Implementing a Ring-based Real-time Capable Network Using a Multithreaded Java Processor

Sascha Uhrig

TU Dortmund

Outline

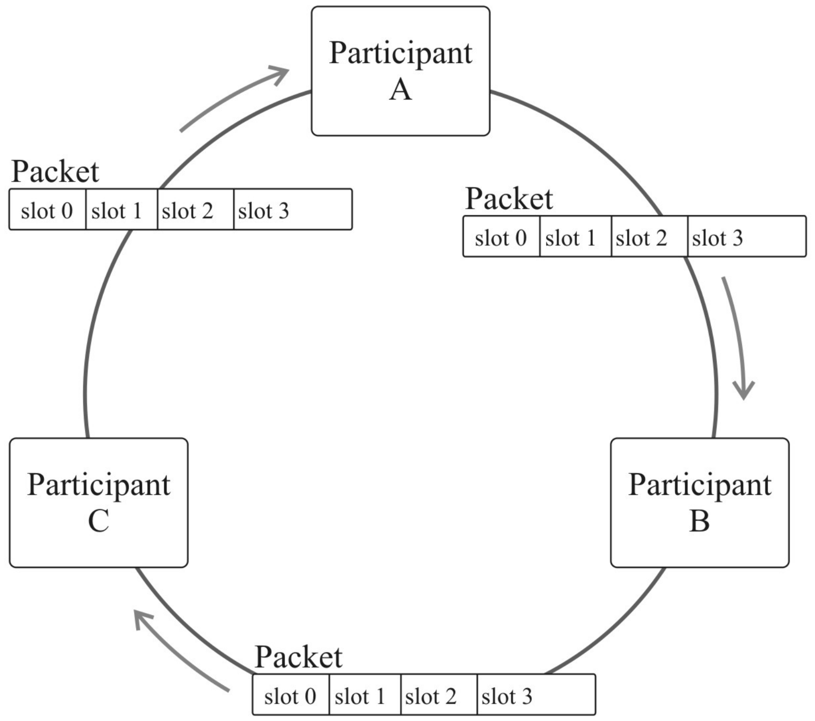
- Motivation
- Communication Ring
- Multithreaded Guaranteed Percentage Scheduling
- Implementation
- Evaluation, Real-time Capability
- Conclusion & Future Work

Motivation

- Cars show a highly heterogeneous embedded distributed system
  - Requirements
  - Software development
  - Hardware
  - Communication
- Can this be improved?
  - Unifying the hardware
  - Unifying the software development
- We target at replacing all ECUs in a car by a Java ECU, so we need suitable
  - Hardware system
  - JRE
  - Communication system

The Communication Ring

- Based on
  - 1-to-1 connections of all participants
  - Fixed number and size of packets
  - 2-fold bidirectional data transfer (clock- and counterclockwise)
  - Publishing information within slots inside the packets
  - Uniform movements of the packets
- Communication is predefined
  - It is predefined which information is transported in which slot
- Active forwarding of packets to next participant in both directions



