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

Stefan Bosse[1], Evangelos Pournaras[2]

[1]*University of Bremen, Department of Mathematics & Computer Science, ISIS Sensorial Materials Scientific Centre, Bremen, Germany*
[2]*Chair of Sociology, in particular Modeling and Simulation ETH Zürich, Zürich, Switzerland*

Abstract **Smart mobile devices are fundamental date sources for crowd activity tracing. Large-scale mobile networks and the Internet-of-Things (IoT) expand and become part of pervasive and ubiquitous computing offering distributed and transparent services. With the IoT, Crowd Sensing is extended by Things Sensing, creating heterogeneous smart environments. A unified and common data processing and communication methodology is required so that the IoT, mobile networks, and Cloud-based environments seamlessly integrate, which can be fulfilled by self-organizing mobile agents, discussed in this work. Currently, portability, resource constraints, security, and scalability of Agent Processing Platforms (APP) are essential issues for the deployment of Multi-agent Systems (MAS) in highly heterogeneous networks. Beside the operational aspects of MAS, an organizational structure is required for the deployment of MAS in crowd sensing and social mining applications. The Planetary Nervous system (Nervousnet) consists of virtual sensors building the core functionality for such applications running on smart phones with a Cloud-like architecture. The virtual sensors enable a holistic composition and modelling approach. Self-organizing and adaptive mobile agents are well known as the core cells of holistic and modular systems. In this work, both concepts are combined. JavaScript agents are introduced as virtual sensors in the Nervousnet environment, evaluated with a simulation of a distributed sensor fusion use-case in a mobile network based on real-world data from Nervousnet, showing the suitability of the hybrid approach, benefiting from local and event-based sensor processing performed by the MAS.**

Keywords: Pervasive Computing, Ubiquitous Computing, Agents, Self-organizing systems

I. Introduction

Mobile networks, wearable devices, and the Internet-of-Things get more and more pervasive in today's digital society. Millions of devices can contribute to pervasive and ubiquitous computing, forming one big machine, which offers distributed and transparent services. These networks are highly dynamic with loosely coupled ad-hoc connectivity (temporal networks) and inter-communication, producing a large amount of data, often with a low degree of data correlation. Classical distributed systems can deal with a large amount of data, but do not meet the requirements for these dynamic and loosely coupled environments. Agents and distributed agent-based systems are already deployed successfully in heterogeneous large-scale environments, e.g., production and manufacturing processes **[1]**, or the control of manufacturing processes **[2]**, facing manufacturing, maintenance, evolvable assembly systems, quality control, and energy management aspects, and in sensing applications, e.g., monitoring of mechanical structures and devices **[3]**. Finally the paradigm of industrial agents meeting the requirements of modern industrial applications by integrating sensor networks was introduced **[4]**.

Newer trends show the suitability of Multi-agent Systems (MAS) in crowd sensing applications, e.g., **[5]**, implementing different roles of crowd sensing campaigns with agents. The agents control sensing, collect and distribute data. But the agents and the software framework (Agent processing Platform, APP) rely on Internet services, not always available, limiting the use space. A lightweighted approach and APP is required for a wide-area deployment, enhanced by a distributed adaptive organizational structure.

In large-scale dynamic networks an organizational structure is required for interaction and communication. In **[6]** sensors are considered as devices used by an upper layer of controller agents. Agents are organized according to roles related to the different aspects to integrate, mainly sensor management, communication and data processing. This organization isolates and decouples the data management from changing networks, while encouraging reuse of solutions.

Most crowd sensing platforms are using cloud- or centralized data base approaches for the aggregation and processing of user and sensor data, e.g., McSense **[7]** or Nervousnet **[8]**, based on a client-to-server architecture, though supporting distributed pre-processing.

The Planetary Nervous System (Nervousnet) is an environment and platform consisting of sensors that process a set of input streams of data generated from physical or virtual sensors. The environment defines the context within that the virtual sensor operates to generate its output stream **[8]**. The platform provides ubiquitous social mining as a public service. Sensing systems consist of three different functional layers: Sensing, Aggregation, and Application. All three layers can be represented by virtual sensors and agents. Data sharing, data collection, and data fusion are main building blocks of such a system. In contrast to traditional embedded sensor networks, social mining in public networks requires privacy rules, as discussed in **[9]**. Sensing devices, e.g., smart phones, commonly interact with a Cloud-like service architecture (device-to-cloud communication).

Crowd sensing has already been successfully applied to different purposes. In **[10]**, smart phones were used to compose a seismic network in urban environments for spatially fine-grained earthquake monitoring. But as in the current Nervousnet approach, the distributed data is evaluated either locally or at a central instance. With the emerging IoT, Crowd Sensing is extended by Things Sensing.

Distributed data mining and Map-Reduce algorithms are well suited for self-organizing MAS. Cloud-based computing with MAS, e.g., as a base for crowd sensing and participatory social mining use cases, means the virtualization of resources, i.e., storage, processing platforms, sensing data or generic information. Mobile Agents reflect a mobile service architecture. Commonly, distributed perceptive systems are composed of sensing, aggregation, and application functional layers, shown in Fig. **1**.

