A Rule System for Network-centric Operation in Massively Distributed Systems

Falko Dressler and Reinhard German

Computer Networks and Communication Systems,
University of Erlangen-Nuremberg, Germany
{dressler,german}@informatik.uni-erlangen.de

Abstract. Sensor and Actor Networks (SANETs) represent a specific class of massively distributed systems in which classical communication protocols often fail due to scalability problems. New control paradigms are needed in this place. This paper outlines biological communication techniques as known cellular biology, which are known as cellular signaling pathways. We show the adaptation of these principles to the world of SANETs by discussing a rule-based control system for network-centric communication and data processing. This system is able to perform data pre-processing such as data aggregation or fusion as well as data-centric communication based on rules that are distributed throughout the entire network. First simulation results demonstrate that this system is able to outperform classical routing approaches in specific SANET scenarios.

Keywords. sensor and actor networks, self-organization, bio-inspired networking, rules-based sensor network, cellular signaling

1 Introduction

Recent advances in microelectronics enabled the development of even smaller and cheaper devices that are primarily used in the domain of Wireless Sensor Networks (WSNs). At the beginning of this research, the envisioned scenario has been smart dust [1], i.e. the deployment of millions of tiny sensor nodes that cooperate on monitoring a given area. Whereas this scenario has not yet become reality, a multitude of algorithms for operation of massively distributed sensor systems have been developed. Based on the research the need for network-centric data preprocessing has been identified as a key challenge due to the observation that communication is much more expensive in terms of energy requirements compared to local processing. Similarly, Sensor and Actor Networks (SANETs) introduced further challenges and requirements. SANETs represent a specific class of sensor networks enriched with network-inherent actuation facilities [2]. In addition to the requirements known from sensor networks, actuation devices, usually named actors [3], are included into the scenario. This requires real-time operation in massively distributed systems and coordination capabilities on a higher abstraction layer.

In this paper, we present a system for network-centric operation in WSNs and SANETs that we named Rule-based Sensor Network (RSN). This system is the result from studies in the context of *bio-inspired networking* – precisely, in the context of cellular signaling cascades. In the following, we summarize the requirements in the domain of massively distributed systems and introduce the biological background. Then, we present RSN in detail and conclude with two first application scenarios.

1.1 Requirements in massively distributed systems

Whereas other application domains exist, we concentrate on WSNs and SANETs. Starting with the first domain, we can identify *scalability* and *energy efficiency* as the most challenging characteristics. Self-organizing algorithms have been developed relying for example on clustering and aggregation techniques to improve scalability and network lifetime [4]. In SANETs, *coordination* aspects need to be solved for sensor-actor coordination as well as for actor-actor coordination [3]. This includes additional communication constraints for network-wide coordination or, at least, local decision taking strategies that lead to an emergent behavior on a higher abstraction layer. Additionally, *real-time constraints* need to be considered as demanded by feedback control in sensor-actor coordination. Some of these challenging requirements are addressed by RSN. This approach basically provides the building blocks for developing network-centric operation and control techniques needed in massively distributed systems such as SANETs.

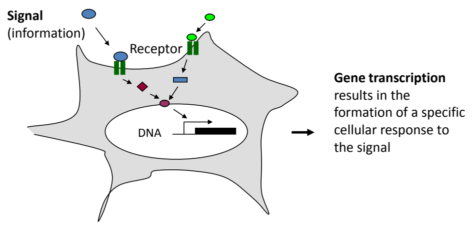

Fig. 1. Cellular signaling refers to the specific reaction according to received signaling molecules; shown is the multilevel transcription of a received protein

1.2 Biologically inspired operation

In the last few years, bio-inspired networking has become a new trend for addressing yet unsolved problems by adapting solutions known in nature [5]. Whereas a broad range of techniques and methods have been studied (e.g., the artificial immune system, swarm intelligence, and evolutionary algorithms), we focus in this paper on a rather new domain, the adaptation of communication and coordination techniques from cellular signaling. Figure 1 sketches the principles of cellular information exchange. Information particles, e.g. proteins, are received by a cell according to the specific binding to a locally expressed receptor (or even a set of receptors) [6]. The activation of the receptor initiates a signaling cascade in which new proteins are created or activated and, finally, a cellular response can be observed, which represents the *specific reaction* of the cell according to the received information. Thus, cellular processes are regulated by interactions between various types of molecules, e.g. proteins. A key challenge for biology in the 21st century is to understand the structure and the dynamics of the complex intercellular web of interactions that contribute to the structure and function of a living cell [7]. To uncover these structural design principles, *network motifs* have been defined as patterns of interconnections occurring in complex networks at numbers that are significantly higher than those in randomized networks [8].

