Efficient Operation in Sensor and Actor Networks Inspired by Cellular Signaling Cascades

Invited Paper

Falko Dressler
Computer Networks and Communication Systems, University of Erlangen, Germany
dressler@informatik.uni-erlangen.de

Isabel Dietrich
Computer Networks and Communication Systems, University of Erlangen, Germany
isabel.dietrich@informatik.uni-erlangen.de

Reinhard German
Computer Networks and Communication Systems, University of Erlangen, Germany
german@informatik.uni-erlangen.de

Bettina Krüger
Dept. of Cellular and Molecular Physiology, University of Erlangen, Germany
bettina.krueger@physiologie2.med.uni-erlangen.de

ABSTRACT

The investigation and the development of self-organizing systems are especially needed for operation and control in massively distributed systems such as Sensor and Actor Networks (SANETs). The main issues addressed by self-organization techniques are scalability, network lifetime, and real-time support. In the literature, biological principles are often cited as inspirations for technical solutions, especially in the domain of self-organization. This concept already resulted in a good number of solutions with significant impact such as ant-based routing and immune system inspired network security solutions. In this paper, another specific biological field is investigated: cellular signaling cascades for event-specific reaction initiated by individual cells in collaboration with their direct neighbors. Information between cells are transmitted via proteins and result in the cascade of protein–protein or protein–DNA interactions to produce a specific cellular answer, e.g. the activation of cells or the transmission of mediators. These processes are programmed in every individual cell and lead to a coordinated reaction on a higher organization platform. We transferred these mechanisms to operation and control in SANETs. In particular, a rule-based processing scheme relying on the main concepts of cellular signaling cascades has been developed. It is relying on simple local rules and providing problem specific reaction such as local actuation control and data manipulation. We describe this Rule-based Sensor Network (RSN) technology and demonstrate comparative simulation results that show the feasibility of our approach.

AUTONOMICS 2007, 28-30 October 2007, Rome, Italy

DOI 10.4108/ICST.AUTONOMICS2007.2191

Categories and Subject Descriptors

C.2.4 [**Computer-Communication Networks**]: Distributed Systems

Keywords

Sensor and actor networks, network-centric operation, rule-based sensor network, cellular signaling cascades

1. INTRODUCTION

Wireless Sensor Networks (WSNs) have become a major research domain in the networking community over the last decade. It has been shown that classical networking techniques are often not suitable or at least insufficient in terms of communication and storage requirements. The main problems are the necessary energy efficiency and the capability to work on low-resource embedded systems. Actually, WSNs are meant to be composed of small battery-driven embedded systems that are communicating over a wireless channel [2, 5].

The requirements are becoming even stronger when Sensor and Actor Networks (SANETs) are considered. In many cases, SANETs represent networks similar to WSNs but with inherent actuation facilities. Such actuators can be a heater or a switch – both activated and driven by network-inherent sensor measures. In other cases, actuators can be mobile robot systems able to perform much more complex actuation. In contrast to typical WSNs, SANETs also face critical real-time operation requirements [1].

The coordination and control of SANETs is still an emerging research area. Usually, the applications follow the classical approach as depicted in Figure 1 (left). Sensor nodes

are continuously analyzing the environment (measurement). The measurement data is transmitted to one or more fixed systems for further processing. Then, the actuators are controlled by explicit commands that are finally executed (actuation). The measurement and the control loop are shown by corresponding arrows. Obviously, long transmission distances have to be bridged leading to unnecessarily high transmission delays as well as to a questionable communication overhead in the network, i.e. possible network congestion and energy wastage.

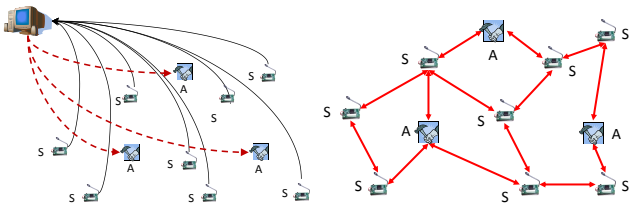

Figure 1: Operation and control of SANETs: centralized (left), network-centric (right)

Self-organization of the SANET is considered the final solution to build energy efficient SANETs that allow real-time operation without complex global state maintenance [10]. The favored system behavior is shown in Figure 1 (right). Self-organization methodologies are used to provide network-centric actuation control, i.e. a processing of measurement data within the network and a direct interaction with associated, i.e. co-located actuators.

A number of approaches related to the main ideas of *autonomic networking*, i.e. the development of self-managing networks, have recently been proposed. One idea is to cluster the available sensor and actor systems into groups that enable simple coordination and control strategies. An example is the distributed coordination framework developed by Melodia et al. [16]. Another approach is to group nodes according to the main objectives of the sensor network such as a given degree of coverage. Gupta et al. [11] have shown that queries into a sensor network can be optimized based on this measure.

Higher level task allocation strategies are also related in the discussed context because actuation represents a specific class of remotely executed tasks. For example, Low et al. [15] employed autonomic networking techniques for task allocation in mobile sensor networks. The use of general self-organization techniques has often been suggested in the domain of communication networks [18]. With respect to SANETs, only few approaches have been published.

In the last few years, we studied some aspects of conceptual similar techniques that have been studied in the domain of cellular biology. These investigations lead to completely different communication and control paradigms in an area that is widely known as *bio-inspired networking*. A great number of solutions are thinkable based on bio-inspired approaches [9].

In this paper, we present a system that we named Rule-based Sensor Network (RSN). It follows the concept of network-centric operation and control [8] based on adapted mechanisms as known from cell biology. The result is an architecture for data-centric message forwarding, aggregation, and processing. We evaluated the performance of this system using a comprehensive simulation model. According to the simulation results, RSN outperforms classical ad hoc routing techniques by far – in a typical SANET scenario.