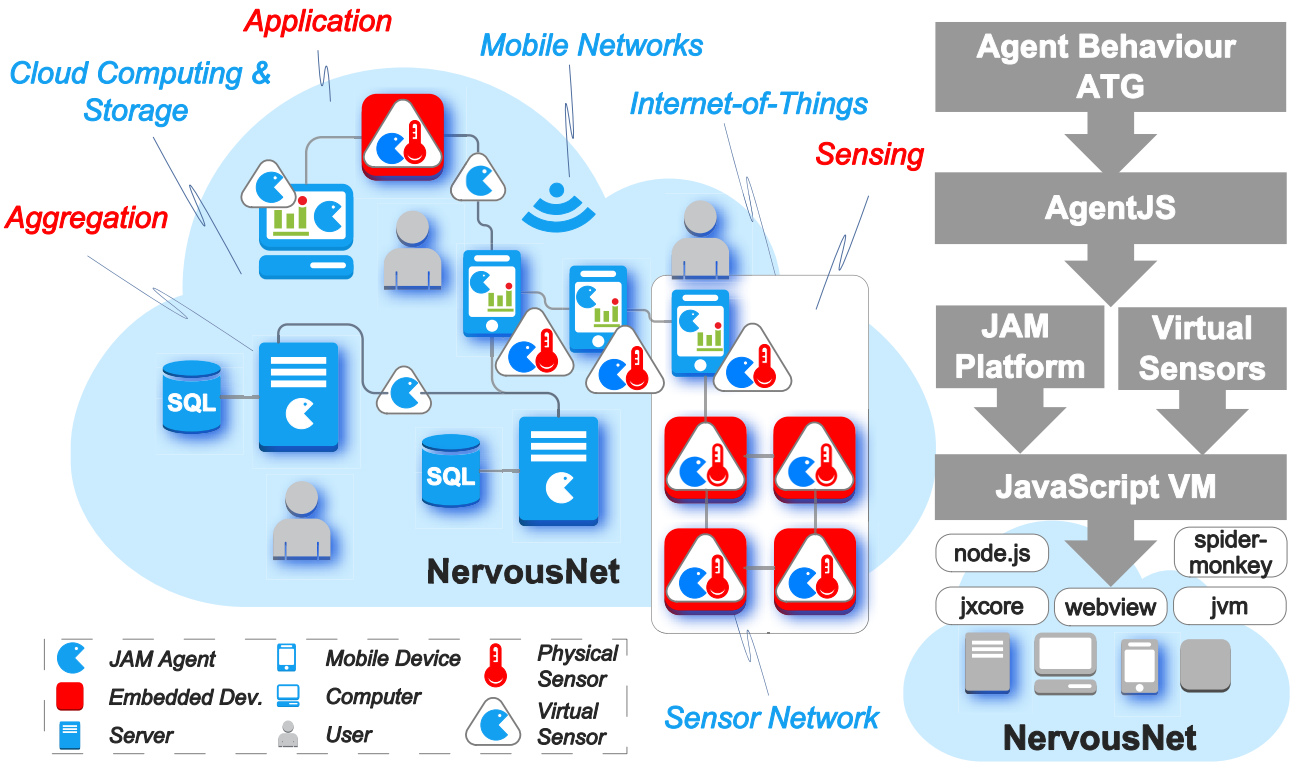

Figure 1. Unified MN/IoT/ Cloud Distributed Perception and Information Processing with mobile agents, the JavaScript (JS) Agent Machine Platform (JAM) and the Nervousnet Service as the organizational layer composed of virtual sensors, represented in this work with JAM agents.

But IoT, mobile, and Cloud environments differ significantly in terms of resources: The IoT and mobile networks consist of a large number of low-resource devices interacting with the real world and having strictly limited storage capacities and computing power, and the Cloud consists of large-scale computers with arbitrary and extensible computing power and storage capacities in a basically virtual world. A unified and common data processing and communication methodology is required to merge the IoT with Cloud environments seamlessly, fulfilled by the mobile agent-based computing paradigm, addressed in this work.

The scalability of complex ubiquitous applications using such large-scale cloud-based and wide area distributed networks deals with systems deploying thousands up to million agents. But the majority of current laboratory prototypes of MAS deal with less than 1000 agents **[2]**. In the past, many traditional processing platforms could not handle a big number of agents with the robustness and efficiency required by the industry **[2]** and Cloud applications. In the past decade the capabilities and the scalability of agent-based systems have increased substantially, especially addressing efficient processing of mobile agents. The integration of perceptive and mobile devices in the Internet raises communication and operational barriers, which must be overcome by a unified agent processing architecture and framework, discussed in this work.

In this work the behaviour of mobile agents are modeled with dynamic Activity-Transition Graphs (ATG), which are directly implemented in JavaScript (*JS*) program code holding the entire control and data state of an agent (Fig. **1**, right side). The used agent model bases on the mobile processes model introduced by Milner **[11]**. The code can be modified by the agent itself using code morphing techniques required for behaviour adaptation (directly supported by *JavaScript* Just-in-time and Bytecode Compiler VM platforms). The code is capable of migration in the network between nodes required for distributed data processing. This approach requires only a minimal Agent Processing Platform Service (APPS). The *AgentJS* code can be directly executed by the underlying *JS* VM (e.g., *node.js*, *jxcore*, *JVM*, *webview* for mobile App. development, or *spidermonkey* used in browsers).

