DOI 10.1109/FAS-W.2017.123
IEEE 2nd Int. Workshops on Found. and Appl. of Self* Systems (FAS*W)

Towards Large-scale Material-integrated Computing: Self-Adaptive Materials and Agents

Stefan Bosse
University of Bremen
Department of Mathematics & Computer Science
Bremen, Germany
sbosse@uni-bremen.de

Dirk Lehmhus
University of Bremen
ISIS Sensorial Materials Scientific Centre
Bremen, Germany
lehmhus@uni-bremen.de

Abstract -**In the past decades there was an exponential growth of computer networks and computing devices, connecting computers with a size in the m^3 range. The Internet-of-Things (IoT) emerges connecting everything, demanding for new distributed computing and communication approaches. Currently, the IoT connects devices with a size in the cm^3 range. But new technologies enable the integration of computing in materials and technical structures with sensor and actor networks connecting devices in the mm^3 range. This work investigates issues in large-scale computer networks related to the deployment of low- and very-low resource miniaturized nodes integrated within materials. These networks operate under harsh conditions with possibility of technical failures requiring robustness. Despite sensor networks used for structural monitoring, self-adaptive materials can profit from self-organizing and autonomous distributed data processing using Multi-agent systems, demonstrated in this paper. Self-adaptive materials are able to adapt the material or mechanical structure properties based on their environmental interaction (load/stress) to minimize the risk of overloading. A structure that could change its local properties in service based on the identified loading situation could thus potentially raise additional weight saving potentials and thus supporting lightweight design, and in consequence, sustainability.**

Keywords - Big Data, Pervasive Computing, Material Computing, Adaptive Materials, Agents, Self-organizing systems

I. Introduction

In the last decades there was a shift from passive single sensors towards networks of smarts sensors equipped with Information-Communication Technologies (ICT). Furthermore, there was a significant increase of the sensor density in sensor networks **[22]**. Optimization problems are commonly mathematically solved **[1][2]** or being global problems **[3][4]** that can only be solved by processing the entire sensor data set on every system iteration. But the need for a transition to distributed approaches with local processing is eminent to satisfy future scaling of large systems consisting of Thousands and millions of small perceptive and computational nodes, each equipped with low computational power and storage capacity.

In large-scale dynamic networks an organizational structure is required for interaction and communication. In **[5]** 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. Self-adaptive materials require a control mechanism behind the material property adaptation, preferably a two-stage approach combining Multi-agent systems (MAS) and Machine Learning (ML). MAS are used to analyze the loading situation based on sensor information - preferably, highly localized strain data - and negotiate a matching redistribution of, e.g., local stiffness values according to some higher-level aim like minimizing total elastic strain energy or maximum stress levels in the system. The machine learning approach in contrast would recognise loading situations that have already been encountered in the past and on this basis avoid the MAS approach by directly proposing the solution found in the preceding case. Agents should perform sensor distribution, negotiation of material property changes, and monitoring of the structure. All these tasks involve agent communication. In this work, the property subject to negotiation is the stiffness of the material or dedicated mechanical elements like springs. The goal is to reduce strain and stress in the material under varying loading situations. Considering low-resource platforms the efficiency and optimization of communication is a major challenge, which is addressed in this work. In **[6]**, the massive increase of data in such material-integrated networks are predicted towards Big Data.

This work introduces the following novelties;

- Self-adaptive materials using self-organizing MAS controlling an artificial mass-body-spring system with a self-regulating approach;
- New concepts for large-scale material-integrated computing based on low-resource computing nodes and MAS;
- A new multi-domain simulation framework *SEJAM* 2.0 coupling MAS and physical simulation based on *JAM* 2.0 and the *CANNON* multi-body physics engine;
- A first attempt to create a new agent processing platform framework with enhanced agent communication support, and combining two different platform technologies, *JAM* 2.0 and *AFVM* 2.0, addressing the IoT and material-integrated computing, respectively.

The next section introduces self-adaptive materials and structures from the material science point of view, in conclusion defining hard constraints on the data processing integrated in materials. The following sections discuss the design of appropriate data processing architectures and algorithms for self-adaptivity, finally evaluated with a multi-domain simulation of a plate using the new *SEJAM* 2.0 multi-domain simulator.

II. Self-adaptive Materials And Structures

The concept of adaptive materials draws much of its appeal from the fact that engineering design has to decide which are

the load cases or service conditions to base dimensioning on, while reality typically knows no such limitations: Conventional techniques like shape, topology or multi-material optimization all need to single out few sets of boundary conditions to adapt to.

Any real world load-bearing structure, in contrast, may see different scenarios including misuse, which might be impossible to capture/define in the design stage. Such unforeseen loads may in the worst case lead to immediate failure - in other cases, they might just wear out a structure, causing it to fail prematurely and/or in places determined by unexpected local load histories.

An adaptive material is capable of adjusting its mechanical characteristics - like stiffness - to external loads could actively manage this local load history. It could protect areas already worn out and distribute loads to others instead. Under the fundamental assumption that design loads can only reflect an - above all estimated - average of stresses to be sustained, the redistribution problem is not deterministic, but requires the adaptive structure to adjust its decisions over time, basing its reaction on the actual load pattern observed, and on the accumulation of previous events.

As a direct consequence, it is necessary that the respective structure must be able to analyze previous and ideally predict future loads. Adaptivity in the suggested sense thus has to be represented in a material thus equipped in terms of several individual aspects:

- The material would require self-sensing capabilities;
- It would need to internally process the sensory information;
- It would afford memory to keep track of load histories, or to store load patterns and associated response strategies;
- It would have to have mechanisms for property adaptation, where - in view of the load bearing structure scenario - properties would be mechanical characteristic like stiffness, or strength;
- It would require means of internal communication, as (and we will show this further below) its organization would have to be of distributed nature.