The rest of the paper is organized as follows. Section 2 introduces the concepts of cellular signaling cascades. Section 3 outlines the ideas and the internal system aspects of RSN. The simulation model as well as the obtained results from the performance evaluation are depicted in Section 4. Finally, Section 5 concludes the paper.

2. CELLULAR SIGNALING

The focus of this section is to briefly introduce the information exchange in cellular environments [3, 17, 20]. Information exchange between cells, called *signaling pathways*, follows the same principles that are required by network nodes. A message is sent to a destination and transferred, possibly using multiple hops, to this target.

Within complex organisms, such as mammals, cells are organized according to their physiological function. Permanently, neighboring cells have to inform each others that everything is normal, e.g. by sending growth factors, telling the neighbor: "Keep on growing ". But also information from far away in the body can be received via a "telephone wire" called the blood. Via these pathways, information can be received and sent and have to be processed by the receiving cell. From a local point of view, the information transfer works as follows. One way is that the reception of signaling molecules via receptors. The receptor can be located on the surface of the cell. Typically, these receptors can bind an information molecule on the outside of the cell and during this binding it is activated, e.g. by a change in its sterical or chemical conformation (phosphorylation of defined amino acids). The activated receptor molecule is able to further activate signaling molecules inside the cell resulting in a "domino effect", because these activated signaling molecules in turn can activate further downstream signaling molecules see Figure 2 (1-a). As an example the signaling via several growth factors can be mentioned.

Another example for the information transfer via receptors is the following. Small molecules like steroid hormones reach the cell of destination via the blood. This remote information exchange works as follows. A signal is released into the blood stream, the medium that carries it to distant cells. The hormone can pass the cell membrane and enter the cell. Within the cell the receptor binds the hormone. The ligand (hormone)-receptor complex can enter the nucleus of the cell and initiate gene transcription which leads to the production of an "answer".

This answer can be a different behavior of the cell. As an example may serve the signaling via the hormone aldosterone binding to mineralocorticoid receptor expressed in e.g. some cells of the kidney (e.g. the Renin-Angiotensin-Aldosteron system [12]. A schematic construction is shown in Figure 2 (1-b). Another example is the activation of the immune system.

The interesting property of this transmission is that the information itself addresses the destination. During differentiation a cell is programmed to express a subset of receptor in order to fulfill a specific function in the tissue. In consequence, hormones in the bloodstream affect only those cells expressing the correct receptor. This is the main reason for the specificity of cellular signal transduction. Of course, cells also express a variety of receptors which regulate the cellular metabolism, survival, and death.

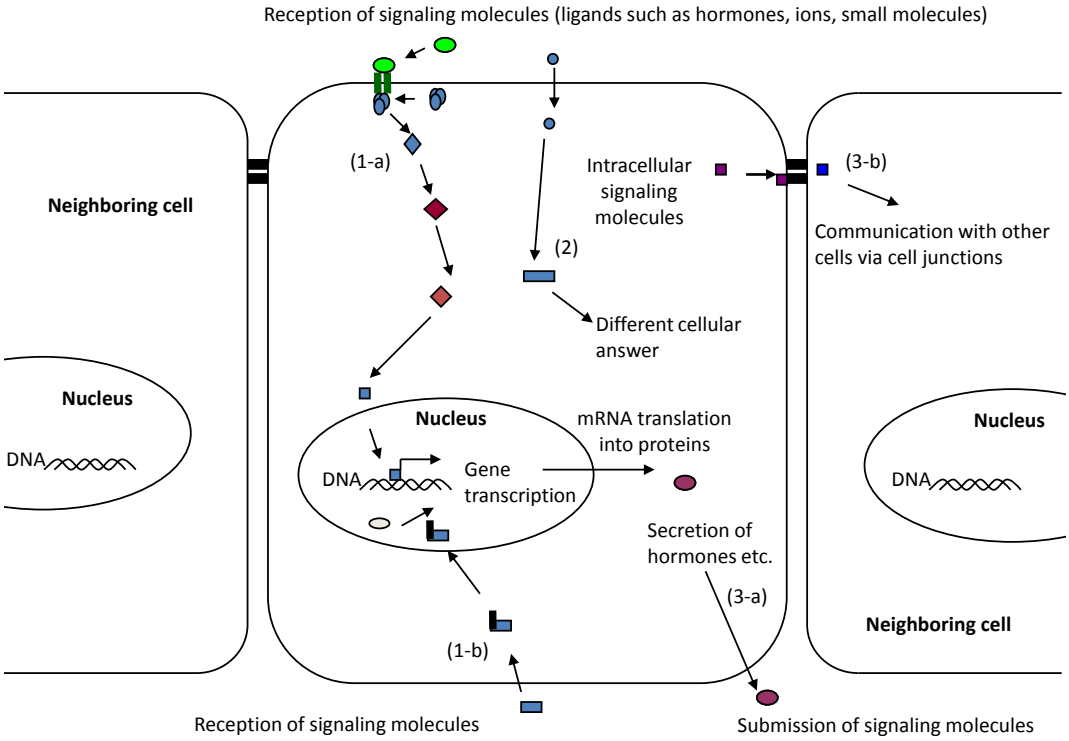

Figure 2: Detailed overview to signaling cascades for intra cellular and inter cellular communication

In principle these signaling pathways are not as simple as described here. Many of these signaling pathways are interfering and interacting. Different signaling molecules are affecting the same pathway. Inhibitory pathways are interfering with the straightforward signal transduction. To sum up, the final effect is dependent on the strongest signal. The effect of such a signal transduction pathway is mostly gene transcription. Gene transcription means that the cell respond to incoming the signal by translation of specific mRNA into new proteins, which are then secreted (transported out of the cell), where it can induce signaling processes in the cell's direct environment. The cellular answer is a specific response according to the received signaling molecules and the current constitution of the cell. For example, signaling molecules can be created to send messages to other cells. Additional signaling molecules may affect the established signaling cascade towards the nucleus. The cellular answer is relying on the nucleus to initiate the desired process.

