HW to software platforms

- ✓ Automotive in the 90s: one function per ECU
- ✓ Avionics before : federated architectures with independent units hosting one function each
- ✓ Not sustainable with the increasing # of functions: cables (up to 2km in cars, >100km in aircrafts), # nodes, overall complexity

Today: Independent functions / multi-source software running on standardized execution platforms: e.g. Arinc653, Autosar

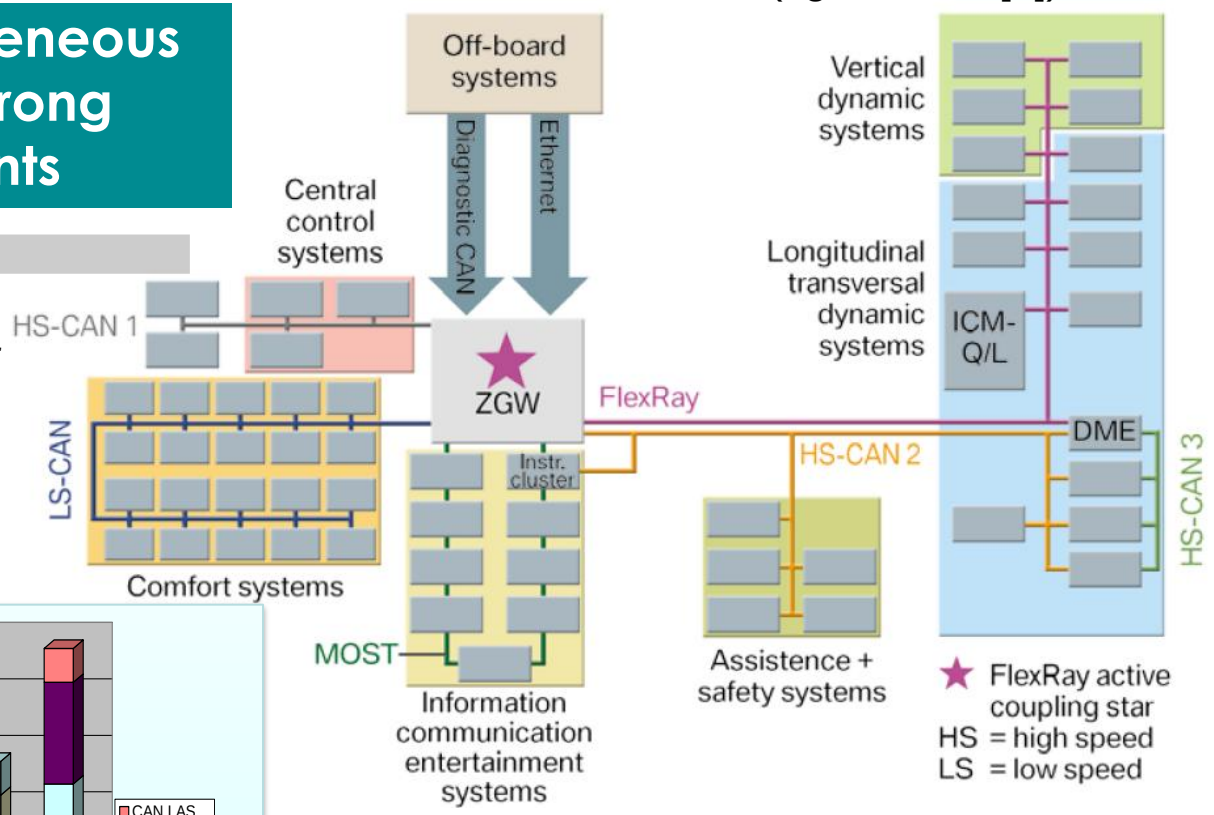
Upcoming: powerful multi-processor stations interconnected by high-speed backbones

