Simulation-Based Fault Injection as a Verification Oracle for the Engineering of Time-Triggered Ethernet networks

Loïc FEJOZ, RealTime-at-Work (RTaW)
Bruno REGNIER, CNES
Philippe, MIRAMONT, CNES
Nicolas NAVET, University of Luxembourg / designCPS.com

Embedded Real-Time Software and Systems (ERTS 2018)
Toulouse, France | January 31 – February 2, 2018

Context & Objective

- ✓ TTEthernet from TTTech, based on SAE6802, is considered for use as high-speed data rate in future launchers (MIL-STD-1553B replacement)
- ✓ Time-Triggered communication eases the design of applications with dependability constraints

- ✓ This work: assess the precision and robustness of TTE clock synchronization algorithm
  - o In the fault-free case
  - o With permanent failures such as a link loss
  - o With permanent and transient failures such as transmission errors – outside the fault-hypotheses of the design

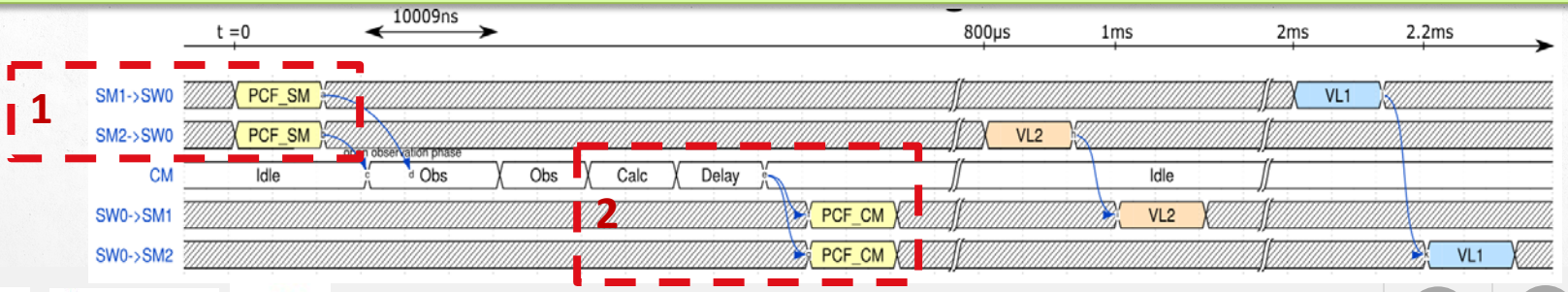
- ✓ Parts of a collaboration between CNES, RTaW and ONERA

A primer on TTE and its clock-synchronization algorithm

- ✓ TTEthernet (TTE) is a switched Ethernet technology for critical systems marketed by TTTech and standardized as SAE6802
- ✓ Combines Time-Triggered (TT) + (AFDX-like) Rate-Constrained (RC) + Best-effort (BE) traffic

✓ Clock synchronization through the exchange of ***Protocol Control Frames*** (PCF)

- o Step 1: ***Synchronization Masters*** (SM) "send" local clock to ***Compression Master(s)*** (CM)
- o Step 2: CM calculates new clocks based on received SMs clocks and sends back to SMs
- o Step 3: SMs adjust their local clock