Other possibilities are the reorganization of intracellular structure such as the cell cytoskeleton or the internalization and externalization of molecules in and out of the cell as a response to the received message.

This *specific response* is the key to information processing. It depends on the type of the signal and the state of the cells (which receptors have been built and which of them are already occupied by particular proteins). Finally, a specific cellular response is induced: either the local state is manipulated and/or a new messaging protein is created. In this scheme different possibilities are shown how cells can transfer answers. In Figure 2 (3-a) the response to a received information particle is gene transcription and the production of a specific protein serving as a new message. This protein can be submitted into the extracellular space, e.g. secretion of hormones into the blood stream to activate cells far away as described above.

Additionally, messages can be forwarded to a neighboring cell via a paracellular pathway. In this case intracellular signaling molecules are transfered via junctions between cells. Congeneric cells develop several forms of junctions. One example are so-called "gap-junction" which represent tunnels where small molecules such as calcium ions or cAMP (cyclic-adenosine-mono-phosphate) can be transferred to the neighboring cell. This pathway is shown in Figure 2 (3-b).

Finally, other non-protein molecules such as nitric oxide can enter the cell which are directly processed in a biochemical reaction. The resulting product of the reaction directly changes the behavior or state of the cell. For example, nitric oxide leads to smooth muscle contraction, schematically shown in Figure 2 (2).

The lessons to learn from biology are the efficient and, above all, the very specific response to a problem, the shortening of information pathways, and the possibility of directing each problem to the adequate helper component. Therefore, the adaptation of mechanisms from cell and molecular biology promises to enable a more efficient information exchange. Besides all the encouraging properties, bio-inspired techniques must be used carefully by modeling biological and technical systems and choosing only adequate solutions.

3. Rule-based Sensor Network

The key objectives motivating the development of RSN were improved scalability and real-time support for operation in Sensor and Actor Networks. RSN is based on the

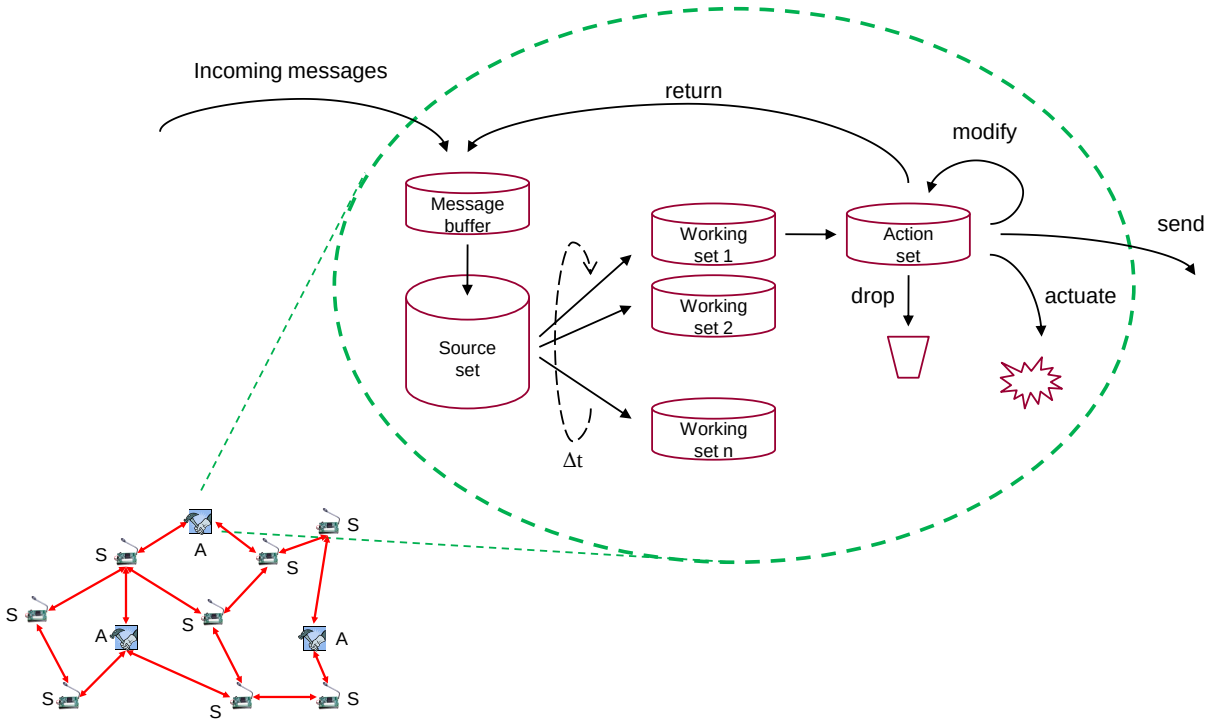

Figure 3: The working behavior of a single RSN node. Received messages are stored in a buffer, selected to a working set according to specific criteria, and finally processed, i.e. forwarded, dropped, etc.

following three design objectives that enable the mentioned 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).

In the following, the concepts of RSN are outlined and the intended use is depicted according to some examples relevant in the domain of SANETs.

3.1 Data-centric operation

The RSN architecture has been developed for SANET programming and operation that consequently follows the data-centric communication approach and enforces a complete network-centric operation [8]. Thus, instead of carrying address information, each message is encoded using a (type, content) pair. The type describes the message and the attached content. The data itself will usually include a value and application specific meta information such as a geographical position or priority information.