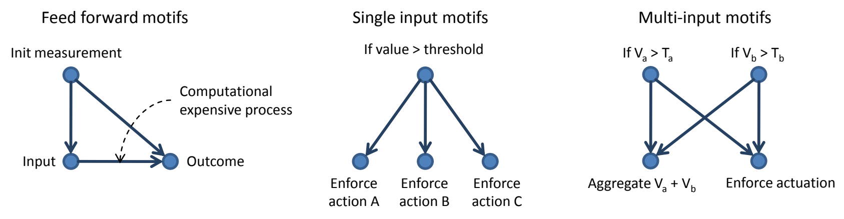

Fig. 2. Typical network motifs in integrated cellular networks

The concept of network motifs is depicted in Figure 2. Please note that this is only a small sample of network motifs in integrated cellular networks [8]. The three basic building blocks of complex networks are shown in this figure together with application examples relevant in SANETs. Feed-forward motifs represent network-inherent mechanisms for controlling (expensive) processes. This can also be seen as an amplification technique. Single-input motifs allow to initiate multiple reactions on a single stimulus. Furthermore, multi-input motifs are depicted. The basic concept is twofold. First, inhibitory or controlling effects can be achieved as two stimuli are required to continue in the signaling cascade. Secondly, if the threshold, i.e. the multiple simultaneous stimuli, is exceeded, a number of parallel actions can be initiated at once.

2 RSN – Rule-based Sensor Network

Inspired by the capabilities of cellular signaling, i.e. the specific reaction to received information and the possibility to build signaling networks defining complex reaction pattern, we developed a rule-based programming system for application in SANETs. The primary design goals were a small footprint to enable the application of RSN on small embedded systems, easily transferable code, flexibility, and scalability for network-wide operations (basically, RSN provides the tools and concepts but the specific application needs to be designed properly as well). The rule-system greatly helps in designing distributed algorithms for use in self-organizing massively distributed systems. Additionally, RSN was inspired by early rule-based systems that have been developed in the context of active networking solutions [9]. Examples are the mobile object system [10] and communicating rules [11].

2.1 Basic concept

The key objectives motivating the development of RSN were improved scalability and real-time support for operation in SANETs. RSN is based on the following three design objectives:

– *Data-centric communication* – Each message carries all necessary information to allow data specific handling and processing without further knowledge, e.g. about the network topology.
– *Specific reaction on received data* – A rule-based programming scheme is used to describe specific actions to be taken after the reception of particular information fragments.
– *Simple local behavior control* – We do not intend to control the overall system but focus on the operation of the individual node instead. Simple state machines have been designed, which control each node (being either sensor or actor).

These goals are achieved by using a simple rule system that enables the node to process received messages and to initiate adequate state and message specific operations. Thus, all received messages are stored in a buffer (source set). Periodically, after a configurable timeout Δt, all messages in the source set are processed by the instructions defined by the rules. Every rule has the form **`if`** `CONDITION` **`then`** `{ ACTION }` as depicted in Figure 3. Each rule specifically selects messages from the source set to apply the corresponding action. Details about the actions and further RSN parameters are described in the following.

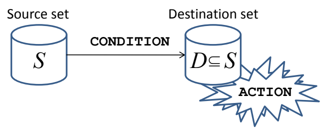

Fig. 3. Each rule selects a number of messages form the source set (`CONDITION`) and applies a (set of) actions to the selected messages (`ACTION`)

2.2 Available actions

The following actions have been implemented in the current version of RSN. Basically, the following categories of actions can be distinguished: *rule execution*, i.e. operations on the received messages; *node control*, i.e. control of the local node behavior (e.g., addition of sensors); and *simulation control*, i.e. actions needed for experiment control without influence on the node behavior.

Rule execution The following actions are meant to be used for network-centric processing of messages. All these actions work on the source message set that has been created by the condition element, i.e. by selecting messages according to a well-defined specific pattern. Examples for the application of the described actions are provided in the next section.

