An Ubiquitous Multi-Agent Mobile Platform for Distributed Crowd Sensing and Social Mining

Stefan Bosse*
University of Bremen
Department of Mathematics & Computer Science
ISIS Scientific Centre for Sensorial Materials
Bremen, Germany
* Presenting Author

Evangelos Pournaras
ETH Zürich
Chair of Sociology
Modeling and Simulation
Zürich, Switzerland

Last mod. August 23, 2017 11:16 pm / ficloud2017_pres.fm / 28 pages

Overview

A. Ubiquitous Computing and Crowd Sourcing

B. Virtual Sensors and Agents

C. Nervousnet: Crowd Sourcing Framework

D. JAM: The JavaScript Agent Machine Platform

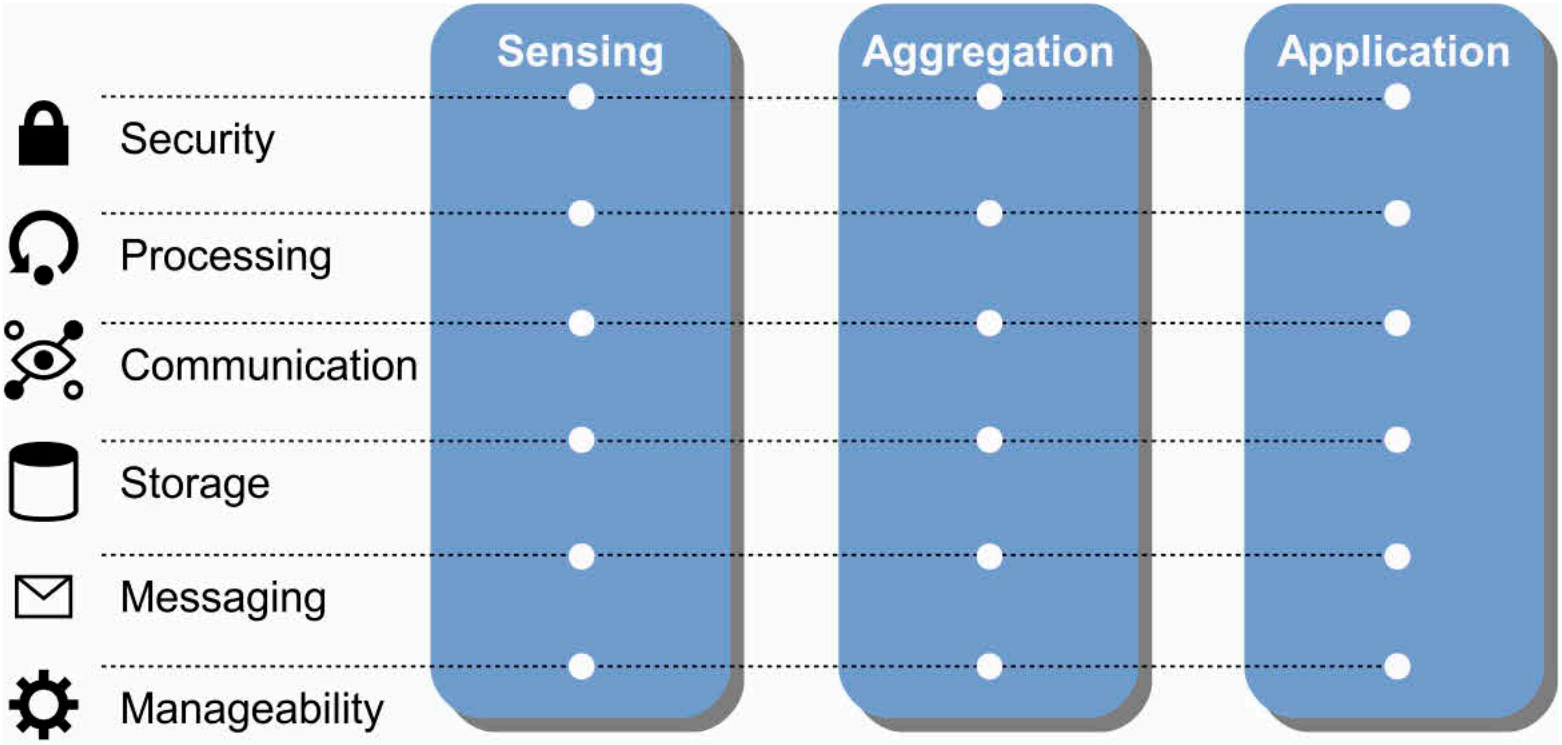
E. Case Study and Simulation: Crowd Sourcing with Agents

