Realtime Publish/Subscribe for Cyber-Physical Systems

KSWS / Projekt / NEidI / Projekt CSI

Dr.-Ing. Peter Danielis
Verteiltes Hochleistungsrechnen (VHR)

Dr.-Ing. Helge Parzyjegla
Architektur von Anwendungssystemen (AVA)
What is Realtime (Echtzeit)?

Not necessarily fast, but **predictable**!
→ Do the **right thing** at the **right time**.
What is Publish/Subscribe?

Publishers produce data and offer them to everyone interested

$\text{publish}(n)$

Subscribers subscribe those data, in which they are interested

$\text{subscribe}(F_n)$

Publish/Subscribe Notification Service
Scalable $m:n$-group communication

$subscribe(F_n)$

$publish(n)$
What are Cyber-Physical Systems?

- Systems containing software components and mechanical or electronic parts that are interconnected via network
- Interact with the real, physical world
  - are subject to physical laws
  - have requirements w.r.t (real) time
- Examples
  - Industry robots
    - Production line in the smart factory
    - Reconfigurable production cell of a smart factory
  - Modern (autonomous) vehicles
    - Steer/fly by wire
    - Autopilots of any kind
Industry Robots in a Smart Factory

Time-critical communication when handing over work pieces.

Industry robots made by Kuka
Reconfigurable Production Cell

Flexible communication in case of task changes.

Industry robots made by Kuka
Communication Schedule

Streams
  > From node 6 to nodes 7 and 10 (multicast)
  > From node 10 to node 7

Schedule
  > Determines exactly when which packet is sent over which link
  > Has to be always without conflicts → provable correct
  > Needs to be adapted whenever communication pattern changes
  > Additional traffic of lesser importance is possible
Underwater-Scenario 1:
Maintenance of the foundations of offshore wind turbines
Underwater-Scenario 2: Clearance of Unexploded Ordnance (UXO) from World War II
Projects and Collaborations

> Realtime publish/subscribe communication
  > Part of a DFG project
  > Planning of flexible communication patterns and reservation of required time slots on communication links
  > Formal models and methods for scheduling
  > Estimation of the worst case runtime for publishing and filtering (content-based if necessary) a notification
  > Application scenario within a smart factory

> Autonomous Underwater Vehicles (AUVs)
  > Cooperation with the Institute for the Protection of Maritime Infrastructures, Resilience Department of Maritime Systems, German Aerospace Center (DLR) Bremerhaven
  > Cooperative navigation of several AUVs
  > Limited Energy restricts movement and usage of sensors
  > Opportunistic communication via acoustic modems
Tasks: Realtime Publish/Subscribe

> Simulation models for realtime communication (TSN standards)
  > Simulation of mixed-critical data traffic
  > Configuration of time-critical networks with mixed-critical traffic
  > Simulation model for per-stream filtering and policing
  > Test and extension of new TSN features of OMNeT++/INET

> TSN-Scheduler
  > ILP-Models for Gurobi or CPLEX (→ Python)
  > Own heuristics in different programming languages
    (C++, Java, Go, Rust, …)
  > Input/Output of constraints and configurations, respectively
  > Checking of computed solutions (→ Python)
  > Benchmarking (→ Docker container)
Tasks: Autonomous Underwater Vehicles

> Further development of motion models for AUVs
  > Reaction to obstacles
  > Autonomous adaptation

> Integration of simulation models for AUVs
  > Energy consumption
  > Communication with acoustic modems
  > Motion

> Simulation of cooperative missions
  > Formation of multiple AUVs
  > Mapping of the seafloor
  > Cooperative hunting

> Implementations using Simulator OMNeT++ and C++
  > Python for scripting and evaluation of simulation results
Organizational Matters

> Up to two teams
  > Team A: Realtime publish/subscribe
    (probably more fine-grained distribution of tasks)
  > Team B: Autonomous Underwater Vehicles (AUVs)

> Design methodology
  > Agile development
  > Three milestones w.r.t. design, implementation, documentation

Type and sizeSCALE of tasks depends on number and interests of participants!
Registration and Contact

> Enrollement in respective Stud.IP course

1. 23850 (Lecture) KSWS: Verteiltes Hochleistungsrechnen
2. 23848 (Integrierte Lehrveranstaltung) Neueste Entwicklungen der Informatik (Verteiltes Hochleistungsrechnen)
3. 23851 (Project) Projekt: Verteiltes Hochleistungsrechnen
4. 23897 (Integrierte Lehrveranstaltung) Projekt Master Computer Science International : AVA

> Questions via email to Peter Danielis and Helge Parzyjegla
  > peter.danielis@uni-rostock.de
  > helge.parzyjegla@uni-rostock.de