Is AIS Based Misbehavior Detection Suitable for Wireless Sensor Networks?

Martin Drozda Sven Schaust Helena Szczerbicka
Leibniz University of Hannover, FG Simulation und Modellierung, Dept. of Computer Science
Welfengarten 1, 30167 Hannover, Germany.
Email: {drozda,svs,hsz}@sim.uni-hannover.de

***Abstract*— Sensor networks are a flavor of ad hoc wireless networks with limited computational capabilities. The task to protect such networks against misbehavior is therefore more complicated as any detection mechanism has to be simple and efficient. We employed mechanisms based on Artificial immune systems (AIS) in order to detect misbehavior. We conclude that AIS based misbehavior detection offers a decent detection performance at a very low computational cost. We show that misbehavior detection when applied at both the MAC and network layers may still not be sufficient, instead it will be necessary to extend it to layers with end-to-end connection information; this would also allow for classifying misbehavior by its potential to cause harm. These results have a direct impact on the design of AIS for sensor networks and on engineering of sensor networks.**

I. Introduction and Motivation

A sensor network is a collection of small wireless devices (sensors) that are able to monitor environmental or physical conditions such as humidity, temperature, motion or noise. These sensors are suitably spatially distributed in the monitored area and are able to communicate with each other. Sensor networks lack a centralized authority that would control the flow of individual data packets, instead data connections can be established between any sensor in an ad hoc way; when two sensors are unable to communicate directly, data packets are forwarded by intermediate sensors that serve as routers.

Due to the lack of a centralized authority sensor networks are vulnerable to misbehavior, malfunction or failure. Since sensors are expected to have limited computational power and be battery powered, a system that is going to protect them has to be *lightweight*. Additionally, it has to be adaptive as sensor networks are expected to operate autonomously with sporadic maintenance, it has to be able to undertake action against misbehaving sensors, possibly isolate them, in an extreme case to alarm a human operator. Therefore classical intrusion detection approaches, many of which are based on intrusion signatures that must be frequently updated, or on pre-programmed rules that do not offer any self-configuration, are not suitable for this task.

An example of systems that fulfill the above requirements are Artificial immune systems (AIS). AIS are based on a mechanism that is present in human bodies, namely, on *the Human immune system (HIS)*; see [8], [15], [2] and references therein. AIS are a part of recent promising advances in Intrusion detection systems.

Motivated by results in [8], [15] we have undertaken a detailed performance study of AIS with focus on sensor networks. The general conclusions based on building an AIS for sensor networks can be summarized as follows:
1. Given the ranges of input parameters that we used and considering the computational capabilities of current sensor devices, we conclude that AIS based misbehavior detection offers a decent detection rate at a low computational cost; this makes it an ideal solution for sensor networks.
2. One of the main challenges in designing well performing AIS for sensor networks is the set of "genes". Genes are necessary to measure a network's performance from a node's viewpoint, must be easy to compute and robust against misbehavior. This is similar to observations made in [16].
3. Our results suggest that to increase the detection performance, an AIS has to benefit from information available at all layers of the OSI protocol stack; this includes also detection performance with regards to a simplistic flavor of misbehavior such as packet dropping. This supports ideas shortly discussed in [10] where the authors suggest that information available at the application layer deserves more attention.

II. Artificial Immune Systems

A. Background

The Human immune system is a rather complicated mechanism that is able to protect humans against an amazing set of extraneous attacks. This system is remarkably efficient, most of the time, in discriminating between *self* and *non-self* antigens.[1] A non-self antigen is anything that can initiate an immune response; examples are a virus, bacteria, or splinter. The opposite to non-self antigens are self antigens; self antigens are human organism's own cells.

The important features of HIS have often a dual nature. These dual natures include self vs non-self recognition, innate vs acquired immunity, primary vs secondary response, or general vs specific response. Some immunity mechanisms are antigen specific, systemic (not confined to a local area), or have memory (they are able to launch a stronger response next time a specific antigen is encountered).

