

Ambient environments

A programming model to make them reality

Harmke de Groot

harmkedg@microsoft.com

European Microsoft Innovation Center



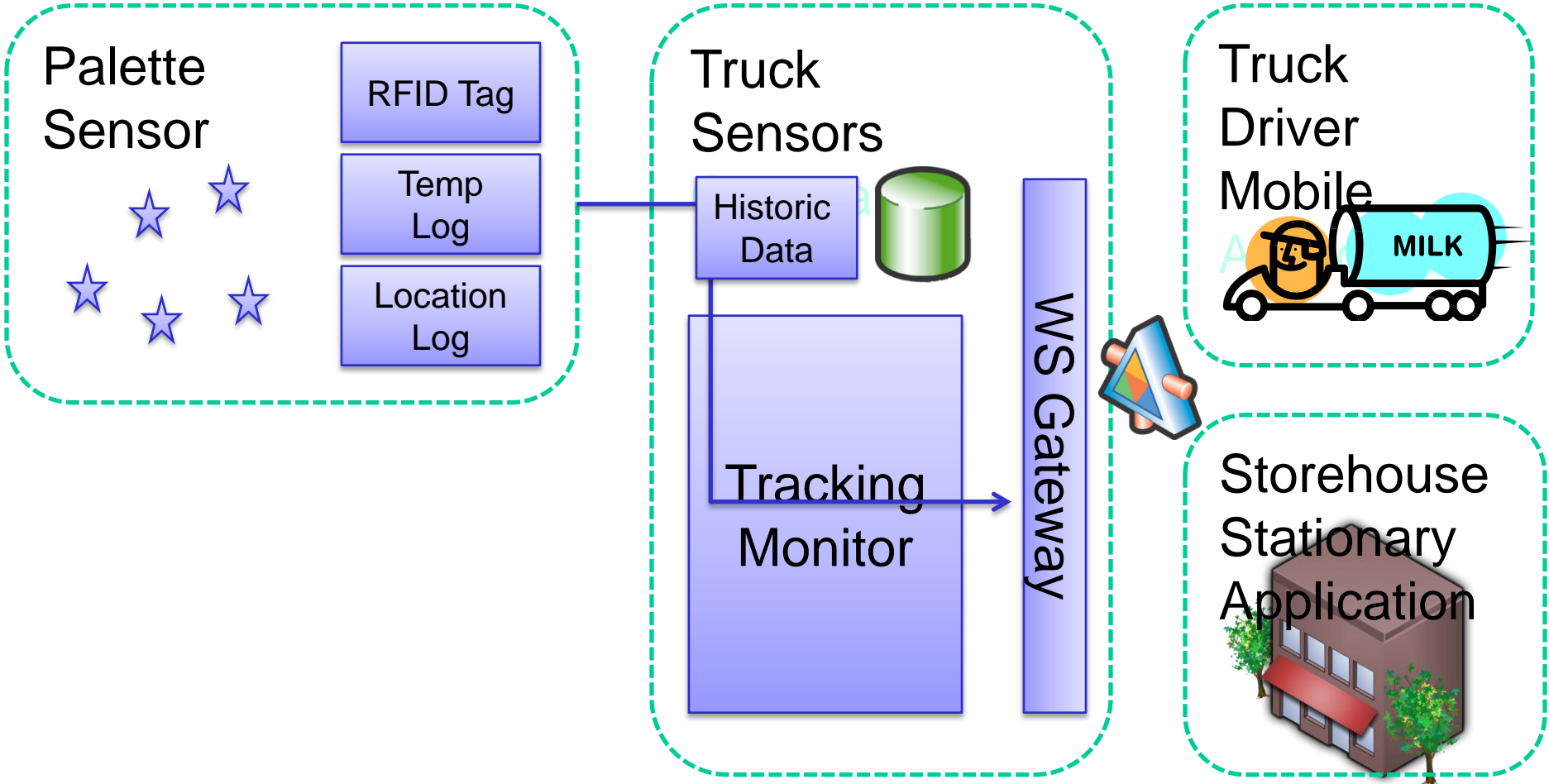
Content

- Background and scenarios
- Motivation
- Integrated .NET programming model
- Integration through web services with enterprise applications
- Conclusions
- Next steps