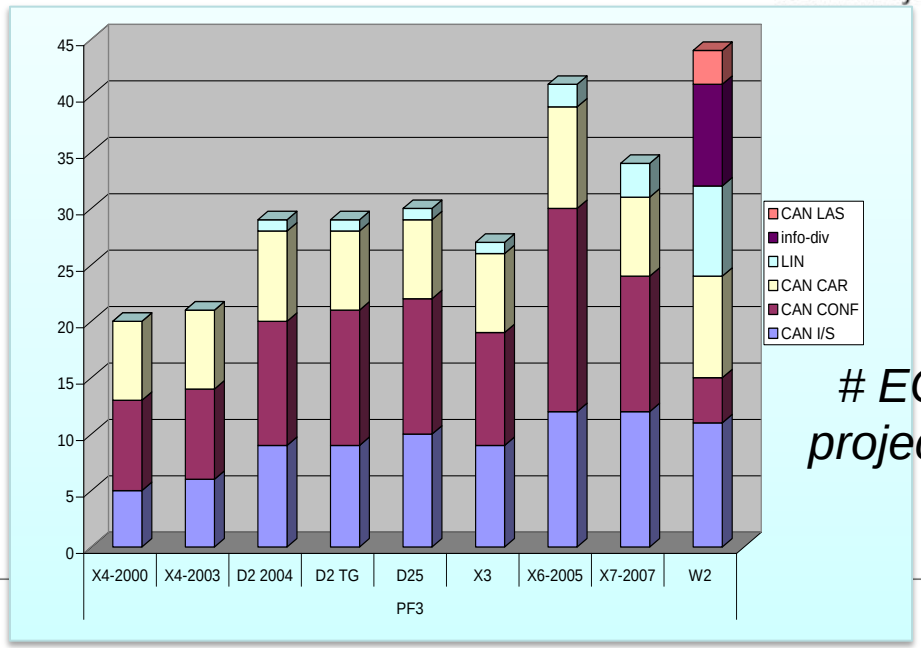
Automotive systems

Complex and heterogeneous architectures with strong real-time constraints

70 ECUs, 2500 signals, >6 comm. protocols, number of variants, etc

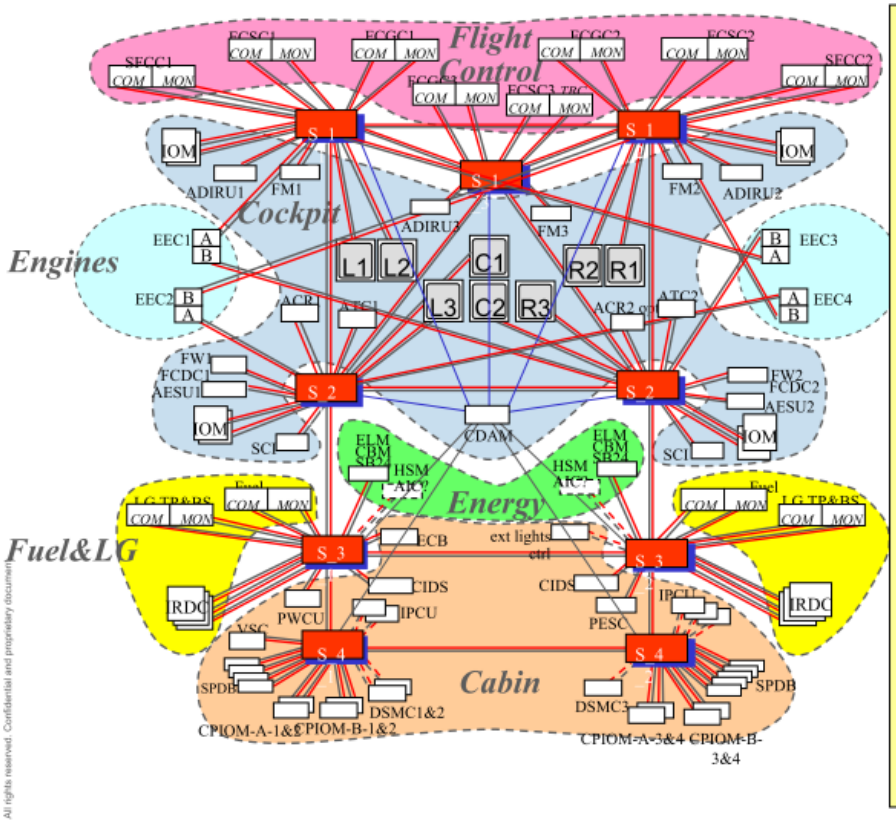
BMW 7 series (figure from [2])





 # ECUs and buses in some PSA projects between 2000 and 2010 [3]

Avionics systems [10,11,14]



AFDX Network:

- 100 Mbits
- Redundant Network (A&B) with independent alimentation
- AFDX switches = 2 x 8
- NB of ports (connections) possible on each switch (20-24)
- MTBF of the switch is very high (100 000 hours expected)
- Up 80 AFDX subscriber

A380 AFDX architecture
Figure from [11]

Realistic AFDX network from Thales [10]

104 end-systems	**8x2 routers**	4 prio levels	974 data flows
6501 latency constraints	**Constraints: from 1 to 30ms**	Period: from 2 to 60ms	

Research issues for the RT community

- End-to-end scheduling with heterogeneous resources
- Hierarchical scheduling, esp. in multicore systems
- Predictable HW multicore platforms
- Mixed-criticality scheduling
- Incremental validation / certification

Typical automotive scheduling setup [13]

