Multicore scheduling in automotive ECUs

Aurélien Monot - PSA Peugeot Citroën, LORIA
Nicolas Navet - INRIA, RealTime-at-Work
Françoise Simonot - LORIA
Bernard Bavoux - PSA Peugeot Citroën

Talk at ERTS2 2010
Toulouse, May 21st 2010

Outlook

Context:
New tools and methodologies are needed as multicore ECUs are being introduced in the automotive EE architecture.

Problem:
How to address the scheduling of numerous runnables on a multicore ECUs in the context of the automotive domain?

Method:
Deployment of load balancing algorithms in ECU configuration tools.

The case of a generic car manufacturer

Typical number of ECUs in a car in 2000 : **20**

Typical number of ECUs in a car in 2010 : **over 40**

The number of ECUs has more than doubled in 10 years

Other examples

Between **60** and **80** ECUs in the Audi A8

Over **100** ECUs in some Lexus !

Moving towards multicore architecture

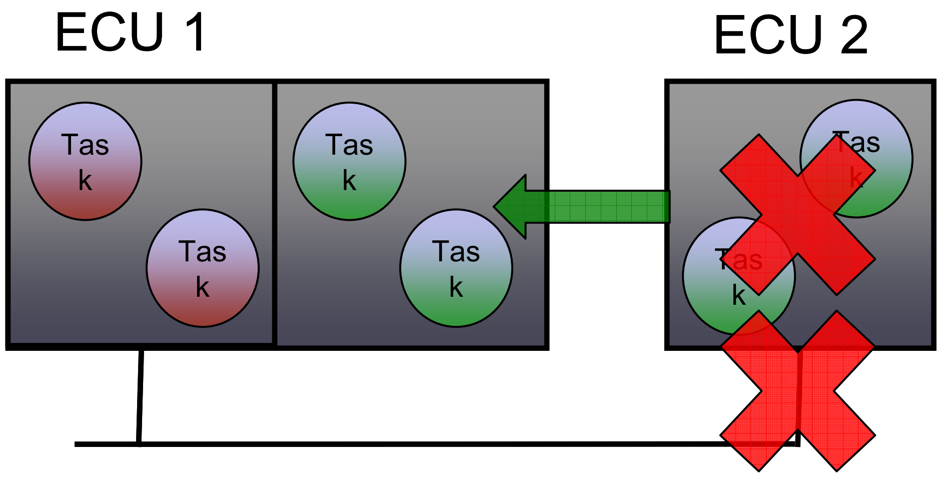
Decreasing the complexity of in-vehicle architecture:

