Communication algorithms and principles for a prototype of a wireless mesh network

Zachodniopomorski Uniwersytet Technologiczny w Szczecinie
Wydział Informatyki

Sergiusz Urbaniak

Outline

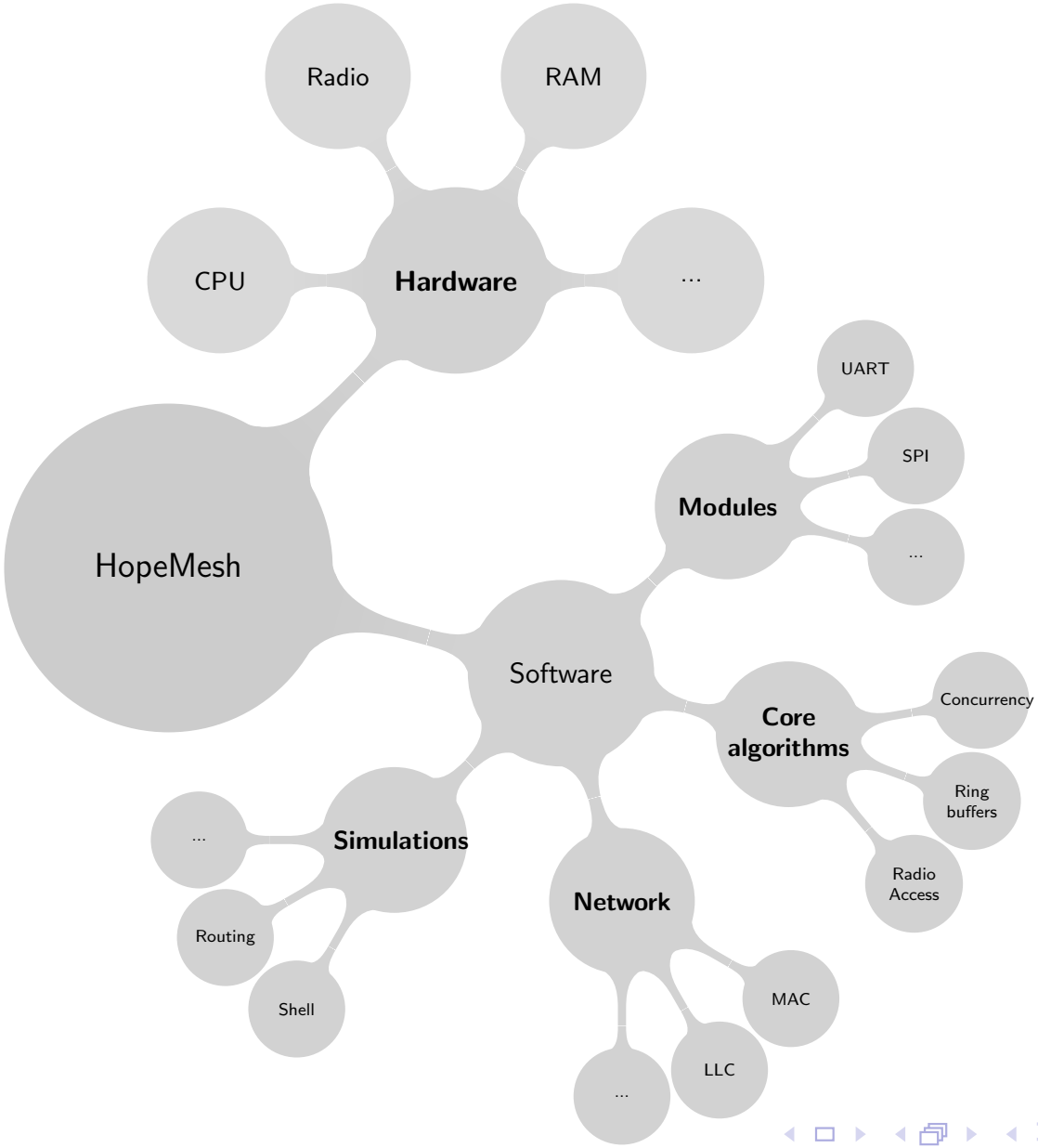
Goals

Main Goal

Research and implement a fail-safe wireless mesh network prototype using embedded technologies.

- **Hardware:** Research and develop easily reproducible hardware design.
- **Software:** Research, analyze and implement enhanced concurrent algorithms.
- **Network:** Implement a pragmatic network stack and a B.A.T.M.A.N. based routing algorithm.
- **Simulation:** Make it possible to simulate algorithmic behaviour on x86 based PCs.