Background

- Increasing number of “real” business scenarios
 - Supply chain management
 - Home control and energy management
 - Health care
 - Environmental monitoring
- Wide variety of sensors and platforms
 - High degree of heterogeneity
 - Different programming platforms
 - Resource constraint

Track & Trace



Sleep Monitoring

Pressure Foil Sensor

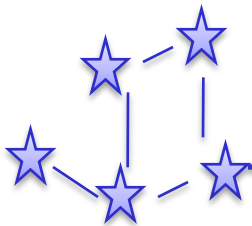


Breathing Analyser

Pulse Analyser

Movement Analyser

Presence Sensors



Sensors Gateway

Historic Data



Sleep Pattern Analyser

Presence Alerter

WS Gateway

Sleep Monitoring Application



Motivation

- Multitude of stakeholders
 - Sensors (body-worn sensors and ambient sensors)
 - Handheld devices and gateways
 - Background infrastructure (servers, databases)
- Basic building blocks
 - Data gathering and storing
 - Data fusion
 - Data presentation/GUI
- Application development is difficult, error-prone, and time consuming
- Integration with backend systems is difficult
- NO support in Microsoft's programming environments for sensors



Objective

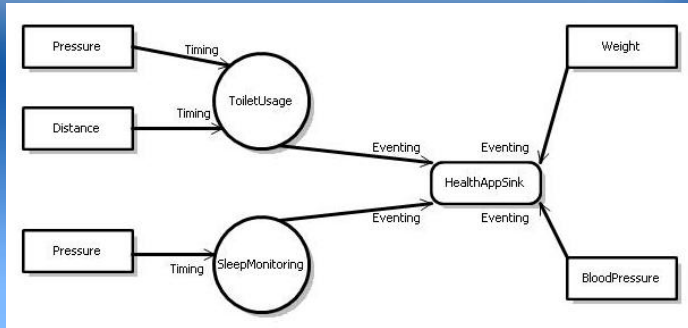
- Develop and integrated, cross-platform .NET programming model for sensor environments
 - Energy Limitations
 - Platform Limitations
- Make sensor data easy to integrate with enterprise applications
- Enable an ambient environment in which sensors do not belong to one single application but are part of the shared resources



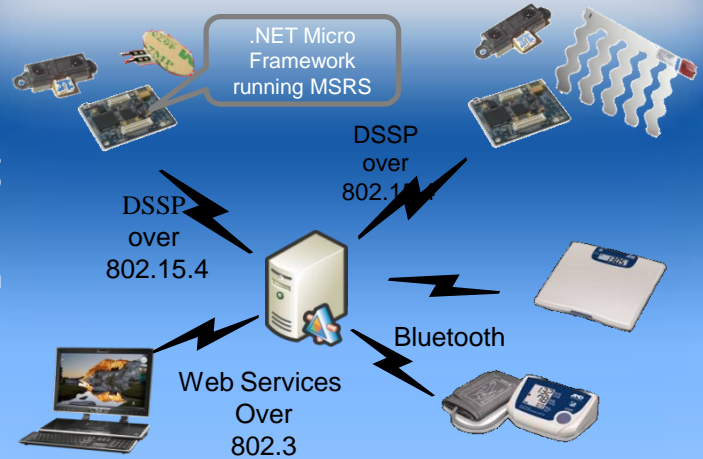
Integration with enterprise applications

- Web Services based Middleware
- Mapping Sensor Protocols to Web Services
 - Expose capabilities of heterogeneous sensor nodes as **Web Services**
 - Get and set sensor properties by means of Web Services
 - **Eventing and notification architecture** based on WCF callbacks and WS-Eventing
 - Discovery of sensor nodes using WS-Discovery
 - Sensor metadata to optimize handling of sensors
- Enabling Web Services over non-IP
 - Bluetooth, 802.15.4
- Integration using IP
 - 6lowPAN / 802.15.4