➪reduces EE design and verification efforts

➪decreases number of network interfaces

➪decreases traffic on CAN network

➪reduces costs



Moving towards multicore architecture

Other use cases for the automotive domain

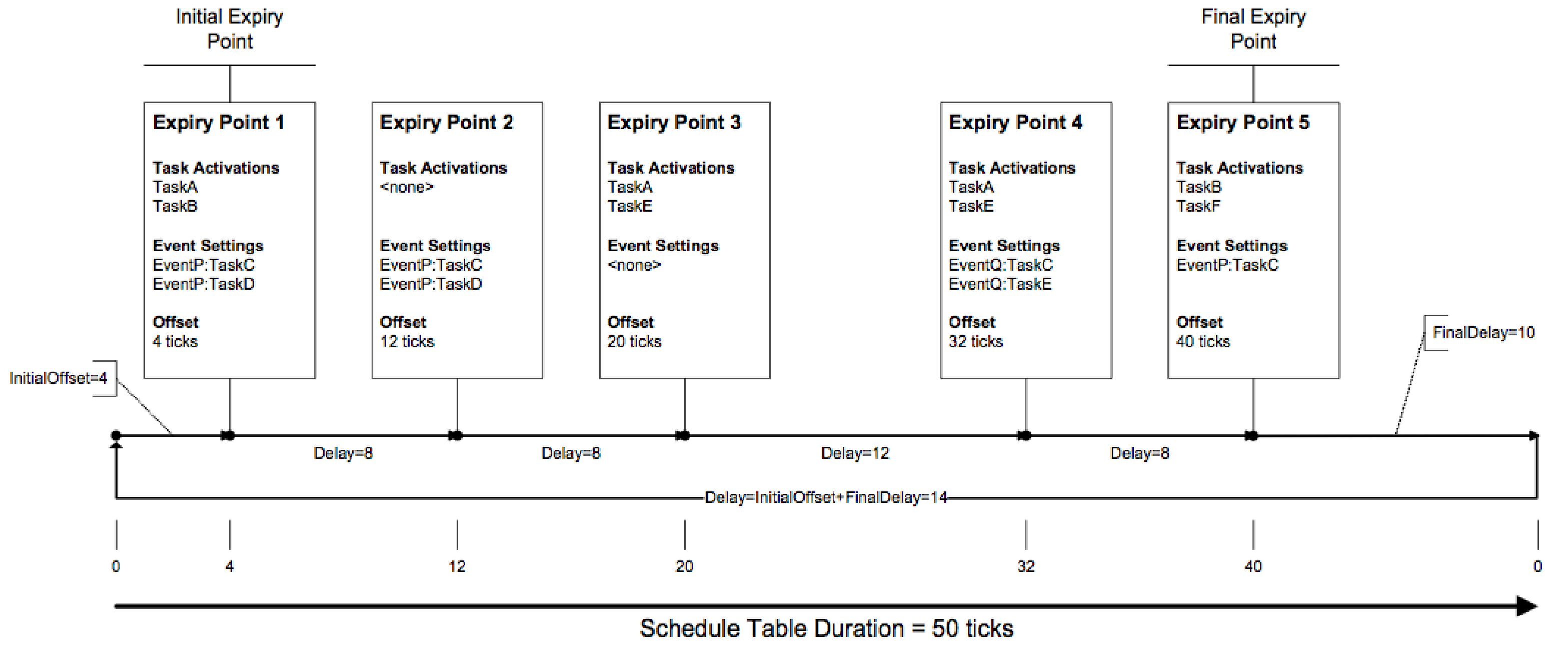
- Dealing with resource demanding applications
  - *engine control, image processing...*
- Improving the safety
  - *segragation of multi-source software, ISO26262...*
- Dedicated use of core
  - *monitoring, event-triggered tasks*

General benefits of multi-core

- reduced power consumption
- reduced heat
- reduced EMC

AutoSAR requirements

- Static partitioning
- Static cyclic scheduling using schedule tables
- BSW are all allocated on the same core



Problem

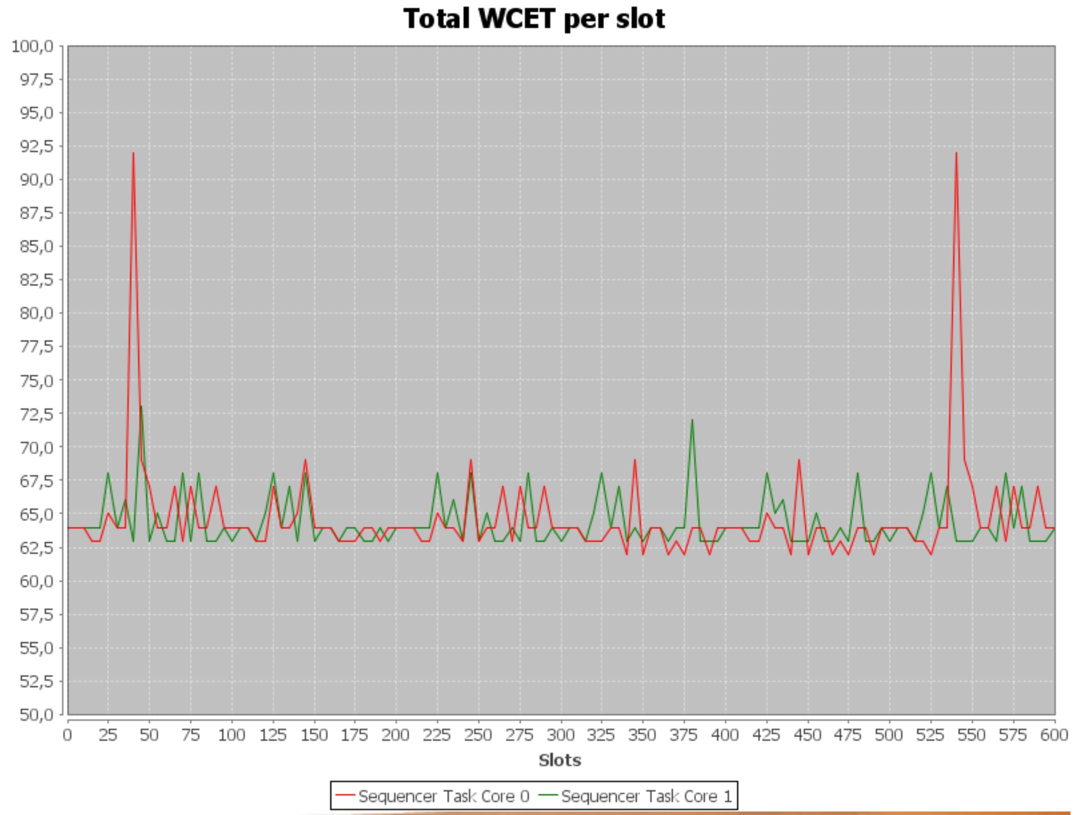
Goal: schedule numerous runnables on a multicore ECU

