WOA 2010 – Undicesimo Workshop Nazionale "Dagli Oggetti agli Agenti"

Rimini, 6-7 Settembre 2010

Programming Wireless Body Sensor Network Applications through Agents

Giancarlo Fortino, Stefano Galzarano

fortino@unical.it, galzarano@si.deis.unical.it

UNIVERSITÀ DELLA CALABRIA

Dipartimento di ELETTRONICA,
INFORMATICA E SISTEMISTICA

PLASMA Group

Outline

- Introduction
- Programming WSN
- Agent frameworks for WSN
- A real application: Human Activity Recognition System
- Conclusions

Introduction

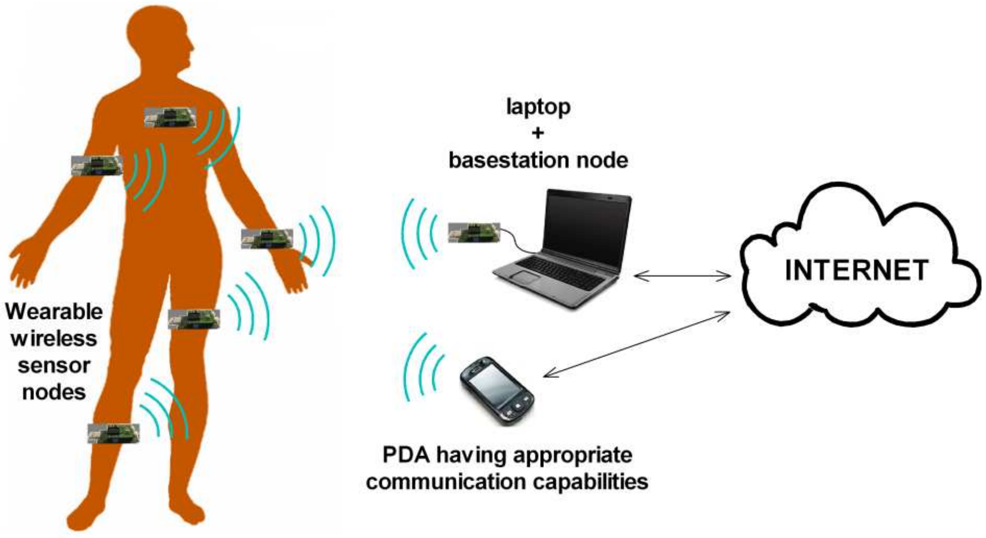
- Wireless Sensor Networks (WSNs) are collection of tiny, low-cost devices with sensing, computing, storing, communication and potentially actuating capabilities.

- WSNs are a powerful technology for supporting a variety of high-impact applications in a wide range of domains including:
  - health-care
  - environment and infrastructures monitoring
  - smart home automation
  - emergence management
  - military support

Introduction

- WSNs applied to the human body are usually called Wireless Body Sensor Networks (WBSNs).

- WBSNs can be very effective for providing continuous monitoring and analysis of physiological or physics parameters aiming at improving the quality of life of human beings by enabling continuous and real-time non-invasive medical assistance at low cost.


laptop
+
basestation node
INTERNET
Wearable wireless sensor nodes
PDA having appropriate communication capabilities

Programming WSN

- Unfortunately, designing application for such networks is not an easy work because it implies knowledge from many different areas, ranging from low-level aspects of the sensor nodes hardware and radio communication to high-level concepts concerning final user applications.

- Framework supporting high-level abstraction model can be adopted for addressing these programming problems and assisting users in a fast and effective development of applications.

- Programming abstractions definition is one of the most fermenting research areas in the context of sensor networks.

Programming WSN

- The basic functions required by high-level programming tools are (1) to provide standard system services to easily deploy current and future applications and (2) to offer mechanisms for an adaptive and efficient utilization of system resources.

