Service and Resource Discovery in Smart Spaces Composed of Low Capacity Devices

Önder Uzun, Tanır Özçelebi, Johan Lukkien, Remi Bosman

System Architecture and Networking
Department of Mathematics and Computer Science
Eindhoven University of Technology

Outline

- ***Introduction and Problem Description***
- Proposed Solution
- Experimental Results
- Conclusions and Future Work

Johan J. Lukkien, j.j.lukkien@tue.nl
TU/e Informatica, System Architecture and Networking

Smart Spaces

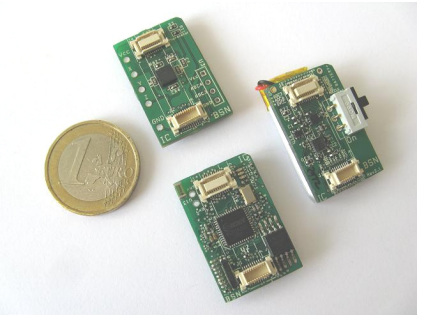
- Smart Space: a (living) space with
  - embedded technology...
    - 'regular' CE & computing devices
    - smart nodes: sensing, actuating, communicating, computing
  - ...that is networked...
    - wireless, mostly
  - ...designed to collaborate and share...
  - .... to interact with / serve users...
    - maximum autonomous operation
    - provide services with minimal user effort
  - ... and the electronics these users carry
    - e.g. body sensor networks, personal electronics

Johan J. Lukkien, j.j.lukkien@tue.nl
TU/e Informatica, System Architecture and Networking

Low Capacity Devices

- Sample specifications:
  - CPU: 1-25 MHz clock frequency
  - 1-10 kilobytes of RAM / 1-128 kilobytes of programmable ROM
  - Communication device: Frequency band 433 MHz – 2.4 Ghz
  - 2.0 – 2.5 Ah for an AA size battery
    - perhaps much smaller even
  - sensors, actuators

- Example: Wireless Sensor Network
  - networks of cheap and 'intelligent' sensors.
    - Representative of 'low-capacity device' class
    - Naturally pervasive: automatic operation, seamless integration, collaboration
  - but: little energy
  - and: small packet sizes, bit rates



Johan J. Lukkien, j.j.lukkien@tue.nl
TU/e Informatica, System Architecture and Networking

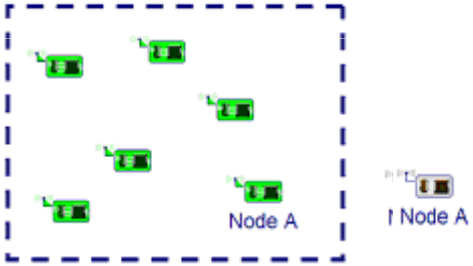
Smart Space Architectures

- Many SS architectures are heavy-weight!
  - layered architectural design for the sake of abstraction (suboptimal)
    - messaging overheads, increased energy usage.
  - State-of-the-art frameworks (e.g. OSGi, NoTA-M3) have high resource requirements.
    - Execution platform, message (xml) size and interpretation overhead
    - Implicit assumption: SS network is composed of relatively high capacity nodes (e.g. laptop computers, mobile phones, PDAs).
- Unusable directly with low capacity nodes (e.g. WSN nodes)
  - energy, computational power, network bandwidth limitations.
  - needs gateways/proxies
    - configuration overhead, reliability
    - latencies

Fundamental requirements

- Smart Space requirements
  - *programmable and configurable* – wished behavior can be imposed
  - *sharing of services* – by concurrent (distributed) applications
  - *separation of coordination and service* – separate application logic from device functions
  - *integration with the regular IP infra structure*

- Basic functionalities provided by nodes to *the network*
  - *Code load* – to define services, configuration
  - *Discovery* - detection of nodes, services, resources
  - *Service/Resource Monitoring & Management*
  - *Service composition/coordination*– by third party



- Lightweight architecture: how to provide this with low-capacity nodes?
  - This work focuses on example of service/resource discovery and monitoring:
    - Node is discovered (registered to the system)
    - Node is monitored (report introspection information)