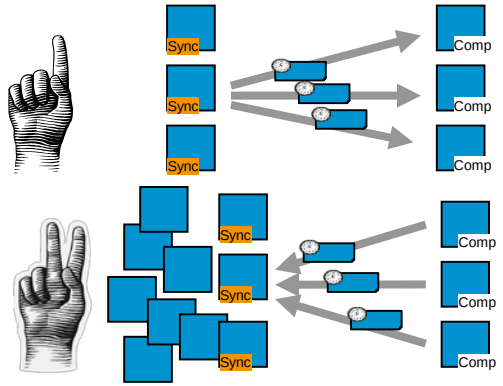


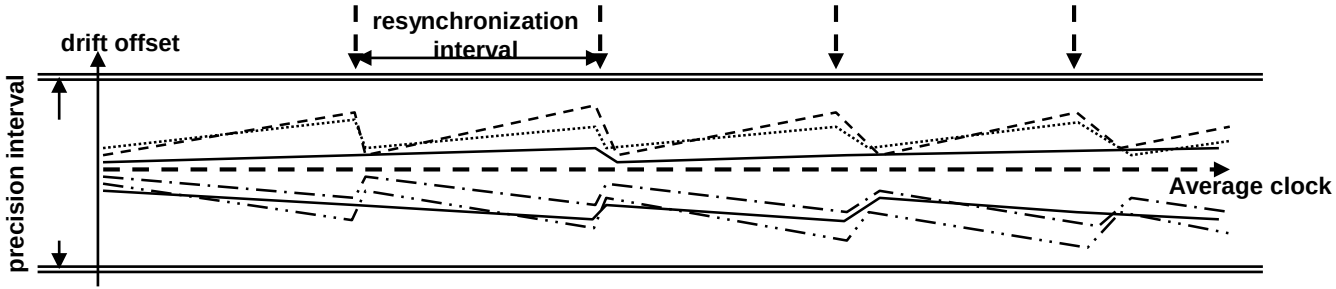
Time-triggered Ethernet – two step synchronization

Protocol Control Frames called “Integration Frames” are used to perform all synchronization functions.

They are transmitted accordingly:

1. The Synchronization Masters send Integration Frames at the beginning of each Integration Cycle. The timing of these frames is used for the “voting”
2. The Compression Masters send Integration Frames to everybody, timing them in a special way so that everybody can correct their clocks





Slide courtesy TTTech – all rights reserved

The need for Simulation-Based Fault Injection (SBFI)

Key correctness properties of TTA/TTE have been formally established but

- ✓ Formal models do not cover all properties of interest
- ✓ Proofs are made with assumptions (e.g., simplifications) not always met by actual systems
- ✓ Proofs usually do not go beyond the design fault-hypotheses, but what happens outside? → SBFI helpful here
- ✓ Proofs are based on standards/specifications but implementation may not fully comply and implementation choices may matter
- ✓ Fine-grained simulation models requires complete understanding of the system
- ✓ …

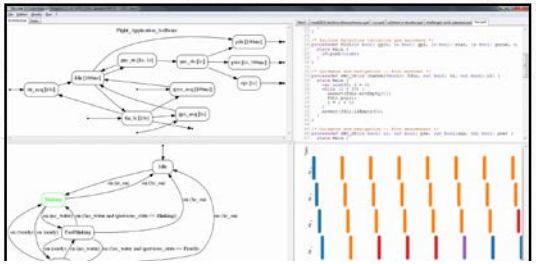
Verification and comprehensive understanding of new technologies in critical systems best achieved through combined use of testbeds, formal verification and simulation

TTE Model in CPAL and its validation

CPAL - a *real-time embedded systems* specific language

1 **Model and program**

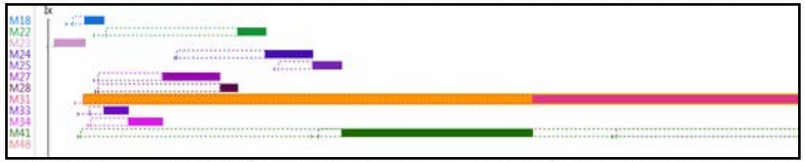
functional and non-functional concerns

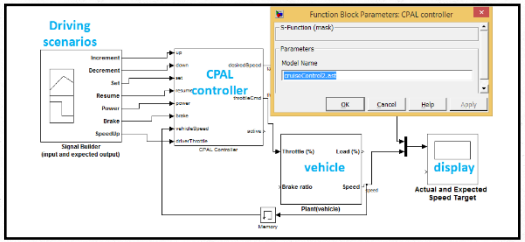


2 **Simulate**

Available from www.designcps.com

possibly embedded within external tools such as RTaW-Pegase™ and Matlab/Simulink ™

