

Measuring and Modeling Quantitative Aspects of Networked Embedded Systems

Reinhard German

Computer Networks and Communication Systems,
University of Erlangen-Nürnberg

March 5, 2007

Dagstuhl Seminar “Quantitative Aspects of Embedded Systems”

Contents

■ Measurements

- joint work with Kai-Steffen Hielscher, Kemal Köker

■ UML-Based Simulation and Test

- joint work with Isabel Dietrich, Matthias Beyer (Fraunhofer IIS)

■ 2 Automotive Applications

- joint work with Thomas Herpel (INI.FAU)
- and with Thorsten Frunzke (Audi Electronic Ventures)

Measurements

- Measurements are needed for
 - a direct study of the performance of a system
 - in modeling: fitting for input modeling, calibration and validation
 - a performance model of networked computer systems should contain relevant aspects of applications, operating systems, interactions between hardware, components, and computing systems
 - this holds both for desktop and embedded systems

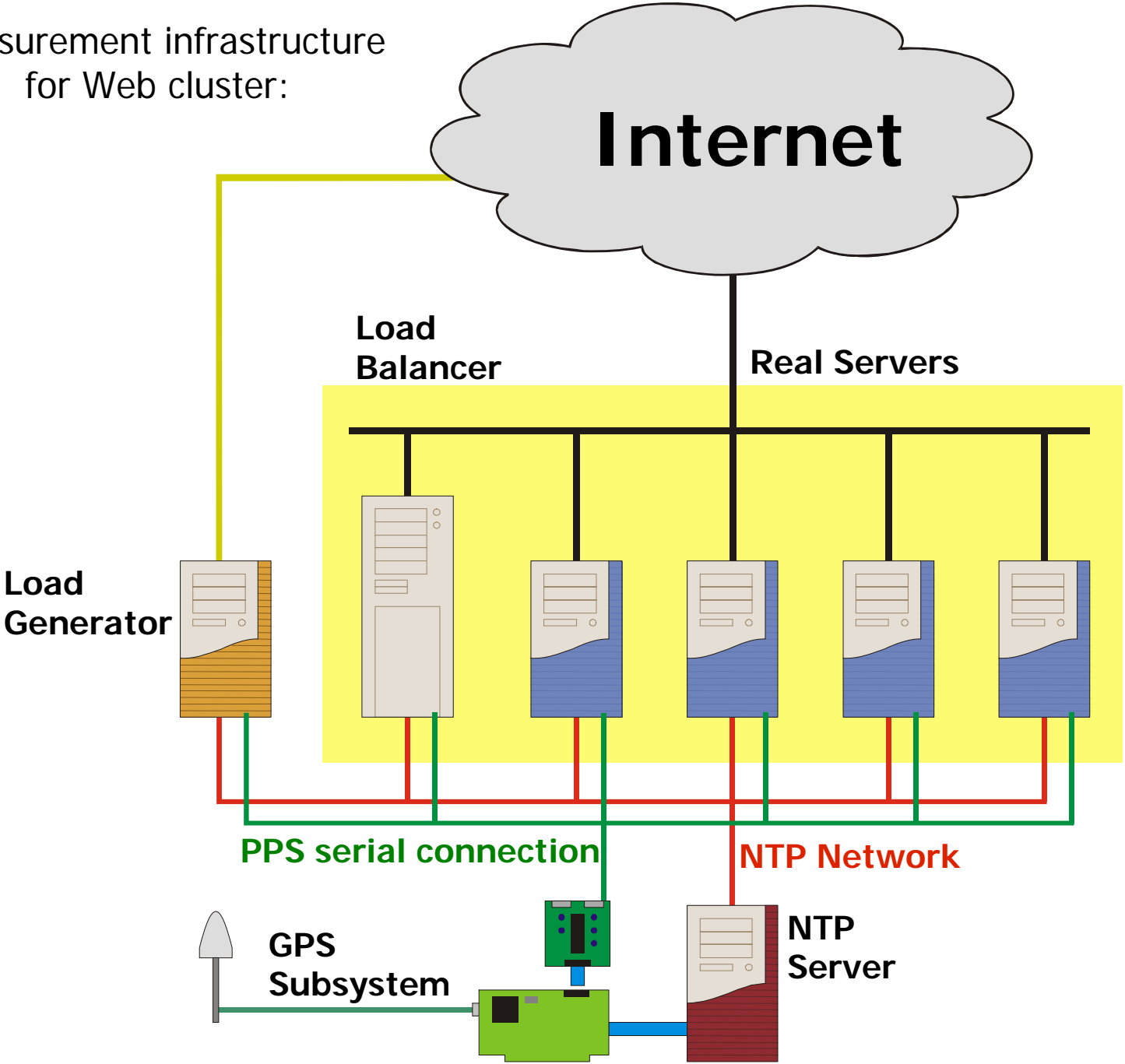
- Here we present 3 measurement efforts
 - one-way network delays
 - external interrupt response
 - internal communication delays

Measurements: One-Way Network Delays

■ Measurement of one-way network delays

- challenge: time synchronization
- measurement concept
 - all systems get GPS pulse per second (PPS) of high precision (e.g., $< 0.5 \mu\text{s}$ deviation from UTC), kernel timestamps with CPU clock (e.g., with TSC, ns resolution), correction by PPS information, can achieve μs precision
 - hardware requirements: roof-mounted GPS antenna, GPS receiver card, channel for the PPS (e.g., serial port), NTP server and additional network for coarse time synchronization and data collection
 - software requirements: Linux, PPS API kernel patch, own patch for netfilter framework
- has been applied for Web cluster, operating IT infrastructures, WLANs (not embedded)
- and for WLAN-connected soccer robots, PPS signal is transmitted over proprietary radio channel (embedded)

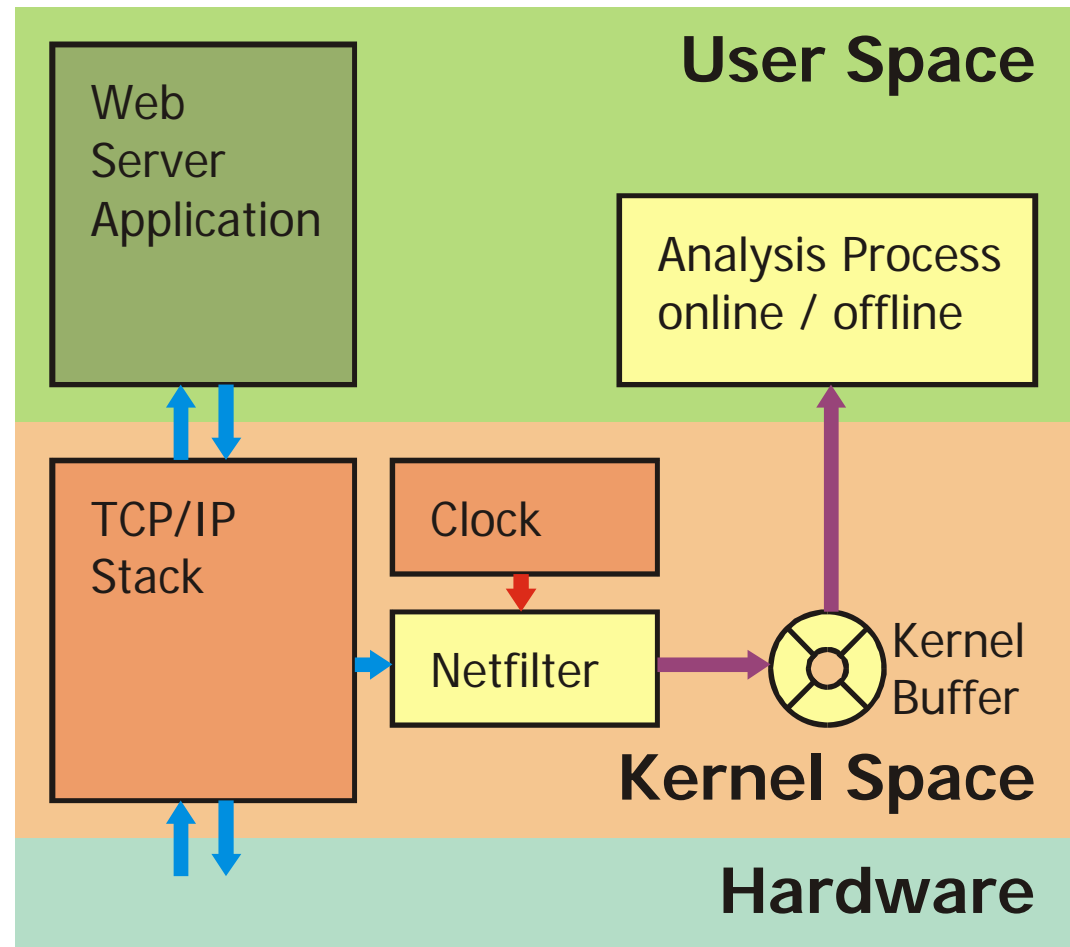
Measurement infrastructure
for Web cluster:



Measurements: One-Way Network Delays

Object system architecture

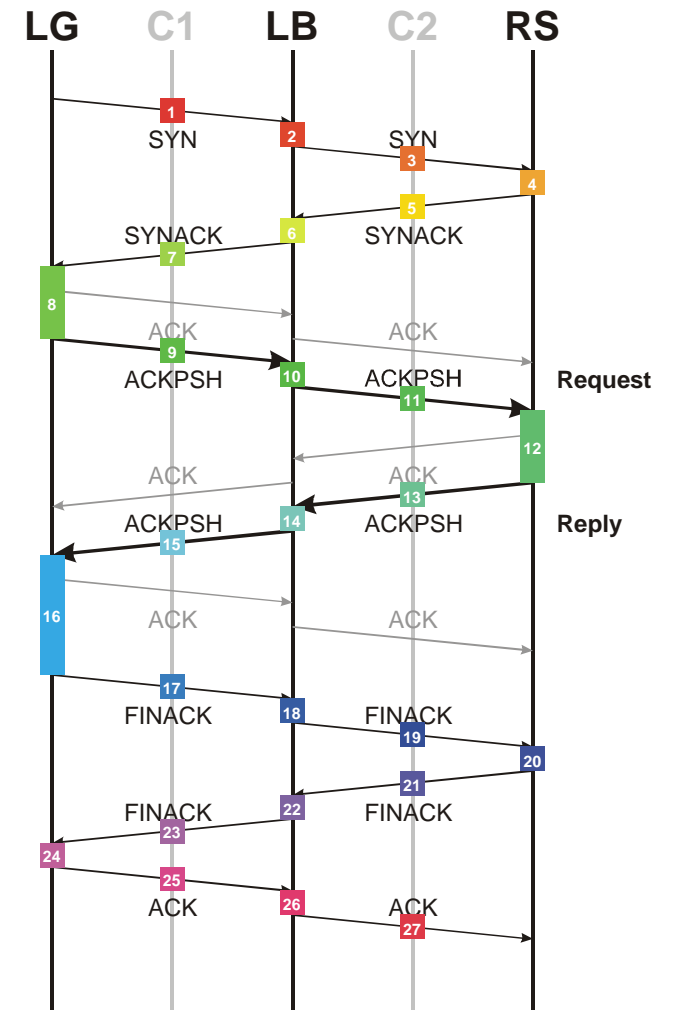
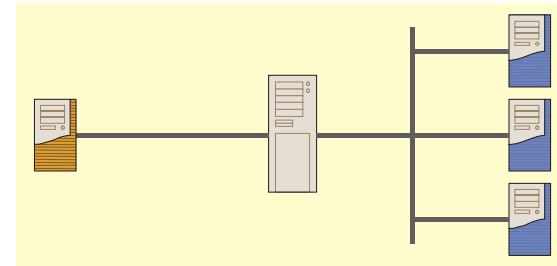
- received packets get timestamp of clock (ns)
- PPS received over different channel (0.5 μ s)
- transfer via ring buffer to user process for data collection and analysis
- **online** synchronization: PPS is used to adjust clock frequency
- or **offline**: timestamp with unsynchronized clocks, correct with PPS afterwards