Similar data-centric communication schemes have been proposed in the context of probabilistic data dissemination. The best known approach is gossiping [13]. Its key objective is essentially reduced communication overhead compared to other approaches – whereas the probability that messages reach the destination might be very low in specific scenarios such as linear setups. Optimized gossiping approaches are available but out of scope of this article.

The message encoding and processing in RSN are similar to the ones suggested by directed diffusion [14]. Even though the communication scheme is completely different, directed diffusion and RSN both rely on the identification of messages according to representative type information.

Each message could be encoded as follow:

```
M := { type, region, confidence, content }
```

At least `type` and `content` are needed for every message processed by RSN. Additional parameters such as a geographical region or a confidence level can be added in order to provide meta information for optimized data processing. Examples for such messages are:

- `{ temperatureC, [10, 20], 0.6, 20 }`
  A temperature of 20°C was measured at the coordinates `[10, 20]`. The confidence is 0.6, therefore, a low-quality sensor was employed.
- `{ pictureJPG, [10, 30], 0.9, "binary JPEG" }`
  A picture was taken in format JPEG at the coordinates `[10, 30]`.

3.2 RSN architecture

An extensible and flexible rule system is used to evaluate received messages and to provide the ”programming” in a similar way as performed for the cellular response. Even though the message handling in biological cells is more so-

phisticated, the basic principles including the processing instructions (the DNA) are modeled.

The local behavior is controlled by a rule interpreter in form of simple state machines. The interpreter is applying the installed rules to previously received messages. It uses a queuing subsystem that acts as a generic receptor for all messages and keeps them for a given time. This time control is necessary to prevent queue overflows due to received messages of unknown type.

Figure 3 depicts the working behavior of a single RSN node. After receiving a message, it is stored in a message buffer. The rule interpreter is started periodically (after a fixed Δt) or after the reception of a new message. The period Δt is critical for particular applications such as data aggregation: the longer messages are stored before being processed, the better the possible aggregation ratio (more messages can be aggregated into a single one); and the longer the period, the longer the artificially introduced per hop delay.

Each rule that is used to process the received messages consisting of two parts, a condition and an action, as shown in Figure 4. Starting with this overview, we will continue to use the specific RSN syntax to outline rules in the examples. The condition is intended to associate messages to a given rule, i.e. an action. In RSN, the specific reaction on received data is achieved by means of predicates. RSN is able to select all messages of a given type or messages with specific content attributes. All selected messages are stored in so called working sets.

```
if PREDICATE then {
  ACTION
}
```

Figure 4: Basic rule composition depicted in RSN syntax. Messages are selected by a predicate and processed by an action

The predicate work on parameters of the received messages or on local state information. All parts of a message can be accessed, e.g. to select all messages of a given type (`$type == "Temperature"`), to identify important messages (`$priority > 0.8`), or to test whether given thresholds have been exceeded (`$value > THRESHOLD`). Local state information includes the current time or parameters specific to the current message evaluation such as the number of messages in the buffer (`:count > 1`) or just a random value (`:random > 0.5`).

All messages in the temporary working set are processed by the given actions. In particular, RSN performs one of the following actions:

- `modify` – A message or a set of messages can be modified, e.g. to fuse the carried information with locally available meta information.
- `return` – Messages may be returned to the message buffer for later processing, e.g. for duplicate detection or improved aggregation.
- `send` – Obviously, a node needs to be able to send messages. This can be a simple forwarding of messages that have been received or the creation of completely new messages needed to coordinate with neighboring nodes.
- `actuate` – Local actuators can be controlled by received messages, e.g. to enable sensor-actor feedback loops.
- `drop` – Finally, the node needs to be able to drop messages, which are no longer required, e.g. because they represent duplicates or because an aggregated message has already been created and forwarded.

The `send` and `return` actions may send or return the original message but also create completely new messages. This action is used for example for data aggregation. The aggregated message may for example be used to carry the mean value of all source messages and the standard deviation.

Each rule may contain any number of actions. For example, a message can be modified and forwarded. Additionally, the rule may need to discard the message to prevent it being processed by other rules in the system.

3.3 Application examples

In order to demonstrate the capabilities, two simple application examples are depicted in the following. First, the probabilistic data forwarding technique gossiping is reproduced in RSN. The algorithm according to [13] forwards packets with a given gossiping probability p. In order to cope with special cases (problems) such as linear networks, flooding is used for the first n hops.

Each message is assumed to be encoded in the following way:

```
M := { hopCount, content }
```

Then, the gossiping algorithm can be formulated as follows (again, we are using the implemented RSN syntax for the examples):

```
# infinite loop prevention
if $hopCount >= networkDiameter then {
    !drop;
}
# flooding for the first n hops
if $hopCount < n then {
    !sendAll;
    !drop;
}
# gossiping
if :random < p then {
    !sendAll;
    !drop;
}
# clean up
!drop;
```

In the first block, all messages are selected that have a `hopCount` greater or equal to `networkDiameter`. These messages are silently dropped (`!drop`). This command is included to prevent infinite loops. The second block selects all messages with `hopCount` smaller than n and forwards these messages (`!sendAll`). After processing the messages, they are discarded. The third block selects all remaining messages in the working set if an on-demand calculated random value (`:random`) is smaller than the gossiping probability p.

These messages are forwarded and all remaining messages are dropped.

From this simple example, two mechanisms become obvious. First, each command operates on sets of messages instead of single messages. Secondly, messages remain in the working set until they are dropped. Thus, multiple commands may be applied to particular messages.

A second example should demonstrate more sophisticated applications. In this example, the sensors are used to measure the temperature. Data aggregation is performed to reduce the number of messages in the system. Additionally, critical temperature values are observed and alarm messages are created if a threshold has been exceeded.

