Resource-Aware Service Architecture for Mobile Services in Wireless Sensor Networks

Jan Blumenthal, Dirk Timmermann
University of Rostock

ICWMC, Bucharest
2006/07/29
Software for Sensor Networks

**Required Features:**

- Adaptable to new requests
- Extracting implicit information (group dimensions)
- Collaboration of several sensor nodes
  - Location dependent services
  - Detection of dynamical phenomena

What is the problem?

Where is the phenomenon?
Implementation Requirements

- Applicable for resource-critical devices
- Extremely low memory usage (code and data)
- Adaptable to different memory architectures
- On-the-fly update of software parts
- Reusing existing software
- Low communication effort and low energy consumption

What is the problem?

Where is the phenomenon?

Sensor Network
Assumptions

Properties in Sensor Networks

• Cheap homogenous sensor nodes (mass production)
• Small, simple and suitable tasks focussed on measurement and pre-calculation
• Hardware-dependent programming required
• Unsafe communication
• Failure of nodes over time
Types of Collaboration

a) Sensor nodes **build groups** collaboratively based on an object definition
b) **Groups move** with the observed object and may travel around

---

**a) Enclose objects (areas)**
*Where is the temperature > 20°?*

**b) Observation of mobile objects**
*What is the average speed of formed object (temperature group)?
Available Software Architectures

- Software updates mostly stored in RAM
  - not reset-safe
  - only small RAM available (2KB) and usually occupied by operating system
  - data memory needs a lot of energy

- Software is not resource-aware
  - Interfaces between software parts require overhead for adaption and HAL
  - Virtual Code needs Virtual Machine (VM)

- Software (e.g. additional Services) has not full control
  - Often specifics of the microcontroller not suitable configurable (Timers, Interrupts, Ports)
  - Energy-saving mechanisms not applicable

- Mobility support is missing
Service Architecture (RASA)
Layered Software Model

- Object and Group Definitions
- Group Management
- Communication Behavior

Service Layer
- Service Management
- Memory Management
- Interrupt Handling

System Layer
- Startup
- System-Update

Base Layer

3 Layer Structure
- Minimizes interfaces
- Simplifies development of software
- System layer and service layer are exchangeable at runtime
System Layer

• Contains functionalities of
  – Operating system (transmission, interrupt handling, routing)
  – Lightweighted middleware (service management)
• Not comparable with well-known PC pendants (CORBA)
Service

- Implementation of services as mobile code (native or virtual)
- Mobile native code preferable
  + very fast and highly flexible
  + optimizable by compilers
  + partly platform independent at source code level (ANSI-C)
- No dynamical data at service runtime!
- Three parts:
  - Code
    - service description
    - executable code
    - default definitions
  - Private data
    - internal data, e.g. current state
    - measurement data
  - Public data
    - local aggregation results

After reception of a message:
- Initialize local instance
- Update own collaboration or calculation results
Modules

- Splitting a service into several modules
- This increases
  - interoperability of source code
  - modularity
- Modules are executed
  - in sequence
  - any order, if specified

Service consists of modules

- Code of Modules
  - Analysis
  - Grouping
  - Aggregation
  - Transmission

- Private Data
  - Analysis
  - Grouping
  - Aggregation
  - Transmission

- Public Data
  - Analysis
  - Grouping
  - Aggregation
  - Transmission

State machine called

- Analysis
- Grouping
- Aggregation
- Transmission

State machine left
Forwarding Services

Sensor node A
- Code
  - Private data
  - Public data
- Middleware Management

Sensor node B
- Code
  - Private data
  - Public data
- Middleware Management

1. Create message
2. Transmit message
3. Install service?
4. Compare and evaluate public data

- Service is flashed and installed at runtime once
- **Simply**, code and public data are *copied and transmitted* with each service’s message
- Service data may combined in one message by middleware
Robustness

Simply, code and public data are transmitted with each service’s message.

- Prevents a costly protocol overhead normally required for:
  - installing and forwarding a service
  - if a new sensor node enters the network and requires all neighboring services
- Easy collaboration between sensor nodes
- High robustness in sensor network
- Automatically self-healing effect, if software is able to work with changing neighbors (e.g. group building)

Services must be very small in size (code as well as public data)
Internal Modules

- **Outsourcing** of most used parts into node’s middleware, e.g. group building
- Internal module description:
  - is abstract
  - defines a link to an already existing internal module
- Data memory of internal module is mapped to service memory

![Diagram of Internal Module: Grouping](image)

- **Property Management**
- **Border Management**
- **Propagation Module**
- **Middleware Management**

**Module**

- Definitions:
  - Property
  - Border
  - Propagation method

**Private Data**

- Grouping
- `int currentState;`

**Public Data**

- Grouping
- `Position propertySectors[8];`
- `Position nonPropertySectors[8];`
Memory Organization: Chipcon CC1010

- Harvard-Architecture (Chipcon CC1010, 8051 Microcontroller)
- Two service instances (instance 1 and instance 2)
- Both instances reference to the internal Module[0] and provide a part of its own data memory to internal module[0]
Current State

Implementation

Service Layer
- Services

System Layer
- Middleware
- Operating System

Base Layer
- BIOS
- Hardware

Timeline:
- Concept
- Design
- Implementation
- Test
- Ready
Conclusion

Service-oriented software architecture supporting

- Dynamical updates/requests
- Mappable to different memory architectures of microcontrollers
- Simple interfaces and protocols
- Collaboration of nodes
- Robustness and self-healing effects
Thank You!

www.sensornetworks.org