B. Learning

The process of T-cells maturation in thymus is used as an inspiration for learning in AIS. The creation of T-cells

[1] Self and non-self in short.

(detectors) in thymus is a result of a pseudo-random process. After a T-cell is created (see Figure 1), it undergoes a censoring process called *negative selection*. During negative selection T-cells that bind self are destroyed. Remaining T-cells are introduced into the body. The recognition of non-self is then done by simply comparing T-cells that survived negative selection with a suspected non-self. This process is depicted in Figure 2. It is possible that the self set is incomplete, while a T-cell matures (tolerization period) in the thymus. This leads to producing T-cells that should have been removed from the thymus and can cause an autoimmune reaction, i.e. it leads to *false positives*.

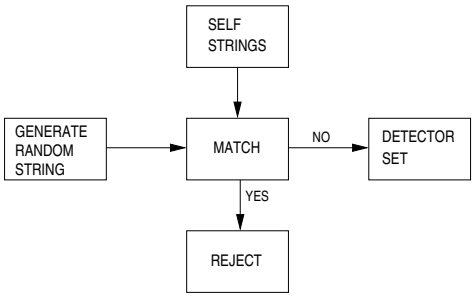


Fig. 1. Detector generation by random-generate-and-test process. Only strings that do not match anything self become detectors.

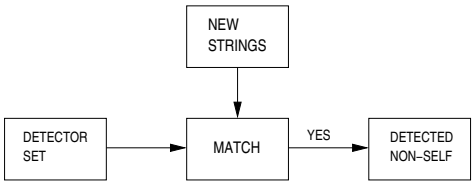


Fig. 2. Recognizing non-self is done by matching detectors with suspected non-self strings.

C. Theoretical Background

The generate-and-test approach for producing T-cells (detectors) described above is analyzed in [6]. They assume that both self and non-self sets, as well as detectors can be modeled as bit-strings of length l. Let the size of the self set be N_S, the probability that a randomly chosen detector and a string from the self set match be P_m and the probability that a string from the non-self set is not matched by any detector be P_f. Then the time and space complexity of this algorithm for a fixed matching probability P_m is $O(\frac{-ln(P_f)}{P_m(1-P_m)^{N_S}}N_S)$ and $O(lN_S)$, respectively. This algorithm requires that the number of required candidate detectors is exponential to N_S. The advantage of this algorithm is its simplicity and good experimental results in cases when the number of detectors to be produced is fixed and small [15]. A review of other approaches to detector computation can be found in [2].

III. Sensor Networks

A sensor network can be defined in graph theoretic framework as follows: a sensor network is a net $N = (n(t), e(t))$ where $n(t), e(t)$ are the set of nodes and edges at time t, respectively. Nodes correspond to automated sensors (or mobile users) that wish to communicate with each other. An edge between two nodes A and B is said to exist when A is within the radio transmission range of B and vice versa. The imposed symmetry of edges is a usual assumption of many mainstream protocols. The change in the cardinality of sets $n(t), e(t)$ can be caused by switching on/off one of the sensors, failure, malfunction, removal, signal propagation, link reliability and other factors.

Data exchange in a point-to-point (uni-cast) scenario usually proceeds as follows: a user initiated data exchange leads to a route query at the network layer of the OSI stack. A routing protocol at that layer attempts to find a route to the data exchange destination. This request may result in a path of non-unit length. This means that a data packet in order to reach the destination has to rely on successive forwarding by intermediate nodes on the path. An example of an on-demand routing protocol designed specifically for ad hoc networks is DSR [9]. Route search in this protocol is started only when a route to a destination is needed. This is done by flooding the network by RREQ[2] control packets. The destination node or an intermediate node that knows a route to the destination will reply with a RREP control packet. This RREP follows the route back to the source node and updates routing tables at each node that it traverses. A RERR packet is sent to the connection originator when a node finds out that the next node on the forwarding path is not replaying. We refer the reader to [11] for more information on sensor networks.

Movement of nodes can be modeled by means of a movement model. A well-known mobility model is the *Random waypoint model*. In this model, nodes move from the current position to a new randomly generated position at a predetermined speed. After reaching the new destination a new random position is computed. Nodes pause at the current position for a time period t before moving to the new random position.