Project structure

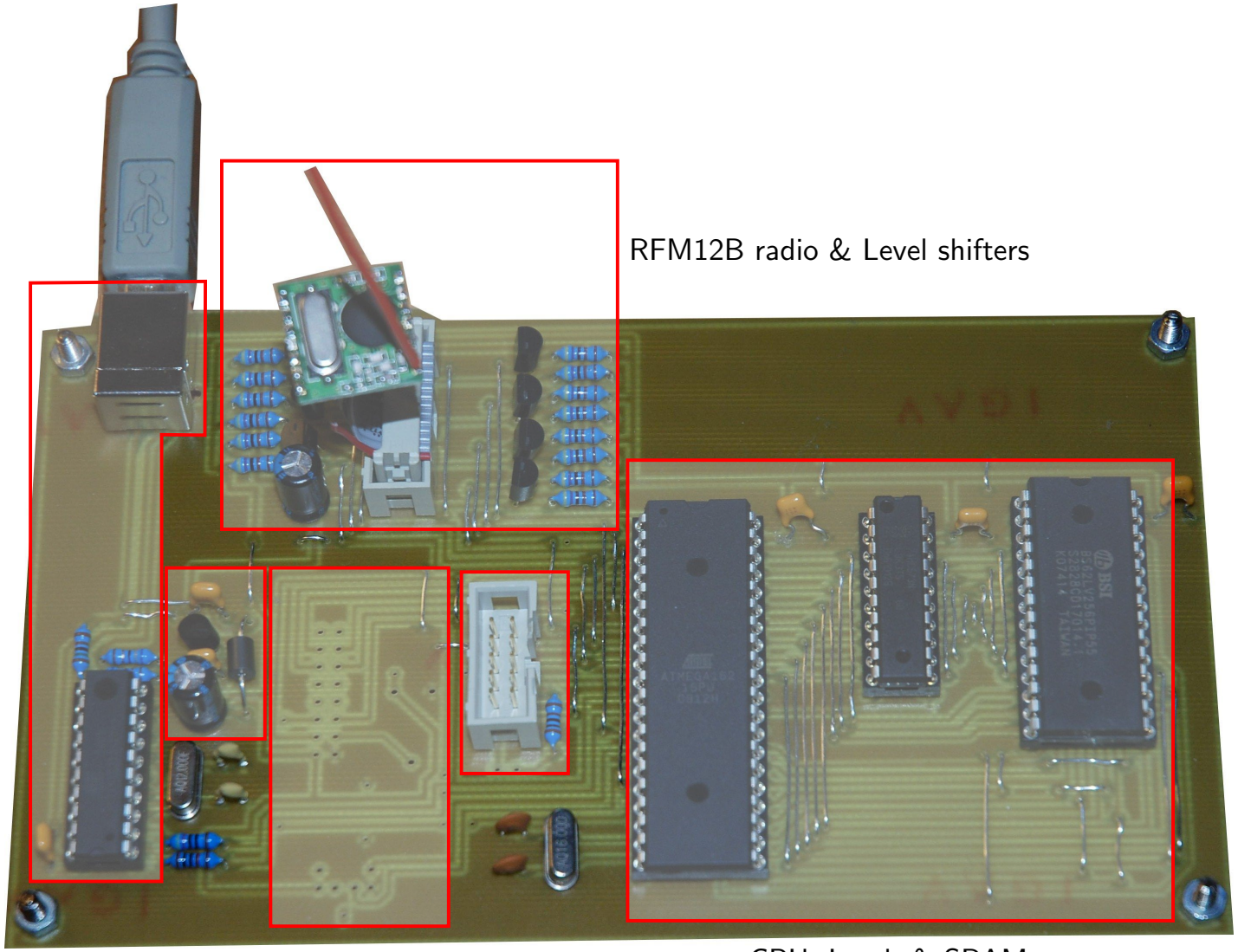


Hardware modules

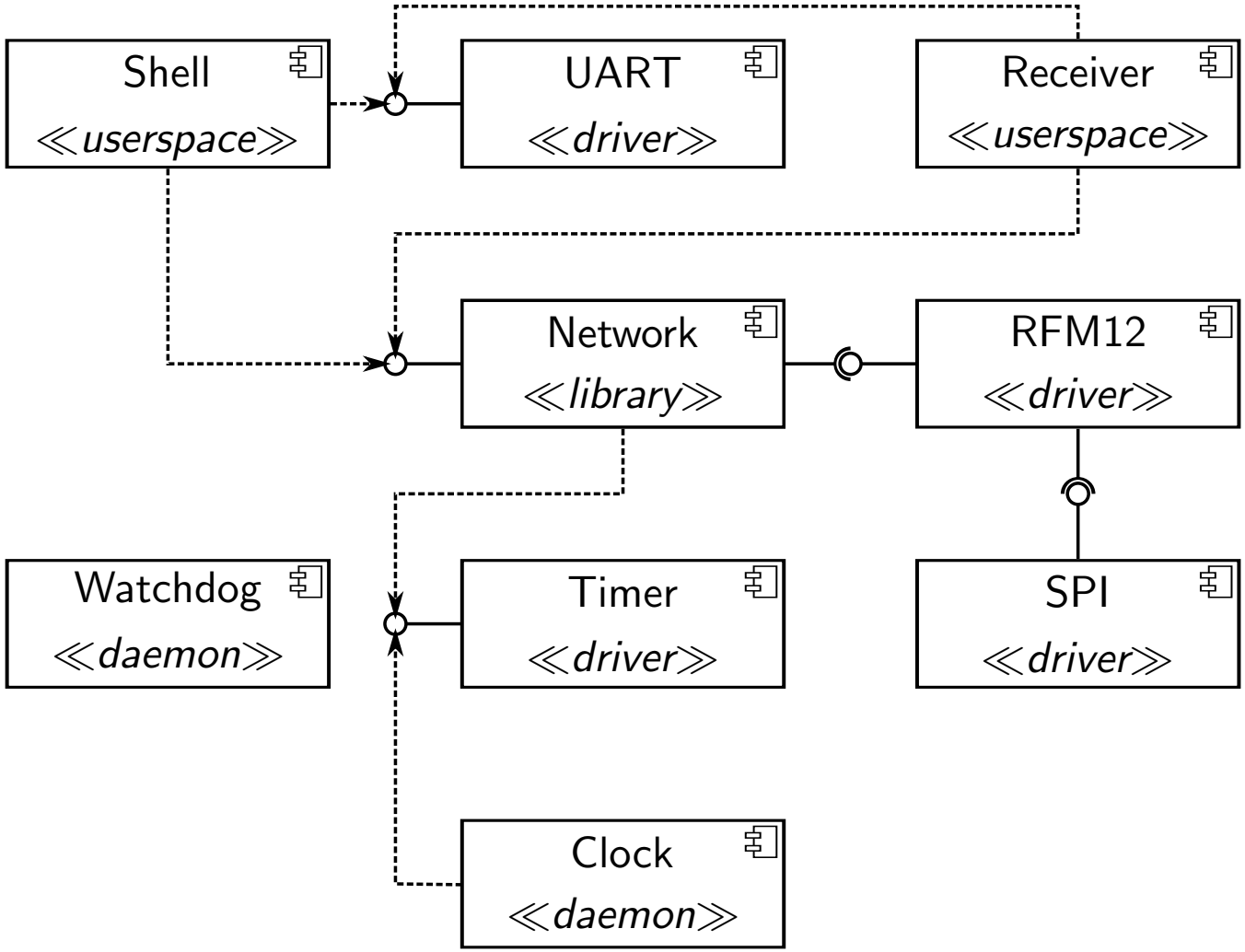
The goal was to use easily accessible embedded hardware parts:

- **CPU:** ATmega162: Includes the XMEM extension allowing to use external RAM natively.
- **RAM:** 62256 32kB SRAM: Connected to the CPU using a Latch buffer.
- **Periphery:** USB connection to PC based terminal emulators, LCD and PS/2 keyboard connection.
- **Wireless connection:** RFM12B radio module from HOPERF.

PCB - Printed Circuit Board



Module Architecture



Concurrency Model

Concurrency models for embedded systems without any operating system:

Sequential Execution

Executes modules inside an infinite main loop starting from the first module until the last one. Once the last module ends the execution starts again from the first module.

Concurrent Execution

The main function only initializes and launches concurrent modules. Modules run in their own stack space and can execute individual main loops.