Johan J. Lukkien, j.j.lukkien@tue.nl
TU/e Informatica, System Architecture and Networking

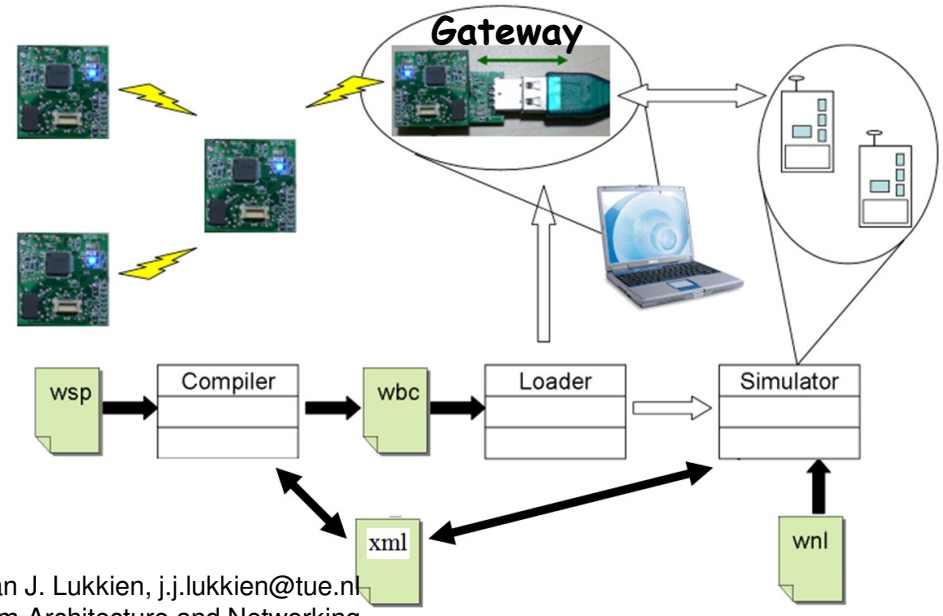
Outline

- Introduction and Problem Description
- ***Proposed Solution***
- Experimental Results
- Conclusions and Future Work

Baseline Architecture: OSAS Framework

- Open Service Architecture for Sensors (OSAS)
  - *language, to program the entire network ('macro-programming')*
    - *with concepts of a service, eventing and subscription*
    - *and* content-based *addressing*
  - *runtime system contains virtual machine (byte code)*
    - *has access to a set of system calls (OS-provided functions)*
  - *message format*

- Four components
  - *Compiler*
  - *Loader*
  - *Runtime system*
  - *Simulator*
    - runs same runtime system



Johan J. Lukkien, j.j.lukkien@tue.nl
TU/e Informatica, System Architecture and Networking