Materials that combine many of these features have been suggested by several authors: Lang et al. **[7][8]** discussed sensorial materials, which stress the capability of sensing. A similar perspective has recently been taken by Saheb and Mekid as well as Mekid et al., designating their concept as nervous or sensor array materials **[9][10]**. The additional aspect of adaptivity is covered in the concept of robotic materials proposed by McEvoy and Correll **[11][2]** as well as Hughes and Correll **[12]**. McEvoy and Correll also offer a practical demonstration, in macroscopic scale, of their concept which employs a thermal effects to adapt stiffness and thus induce shape change in a thermoplastic composite and a robotic application **[3][13]**.

We suggest an alternative term to describe these materials we envisage that stresses their inherent data evaluation capabilities and their adaptivity while also underlining their versatility, which extends far beyond robotic applications: Adaptronic Materials. They perform the adaptation with an integrated sensor-actuator-electronics network.

In any such material, adaptation could be employed on different time scales:

- The shortest time scale, or fast adaptation can cover impact and vibration;
- Medium time scale would address changing quasi-static loads;
- The longest time scale would facilitate adjustment in response to short- and/or medium time scale events, but base its targets in this respect on accumulated, life cycle loads.

While the way such materials can be deployed differs from a temporal perspective, it may also do so from a spatial point of view:

- Global load management may be implemented in load bearing structures subjected to varying loads; while
- Local load management may response to a locally weakened or damaged state of a structure.

In all the above cases, however, the principle of finding an optimum configuration must necessarily be local: A full model of the respective structure cannot be calculated given the limited resources a material-integrated system can rely on. Besides, an inherent disadvantage of a comprehensive system model is its lack of flexibility, which is a major drawback given the fact that one of the major aims of the suggested material is to cope with local overload/wear-out and/or damage, which either imply a change in the systems long-term or immediate load-bearing capacity.

Furthermore, the concept as such can be adapted to several application scenarios: A structural component as implied above may be the most simple one of these. Others include packaging systems that autonomously balance critical factors like force/pressure or acceleration. Besides, gripping systems that have to cope with geometrically ill-defined or fragile objects might be envisaged - here, stiffness adaptation would help once more to control contact pressure and forces.

Several mechanisms are known that allow structural materials to adapt their stiffness. In a recent review, Saavreda Flores et al. have discussed and contrasted these in terms of their underlying principles **[14]**.Examples include …

- Wood cells (mechanism: Change in moisture levels);
- Polymers/spider silk etc. (temperature, stiffness change due to transgression of the glass transition temperature);
- Polymer/cellulose nanofibre composite (Chemical agent induced change of nanofibre and nanofibre-matrix interaction);
- Proteins (sacrificial bonds among molecular cross links between polymer chains);

A major feature of these materials and responsive principles is the time constant of their reaction, as this aspect determines their applicability to managing events on different time scales.

Since practical realization of adaptive materials in large amounts is usually difficult, and since the fundamental mechanical characteristics of many candidate materials are unsatisfactory, or the energy demand to effect a property change within the full volume of an engineering component is forbidding, practical applications of the concept of adaptive structures are most likely to be based on composite or hybrid material solutions.

This is further highlighted by the fact that the materials in question require a distributed control system/architecture

embedded in the material itself, which would contribute to constituting a composite character.

III. Material-integrated Computing Networks

Integrating computation in materials and structures requires a down-scaling of established algorithms, computer architectures, and co-ordination principles. A mobile computer equipped with an Intel *i5*-2520M and 4GB DRAM requires roughly A=150 (CPU) + 500 (DRAM) mm^2=650mm^2 chip area, delivering C=50000 MIPS computing power, but demanding about 35(CPU)+5(DRAM)=40W electrical power. Mid-scale computers, e.g., smart phones, are equipped with low-power devices, e.g., an ARM Cortex delivering C=7500 MIPS, requiring only A=7mm^2 chip area and P=2W power. A normalized computing power weight factor of a computer (considering only the data processing unit) can be defined by $\varepsilon=C/(AP)$, and a relative down-scaling ratio factor is given by $s=\varepsilon_1/\varepsilon_2=C_1A_2P_2/(C_2A_1P_2)$. A scaling factor $\gg 1$ is desired. The down-scaling from an Intel *i5* to an ARM Cortex is expressed in a scaling factor of about 50. Material-integrated computing systems limit the size of a computer to roughly 1mm^2 chip area to limit the mechanical impact of electronic components on the structure. An ATMEL *ATtiny* 20 micro-controller delivers C=12MIPS and requiring A=1mm^2 and P=20mW. Compared with an *i5* this gives a scaling factor of s=63. To summarize, current low-price consumer electronics can deliver about 20MIPS/1mm^2 via a high scaling factor, while current leading edge digital technologies (TSMC 28nm process) expect 4M logic gates/mm^2, an estimated 1000MIPS/mm^2, and 5MBit SRAM/mm^2 (based on **[15]**). New trends in microelectronics pose 3D structuring of electronic devices, increasing the computational power and memory storage multiple times. The functional layer of a silicon die has a width about 10µm (e.g., extracted by chip thinning **[23]**), delivering up to 1000 MIPS/mm^2 or 250MBit/mm^3 assuming a simplified functional-interconnect-isolation layer sandwich structure..

IV. The AAPL Agent Model

The *AAPL programming model* (details in **[16]**) relies on the Activity-Transition Graph behaviour model given by an agent class template containing activity and transition sections. Originally, *AAPL* is an abstract programming meta model that can be implemented with different programming languages. The agent is composed of activities (graph nodes) with conditional or unconditional transitions based on agent data (perception and state). *AAPL* offers statements for parameterized agent instantiation, like the creation of new agents and the forking of child agents inheriting the parent state. Unified agent interaction is provided by using synchronized Linda-like tuple database space access and signal propagation (messages carrying simple data delivered to asynchronous executed signal handlers), discussed in the next section. Agent mobility (migration) is provided. The destination can be a geometric direction (e.g., North, South, ..), a delta-distance vector usable in mesh-grids, a host port, or an URL/IP address.

