Sensor Network Research at IBM

Dr. John Davis
IBM T.J. Watson Research Center
davisjs@us.ibm.com
Overview

- **Traditional IBM Business**
  - A large part of IBM’s business: back-end data processing, middleware
    - Large-scale databases
    - Enterprise messaging
    - Data mining
    - Application servers
    - Business workflow management
    - E-Commerce
    - ...
    - *and* Development tools for the above

- **New Challenges**
  - Sensor networking means small devices will generate the data to be processed
  - Two Example Research Projects
    - RISE – development/management tool for RFID solutions
    - A messaging interface for sensor networks
Computing at the Edge

Typical e-Commerce Application

P Cs

Network

Edge Server
Load Balancing

Back-End Server

Application Server

Database

Typical “Sensor” Application

Tremendous Amount of Information

Edge Server
Load Balancing

Back-End Server

Network

Pipeline processing

Application Dependent processing
RISE: RFID Integrated Solution Enablement

- Drastically simplify the development, validation, deployment, and customization of RFID solutions
- Reduce solution TCO for clients to accelerate technology adoption
- Enable an efficient and open ecosystem for RFID solutions to promote industry growth

**Approach**

- Provide integrated development and runtime support for solution life-cycle management
- Build on a model-driven methodology that exploits graphical, component composition with well-defined component interaction semantics
- Provide interfaces and protocols for user customization and remote management of solutions
- Facilitate component sharing and reuse through open interface and domain polymorphism

**Research Focus**

- Build overall architecture and PoC
- Assess the applicability of models of component interaction for the RFID application domain
- Evaluate and validate methodology through user studies
- Investigate performance evaluation and optimization mechanisms
- Study mechanisms for coordinating and managing distributed runtimes across multiple solutions

**Technology Leverages**

Ptolemy II (UCB), Eclipse, OSGi, Java, XML
Position in the RFID Component Model

- Tagged Object Domain
- Antenna & Reader Domain
- Edge Domain
- Premises Domain
- Business Process Integration Domain
- Enterprise & Business Application Domain
- Object Directory Domain

Integrated RFID Solution Enablement

Security & Privacy Management

T  Tooling – support for customized business logic
Semantics Considerations

- **Web Services - BPEL**
  - Focuses on messages between web services
  - Ignores the internal semantics of web services
    - Web services are *stateless*; computation is logically instantaneous
    - Message dataflow is sole consideration

- **Sensor & Actuators**
  - Focuses on representation of S&A devices
    - S&A devices are *stateful*

- **S&A-based Adaptive Control**
  - Web Services Interfaced with S&A
    - Semantics Heterogeneity is needed
    - Multi-level semantics
Model-driven component-based software development

- Components with well-defined interface as development/deployment units.
  - Software reuse
  - Device polymorphism
- Hierarchical development using a hybrid data/control flow modeling methodology
  - Data flow is modeled by component composition diagram
  - Control flow is modeled by finite state machine
  - Composition diagrams are switched based on the state of the controller
End-to-end life-cycle management of components

- **Creation**
  - Component code is generated from model.
  - Library is an augmented OSGi bundle. It contains multiple components
  - Library is the unit of publishing and deployment.

- **Deployment**
  - Runtime dynamically downloads components from sources, e.g. library servers.

- **Testing (wip)**
  - Live feedback of runtime monitoring events to enable debugging and profiling.

- **Management (wip)**
  - Runtime can be remotely configured, e.g., via HTTP.
RISE Architecture and Components

Integrated Development Environment
- Composition Editors
- Wizards
- Views & Perspectives
- Class Builders
- Bundle Server Interface
- Simulation Engine
- Remote Runtime Interface

Reusable Libraries & Execution Flow Components
- RISE Libraries
  - Actor Libraries
  - Solution Libraries
- Execution Flow Components
  - Bootstrapping Agent
  - Factory
  - Execution Engine

Middleware Technologies
- Eclipse 2.1.2 & Related Technologies
  - SWT
  - PDE
  - EMF
  - GEF
- Vendor Middleware
  - Device Drivers
  - Communications
  - Data Access
  - Event Correlation
  - Presentation
- Ptolemy
  - Component Composition
  - Runtime Directors

Platform Framework
- J9 Runtime (RM configuration)
- Service Management Framework
- SMF Services (Configuration, Log, HTTP, etc.)