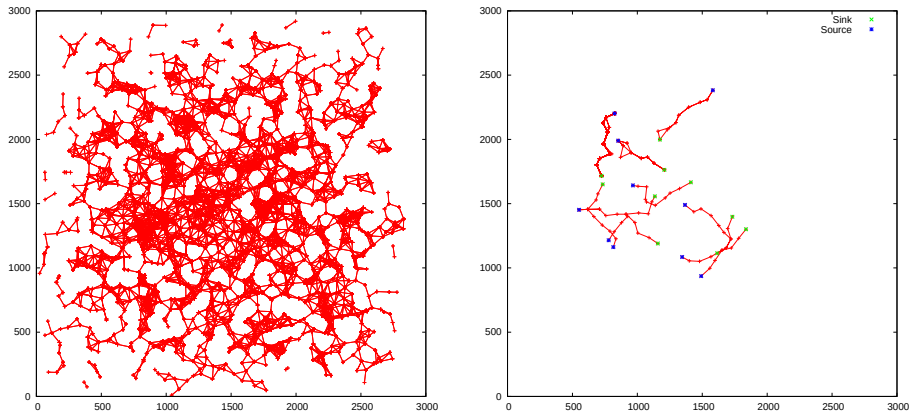
IV. Experimental Setup

The purpose of our experiments is to show that AIS are a viable approach for detecting misbehavior in sensor networks. In a companion paper [7] we have reviewed different types of misbehavior at the MAC, network and transport level of the OSI protocol stack. We note that solutions to some of these attacks have been already proposed; these are however specific to a given attack.

We represent self, non-self and detector strings as bit-strings. The matching rule employed is the *r-contiguous bits matching rule*. Two bit-strings of equal length match under the r-contiguous matching rule if there exists a substring of length r at position p in each of them and these substrings are identical. Detectors are produced by the process shown in Figure 1, i.e. by means of negative selection when detectors are created randomly and tested against a set of self strings.

Definitions of input and output parameters: The input parameters for our experiments were: r parameter for the r-

[2]RREQ = Route Request, RREP = Route Reply, RERR = Route Error.



(a) (b)

Fig. 3. (a) Topology of our 1,718-node network with 100m radio radius. (b) Measured forwarding path of the 10 connections for a single simulation run without misbehavior; connections shown with all alternative forwarding routes, if they exist.

1) Negative selection algorithm: random generate and test. Implemented in C++, compiled with GNU g++ v4.0 with -O3 option.
2) **Input parameters:** 1. r-contiguous matching rule with $r = \{7, 10, 13, 16, 19, 22\}$. 2. Encoding: 5 genes each 10 bits long = 50 bits. 3. Number of detectors $\{500, 1000, 2000, 4000\}$. 4. Misbehavior level $\{10, 30, 50\%\}$ 5. Window size 500 seconds; 28 complete windows over 4-hour simulation time.
3) **Performance measures:** real time to compute detectors, number of iterations to compute detectors, detection rate, rate of non-valid detectors, data traffic rate at nodes; their arithmetic averages and 95% confidence intervals.
4) **Network topology:** Snapshot of movement modeled by random waypoint mobility model i.e. it is a static network. There were 1,718 nodes. The area was a square of 2,900m×2,950m. The transmission range of transceivers was 100 meters.
5) **Number of connections:** 10 CBR (constant bit rate) connections. **MAC protocol**: IEEE 802.11b DCF. **Routing protocol**: DSR. Other parameters: (i) Propagation path-loss model: two ray (ii) Channel frequency: 2.4 GHz (iii) Topography: Line-of-sight (iv) Radio type: Accnoise (v) Network protocol: IPv4 (vi) Connection type: UDP.
6) **Injection rate:** 1 packet/second. 14,400 packets per connection were injected. Packet size was 512 bytes.
7) The number of independent simulation runs for each combination of input parameters was 20. The simulation time was 4 hours.
8) **Simulator used:** GlomoSim 2.03; hardware used: 30× Linux (SuSE 10.0) PC with 2GB RAM and Pentium 4 3GHz microprocessor.

