AIS for Misbehavior Detection in Wireless Sensor Networks: Performance and Design Principles

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*— A sensor network is a collection of wireless devices that are able to monitor physical or environmental conditions. These devices are expected to operate autonomously, be battery powered and have very limited computational capabilities. This makes the task of protecting a sensor network against misbehavior or possible malfunction a challenging problem. In this document we discuss performance of Artificial immune systems (AIS) when used as the mechanism for detecting misbehavior. We concentrate on performance of respective genes; genes are necessary to measure a network's performance from a sensor's viewpoint. We conclude that the choice of genes has a profound influence on the performance of the AIS. We identified a specific MAC layer based gene that showed to be especially useful for detection. We also discuss implementation details of AIS when used with sensor networks.**

I. Introduction and Motivation

Sensor networks can be described as a collection of wireless devices with limited computational abilities which are, due to their ad-hoc communication manner, vulnerable to misbehavior and malfunction. It is therefore necessary to support them with a simple, *computationally friendly* protection system.

Artificial immune systems (AIS) are based on principles adapted from the Human immune system (HIS) [13]; the basic ability of HIS is an efficient detection of potentially harmful foreign agents (viruses, bacteria, etc.). The goal of AIS, in our setting, is identification of nodes with behavior that could possibly negatively impact the stated mission of the sensor network. The mission of a sensor network can be monitoring and processing physical or environmental conditions such as humidity, temperature, motion or noise. In order to fulfill this task, individual sensors are suitably distributed in the monitored area. The communication between sensors is done without any additional infrastructure, instead data packets get routed over intermediate sensors until the destination (point of data collection) is reached.

Misbehavior in wireless sensor networks can take upon different forms: packet dropping, modification of data structures important for routing, modification of packets, skewing of the network's topology or creating ficticious nodes (see [11] for a more complete list). The reason for sensors (possibly fully controlled by an attacker) to execute any form of misbehavior can range from desire to save battery power to making a given wireless sensor network non-functional. Malfunction can also be considered a type of unwanted behavior. Any system aimed at protecting a wireless sensor network should work autonomously with none or a sporadic human intervention.

Motivated by results in [13], [21] we have undertaken a detailed performance study of AIS with focus on sensor networks. In an earlier paper [9], we concluded that AIS based misbehavior detection offers a decent detection rate at a low computational cost which could make it a good solution for sensor networks. The general conclusions that can be drawn from the study presented in this document are:

1) Genes from different layers of the OSI stack need to be combined in order to achieve a good performance of AIS. Genes are necessary to measure a network's performance from a node's viewpoint, they must be easy to compute. Our results show that somewhat surprisingly a gene that was based purely on the MAC layer significantly contributed to the overall detection performance. This gene poses less limitations when a MAC protocol with a sleep-wake-up schedule such as the S-MAC [23] is used.

2) We only used a single instance of learning and detection mechanism per node. This is different from approach used in [13], [21], where one instance was used for each of m possible neighbors. Our performance results combined with results in [9] show that the approach in [13], [21] may not be feasible for sensor networks. It may allow for an easy Sybil attack and, in general, $m = n-1$ instances might be necessary, where n is the total number of sensors in the network. Instead, we suggest that flagging a node as misbehaving should, if possible, be based on detection at several nodes (see [24] for architecture oriented thoughts on global detection).

3) Only less than 5% detectors were used in detecting misbehavior. This suggests that many of the detectors do not comply with constraints imposed by the communications protocols; this is an important fact when designing AIS for sensor networks because the memory capacity at sensors is expected to be very limited.

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.

[1] Self and non-self in short.

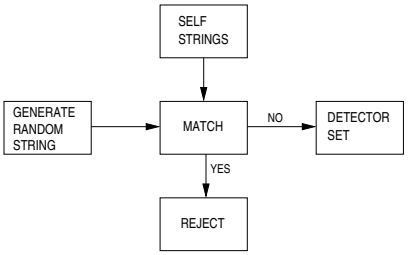


Fig. 1. Detector generation by random-generate-and-test process. Only strings that do not match anything self become detectors.

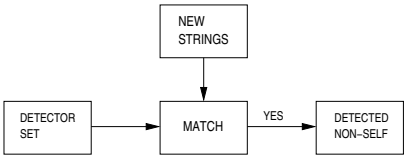


Fig. 2. Recognizing non-self is done by matching detectors with suspected non-self strings.

B. Learning

The process of T-cells maturation in thymus is used as an inspiration for learning in AIS. The maturation of T-cells (detectors) in thymus is a result of a pseudo-random process. After a T-cell is created (see Fig. 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 Fig. 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*.

The random-generate-and-test approach for producing T-cells (detectors) described above is analyzed in [8]. In general, the number of candidate detectors to the self set size needs be exponential (if a matching rule with fixed matching probability is used). Another problem is a consistent underfitting of the non-self set; there exist "holes" in the non-self set that are undetectable. In theory, for some matching rules, the number of holes can be very unfavorable [22]. In practical terms, the effect of holes depends on the characteristics of the non-self set, representation and matching rule [12]. 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 [21]. 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 sensors 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 [15]. Route search in this protocol is started only when a route to a destination is needed. This is done by flooding the network with 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.