Platform Software
- Views & Perspectives
- Editors
RISE IDE

- Runs on J2SE (Java2 Platform Standard Edition)
- Based on open source/standard projects including Eclipse/EMF/GEF and OSGi

RISE library:
Set of components that can be employed by drag-and-drop

A solution model
The Use of Messaging Protocols

- **Key design element:** *build a chain of loosely coupled elements*
  - Elements can come and go without affecting the overall system
  - Brings ease of configuration
- **Publish/Subscribe protocols provide the right abstraction**
  - Producers and consumers are ‘matched’ through their interests: *topics*
- **Channel elements provides various semantics**
  - Queues provide a “consume” semantic (e.g. MQ, Tuple-Spaces)
  - Brokers provide a “multicast” semantic (e.g. MQBroker, CORBA, WebServices notifications, Siena)

![Diagram of messaging protocols]

- Light Weight Protocol: MQTT
  - Point-to-Point and relies on TCP/IP
  - Microbroker
- Enterprise Protocol: MQ
  - Heavy but transactional Integration Broker

**Sensors/Actuators** ➔ **Gateways** ➔ **Edge Servers** ➔ **Back-End Server**
MQTT and Sensor Networks

- Protocol highlights
  - The acronym stands for MQ Telemetry Transport
  - Simple brokered messaging protocol
  - It operates on top of TCP as it requires reliable and in-order packet delivery
  - Supports three types of QoS:
    - QoS 0: at most once delivery
    - QoS 1: at least once delivery
    - QoS 2: exactly once delivery
  - Supports some form of “last will” mechanisms

- Is it adapted to sensor networks?
  - A bit yes 😊 but mostly no 😞
  - Yes: as it is relatively “light weight”
  - No:
    - Relies on TCP/IP
    - It is a point-to-point protocol
Pub/Sub With Sensor Networks

- In sensor networks there is no real point-to-point connection between sensors and a broker. Therefore MQTT (Direct) cannot be used as such.
- Our approach is to consider the sensor network — as a whole — a publisher and/or a subscriber
- We have developed two protocols:
  - Messo: Publish-only protocol (i.e. sensor to broker)
    - Very simple protocol
    - Relies a on the underlying routing spanning tree (TOS or ZigBee)
    - Reliability based on underlying MAC
  - Preso: Subscribe-only protocol (i.e. broker to actuator)
    - Relies a on the underlying routing spanning tree (TOS or ZigBee)
    - Reliability based on own retransmission protocol
- These protocols have been implemented on TinyOS.
Messo

The nodes periodically send a PUBLISH-SET to the root describing the topics they can publish on. Parent nodes combine the subscribe-set of their children with that of their own before forwarding it on. Nodes learn about the topic their descendants can publish on and this is retained as soft state.

The broker periodically transmits the SUBSCRIBE_SET of topics that subscribers are currently interested in. This message is broadcast down the tree, but only repeated by nodes whose PUBLISH_SET has a non null intersection with the SUBSCRIBE_SET.

Nodes that can publish on a topic contained in the SUBSCRIBE_SET start to publish, all other nodes remain silent.
Nodes state the topic that they are interested in and ancestors remember those of their descendents.

Transient loss is handled through ACKs. Parent ACKs when all children have ACKed. Resend if timeout. Send error message to root after N resends.
If no ACK is received after a given number of attempts a PUBLISH_FAILED message is sent to the broker. The broker stores the message corresponding to the error sequence number.

If node 1 reconnect the mismatch in sequence numbers is detected and propagated to the broker. The missing message(s) can be resent.
# A Family of Protocol Stacks for the Microbroker

<table>
<thead>
<tr>
<th>Dispatcher</th>
<th>MQTT</th>
<th>MQTT</th>
<th>MQTT</th>
<th>Messo Preso</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCP Adapter</td>
<td></td>
<td></td>
<td></td>
<td>TOS</td>
</tr>
<tr>
<td>Stream Socket</td>
<td></td>
<td>Datagram Socket</td>
<td></td>
<td>PPP-like Framing</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Error Detection Correction</td>
<td></td>
<td>Serial Line Stream</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Segment Reassembly</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Segmentation Reassembly</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Serial Line Stream</td>
</tr>
</tbody>
</table>
Prototype on Crossbow Motes

Mote

Application

Sensor Based Messaging

TOS

Radio Framing

Radio

Serial Gateway

Serial Line

PPP-like Framing

Radio Framing

Relaying Gateway

TOS

TOS

PPP-like Framing

Application

Pub/Sub Engine

Sensor Based Messaging

TOS

PPP-like Framing

Serial Line

Microbroker

Current Subscriptions
Contacts

- RISE
  - Paul Chou, pchou@us.ibm.com
  - Johnathan Reason, reason@us.ibm.com
- MQ on Motes
  - Paolo Scotton, psc@zurich.ibm.com