F. Conclusions

A. Ubiquitous Computing and Crowd Sourcing

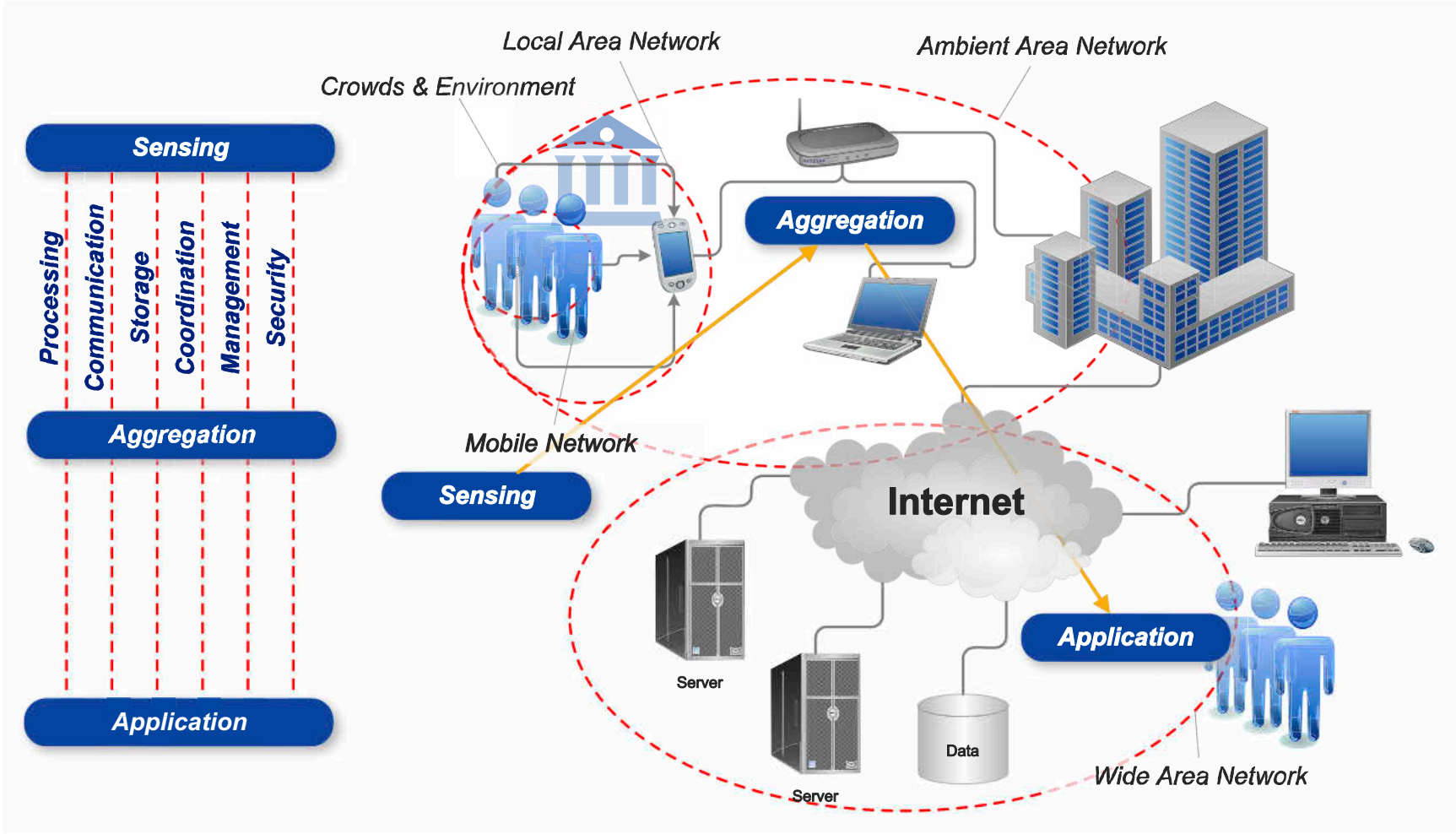
Sensing Process

- A sensing system consists of three vertical functional layers:
  - Sensing, Aggregation, Application ➠ SAA Architecture
- All functional layers consist of six horizontal operational layers:
  - Security, Processing, Communication, Storage, Messaging, Manageability



Sensing Process (cont.)

Sensing ➠ Aggregation ➠ Application Layers: Crowd Sensing and Sourcing



Mobile Networks

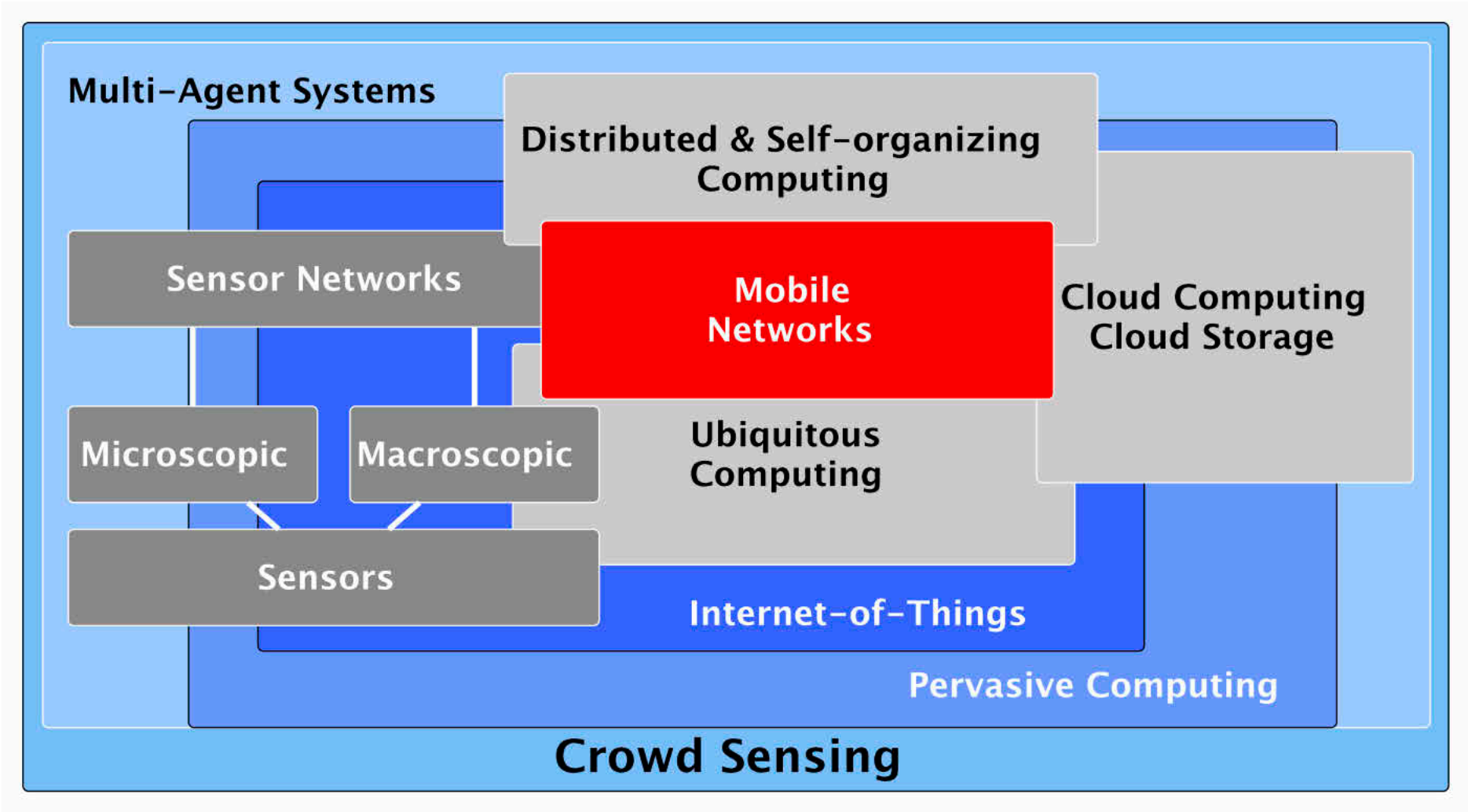
- Mobile networks, wearable devices, and the Internet-of-Things get more and more pervasive in today's digital society
- Billions of devices can contribute to pervasive and ubiquitous computing, forming one big machine

Constraints

- Dynamic Network Structures (in temporal and spatial domain)
- Connectivity
  - Ad-hoc
  - Unicast: Point-to-Point
  - Multicast (groups)
  - Internet (virtual networking)
- Heterogeneous Devices (different processing capabilities, networking, ..)
- Heterogeneous and unreliable Sensors (not calibrated, noise, ..)