Fig. 4. Parameters used in the experiment.

contiguous matching rule, the (desired) number of detectors, misbehavior level and traffic rate at nodes. Misbehavior was modeled as random packet dropping at selected nodes.

The performance (output) measures were arithmetic average and 95% confidence intervals $ci_{95\%}$ of detection rate, number of false positives, real time to compute detectors, number of iterations to compute detectors (number of random tries) and number of non-valid detectors. The detection rate dr is defined as $\frac{dns}{ns}$, where dns is the number of detected non-self strings and ns is the total number of non-self strings. A false positive in our definition is a string that is not self but can still be a result of anomaly that is identical with the effects of a misbehavior. A non-valid detector is a candidate detector that matches a self string and must therefore be removed.

Scenario description: The purpose of this experiment was to capture "self" and "non-self" packet traffic in a synthetic static sensor network and test whether using an AIS we are able to recognize non-self, i.e. misbehavior. We only considered packet traffic at the MAC and network layer. The set of genes that represent certain chosen properties of packet traffic in wireless networks was chosen so that a thorough functionality test of our AIS is possible. The set is not complete, i.e. it does not allow is to recognize a large range of misbehavior activities, in contrary, the idea was to choose a set of a modest size. In the future we plan to undertake a more complex simulation experiments with packet traffic information ranging all over the OSI protocol stack.

The topology of this network was determined by making a snapshot of 1,718 mobile nodes (each with 100m radio radius) moving in a square area of 2,900m×2,950m as prescribed by the random waypoint mobility model; see Figure 3(a). The motivation in using this movement model and then creating a snapshot are the results in our previous paper [5] that deals with structural robustness of sensor network. We chose source and destination pairs for each connection so that several alternative independent routes exist; the idea was to benefit from route repair and route acquisition mechanisms of the DSR routing protocol, so that the added value of AIS based misbehavior detection is obvious.

We have used 10 CBR (Constant bit rate) connections. The connections were chosen so that their length is ∼7 hops and so that these connections share some common intermediate nodes; see Figure 3(b). For each packet received or sent by a node we have captured the following information: IP

header type (UDP, 802.11 or DSR in this case), MAC frame type (RTS, CTS, DATA, ACK in the case of 802.11), current simulation clock, node address, next hop destination address, data packet source and destination address and packet size. Let us assume that the routing protocol finds for a connection the path $s_s, s_1, ..., s_i, s_{i+1}, s_{i+2}, ..., s_d$ from the source node s_s to the destination node s_d, where $s_s \neq s_d$. We have used the following *genes* to capture certain aspects of MAC and routing layer traffic information:

MAC Layer:

#1 Ratio of complete MAC layer handshakes between nodes s_i and s_{i+1} and RTS packets sent by s_i to s_{i+1}. If there is no traffic between two nodes this ratio is set to ∞ (a large number). This ratio is averaged over a time period. A complete handshake is defined as a completed sequence of RTS, CTS, DATA, ACK packets between s_i and s_{i+1}.

#2 Ratio of data packets sent from s_i to s_{i+1} and then subsequently forwarded to s_{i+2}. If there is no traffic between two nodes this ratio is set to ∞ (a large number). This ratio is computed by s_i in promiscuous mode. This ratio is also averaged over a time period. This gene was adapted from the watchdog idea in [13].

#3 Time delay that a data packet spends at s_{i+1} before being forwarded to s_{i+2}. The time delay is observed by s_i in promiscuous mode. If there is no traffic between two nodes the time delay is set to zero. This measure is averaged over a time period. This gene is a quantitative extension of the previous gene.

Routing Layer:

#4 The same ratio as in #2 but computed separately for RERR routing packets.

#5 The same delay as in #3 but computed separately for RERR routing packets.

The above mentioned time period is 500 seconds.