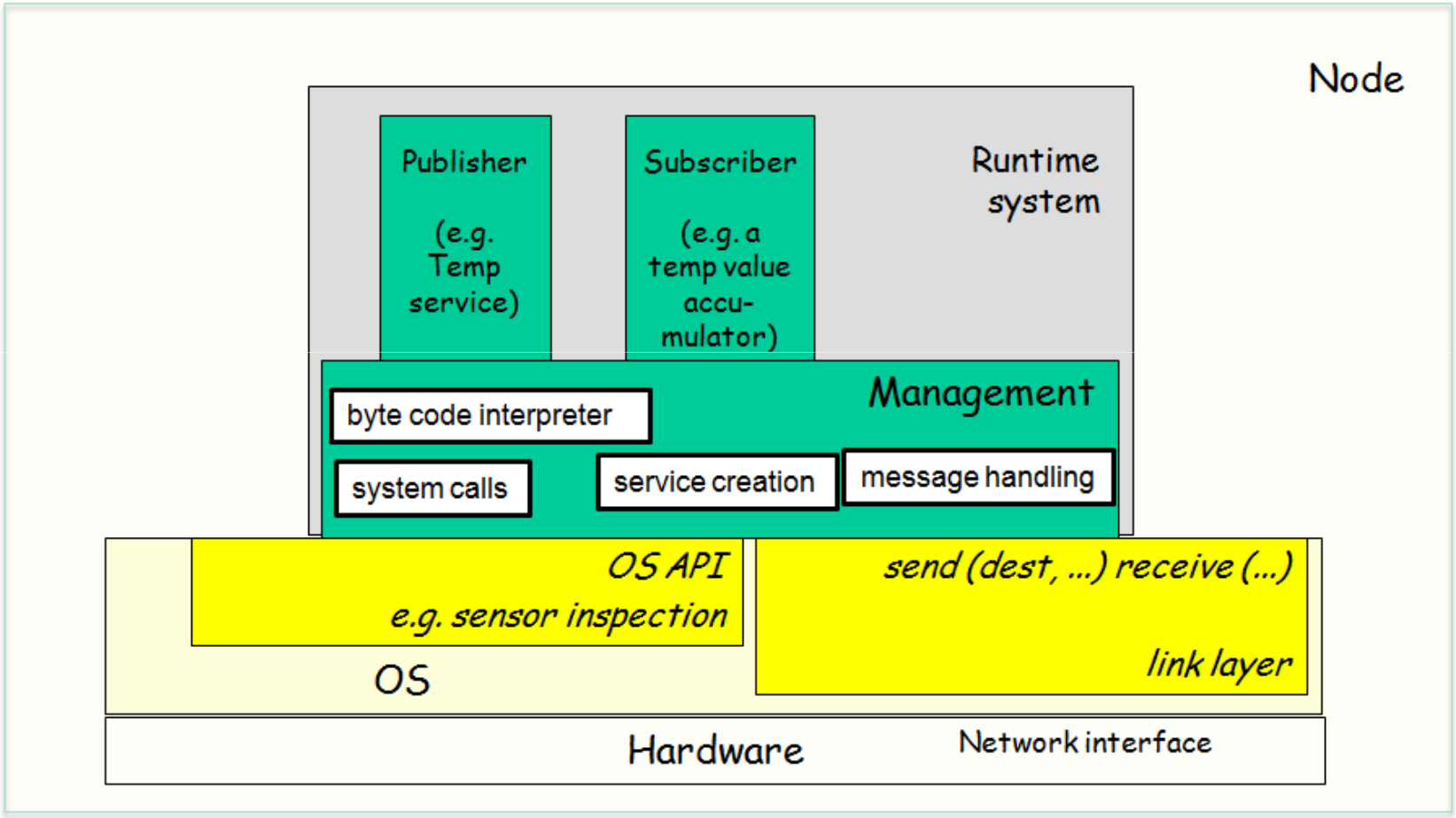
OSAS Framework: *Programming Model*

- Nodes run services:
  - A service has event *generators* and event *handlers*.
  - A link between a generator and a handler is called a *subscription*.
  - Interaction between services by a *asynchronous remote procedure call*
    - upon firing of the event
    - or simply as part of a handler