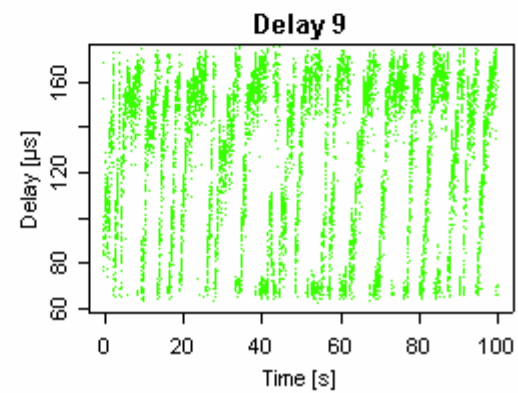
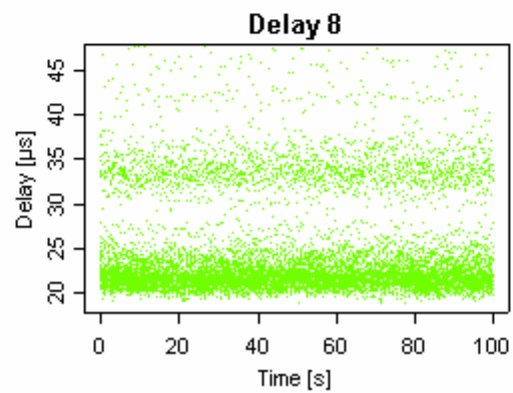
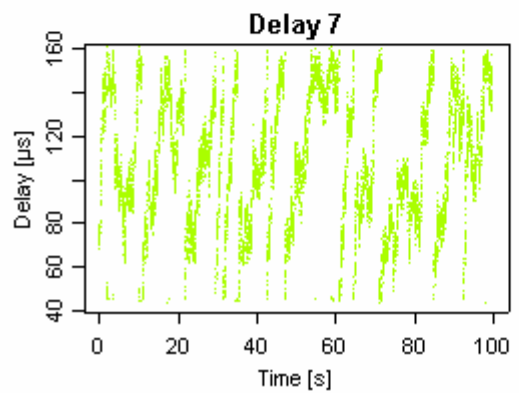
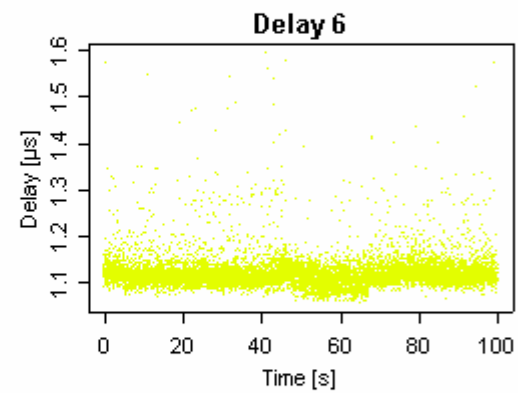
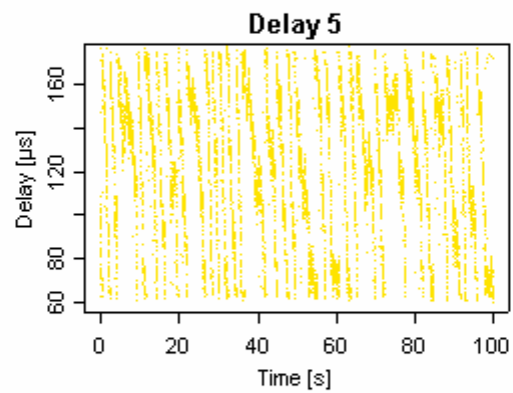
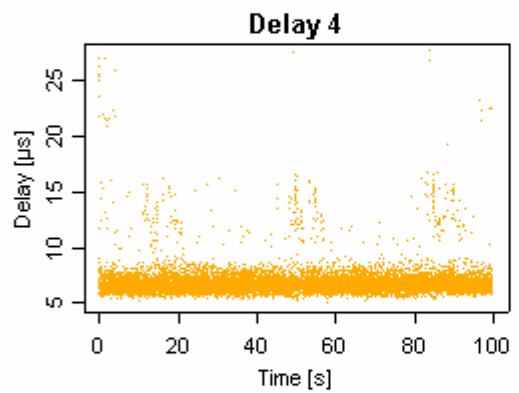
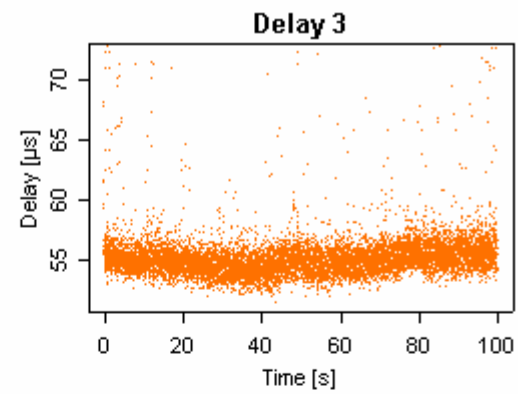
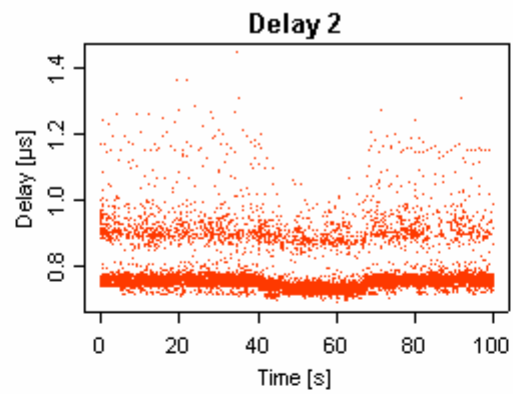
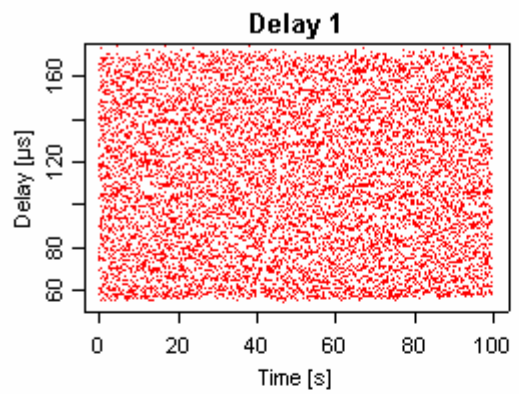


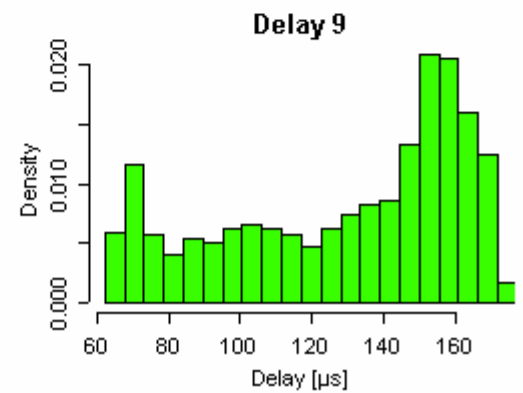
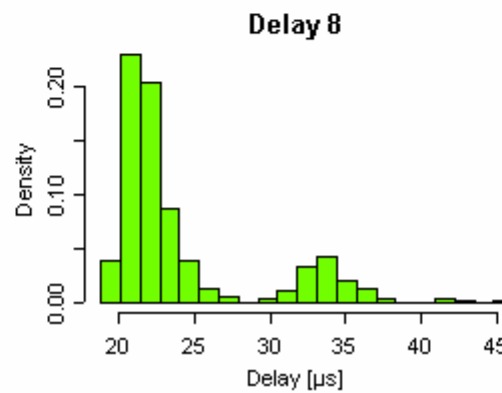
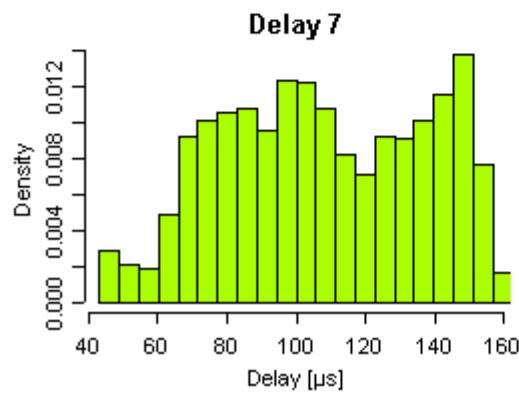
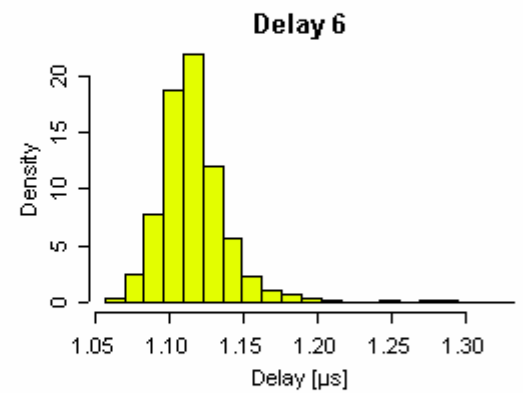
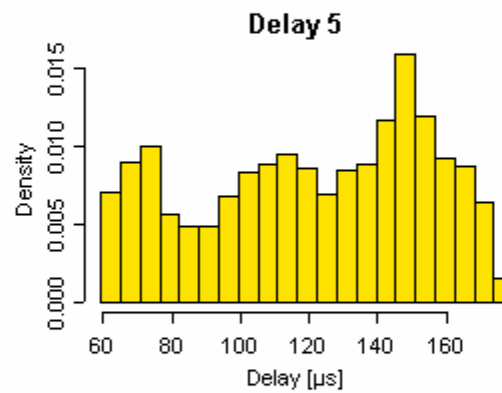
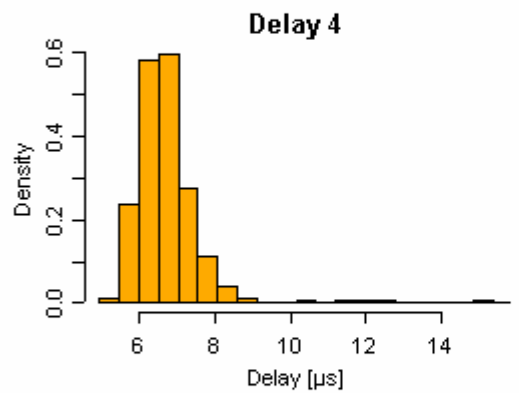
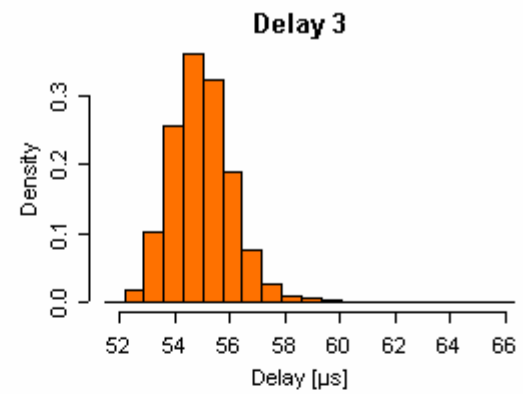
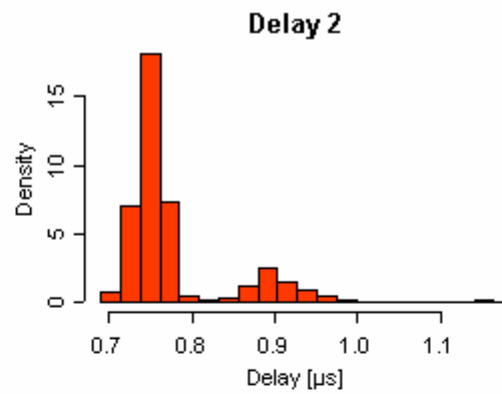
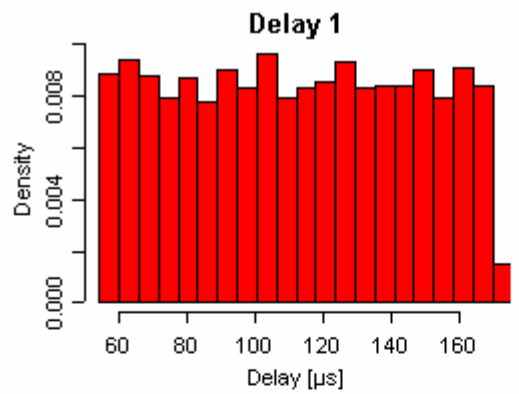
Measurements: OW Network Delays

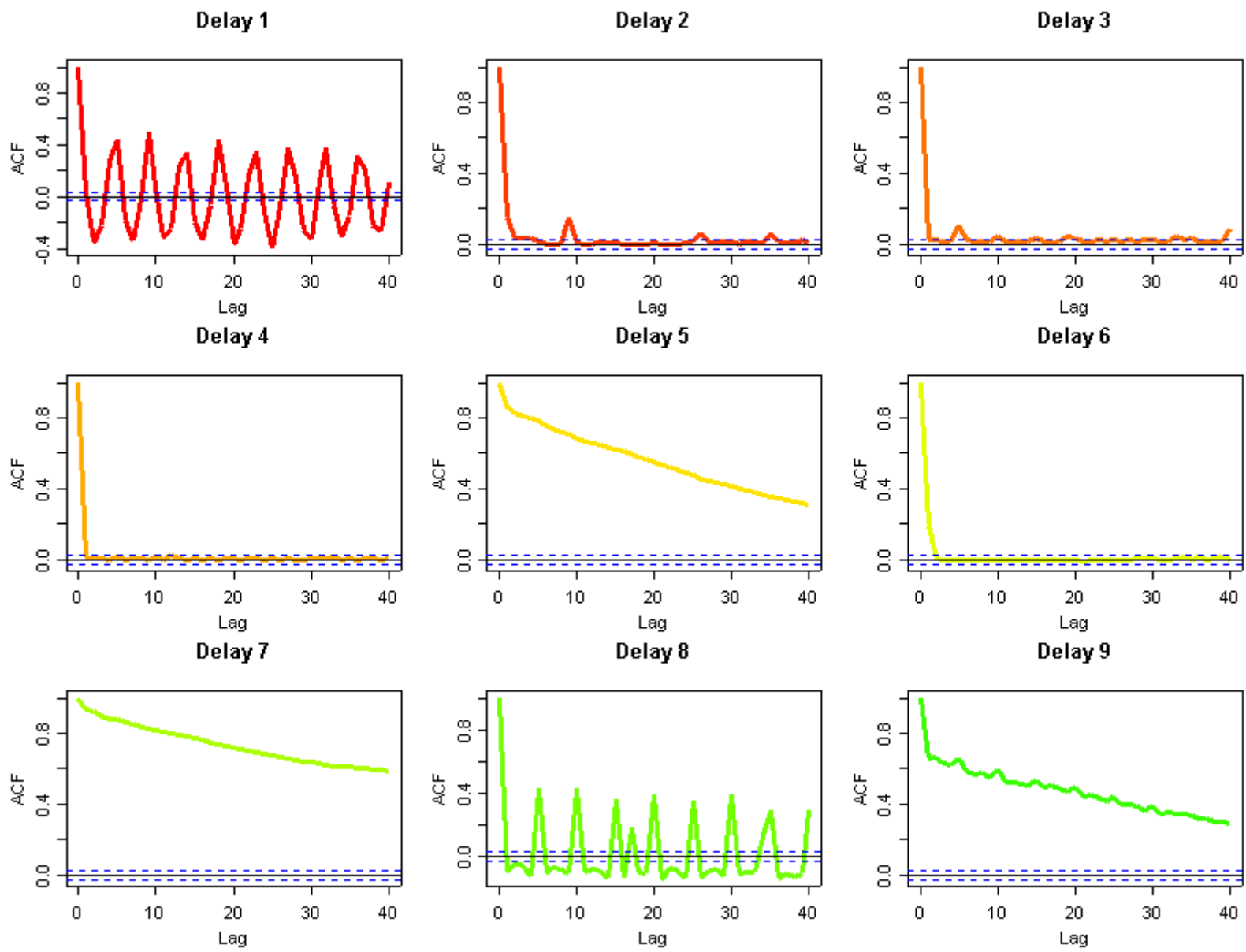
Some typical results

- load generator (LG), load balancer (LB), real servers (RS)
- successive TCP-connections using HTTPperf between LG and RS for HTTP request and response: 27 single delays per connection
- traces, histograms, correlation plots:





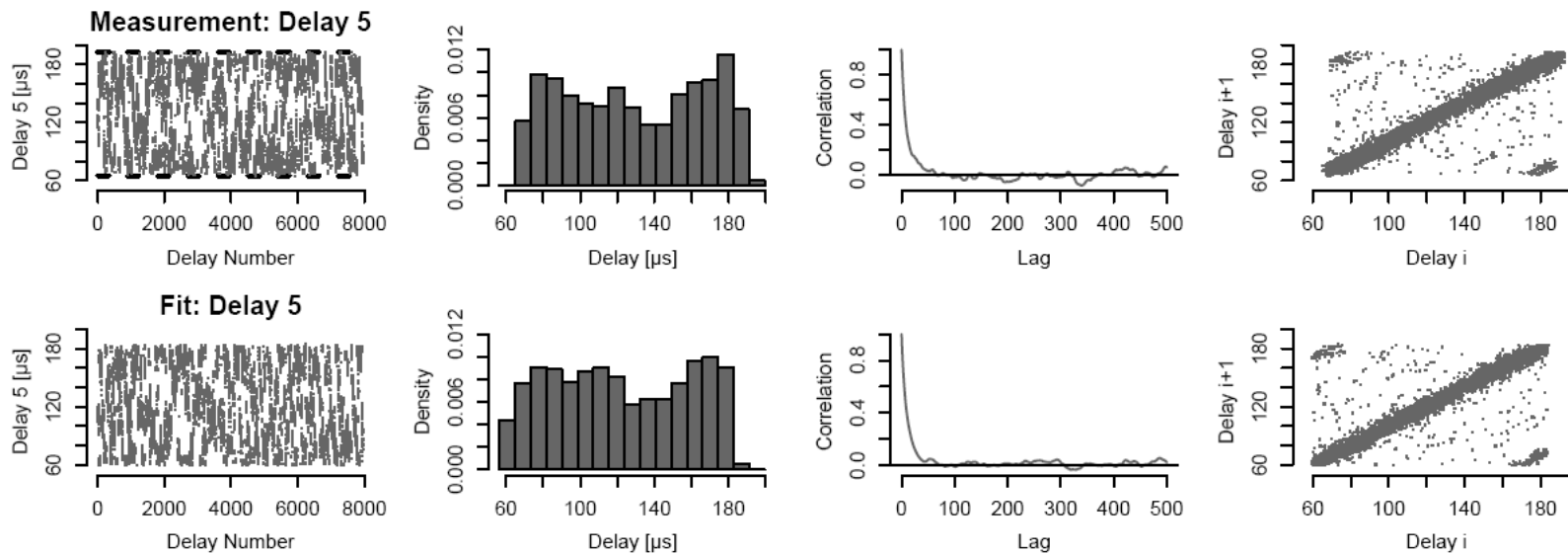




Measurements: One-Way Network Delays

■ Fitting for input modeling

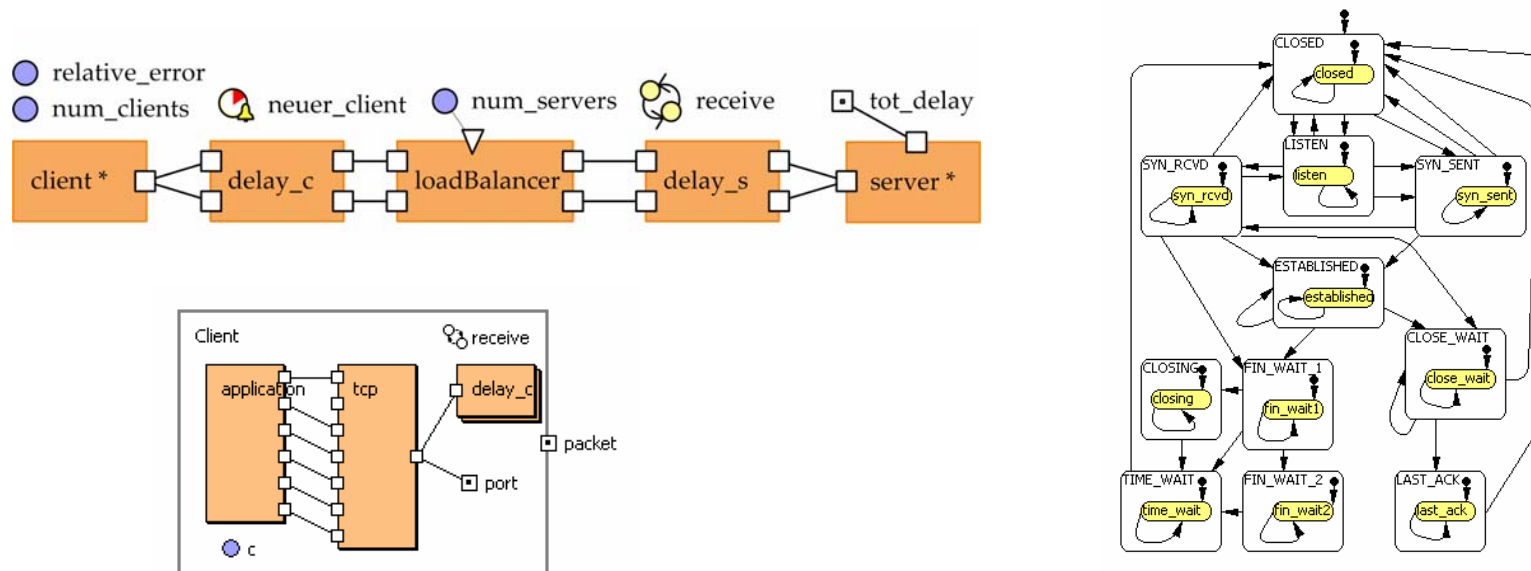
- stochastic models for the single delays: theoretical distributions, mixed multimodal distributions, phased delays (on-off etc.), Bézier distributions
- and a model for autocorrelated delays: sampling of differences instead of the delays directly
- e.g., delay 5 (SYNACK RS → LB):



Measurements: One-Way Network Delays

UML Simulation model

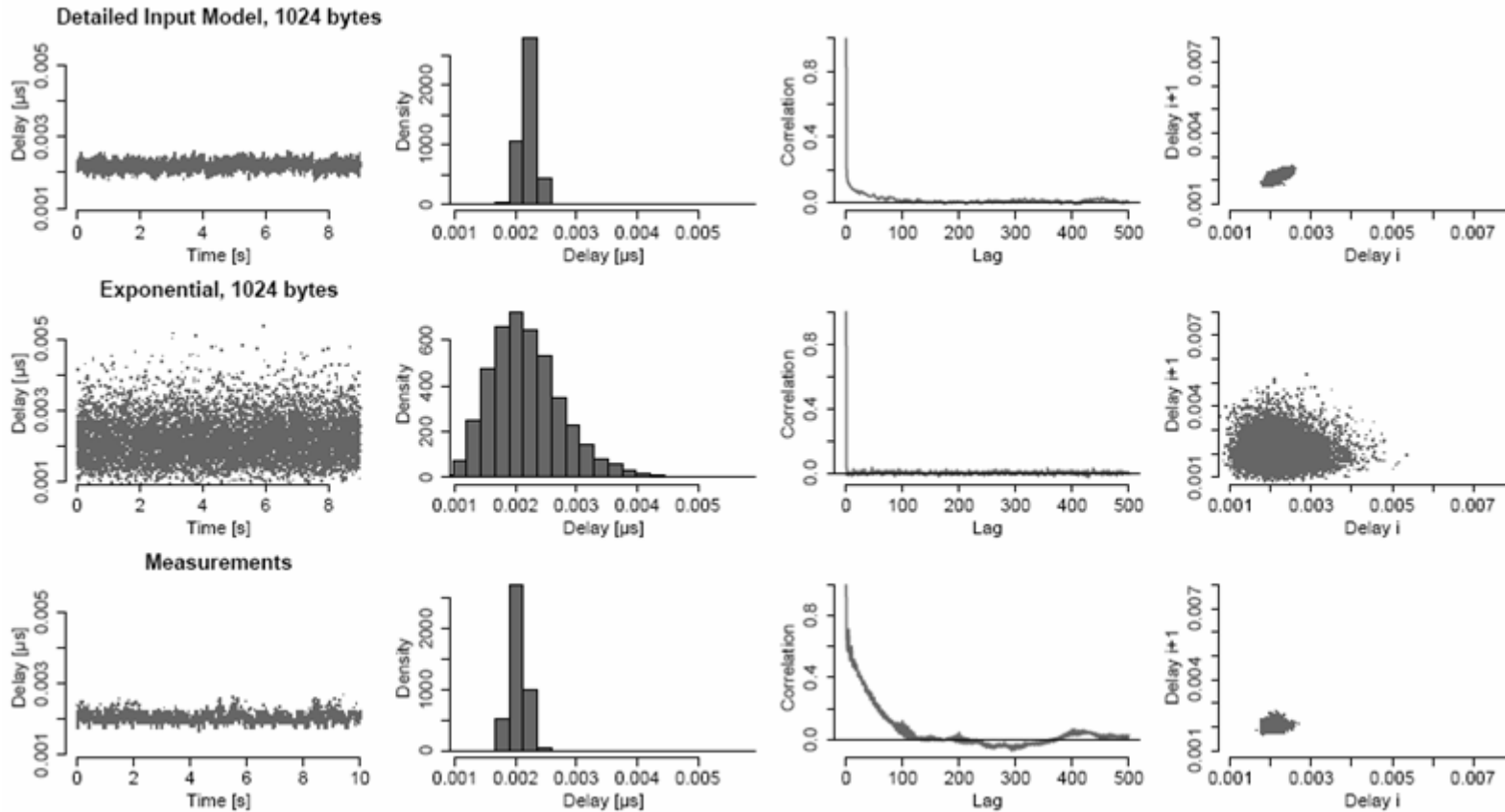
- we have built a detailed simulation model which includes
 - TCP protocol mechanisms (RFCs 793, 1122, 2581, 2988)
 - channel characteristics: signal delays, transmission delays
 - processing times, overhead of operating system
- the model has been coded with Anylogic which supports a proprietary variant of UML-RT with **communicating state machines**: message passing objects with internal statecharts



Measurements: One-Way Network Delays

■ Comparison of end-to-end-delays

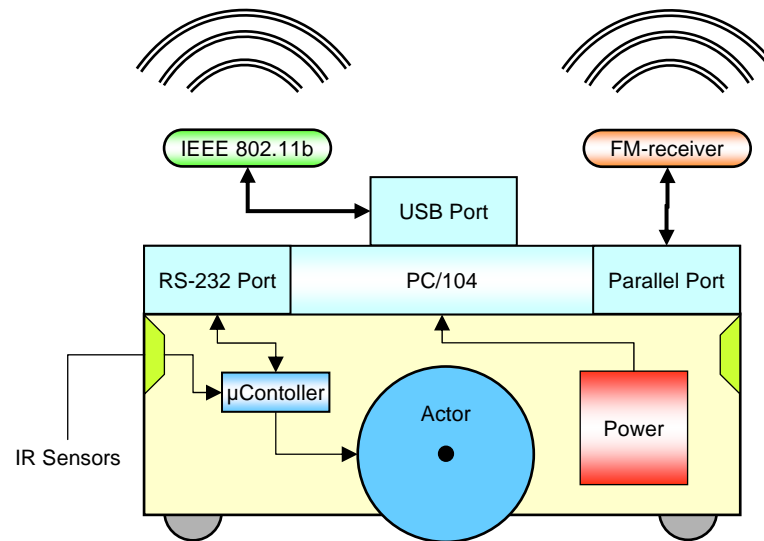
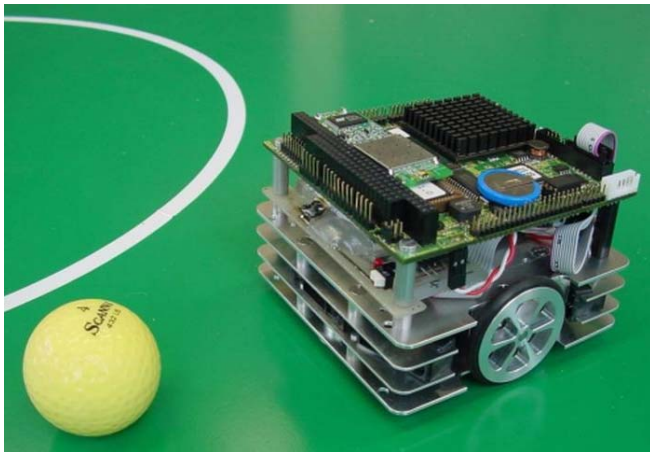
- 1) model with detailed input models vs.
- 2) model with exponential distribution with same mean vs.
- 3) measurements



Measurements: External Interrupt Response

■ Soccer robot scenario

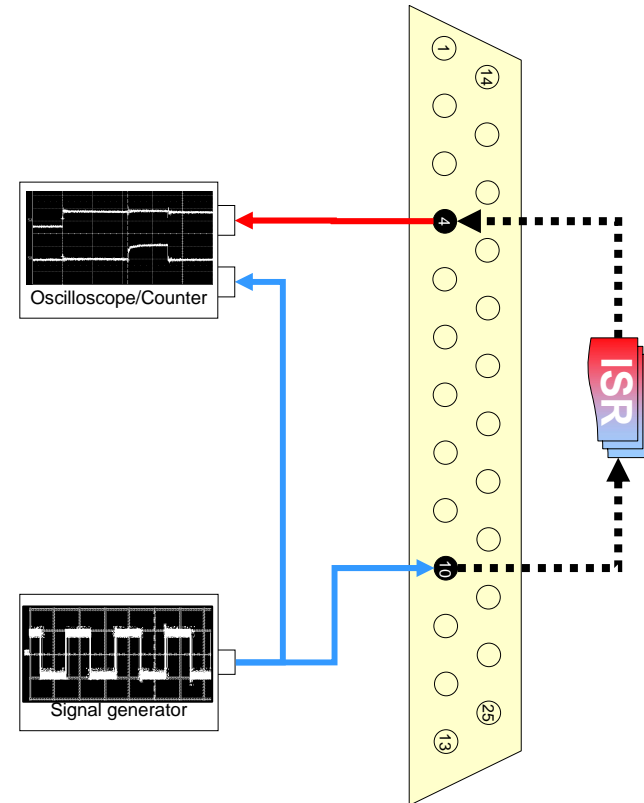
- benchmark for studying the behavior of a networked embedded system
- Yannik robot: own design, we can access all internals
- PC/104, Linux, RTAI, μ C for actors and sensors, IR sensors, WLAN
- one-way delays again with PPS, over radio channel (PHY challenges)
- further question: how fast can the system react to, e.g., signal of the IR sensors under the possible set of OS scheduling strategies (e.g., one-shot vs. periodic modes)



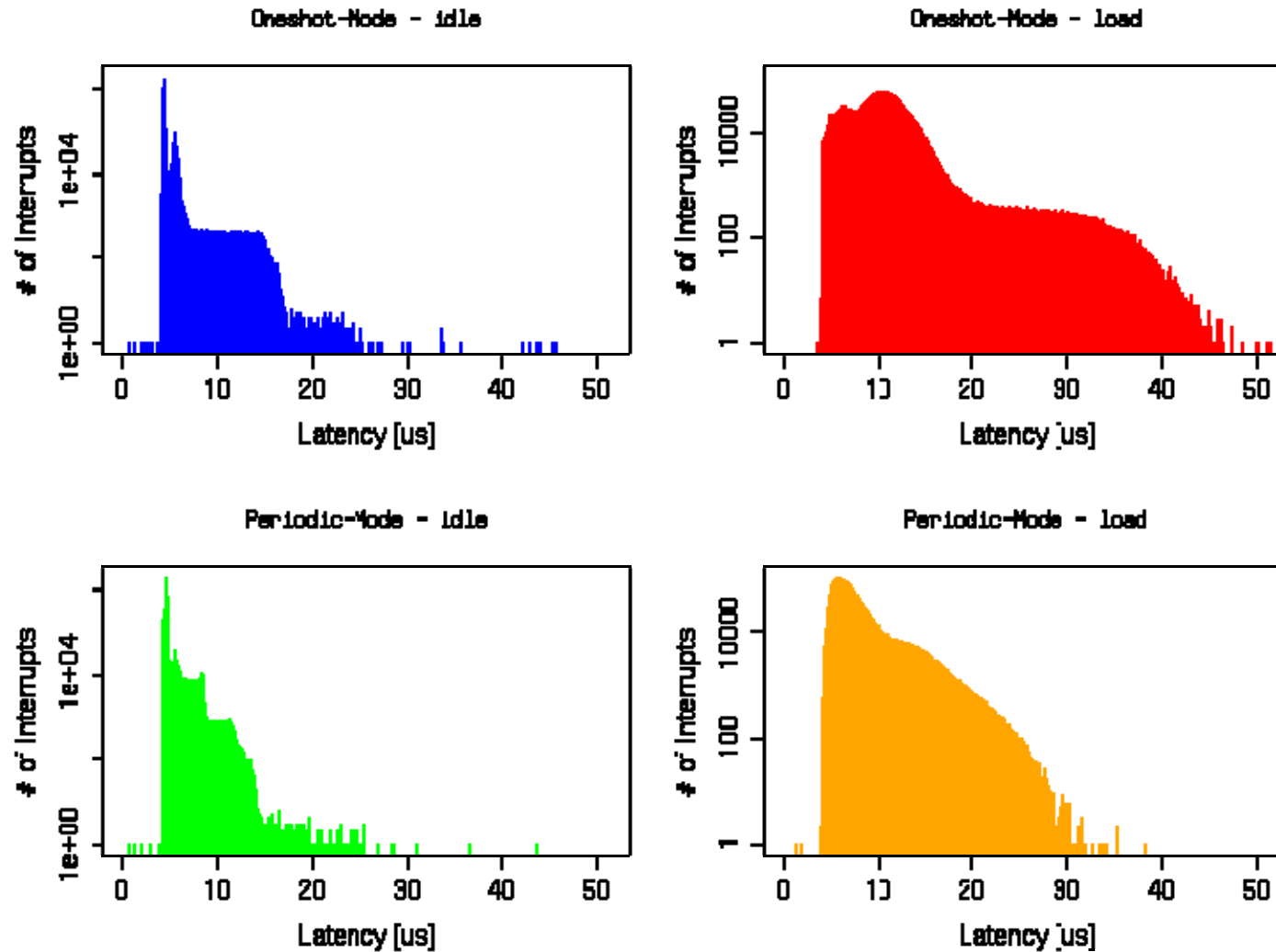
Measurements: External Interrupt Response