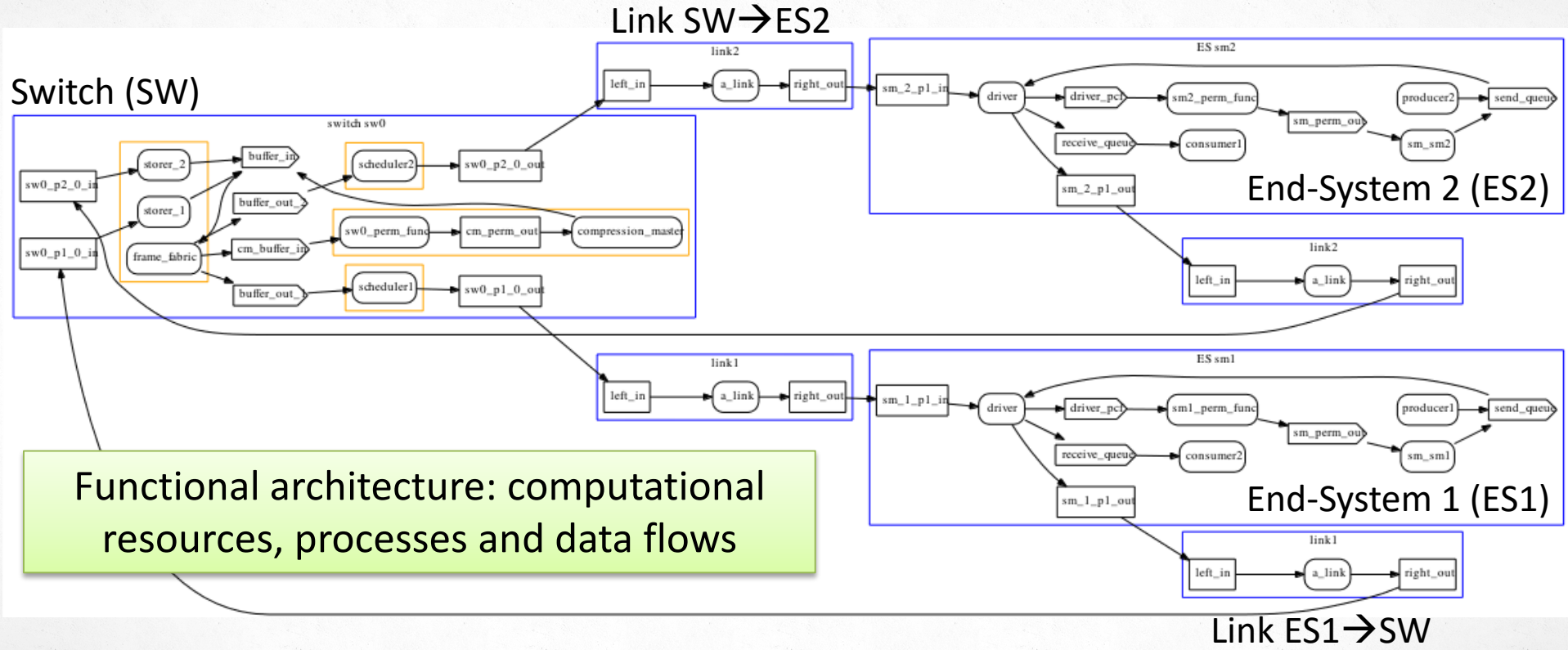




3 **Execute** bare metal or hosted by an OS

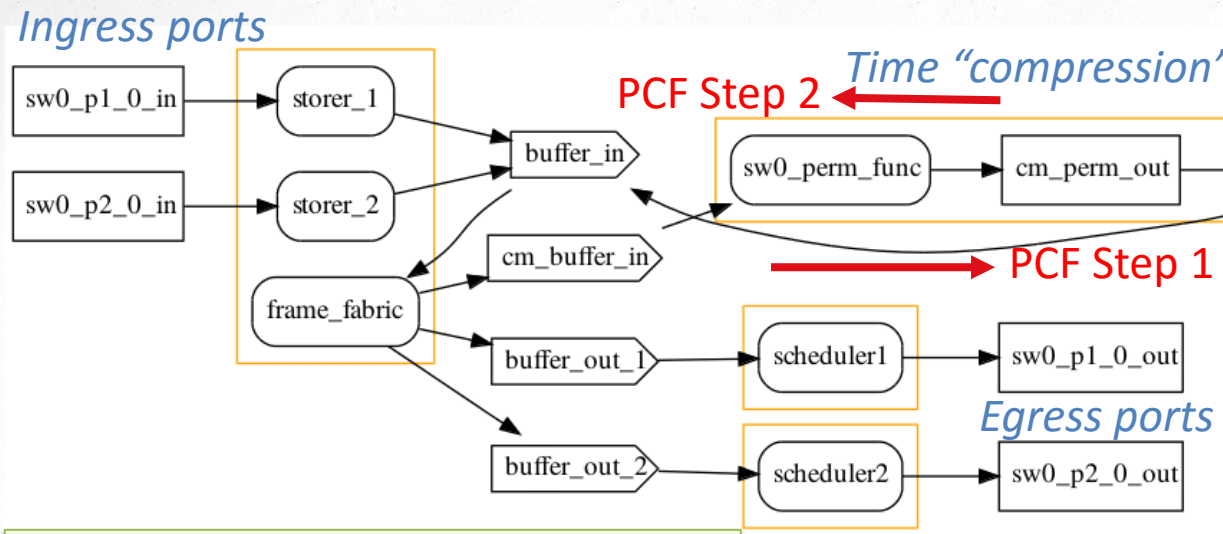
A joint project of RealTime-at-Work and University of Luxembourg since 2012