At the MAC layer, the medium reservation is often contention based. In order to transmit a data packet, the IEEE 802.11 MAC protocol uses carrier sensing with an RTS-CTS-DATA-ACK handshake.[3] Should the medium not be available or the handshake fails, an exponential back-off algorithm is used. This is combined with a mechanism that makes it easier for neighboring nodes to estimate transmission durations. This is done by exchange of duration values and their subsequent storing in a data structure known as Network allocation vector (NAV). With the goal to save battery power, researchers suggested, a sleep-wake-up schedule for nodes would be appropriate. This means that nodes do not listen continuously to the medium, but switch themselves off and wake up again after a predetermined period of time. Such a sleep and wake-up schedule is similarly to duration values exchanged among nodes. An example of a MAC protocol, designed specifically for sensor networks, that uses such a schedule is the S-MAC [23]. A sleep and wake-up schedule can severely limit operation of a node in *promiscuous mode*. In promiscuous mode, a node listens to the on-going traffic in the neighborhood and collects information from the overheard packets. This technique is used e.g. in DSR for improved propagation of routing information. For more information on sensor networks, we refer the reader to [16].

IV. AIS for Sensor Networks: Design Principles

In our approach, each node produces and maintains its own set of detectors. This means that we applied a direct one-to-one mapping between a human body with a thymus and a node. 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

[2]RREQ = Route Request, RREP = Route Reply, RERR = Route Error.
[3]RTS = Ready to send, CTS = Clear to send, ACK = Acknowledgment.

are identical. Detectors are produced by the process shown in Fig. 1, i.e. by means of negative selection when detectors are created randomly and tested against a set of self strings.

Each antigen consists of several genes. *Genes* are performance measures that a node can acquire locally without the help from another node. In practical terms this means that an antigen consists of x genes; each of them encodes a performance measure, averaged in our case over a time window. An antigen is then created by concatenating the x genes.

In choosing the correct genes for an AIS to be applied to sensor networks, the choice can be limited due to the simplified OSI protocol stack of sensors. For example, Mica2 sensors [1] using the TinyOS operating system do not guarantee any end-to-end connection reliability (transport layer), leaving only data traffic at the lower layers for consideration.

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 (we averaged over a time period (window size) of 500 seconds):

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 [20].

#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 Gene #1 is MAC protocol oriented, the remaining ones are watchdog oriented. The first gene can also be characterized as MAC layer quality oriented – *it indirectly measures the medium contention level*, whereas the remaining ones are misbehavior oriented. As we will show later, in the particular type of misbehavior (probabilistic packet dropping) that we applied, the first two genes come out as "the strongest". The disadvantage of the watchdog based genes is that due to limited battery power, nodes could operate using a sleep-wake-up schedule similar to the one used in the S-MAC. This would mean that the node s_i has to stay awake until the node s_{i+1} correctly transmits to s_{i+2}. The consequence would be a longer wake-up time and possible restrictions in publishing sleep-wake-up schedules.

In [19] the authors applied a different set of genes. The observed set of events was the following: A = RREQ sent, B = RREP sent, C = RERR sent, D = DATA sent and IP source address is not of the monitored (neighboring) node, E = RREQ received, F = RREP received, G = RERR received, H = DATA received and the IP destination address is not of the monitored node (the DSR routing protocol was used). The events D and H take into consideration that the source and destination nodes of a connection might appear as misbehaving as they seem to "deliberately" create and delete data packets. Then the set of their four genes is as follows:

#1 Number of E over a time period.
#2 Number of (E*(A or B)) over a time period.
#3 Number of H over a time period.
#4 Number of (H*D) over a time period.

The time period (window size) in their case was 10s; * is the Kleene star operator (zero or more occurrences of any event(s) are possible). Similar to our watchdog genes, these genes impose additional requirements on MAC protocols such as the S-MAC. Their dependence on the operation in promiscuous mode is, however, more pronounced as a node has to continuously observe packet events at all monitored nodes.

The research in the area of what and to what extent can be or should be locally measured at a node, is independent of the learning mechanism used (negative selection in both cases). Performance of an AIS can partly depend on the ordering and the number of used genes. As longer antigens (consisting of more genes) indirectly imply more candidate detectors, the number of genes should be carefully considered. Given x genes, it is possible to order them in $x!$ different ways. In our experience, the rules for ordering genes and the number of genes can be summed up as follows:

1) Keep the number of genes small. In our experiments, we show that with respect to the learning mechanism used and the expected deployment (sensor networks), 2-3 genes are enough for detecting a basic type of misbehavior.

2) Order genes either randomly or use a predetermined order. Both of these possibilities are computationally friendly (with respect to sensor networks). Defining a utility relation between genes, and ordering genes with respect to it can, in general, lead to problems that are considered intractable.