Ubiquitous Computing

The Future of Ubiquitous and Pervasive Computing: Crowd and Thing Sensing



Future ICT

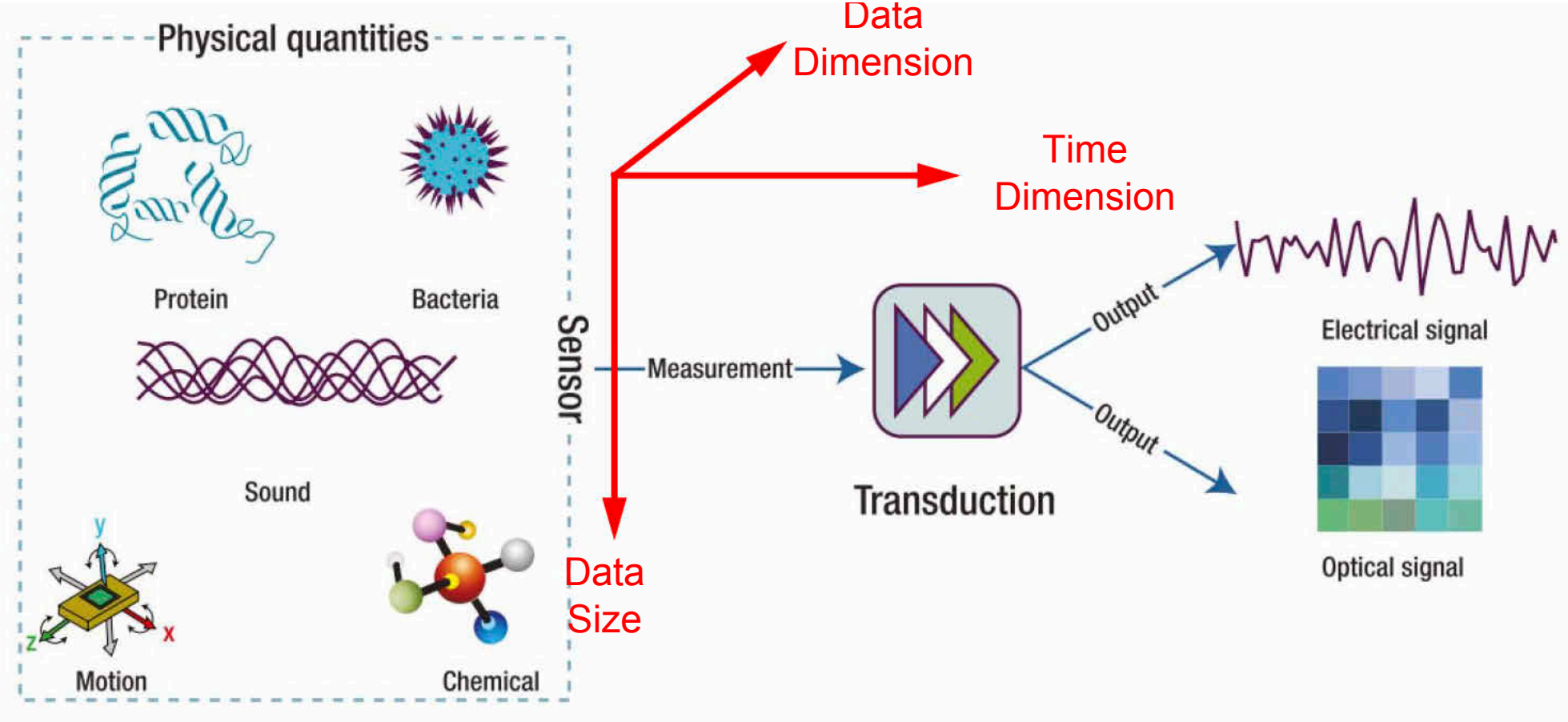
- New Information and Communication Technologies are required!
- One unified approach ➠ Deployment on all functional/operationsl layers and host platforms
- Features and Requirements:
  - Robustness + Reliability
  - Soft Computing / Computational Intelligence ➠ Distributed Intelligence
  - Self-*: Self-Organization / Self-Adaptivity / Self-Awareness
  - Processing on heterogeneous host platforms (mobile, embedded, server)
  - Learning

Multi-Agent Systems can offer all these features and fulfil the requirements!

B. Virtual Sensors and Agents

Physical Sensors

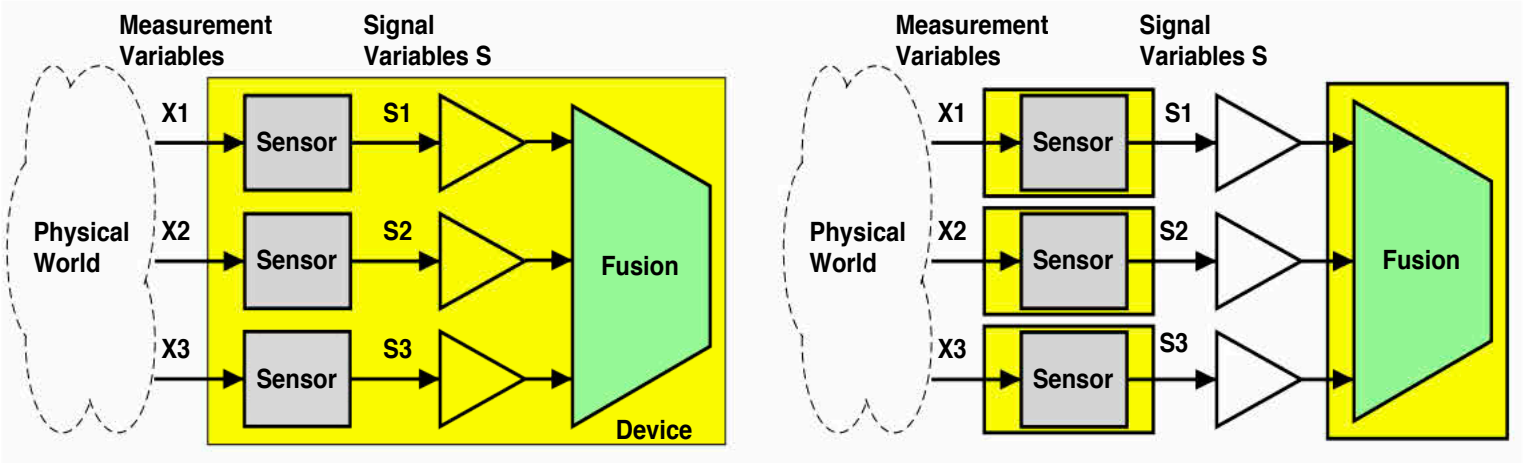
» Broad diversity of Sensors and Sensor Information for Processing [kind, size, dimension]



[McGrath et al., ST, 2014]