There are actually two different concrete programming languages basing on the *AAPL* model with existing compilers and agent processing platforms: *AgentFORTH* (*AFVM* platform) and AgentJS (*JAM* platform).

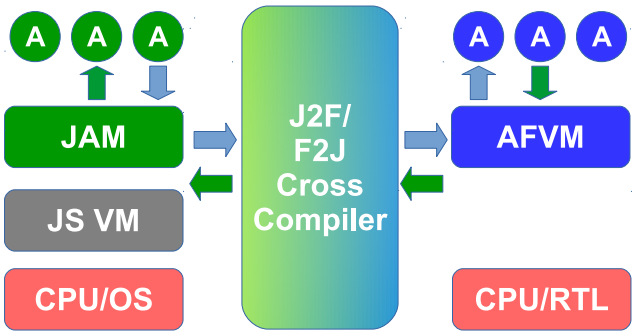

Figure 1. Dual-machine approach coupling JAM and AFVM by using a on-the-fly agent code cross-compiler (J2F: JavaScript-to-FORTH)

AgentJS implements the full *AAPL* model and operational set, whereas *AgentFORTH* implements a sub-set only (though nearly complete).

V. Agent Platforms

Two different programmable agent processing platforms supporting *AAPL* agents are available: (1) *Agent FORTH* Virtual Machine (*AFVM*)**[16][17]**; (2) JavaScript Agent Machine (*JAM*)**[18]**. The *AFVM* is a low-resource platform suitable for hardware integration, requiring about 1000k gates (~1mm^2 chip core area with a TSMC 65nm manufacturing process). Among hardware implementation there are different software implementations fully operational and code compatible with each other. The *JAM* platform addresses the deployment in large-scale networks, i.e., the Internet (of Things), mobile networks, and Clouds. It requires a *JavaScript* execution platform. The deployment of *JAM* in large-scale world-wide networks in an Earthquake monitoring use-case has been demonstrated in **[18]**. Though both platforms support agents with nearly the same behaviour and operational model they are not compatible on code level.

Though in this work MAS are deployed in and evaluated with *JAM* networks, the technological realization of material-integrated computing requires the execution of MAS on the *AFVM* platform. To enable the composition of future large-scale applications ranging from very-low-resource nodes (1mm^3 computers integrated in materials, smart materials and structures) to high-resource devices (generic computers, servers, mobile and embedded devices), both platform architectures must co-exist supporting agent migration seamlessly. This co-existence demands a compatibility layer by introducing an on-the-fly cross-compiler enabling agent mobility between different platform technologies seamlessly, shown in Fig. **1**. This cross-compiler translates agents in *AgentFORTH* object code to *AgentJS* text code agents and vice versa by preserving the entire agent state. This compiler is optimally contained in *JAM* as a service.

AgentJS is a dynamic language with dynamic storage management, whereas *AgentFORTH* is a static language with storage allocated on agent creation. The F2J direction is always possible, but the opposite J2F direction is currently not fully supported and requires constrained *JS* (with pseudo-static memory allocation).

VI. Agent Communication

Communication is central in large-scale distributed systems with a high impact on the overall system performance and stability. Two main issues arise affecting design considerations: 1. Efficiency of the communication with respect to latency, computational complexity, and storage; 2. Addressing and delivery of messages to destination entities. Efficiency is a key factor in self-powered material-integrated low-resource computing networks. Commonly, end-to-end communication is established between processes using IP protocols and computer nodes having unique addresses, which is not suitable for large-scale material-integrated networks and mobile agents. The agent model usually announces autonomy and loosely coupling to the environment, thus without a strong binding to a specific node. Moreover, there is usually no global knowledge of the current position of an agent in the network at all. Basically three approaches are available to exchange information between agents: Tuple-space access, signal messages, and agent migration. Tuple-spaces were proposed in **[19]** and **[20]** as a suitable MAS interaction and co-ordination paradigm.

A. Tuple-Spaces

Tuple-space communication (see Fig. **3** a) exchanges data tuples between entities based on pattern matching, i.e., between processes and agents. The information exchange is data-driven and bases on the data structure and content of the data, and do not require any destination addressing or negotiation between communicating entities. Furthermore, tuple-space communication is generative, i.e., the lifetime of data can exceeds the lifetime of the producer. There are producer and consumer agents. A tuple-space can be localized with a limited access region, commonly limiting data exchange to entities executed on the same network node. Distributed tuple-spaces require inter-node synchronization and base on replication and some kind of distributed memory model.

The *AAPL* agent model originally uses tuple-spaces only for agent communication executed on the same node.

B. Agent-to-Agent (A2A) Signals

Signals are lightweighted messages that are delivered to specific agents (Agent-to-Agent A2A, see Fig. **3** b), in contrast to the anonymous tuple exchange. One major issue in distributed MAS is remote agent communication between agents executed on different network nodes. Though an agent can be addressed by an unique identifier, the path between a source and destination agent is initially unknown. For the sake of simplicity and efficiency, routing table management and network exploration in advance is avoided. Instead the *AAPL* platforms (*JAM*/ *AFVM*) support signal delivery along paths of mobile agents only. That means, a signal from a source node *A* can only be delivered to a destination agent currently on node *B* iff the destination agent was executed (or created) on node *A* some time ago. I.e., two agents must have been executed on the same node in the past. Agent migration and signal propagation is recorded by the agent platform using look-up table caches with time limited entries and garbage collection.