- `!stop` – Early termination of the rule execution. Depending on the current state (i.e., the number and kind of received messages), it may be necessary to stop the current processing of the rule set. The next iteration will start with the first available rule.
- `!drop` – Erases all messages in the current set. Needs to be called if messages have been successfully processed.
- `!dropDuplicates` – All duplicates are discarded according to a unique identifier in each message. This command is needed to emulate for example standard gossiping algorithms.
- `!return` – A new message is created and appended to the source message set.
- `!returnAll` – Copies of all messages in the current set are created and stored in the source message set.
- `!send` – A new message is created and submitted to the lower layer protocol for transmission to neighboring nodes.
- `!sendAll` – Copies of all messages in the current set are created and submitted to the lower layer protocol for transmission to neighboring nodes.
- `!actuate` – A message is sent to locally connected actuators.

Node control Besides the actions for message processing, actions have been integrated to control the local node behavior. Such node control actions allow to enable/disable locally attached sensors and actuators as well as to modify the current rule set, i.e. the local programming of a node.

- `!controlSensor` – A control message is sent to all attached sensors. According to the submitted attributes in `$control`, the behavior of the sensors can be controlled: `rsnSensorEnable` and `rsnSensorDisable` enable or disable the sensor, `rsnSensorSetType` updates the type field of the sensor, and `rsnSensorSetMeasuringInterval` changes the sampling frequency.
- `!controlActuator` – Similarly, this command controls locally attached actuators. The attribute `$control` defines the action: the actuator is enabled or disabled by `rsnActuatorEnable` and `rsnActuatorDisable`, respectively, and `rsnActuatorSetType` updates the type field of the actuator.
- `!controlManagement` – The management plane defines the rule set itself. Again, the `$control` attribute is used to specify the intended action: the rule interpretation can be started or stopped by `rsnManagementEnable` and `rsnManagementDisable`, respectively, the rule set can be replaced in order to modify the behavior of this node using `rsnManagementFromRsnString` or `rsnManagementFromRsnFile`, and the evaluation interval can be configured by `rsnManagementSetEvaluationInterval`.

Simulation control The following actions have been integrated for simplified control of simulation experiments. These actions are not working on a given set of messages. Nevertheless, it is possible to initiate these actions based on the current state of the node, e.g. after the reception of a specific message.

– `!recordAll` – Statistics are recorded for all messages in the current working set. In particular, the following information are stored: ID of the current node, ID of the node that generated the message, node specific ID of the message, globally unique ID of a message, hop count, current time, and delay (elapsed time since message creation).
– `!endSimulation` – This action terminates an experiment. In our OMNeT++ based implementation, the simulation core is notified accordingly.

2.3 Variables and variable handling

All the described conditions and actions work on a set of message parameters or local variables describing the state of the node. In the following, some of the most important variables are introduced. Additionally, selected statistical preprocessing techniques for data aggregation have been integrated into the current version of RSN in order to enable selected application examples. In the following section, we describe and analyze two application examples that inherently benefit from the network-centric preprocessing features provided by RSN.

Message attributes Each message is specifically encoded to allow receiving nodes to determine the meaning of the message and the necessary behavior. This encoding can be changed according to the application scenario. Possible parameters (currently used in the RSN implementation) are listed in Table 1.

Attribute	Description
`$name`	Descriptive name of the message
`$type`	Type of the message; describes the content
`$position`	Position of the source node
`$hopCount`	Number of traversed nodes
`$priority`	Importance factor of this message
`$length`	Length of the message
`$creationTime`	Timestamp describing the creation of the message
`$value`	Message type specific value
`$text`	Further informative text, e.g. to qualify the value

Table 1. Currently implemented message attributes

Node attributes Each node can store and update state information locally. In the context of self-organization, this refers to the local state of an autonomous system. Such information can be updated according to received messages or by other local observations. Table 2 lists the currently implemented node attributes.

Attribute	Description
`:count`	Number of messages in the current working set
`:totalMessageCount`	Number of all messages received by the node
`:hostName`	ID of the current host
`:position`	Position of the node
`:random`	Random value for probabilistic decisions

Table 2. Currently implemented node attributes

Preprocessing features Data aggregation is an important issue in massively distributed systems. Usually, statistical measures are used to describe results received from several nearby nodes. RSN supports such data aggregation techniques by providing a set of preprocessing techniques as summarized in Table 3. All the listed operations process the messages in the current working set.