- Many different solutions have been proposed in the last years, differing on the basis of the model assumed for providing the high-level programming abstractions:
    - Database
    - Macroprogramming
    - Agent-based
    - Virtual machine
    - Application-driven

Agent & WSN

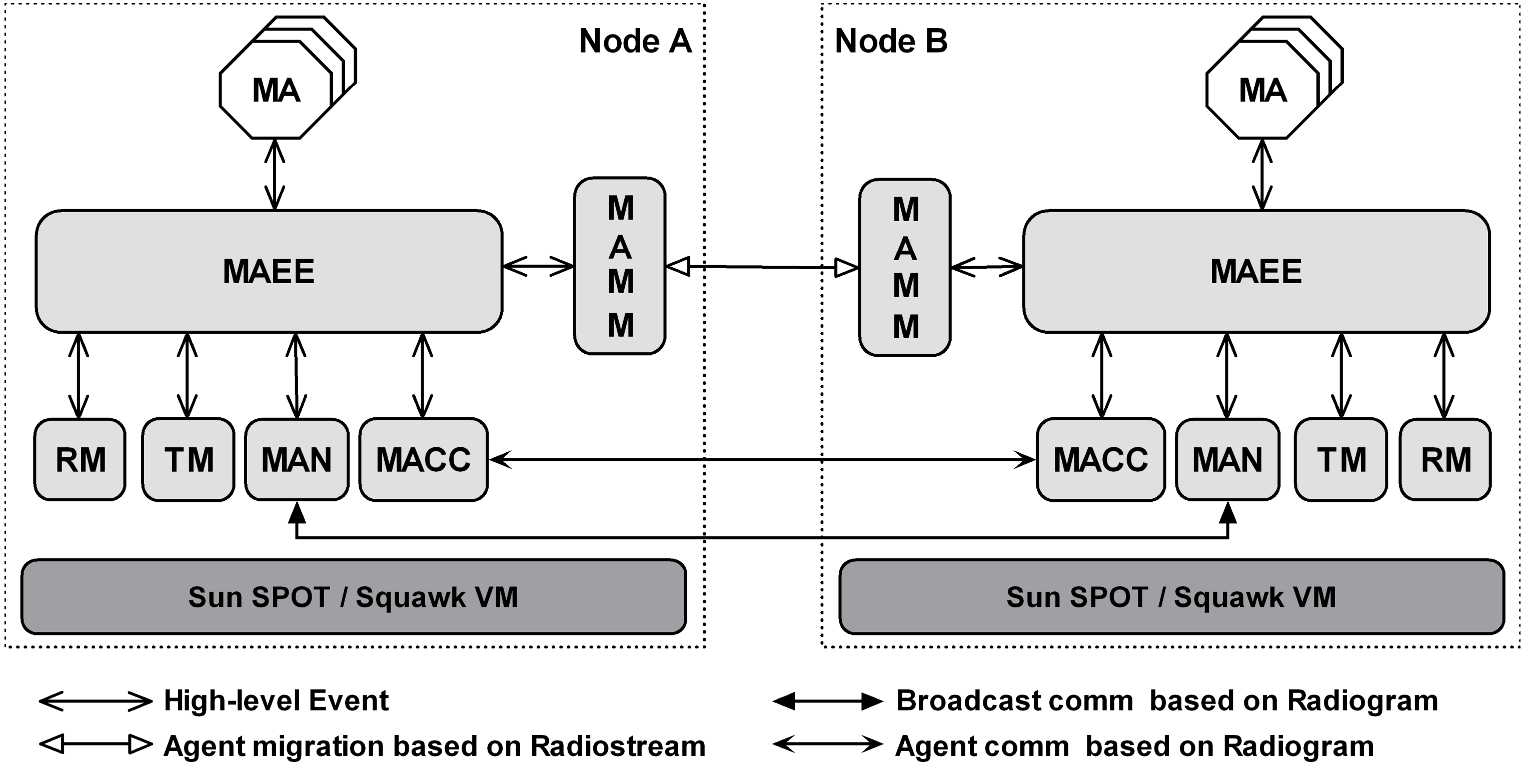
- Among the programming paradigms proposed for the development of WSN applications, the **mobile agent-based paradigm**, which has already demonstrated its effectiveness in conventional distributed systems as well as in highly dynamic distributed environments, can effectively deal with the programming issues that WSNs have posed.