C. Agent-to-Node (A2N) Signals

Previous agent platform implementations only support signal delivery along migration paths based on the destination agent identifier (private, uni-cast) or the agent class (public, broadcast). The new *JAM* platform 2.0 introduces signal delivery of signals to specific remote platforms (remote signaling) based on paths specified by the signal sender agent, shown in Fig. **3** (c). The destination platform node broadcasts the signal to all listening agents executed on this particular node. To simulate private A2A uni-cast (or multi-cast) communication, agents can use a randomly generated signal name only known by the sender and the receiver. This new approach enables interaction between agents never executed on the same node. Furthermore, these remote signals are used to implement distributed tuple-spaces, discussed later.

D. Mobile Agents

Mobile agents can be used to distribute information in networks. They can get data/information from the tuple-space of the current node and store them in remote tuple-spaces by migrating to the respective nodes. The advantage of this approach is the ability to find suitable remote nodes and paths to nodes autonomously or based on content negotiation, and to filter, map, or process the collected data, e.g., using data fusion techniques. The disadvantage is a high communication and processing overhead due to the agent process migration.

Additionally, mobile agents can be used to deliver data in mobile network environments by using a mobile device for spatial migration (piggyback approach, see **[21]** for details), not possible with A2A/A2N signals or remote tuples.

E. Distributed Tuple-Spaces

The new *JAM* platform 2.0 introduces the support for tuple migration using the *collect*, *copyto*, and *store* operations performed by agents. This feature enables the composition of distributed tuple-spaces controlled by agents. The *collect* and *copyto* operations transfer tuples from the local tuple-space to a remote using pattern matching, similar to the *inp* and *rd* operations. The *store* operation send a tuple to a remote tuple-space, similar to the *out* operation.

Remote tuple space access is performed via A2N signals, shown in Fig. **3** (d).

Evaluation

Fig. **2** shows the analysis of simulation results as part of the use-case study discussed in Sec. VIII. for different agent communication strategies. In this simulation, sensor data derived from an artificial physical system, was distributed event-based, leading to varying activity patterns depending on the dynamic change of the mechanical structure under test within a given time window (15000 MAS simulation steps with 100 physical simulation iterations). Note: A simulation step usually executes one agent activity or signal handler. The first approach uses mobile notification agents to distribute sensor data to neighbour nodes (created on each event), which causes an up to 40 times higher communication cost compared with the other approaches. In the second approach distributed notification agents are sent to neighbour nodes one time, and using A2A signals to update sensor data on neighbour nodes managed by the notification agents. The third approach uses remote tuple access based on A2N signals, which causes the lowest communication cost and do not require previous agent distribution.

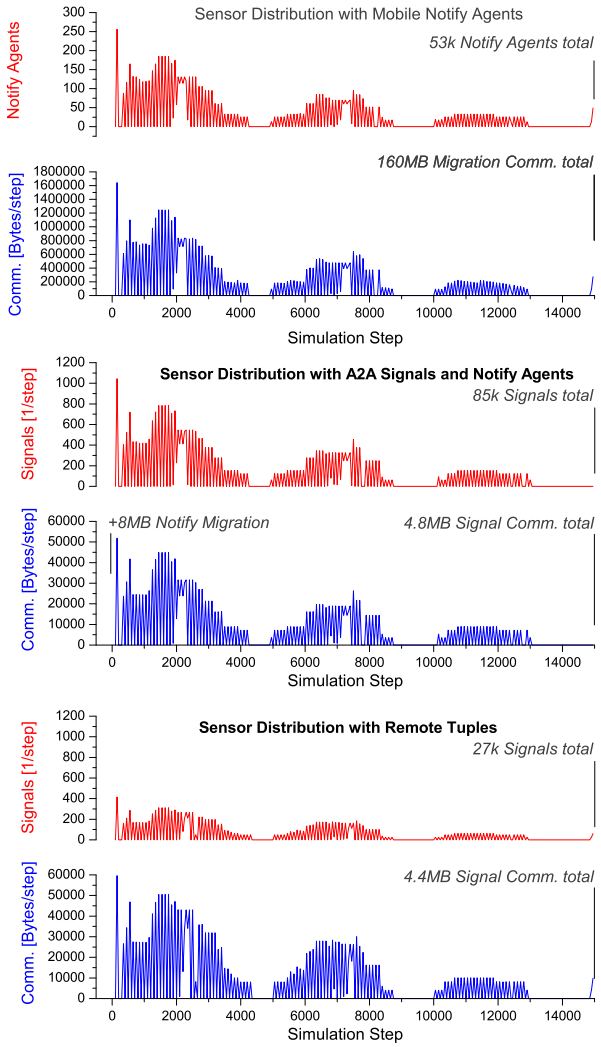

Figure 2. Analysis of different sensor data distribution approaches retrieved from a use-case simulation using (Top) Mobile agents; (Middle) A2A signals and notify agents; (Bottom) Remote tuple access using remote A2N signals.