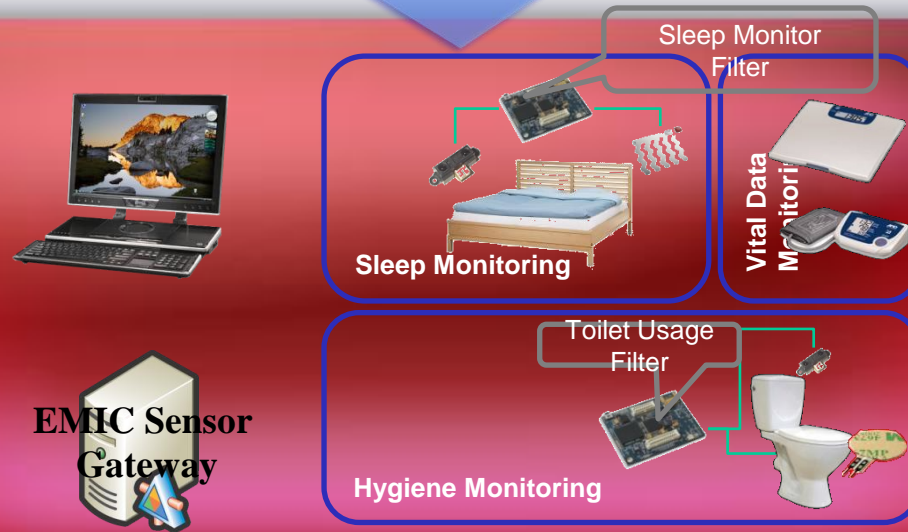
Data Flow



Topology



Deployment



Integrated Programming Model

What?

- Expose unified .NET based programming experience regardless of hardware used
- Abstract from the physical topology of devices
- Automatic code generation and dynamic deployment of sensor code from .NET applications

How?

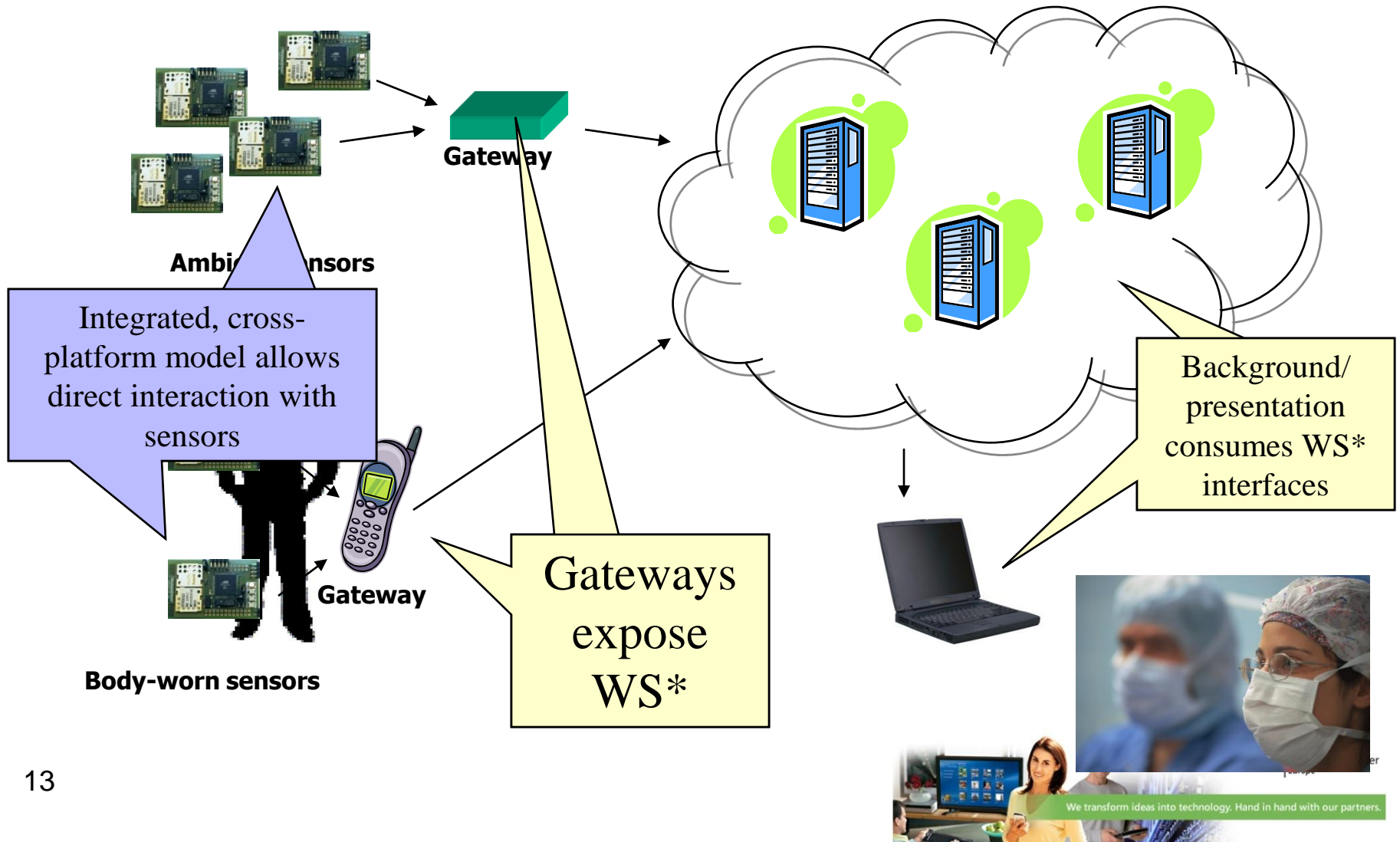
- Programmer describes data flow graph of application
- Programming environment generates code, that can be extended by developer depending on desired application logic
- Automatic deployment on physical nodes