■ Measurement Infrastructure

- input rectangle signal (up to 100 KHz) from signal generator to the parallel port
- high level causes high-priority interrupt on input port
- interrupt service routine (ISR) to respond to the interrupt
- monitor system response time using an oscilloscope or a counter with memory (here with 20 ns precision)
- non-intrusive
- oscilloscope good for visualizations and first investigations
- counter allows more sophisticated statistical analyses
- shows significant differences of schedulers when non-real-time system is loaded with ping flood:



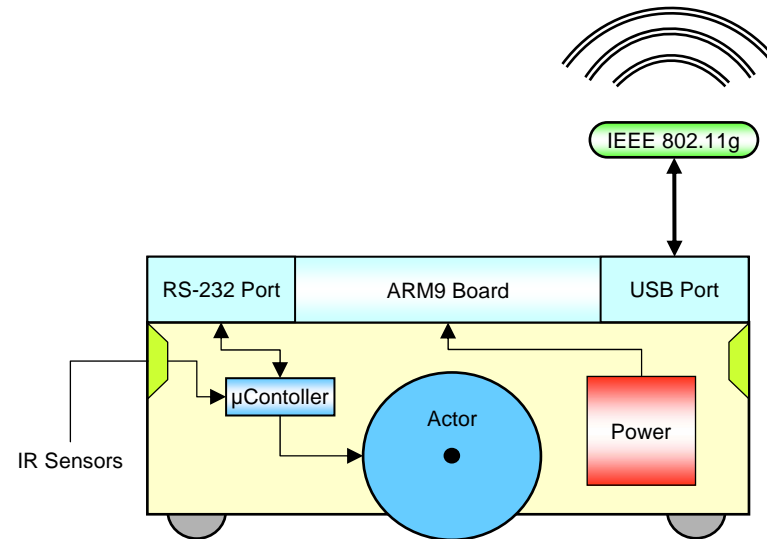
Measurements: External Interrupt Response



Measurements: Internal Communication Delays

■ Soccer robot scenario II

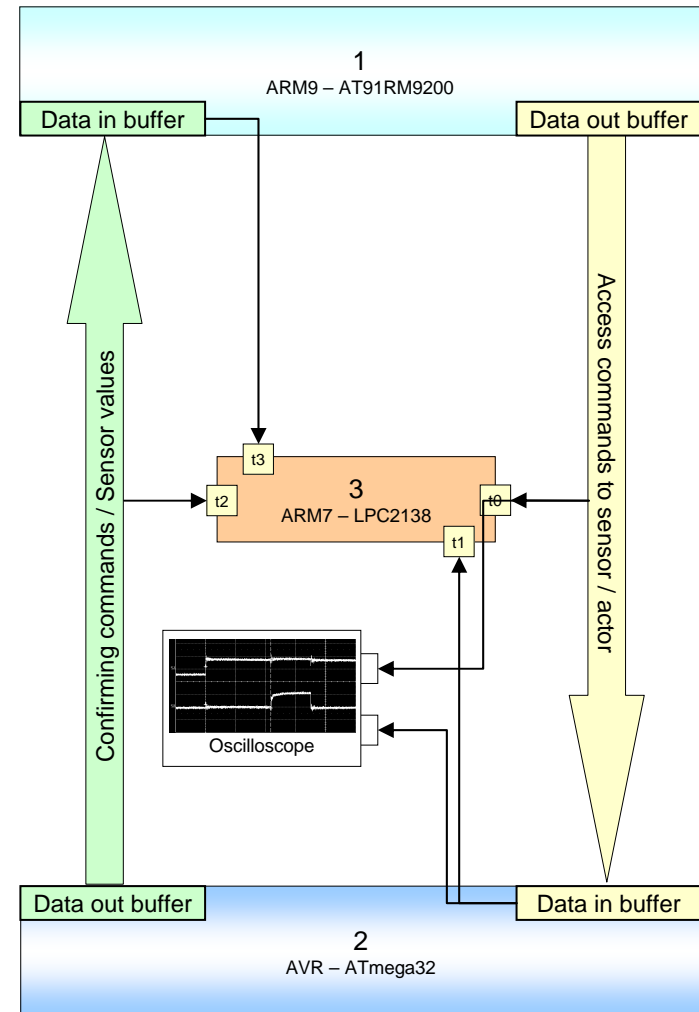
- Buggy robot: improved design, truly embedded
- ARM9, Gentoo-Linux on SD-Card, serial, USB ports
- question: how fast is the communication between the processor and the μ C, especially when the μ C is busy with access to actors/sensors?



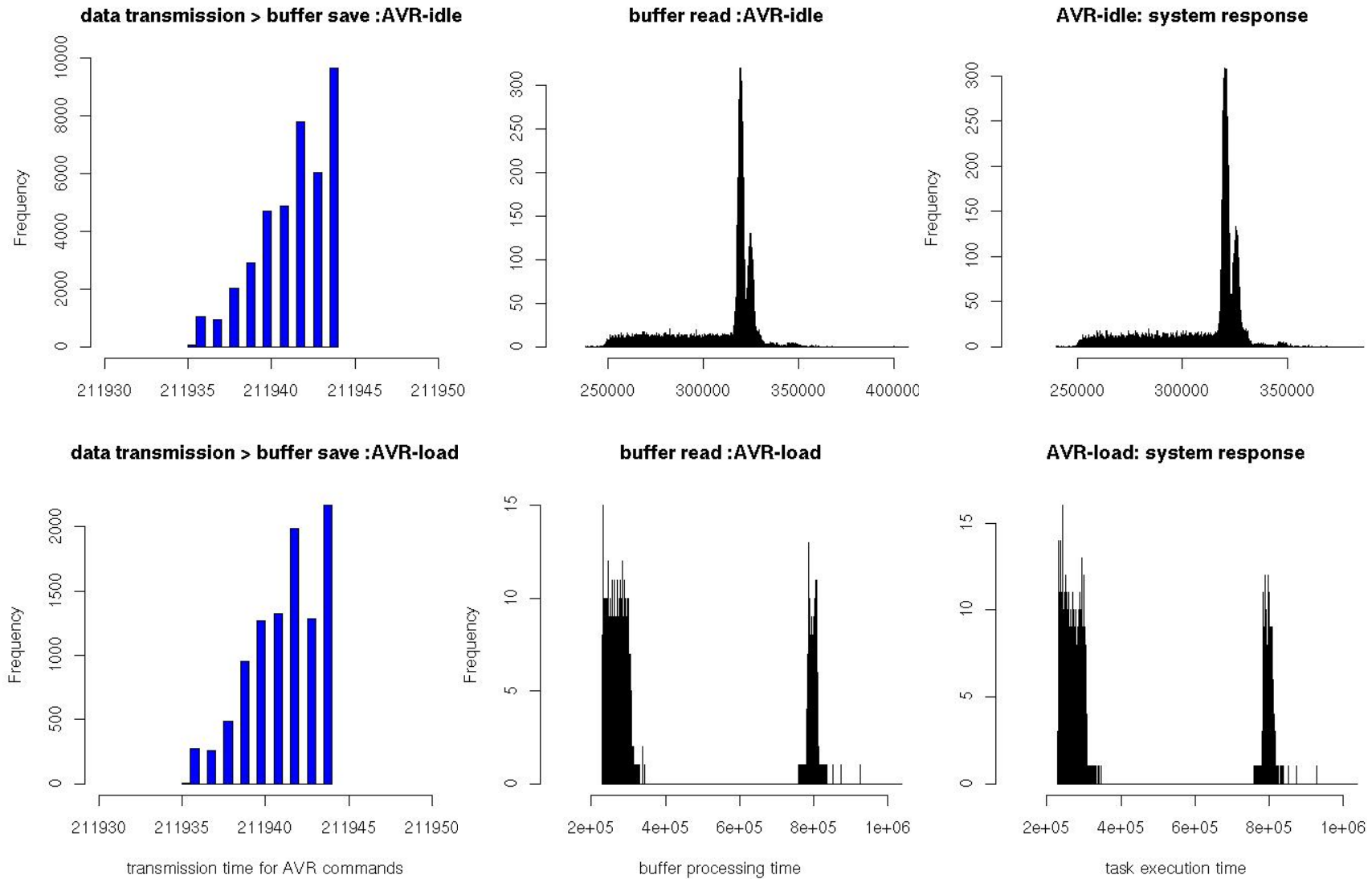
Measurements: Internal Communication Delays

■ Measurement infrastructure

- dedicated microcontroller based probe unit (ARM7) for recording of delays
- events are signaled on RS-232 pin
- non-intrusive
- oscilloscope for visualization of communication between layers
- internal 58,9824 MHz clock -> 17 ns resolution
- shows significant differences when microcontroller is highly loaded:



Measurements: Internal Communication Delays



Measurements

■ Conclusions for measurements

- have shown 3 measurement approaches
 - one-way network delays → concept based on PPS
 - external interrupt response → counter + ISR
 - internal communication delays → dedicated unit
- all illustrate statistical character of delays
- have also shown model for Web cluster
- we develop a similar model for the soccer robot scenario and can then fit input models, calibrate, validate, ...
- significant technical challenges for measurements
- not always acknowledged by modelers

Contents

■ Measurements

- joint work with Kai-Steffen Hielscher, Kemal Köker

■ UML-Based Simulation and Test

- joint work with Isabel Dietrich, Matthias Beyer (Fraunhofer IIS)

■ 2 Automotive Applications

- joint work with Thomas Herpel (INI.FAU)
- and with Thorsten Frunzke (Audi Electronic Ventures)

UML-Based Simulation and Test

■ UML-based modeling

- in cooperation with Fraunhofer Institute for Integrated Systems (IIS)
- UML a major approach, both in the IT and embedded domain
- system descriptions: often communicating state machines
- system requirements: often sequence diagrams
- documentation, specification, visualization, code generation (mainly)

■ Discrete-event simulation

- communicating state machines also a common modeling concept
- but different tool environments

■ Model-based testing

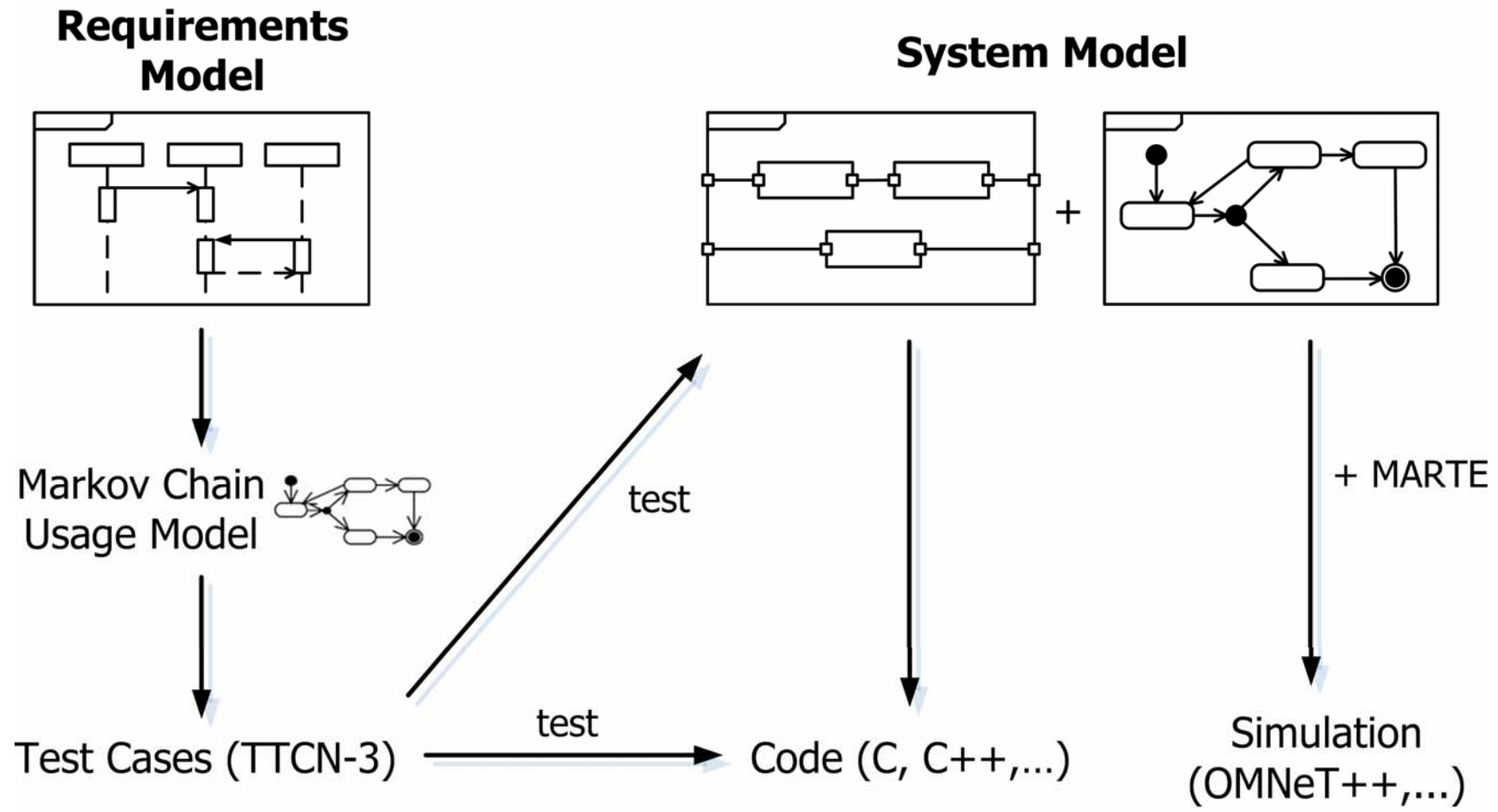
- automatic generation of test cases from system requirements
- can be applied to manually or automatically generated code