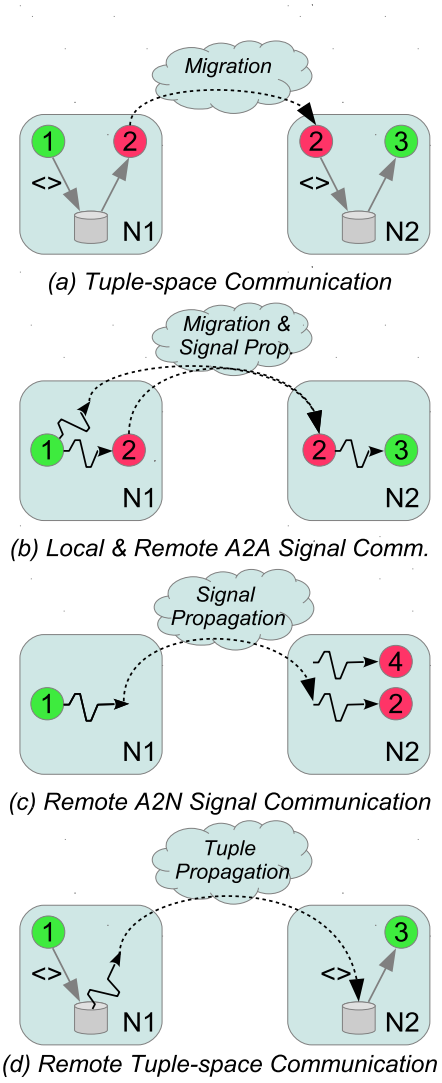

Figure 3. Agent communication in AAPL (a) Tuple-space communication between agents on same node (b) Agent-to-Agent Signals (c) Remote Agent-to-Node Signals (d) Remote Tuple Operation

VII. Multi-Domain Simulation

Combining physical world with MAS simulation is a challenge. Such a multi-domain simulation enables the direct derivation of sensor data from a physical model (sensing part) provision to a computational system, and the investigation of the action of this computational system on the physical system (actuating part). The goal of this multi-domain simulation is to investigate the behaviour of MAS on dynamic (or semi-static) changes of mechanical structures and vice versa. The physical world consists of the mechanical structure given by a physical model (FEM or mass-spring multi-body), sensors, and external loading having impact on the static and dynamic reaction of the structure, including gravity. For the sake of simplicity, the mechanical structure under test is modelled with a mass-spring multi-body system. The multi-body physics simulation is well known from computer games and animations. It can be easily and efficiently integrated in computational systems like MAS simulators.

To this end, the *SEJAM* simulator (*JAM* platform + GUI + simulation control) was extended with a modified physics engine (based on *Cannon.js* and *Three.js*) enabling the direct coupling of the physical model with the computation of agents, shown in Fig. **4**. The *JAM* platform and the *SEJAM* simulator are entirely implemented in *JavaScript*, executed with the Node Webkit (node.js+Chromium HTML browser). A simulation model consists basically of three parts: (1) The MAS behaviour models; (2) The JAM virtual network world; (3) The phyiscal model. All parts are specified in *JavaScript*. In this environment, the physical model can be accessed by all agents. A distributed MAS simulation consists of node and worker agents executed on different virtual *JAM* nodes. There is one artificial agent (world agent) representing the world and managing the simulation, i.e., generating and updating sensor data accessed by the node and worker agents by using the unified tuple-space interface. The world agent can read and modify physical simulation variables, i.e., reading strain, force parameters, and setting material/structure properties (stiffness).

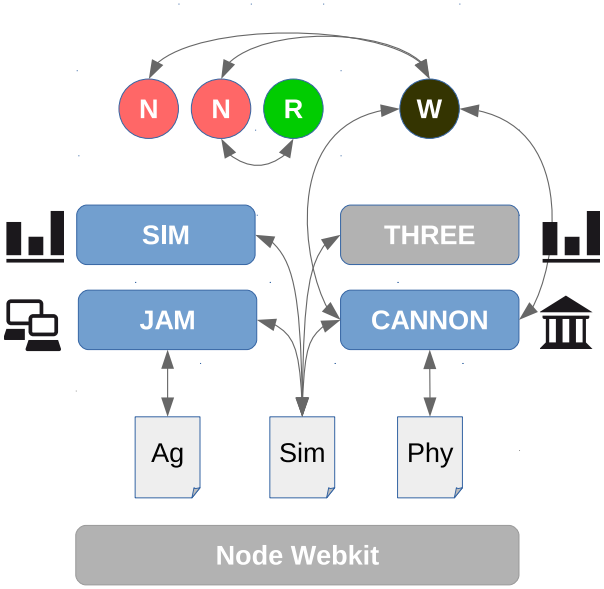

Figure 4. SEJAM2 Simulation Environment (Left) MAS (Right) Physics (Top) Agents, N: Node, R: Mobile Worker, W: World (Middle) Model, Ag: AgentJS, Sim: Simulation, Phy: Cannon Model

The simulator directly controls *JAM*, and a simulation step usually executes one activity or a signal handler of all ready agents.

VIII. Use-Case: Self-Adaptation with MAS

A use-case should demonstrate the coupling of computational with physical systems by MAS. The following MAS is deployed in an three-dimensional mesh-grid network (Fig. **5**, middle) integrated in an artificial mechanical structure. For the sake of simplicity this structure is a plate composed of 5x4x2 mass nodes connected by springs, shown in Fig. **5** (top). It is assumed that each network node provides a *JAM* platform to process agents. Each node is connected to up to six neighbours with communication links. Additionally, each node is connected to neighbour nodes by a set of springs, but only a sub-set is controlled by a specific computer and mass node, shown in Fig. **5** (bottom, spring organization).

The total strain energy U of a mechanical structure is the integral of the strain tensor ε over the volume V of the structure. The discretized strain energy for each node U_i is computed from all attached springs $j=\{1,..,n\}$ with their displacement d_j and current stiffness s_j. The total discretized strain energy of the structure is the sum over all node strain energies:

$$U = \int_V \varepsilon^T \sigma \varepsilon dV$$

$$U_{node} = \sum_{i=1}^{n} d_i^{\,2} s_i \qquad (1)$$

$$U = \sum_{i=1}^{N} U_i$$

The MAS is used to distribute sensor data to neighbour nodes and to negotiate a stiffness variation of spring with the goal to reduce the mechanical strain and total strain energy U either globally or limited to a spatial region.