Command	Description
`@minimum`	Minimum of the selected value
`@maximum`	Maximum of the selected value
`@sum`	Sum of the selected value
`@average`	Average of the selected value
`@median`	Median of the selected value
`@count`	Number of the selected value

Table 3. Implemented preprocessing features

2.4 Implementation

We implemented RSN in form of a C++ library. This library contains all functionality that is necessary to process RSN statements. RSN statements are formulated in a flexible script language. We integrated the RSN library into the OMNeT++ simulation framework in order to execute intensive tests and experiments with different algorithms for data aggregation, probabilistic data communication, and distributed actuation control. OMNeT++ 3.3 is a discrete event simulation environment free for non-commercial use. We also used the INET Framework 20060330, a set of simulation modules released under the GPL. Scenarios in OMNeT++ are represented by a hierarchy of reusable modules written in C++. Their relationships and communication links are stored as Network Description (NED) files. Simulations are either run interactively in a graphical environment or executed as command-line applications.

The developed simulation model is depicted in Figure 4 (left). A single node is depicted consisting of a number of modules. Bottom-up, a wireless communication module is included (WLAN) as well as the rsnRouting module that is currently represented by a simple broadcast module (routing issues can be handled by RSN). The rsnManagement contains all core functions of RSN, i.e.

message handling and rule processing. Finally, the rsnDispatcher module interconnects attached rsnSensor and rsnActuator modules with the rsnManagement.

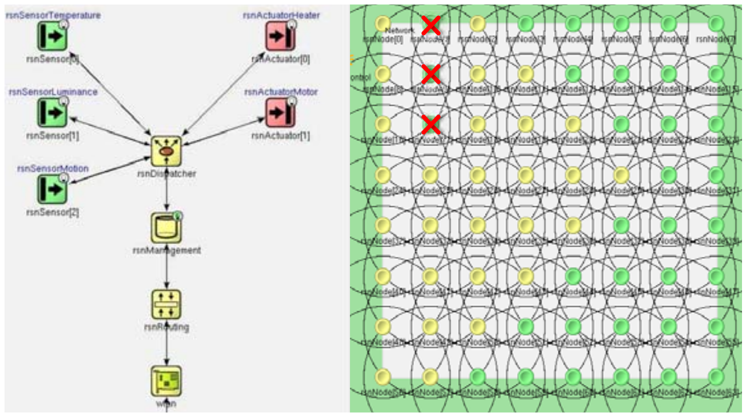

Fig. 4. Simulation model of RSN: integration in OMNeT++ and example setup

3 Applicability of RSN

We evaluated the applicability of RSN in two scenarios. First, we explored network-centric data aggregation as an option to improve the efficiency of probabilistic data communication (gossiping). Secondly, we investigated the capabilities of network-centric actuation control in SANETs in terms of scalability and real-time behavior.

3.1 Data aggregation scenario

Aggregation as a major building block for efficient and scalable data communication and preprocessing in WSNs and SANETs because communication is much more expensive (in terms of energy consumption) compared to processing. In particular, we investigated a typical probabilistic communication approach using RSN: gossiping [12]. The principles are shown in the following RSN program. If a message travels further than `DIAMETER`, it will be discarded. In the first 4 hops, the message is flooded. In all other cases, the message is forwarded according to a random experiment.

```
if $hopCount >= DIAMETER then {
    !drop;
}
if ANY ($hopCount < 4 || :random > GOSSIP-PROB) then {
    !sendAll;
}
!drop;
```

We prepared two scenarios as discussed in [12]. In the first scenario, 100 nodes are distributed on a grid – one corner node is generating 300 messages to be transmitted by the network. Thus, the probability to duplicate messages is very high (according to the initial flooding for the first four hops). In the second scenario, three nodes are turned off in order to build a small linear network at the sending corner node as depicted in Figure 4 (right). Figure 5 shows the number of messages forwarded by a host. Obviously, the nodes close to the source are getting overloaded while distant nodes receive the messages with a pretty low probability.

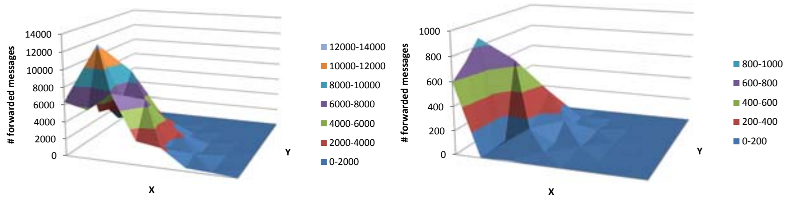

Fig. 5. Gossiping scenario: 100 nodes in a grid (left); linear network at the sending node (right)