Two sub-problems

- Partitioning
  - 600 runnables on 2 cores
- Build schedule table
  - 300 runnables in 200 slot

Sub-objectives and criteria

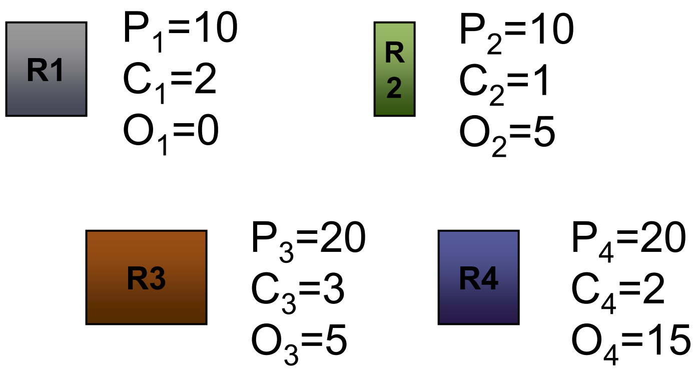
- Avoid load peaks
  - Max
- Balance load over time
  - Standard Deviation



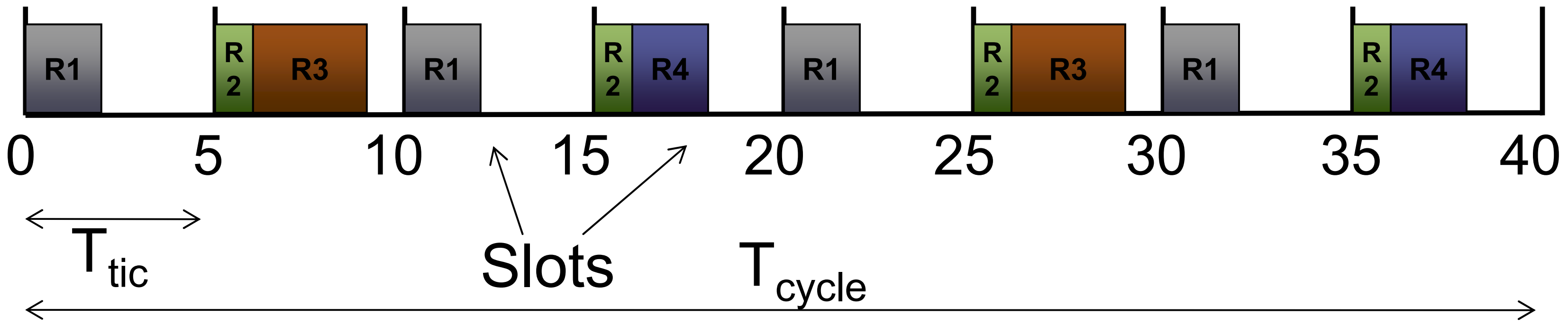
Model

Runnables

- **P**eriod
- W**C**ET
- Initial **O**ffset
- Core allocation constraint
- Colocation constraint



Sequencer task



Solution

Partitioning is dealt with as a bin packing problem

➪**Worst fit decreasing algorithm** with fixed number of bins

Load Balancing is done with **the Least Loaded algorithm (LL)** inspired from CAN domain [Grenier and Navet ERTSS2008]