- Agent-based frameworks for WSN:
  - Agilla (TinyOS sensor platforms)
  - ActorNet (TinyOS sensor platforms)
  - AFME (MIDP-compliant devices)
  - MAPS (specifically for SunSPOT sensor nodes)

MAPS: Mobile Agent Platform for Sun SPOTs

- Innovative Java-based framework expressly developed on Sun SPOT technology for enabling agent-oriented programming of WSN applications.
  - Component-based lightweight agent server architecture to avoid heavy concurrency and agents cooperation models.
  - Minimal core services involving agent migration, agent naming, agent communication, timing and sensor node resources access (sensors, actuators, flash memory, and radio).
  - Plug-in-based architecture extensions through which any other service can be defined in terms of one or more dynamically installable components implemented as single or cooperating (mobile) agents.
  - Use of automaton for defining the mobile agent behavior.

MAPS: Architecture



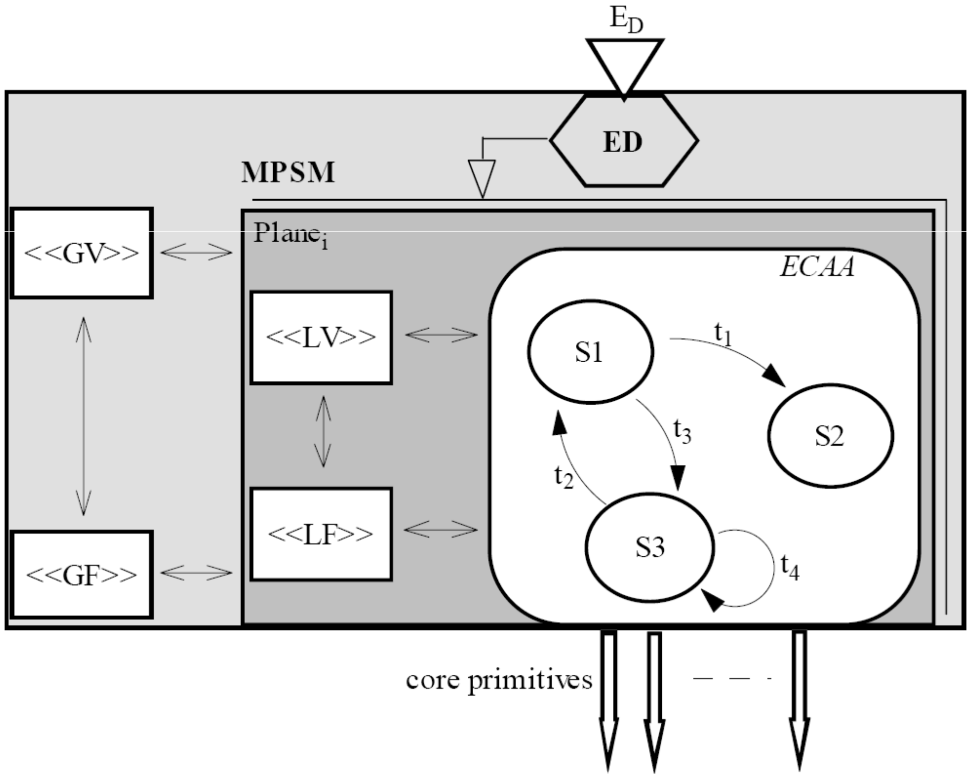
MA - Mobile Agent
MAEE - Mobile Agent Execution Engine
MAMM - Mobile Agent Migration Manager
MACC - Mobile Agent Communication Channel
MAN - Mobile Agent Naming
RM - Resource Manager TM - Timer Manager

Agent definition in MAPS

- MAPS agent are modeled as a multi-plane state machine (MPSM) communicating through events.
- Each plane may represent the behavior of the MA in a specific role so enabling role-based programming.
- A plane is composed of local variables, local functions, and an automaton whose transitions are labeled by Event-Condition-Action (ECA) rules *E[C]/A*, where E is the event name, [C] is a boolean expression evaluated on global and local variables, and A is the atomic action.