Agents operating on a particular node can interact and synchronize by using a tuple-space, which were proposed in **[12]** and **[13]** as a suitable MAS interaction and co-ordination paradigm. This approach provides a high degree of computational independency from the underlying platform and other agents, and enhanced robustness of the entire heterogeneous environment in the presence of node, sensor, link, data processing, and communication failures.

A sensor network as part of the IoT is composed of nodes capable of sensor processing and communication. Modern mobile phones (smart phones) are equipped with multiple sensors. Hence a mobile network consisting of smart phones can be considered a sensor network, too. Smart systems and environments are composed of more complex networks (and networks of networks) differing significantly in computational power and available resources, rising inter-communication barriers. They provide higher level information processing that maps the raw sensor data to condensed information. They can

provide, e.g., Internet connectivity of perceptive systems (body area networks...). These smart systems unite sensing, aggregation, and application layers **[14]**, offering a more unified design approach and more generic and unified architectures. Smart systems glue software and hardware components to an extended operational unit, the basic cell of the IoT.

The central approach in this work focuses on mobile agents and the ability to support mobile reconfigurable code embedding the agent behaviour, the agent data, the agent configuration, and the current agent control state, finally encapsulated in portable *JavaScript* code. The mobility is granted by converting the agent program in a textual *JSON*+ representation, and finally by parsing this text and executing the code again. This agent-specific mobile program code can be executed on a variety of different host platforms including mobile devices, embedded devices, sensor nodes, and servers, using *JAM* and a *JS* VM, bridging the gap between the IoT and Cloud infrastructures.

One of the major challenges in sensing systems is the derivation of meaningful information from sensor input. Often the sensors of mobile consumer devices (such as accelerometer, humidity, light, battery, temperature, location) suffer from a poor quality. Distributed sensor fusion can be applied to improve the statistical significance of such sensor signals by collecting sensor data in a region of interest from multiple devices. Fusion can profit from Machine Learning (ML), which usually bases on classification algorithms derived from supervised machine learning or pattern recognition using, e.g., self-organizing **[14]** and distributed multi-agent systems with less or no a-priori knowledge of the environment.

This work introduces distributed crowd sensing with sensor and information fusion using mobile agents, enabling the design of large-scale sensing systems using virtual sensors and the Nervousnet providing the organizational structure.

The next sections introduce the conceptual fusion of virtual sensors and agents, the *JAM* platform, and *JAM* agents. The agent platform and approach is evaluated with a case study demonstrating self-organizing sensor fusion, and further applications are discussed. The case-study relies on mobile phone data sampled the 2014 Chaos Communication Congress in Hamburg, used to investigate temporal network events in **[9]**.

II. Virtual Sensors and Agents

In **[8]**, a large-scale sensing application is composed of virtual sensors. A virtual sensor is a software component being the core cell of the Nervousnet platform. Each software component is treated as a sensor, processing an input stream and computing an output stream. Each physical sensor is a "data stream" transformer, too, but based on physical principles. A virtual sensor is a processing system as well as a data storage (data base).

In this work, virtual sensors are represented by mobile agents, performing the sensing, aggregation, and application (or delivery) of accumulated and processed sensor data. As discussed in the next section, these agents are highly portable and can be executed by a wide range of devices including smart phones. The mobility enables self-organizing and adaptive mining systems controlled by environmental constraints rather than by individual users. In **[9]**, users using a smart phone App. are considered as agents. This role is replaced in this work by the deployment of agents that perform tasks autonomously.

The agents interact with each other by accessing tuple spaces or by exchanging signals. The advantage of tuple spaces and mobile agents is the generative nature. A sensor data tuple can be stored in an environment without physical sensors by mobile agents, enabling the access of remote sensor data by other agents.

In the original Nervousnet platform, mobile Apps. deliver sensor data to the Nervousnet data bases, and access control is performed by the Nervousnet platform. The autonomy of agents and the anonymous nature of tuples introduce privacy issues, which require dedicated privacy control mechanisms. Although data encryption can be used to protect sensor data, a privacy protection layer applied to sensor data without encryption stripping private device and user data can be considered as a more powerful and useful technique. One private information still exists: The location of agents and the sensor data they collect from devices, which can be easily recognized by mobile agents applying path tracing and other relative localization methods. Therefore agents require encrypted keys to access personal and sensitive sensor data on mobile devices, granted by the user or trusted platform.

III. The JAM Platform and Security

The deployment of mobile agents in strong heterogeneous environments require a scalable agent processing platform, which fit low-resource host platforms (smart phones, sensor nodes, beacons) as well as high-performance host platforms such as Cloud data centers. Furthermore, the migration of agents between different host platforms must be seamless (portability) and should require as low as possible resources. *JavaScript* (JS) programs are highly portable and are executed on a wide range of platforms without recompilation or modification.