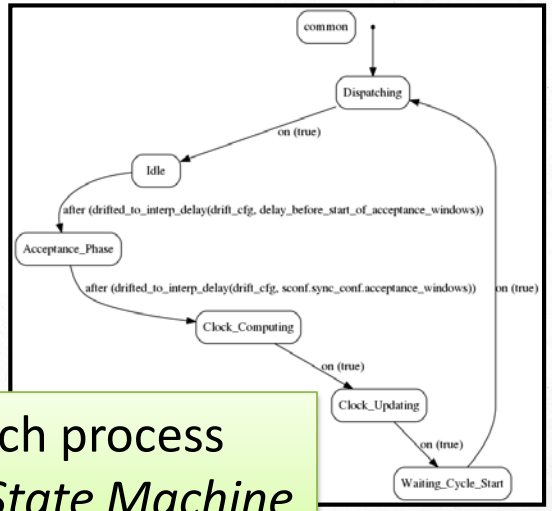
Global TTE Model with 1 SW and 2 ES



Functional architecture: computational resources, processes and data flows

Model of a Switch serving as CM





Yellow rectangles are *processes* that execute in parallel

Internal logic of each process described by a *Finite State Machine*

Experimental Workflow

- ✓ Network description
- ✓ TT schedule generation
- ✓ …

TTE simulator: Tools Operational Requirements (TOR) specified in Capella

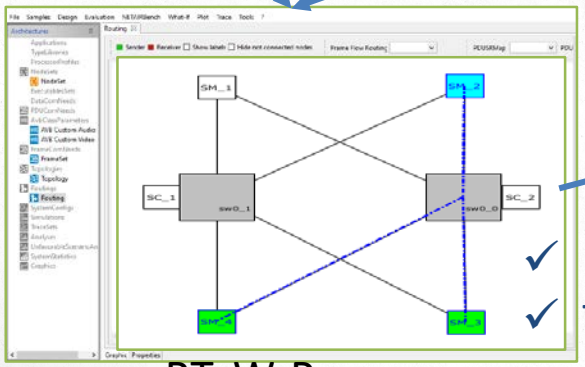
TTEPlan
TTEBuild

Deployment

TTE Network Monitoring

Import description

Import traces

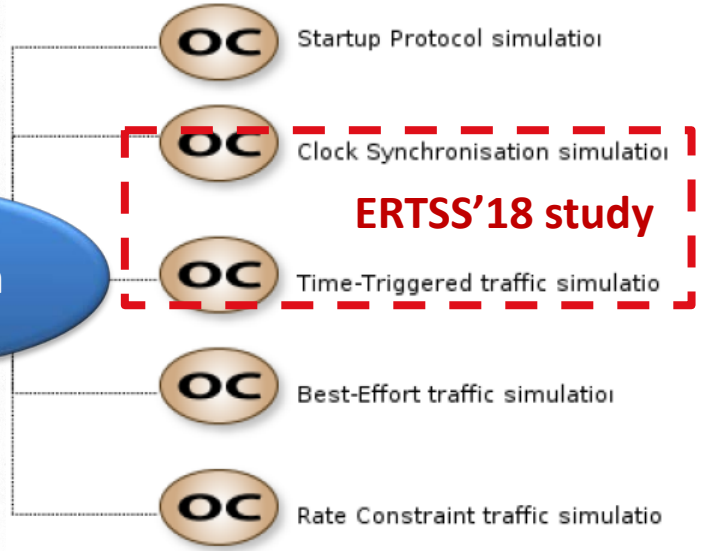


RTaW-Pegase

Generate CPAL model

Simulation

- ✓ Network visualisation
- ✓ Trace analysis



Model development and validation

- ✓ Project requirements: ***white box models*** so that new failures scenario can be added → motivated the choice of CPAL
- ✓ CPAL library to enable the re-use of the automata defined in the standard
- ✓ Extension of the simulation engine to handle drifts to simplify code of model
- ✓ TTE model is very fine-grained: need for specifications and code reviews