■ Integrated approach

- integration of UML-based modeling, code generation, simulation and testing considering existing tools and standards

UML-Based Simulation and Test

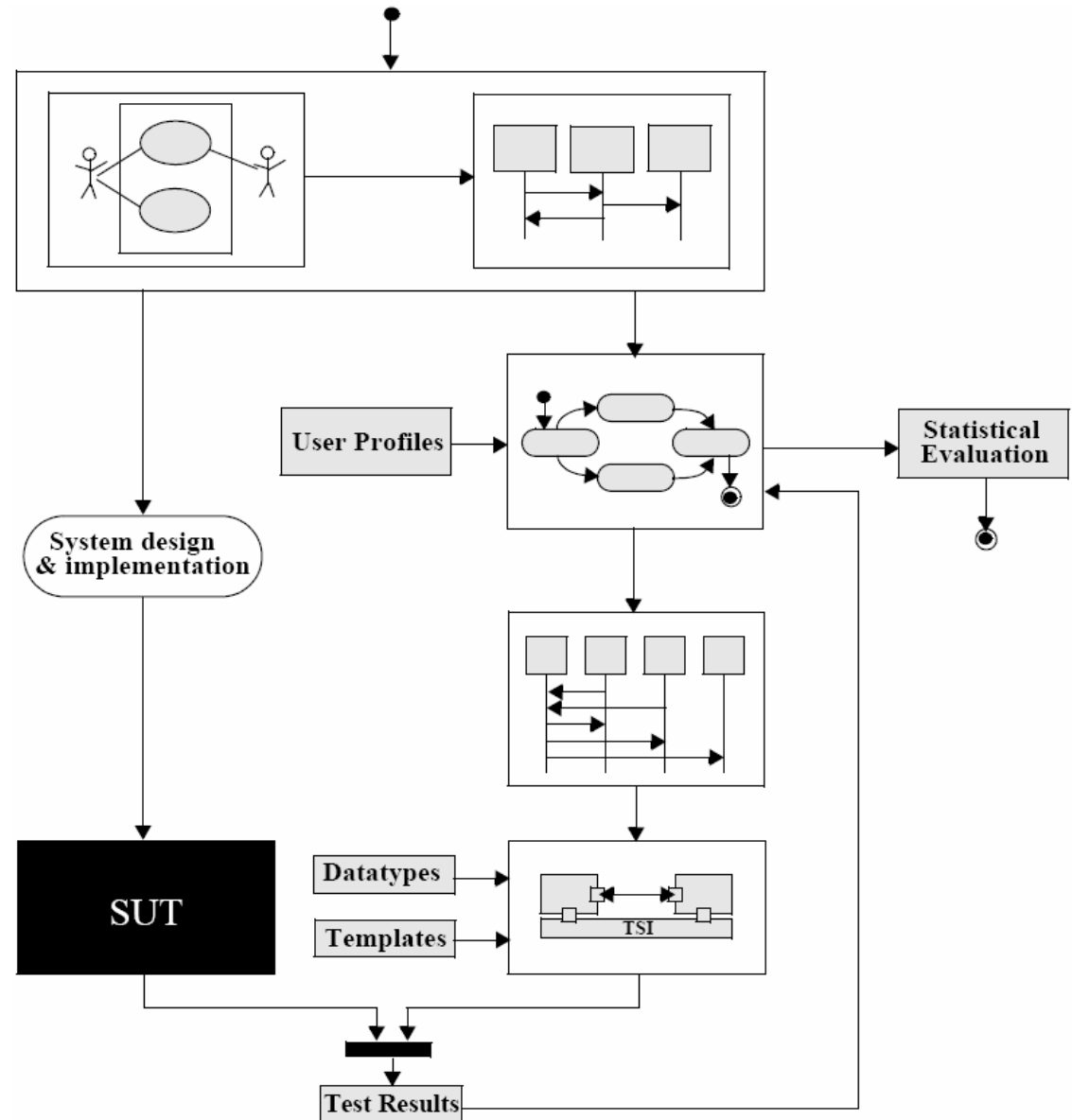
■ Overview



UML-Based Simulation and Test

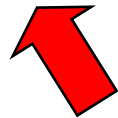
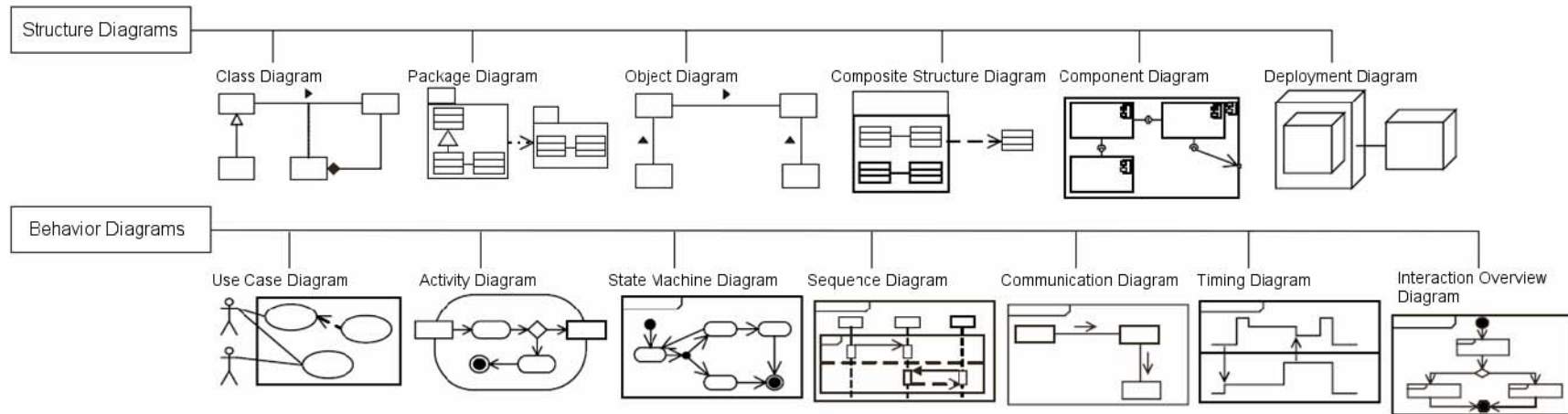
■ Testing approach

- start with sequence diagrams as system requirements
- combine them to a statemachine diagram
- add frequencies of user profiles
- generate usage-specific test cases as paths through this statemachine, can be visualized as sequence diagrams
- transform to TTCN-3 and execute tests
- a blackbox testing approach



UML-Based Simulation and Test

Employed UML elements:



Profiles

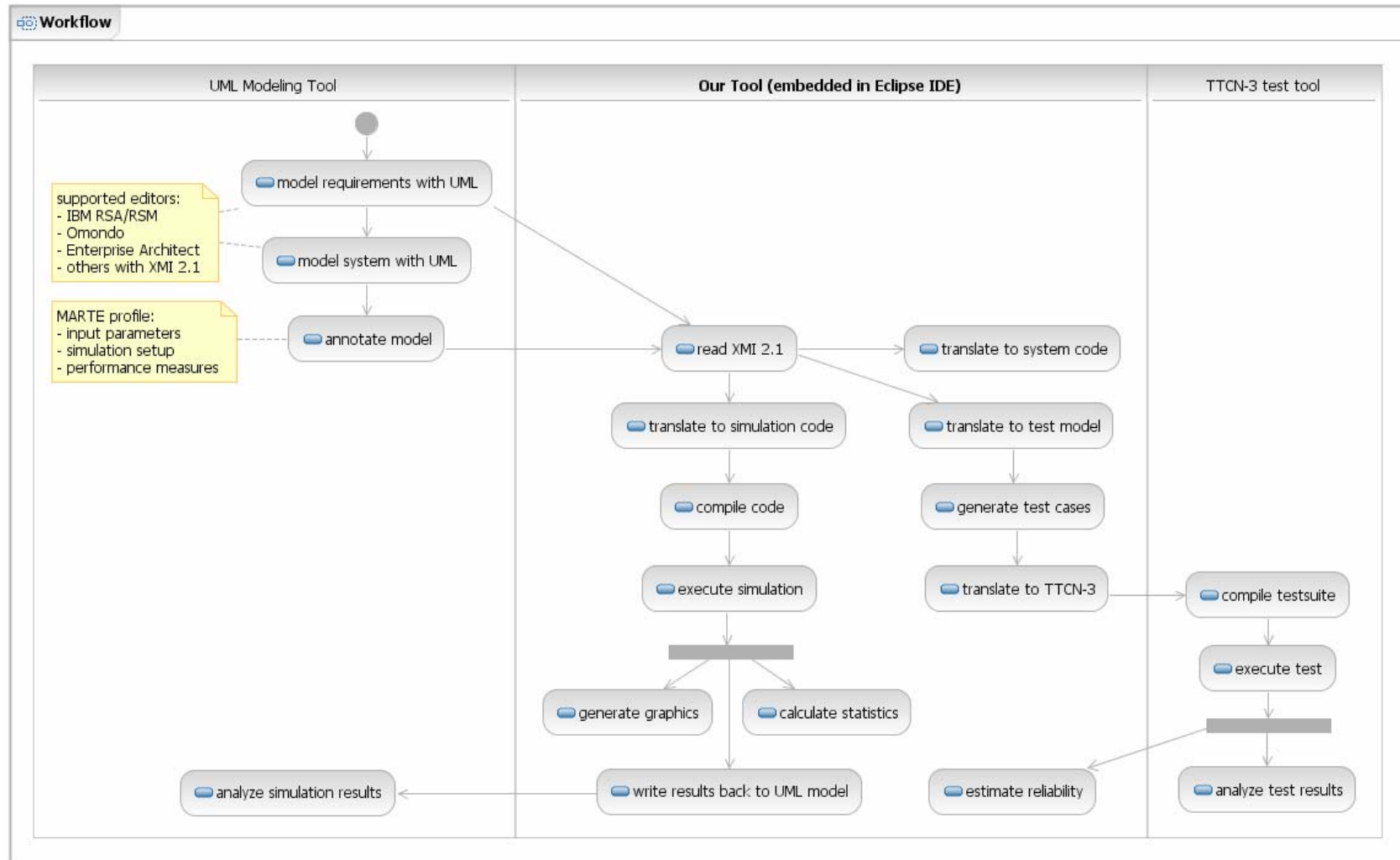
- announced UML Profile for Modeling and Analysis of Real-Time and Embedded Systems (MARTE), successor of the UML profile for Schedulability, Performance and Time (SPT)
- used to annotate the UML system model with time consumption, resource usage, ...

UML-Based Simulation and Test

■ Realization aspects

- UML modeling
 - editors with XMI 2.1 export
- code generation
 - UML model consisting of composite structure, statemachine, activity diagrams is transformed to C++
- simulation
 - from code direct transformation to OMNeT++
 - other simulation engines planned
- test
 - from sequence diagrams to TTCN-3
- user interface, visualization
 - Eclipse, XMI 2.1
 - workflow between tools:

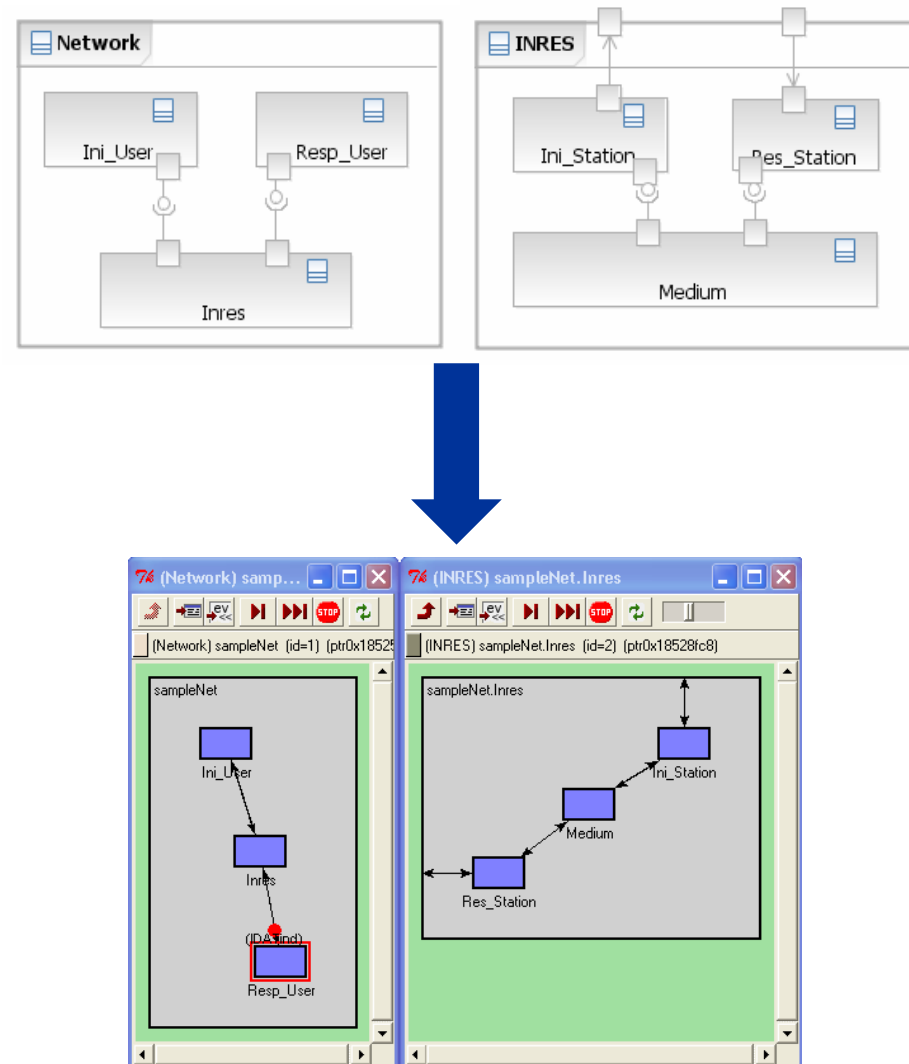
UML-Based Simulation and Test



UML-Based Simulation and Test

■ Simulation with OMNeT++

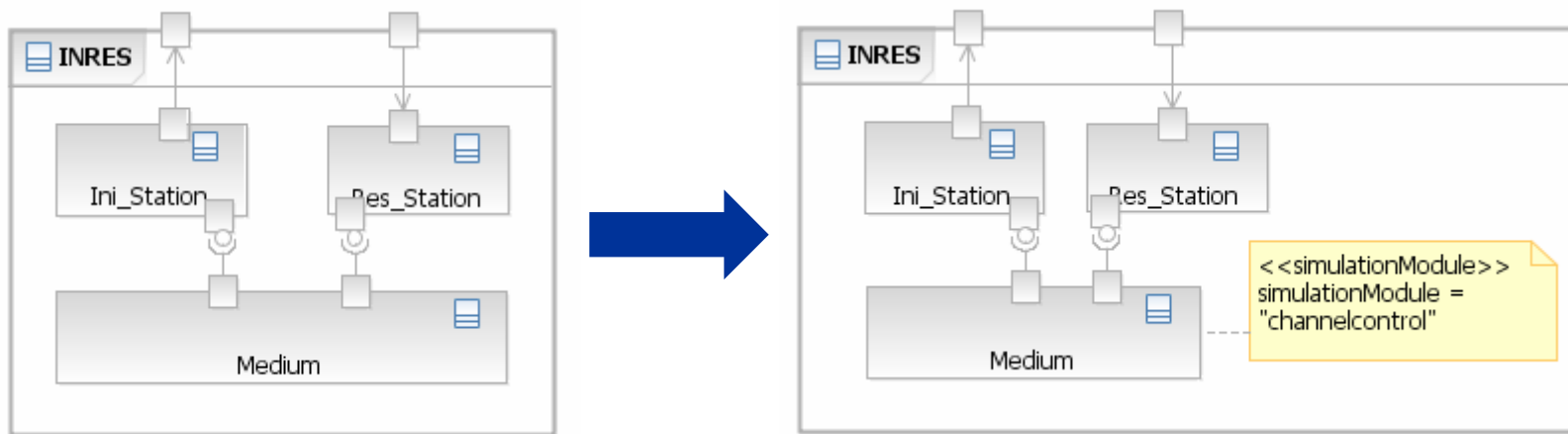
- OMNeT++ simulation code compiles and runs on Linux and Windows
- GUI for visualization and debugging
- commandline user interface for production simulation runs
- statistical evaluation possible