JAM is the *JavaScript* Agent Machine, implemented entirely in JS. *JAM* executes reactive agents based on the Activity-Transition-Graph behaviour model (ATG), programmed in *AgentJS*. *JAM* can be executed on any JS Virtual Machine (VM), e.g., node.js, *spidermonkey* with WEB browsers, and the new low-resource machine *JVM* (based on *jerryscript* and *Iot.js*). *JVM* can be used on any host platform, including Android and iOS mobile devices or micro controller-based beacons. The architecture and agent programming is discussed in detail in **[15]** and **[16]**. To summarize, *JAM* agents are programmed in JavaScript (*AgentJS*) and are executed in a sand-boxed environment isolating agents from each other and the computer system by the *JAM* Agent Input/Output System (AIOS) with a specific set of operations. The agent behaviour is composed of activities with conditional or unconditional activity transitions based on agent data (forming the ATG). Upon migration of agents between APPs, an agent process snapshot is created (JSON+ text format) that carries the agent state (code, private data, and control state) and the agent behaviour (ATG). The snapshot can be directly transferred via any kind of communication channel between nodes and networks, e.g., Bluetooth links, Internet connections.

JAM is capable of handling thousands of agents per node. *JAM* supports virtualization and resource management. Depending on the used JS VM, agent processes can be executed with nearly native code speed. *JAM* can be embedded in any

host application by using the library version *JAMLIB* (400kB code size).

For security reasons and to limit Denial-of-Service attacks, agent masquerading, spying, or other misuse, agents have different access levels (roles). There are four levels:

1. Guest (not trusting, semi-mobile)
2. Normal (maybe trusting, mobile)
3. Privileged (trusting, mobile)
4. System (trusting, locally, non-mobile)

The lowest level (1) does not allow agent replication, migration, or the creation of new agents. The *JAM* platform decides the security level. The highest level (4) has an extended AIOS operation set with host platform device access capabilities. Agents can negotiate resources (e.g., CPU time) and a level raise secured with a capability-key that defines the allowed upgrades. The system level can not be negotiated. The capability is node specific. A group of nodes can share a common key (identified by a server port). A capability consists of a server port, a rights field, and an encrypted protection field generated with a random port known by the sever (node) only and the rights field.

Among the AIOS level, other constrain parameters can be negotiated:

- Scheduling time (maximal slice time for one activity execution, default is 20ms)
- Run time (accumulated agent execution time, default is 2s)
- Living time (overall time an agent can exist on a node before it is killed, default is 200s)
- Tuple space access limits
- Memory limits (practically fuzzy, usually the entire size of the agent code including private data, actually not limited)

JS VM/Benchmark	Host C1	Host M1	Host M2	Host E1
JVM, A1, N4, n100 Creation/Migration	1.6/9.7ms 3/4MB	4.8/21.6ms 2/3MB	8.4/49ms 2/3MB	-
ND1, A1, N4, n100 Creation/Migration	0.2/1.1ms 27/36MB	-	-	-
ND2, A1, N4, n100 Creation/Migration	0.13/1ms 21/32MB	-	-	2.2/15ms 15/27MB
JVM, F1, D2000, n10 Computation,	1300ms 6MB	3000ms 5MB	4200ms 5MB	
ND1, F1, D2000, n10 Computation	400ms 45MB	-	-	-
ND2, F1, D2000, n10 Computation	550ms 35MB	-	-	300ms 40MB

Table I. All times per agent and action, all memory values are total JAM resident memory after the benchmark operation, A1: Simple Explorer Agent, F1: FFT Computation Agent, C1: Hewlett-Packard HP xw9400 Workstation, AMD Opteron 2216 2.4GHz x64, M1: Lenovo, ideapad, M1: Smartphone, Toughshield R500+, E1: Raspberry Pi Zero, Broadcom 1GHz ARM11, ND1:node.js v5.11, ND2: node.js v0.10, n: Number of agents, N: Number of logical nodes, D: Size of data vector

The processing performance of *JAM* depends on the host platform (computer, server, smart phone, embedded system) and the used JS engine (node.js/V8, *jxcore*/V8/*Spidermonkey*, *JVM*). *JVM* is a bytecode interpreter compiling JS text code to bytecode at run-time, and V8-based machines are hybrid interpreters with just-in-time (JIT) native code generation. Bytecode engines compared with native code engines have the advantage of a high degree of portability, but the disadvantage of slower execution speed (~100 times). Native code engines have a much higher memory consumption, shown in the experimental evaluation and comparison in Tab. **I**.

In all benchmark experiments in Tab. **I** the *JAMLIB* code library was used. The creation and migration of agents require text-to-code and code-to-text transformations that are computational expensive (see A1 benchmark). Though the computation speed of *JVM* is about 100 times slower compared with V8 engines, the agent activity computation performance is only 3-4 times slower (see FFT benchmark F1). This is a result of a watchdog timer built in the *JVM* byte code interpreter required by *JAM* for agent scheduling and time slicing. The watchdog raises an exception if an agent activity exceeds the (negotiated) time slice. In V8 engines this watchdog approach cannot be implemented (due to the native code compilation), and must be emulated by check pointing, which injects checkpoint function calls in the agent code (in all functions and loops), slowing down the agent activity execution significantly.