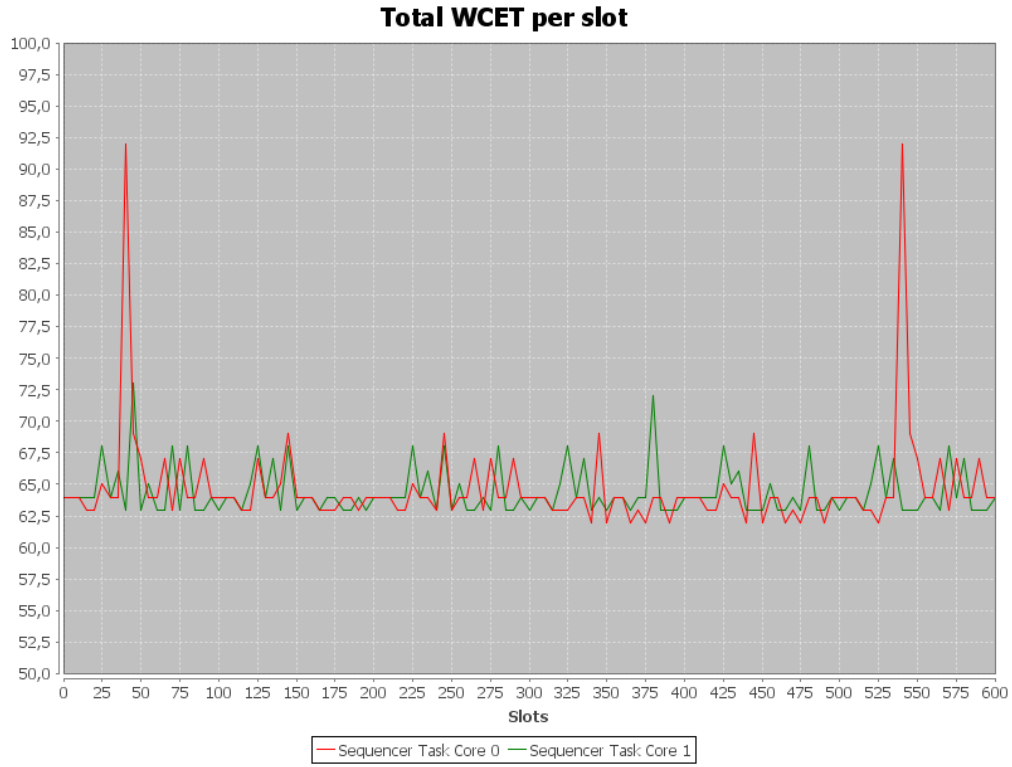
Goal: schedule hundreds of software modules (“runnables”) on a (multicore) ECU so as to minimize peaks of load

Two sub-problems

- **Partitioning :** allocating each runnable to a core
- **Build schedule table:** schedule the execution of the runnables on each core