The message encoding is similar to the previous example:

```
M := { type, position, content, priority }
type := ( temperature || alarm )
```

The complete algorithm can now be written as follows:

```
# test for exceeded threshold and
# generate an alarm message
if $type = temperature &&
   $content > threshold then {
    !actuate(buzzerOn);
    !send($type := alarm, $priority = 1);
}
# perform data aggregation
if $type = temperature &&
   :count > 1 then {
    !send($content := @median of $content,
      $priority := 1 - @product of $priority);
    !drop;
}
# message forwarding, e.g. according
# to a simplified gossiping algorithm
if :random < $priority then {
    !sendAll;
    !drop;
}
!drop;
```

In this example, the three command blocks actually perform different operations. The first block tests the temperature value and, if the threshold is exceeded, a local actuation is enforced (a buzzer is turned on – `!actuate`) and a new alarm message is generated with message priority set to one (`!send`). In the second block, all temperature messages are aggregated (if more than one has been received – `:count`). The content is set to the median of all temperature values and the message priority is increased. Finally, the last block is in charge of message forwarding.

From these two examples, it can be seen that RSN provides a powerful set of commands to enable in-network operation and control for SANETs. Nevertheless, a number of open issues still exist:

- Handling of unknown messages – Which action should be performed if unknown messages, i.e. messages of unknown type, have been received? Basically, two decisions are possible, drop vs. seamless forwarding, while not being appropriate in all application scenarios.
- Period of RSN execution Δt – The duration of messages stored in the local node introduces an artificial per-hop delay. The optimal value for Δt affects the aggregation quality vs. real-time message processing.
- Rule generation and distribution – So far, we considered homogeneously programmed nodes. This is not necessarily the optimal case. Also, new rules may be required during the lifetime of the network. The rule deployment needs further research in terms of diffuse or random distribution vs. global optimization.

4. SIMULATION EXPERIMENTS

In order to evaluate the efficiency of RSN, we compared it to the typical setup used in sensor network scenarios. Multiple sensor nodes are continuously measuring environmental conditions and transmit this information to a central base station. This, in turn, will analyze the received results and engage the installed actors accordingly. For the communication, we chose Dynamic MANET on Demand (DYMO), which is a popular routing protocol used in the ad hoc and sensor network community. We also created the same setup with RSN for a direct comparison.

4.1 Setup and scenario

For the simulations, we developed a simulation model using OMNeT++ 3.3 [19], a simulation environment free for non-commercial use, and the INET Framework 20060330, a set of simulation modules released under the GPL. OMNeT++ runs discrete, event-based simulations of communicating nodes on a wide variety of platforms and is getting increasingly popular in the communications community. 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.

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.

For comparison of network-centric actuation control with classic base-station approaches, we investigated the following scenario. A large number of sensor nodes are considered to measure environmental conditions such as the temperature. If measurements exceed a given threshold, actuation devices are triggered. Such actuators are able to interact with the environment or to initiate secondary events and actions.

In order to evaluate the communication behavior in this scenario, we created a simulation model in which 100 sensor nodes are placed on a rectangular playground. The nodes are either distributed in form of a regular grid or on a random pattern. In addition to these sensor nodes, four actor nodes are included in the middle of each quadrant. This setup is depicted in Figure 5.

In our example, we configured all sensors to periodically send their sensor readings towards the actuators. In the base-station scenario, the central base station checks the received measures whether they exceed the given threshold

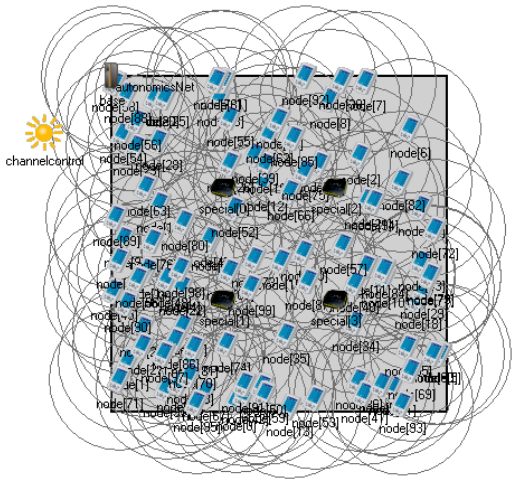

Figure 5: Simulated scenario

and forwards only appropriate messages to the actors. In the network-centric operation scenario, all messages are distributed with a certain gossiping probability and only the actors are able to check the threshold. All the variable parameters using in the simulation are summarized in Table 1.

Table 1: Variable simulation parameters

Parameter	Values
Sensor reading period	60 s, 600 s
Sensor readings	uniform in `[0, 100]`
threshold for actuation	50, 70, 90
RSN gossiping probability	0.2, 0.5, 0.8

For the base-station scenario, we used the DYMO routing protocol [4], which can be considered a de-facto standard in the ad hoc and sensor networking community. In particular, we used the implementation of DYMO available for OMNeT++ [7].

For all communications, wireless modules working according to the IEEE 802.11b standard have been used. All simulation parameters used to parameterize the modules of the INET Framework are summarized in Table 2.

In the RSN scenario, the sensor nodes have been configured with the following program. It ensures that all messages are forwarded with a probability of `GOSSIP-PROB` (set to 0.2, 0.5, and 0.8, respectively) over a maximum distance of `DIAMETER` (for the presented simulation results, we configured the maximum hops count to four). The `!recordAll` command is used for statistical purposes only.