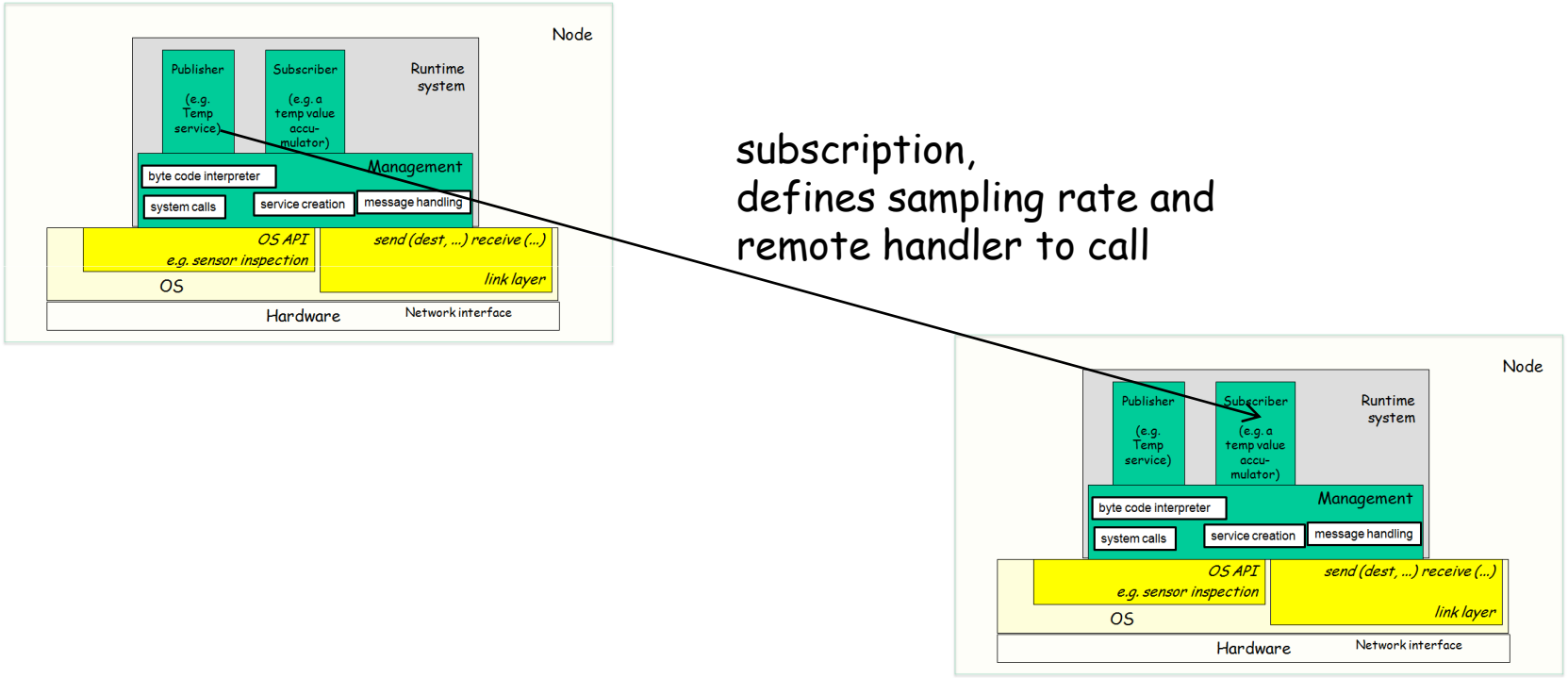
- Supported by WASP Service Composition Language (WaSCoL)
  - Specification of services with event-condition-action rules
  - Composition of services (i.e. subscriptions)
  - Use content-based addressing to associate node with service
  - *Programs can be developed and loaded incrementally*

Johan J. Lukkien, j.j.lukkien@tue.nl
TU/e Informatica, System Architecture and Networking