Main objective

- Feasibility
- Avoid load peaks
  - Criteria: max load

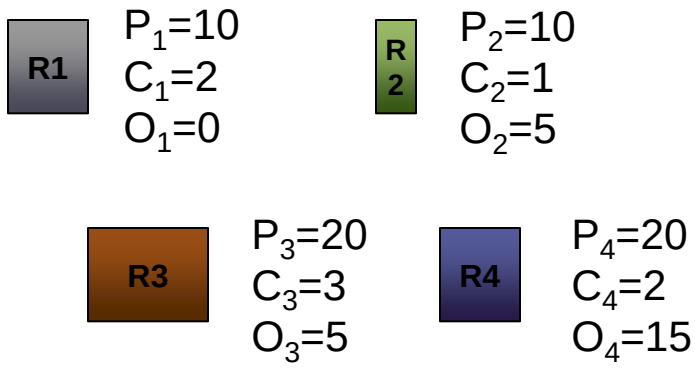


Example schedule: CPU load over time on a dual-core processor

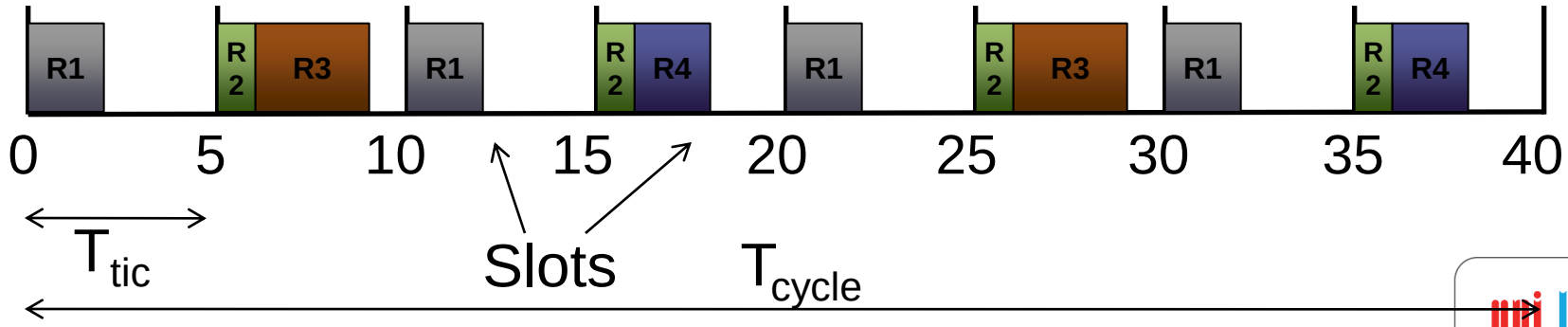
Scheduling model: runnables, tics, slots, offsets and schedule cycles

Runnables

- **P**eriod : runnable execution must be strictly periodic
- W**C**ET: Worst-Case Execution Time (in ms)
- Initial **O**ffset: start time of the slot where the first runnable instance is executed
- *Core allocation constraint [optional]*
- *Colocation constraint [optional]*



Sequencer task



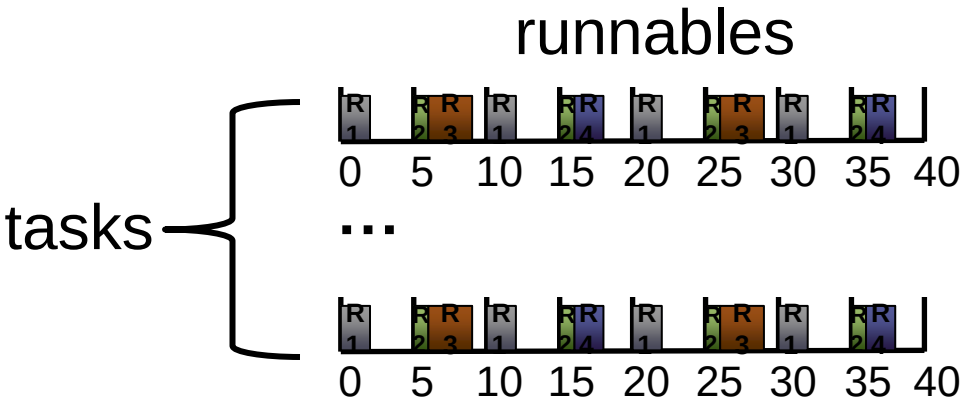
Two-level scheduling

1. **Static-cyclic for runnables within tasks**
2. **Priority driven (static) among tasks**

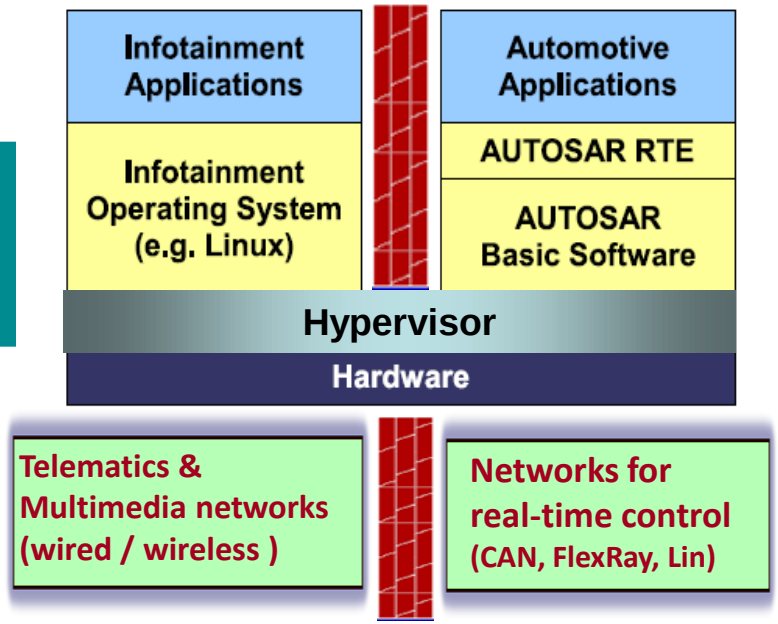
Possible additional requirements:

- ✓ Precedence constraints between runnables
- ✓ Several sequencer tasks :
  - memory protection at the task level
  - not all runnables require the same priority
- ✓ Incremental scheduling wo changing execution orders
- ✓ Sequencer tasks may be or not synchronized / driven by different clocks (time vs RPM)
- ✓ Synchronization between task and message scheduling might be needed