Clockwise packet movement

Features of the used jamuth Processor

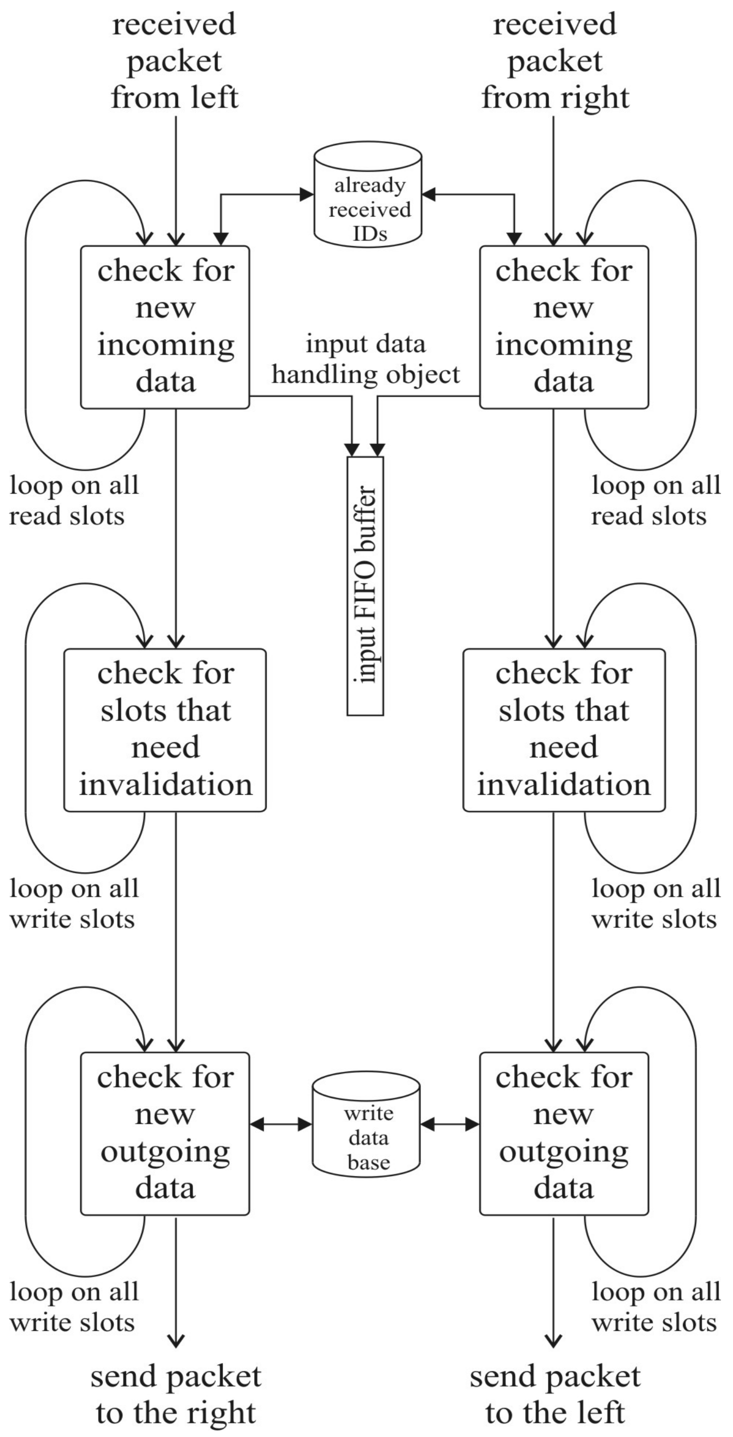
- Multithreading:
  - Multiple threads can be executed in an overlapped parallel fashion
  - Fine-grained parallel execution of Java threads
  - Latencies of one thread cause execution of another one
- Guaranteed Percentage (GP) scheduling:
  - Based on intervals of 100 clock cycles
  - Number of clock cycles a thread is executed can be defined
    (number of cycles is equals to the percentage of overall processor performance)
  - Overall processor performance can be assigned (nearly) arbitrarily to up to four hardware thread slots
  - Restrictions come from slow memory accesses
- One slot is reserved for the garbage collection
  - One slot is required for the application (software scheduling)
  - Remaining two slots will be used for the communication ring

technische universität
dortmund

Microcontroller Systems

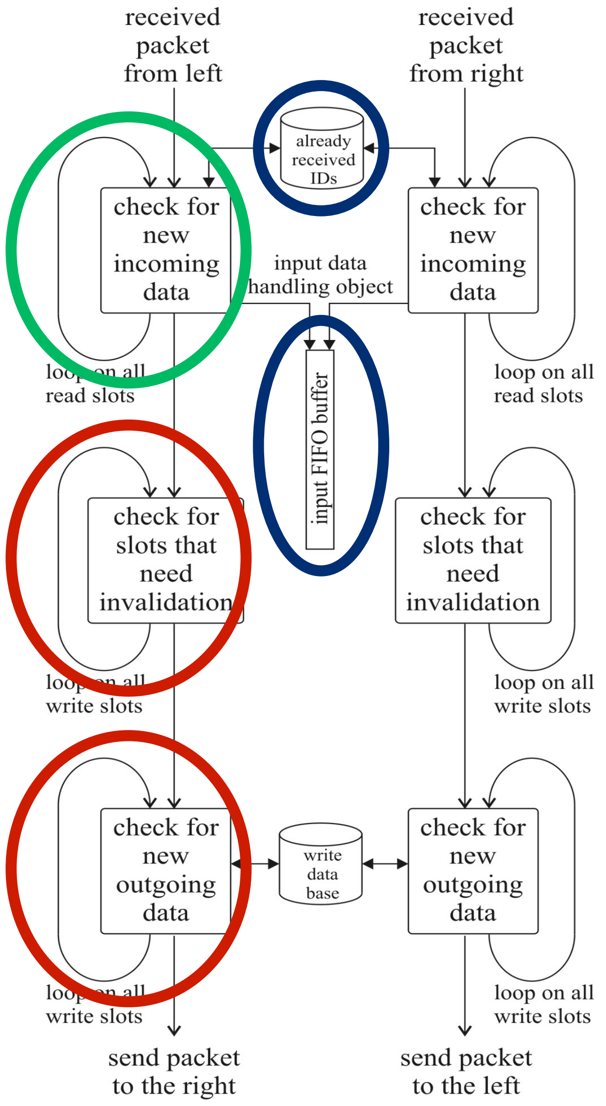
Implementation Details

- One thread per direction
- Executed in a hardware slot using GP scheduling
- Three separated packet handling steps
  - Incoming data
  - Invalidation of own data
  - Sending new data
- Communication with application threads
  - Receive FIFO
  - Single send buffer per slot