Run-time organization



Connection by subscriptions



OSAS Framework: *Messaging and Communication*

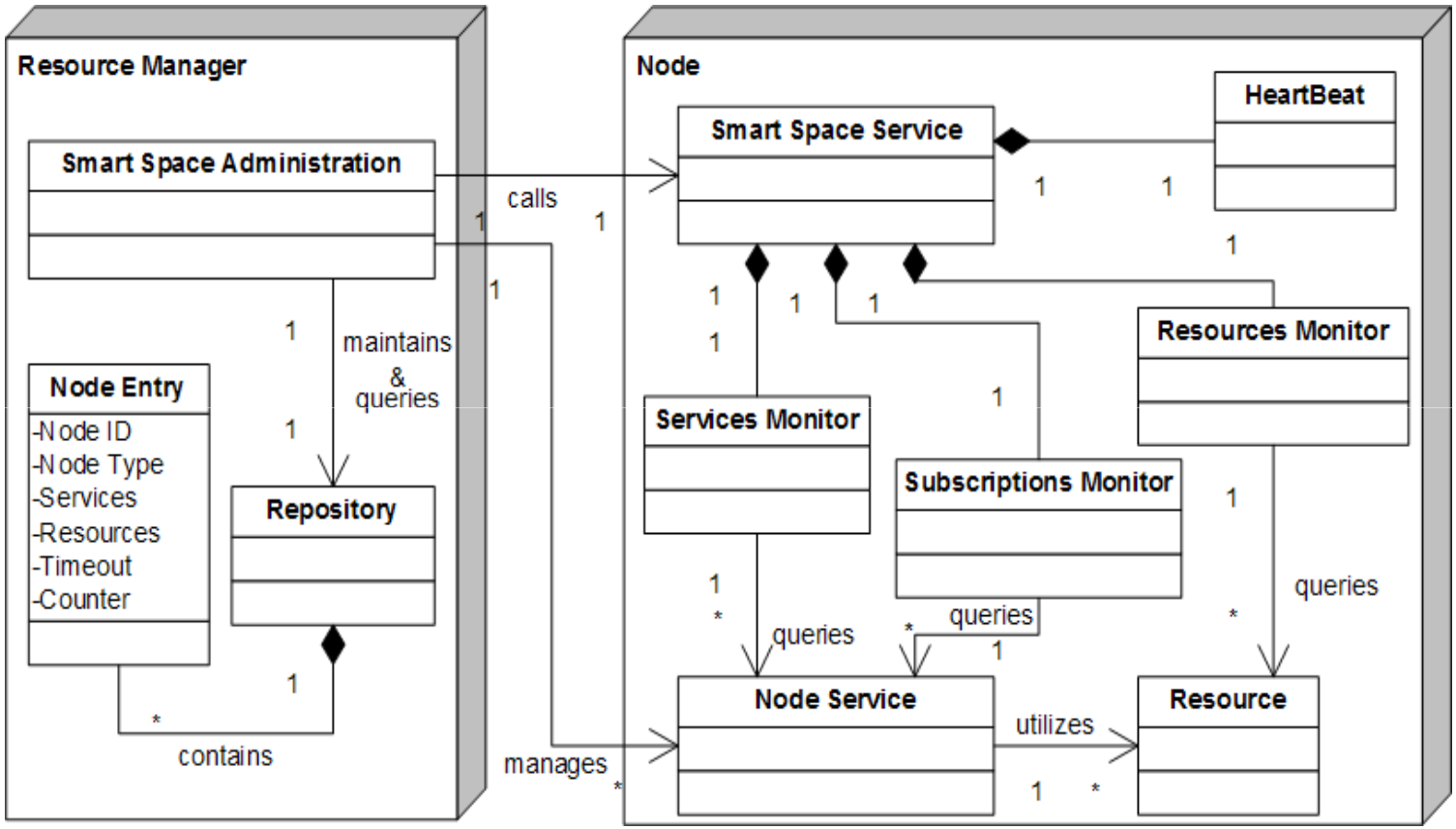
- Communication:
  - *SendToSubscribers()*, in case event 'fires'
  - *Send()* – i.e. aRPC in local neighbordhood
  - Two implicit routing methods:
    - Flooding, to transmit installation messages
    - Point-to-point: route is setup via a subscription
  - application-level routing
- Message format – asynchronous remote procedure call
  - Header; (Handler, arguments) series
  - can be bound to MAC, or other carriers (e.g. UDP)

Johan J. Lukkien, j.j.lukkien@tue.nl
TU/e Informatica, System Architecture and Networking