Each node controls a number of springs to neighbour nodes, shown in Fig. **5** (bottom, a). Each spring has a virtual strain (displacement) sensor attached. Together with the current stiffness parameter value, the spring energy can be computed. It is assumed that the stiffness of each spring can be varied between a lower and an upper boundary $[s_0,s_1]$. A technical reasonable range is ±50% around a nominal stiffness value. The missing sensor values (spring displacement and current stiffness) from other neighbour nodes are delivered by remote tuple operations (see Fig. **5**, bottom, b).

Using the *SEJAM* simulator, the physical and computational system can be simulated simultaneously. The start condition of the physical plate is shown in Fig **5**. Spring and gravity forces have an effect on each mass of the plate. Therefore, the plate will swing between a maximal and a minimal bend until is converges to a static state. The MAS has the goal to minimize the total strain energy U and the maximal strain by modifying the spring stiffness using a simple stiffness swap algorithm, discussed below. The MAS is composed of two different agent classes: The node and the broker agents, having a distinct behaviour described below.

A. *Agent Behaviour: Node Agent*

The node agent is a stationary agent. Its behaviour consists of the following main activities:

init

The initialization activity mainly creates a broker agent, starts the adapt handler timer, and set up the sensor acquisition and spring control.

percept

Getting notifications via the tuple-space data base. Listening for tuples: {`SENSOR`, `STIFFNESS`, `ADAPT`}. The `SENSOR` tuple is either send by the world agent or by surrounding neighbour node agents. The ADAPT tuple is send by the broker agent if there is a change in the spring stiffness was negotiated (either an increase or decrease). Always all controlled spring of a node are set to the same stiffness.

process

New sensor data is processed, and the current strain of each spring and the total node strain energy is computed. Relevant sensor distribution and negotiation events are started. In the case of an energy event (i.e., the current strain energy is above a threshold), an alarm tuple `ALARM(energy, stiffness, ..)` is created that is consumed by a waiting and listening broker agent.

notify

The notification activity distributed sensor and spring data to neighbour nodes by sending remote `SENSOR` tuples. The attached broker agents is notified by `ENERGY` and `ALARM` tuples. Based on sensor data of all known springs attached to this node a `NEIGHBOURS` tuple is send to the broker to inform about connectivity.

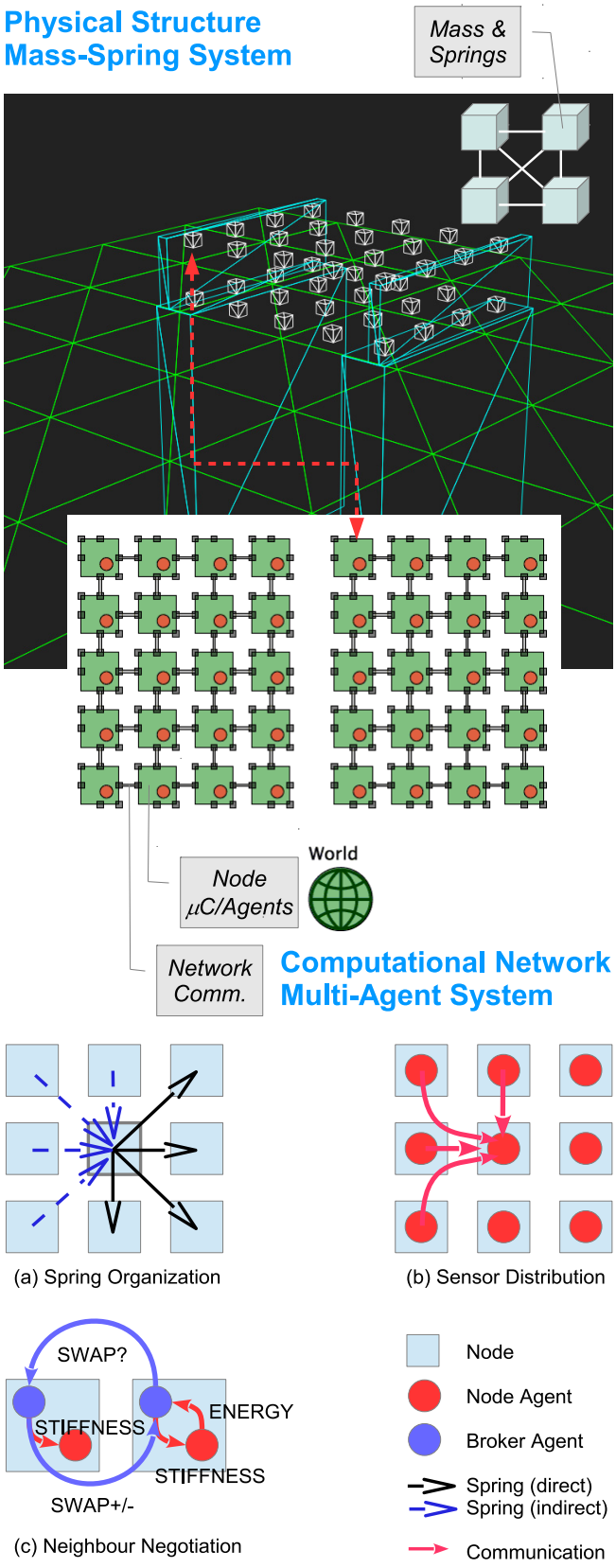

Figure 5. (Top) Physical structure (Middle) The associated computational network deployed with agents. Each node represents a physical mass and a computer, too. (Bottom) Data Distribution and Negotiation in a Region-of-Interest performed by broker and node agents.

adapt

This handler changes the stiffness parameters of all controlled springs. To avoid high stiffness changes, a low-pass filter is applied. This activity is executed periodically by a timer. Always all springs are set to the same stiffness.

B. Agent Behaviour: Broker Agent

The negotiation and broker agent is a mobile agent and is responsible for the self-adaptation of the material based on the current load situation in the neighbourhood and self-regulation. It interacts with the node agents and other broker agents via tuple exchange only.