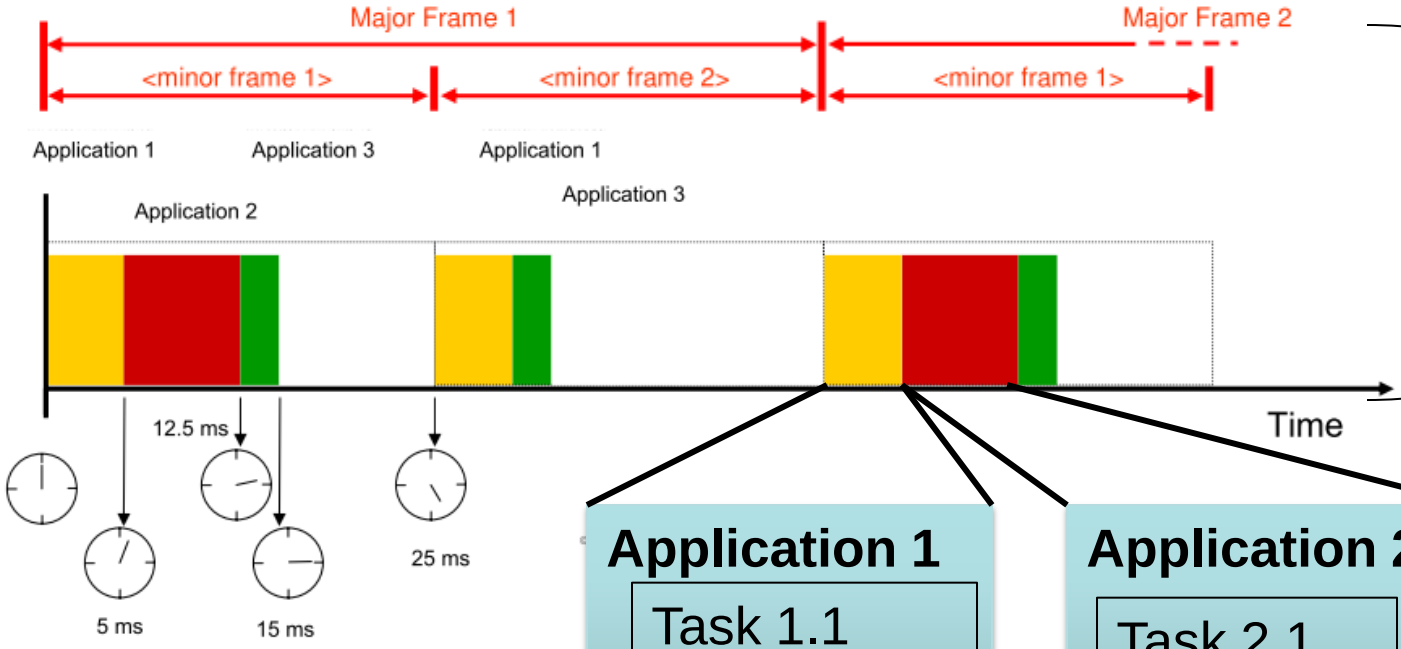
Three level scheduling with an hypervisor! [3,4]



Third level : scheduling of the Virtual Machines and hypervisor activities (e.g. drivers)

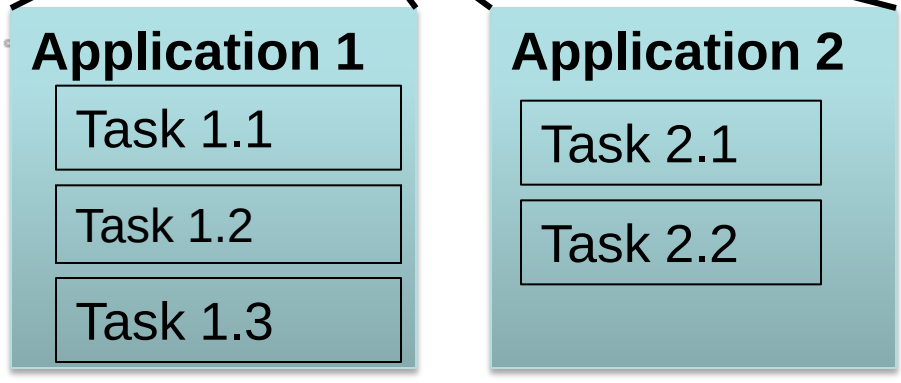


Hierarchical scheduling in avionics



static cyclic
non preemptive

figure from [11] and [14]