UML-Based Simulation and Test

■ Inclusion of external models

- e.g., libraries in OMNeT++
- can be included in the UML models via stereotypes
- example: model for a lossless medium is replaced by an OMNeT++ model for a wireless channel



UML-Based Simulation and Test, Case Study

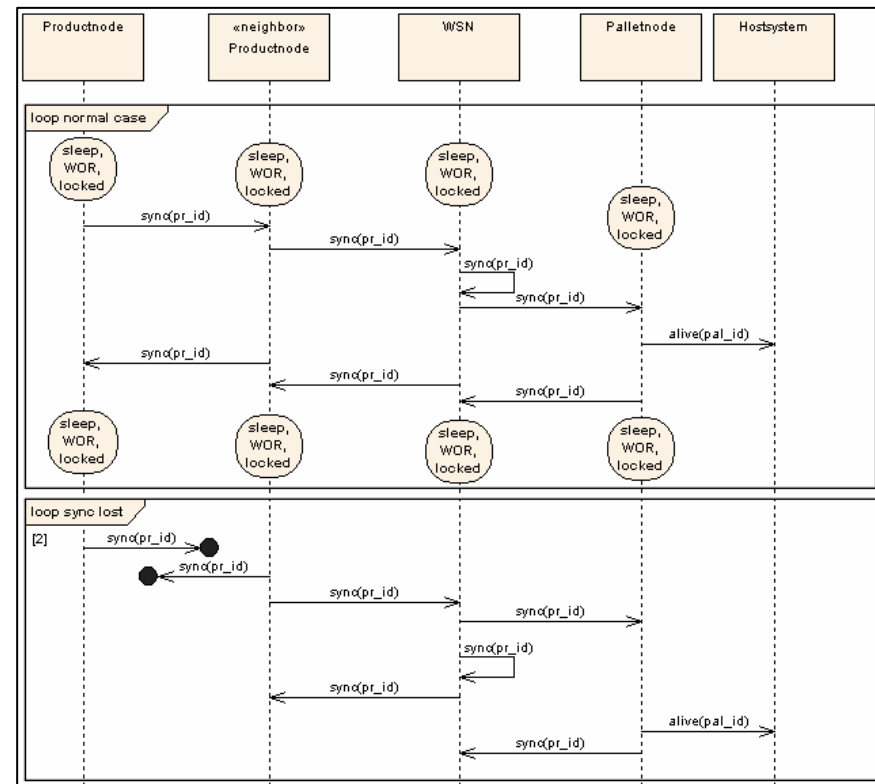
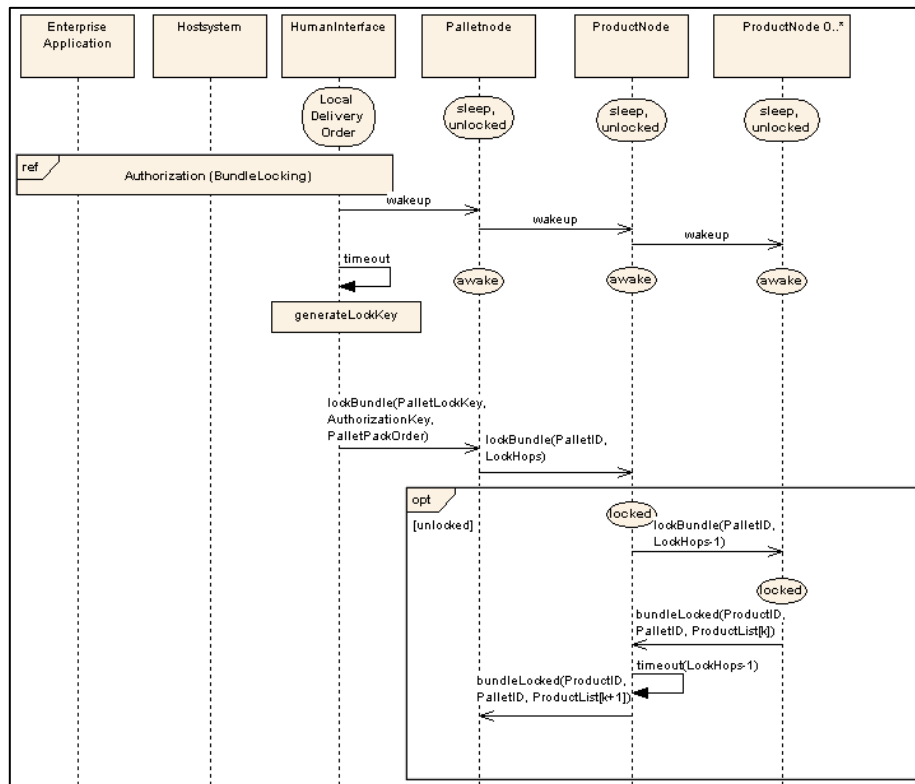
■ Anti-theft protection in logistics

- for illustration purposes, we take an example from an FhG research project in logistics
- sensor node platform with SW has been developed by IIS, extended duty cycles, wakeup possible, can measure distance via RSS
- products on a pallet are equipped with sensor nodes
- all product nodes form a network
- if connection to a product node is lost an alarm is raised



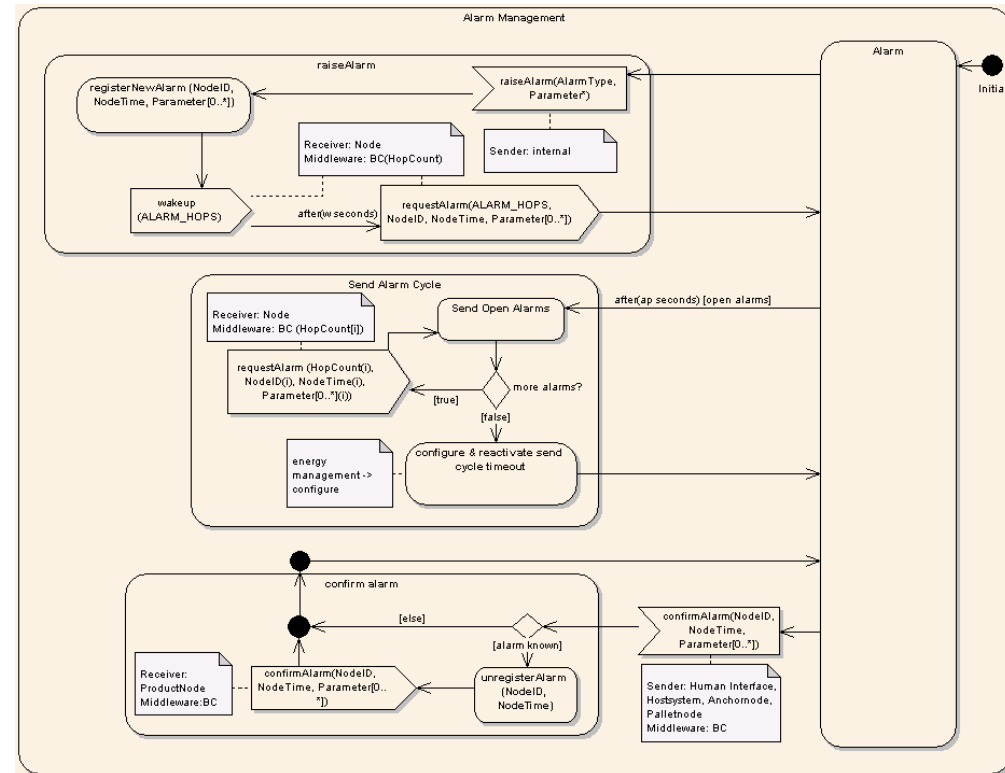
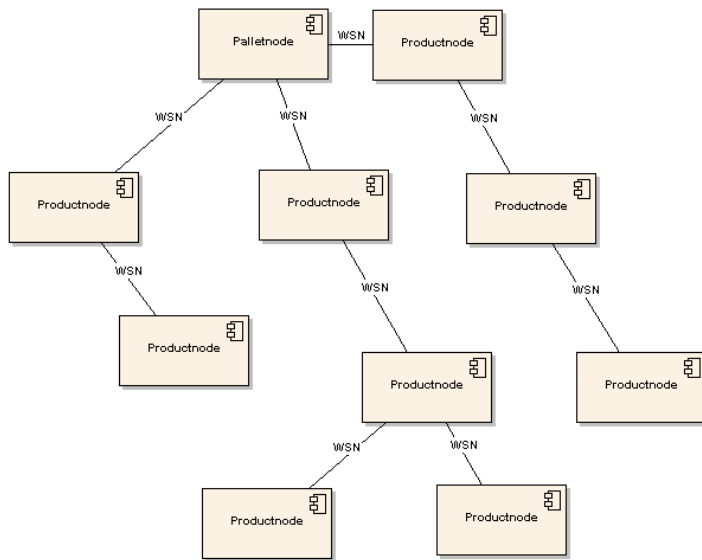
UML-Based Simulation and Test, Case Study

■ System Requirements (partially):



UML-Based Simulation and Test, Case Study

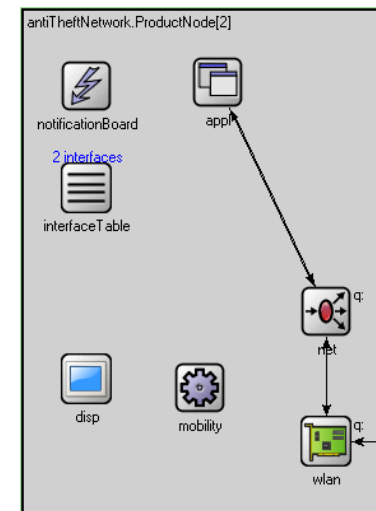
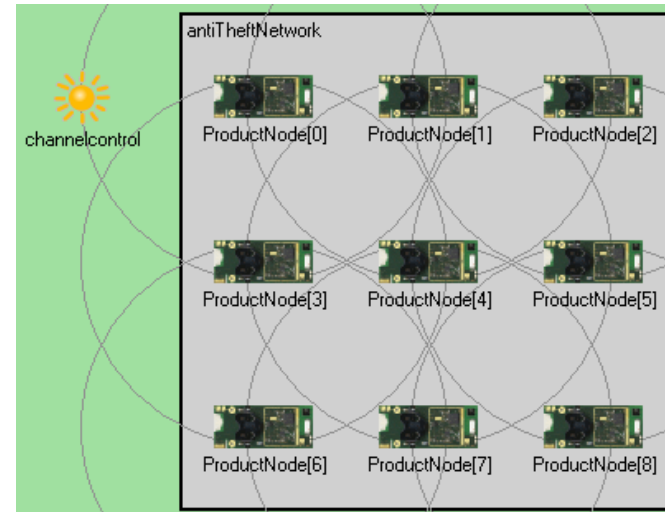
System model (partially):



UML-Based Simulation and Test, Case Study

■ Simulation

- network with product nodes of a pallet
- channelcontrol: manages the wireless channel in the simulation
- node behavior modeled on several layers
- simulation can deliver time to detect node theft



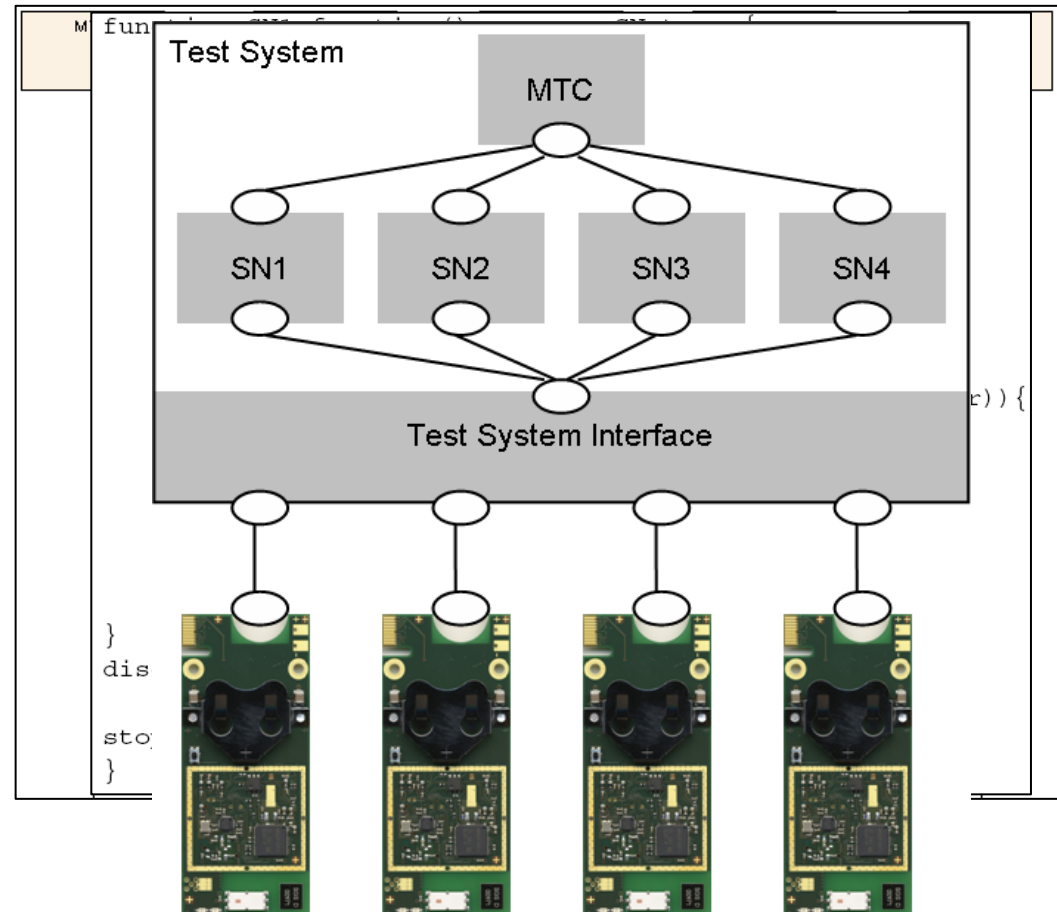
UML-Based Simulation and Test, Case Study

■ Example test case

- checks correct behaviour after alarm recognition
- multi-hop forwarding of an alarm signal

■ Test configuration

- master test component (MTC)
- one parallel test component (PTC) per node



UML-Based Simulation and Test, Case Study

■ Conclusions for UML-Based Simulation and Test

- integration of different system engineering activities: modeling, code generation, discrete-event simulation, test automation
- the tool works for first examples
- due to many incompatibilities (e.g., vendor-specific XMI) this is not so easy
- not a rigid approach, semantic issues
- gap between UML modeling and simulation is in any case unnecessary
- can be extended by various other functions (e.g., Markovian analysis, non-Markovian analysis, aggregation and approximation methods, real-time analysis, animation, ...)