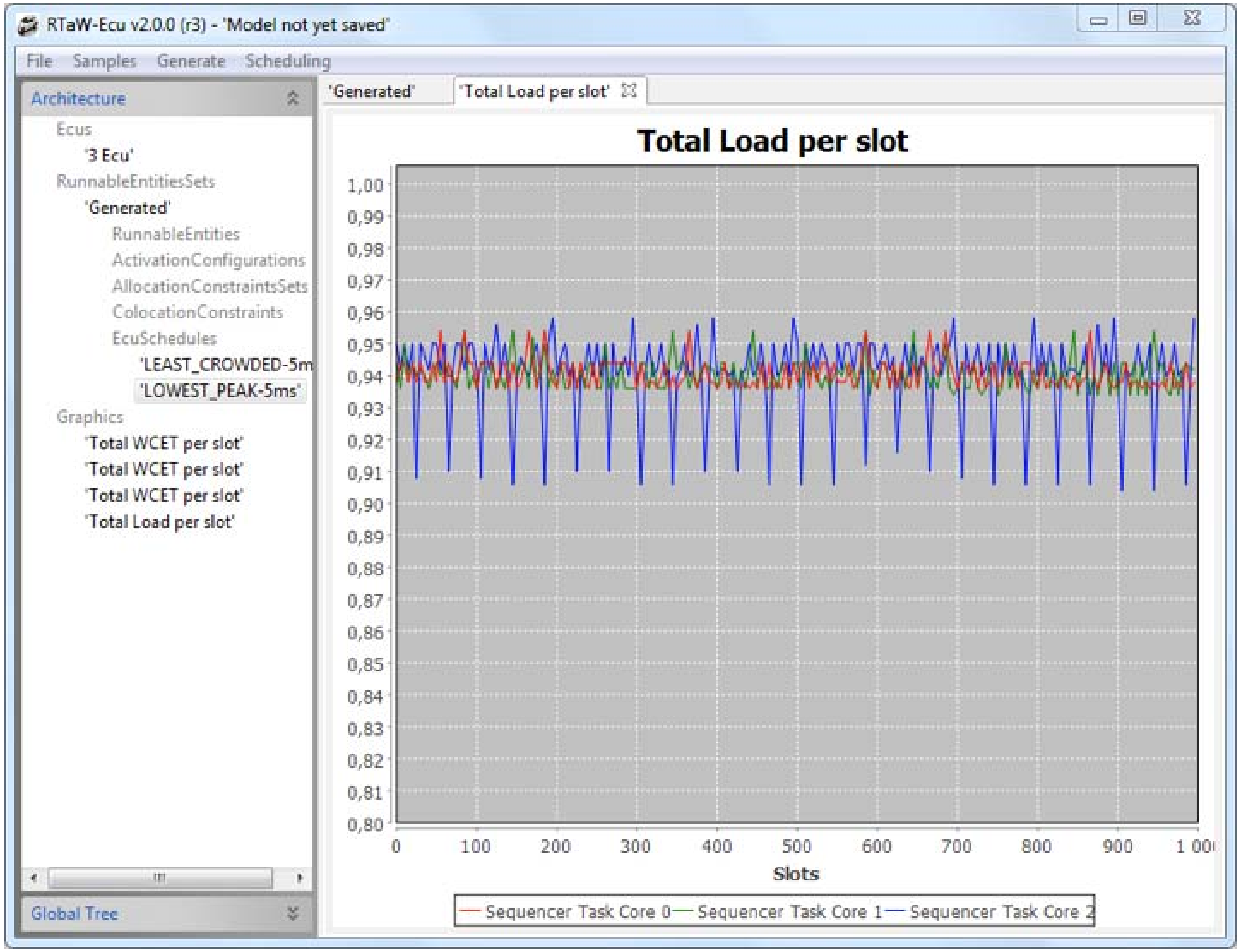
➪Extended to handle non harmonic runnable sets **(G-LL)**