Model validation

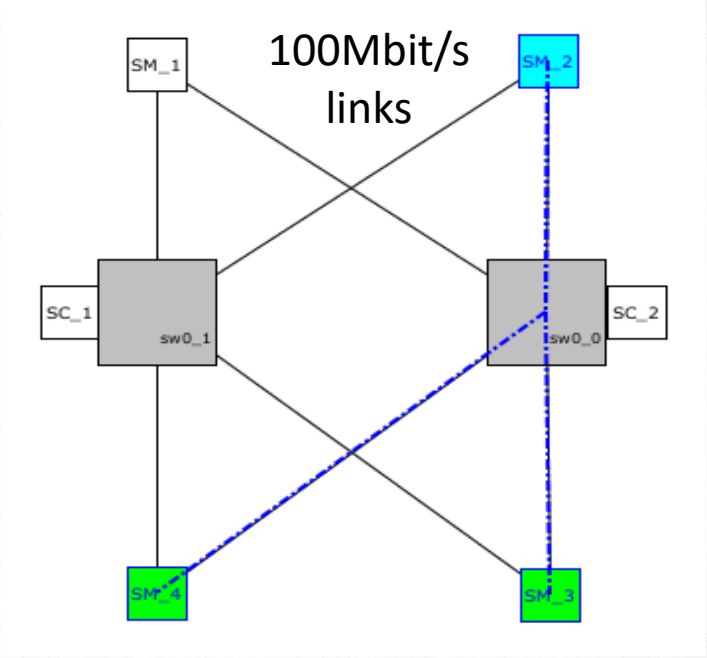
1. On a small configuration, comparison of model traces against “pen & paper” solution
2. Comparison against the black-box TTE simulation model in RTaW-Pegase implemented by another team
3. On a small configuration, comparison against monitored communication traces
4. Comparison with formal results from the literature [Dutertre, Steiner et al] and ad-hoc math. analysis

Accuracy of TTE Clock Synchronization Service

Experimental setup – 2 CMs and 4 SMs configuration

- ✓ Switches are CMs and end-systems are SMs
- ✓ **Clock drift:** linear model, per node drift value drawn at random at start of each cycle, **±50 ppm** thus 500ns max. per 5ms cycle between any two clocks
- ✓ Done w/wo implementation delays in end-systems and switches, hereafter with delays
- ✓ Statistics with samples of size 100 000

mac_pcf_transmission_delay	**143µs600ns**
cm_function_delay	30µs27ns
integration_cycle	**5ms**
acceptance_windows	**20µs18ns**
cm_scheduled_receive_pit	173µs627ns
sm_schedule_receive_pit	337µs245ns



Multicast stream received by the green nodes [RTaW-Pegase screenshot]