Encoding of self and non-self antigens:[3] Each gene value was transformed in a 10-bit signature where each bit defines an interval[4] of a gene specific value range. We created self and non-self antigen strings by concatenation of the defined genes. Each self and non-self antigen has therefore a size of 50 bits. The interval representation was chosen in order to avoid carry-bits that make the binary representation less compact.

Constructing the self and non-self sets: We have randomly chosen 28 non-overlapping 500-second windows in our 4-hour simulation. In each 500-second window self and non-self antigens are computed for each node. This was repeated 20 times for independent Glomosim runs.

Misbehavior modeling: Misbehavior is modeled as random data packet dropping; we have randomly chosen 236 nodes and these were forced to drop $\{10, 30, 50\%\}$ of data packets. However, there were only 3-10 nodes with misbehavior and with a statistically significant number of packets for forwarding in each simulation run.

Simulation phases: The experiment was done in four phases.
1. 20 independent Glomosim runs were done for one of $\{10, 30, 50\%\}$ misbehavior levels and "normal" traffic with no misbehavior.
2. Self and non-self antigen computation.
3. The 20 "normal" traffic runs were used to compute detectors. Given the 28 windows and 20 runs, the sample size was $20 \times 28 = 560$, i.e. detectors at each node were discriminated against 560 self antigens.
4. Using the runs with $\{10, 30, 50\%\}$ misbehavior levels, the process shown in Figure 2 was used for detection.
Experiment was then repeated with different r, desired number of detectors and misbehavior level.

The parameters for this experiment are summarized in Figure 4. The injection rate and packet sizes were chosen in order to comply with usual data rates of sensors (e.g. 38.4kbps for Mica2; see [1]). One can consider packet traffic in sensor networks be more bursty and less frequent than in our model but there is not much experience with these types of networks and their use can vary in the future.

V. Results Evaluation

When considering results presented in this section one should remember that the computational throughput of sensors lies at max. 1% of the used PCs.[5] On the other hand, it is reasonable to expect that computation of detectors[6] will be very infrequent, once per several weeks or months. An initial set of detectors can be provided at the first deployment. It is also reasonable to expect that several sensors will be able to detect a single misbehaving sensor.

The results connected to computation of detectors are shown in Figure 5. In our experiments we have only considered the desired number of detectors to be max. 4,000; over this threshold the computational requirements might be too high for current sensor devices. Also, each time the r parameter is incremented by 1, the number of detectors should double in order to make these two cases comparable.

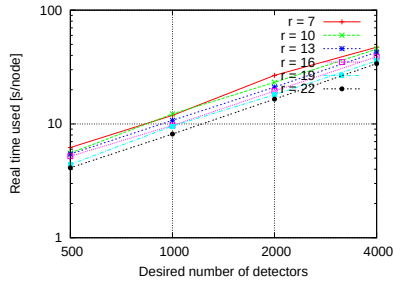
Figure 5(a) shows the real time needed to compute the desired set of detectors. We can see the real time necessary increases proportionally with the desired number of detectors; this complies with the theoretical results presented in [6]. Figure 5(b) shows the percentage of non-valid detectors, i.e. detectors that were found to match a self string (see Figure 1). This result points to where the optimal operation point of an AIS might lie with respect to the choice of r parameter and the choice of a fixed number of detectors to compute. We remind the reader that the larger is the r parameter the smaller is the probability that a detector will match a self string. Therefore overhead connected to choosing the r parameter prohibitively

[3]The non-self antigens are a mixture of self antigens, non-self antigens and antigens that is not possible to classify due to their similarity to non-self antigens.

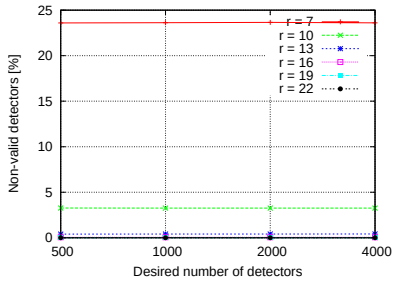
[4]The interval encoding of genes is adapted from [15]. This way only one of the 10 bits is set to 1, i.e. there are only 10 possible value levels that it is possible to encode in this case.