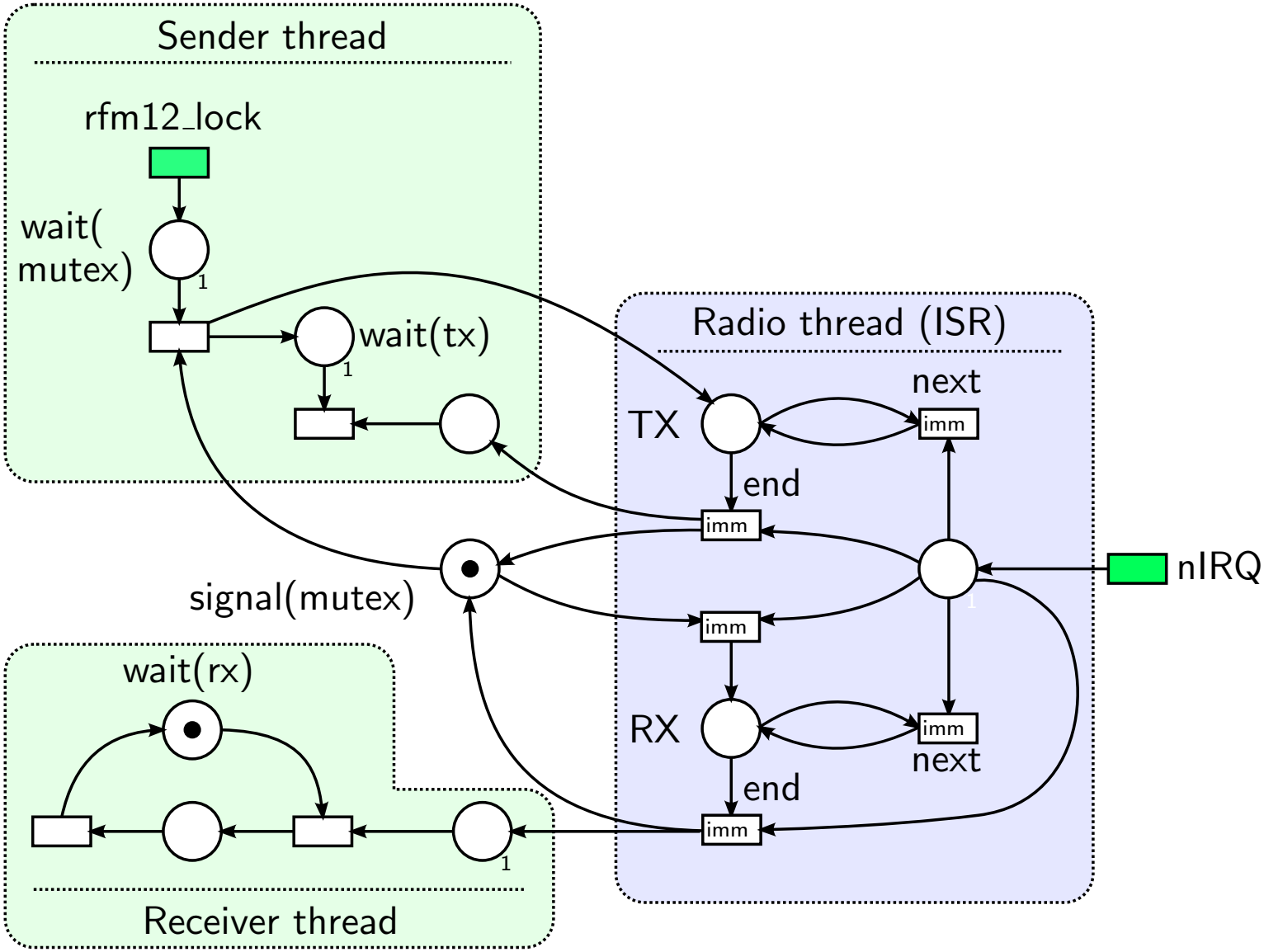
Protothreads

- **State machine based concurrency:** Bloats complexity and cannot be modeled using linear thread-based algorithms.
- **Thread based concurrency:** Results in high runtime overhead due to heavy-weight implementations.

Solution

Protothreads is a C-macro based implementation of state machines exposing a thread-like linear API. Classical thread based concurrency modelling can be used.

Petri net driver



Network stack

Modeled and implemented after Tanenbaum's hybrid model:

- **1. Physical layer**: RFM12B low cost half-duplex transceiver.
- **2. Data link layer**: A simple MAC frame format, error correction (using a Hamming code) and error detection (CRC-16).
- **3. Network layer**: RFC conform B.A.T.M.A.N. routing.
- **4. Transport layer**: NUL-terminated frame
- **5. Application layer**: Shell and receiver thread.

Simulations

x86 based Simulator

Implemented a PC based simulator by mocking hardware parts.

Shell

Implemented a shell simulator allowing to inspect and validate raw SPI streams to the radio modules.

Routing

Implemented a routing simulator which executes different mesh network topology scenarios.

Achievements

- Hardware works as expected.
- RAM is sufficient for addressing >2000 nodes.
- Concurrency model and petri net driver model works correctly.
- All simulations run successfully.
- Packet drop count (1.6%) is very low.
- Communication speed was maximised (57kbps).
- CPU load still leaves headroom for additional computational complexity.

ToDos

- **Network stack**: Enhance network stack implementation.
- **Routing**: Automatic address configuration.
- **Real Setup**: Build and test a real mesh setup involving many mesh nodes.
- **Periphery**: Add PS/2 keyboard and HD4780 LCD display.

Thank you for your patience.