Example: Smart Space Management

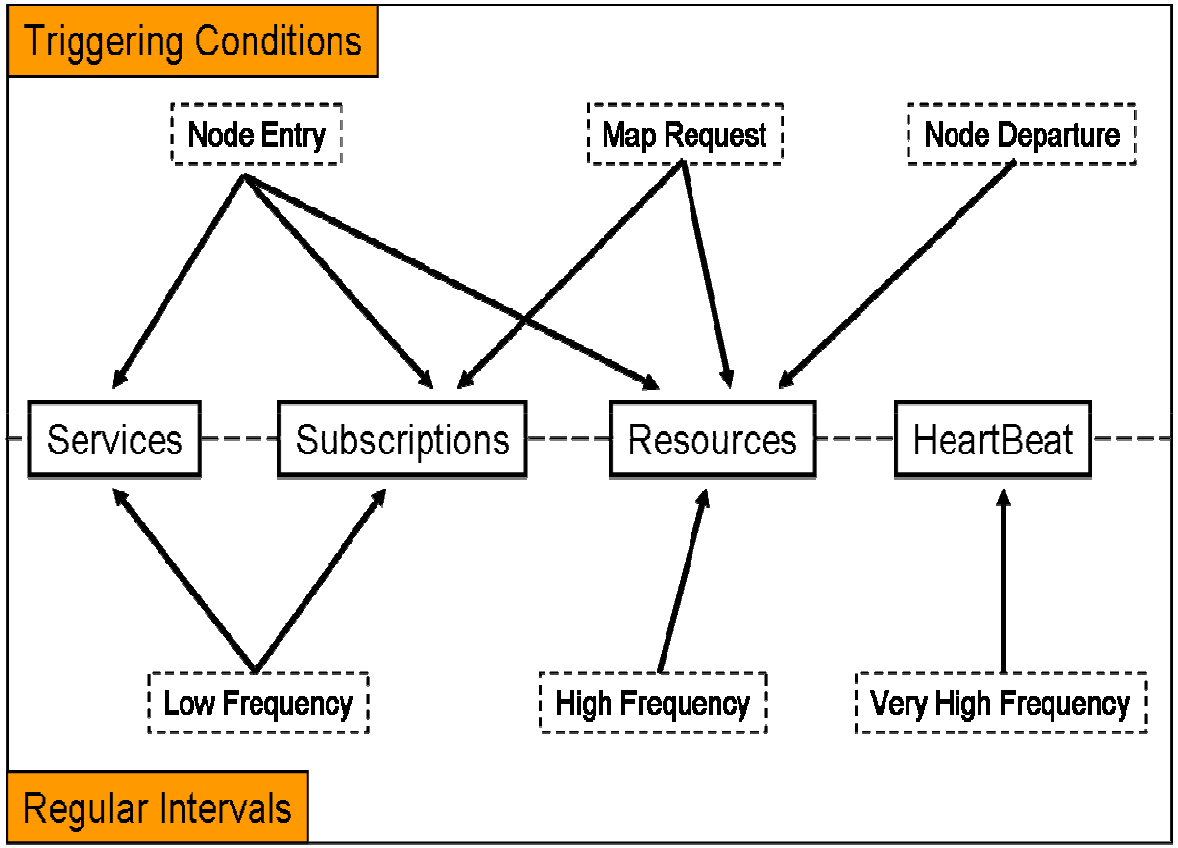
- Given: a network running an OSAS application
- A resource manager (RM, special node) wants to
  - know which nodes exist
  - which services these run and subscriptions they have
  - which resources are available on these nodes
    - memory, battery, ...
- RM may make this information available again as a service, or use it for taking *mapping decisions*

Smart Space Management Architecture



Johan J. Lukkien, j.j.lukkien@tue.nl
TU/e Informatica, System Architecture and Networking

Service/Resource Monitoring: *Reporting Intervals & Triggers*



Johan J. Lukkien, j.j.lukkien@tue.nl
TU/e Informatica, System Architecture and Networking