Agent definition in MAPS

The MAPS agent model



GV: global variables

GF: global functions

LV: local variables

LF: lcal functions

ECAA: ECA-based Automaton

ED: event dispatcher

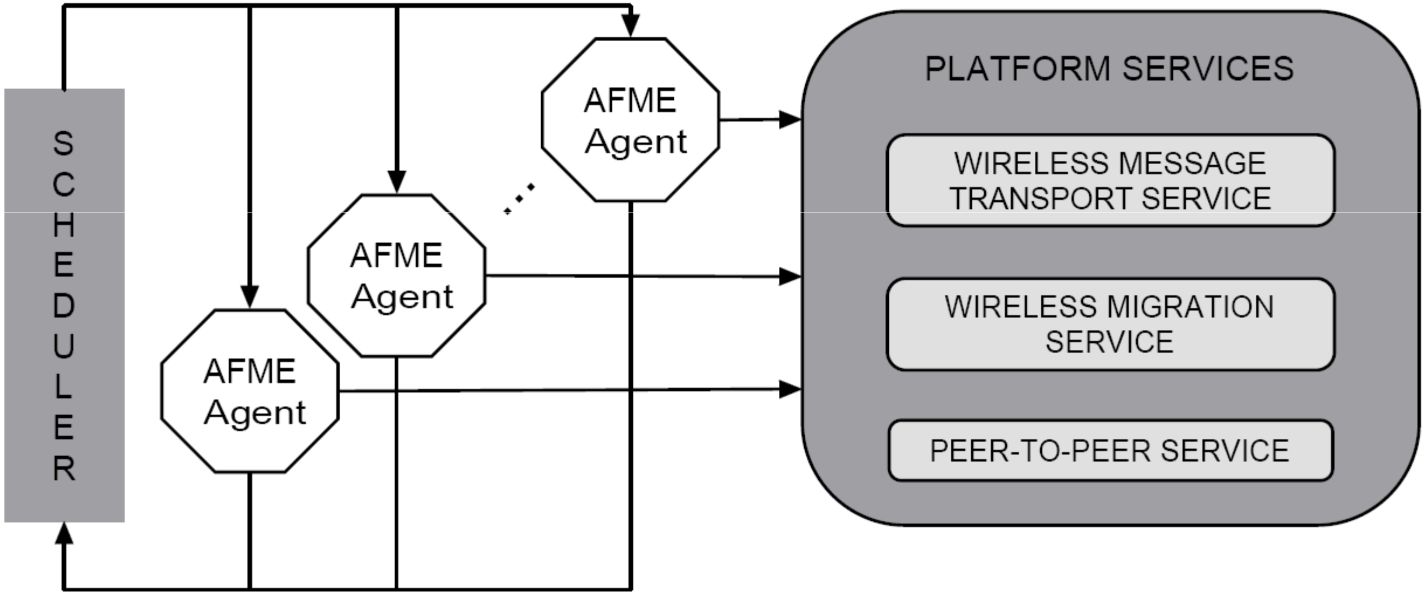
AFME: Agent Factory Micro Edition

- Open-source lightweight J2ME MIDP compliant agent platform based upon the preexisting Agent Factory framework and intended for wireless pervasive systems.

- Thanks to a recent support of J2ME onto the Sun SPOT sensor platform, it can be adopted for developing agent-based WSN applications.

- Agents are defined through a mixed declarative/imperative programming model.