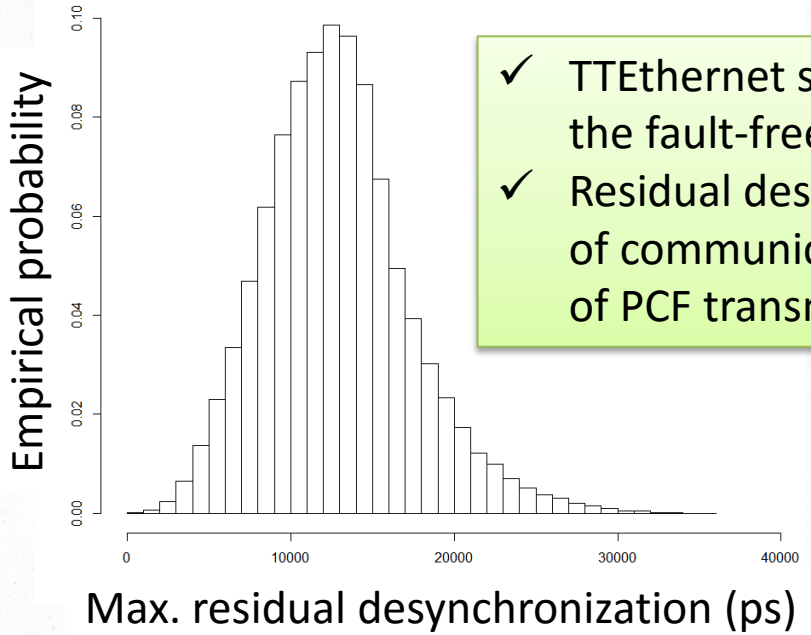
Performance Metrics

- ✓ Maximum residual desynchronization during a communication cycle: *"max. difference between any two clocks after the last SM has re-synchronized"*
- ✓ Maximum desynchronization during a communication cycle: *"max. difference between any two clocks"*

Knowing the actual clock accuracy – and factors having an influence on it - is crucial to set in a safe manner:

- ✓ the TT transmission window length → dependability & bandwidth efficiency
- ✓ Specifications to suppliers

Maximum residual desynchronization in the fault-free case

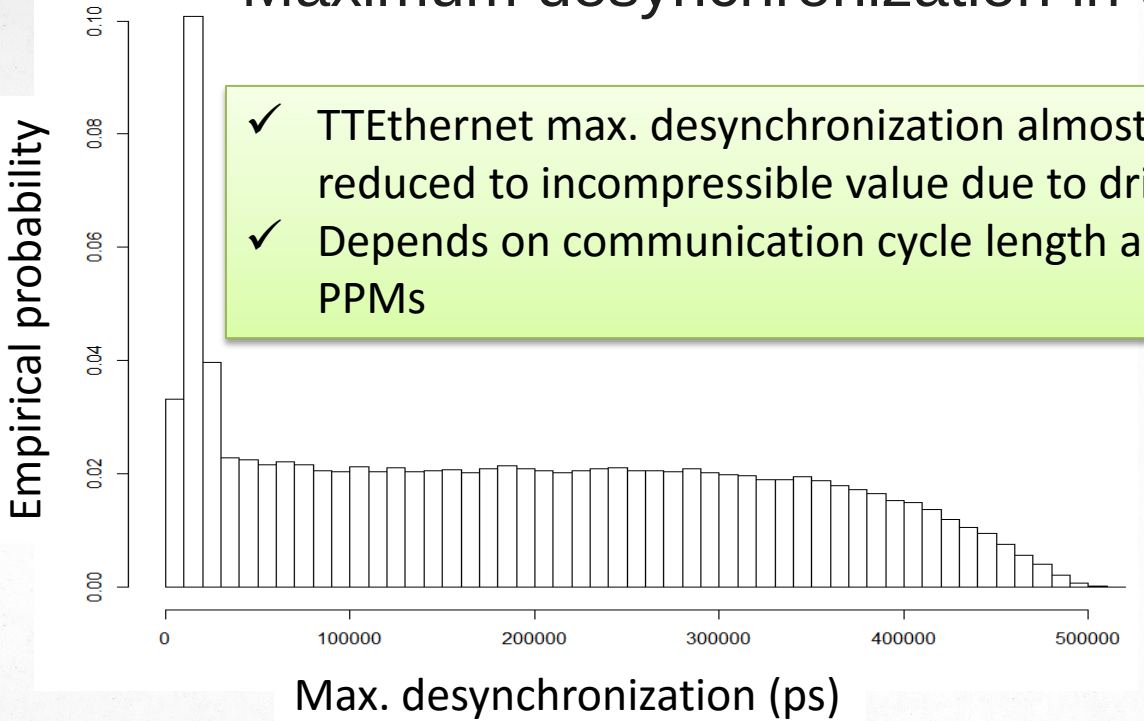


- ✓ TTEthernet synchronization very efficient in the fault-free case
- ✓ Residual desynchronization is independent of communication cycle length but depends of PCF transmission delays and drifts PPMs