[5]For example a Mica2 sensor is equipped with an Atmel ATmega 128 8-bit processor that has peak throughput 16 MIPS, program memory 128kB, storage memory 512kB; the outdoor radio range is app. 150 meters [1].

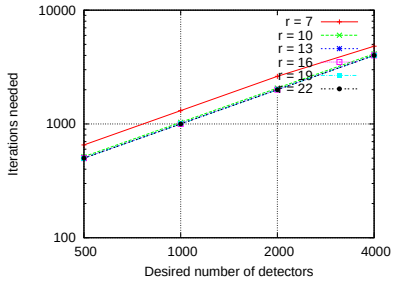
[6]Issues connected with availability of misbehavior-free periods for detector computation are beyond the scope of this paper.



(a) Real time to compute the desired number of detectors at a node; $ci_{95\%} < 1\%$.

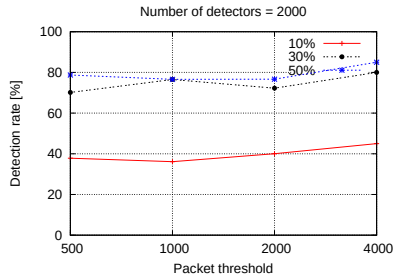


(b) Rate of non-valid detectors; for $r \leq 13$ is $ci_{95\%} < 1\%$, for $r \geq 16$ is the sample size not significant.

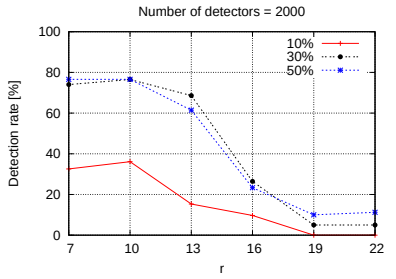


(c) Number of iterations needed in order to compute the desired number of detectors; for $r \geq 10$ is $ci_{95\%} < 1\%$, for $r = 7$ is $ci_{95\%} < 2\%$.

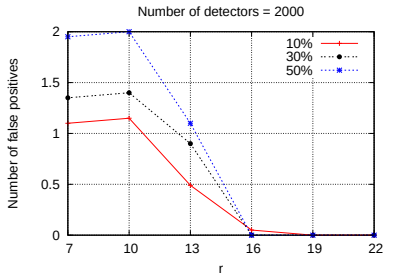
Fig. 5. Performance of detectors computation.



(a) Detection rate vs packet threshold; conf. interval ranges: for mis. level 10% is $ci_{95\%}$ = 3.8-19.8%; for 30% is $ci_{95\%}$ = 11.9-15.9%; for 50% is $ci_{95\%}$ = 11.0-14.2%.



(b) Detection rate vs r; $ci_{95\%}$ range similar to (a).



(c) Number of false positives; for $r \leq 10$ is $ci_{95\%}$ = 0.47-0.68, for $r \geq 13$ is the sample size not significant.

Fig. 6. Performance of misbehavior detection. Misbehavior level = {10, 30, 50}%. In (a) $r = 10$, in (b) and (c) the packet threshold was 1000.

small should be consider when designing an AIS. Figure 5(c) shows the total number of generate-and-test tries needed for computation of detector set of a fixed size.

The task of misbehavior detection (see Figure 2) requires comparison of the computed detectors with a non-self string. In our case there is one non-self string computed per a 500-second window (in general, the window size can be changed with respect to the traffic properties). The time complexity of misbehavior detection is proportional to the number of detectors.