AFME: Architecture



Agent definition in AFME

- AFME is strongly based on the *Believe-Desire-Intention* (BDI) paradigm, in which agents follow a sense-deliberate-act cycle.

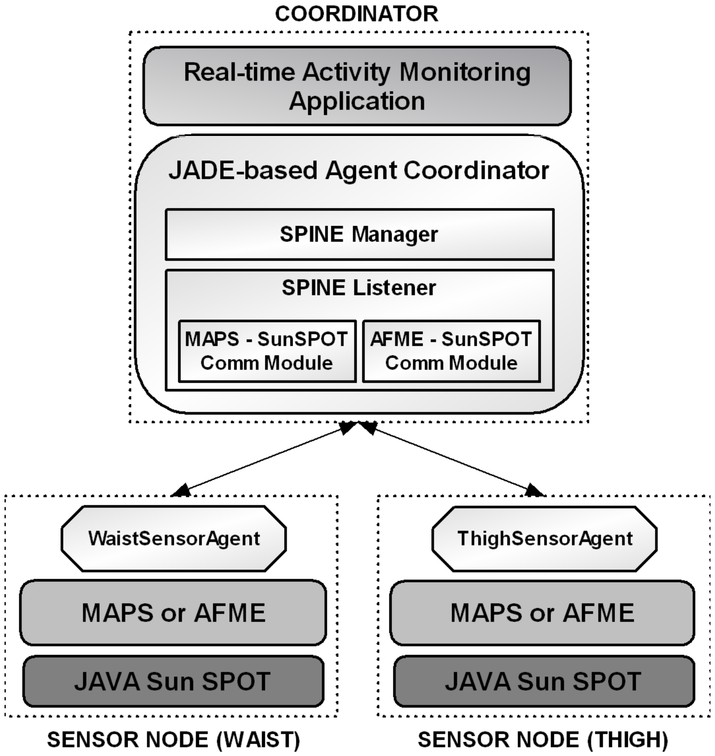
- Agents are defined through a mixed declarative/imperative programming model.
  - The declarative part is used to encode an agent's behavior by specifying rules defining the conditions under which commitments are adopted. A rule is expressed in the form ***b1, b2, …, bn > doX*** where b1… bn represent beliefs, whereas doX is an action.
  - The imperative Java code is instead used to encode perceptors and actuators.

- The framework supports a number of system components which developers have to extend when building their applications: *perceptors, actuators, modules,* and *services*.