Then, we installed a simple aggregation rule that controls the transmission of a single message if multiple messages have been received. The results are depicted in Figure 6. This time, much less duplicates are transmitted and, thus, the load of the network is reduced. At the same time, the dispersion of the messages in the network is increased. Unfortunately, the gossiping algorithm drops all messages (including aggregated ones) with the same probability. Thus, additional rules need to be implemented that correct this behavior, e.g. by defining priorities for aggregated and non-aggregated messages.

```
if :count > 1 then {
    !send($hopCount := @minimum of $hopCount,
          $value := @average of $value);
    !drop;
}
```

3.2 Network-centric actuation

In the following, we present an excerpt from extensive simulations to study network-centric actuation using RSN published in [13]. We compared network-centric actuation control with a classical base station scenario. For the latter one, we used the Dynamic MANET on Demand (DYMO) routing protocol to transmit messages from sensor nodes to a base station and the results back to one out of four available actors. In the RSN scenario, the sensor nodes have been configured with the following program – it represents a simple version of gossiping.

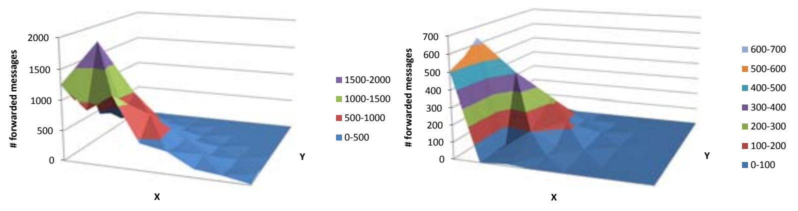

Fig. 6. Gossiping scenario with data aggregation: 100 nodes in a grid (left); linear network at the sending node (right)

```
if $hopCount >= DIAMETER then {
  !drop;
}
if :random <= GOSSIP-PROB then {
  !sendAll;
}
!drop;
```

The actors have an even simpler programming. For each received message, they check whether the THRESHOLD (set to 50, 70, and 90, respectively) has been exceeded and, if necessary, local actuation is initiated.

```
!recordAll;
if $value > THRESHOLD then {
  !actuate($type:=rsnActuatorLightSource,
           $value:=@average of $value);
}
!drop;
```

A number of simulations have been executed with the primary objective to analyze the following characteristics of both evaluated communication and control approaches:

- Real-time support, i.e. the overall latency between measuring a value higher than the particular threshold and the time the message successfully arrived at the actuators. In this context, also the path length is of interest, which is directly proportional to the end-to-end latency and to the message loss probability.
- Overhead, i.e. the number of messages that need to be processed by all the nodes to transmit the necessary data messages. This includes protocol overhead from routing protocols as well as overhead due to duplicated messages for gossiping approaches.

First, the latency of the application messages has been analyzed. We measured the time from creating a sensor message until it was successfully received by the actor. Because only messages exceeding a given threshold are of interest for the actors, we just analyzed the latency after identifying the message as matching this criterion. In Figure 7 (left and middle), results for the RSN scenario are shown in form of boxplots. The graphs differentiate between the deployment scenarios (we evaluated grid and random deployment) and the gossiping probability (set to 0.2, 0.5, and 0.8, respectively). If only the reception of the first copy of the message is considered, the end-to-end delay slightly oscillates around 1.4 ms. The measured maximum is at about 16 ms. The results are nevertheless only meaningful, if all sensor messages can be differentiated, e.g. by a unique id. If this is not possible, the reception of further copies cannot be distinguished form the first one. The measurement results taking this effect into account slightly oscillate around 2.2 ms with a maximum peak at 33 ms.

If we compare these results to the base station scenario as shown in Figure 7 (right), we obviously see that the delays in this scenario are significantly higher (median: 20 ms, mean: 55 ms, and max: 5.700 ms). There are two reasons for this behavior. First, the mean path length is essentially longer as discussed below and, secondly, the on-demand routing protocol takes some time for setting up the routing path before being able to transmit a message. This effect is shown by the comparison between the 60 s and 600 s message generation setups. The route timeout of DYMO has been configured to 120 s. Thus, in the 600 s scenario, almost always the route towards the base and towards the actor nodes will timeout and needs to be reestablished. Further results and more details are available in [13].