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

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

```
!recordAll;
if $value > THRESHOLD then {
  !actuate($type:=rsnActuatorLightSource,
           $value:=@average of $value,
           $priority:=2);
  !drop;
}
!drop;
```

Table 2: INET framework module parameters

Parameter	Value
`net.headerLengthByte`	20 byte
`net.ROUTE_TIMEOUT`	120 s
`net.ROUTE_DELETE_TIMEOUT`	200 s
`net.NET_DIAMETER`	10
`mac.address`	auto
`mac.bitrate`	2 Mbit/s
`mac.broadcastBackoff`	31 slots
`mac.maxQueueSize`	14 Pckts
`mac.rtsCts`	true
`decider.bitrate`	2 Mbit/s
`decider.snirThreshold`	4 dB
`snrEval.bitrate`	2 Mbit/s
`snrEval.headerLength`	192 bit
`snrEval.snrThresholdLevel`	3 dB
`snrEval.thermalNoise`	−110 dB
`snrEval.sensitivity`	−85 dB
`snrEval.pathLossAlpha`	2.5
`snrEval.carrierFrequency`	2.4 GHz
`snrEval.transmitterPower`	1 mW
`channelcontrol.carrierFrequency`	2.4 GHz
`channelcontrol.pMax`	2 mW
`channelcontrol.sat`	−85 dBm
`channelcontrol.alpha`	2.5

4.2 Measurement results

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.

In order to increase the statistical significance of the simulation experiments, all simulations have been executed five times (runs). In each experiment, all the 100 sensor nodes send exactly 200 packets. After starting the simulation, the time for each sensor to start its local activities is uniformly distributed over the first 60 s. This behavior first models the initialization of real sensor nodes at arbitrary times and,

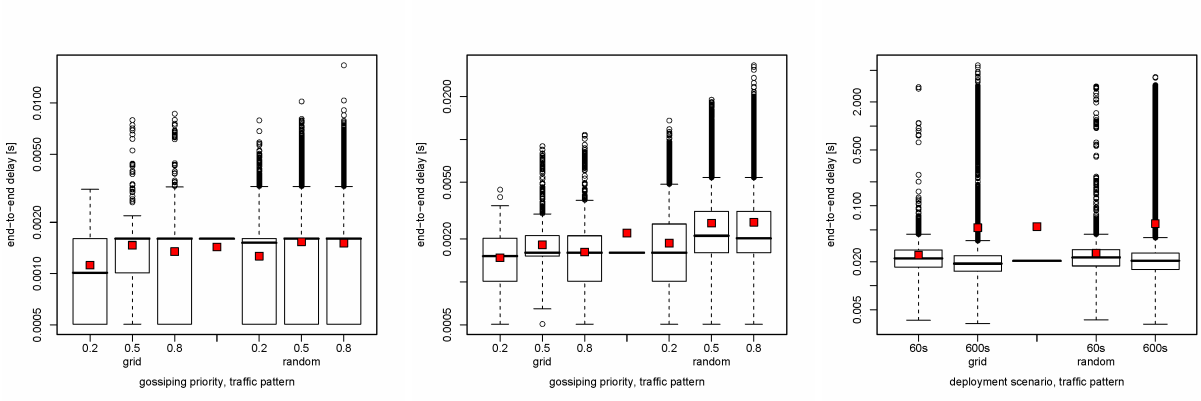

Figure 6: End-to-end latency. Left (RSN): time until the first copy of a message arrives; middle (RSN) time until any copy arrives; right (DYMO): end-to-end latency as observed from the application

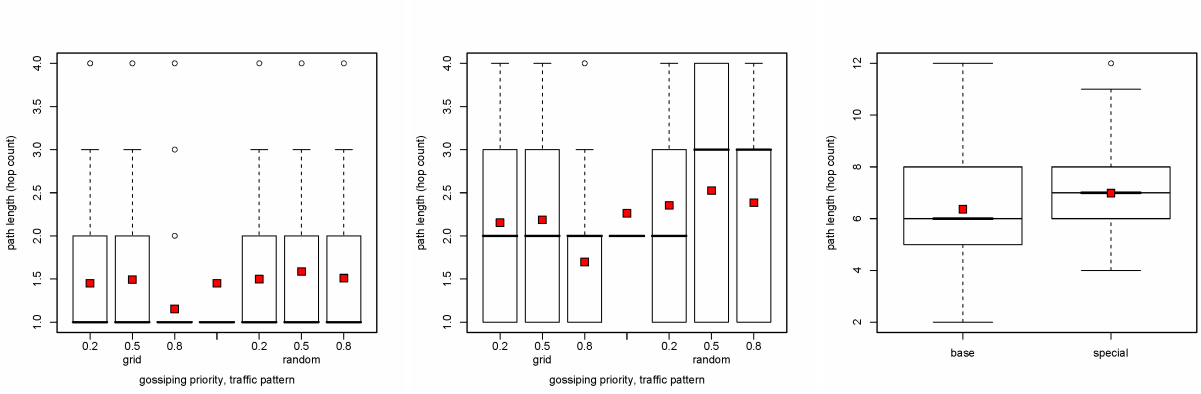

Figure 7: Path length (number of hops). Left (RSN): path length for the first copy of a message; middle (RSN): path length for any copy; right (DYMO): path length towards the base station and between the base and the actors

secondly, it prevents collisions on the MAC layer due to synchronization effects.

All results are shown as boxplots. For each data set, a box is drawn from the first quartile to the third quartile, and the median is marked with a thick line. Additional whiskers extend from the edges of the box towards the minimum and maximum of the data set, but no further than 1.5 times the interquartile range. Data points outside the range of box and whiskers are considered outliers and drawn separately. Additionally, the mean value is depicted in form of a small filled square. In most graphs, the overall mean and median are shown in the middle bar.

4.2.1 *Real-time support*

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.