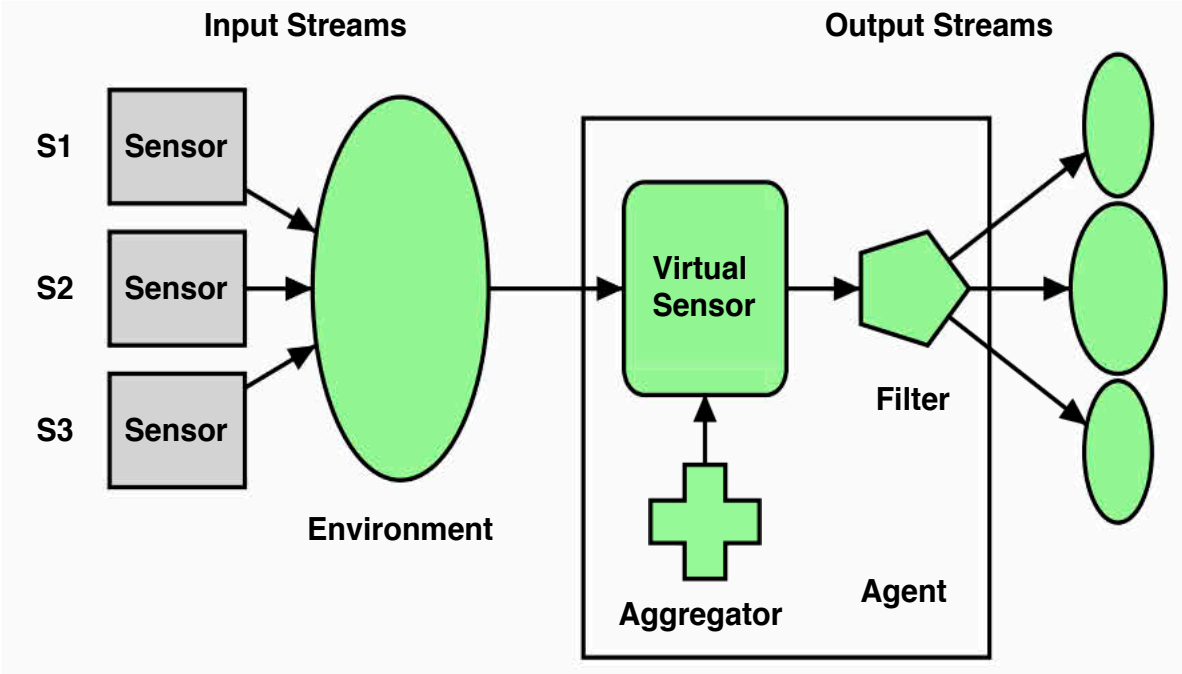
Sensor Fusion

- Physical sensors are unreliable and noisy depending on device class:
  - Measuring Devices
  - Consumer Devices
- Fusion of multiple sensors can increase accuracy and certainty by using:
  - Sensors measuring the same physical quantity (mean filter)
  - Sensors measuring different physical quantities (e.g., strain + temperature)
  - Distributed correlated Sensors in a region
  - Distributed uncorrelated but tagged Sensors (spatially/user related)



Virtual Sensors and Agents

- The environment of a sensor is a set of input streams of data generated from physical or virtual sensors.
- The environment defines the context within the virtual sensor operates.
- The aggregator processes the input streams ➠ Fusion
- The filter produces a set of output streams
- ➠ Agents can represent and implement virtual sensors!

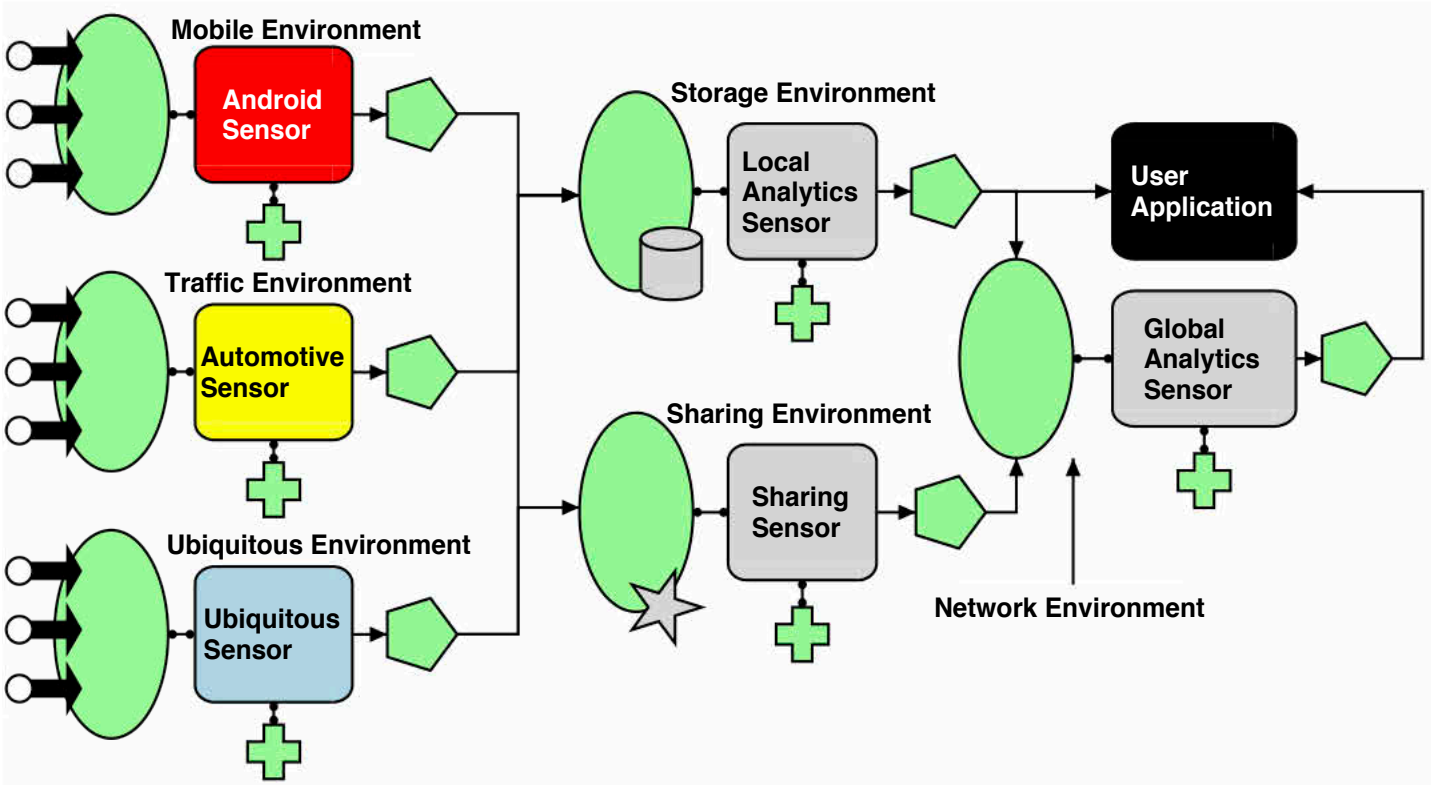


C. NervousNet: Crowd Sourcing Framework

NervousNet

NervousNet: The Planetary Nervous System (Pournaras)

- Hierarchical Structure based on Virtual Sensors featuring:
  - SAA-Architecture, Privacy Control, Storage, Com., Local & Global Analytics
  - Heterogeneous Host Platforms (Mobile, Embedded, Server)



NervousNet: The Agent Approach

» Available Sensors:

⟨ *Position*, *Connectivity*, *Context*, *Temperature*, *Accelaration*, *Sound*, *Light* .. ⟩

» Currently the NervousNet software must be installed on all devices:

- » Strong dependency on underlying platform and communication network
- » Difficult to maintain / update; binary releases for each device/platform ..
- » One Virtual Sensor is one device/program & Client-Server Architecture