priority
driven
preemptive

TIMING VERIFICATION: TECHNIQUES & MODELS

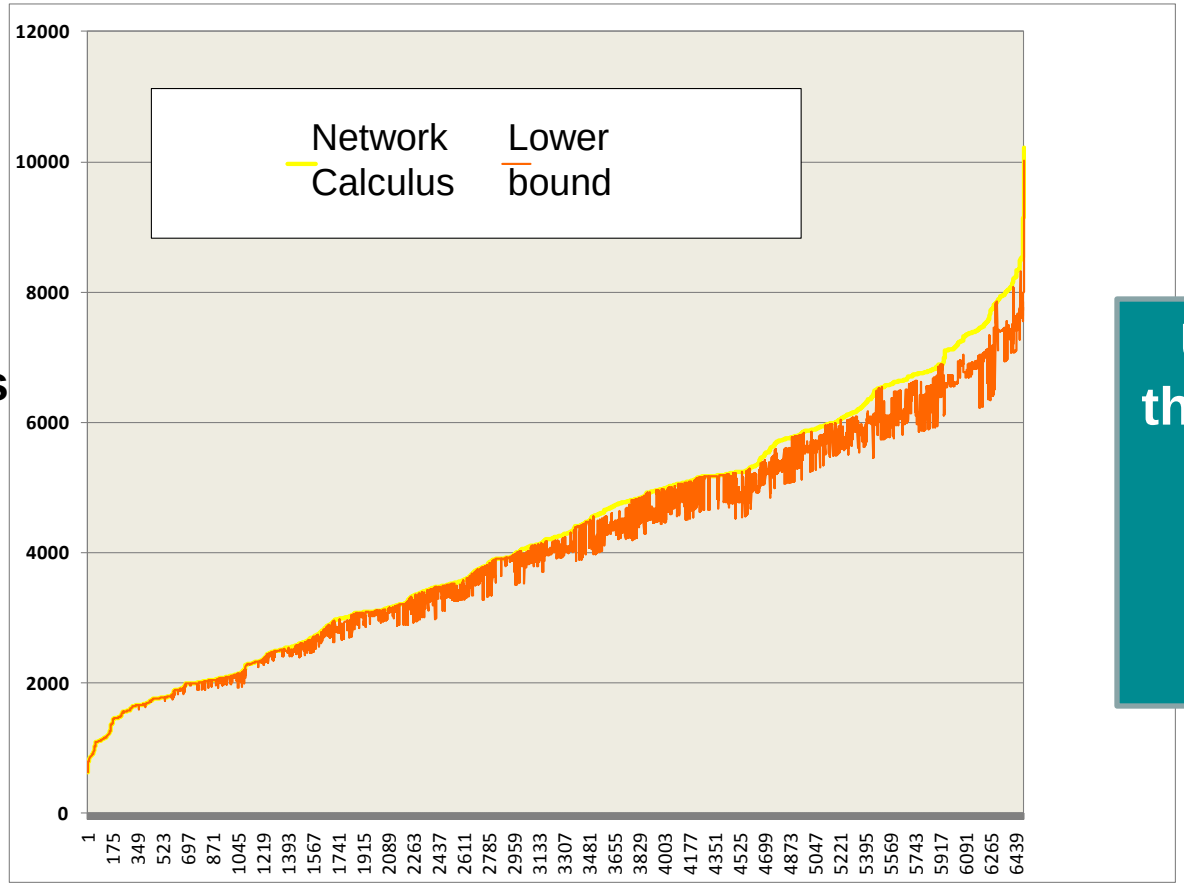
Timing verification techniques

Deterministic resource + bounded workload → timing verification feasible

- ✓ By construction: Time-Triggered Architecture
- ✓ Schedulability tests / utilization bounds
- ✓ Response time analysis
  - Single resource : e.g. critical instant,
  - Several resources interconnected: holistic, event-stream, …
- ✓ Network Calculus
- ✓ Model checking