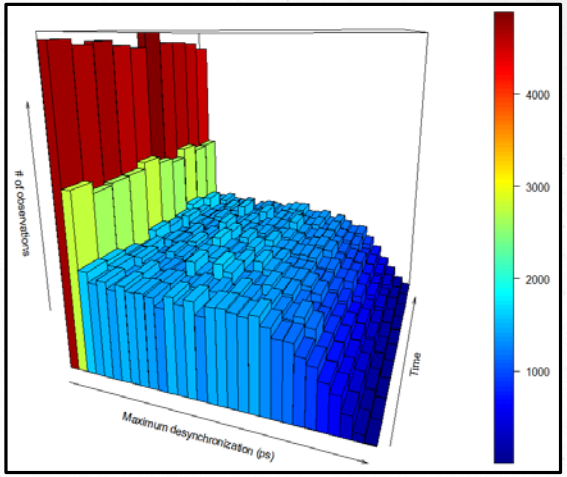
Min.	1st Qu.	Median	Mean	3rd Qu	Max.
431	9 828	12 545	12 840	15 331	**35 569**

Unit: picoseconds

Maximum desynchronization in the fault-free case



- ✓ TTEthernet max. desynchronization almost reduced to incompressible value due to drifts
- ✓ Depends on communication cycle length and PPMs

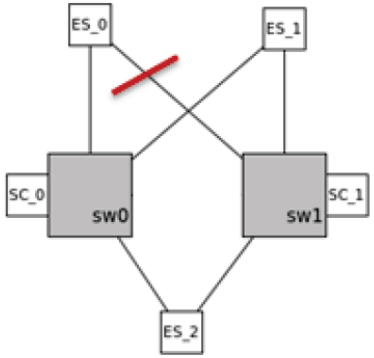


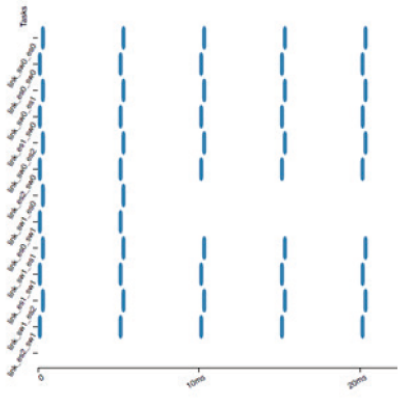
Small variations over time but not trends (eg errors accumulation)

Min.	1st Qu.	Median	Mean	3rd Qu	Max.
478	64 090	184 713	193 091	306 316	**512 821**

Unit: picoseconds

Maximum desynchronization will permanent link failure and transmission errors



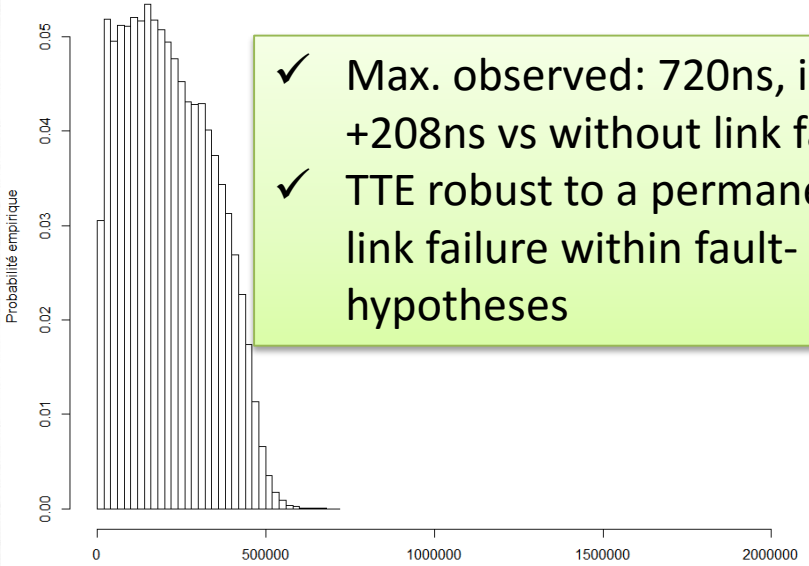


```
[0.005337070946] es0 SM Clock updating
with 2 PCFs
[0.005337070946] es1 SM Clock updating
with 2 PCFs
[0.005337070946] es2 SM Clock updating
with 2 PCFs
[0.010343805556] es0 SM Clock updating
with 1 PCFs
[0.010343805556] es1 SM Clock updating
with 2 PCFs
[0.010343805556] es2 SM Clock updating
with 2 PCFs
[0.015350540165] es0 SM Clock updating
with 1 PCFs
```