IV. The Multi-Agent System for Flow Monitoring

The goal of the MAS is the self-organized collection and fusion of mobile device sensor data. Though mobile devices can connect to the Internet, it is assumed that mobile devices interact with beacons deployed in the environment. These beacons provide a low range locally limited connectivity, e.g., based on Bluetooth technology. Beacons can be optionally connected to the Internet to interact with the Nervousnet data base or other beacons, but this is not a prerequisite. The beacons are part of Nervousnet and represent virtual sensors. Hence, a beacon can be any thing or technical device that is part of the IoT. Moreover, mobile agents should be used to exchange information between beacons and the data base, creating virtual paths. One major task in crowd sensing and social mining is the estimation and prediction of people flows in public environments, e.g., for building automation (controlling temperature, light, air condition) and emergency management. Furthermore, such flow monitoring can be used for the evaluation of advertisement placement in public environments or traffic control.

Acceleration sensor data from mobile devices can be used to classify different crowd situations, i.e., low, middle, and high individual movement within a ROI, defining some kind of crowd agitation (comparable to an entropy). But acceleration data is very unreliable and spatially relative. The three-dimensional orientation of smart phones is usually unknown if compass/tilt sensor data is missing. But sensor fusion of ensemble data from multiple devices in the same region can be used to derive information of crowd movement patterns within a ROI.

The MAS is composed of different agents having different goals and performing different behaviours:

Beacon Node Agent

The beacon node agent is non-mobile and responsible for the connection management, collection, and processing of sensor data delivered by mobile explorer agents sent from mobile devices. The agent performs sensor fusion and event detection using an exponential fusion filter, discussed in Sec. V..

Mobile Device Node Agent

The mobile device node agent is non-mobile and responsible for the sensor data acquisition and event detection on the mobile device. If sensor events are recognized it sends out explorer agents distributing the sensor data to beacons in the neighbourhood.

Explorer Agent

The mobile explorer agent is used to distribute mobile device sensor data to beacons and to get information from beacons in the neighbourhood of the mobile device.

Deliver Agent

Accumulated sensor data collected by beacons (including beacon identification and GPS position) can be distributed to other beacons by using mobile deliver agents and mobile devices as a carrier host. The deliver agent only exchanges data on beacons by using the tuple space data base on each node. To avoid flooding of data bases, markings with a limited time life are used instead of persistent tuples. The mobile device is a transportation unit only. If new beacons are detected by the deliver agent, it replicates itself and the replicates migrate to the new beacons to deliver the sensor data. After delivery, the replicated agents terminate (see Fig. **2**).
To avoid deliver agent flooding, a beacon only sent one deliver agent to a mobile device and the deliver agent has a limited lifetime.

Notification Agent

The notification agent is sent to mobile devices or other beacons to notify about important events. Notification agents are broadcast messenger, with a similar behaviour of the deliver agent, including replication.

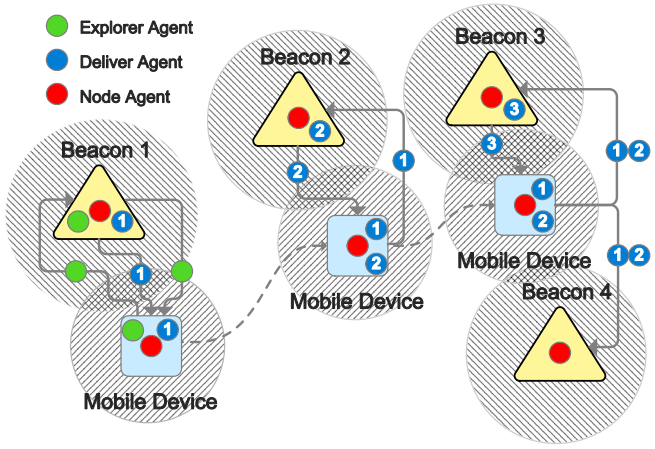

Figure 2. Principle network topology with spatially distributed beacons (non-mobile) and mobile devices, the MAS and the agent-node interactions.

The principle network topology and the deployment of agents are shown in Fig. **2**. It is assumed that mobile devices can communicate with beacons only (and vice versa).

The position of a mobile device given in geographic coordinates by the vector O=(*latitude*,*longitude*) can be either measured by a GPS receiver or can be estimated by using a weighted position triangulation with a set of detected beacons delivering their GPS-based position O_i using short range communication (e.g., Bluetooth):

$$O_{est} = \frac{1}{\sum N(rssi_i)} \sum_i N(rssi_i) O_i \qquad (1)$$

with *rssi* as the received signal strength indicator (-dB units) used as the weight and a measure for the distance between the mobile device and the beacon. and N a normalization function mapping *rssi* levels on linear distance units, e.g., $N(r)=(r0+r)$. Usually users of smart phones do not allow GPS position tracking, and beacon triangulation can be a valuable solution to get the position of mobile devices.

Agents exchange information/sensor data via the node tuple space without knowing each other, avoiding agent identification. Data from one node A (read from the node A tuple space) can be transported to another node B (written to the node B tuple space) by agents carried on mobile devices.

The data collected by explorer and deliver agents can be used for distributed learning delivering a prediction of crowd situations. Distributed agents-based on-line learning was discussed in **[16]**, and is not addressed in this work, but can easily be added with mobile learner agents. These learner agents can hop between different mobile devices staying within a Region-of-Interest (ROI).