A .NET-Based Programming Platform for Distributed Sensor Applications

Sensor Abstraction Layer (SAL)

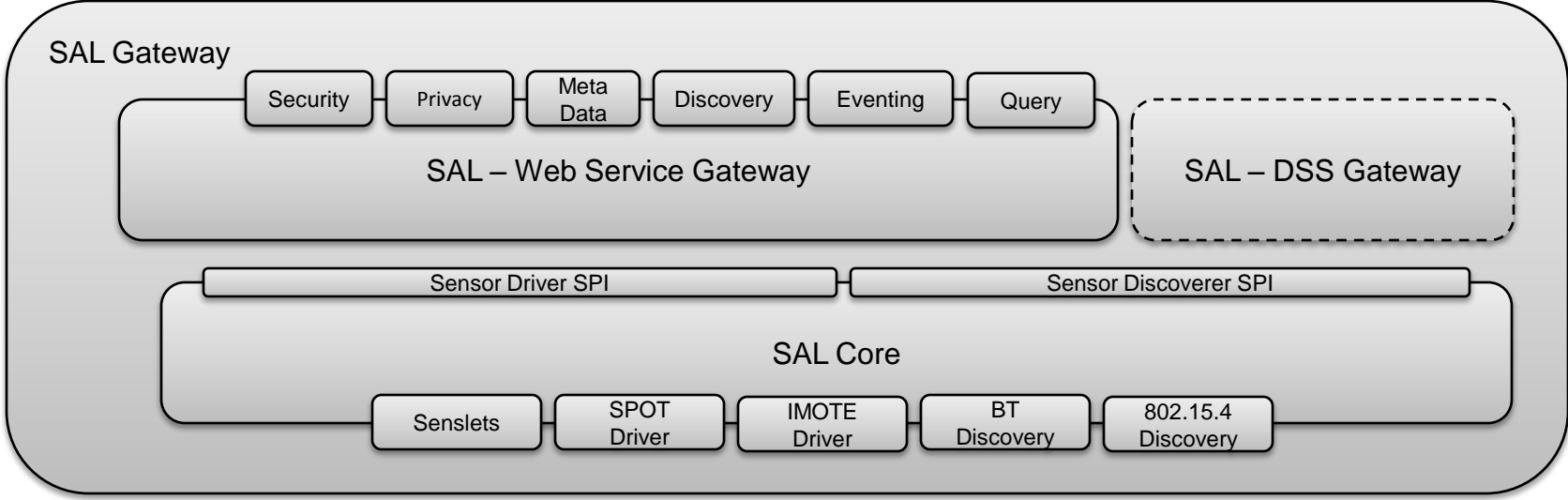
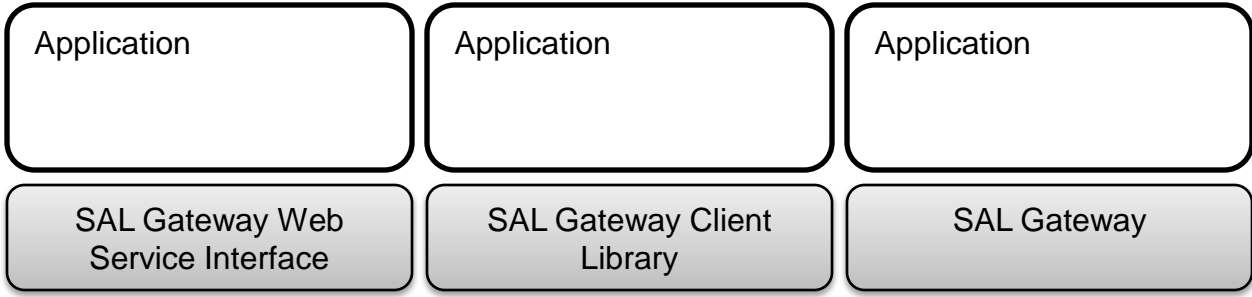
- Installing components of the application data flow graph requires that the node supports the service model and with that a certain programmability
- In case the sensor capability is limited, gateways are used to expose non-programmable sensor nodes using in this case Web Services
- During the deployment process, components in the data flow graph are mapped to specific sensor nodes
- When not programmable, components of the data flow graph are installed on the gateway instead of on the sensor node itself.