received packet from left
received packet from right
already received IDs
check for new incoming data
input data handling object
check for new incoming data
loop on all read slots
loop on all read slots
input FIFO buffer
check for slots that need invalidation
check for slots that need invalidation
loop on all write slots
loop on all write slots
check for new outgoing data
write data base
check for new outgoing data
loop on all write slots
loop on all write slots
send packet to the right
send packet to the left

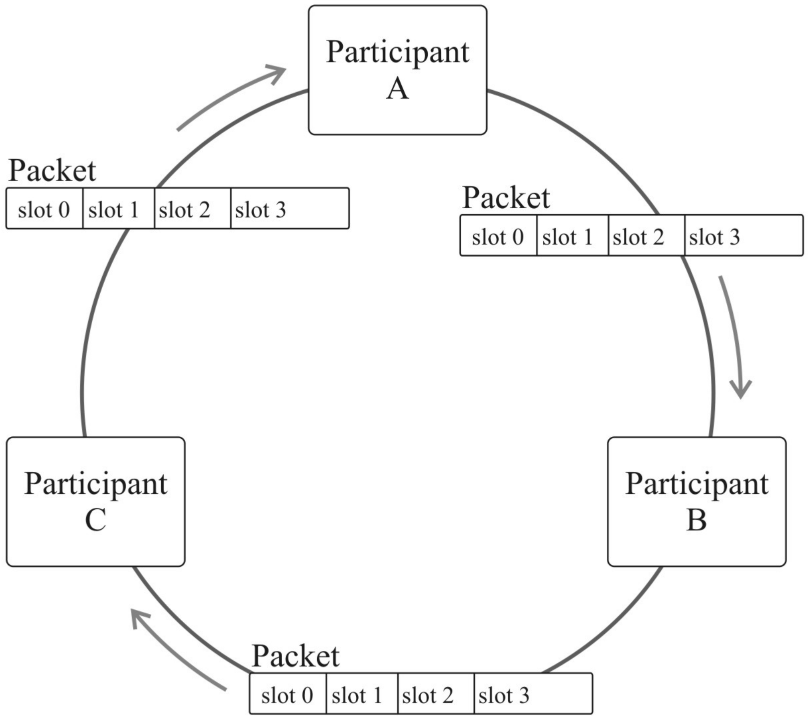
Evaluation, Real-time Capability

- Overall performance depends on the performance of the slowest partner
- GP parameter should be adjusted for each partner individually
  - Lower value reduces communication performance
  - Higher value does not bring any advantage but reduces performance of the remaining system
- Impact on the WCE cycles?
  - Read slots
  - Write slots (major issue)
  - Synchronization overhead


received packet from left
received packet from right
already received IDs
check for new incoming data
input data handling object
check for new incoming data
loop on all read slots
input FIFO buffer
loop on all read slots
check for slots that need invalidation
check for slots that need invalidation
loop on all write slots
loop on all write slots
check for new outgoing data
write data base
check for new outgoing data
loop on all write slots
loop on all write slots
send packet to the right
send packet to the left

Performance Evaluation (Average Perf.)

- FPGA Prototype
- Direct Ethernet connections (each board offers 2 Ethernet connectors)
- 5 participants
- One participant generates network load
  - 3…24 write slots
- Assigned percentage of all participants has been modified
  - 2…16 percent per thread