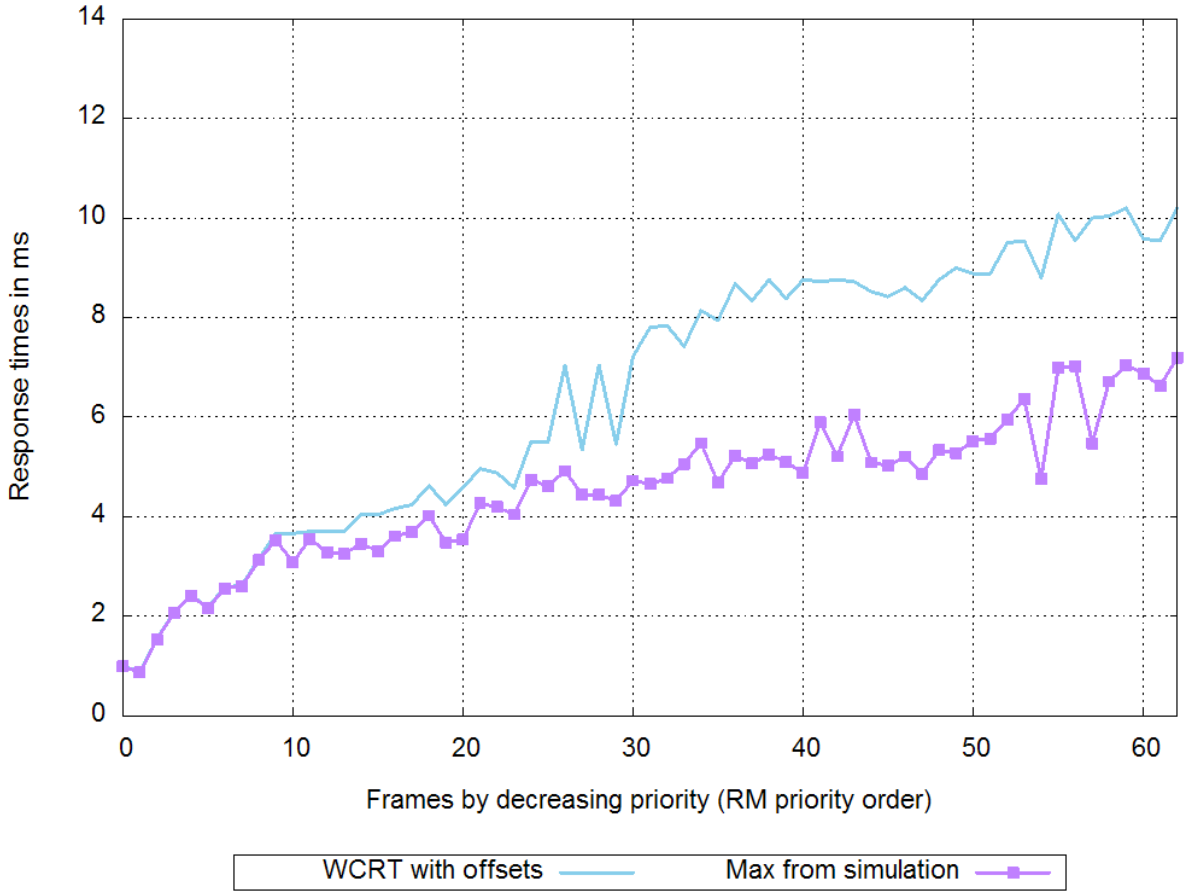
Exact analysis usually out of reach for end-to-end constraints in large (asynchronous) systems

Illustration: AFDX network analysed with Network Calculus state-of-the-art [10]



Upper-bound on the overestimation: only 16% on average over 100 configurations

What about simulation? not for maximum latencies – exp. on CAN networks from [12]



Difference WCRT vs simulation
max:
avg: 25%
max: 45%

Bridging the gap to realistic timing models [8]

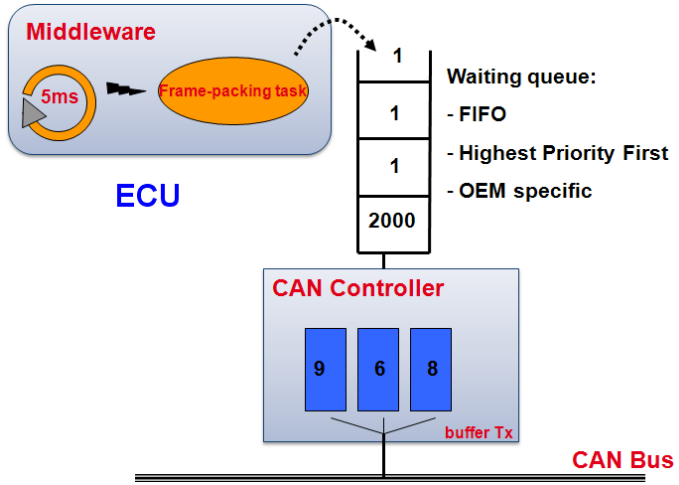
Higher load → less margin
→ **more accurate models**

1. Hardware models

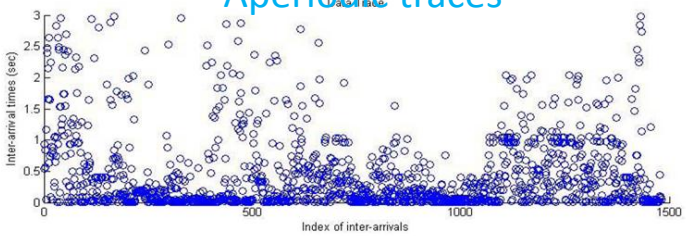
2. Software models (producer, sender, receiver, device drivers, etc)