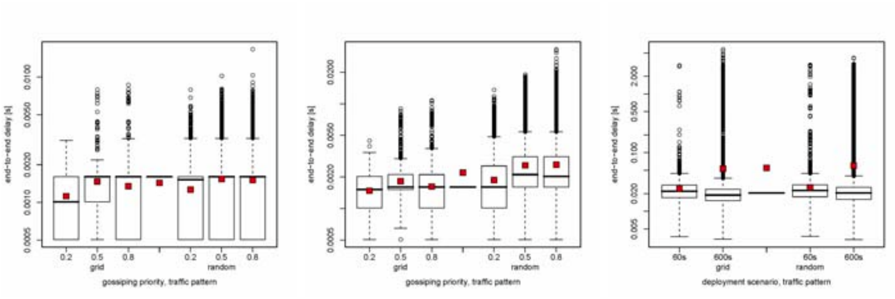

Fig. 7. End-to-end latency. Left (RSN): time until the first copy of a message arrives; middle (RSN) time until any copy arrives; right (base station): end-to-end latency as observed from the application

4 Conclusion

In this paper, we investigated techniques for network-centric data processing in WSNs and SANETs. Based on a sketched overview to cellular information processing, we developed RSN, a rule-based system for sensor network programming. The application range of this approach is manifold; we outlined the advantages based on two examples: data aggregation for optimized probabilistic communication and network-centric actuation control.

The main advantages of RSN are the small footprint of rules and the simple local programming of nodes – making self-organization possible even in large scale sensor and actor networks. In particular, this system allows the quick and heterogeneous reprogramming of (individual) nodes. Therefore, network-centric optimization of the placement of computational intensive rules becomes possible – some concepts can be adapted from the database community: the data stream query optimization problem. Our future work in the context of RSN includes further evaluation of aggregation techniques, the implementation on sensor nodes for "real world" experiments, and intensified investigations of reprogramming techniques.

References

1. Kahn, J.M., Katz, R., Pister, K.: Emerging Challenges: Mobile Networking for "Smart Dust". Journal of Communications and Networking **2**(3) (September 2000)
2. Akyildiz, I.F., Kasimoglu, I.H.: Wireless Sensor and Actor Networks: Research Challenges. Elsevier Ad Hoc Networks **2** (October 2004) 351–367
3. Melodia, T., Pompili, D., Gungor, V.C., Akyildiz, I.F.: A Distributed Coordination Framework for Wireless Sensor and Actor Networks. In: 6th ACM International Symposium on Mobile Ad Hoc Networking and Computing (ACM Mobihoc 2005), Urbana-Champaign, Il, USA (May 2005) 99–110
4. Dressler, F.: Self-Organization in Ad Hoc Networks: Overview and Classification. Technical report, University of Erlangen, Dept. of Computer Science 7 (March 2006)
5. Dressler, F., Carreras, I.: Advances in Biologically Inspired Information Systems - Models, Methods, and Tools. Volume 69 of Studies in Computational Intelligence (SCI). Springer (2007)
6. Alberts, B., Bray, D., Lewis, J., Raff, M., Roberts, K., Watson, J.D.: Molecular Biology of the Cell. 3rd edn. Garland Publishing, Inc. (1994)
7. Barabási, A.L., Oltvai, Z.N.: Network biology: understanding the cell's functional organization. Nature Reviews Genetics **5**(2) (February 2004) 101–113
8. Milo, R., Shen-Orr, S., Itzkovitz, S., Kashtan, N., Chklovskii, D., Alon, U.: Network Motifs: Simple Building Blocks of Complex Networks. Nature **298** (April 2002) 824–827
9. Calvert, K.L., Bhattacharjee, S., Zegura, E.W., Sterbenz, J.: Directions in Active Networks. IEEE Communications Magazine **36**(10) (October 1998) 72–78
10. Vitek, J., Tschudin, C.: Mobile Object Systems - Towards the Programmable Internet. Volume LNCS 1222. Springer (1997)
11. Mackert, L.F., Neumeier-Mackert, I.B.: Communicating Rule Systems. In: 7th IFIP International Conference on Protocol Specification, Testing and Verification. (1987) 77–88
12. Haas, Z.J., Halpern, J.Y., Li, L.: Gossip-Based Ad Hoc Routing. In: 21st IEEE Conference on Computer Communications (IEEE INFOCOM 2002). (June 2002) 1707–1716
13. Dressler, F., Dietrich, I., German, R., Krüger, B.: Efficient Operation in Sensor and Actor Networks Inspired by Cellular Signaling Cascades. In: 1st ICST/ACM International Conference on Autonomic Computing and Communication Systems (Autonomics 2007), Rome, Italy (October 2007)