High-Level Architecture

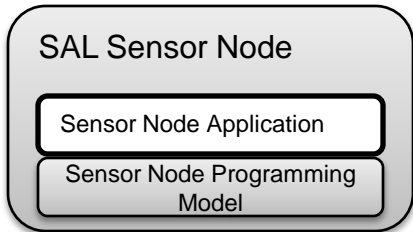
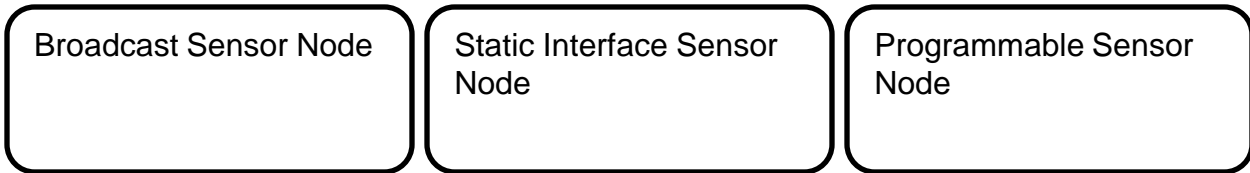


Sensor Abstraction Layer (SAL)

- Installing components of the application data flow graph requires that the node supports the service model and with that a certain programmability by the application programmer.
- In many cases for commercially available sensors this is not true
- In case the sensor capability is limited, gateways are used to expose non-programmable sensor nodes using Web Services. (compare this with drivers for peripherals in windows)
- Expose capabilities of heterogeneous sensor nodes as **Web Services**
- Get and set sensor properties by means of Web Services
- **Eventing and notification architecture** based on WCF callbacks and WS-Eventing
- Discovery of sensor nodes using WS-Discovery
- Sensor metadata to optimize handling of sensors



[ZigBee, BT, 802.15.4
Active Messages, Web Services, sensor specific protocols]



Integrated Sensor Programming Model

Conclusions

- Heterogeneity makes sensor application development difficult
- Integrated, .NET-based development model can help to deal with this problem
- WS*-technologies together with gateways allow the integration of sensors into standard (Microsoft) enterprise applications
- Interfacing sensors directly from .NET possible

*Credits to the emic sensor team:
Alain Gefflaut, Frank Siegemund,
Friedel van Meegen, Louis Latour,
Matthias Neugebauer, Robert Sugar*

Q&A?