3. Error models (reboot,errors)

4. Traffic models incl. aperiodic



Aperiodic traces



Analyses for safety critical systems : simple, peer-reviewed and documented

- ✓ Flawed analyses are dangerous in safety critical systems but (fine-grained) analyses are complex and error prone. Remember “CAN analysis refuted, revisited, etc” [6] ?!
- ✓ Implemented analysis have to make simplifications esp. in a heterogeneous systems (and tools do not document that well)
- ✓ Solutions ?
  - peer-review of the WCRT analyses is needed
  - coarse-grained / conservative but simple models as far as possible: e.g., [5,6] vs [9]
  - no black-box software: at least documentation of implemented analyses, ideally open-source
  - cross-validation between tools on benchmarks

Conclusions

- ✓ Large and growing body of techniques & models but there is a gap between research and industry practices
- ✓ Multiprocessor scheduling not mastered yet, diverse realities, HW still evolving
- ✓ MDE calls for automatic synthesis, configuration & deployment, with time being one facet

REFERENCES

[1] M.S. Mollison, J.H. Anderson, “Virtual Real-Time-Scheduling”, Seventh International Workshop on Operating Systems Platforms for Embedded Real-Time Applications, pp. 33-40, July 2011.

[2] H. Kellerman, G. Nemeth, J. Kostelezky, K. Barbehön, F. El-Dwaik, L. Hochmuth, “BMW 7 Series architecture”, ATZextra, November 2008.

[3] N. Navet, B. Delord (PSA), M. Baumeister (Freescale), “Virtualization in Automotive Embedded Systems : an Outlook”, talk at RTS Embedded Systems 2010, Paris, France, March, 2010. Available at http://nicolas.navet.eu

[4] N. Navet, “Automotive communication systems: from dependability to security”, talk at the 1st Seminar on Vehicular Communications and Applications (VCA 2011), Luxembourg, May 2011. Available at http://nicolas.navet.eu

[5] R.I. Davis, S. Kollmann, V. Pollex, F. Slomka, "Controller Area Network (CAN) Schedulability Analysis with FIFO queues”. In proceedings 23rd Euromicro Conference on Real-Time Systems (ECRTS), pages 45-56, July 2011.

[6] R. Davis, N. Navet, "Controller Area Network (CAN) Schedulability Analysis for Messages with Arbitrary Deadlines in FIFO and Work-Conserving Queues", Proc. of the 9th IEEE International Workshop on Factory Communication System (WFCS 2012), May 21-24, 2012, Lemgo/Detmold, Germany. Available at http://nicolas.navet.eu

[7] R. Davis, A. Burn, R. Bril, and J. Lukkien, “Controller Area Network (CAN) schedulability analysis: Refuted, revisited and revised”, Real-Time Systems, vol. 35, pp. 239–272, 2007.

REFERENCES

[8] N. Navet, H. Perrault, “CAN in Automotive Applications: a Look Forward“, 13th International CAN Conference, Hambach, Germany, March 5-6, 2012. Available at http://nicolas.navet.eu

[9] D. Khan, R. Davis, N. Navet, “Schedulability analysis of CAN with non-abortable transmission requests“, 16th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA 2011), Toulouse, France, September 2011. Available at http://nicolas.navet.eu

[10] M. Boyer, N. Navet, M. Fumey, “Experimental assessment of timing verification techniques for AFDX“, Embedded Real-Time Software and Systems (ERTS 2012), Toulouse, France, February 1-3, 2012. Available at http://nicolas.navet.eu

[11] J.B. Itier, “A380 Integrated Modular Avionics”, http://www.artist-embedded.org/docs/Events/2007/IMA/Slides/ARTIST2_IMA_Itier.pdf, ARTIST2 meeting on Integrated Modular Avionics, 2007.

[12] P. Meumeu-Yomsi, D. Bertrand, N. Navet, R. Davis, “Controller Area Network (CAN): Response Time Analysis with Offsets”, 9th IEEE International Workshop on Factory Communication System (WFCS 2012), May 21-24, 2012, Lemgo/Detmold, Germany.

[13] N. Navet, A. Monot, B. Bavoux, F. Simonot-Lion, “Multi-source Software on Multicore Automotive ECUs – Combining Runnable sequencing with task scheduling“, IEEE Transactions on Industrial Electronics, vol 59, n° 10, 2012. Available at http://nicolas.navet.eu

[14] S. Duarte Penna, “Networking in Modern Avionics: Challenges and Opportunities”, RTN 2011.