Contents

■ Measurements

- joint work with Kai-Steffen Hielscher, Kemal Köker

■ UML-Based Simulation and Test

- joint work with Isabel Dietrich, Matthias Beyer (Fraunhofer IIS)

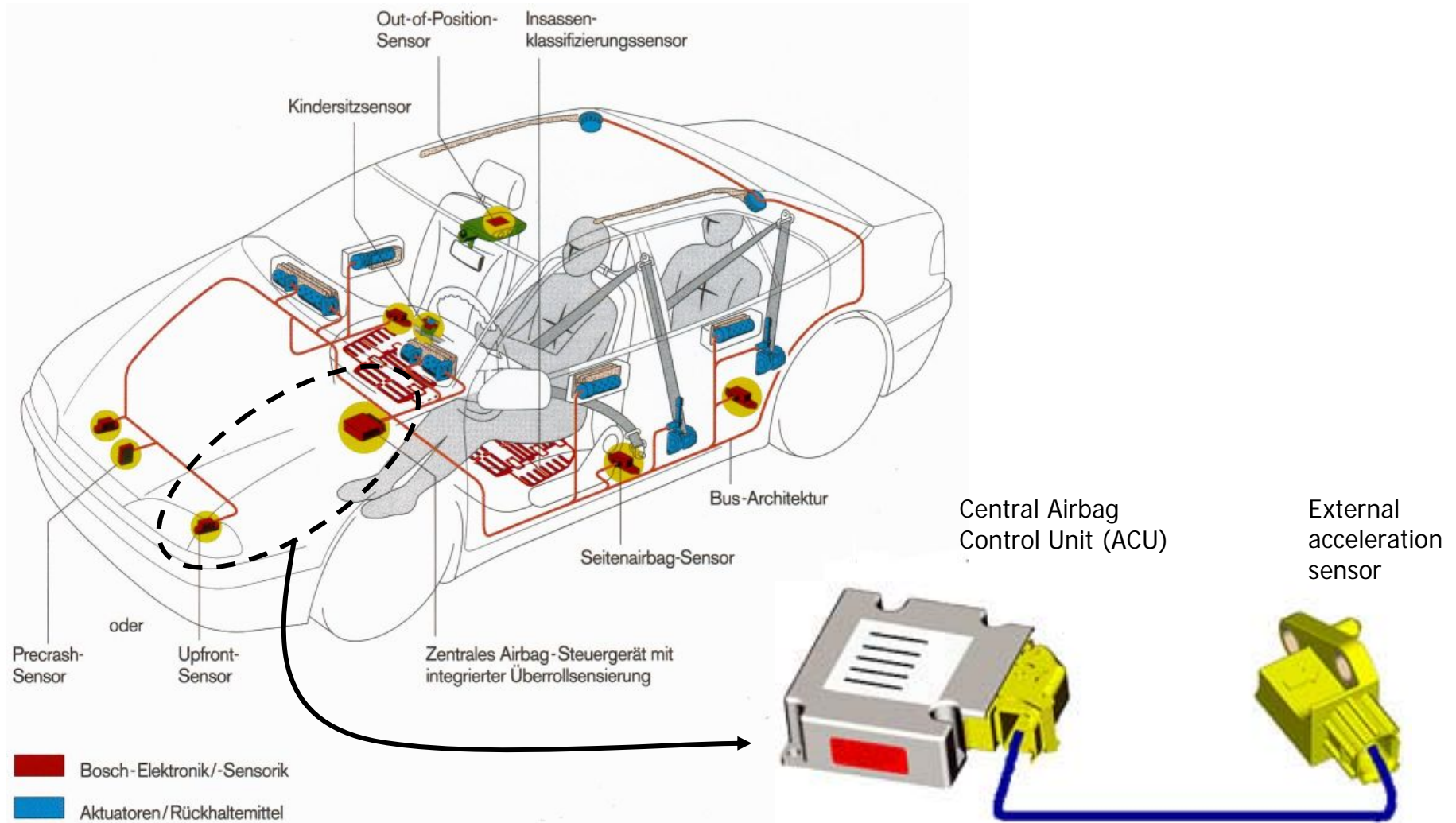
■ 2 Automotive Applications

- joint work with Thomas Herpel (INI.FAU)
- and with Thorsten Frunzke (Audi Electronic Ventures)

Transmission of Safety-Relevant Sensor Data

- Transmission of safety-relevant sensor data
 - in cooperation with Audi, Safety Electronics
 - PhD project of Thomas Herpel (INI.FAU, joint institute of Audi and our university)
- Example: airbag control system (ACU)
 - currently
 - central control unit, external crash sensors
 - plausibility check of firing decision by internal acceleration sensors and redundant microcontrollers
 - future extensions
 - networking with active safety electronics
 - information from other sensors, central sensor cluster
 - more actuators (pedestrian protection, driving assistance systems)
 - need to assess reliability, performance, timeliness

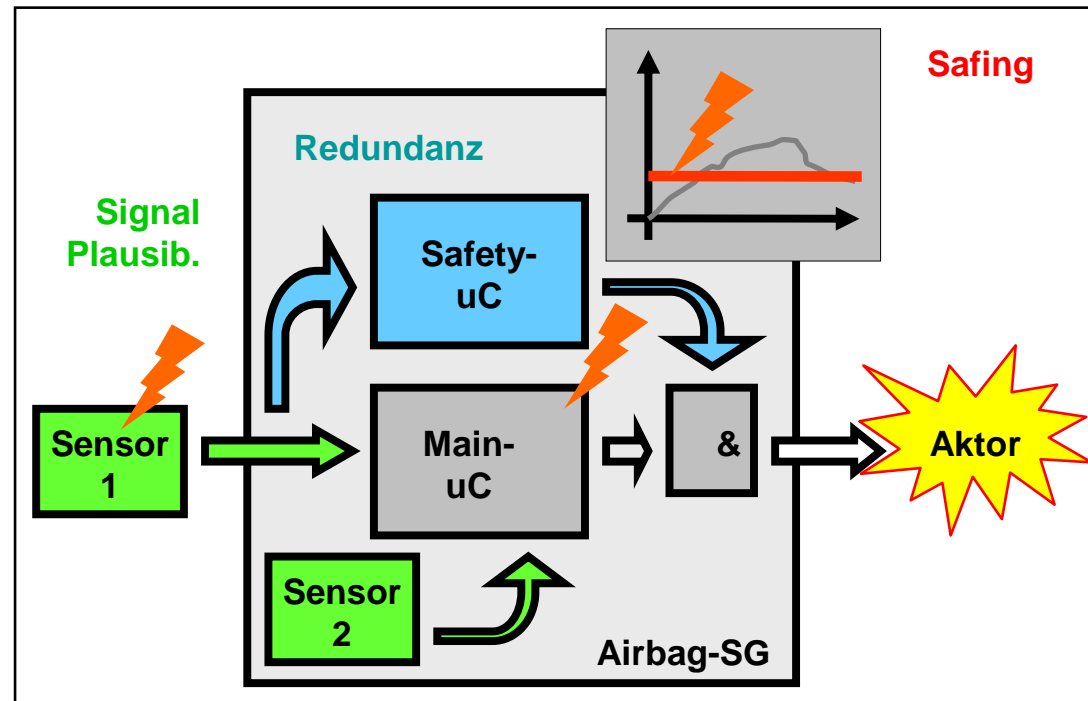
Transmission of Safety-Relevant Sensor Data: ACU



Transmission of Safety-Relevant Sensor Data: ACU

■ Concepts for misuse prevention

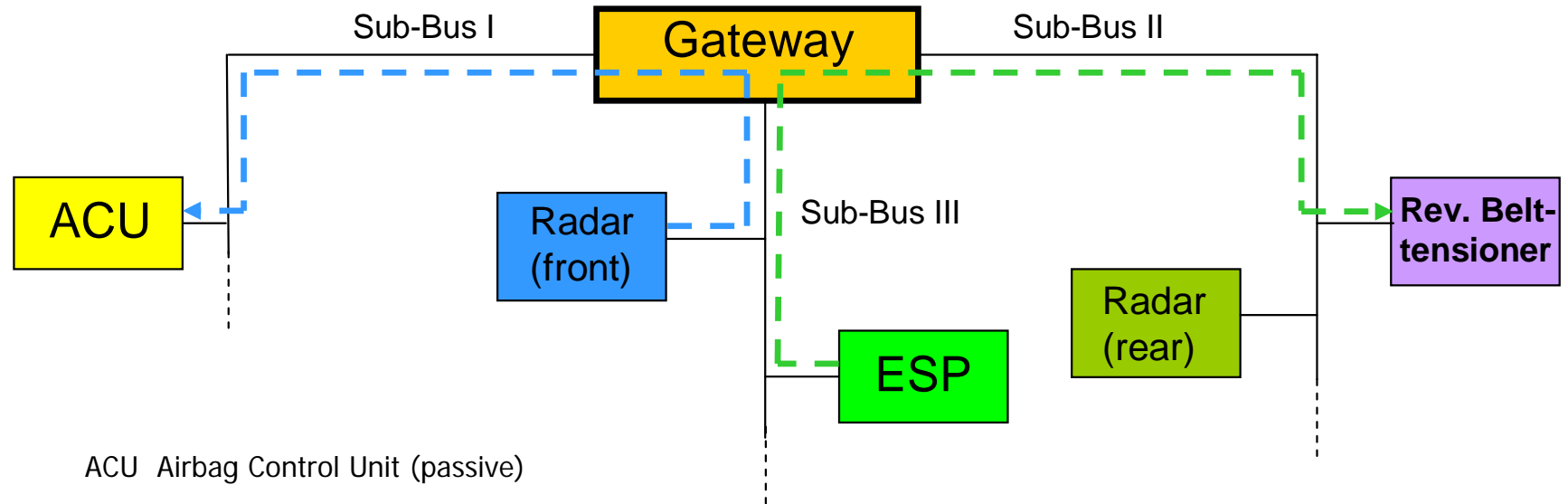
- plausibility: external/internal sensors
- redundancy: main- μ C, safety- μ C
- safing: acceleration must achieve threshold



Transmission of Safety-Relevant Sensor Data: ACU

■ Future extensions

- communication with other systems via car bus systems (such as CAN, FlexRay)



ACU Airbag Control Unit (passive)

ESP Electronic Stability Program (active)

— — Communication between anticipatory sensors and passive safety components

— — Activation of reversible belt-tensioner at detection of critical driving situations

Transmission of Safety-Relevant Sensor Data

■ Objectives

- reliability analysis of existing and future architectures
- for future architectures the performance analysis is needed, e.g.,
 - latency and jitter of data transmissions via bus systems
 - internal data throughput of the ACU
 - data protection by either hardware (redundant components) or software (CRC, filtering)
- documented computation of measures
- validation by simulation and measurement
- tool with GUI

■ Approach

- reliability models (fault trees and extensions)
- performance models (stochastic Petri nets, Markov chains, ...)
- worst-case performance models (network calculus)

Transmission of Safety-Relevant Sensor Data

■ Fault tree analysis

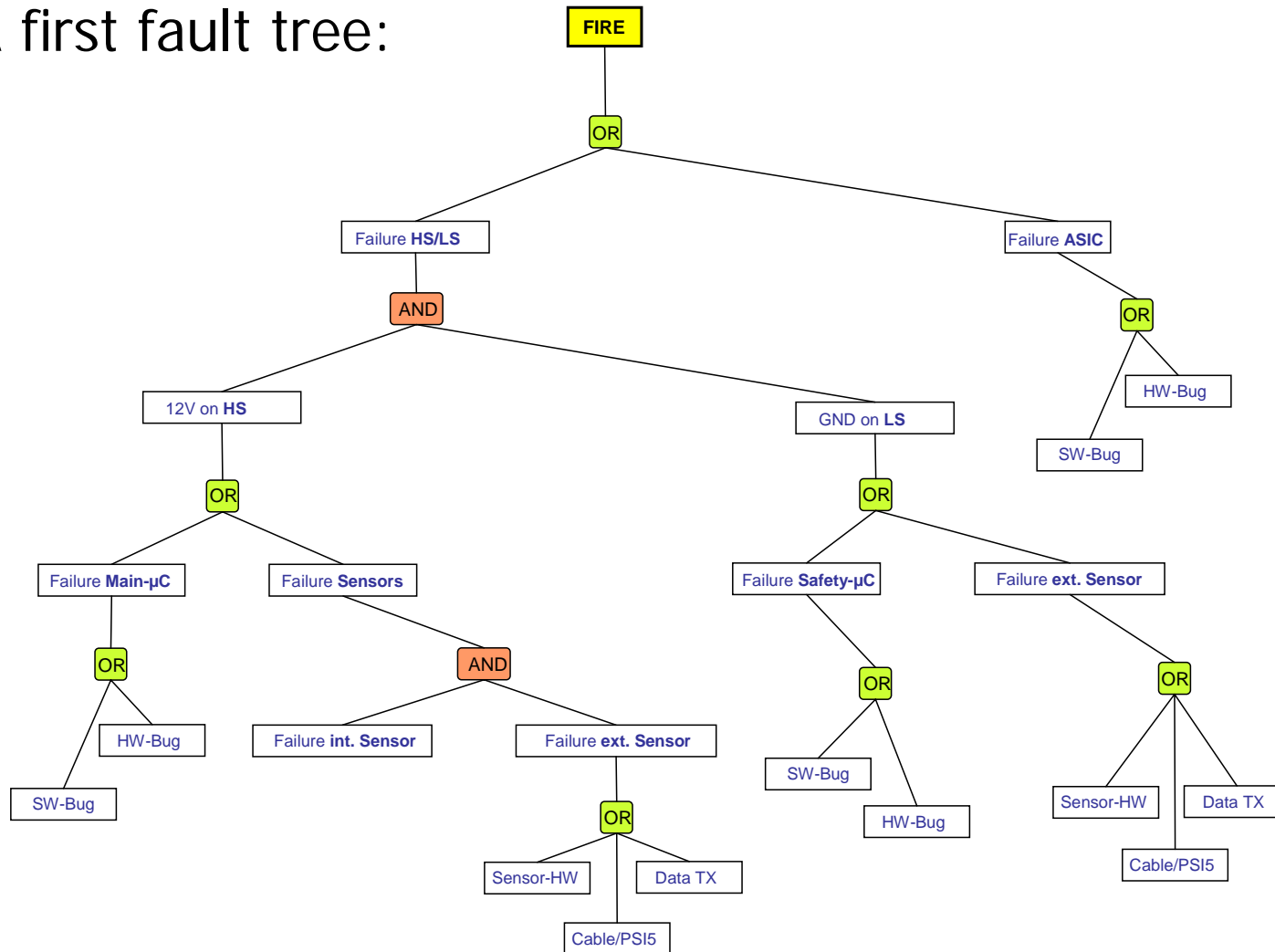
- for the existing system
- failure data available from Quality Management Audi, suppliers and internal laboratory tests
- need to fit distributions
- and to select tool: Isograph Faulttree++, Moebius, OpenSESAME, ...