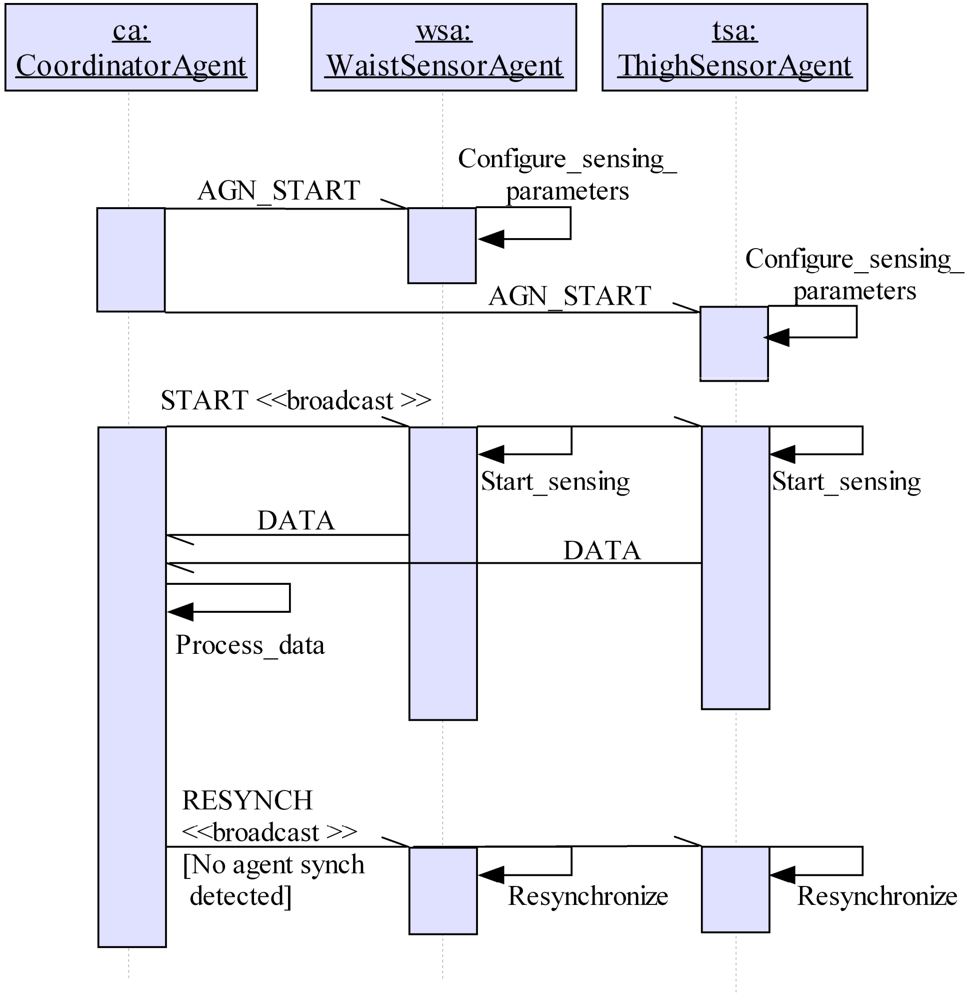
Agent-based Human Activity Monitoring System

- Signal processing in-node system specialized for real-time human activity monitoring able to recognize postures (e.g. lying down, sitting and standing still) and movements (e.g. walking) of assisted livings.

- The system is constituted of one coordinator (laptop) and two sensor nodes (Sun SPOTs positioned on the thigh and on the waist).

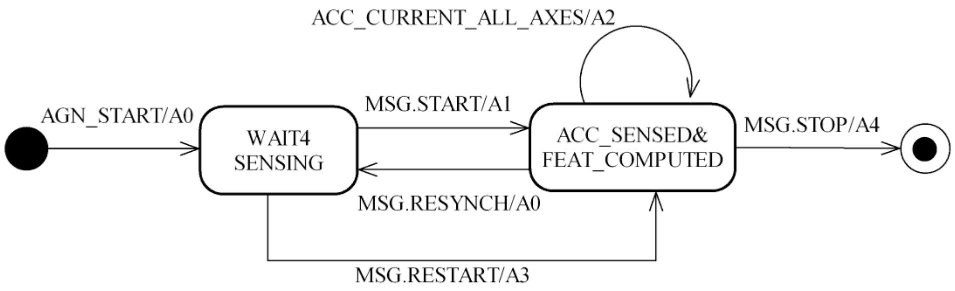


Agent-based Human Activity Monitoring System



WaistSensorAgent in MAPS

Agent behavior modeled through a 1-plane state machine



```
Actions
A0: initVars();
    initSensingParamsAndBuffers(event);
A1: timerSetForSensing();
    doSensing();
A2: bufferFilling(event);
    sampleCounter++;
    nextSampleIndex=(nextSampleIndex+1)%W;
    if (sampleCounter==S){
     sampleCounter==0;
     copySensingBuffersIntoBuffersForComputingFeatures();
     computeFeatures();
     transmitFeaturesComputed();
    }
    timerReset();
    doSensing();
A3: timerDisabling();
    initVars();   A1;
A4: timerDisabling();
```