When evaluating the detection rate it is important that the number of packets that a node forwards is over a certain threshold. If a node lacks packets to forward in the misbehavior-free learning phase, his ability to learn is limited. If it lacks packets to forward and at the same time wishes to execute misbehavior, the impact of misbehavior is limited as there are no packets to drop. Therefore we only considered nodes with some minimum forwarding activity. We define a node to be detected as misbehaving if it gets flagged in at least 14 out of 28 possible windows. This definition is equivalent (under reasonable assumptions) to saying that the time to detection is double the size of the window, i.e. 1000 seconds in this case. In Figure 6(a) we show dependence of detection ratio on this packet threshold. Packet threshold of e.g. 500 means that a node had at least 500 packets to forward in both the learning and misbehavior phases; this number is measured over the whole 4-hour simulation period. We conclude that except for some extremely low threshold values (not shown) the detection rate stays constant. This figure also shows that when misbehavior level was set very low, i.e. 10% the AIS struggled to detect misbehaving nodes. At the 30 and 50% misbehaving levels the detection rate stays solid at about 70-85%. The range of the 95% confidence interval of detection rate is 3.8-19.8%. This is similar to results in [16]. This points out that misbehavior detection at the MAC and network layers may not be sufficient, instead AIS based misbehavior detection should be extended to OSI protocol layers with end-to-end connection information. This would also allow for classifying misbehavior by its potential to cause harm as it is suggested in [3]. It also implicates that watchdog based genes should not be used in isolation, and in general, that the choice of genes has to be very careful.

Figure 6(b) shows the impact of r on detection rate. When $r = \{7, 10\}$ the AIS performs well, for $r > 10$ the detection rate decreases. This is caused by the inadequate numbers of detectors used; in general the number of detectors should double when r is increased by 1.

Figure 6(c) shows the number of false positives. We remind

that in our definition false positives are both nodes that do not drop any packets and nodes that drop packets due to other reasons than misbehavior.

In a separate experiment we studied whether the 4-hour (560 samples) simulation time was enough to capture the diversity of the self behavior. This was done by trying to detect misbehavior in 20 independent misbehavior-free Glomosim runs (different from those used to compute detectors). We report that we did not observe a single case of an autoimmune reaction.

VI. Related Work

In [15], [16] the authors introduced an AIS based misbehavior detection system for ad hoc wireless networks. They used Glomosim for simulating data traffic, their setup was an area of 800×600m with 40 mobile nodes (speed 1 m/s) of which 5-20 are misbehaving; the routing protocol was DSR. Four genes were used to capture local behavior at the network layer. The misbehavior implemented is a subset of misbehavior introduced in this paper; their observed detection rate is about 55%. Additionally, a co-stimulation in the form of a danger signal was used in order to inform nodes on a forwarding path about misbehavior, thus propagating information about misbehaving nodes around the network.

In [8] the authors describe an AIS able to detect anomalies at the transport layer of the OSI protocol stack; only wired TCP networks are considered. Self is defined as normal pairwise TCP connections. Each detector is represented as a 49-bit string. The pattern matching is based on r-contiguous bits with a fixed $r = 12$.

Ref. [12] discusses a network intrusion system that aims at detecting misbehavior by capturing TCP packet headers. They report that their AIS is unsuitable for detecting anomalies in communication networks. This result is questioned in [4] where it is stated that is due to the choice of problem representation and due to the choice of matching threshold r for r-contiguous bit matching.

The main discerning factor between our work and works shortly discussed above is that our genes benefit from information at both the MAC and network layers, we carefully considered hardware parameters of current sensor devices, the set of input parameters was designed in order to target specifically sensor networks and our simulation setup reflects structural qualities of sensor networks with regards to existence of multiple independent routing paths. In comparison to [15], [16] we show that in case of static sensor networks it is reasonable to expect the detection rate to be above 80%.

VII. Conclusions and Future Work

Even though we answered some basic question on the suitability and feasibility of AIS for detecting misbehavior in sensor networks a few questions remain open.

The key question in the design of AIS is the quantity, quality and ordering of genes that are used for measuring behavior at nodes. To answer this question a detailed formal analysis of communications protocols will be needed. The set of genes should be as "complete" as possible with respect to any possible misbehavior. The choice of genes should impose a high degree of sensor network's survivability defined as *the capability of a system to fulfill its mission in a timely manner, even in the presence of attacks, failures or accidents* [14]. It is therefore of paramount importance that the sensor network's mission is clearly defined and achievable under normal operating conditions.