■ Some challenges

- common cause failures, e.g. malfunction of external crash sensors causes system failures in different branches
- need state-based extensions, state-explosion problem
- representation of non-exponential distributions adds to this problem
- typically very low failure frequencies for some events, maybe not enough data

Transmission of Safety-Relevant Sensor Data

■ A first fault tree:



Transmission of Safety-Relevant Sensor Data

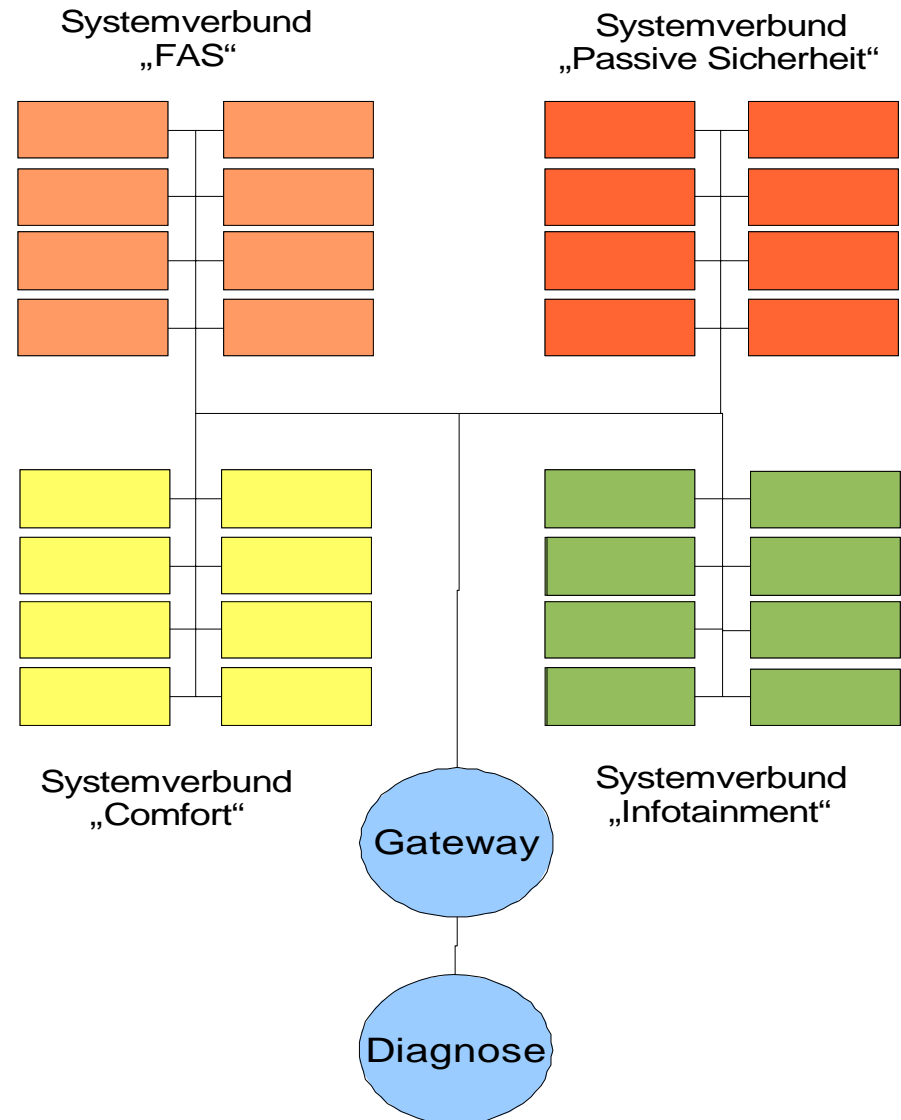
■ Modeling of reliability/performance/timeliness

- future extensions allow more and can be cheaper, but become more complex
- their reliability cannot be assessed by conventional fault trees
- need to consider network effects and response times of other systems
- isolated reliability/performance/real-time analytic models or combined larger “performability” model (Petri nets, Markov chains, ...)?
- have also developed a network calculus model which allows to bound transmission delays for all CAN priority classes
- in any case, these models have to be validated by
 - discrete-event simulation
 - hardware test equipment in laboratory

Simulation of Time-Triggered Architectures

■ Simulation of Time-Triggered Architectures

- in cooperation with Audi Electronics Venture GmbH
- PhD project of Thorsten Frunzke
- today 70+ electronic control units (ECUs)
- independent applications on ECUs
- event-driven computation & communication
- communication via gateway, LIN, CAN, FlexRay, MOST



Simulation of Time-Triggered Architectures

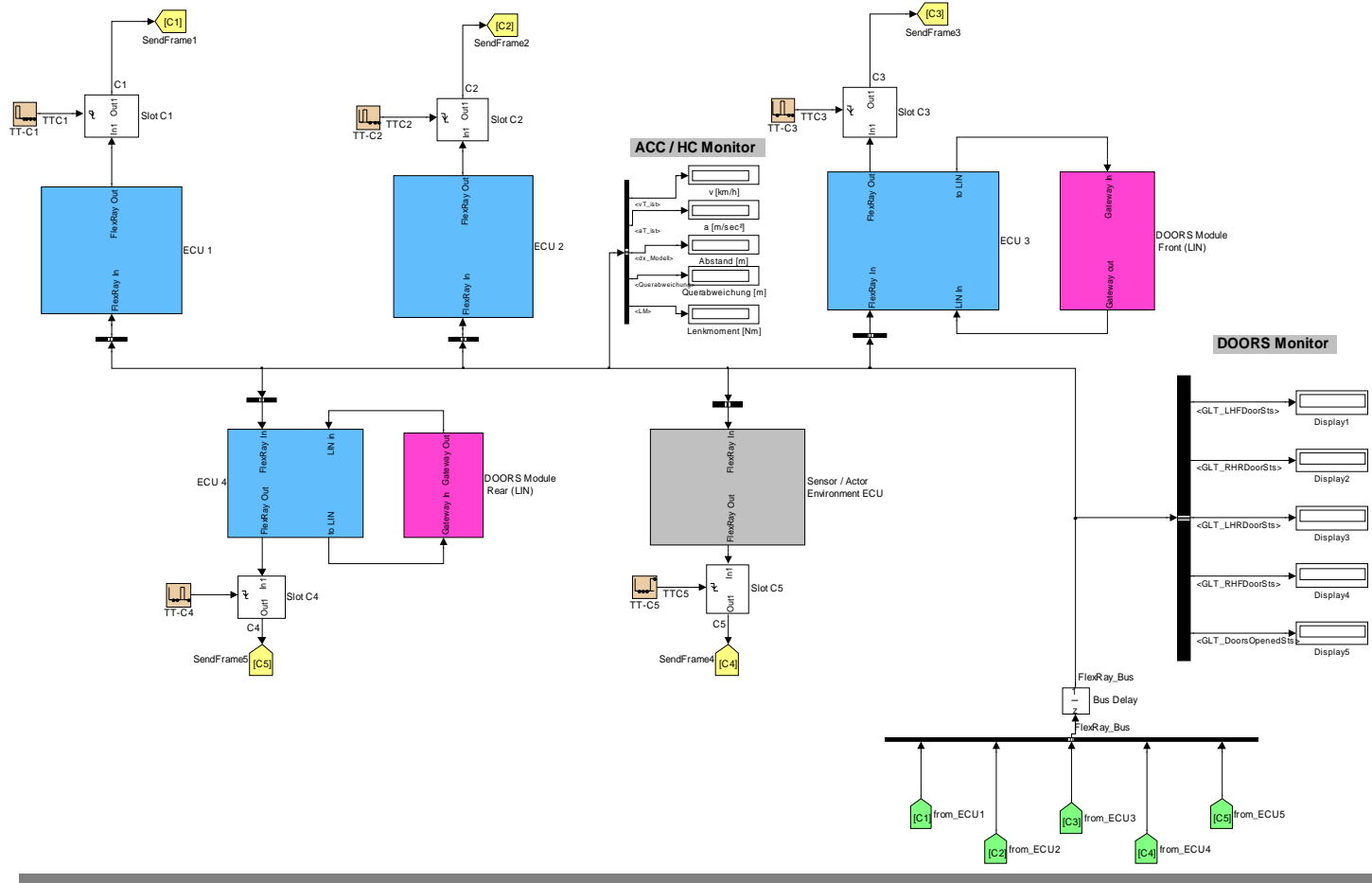
■ Simulation of Time-Triggered Architectures

- time-triggered architectures suggest a paradigm shift, e.g.:
 - communication: TDMA, guaranteed latency, bandwidth, jitter
 - computation: static allocation of resources
 - fault-tolerance mechanisms
 - distributed applications
- goal: study the effect of system architecture by simulation
- extensive continuous Simulink models for system dynamics are available
- however, for the effects of computation and communication discrete-event simulation is more adequate
- StateMate cannot do the job, maybe Simevents is able to do it
- challenge: hybrid simulation including necessary statistics to jointly study the system architecture and system dynamics

Simulation of Time-Triggered Architectures

■ A first simulink model

- Driver Assistance/Collision Warning & Avoidance
- top level view:

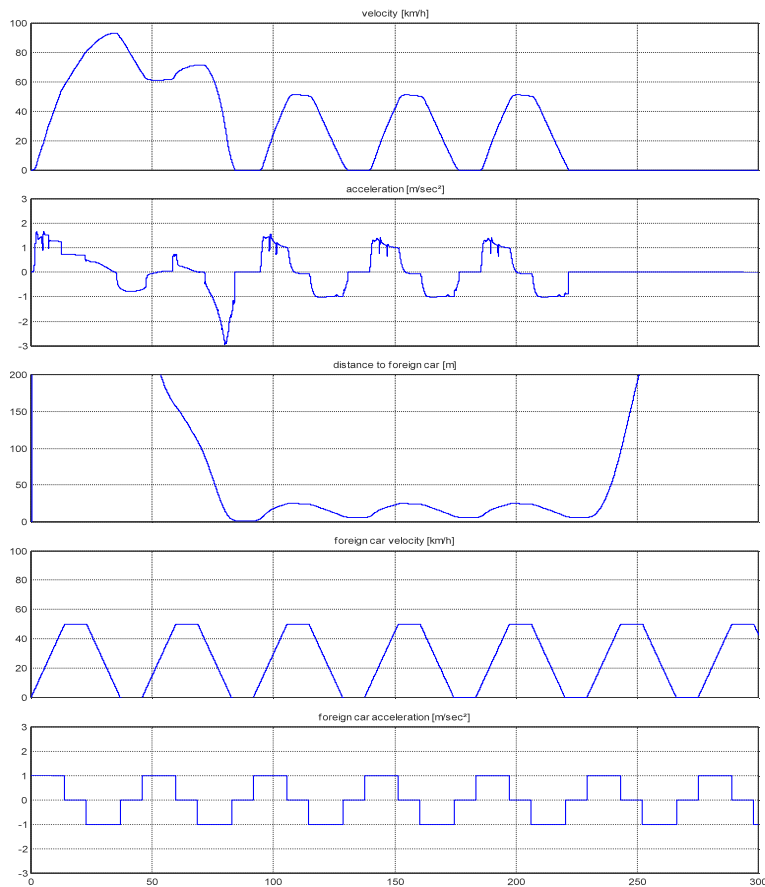


Simulation of Time-Triggered Architectures

■ First results

- two cars in sequence, without and with TT architecture

Functional Simulation



velocity
[km/h]

acc. [m/sec²]

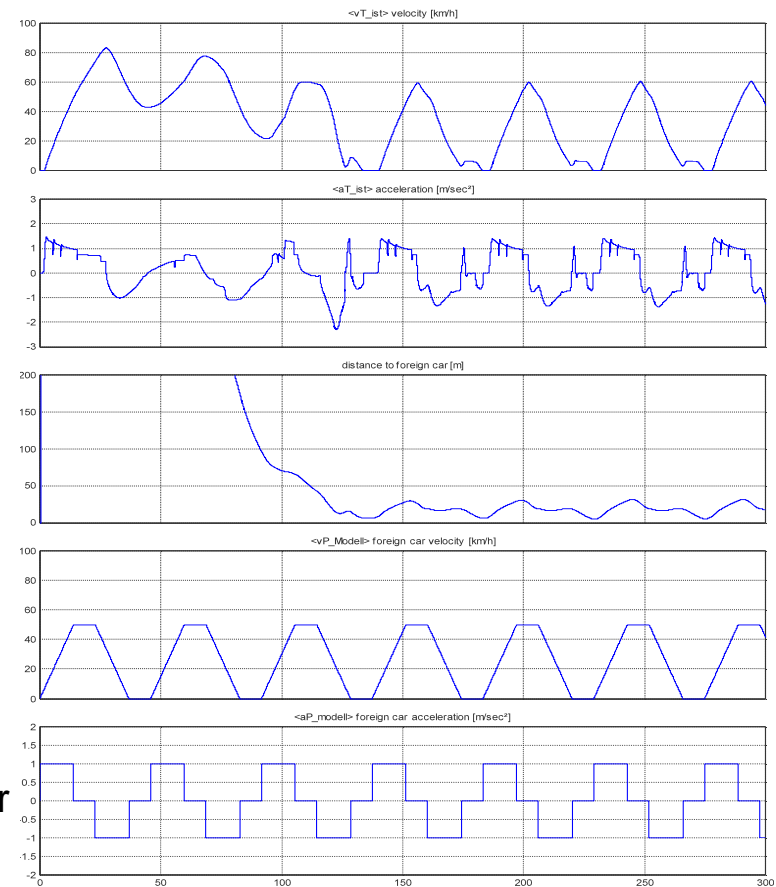
distance [m]

velocity,
predecessor
[km/h]

acc. predecessor
[m/sec²]

Time offset: 0

Simulation containing TT



Time offset: 0

Conclusions

■ Quantitative aspects of networked (embedded) systems

- measurements
- stochastic modeling
- integration with system engineering workflow
- a combination with real-time analysis would be useful