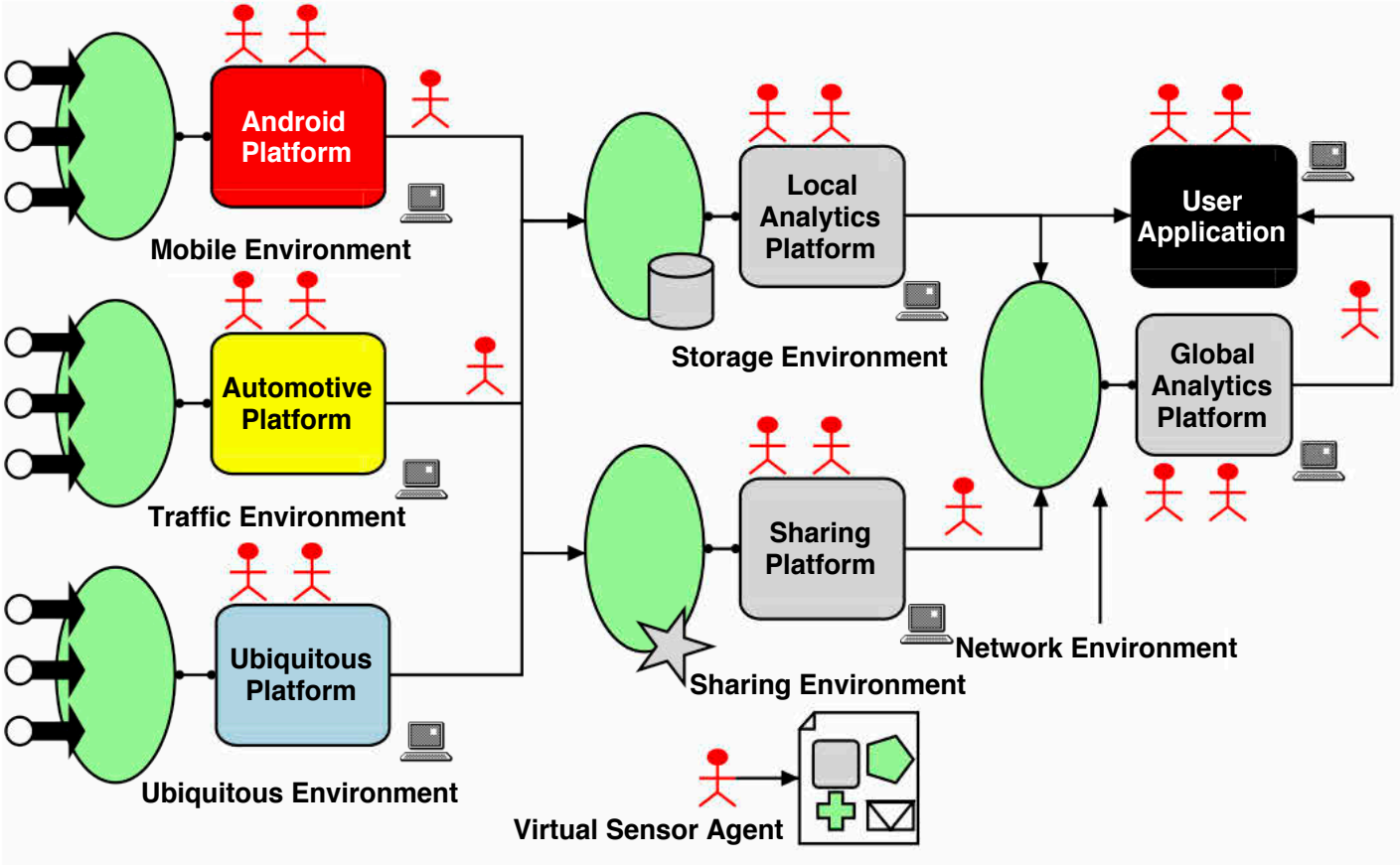
Solution:

» Agents as Virtual Sensors can cover the entire SAA architecture:

- » Agents are loosely coupled to platform and network infrastructure
  ➠ Deployment in heterogeneous environments
- » One Virtual Sensor is one Agent
  ➠ Multiple VS on one device/node
- » Mobile Agents carry their program and behaviour
  ➠ Simplifies updating and adaptation

NervousNet: The Agent Approach (cont.)

- Agents implement the sensor data aggregator and filter function
- Agents are fusion, storage, and communication/transport entities
- Agent processing is encapsulated by a unified Agent Processing Platform



D. JAM: The JavaScript Agent Machine Platform

JavaScript

- Deployment of agents in strong heterogeneous environments requires an unified and portable modelling and programming language!
- Agents require: Mobility ➠ Code Migration; Adaptivity ➠ Code Morphing
- Commonly JAVA is used:
  - ✗ Strong coupling to processing platform and API
  - ✗ No code mobility without expensive transformation; requires Compiler
  - ✗ Code cannot be modified at run-time (no code morphing capability)
- WEB programming uses JavaScript:
  - ✓ Loosely coupled to processing platform and API
  - ✓ Can be processed in text form
  - ✓ Code-Text duality (eases code mobility) provided by platform
  - ✓ Code can be modified at run-time (code morphing capability)
- JavaScript platform available on all devices (mobile, embedded, server)