Related work

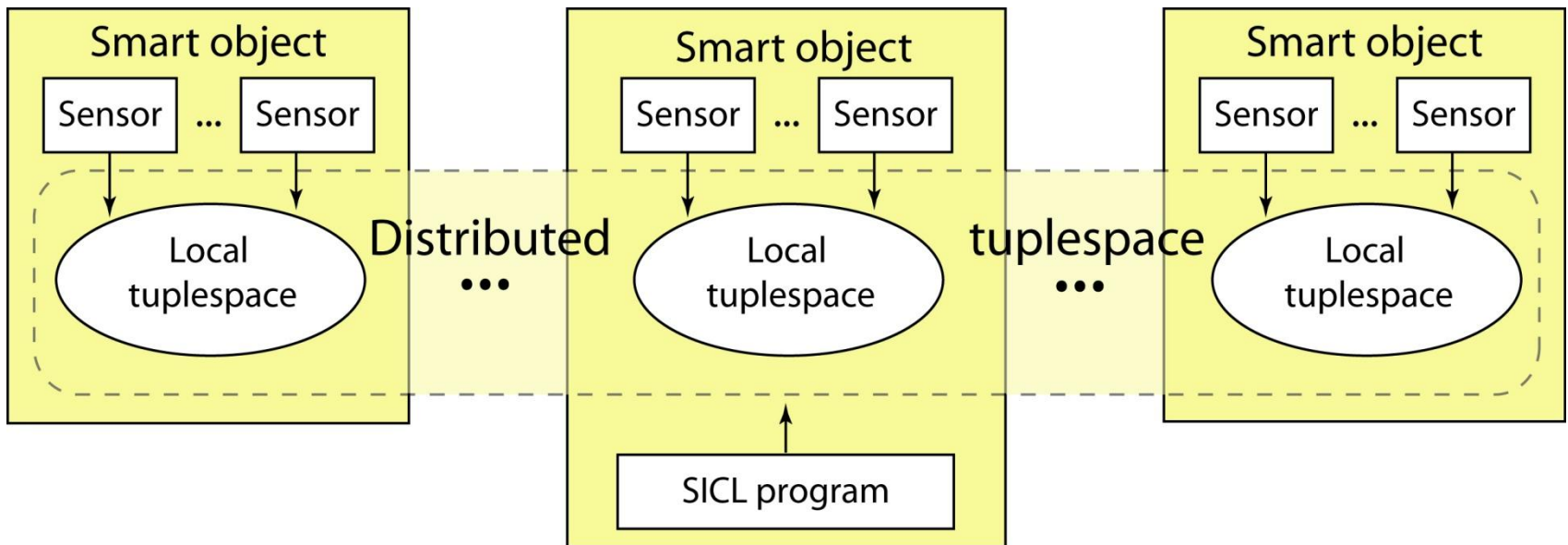
Programming models

- Database-oriented
 - TinyDB, Cougar, SINA
 - Execution plans
 - Network dynamics (?)
 - Mobility and heterogeneity (?)
- Shared dataspace
 - Grouping concept
 - Nice programming abstraction
 - Location where code is executed becomes transparent
 - Distributed Tuple Spaces
 - Domain specific languages (e.g., SICL)



Shared Dataspaces

- Groups of sensors appear to applications as a single entity
- Programming abstraction for implementing collaborative applications



Related work [2]

- Virtual execution systems
 - VM*, PicoJava
 - On-node
 - Good programmer support
 - Hardware abstraction layers
- Mobile code
 - Mate
 - Deployment of code through a network
 - Virtual machine technology
 - High-level instruction set



Related work [3]

- Services
 - Applications composed out of a set of services
 - Dynamic service composition/decomposition
 - Suitable for highly dynamic environment ?
 - Tool support
- Event-based models
 - Contiki, TinyOS provide event-based interfaces
 - Cooperative vs. preemptive multitasking
 - Domain-specific languages



CCR Concepts

- Asynchronous Programming model
 - A concurrency model without manual threading, locks, semaphores, etc.
 - Based on asynchronous message passing
 - Focus on coordination primitives
 - Execution context for services
 - Isolation from infrastructure
 - Isolation from other services



DSS Concepts

- Application Model
 - Simple, flexible, and REST based
 - Coarse grained and loosely coupled
 - Compatible with existing Web infrastructure
- Programming Model
 - Asynchronous
 - Not just another pipeline model
 - Focus on coordination