V. Use-case: Distributed Flow Monitoring in a Building

The use-case should demonstrate the MAS deployment in mobile networks performing distributed and self-organizing sensor fusion and flow monitoring. Sensors of smart phones usually delivering sensor data with low accuracy. If there are many devices within ROI, multiple sensors can be used to compute a more accurate ensemble value. Examples are ambient light, temperature, radio signal strength, humidity, acceleration, or sound level. In Crowd monitoring, the position and the acceleration sensors can be used as an input.

The MAS consists of different non-mobile and mobile agents, discussed in Sec. IV. The goal of the MAS is crowd monitoring, delivering a time-resolved spatial matrix of three parameters: 1. Population (number of mobile devices in a ROI), 2. Agitation (weighted and accumulated movement profile of the usersin a small region), and 3. Flow (paths from other ROI and beacons).

The simulation uses collected smart phone data from the conference event (Chaos Communication Congress, Hamburg, 27.12. - 30.12.14, **[9]**). The simulation world consists of a map of the three floors of the conference building, deployed with multiple non-mobile beacons (triangles), shown in Fig. **3**. The beacons are positioned by their measured GPS position. The mobile devices are positioned either by a submitted GPS positions (not available in this experiment) or/and by a weighted position triangulation of detected beacons (radio signal strength is used as the weight, Eq. **1.**).

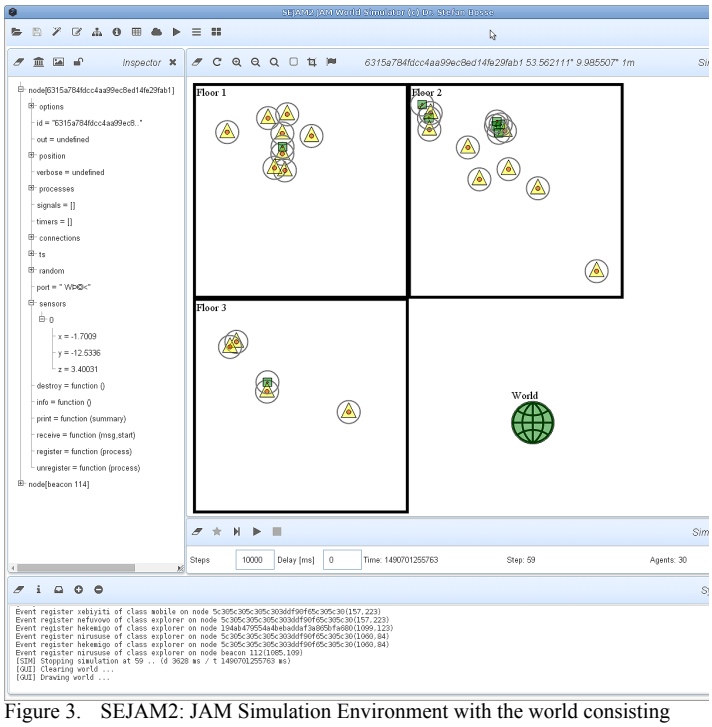

Figure 3. SEJAM2: JAM Simulation Environment with the world consisting of three floors of a building. Shown are beacons (yellow triangles, each populated with a node agent) and some mobile devices (green rectangles). The black circles represent Bluetooth links of the devices, indicating the communication range.

The GPS position of smart phones usually have a poor spatial accuracy, and beacon triangulation can deliver more accurate results.

The *SEJAM* simulator was connected to multiple SQL3 data base servers, storing the sensor and monitoring data.

Experimental simulation results are shown in Fig. **4**. The agent population of the entire conference network depends on smart phone mobility (moving velocity) and the crowd flow in the beacon areas. The event-based approach for the creation of explorer and deliver agents results in an overall low agent density with periods of low activity, reducing communication and computation significantly.

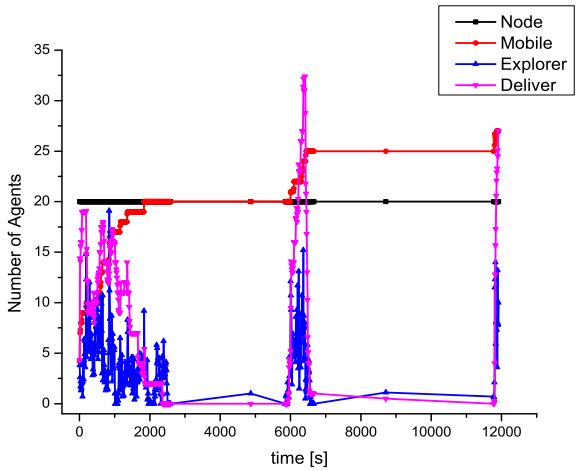

Figure 4. Agent population in the conference network during the first 5000 simulation steps (start time 1419700000s, end time 1419711914s UTC). Total created agents: Mobile: 29, Explorer: 10590, Deliver: 880).