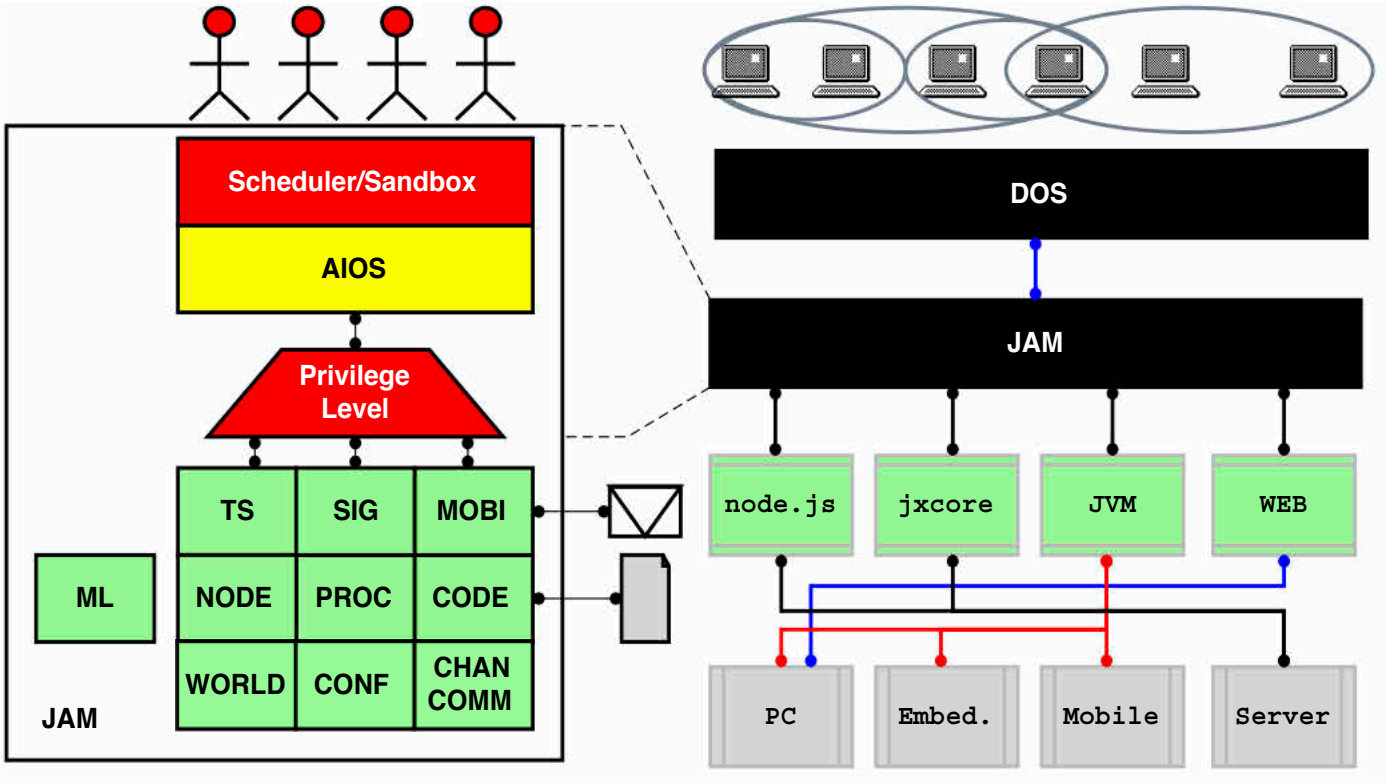
AgentJS: Agent JavaScript

- Agent behaviour modelling: **Activity-Transition Graphs** & Classes
- Agent Interaction: Tuple Spaces, Signals
- Platform access via privilige level protected AIOS: Agent I/O System
- Agent execution in a sandbox; Migration of textual process snapshots

```
function explorer(dir) {
  this.v=0;
  this.dir=dir;
  this.act = {
    init: function () { if (this.dir==DIR.ORIGIN) this.dir=DIR.NORTH },
    migrate: function () { moveto(this.dir) },
    percept: function () { in(['SENSOR',_],function (t) {this.v=t[1]}) },
    deliver: function () { out(['SENSOR',this.v]) },
    end: function () {kill(me)}
  }
  this.trans = {
    init: migrate,
    percept: function () { this.v>10?migrate:deliver },
  ..
```

JAM: JavaScript Agent Machine

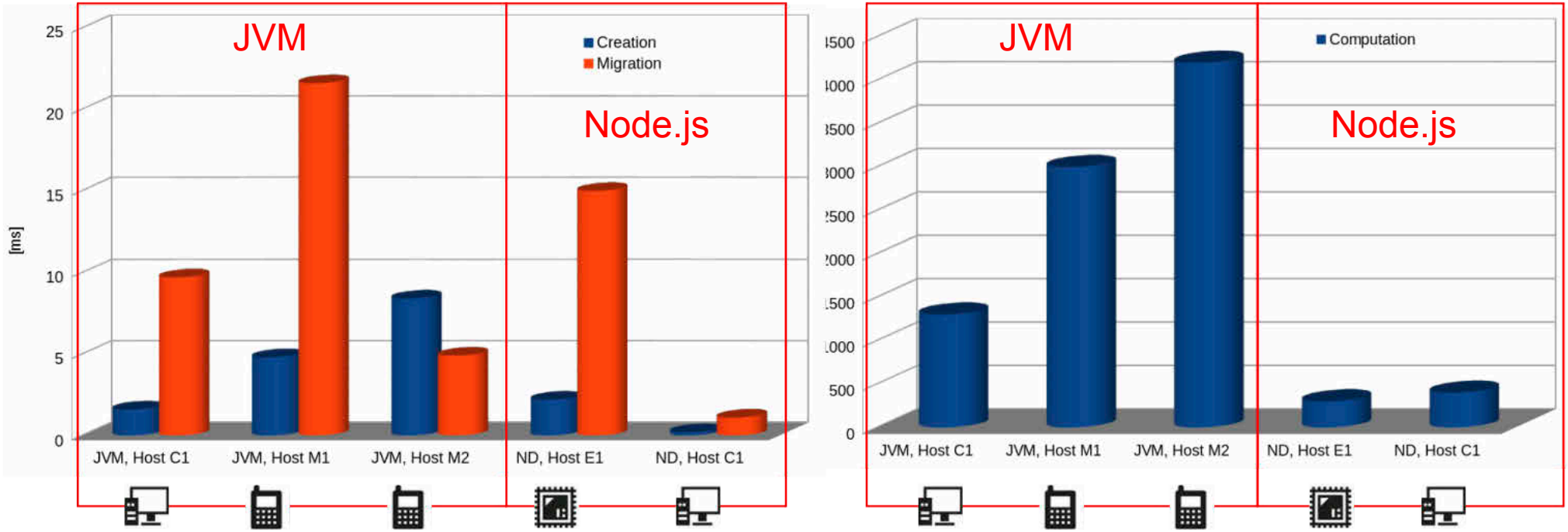
- Modular Approach
  - *TS*: Tuple Space, *SIG*: Signals, *MOBI*: Mobility, *CONF*: Code Morphing ..
  - For Internet Deployment: + Distributed Organization System (DOS)
  - JS Engines: *node.js*, *jxcore*, *JVM* (Samsung *jerryscript+iotjs*), WEB Browser



JAM: Performance

- High Perfomance
  - Low agent process creation and scheduling latency
  - Low agent migration latency
  - High volume agent processing capacity (>1000 agents/*JAM*)
  - But depends on JS engine and host
    ND node.js: ✓ Performance, *JVM*: ✓ Low-resorce

node.js:
JIT Native Code Compiler
JVM:
Bytecode Compiler



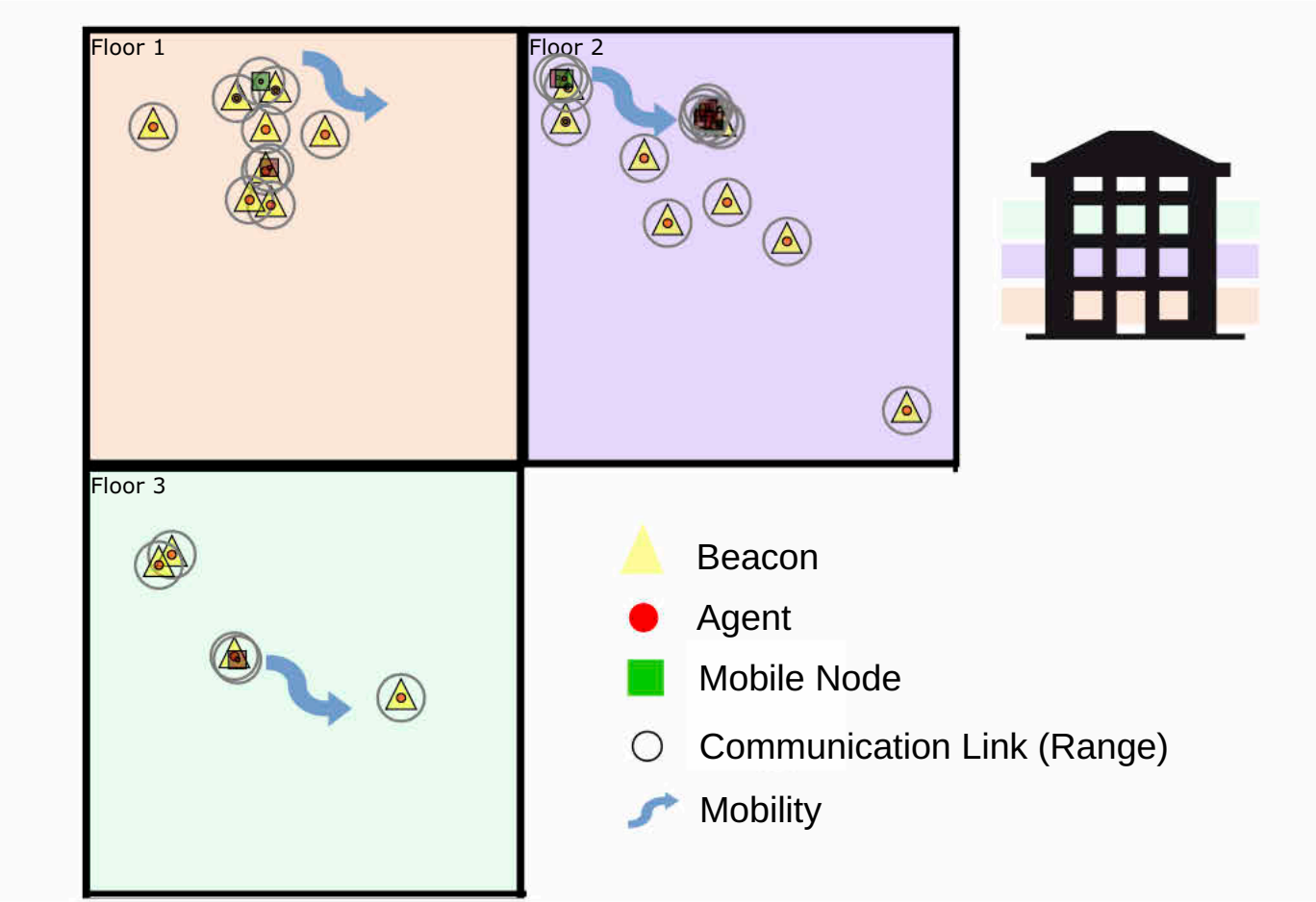
E. Case Study and Simulation: Crowd Sourcing with Agents