Figure 6 shows the measurement results. In all the shown graphs, all setups as depicted in the previous subsection and all the simulation runs are integrated to show the statistical effects of single parameters. In Figure 6 (left and middle), results for the RSN scenario are shown. The graphs differentiate between the deployment scenarios and the gossiping probability. If only the first 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 DYMO scenario as shown in Figure 6 (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.

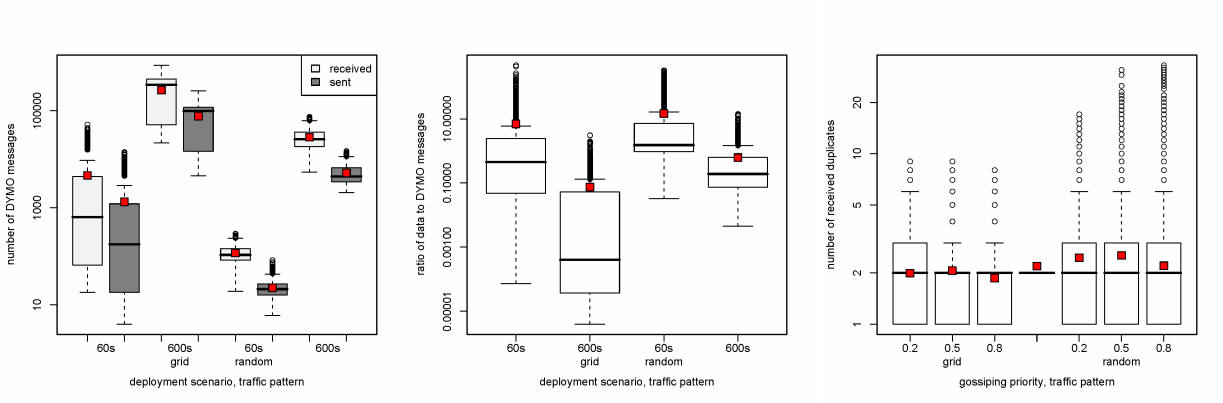

Figure 8: Overhead due to protocol characteristics. Left (DYMO): number of protocol messages; middle (DYMO): ratio of data to DYMO messages; right (RSN): number of received duplicates

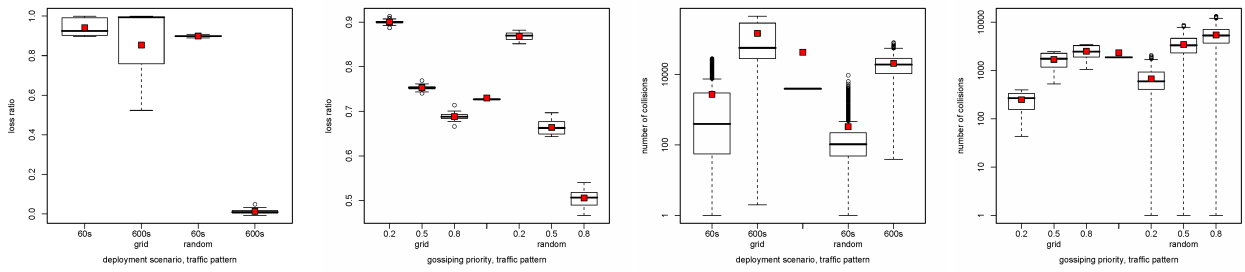

Figure 9: Loss ratio and collisions. 1st: DYMO loss ratio; 2nd: RSN loss ratio; 3rd: DYMO collisions; 4th: RSN collisions

Secondly, we analyzed the path length, i.e. the hop count, in the same experiments. The results are depicted in Figure 7. Obviously most messages are transported over only two hops in the RSN scenario. This also explains the log latency communication. In the DYMO scenario, each message needs to be transmitted first to the base station (which requires in average about six hops) and then it is forwarded to the actor nodes (requiring in average seven hops). Thus, we could expect a factor of about 3-7 for the latency difference between the RSN and the DYMO scenario. Nevertheless, as shown in Figure 6, the factor is about 10-55. The only explanation for this high factor is the overhead according to the on-demand routing.

4.2.2 *Overhead*

In the previous paragraph, we have shown that the overhead may essentially affect the real-time support of the employed communication techniques. Especially in the context of SANETs, the overhead also characterizes the energy efficiency of the entire system, and thus, the possible *network lifetime* [6].

In Figure 8, the protocol overhead is depicted. For the DYMO scenario with 60 s sampling period, we can see that each node needs to send in average between 20 and 1,300 DYMO messages in order to transmit 200 data messages in the random and grid deployment scenario, respectively. If the sampling rate becomes too small, i.e. if the route timeouts of DYMO are triggered, in average between 5,200 and 76,000 DYMO messages need to be sent for delivering 200 data messages. The primary reasons for these high numbers are the high probability of multiple nodes searching simultaneously for a given destination and the increased collision probability (see below). The ratio of data to DYMO messages is shown in Figure 8 (middle) – this figure takes all data messages into account, whether generated at the local node or forwarded on behalf of other nodes.

In the RSN scenario, in almost all measurements about two duplicates are received by the actor nodes. Thus, an overhead factor of two can be noticed as shown in Figure 8 (right). According to the probabilistic forwarding scheme, some peaks up to 33 duplicates can be recognized. This effect has been expected and it can, according to the median of two, be neglected. On the other hand, the loss ratio is quite high in the RSN scenario as depicted in Figure 9 (2nd). The primary reason lies in the working principle of probabilistic communication. A number of sensors need to send their messages of three of four hops towards the actors. Thus, the probability of reaching the destination equals to p^3 or p^4, respectively, which is quite low for gossiping probabilities p of 0.2, 0.5, and 0.8.