Its behaviour consists of the following main activities:

percept

Getting notifications via the tuple-space data base. Listening for tuples: {`ENERGY`, `ALARM`, `NEIGHBOURS`, `VOTE`}. The `ENERGY` tuple informs about the current strain energy and stiffness setting of the node, whereas the `ALARM` tuple triggers a voting cycle (transition to vote activity). A `VOTE` tuple containing a `SWAP?` request initiates an election.

vote

A voting cycle is initiated by sending a `VOTE(SWAP?)` request tuple to a randomly chosen neighbourhood node. During the voting or election cycle further `ALARM` and `SWAP?` requests from other nodes are blocked (ignored).

notify

If this broker started a voting cycle and got a `SWAP+` response, it notify the node agent about the successful stiffness swap negotiation by sending an `ADAPT` tuple to commit the election result.

election

This is the election handler managing a swap election initiated by the first `SWAP?` vote after a time-out. The election determines the major vote of all collected votes. If the major vote with the highest energy can be granted, the voter gets a `SWAP+` declaration, otherwise it is declined with a `SWAP-` declaration, and also all voters loosing the competition get a `SWAP-` declaration. The decision for the stiffness swapping bases on the local strain energy and stiffness, compared with the major requesting node's energy and stiffness, i.e., *swapit=this.energy < major.energy && this.stiffness-major.request > this.stiffness.low*.

C. Results & Analysis

The MAS-PHY simulation was performed with the aformentioned plate structure (with 4x5x2 nodes) and the MAS. The stiffness of the springs can be set in the range [30,80] (arb. units) with an initial pre-set of 50. Selected simulation results after 30000 MAS simulation steps and 300 physical simulation iterations are shown in Fig. **6**. A simulation step usually executes one activity of all ready agents or a signal handler of an agent. Due to the damped plate oscillation with different plate bending and strain situations, and the event-based sensor distribution behaviour of the node agents, the signal activity varies on the time-scale, shown for the case of the sensor distribution only in Fig. **6** (a).

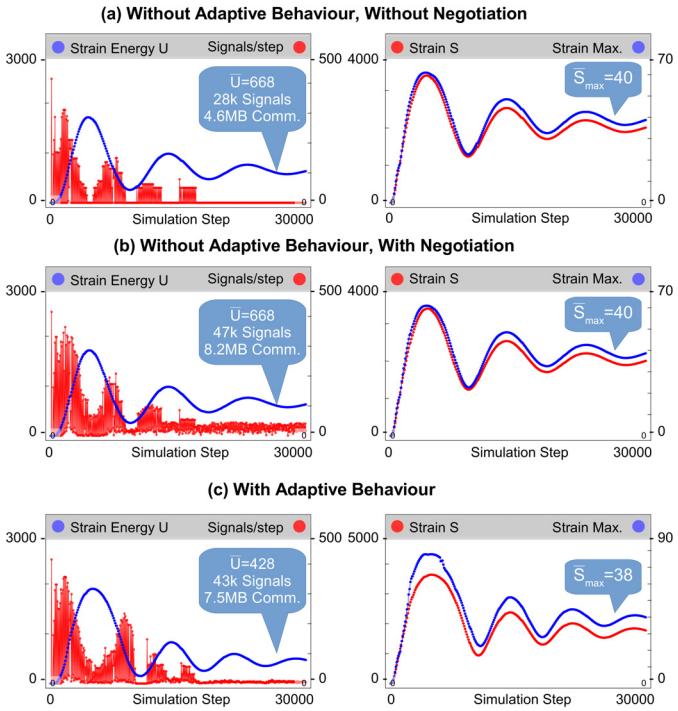

Figure 6. Results from combined MAS-PHY simulation after 30000 simulation steps and 300 physical simulation iterations (blue boxes):
(a) Without physical adaptation of spring stiffness and swap agent negotiation
(b) Without physical adaptarion but with agent negotiation
(c) With adaptive behaviour
(Left) Total physical strain energy U [blue line] of the structure and signal activity [red line]
(Right) Total strain of the structure (red line) and maximum strain [blue line]

If the strain change of the plate decreases, the signal activity decreases, too. Without adaptation behaviour (stiffness swap negotiation), the stationary convergence of the plate results in a maximal strain of S_{max}=40 (arb. units) with a total strain energy of U=688 (arb. units). Fig. **6** (b) shows the signal activity with sensor distribution and negotiation behaviour, but without real spring stiffness modification of the physical system. The negotiation behaviour (using remote tuple with signals) doubles the entire signal activity with a nearly constant level even if the structure is in its stationary state due to the goal of the MAS to reduce U and S_{max}, which has no effect.. Enabling the physical adaptation, this negotiation activity vanishes towards the stationary convergence of the structure, as shown in Fig. **6** (c). Furthermore, the simple swap negotiation behaviour reduces the total strain energy by 40%, though the maximum strain is only reduced by 5%. The overall communication costs in the network are below 10MB.

IX. Conclusions and Outlook

In this work a MAS was successfully deployed in a simulated material-coupled network. Combining MAS with physical simulation is a challenge that was addressed in this work. The *SEJAM* simulator, a GUI and simulation control on top of the *JAM* platform, was extended with a physical simulation engine based on the multi-body simulator *CANNON*. Agents can access and modify the physical model at simulation time directly. The multi-domain simulation speed is overall satisfying. A simulation step (MAS+PHY) requires less than 100ms computation time, typically 10ms, depending on the complexity of the MAS, the *JAM* network, and the physical model.