NervousNet + Agents + JAM = The Unified Approach

Environment & Scenario

» Building (Hamburg Congress Center), three floors

» Each floor populated with stationary beacons and mobile devices



» Beacons and mobile devices can process agents using JAM

» Mobile devices can communicate with beacons via Blue-tooth

» Beacons can have Internet access (or not)

» Agents can migrate to beacons and vice versa

Crowd Sourcing with Agents

- The goal of the MAS is the self-organized perception, collection and fusion of mobile device sensor data using the Virtual Sensor approach.
- The collected sensor data (Position, Connectivity *c*, Acceleration *a*) is used to recognize specific crowd situations in the building.
- Virtual Sensor Fusion in ROI of crowd parameters from individual data ➠ Measure of *P*: Crowd Population, *A*: Crowd Agitation, *F*: Crowd Flow

$$P(t) = \frac{1}{k} \sum_{i=1}^{Nc} 1 / \left(t - t(c_i) \right)$$

$$A(t) = \frac{1}{d} \left(A(t-1) + \frac{1}{k} \sum_{i=1}^{Ns} a_i / \left(t - t(a_i) \right) \right)$$

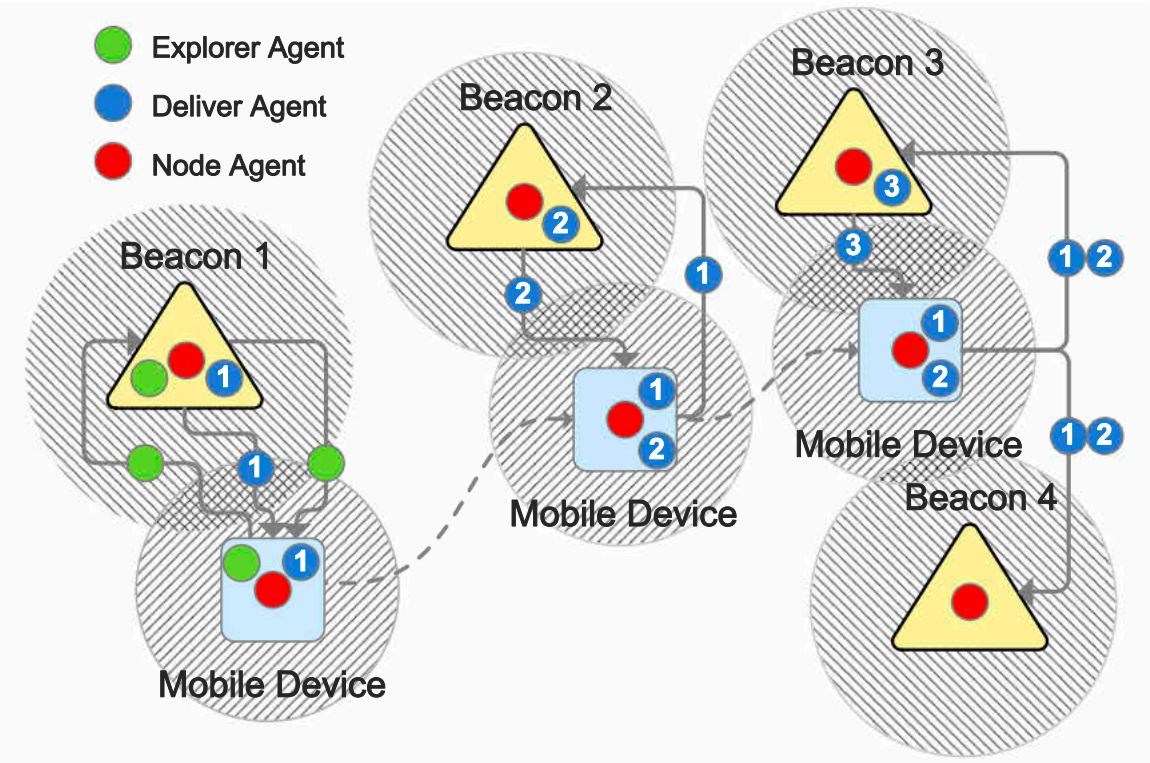
$$F(t) = \frac{1}{d} \left(F(t-1) + \sum_{i=1}^{Ns} f_i \right)$$

Mobile Devices as Carriers

» Node agents perfom event-based sensor fusion (Virtual Sensor)

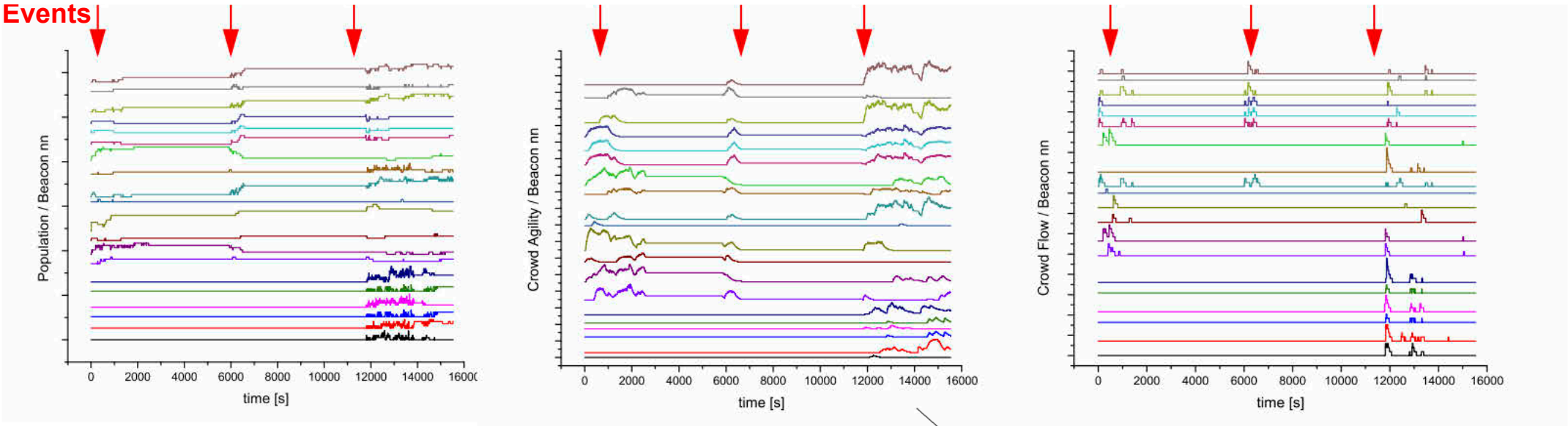
» Explorer Agents are mobile and can visit beacons