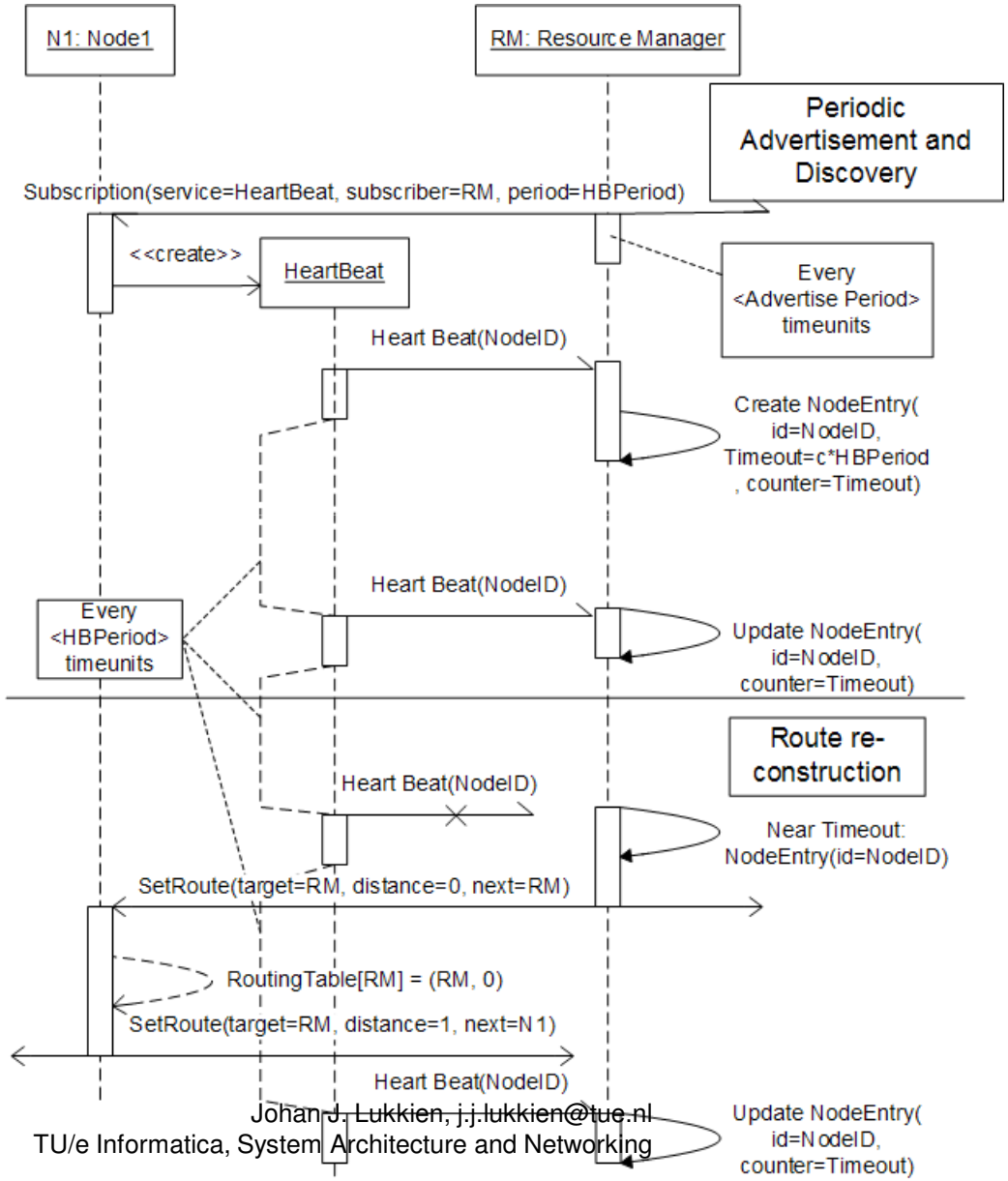
Implementation

- Management functionality is implemented *as (part of) an OSAS application*
  - Node discovery:
    - RM regularly subscribes to a heartbeat service of all nodes (in fact: advertises itself in this way)
    - result: RM receives regular heartbeat indicating nodes' presence
      - first is used to register node
  - Path reconstruction due to mobility
    - Regular subscription solves this
    - RM may subscribe to specific node after some heartbeat misses
      - or just reconstruct the route ('setroute')
  - Service & resource discovery:
    - part of heartbeat response message
    - frequency of resource reporting and service reporting as extra parameter of subscription, or as additional subscriptions
  - RM may inject those services itself

Service Discovery & path maintenance

- Periodic advertisement

- Conditional route reconstruction



Johan J. Lukkien, j.j.lukkien@tue.nl
TU/e Informatica, System Architecture and Networking

Outline

- Introduction and Problem Description
- Proposed Solution
- ***Experimental Results***
- Conclusions and Future Work

Example Scenario

Example scenario with the following parameters:

Parameter	Value
HeartBeat Period	**1 second**
Timeout Period	**6 seconds**
Route Fix Trigger	**3 seconds to timeout**
Max. Route Fix Frequency	**every 3 seconds**
Refresh Registry Period	**1 second**
Advertisement Period	**10 seconds**
LowFrequencySubscription Period	**60 minutes**
HighFrequencySubscription Period	**30 seconds**
Triggering Conditions Check/Response Period	**1 second**

Johan J. Lukkien, j.j.lukkien@tue.nl
TU/e Informatica, System Architecture and Networking

Example Scenario: *Messaging Calculations*

Sent Messages	Received Messages
HeartBeat: 60	**Advertisement: 6**
Report Resources: 2	**Resource Report Subscription: 2**
Report Services: 1/60	**Services Report Subscription: 1/60**
Report Subscriptions: 1/60	**Subscriptions Report Subscription: 1/60**
***Total:* ~ 62**	***Total:* ~ 8**

Message Count of Regular Interval Messages (per minute)

	Sent Messages	Received Messages
Node Entry	**3**	**3**
Set Route	**1**	**1**
Node Departure	**1**	**1**
Service Mapping	**2**	**2**

Message Count of Triggering Conditions Messages (per occasion)

Johan J. Lukkien, j.j.lukkien@tue.nl
TU/e Informatica, System Architecture and Networking

Deployment on Physical Nodes

	ROM **(in bytes)**	RAM **(in bytes)**
TinyOS (w/o OSAS)	**15912**	**947**
TinyOS (w/ OSAS)	**28068**	**1289**

Footprint of TinyOS and OSAS

Service	Size **(in bytes)**
HeartBeatService	**28 (14 ROM + 14 RAM)**
ReportServicesService	**30 (16 ROM + 14 RAM)**
ReportResourcesService	**30 (16 ROM + 14 RAM)**
ReportSubscriptionsService	**30 (16 ROM + 14 RAM)**
Total	***118 (62 ROM + 56 RAM)***

Footprint of Smart Space Application

Johan J. Lukkien, j.j.lukkien@tue.nl
TU/e Informatica, System Architecture and Networking

Deployment on Physical Nodes

- Message header size:
  - MAC protocol (10 bytes)
  - OSAS messaging protocol (5 bytes)

Message payload sizes (assuming 8-bit encoding):

Message	Size **(in bytes)**
HeartBeat Report	**2**
HeartBeat Subscription	**17**
Services Report	**3 + (# services)**
Resources Report	**2 + 2 * (# resources)**
Subscriptions Report	**2 + 4 * (# subscriptions)**
Services Report Subscription	**17**
Resources Report Subscription	**17**
Subscriptions Report Subscription	**17**
Service Mapping	**9**
Service Request Subscription	**17**

Johan J. Lukkien, j.j.lukkien@tue.nl
TU/e Informatica, System Architecture and Networking

Outline

- Introduction and Problem Description
- Proposed Solution
- Experimental Results
- ***Conclusions and Future Work***

Johan J. Lukkien, j.j.lukkien@tue.nl
TU/e Informatica, System Architecture and Networking

Conclusion

- Platform for Smart Spaces, as well as for general sensor networks
  - powerful primitives with little overhead
  - runtime platform
- Node discovery, Service Discovery and Resource monitoring as just another application
  - Heartbeat protocol with enhancements
  - small overheads, given the platform exists
- Service/Resource Monitoring: Classification of information types and reporting intervals-triggers
- Low footprint values, fully operational

Johan J. Lukkien, j.j.lukkien@tue.nl
TU/e Informatica, System Architecture and Networking

Future Work

- Multiple RMs that synchronize
  - reliability, distributed control
- Reduce footprint further
  - remove OS
  - can reduce ROM with 10k, RAM with 0.8K
- Monitoring mechanism:
  - Remove redundant information delivery.
  - Timing durations set by individual nodes.
  - Timing adjustments according to the dynamicity of the network.
- Management mechanism:
  - Implement service *recovery* (remapping).
  - Provide advanced service mapping and arbitration.
  - Use aggregation to reduce #messages.

Johan J. Lukkien, j.j.lukkien@tue.nl
TU/e Informatica, System Architecture and Networking