➪Improved so as to reduce further load peaks **(G-LLσ)**

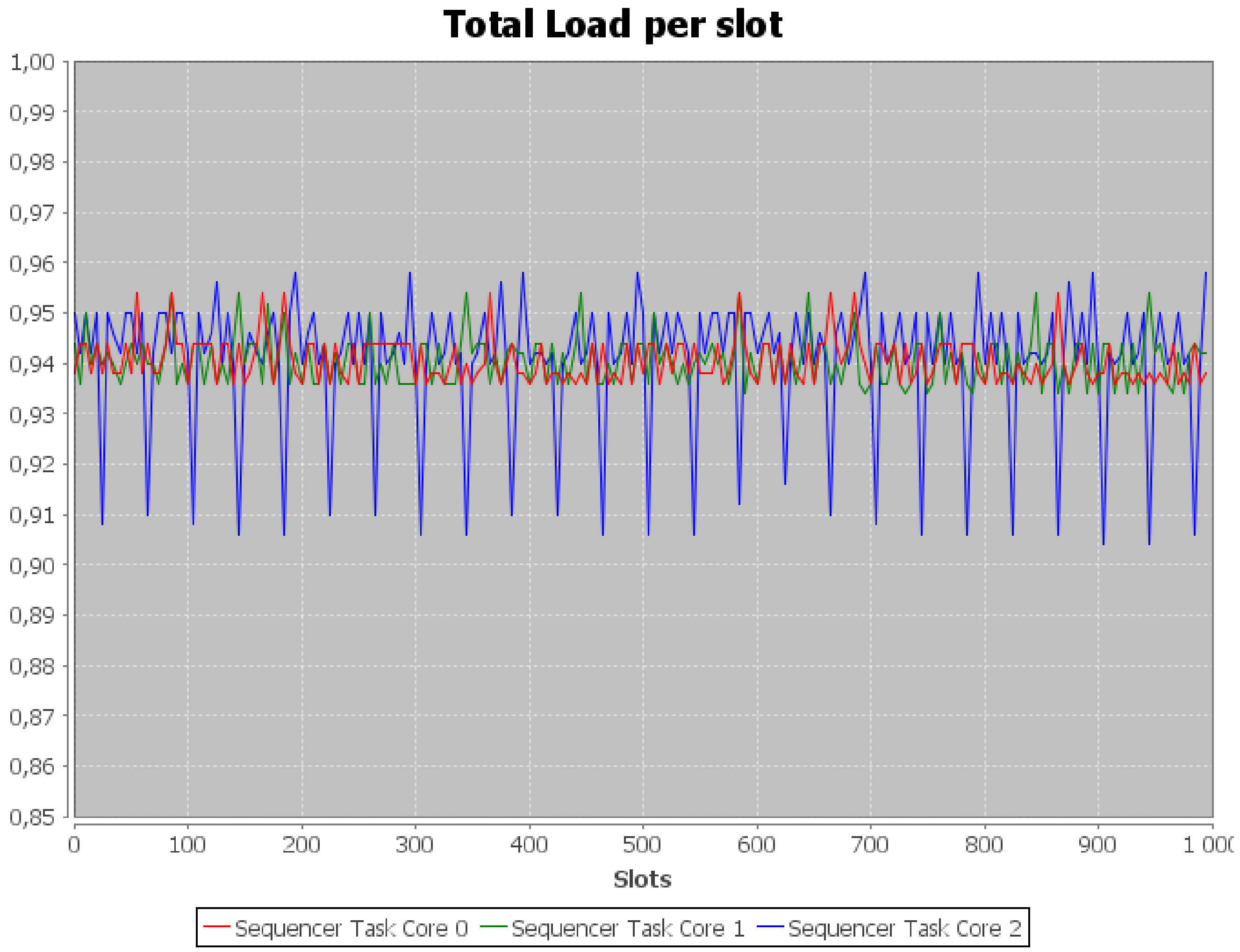
Implemented in a tool

➪Freely available soon at http://www.realtimeatwork.com

Experiments with RTaW-ECU



Harmonic task sets



LL
Max: 4.79
Min: 4.52
StdDvt: 0.038

G-LLσ
Max: 4.75
Min: 4.65
StdDvt: 0.018

Generated load: 94%, T_{tic}=5ms, T_{cycle} = 1s

Loria RTaW RealTime-at-Work PSA PEUGEOT CITROËN

Non harmonic task sets

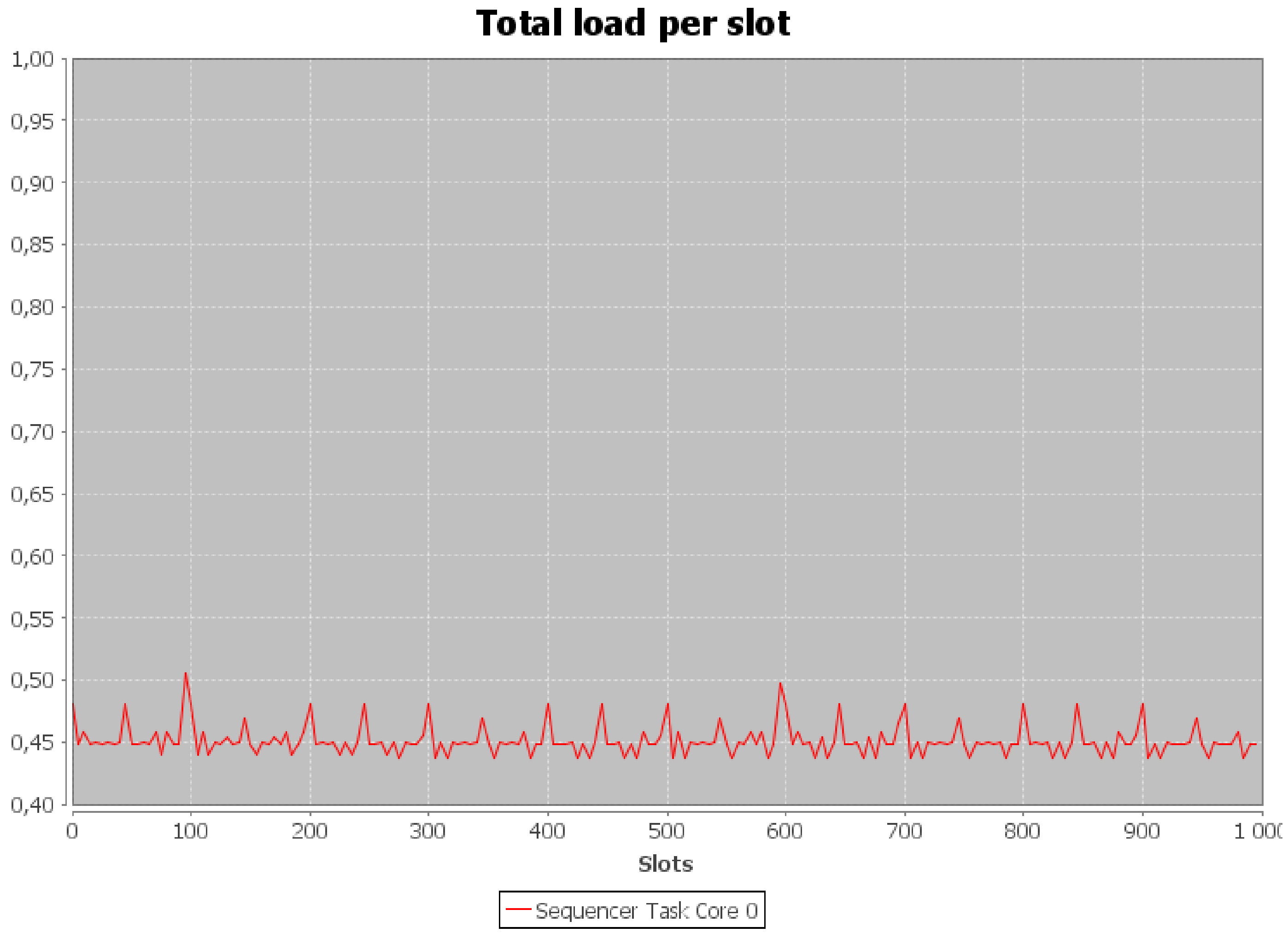
Schedulability bound in the harmonic case $1-\frac{C_{\max}}{T_{tic}}$

Max WCET (µs)	150	300	900
Schedulability bound in the harmonic case	97%	94%	82%
Success % of LL	96%	96%	92%
Success % of G-LL	100%	100%	100%

Generated CPU load	95%	97%	95%	97%
Schedulability bound in the harmonic case	94% Max WCET = 300µs		82% Max WCET = 900µs	
Success % of LL	64%	18%	12%	1%
Success % of G-LL	94%	94%	30%	5%
Success % of G-LL$_{1\sigma}$	100%	100%	97%	76%

Statistics collected over 1000 generated runnable sets

Multiple synchronized sequencer tasks per core



Incremental scheduling of three synchronized sequencer tasks with respective load of 45%, 35% and 15% resulting in 95% of the core capacity.

T_{cycle}=1000ms and T_{tic}=5ms

Multiple non synchronized sequencer tasks per core

Case arises for sequencer tasks using **different tic counters**

- Engine control applications (standard time vs RPM)

Any offset between the sequencer tasks and all clock rates are possible during runtime

- each sequencer task needs to be **balanced independantly**

Verification is possible considering maximum clock rates

- *Multi-frame scheduling* results can be used

Conclusion

Adoption of multicore ECU raises new challenges

- Evolution of software architecture design
- Scheduling of software components

We propose runnable scheduling heuristics for ECUs

- Fast and performant
- Easily adaptable for more advanced applications
- Compatible with AutoSAR R4.0 and its multicore extensions

Future work

- Precedence constraints
- Lockstep synchronization
- Distributed timing chains

Thank you for your attention