» Deliver agents are mobile and sent by node agents to distribute sensor data using mobile devices.



Simulation & Evaluation

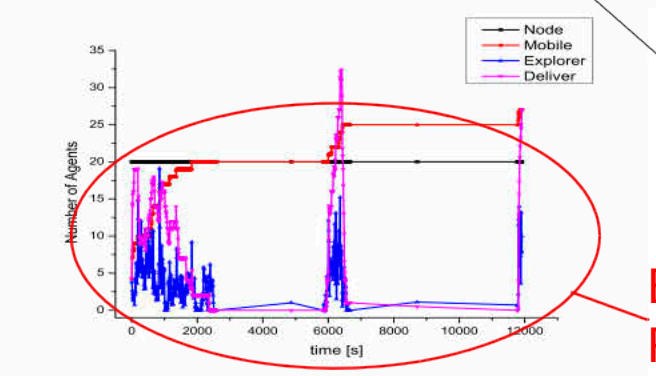
» Simulation was carried out by the *SEJAM* simulator and NervousNet data from Chaos Communication Congress event, Hamburg, 27.- 30.12.14



(a) Population *P*

(b) Agitation *A*

(c) Flow *F*



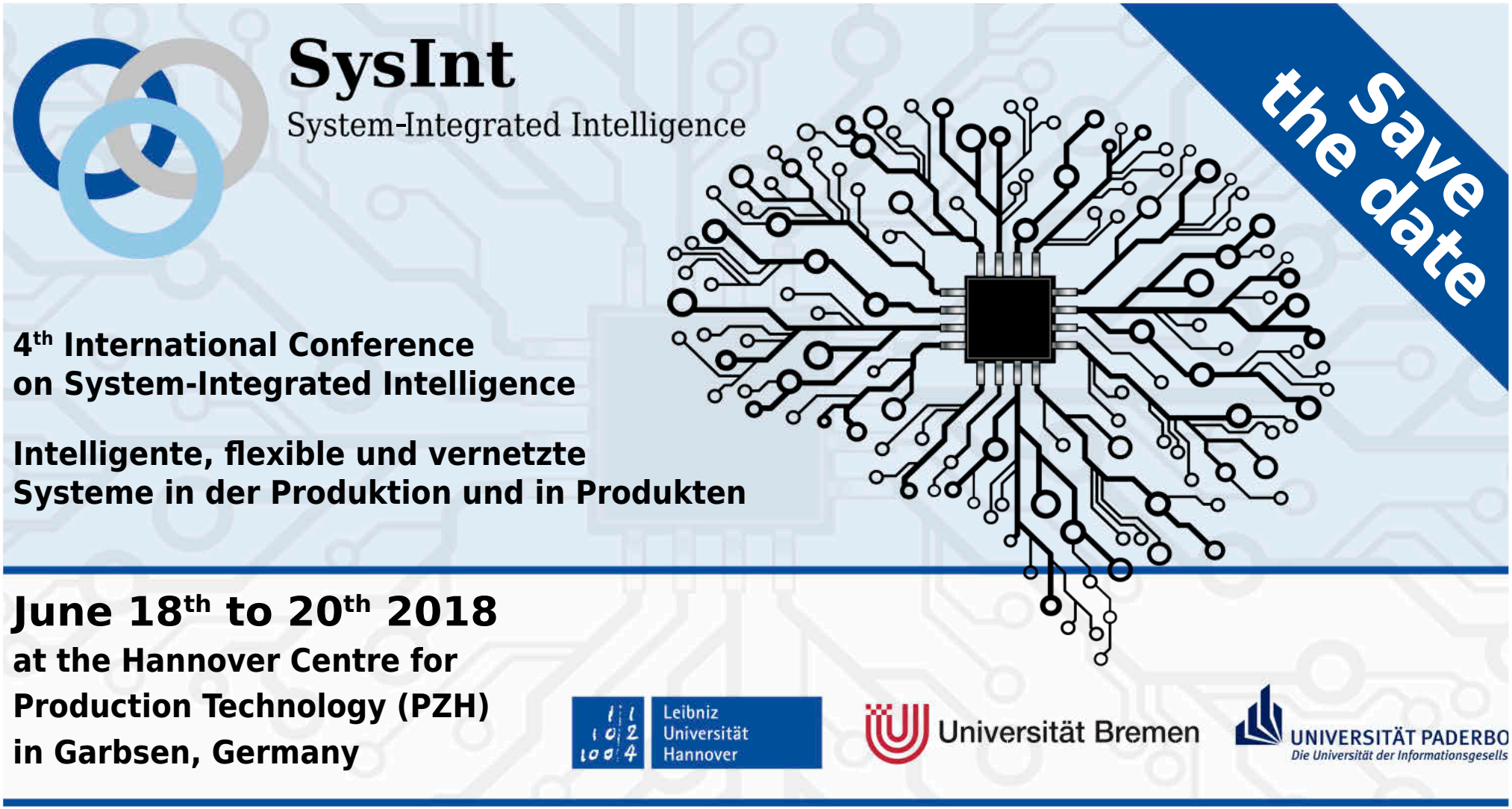
(d) Agent Population

Event-based Processing

F. Conclusions

- The use-case showed the suitability of the distributed deployment of MAS in the context of the Planetary Nervous system used for crowd sensing.
- Virtual Sensor approach was implemented with agents
- The agents can be used to pre-process and reduce the raw sensor data of smart phones in a local ROI, finally distributed across the ROI by agents carried on mobile devices.
- The agents are represented by mobile JavaScript code (*AgentJS*) that is managed and processed by a modular and portable agent platform *JAM* in a protected environment.
- *JAM* is implemented entirely in JS, and can be executed on any mobiledevice.

Announcement



An Ubiquitous Multi-Agent Mobile Platform for Distributed Crowd Sensing and Social Mining

Stefan Bosse*
University of Bremen
Department of Mathematics & Computer Science
ISIS Scientific Centre for Sensorial Materials
Bremen, Germany
* Presenting Author

Evangelos Pournaras
ETH Zürich
Chair of Sociology
Modeling and Simulation
Zürich, Switzerland

Last mod. August 23, 2017 11:16 pm / ficloud2017_pres.fm / 28 pages