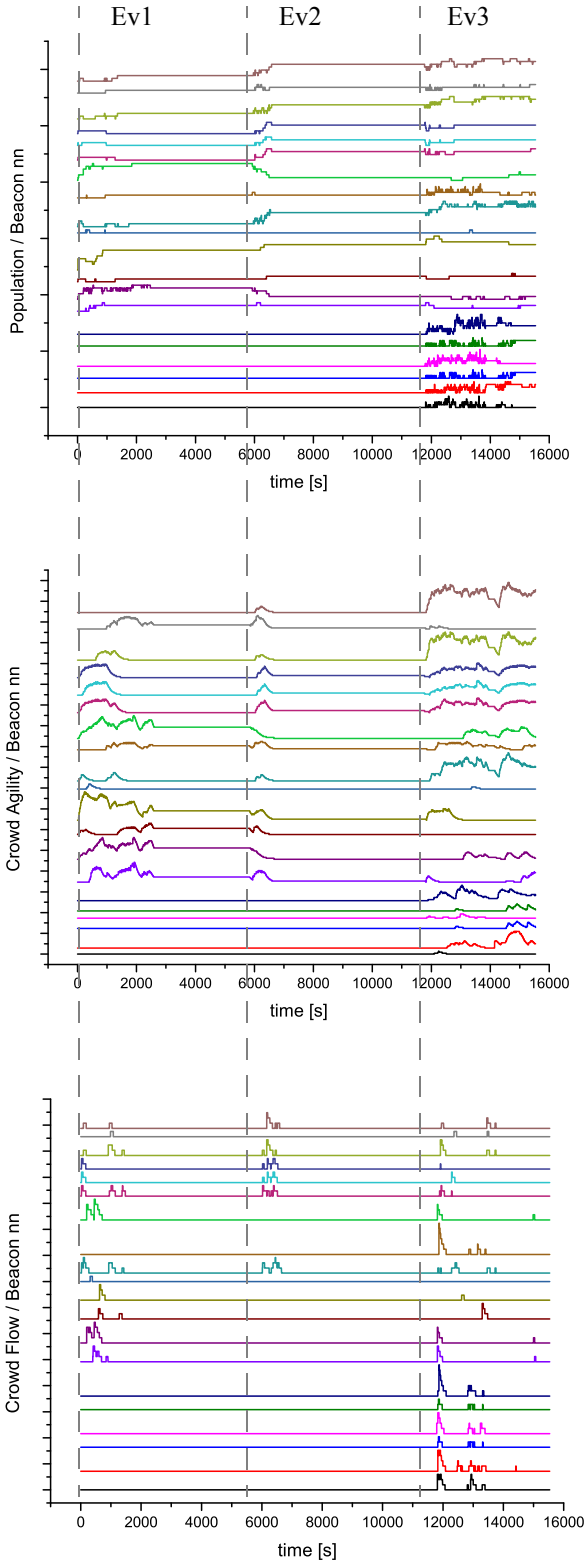

Figure 5. (Top) Crowd Population, (Middle) Crowd Agitation, and (Bottom) Crowd Flow (diffusion) monitoring from the beacon agents in the conference network [Y axis: plot stacking of sensors of individual beacon agents, bottom line: beacon 11, top line: beacon 120]

The population of explorer and deliver agents correlates significantly with three relevant different events happening during the observation time, analyzed in the following discussion.

The MAS performs the estimation of different crowd situations parameters, enabling the distinction of different crowd behaviour situations. These computed parameters can be considered as virtual sensors of the Nervousnet system. The three relevant virtual sensors <*P*,*A*,*F*> implementing sensor fusion are computed by Eq.**2.**, the population measure *P*, based on the actual and past connectivity detection of mobile devices (connection *c*), the acceleration sensor data delivered by explorer agents (*a*, the mean value of x-/y-/z- direction values) computing the crowd agitation measure *A* in a ROI (i.e., a measure for the small range movement of the crowd in the ROI, e.g., a room), and the flow counter delivered by the deliver agents (*f*) used for the computation of the flow *F*. Exponential filtering (with *d* as a decay constant) is applied to the agitation and flow sensors, and the population and agitation sensors use time weighted sensor values (*t* is the current time, *t*(*s*) the sensor acquisition time, and *k* a weight factor). *Nc* is the number of connected nodes, and *Ns* is the number of all aggregated sensors.

$$P(t) = \frac{1}{k}\sum_{i=1}^{Nc} 1/\left(t - t(c_i)\right)$$
$$A(t) = \frac{1}{d}\left(A(t-1) + \frac{1}{k}\sum_{i=1}^{Ns} a_i/\left(t - t(a_i)\right)\right) \quad (2)$$
$$F(t) = \frac{1}{d}\left(F(t-1) + \sum_{i=1}^{Ns} f_i\right)$$

The time-resolved crowd population, crowd agitation, and crowd flow monitoring results of all beacons in the network are shown in Fig. **5**. Specific situations, i.e. crowd movement or crowd forming can be localized and detected based on the locally computed and distributed virtual sensor data delivered by the beacon agents. In the simulation there are three characteristic situations within the time window shown in Fig. **5** (Events 1,2,3). Event 1 shows a crowd formation with moderate population but increased agitation, and sporadic flow diffusion. Event 2 shows a crowd formation with an increased population but low agitation and diffusion. And finally, event 3 shows high population and agitation with increased diffusion behaviour of the crowd.