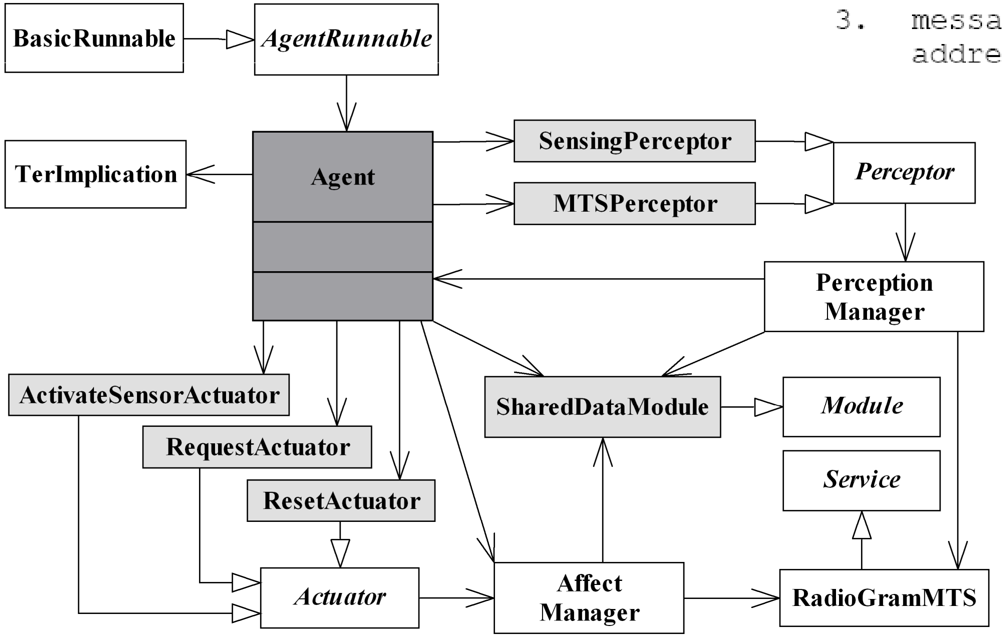
```
LF
initVars():
sampleCounter=0; nextSampleIndex=0; agent.timestamp=0;

initSensingParamsAndBuffers(Event event):
  (WaistSensorAgent)agent.basestationAddress=event.getParam(
                                 "BASESTATION_ADDRESS");
  W=Integer.parseInt(event.getParam("WINDOW_SIZE"));
  S=Integer.parseInt(event.getParam("SHIFT_SIZE"));
  ST=Integer.parseInt(event.getParam("SAMPLE_RATE_MS"));
  windowX = new double[W]; windowY = new double[W]; windowZ= new double[W];
  (WaistSensorAgent)agent.windowX4FE = new double[W];
  (WaistSensorAgent)agent.windowY4FE = new double[W];
  (WaistSensorAgent)agent.windowZ4FE = new double[W];
computeFeatures():
  resultsX = meanMaxMin((WaistSensorAgent)agent.windowX4FE);
  resultsY = meanMaxMin((WaistSensorAgent)agent.windowY4FE);
  resultsZ = meanMaxMin((WaistSensorAgent)agent.windowZ4FE);
```

WaistSensorAgent in AFME

Agent behavior modeled through AFME components and rules

```
1.   message(inform, sender(BaseStation,
     addresses(BSAddress)), begin) > activateSensors(1);
2.   sense(?val), !message(inform, sender(BaseStation,
     addresses(BSAddress)), resynch)  >
     request(agentID(BaseStation,
     addresses("radiogram://"+BSAddress)), ?val);
3.   message(inform, sender(BaseStation,
     addresses(BSAddress)), resynch) > reset;
```



Conclusions

- In this paper we have presented the agent-oriented approach for high-level programming of WBAN applications.
- The agent approach is effective during the design and the implementation of a WBAN application.
- The higher level software abstractions provided by MAPS and AFME are both suitable for a fast and easy WBSN applications development as demonstrated by the proposed case study concerning a real-time human activity monitoring system.
- However, MAPS offers an easier agent modeling approach for non agent-skilled developers.

Thanks…