Our intermediate research direction will be to undertake similar tests as described in this document on Mica2 sensors [1] and verify viability of AIS based misbehavior detection in real world settings.

Acknowledgments

This work was supported by the German Research Foundation (DFG) under the grant no. SZ 51/24-1 (Survivable Ad Hoc Networks – SANE).

References

[1] Crossbow Technology Inc. `www.xbow.com`

[2] Uwe Aickelin, Julie Greensmith, Jamie Twycross. Immune System Approaches to Intrusion Detection - A Review. *Proc. the 3rd International Conference on Artificial Immune Systems (ICARIS 2004)*, 2004.

[3] U. Aickelin, P. Bentley, S. Cayzer, J. Kim, and J. McLeod. Danger theory: The link between ais and ids. *Proc. International Conference on Artificial Immune Systems (ICARIS'03)*, 2003.

[4] J. Balthrop, S. Forrest, M. Glickman. Revisiting lisys: Parameters and normal behavior. *Proc. Congress on Evolutionary Computing (CEC02)*, 2002.

[5] C. L. Barrett, M. Drozda, D. C. Engelhart, V. S. Anil Kumar, M. V. Marathe, M. M. Morin, S. S. Ravi, J. P. Smith. Understanding Protocol Performance and Robustness of Ad Hoc Networks Through Structural Analysis. *Proc. IEEE International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob 2005)*, 2005.

[6] D'haeseleer, P., Forrest, S., and Helman, P. An immunological approach to change detection: Algorithms, analysis and implications. *Proc. IEEE Symposium on Research in Security and Privacy*, 1996.

[7] M. Drozda, H. Szczerbicka. Artificial Immune Systems: Survey and Applications in Ad Hoc Wireless Networks. *Proc. 2006 International Symposium on Performance Evaluation of Computer and Telecommunication Systems (SPECTS'06)*, 2006.

[8] S. Hofmeyr, S. Forrest. Immunity by Design: An Artificial Immune System. *Proc. Genetic and Evolutionary Computation Conference (GECCO-1999)*, 1999.

[9] D. Johnson, D. Maltz. Dynamic Source Routing in Ad Hoc Wireless Networks. *Mobile Computing*, Tomasz Imielinski and Hank Korth, Eds. Chapter 5, pp. 153-181, Kluwer Academic Publishers, 1996.

[10] Zhang, Y. and Lee, W. and Huang, Y.A. Intrusion Detection Techniques for Mobile Wireless Networks. *Wireless Networks (WINET)*, vol. 9, no. 5, pp. 545–556, 2003.

[11] Karl, H., Willig, A. *Protocols and Architectures for Wireless Sensor Networks.* John Wiley & Sons, 2005.

[12] Kim, J., Bentley, P. J. Evaluating Negative Selection in an Artificial Immune System for Network Intrusion Detection, *Proc. Genetic and Evolutionary Computation Conference 2001 (GECCO-2001)*, 2001.

[13] Sergio Marti, T. J. Giuli, Kevin Lai and Mary Baker. Mitigating routing misbehavior in mobile ad hoc networks. *Proc. the 6th annual international conference on Mobile Computing and Networking*, 2000.

[14] James P. G. Sterbenz, Rajesh Krishnan, Regina Rosales Hain, Alden W. Jackson, David Levin, Ram Ramanathan, John Zao. Survivable mobile wireless networks: issues, challenges, and research directions. *Proc. ACM workshop on Wireless security*, 2002.

[15] Slaviša Sarafijanović, Jean-Yves Le Boudec. An Artificial Immune System for Misbehavior Detection in Mobile Ad-Hoc Networks with Virtual Thymus, Clustering, Danger Signal and Memory Detectors. *Proc. the 3rd International Conference on Artificial Immune Systems (ICARIS 2004)*, 2004.

[16] Jean-Yves Le Boudec, Slaviša Sarafijanović. An Artificial Immune System Approach to Misbehavior Detection in Mobile Ad-Hoc Networks. *Proc. Bio-ADIT'04*, 2004.