- ✓ **Link failure:** SW1 calculates time correction with a single SM clock – SM ES0 receives a single clock correction instead of 2
- ✓ **Transmission errors with a frame error rate 3%:** « inconsistent error omission » on all links which is outside design fault-hypotheses – Nodes may not receive clock correction during several cycles..

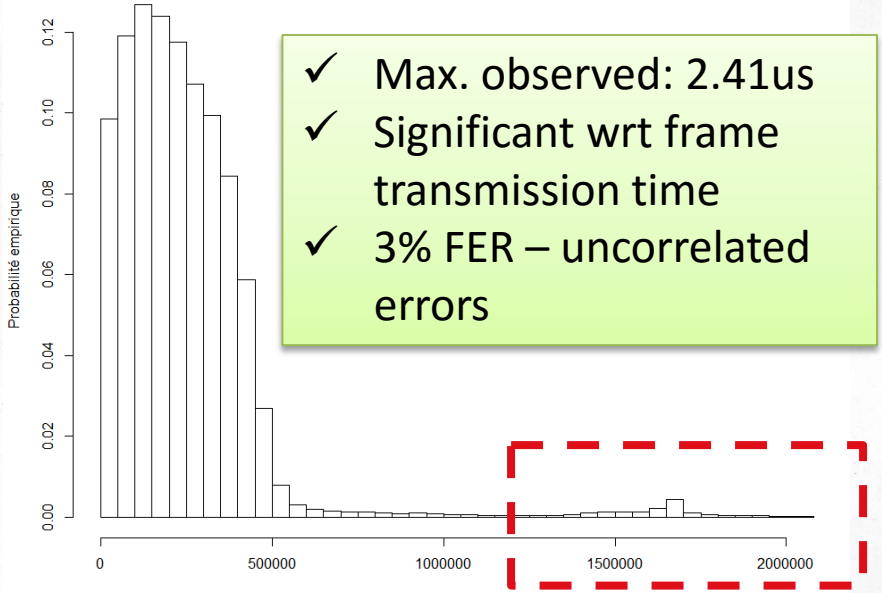
Maximum desynchronization will permanent link failure and transmission errors

Link failure



- ✓ Max. observed: 720ns, i.e. +208ns vs without link failure
- ✓ TTE robust to a permanent link failure within fault-hypotheses

Link failure + transmission errors



- ✓ Max. observed: 2.41us
- ✓ Significant wrt frame transmission time
- ✓ 3% FER – uncorrelated errors

Key takeaways

- ✓ Under realistic clock drifts, experiments suggest **TTE is very effective at maintaining a global clock** with a high precision
- ✓ TTE is robust to a permanent link failure but safety margin must be taken if transmission errors are to be tolerated
- ✓ Clock synchronization builds on the *transparent clock* mechanism – correctness and efficiency of its implementation is absolutely key
- ✓ Simulation model verification and calibration was only possible thanks to trace analysis and, to a lesser extent, formal analyses
- ✓ CPAL has proven to be an adequate modelling environment – importance of specifications, code reviews and the extensions to simplify code and facilitate traceability with standard

Thank you for your attention!

Any questions or comments? contact: nicolas.navet@uni.lu