Though in this work only a simulation with an artificial structure was shown, the MAS can be deployed in a technical real-.world system, integrating a computational network in the adaptive structure. This requires (1) Adaptive material properties; (2) That can be controlled by a distributed computer network consisting of very-low-resource nodes. For the latter requirement there must be a migration from the *JAM* to the *AFVM* platform, requiring only small modifications of the agents.

The first attempt to implement a self-adaptive structure using a simple stiffness swap negotiation showed a reduction of the total strain energy of the structure, i.e., a global effect, although the adaptation was based on multiple local negotiations only. This decentralized operation is a requirement for an efficient scaling towards large-scale networks.

The event-based behaviour and the locally limited access of agents reduces computational and communication costs significantly compared with a traditional centralized global approach.

In future, advanced distributed material-adaptation algorithms should be investigated and evaluated, and finally implemented in a real-world demonstrator using the *AFVM* platform.

Besides, the systems capabilities to alleviate local effects like damage or wear and the capabilities to increase the structure longevity under conditions of fatigue deserve to be studied.

References

[1] C. Jog, R. Haber, M. Bendsøe, *Topology design with optimized, self-adaptive materials*, International Journal for Numerical Methods in Engineering, vol. 37, issue 8 (1994) pp. 1323-1350

[2] M. A. McEvoy, N. Correll, *Distributed Inverse Kinematics for Shape-Changing Robotic Materials*. Procedia Technology 26 (2016) 4 - 11.

[3] M. A. McEvoy, N. Correll,.*Thermoplastic variable stiffness composites with embedded, networked sensing, actuation, and control*.Journal of Composite Materials 49(2014) DOI: 10.1177/0021998314525982.

[4] J. J. Joo1, B. Sanders, G. Washington, *Energy based efficiency of adaptive structure systems*, Published 9 January 2006, IOP Publishing Ltd, Smart Materials and Structures, Volume 15, Number 1

[5] 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.

[6] N. Correll, C. D. Onal, H. Liang, E. Schoenfeld, D. Rus, *Soft autonomous materials-Using active elasticity and embedded distributed computation*, 12th InternationalSymposium on Experimental Robotics, Springer Tracts inAdvanced Robotics 79 (2014)227-240; DOI: 10.1007/978-3-642-28572-1_16.

[7] W. Lang, D. Lehmhus, S. van der Zwaag, Dorey, R. *Sensorial materials-A vision about where progress in sensor integration may lead to*, Sensors and Actuators A 171 (2011) 1-2.

[8] W. Lang, F. Jakobs, E. Tolstosheeva, H: Sturm, A. Ibragimov, A. Kesel, D.Lehmhus, U. Dicke,. *From embedded sensors to*

sensorial materials - The road to function scale integration, Sensors and Actuators A 171 (2011) 3-11.

[9] S. Mekid, N. Saheb, S. M. A. Khan, K. K. Qureshi,. *Towards sensor array materials: Can failure be delayed?,* Science and Technology of Advanced Materials 16 (2015) DOI: 10.1088/ 1468-6996/16/3/034607.

[10] N. Saheb, S. Mekid, F*iber-Embedded Metallic Materials: From Sensing towards Nervous Behavior*. Materials 8 (2015) 7938-7961; DOI: 10.3390/ma8115435.

[11] M. A. McEvoy, N. Correll, *Materials that couple sensing,actuation, computation,and communication*. Science 347 (2015) 1261689; DOI: 10.1126/science.1261689.

[12] D. Hughes, N. Correll, . *Distributed Machine Learning in Materials that Couple Sensing, Actuation, Computation and Communication*.arXiv:1606.03508v1 [cs.LG] 11 Jun 2016

[13] M. A. McEvoy, N. Correll, *Shape Change through Programmable Stiffness*, in: Experimental Robotics, 2015, p. 893.-907, DOI: 10.1007/978-3-319-23778-7_59.

[14] E. I. Saavreda Flores, M. I. Friswell, Y. Xia, *Variable stiffness biological and bio-inspired materials*. Journal of Intelligent Material Systems and Structures 24 (2012) 529-540.

[15] Mark LaPedus, *TSMC devises SRAM cell at 28-nm*, EE Times, 17/6/2009 , original source: talk on Symposia on VLSI Technology and Circuits in Kyoto, Japan, 2009

[16] S. Bosse, *Unified Distributed Computing and Co-ordination in Pervasive/Ubiquitous Networks with Mobile Multi-Agent Systems using a Modular and Portable Agent Code Processing Platform*, in The 6th International Conference on Emerging Ubiquitous Systems and Pervasive Networks (EUSPN 2015), Procedia Computer Science, 2015.

[17] S. Bosse, *Design of Material-integrated Distributed Data Processing Platforms with Mobile Multi-Agent Systems in Heterogeneous Networks,* Proc. of the 6'th International Conference on Agents and Artificial Intelligence ICAART 2014, 2014, DOI:10.5220/0004817500690080.

[18] S. Bosse, *Incremental Distributed Learning with JavaScript Agents for Earthquake and Disaster Monitoring*, International Journal of Distributed Systems and Technologies (IJDST), (2017), accepted, under publication

[19] 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

[20] 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

[21] S. Bosse, E. Pournaras, *An Ubiquitous Multi-Agent Mobile Platform for Distributed Crowd Sensing and Social Mining* , The IEEE 5th International Conference on Future Internet of Things and Cloud, (2017), 21-23 August 2017 , Prague, Czech Republic , accepted, under publication

[22] 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

[23] S. Priyabadini, T. Sterken, M.Cauwe, L. Van Hoorebeke, J. Vanfleteren, *High-Yield Fabrication Process for 3D-Stacked Ultrathin Chip Packages Using Photo-Definable Polyimide and Symmetry in Packages*. IEEE Transactions on Components, Packaging, and Manufacturing Technology 4(2014) 158-167, DOI: 10.1109/tcpmt.2013.2284068.