VI. Conclusions and Outlook

The use-case showed the suitability of the distributed deployment of MAS in the context of the Planetary Nervous system used for crowd sensing. In contrast to many existing crowd sensing solutions, the MAS approach with local sensor aggregation and processing will provide an enhanced scaling in large-scale scenarios. The original Nervousnet is closely related to the Cloud paradigm with server-client communication. The MAS 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. Large-scale distributed applications require an organisational structure layer, which addresses scalability, adaptability, self-organization, robustness, and resource constraints, basically composed of agents implementing virtual sensors together with the Nervousnet organization. Three relevant virtual sensor parameters were computed by beacon agents: Crowd population, agitation, and flow, getting sensor input from explorer and deliver agents, carried by 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 sandbox environment encapsulating agent processes. *JAM* is implemented entirely in *JS*, and can be executed on any mobile device. The virtual sensors are basic cells composing an organisational structure, the Planetary Nervous system, providing the high-level platform for social mining. The presented approach enables the development of perceptive clouds and smart environments of the future integrated in daily use computing environments and the Internet. Specific situations, i.e. crowd movement or crowd forming can be localized and detected based on monitoring data of a simple beacon network. The design and platform approach is suitable to cover the sensing, aggregation, and application layers of large-scale and massively distributed information processing systems efficiently, consisting of heterogeneous devices. The use-case that bases on real sampled sensor data of smart phones collected by Nervousnet showed the suitability of the MAS for efficient event-based Crowd sensing.The MAS composed of agents with different goals and behaviour is able to recognize different crowd situations with different population, agitation, and diffusion (flow) characteristics just by using the three computed virtual sensor values. The low-resource JS machine *JVM* enables the deployment of *JAM* on IoT devices and smart phones.

References

[1] M. Caridi and A. Sianesi, *Multi-agent systems in production planning and control: An application to the scheduling of mixed-model assembly lines*, Int. J. Production Economics, vol. 68, pp. 29–42, 2000.

[2] P. Leitão and S. Karnouskos, Industrial Agents Emerging Applications of Software Agents in Industry. Elsevier, 2015.

[3] S. Bosse, A. Lechleiter, *Structural Health and Load Monitoring with Material-embedded Sensor Networks and Self-organizing Multi-agent Systems*, Procedia Technology, 2014, DOI: 10.1016/j.protcy.2014.09.039

[4] D. Lehmhus, T. Wuest, S. Wellsandt, S. Bosse, T. Kaihara, K.-D. Thoben, and M. Busse, *Cloud-Based Automated Design and Additive Manufacturing: A Usage Data-Enabled Paradigm Shift*, Sensors MDPI, vol. 15, no. 12, pp. 32079–32122, 2015, DOI 10.3390/s151229905.

[5] T. Leppänen, J. Á. Lacasia, Y. Tobe, K. Sezaki, and J. Riekki, "Mobile crowdsensing with mobile agents," Autonomous Agents and Multi-Agent Systems, vol. 31, no. 1, 2017.

[6] M. Guijarro, R. Fuentes-fernández, G. Pajares, *A Multi-Agent System Architecture for Sensor Networks*, Multi-Agent Systems - Modeling, Control, Prog., Simulations and Applications, 2008.

[7] G. Cardone et al., "Fostering ParticipAction in Smart Cities: A Geo-Social Crowdsensing Platform," IEEE Communications Magazine, no. 6, 2013.

[8] E. Pournaras, I. Moise, and D. Helbing, *Privacy-preserving Ubiquitous Social Mining via Modular and Compositional Virtual Sensors*, in IEEE 29th International Conference on Advanced Information Networking and Applications, 2015.

[9] F. Musciotto, S. Delpriori, P. Castagno, and E. Pournaras, *Mining Social Interactions in Privacy-preserving Temporal Networks*, in Advances in Social Networks Analysis and Mining (ASONAM), 2016 IEEE/ACM, 2016.

[10] Q. Kong, R. M. Allen, L. Schreier, and Y.-W. Kwon, "MyShake: A smartphone seismic network for earthquake early warning and beyond," Sci. Adv., vol. 2, 2016.

[11] R. Milner, *The space and motion of communicating agents*. Cambridge University Press, 2009.

[12] L. Chunlina, L. Zhengdinga, L. Layuanb, and Z. Shuzhia, *A mobile agent platform based on tuple space coordination, Advances in Engineering Software*, vol. 33, no. 4, pp. 215–225, 2002

[13] Z. Qin, J. Xing, and J. Zhang, *A Replication-Based Distribution Approach for Tuple Space-Based Collaboration of Heterogeneous Agents*, Research Journal of Information Technology, vol. 2, no. 4. pp. 201–214, 2010

[14] V. Di Lecce, M. Calabrese, and C. Martines, *From Sensors to Applications: A Proposal to Fill the Gap*, Sensors & Transducers, vol. 18, no. Special Isse, pp. 5–13, 2013.

[15] S. Bosse, *Mobile Multi-Agent Systems for the Internet-of-Things and Clouds using the JavaScript Agent Machine Platform and Machine Learning as a Service*, in The IEEE 4th International Conference on Future Internet of Things and Cloud , 22-24 August 2016, Vienna, Austria, 2016.

[16] S. Bosse, Distributed Machine Learning with Self-organizing Mobile Agents for Earthquake Monitoring, in 2016 IEEE 1st International Workshops on Foundations and Applications of Self* Systems (FAS*W) , SASO Conference, DSS, 12 September 2016, Augsburg, Germany, 2016.