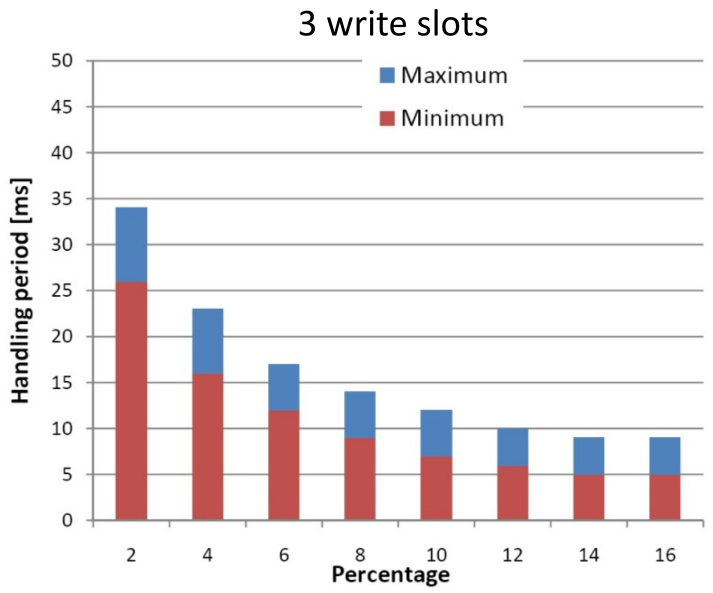


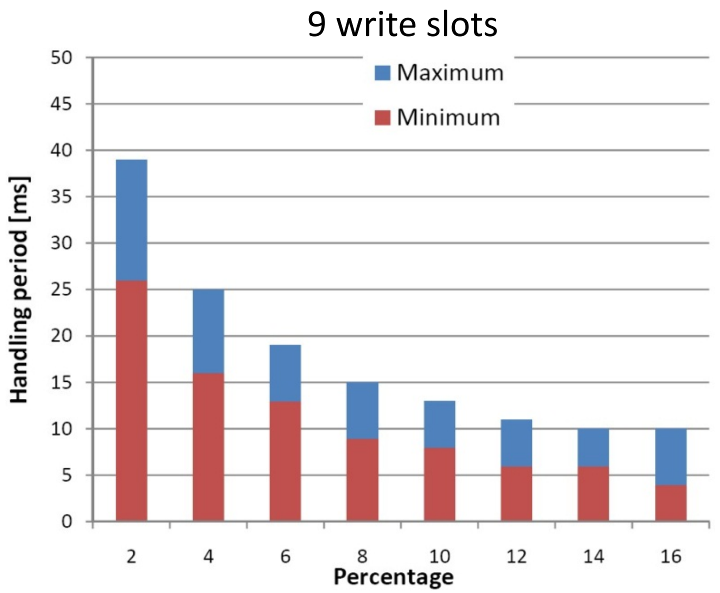
technische universität
dortmund

Microcontroller Systems

Performance Evaluation (Average Perf.)

- Several configurations with ***n*** write slots and ***p*** as GP parameter
- Measured time between arrivals of consecutive packets
  - Recorded the minimum and maximum period





Minimum and maximum period between consecutive packet arrivals

Parameter Estimation

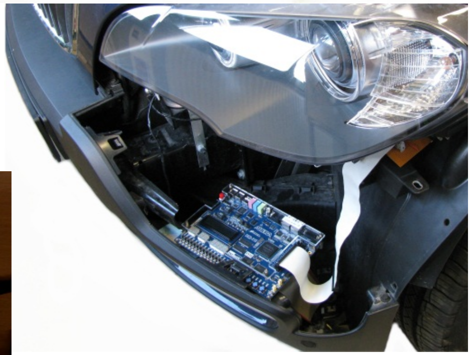
- If no hard real-time requirements need to be met GP parameters can be estimated, depending on
  - Number of write slots of each particular participant
  - The required P2P latency
- Estimation table is derived from the average performance measurements

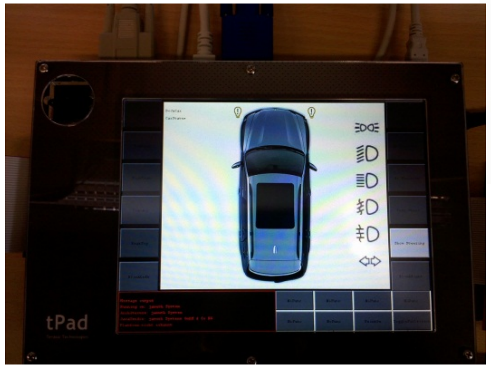
Latency [ms]	10	12	15	20	40
Slots					
3	12%	10%	8%	6%	2%
6	14%	12%	8%	6%	2%
9	14%	12%	8%	6%	2%
12	16%	12%	8%	6%	4%
15	-	16%	12%	8%	4%
18	-	16%	12%	8%	4%
21	-	-	14%	8%	4%
24	-	-	14%	10%	4%

GP parameter estimation table

The Prototype

- Ring is implemented in a BMW X5
- Integrated Java ECUs
  - Front system
    - Lights, distance sensors, rotation sensors
  - Rear system
    - Similar to front system
  - Dashboard
    - Info panel
  - Steering
    - Steering wheel, pedals
- Interconnect by proposed communication ring
- No connection to original system





Conclusion & Future Work

- Case study of using a multithreaded processor with integrated GP scheduling
  - How to handle multiple (two) real-time events in parallel
  - How to adjust the GP parameters for hard real-time systems
  - Relaxed way to adjust the parameters for soft real-time requirements
- Real static WCET analysis of the communication thread(s)
- Integration of additional Java ECUs to get the BMW running…

Thanks for your attention

- Questions?