Another reason for the high loss ratios is the unreliable wireless communication. As shown in Figure 9 (1st), the loss ratio is also high for the DYMO scenario. Thus, we finally analyzed the number of collisions at the MAC layer. This measure allows to determine the load distribution over the time and the ability of the network to afford the necessary number data and protocol message transmissions. The results are shown in figure 9 (3rd and 4th). It can be seen

that the use of RSN leads to reduced network congestion (on average we measured 2,300 collisions) compared to the DYMO scenario (42,300 collisions).

5. CONCLUSION

In this paper, we presented and discussed a methodology for network-centric operation in SANETs. Inspired by biological information processing, we developed three easy to handle building blocks: data-centric operation, specific reaction on received data, and simple local behavior. The resulting architecture, which we named Rule-based Sensor Network (RSN), is able to process sensor data and to perform network-centric actuation according to a given set of rules. In particular, this system is able to perform collaborative sensing and processing in SANETs with purely local rule-based programs. The interaction and collaboration between these nodes finally leads to an optimized system behavior in an emergent way. We also developed a simulation model to compare the system performance with classical base-station approaches. In particular we analyzed the performance of the DYMO routing protocol, which can be considered state of the art in ad hoc and sensor network routing, with RSN. It turned out that RSN provides much better scalability and support for real-time operation. This advantage is achieved by reducing the determinism of the system to a certain degree. Depending on the application scenario, this disadvantage might be feasible considering deployments with huge numbers of sensor and actor nodes. Additionally, the possible parameterization of the RSN approach allows to adjust the reliability vs. overhead ratio according to the current needs in the network.

6. REFERENCES

[1] I. F. Akyildiz and I. H. Kasimoglu. Wireless Sensor and Actor Networks: Research Challenges. *Elsevier Ad Hoc Networks*, 2:351–367, October 2004.

[2] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. A Survey on Sensor Networks. *IEEE Communications Magazine*, 40(8):102–116, August 2002.

[3] B. Alberts, D. Bray, J. Lewis, M. Raff, K. Roberts, and J. D. Watson. *Molecular Biology of the Cell.* Garland Publishing, Inc., 3rd edition, 1994.

[4] I. Chakeres and C. Perkins. Dynamic MANET On-Demand (DYMO) Routing. draft-ietf-manet-dymo-07.txt, February 2007.

[5] D. Culler, D. Estrin, and M. B. Srivastava. Overview of Sensor Networks. *Computer*, 37(8):41–49, August 2004.

[6] I. Dietrich and F. Dressler. On the Lifetime of Wireless Sensor Networks. Technical report, University of Erlangen, Dept. of Computer Science 7, December 2006.

[7] I. Dietrich, C. Sommer, and F. Dressler. Simulating DYMO in OMNeT++. Technical report, University of Erlangen, Dept. of Computer Science 7, April 2007.

[8] F. Dressler. Network-centric Actuation Control in Sensor/Actuator Networks based on Bio-inspired Technologies. In *3rd IEEE International Conference on Mobile Ad Hoc and Sensor Systems (IEEE MASS 2006): 2nd International Workshop on Localized Communication and Topology Protocols for Ad hoc Networks (LOCAN 2006)*, pages 680–684, Vancouver, Canada, October 2006.

[9] F. Dressler and I. Carreras. *Advances in Biologically Inspired Information Systems - Models, Methods, and Tools*, volume 69 of *Studies in Computational Intelligence (SCI)*. Springer, 2007.

[10] V. C. Gungor, O. B. Akan, and I. F. Akyildiz. A Real-Time and Reliable Transport Protocol for Wireless Sensor and Actor Networks. *IEEE/ACM Transactions on Networking (ToN)*, 2007.

[11] H. Gupta, S. R. Das, and Q. Gu. Connected Sensor Cover: Self-Organization of Sensor Networks for Efficient Query Execution. In *4th ACM International Symposium on Mobile Ad Hoc Networking and Computing (ACM Mobihoc 2003)*, Annapolis, Maryland, USA, June 2003.

[12] A. Guyton. Blood pressure control - special role of the kidneys and body fluids. *Science*, 252(5014):1813–1816, June 1991.

[13] Z. J. Haas, J. Y. Halpern, and L. Li. Gossip-Based Ad Hoc Routing. In *21st IEEE Conference on Computer Communications (IEEE INFOCOM 2002)*, pages 1707–1716, June 2002.

[14] C. Intanagonwiwat, R. Govindan, and D. Estrin. Directed diffusion: A scalable and robust communication paradigm for sensor networks. In *6th ACM International Conference on Mobile Computing and Networking (ACM MobiCom 2000)*, pages 56–67, Boston, MA, USA, August 2000.

[15] K. H. Low, W. K. Leow, and M. H. Ang. Autonomic Mobile Sensor Network with Self-Coordinated Task Allocation and Execution. *IEEE Transactions on Systems, Man, and Cypernetics–Part C: Applications and Reviews*, 36(3):315–327, March 2005.

[16] T. Melodia, D. Pompili, V. C. Gungor, and I. F. Akyildiz. 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)*, pages 99–110, Urbana-Champaign, Il, USA, May 2005.

[17] T. Pawson. Protein modules and signalling networks. *Nature*, 373(6515):573–80, February 1995.

[18] C. Prehofer and C. Bettstetter. Self-Organization in Communication Networks: Principles and Design Paradigms. *IEEE Communications Magazine*, 43(7):78–85, July 2005.

[19] A. Varga. The OMNeT++ Discrete Event Simulation System. In *European Simulation Multiconference (ESM'2001)*, Prague, Czech Republic, 2001.

[20] G. Weng, U. S. Bhalla, and R. Iyengar. Complexity in Biological Signaling Systems. *Science*, 284(5411):92–96, April 1999.