3) Produce several types of antigen. These antigens can share genes but the number of genes per type of antigen should be limited. The advantage is that the number of candidate detectors necessary for negative selection stays small. Multiple gene membership with randomized ordering could additionally lead to an improved robustness of the underlying AIS, especially, when one misbehaving node can be detected by several other nodes.

Our experiments show that genes cannot be considered in isolation, i.e. when a detector matched an antigen under the r-contiguous matching rule, usually this match spanned over several genes.

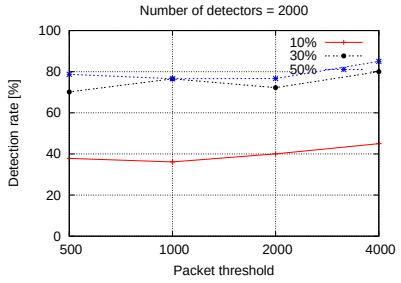
A. Learning and Detection

Learning and detection is done by applying the mechanisms shown in Figs. 1 and 2. The detection itself is very straightfor-

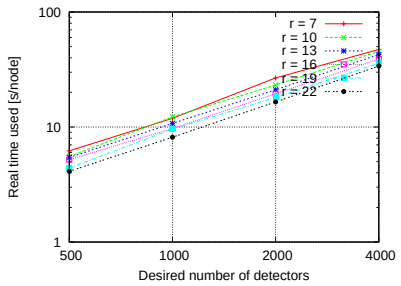
ward. In the learning phase, a misbehavior-free period (see [3] on possibilities for circumventing this problem) is necessary so that nodes get a chance to learn what is the normal behavior. When implementing the learning phase, the designer gets to choose from two possibilities:

1) Learning and detection at a node get implemented for each neighboring node separately. This means that different antigens have to get computed for each neighboring node, detector computation is different for each neighboring node and then, subsequently, detection is different for each neighboring node. The advantage of this approach is that the node is able to directly determine which neighboring node misbehaves; the disadvantage is that m instances (m is the number of neighbors or node degree) of the negative selection mechanism have to get executed; this can be computationally prohibitive for sensor networks as m can, in general, be equal to the total number of sensor. This allows for an easy Sybil attack [11] in which a neighbor would create several identities; the node would then be unable to recognize that these identities belong to the same neighbor. This approach was used in [21], [19].

2) Learning and detection at a node get implemented in a single instance for all neighboring nodes. This means a node is able to recognize anomaly (misbehavior) but it may be unable to determine which one from the m neighboring nodes misbehaves. This implies that nodes would have to cooperate when detecting a misbehaving node, exchange anomaly information and be able to draw a conclusion from the obtained information. An argument for this approach is that in order to detect nodes that misbehave in collusion, it might be necessary to rely to some extent on information exchange among nodes, thus making this a natural solution to the problem. We have used this approach; a post-processing phase, that used available routing information, was necessary to determine whether a node was correctly flagged as misbehaving or not.



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

Fig. 3. Performance of misbehavior detection.

We find the second approach to be more suited for wireless sensor networks. In Fig. 3(b), we illustrate the computational requirements of the negative selection process, if executed in a single instance. We believe that a complete detector set computation will be very infrequent.

Which r is the correct one?

An interesting technical problem is to tune the r parameter for the r-contiguous matching rule so that the underlying AIS offers good detection and false positives rates. One possibility is a lengthy simulation study such as the one described in [9]. Through multiparameter simulation we were to able to show that $r = 10$ offers the best performance for our setup. In [10] we experimented with the idea of "growing" and "shrinking" detectors; this idea was motivated by [14]. The initial r_0 for a growing detectors can be chosen as $r_0 = \lceil l/2 \rceil$, where l is the detector length. The goal is to find the smallest r such that a candidate detector does not match any self antigen. This means that a candidate detector gets randomly created and then tested against the self set. Initially, a larger (more specific) r is chosen; the smallest r that fulfills the above condition can be found through binary search. For shrinking detectors, the approach is reciprocal. Our initial goal was to show that such growing or shrinking detectors would offer a better detection or false positives rate. Short of proving this in a statistically significant manner, we observed that the growing detectors can be used for self tuning the r parameter. The average r value was close to the r that we determined through simulation (the setup in this case was different from the one described here).

B. Further Optimizations

Our experiments show that only a small number of detectors get ever used (less than 5%). The reason for that is that they get produced in a random way, not considering structure of the protocols. For example, a detector that is able to detect whether i) data packets got correctly transmitted and ii) 100% of all MAC layers handshakes were incomplete is superfluous as this case should never happen. In [6], the authors conclude: "... *uniform coverage of non-self space is not only unnecessary, it is impractical; non-self space is too big*". Application driven knowledge can be used to set up a rule based system that would exclude infeasible detectors; see [7] for a rule based system aimed at improved coverage of the non-self set. In [13], it is suggested that unused detectors should get deleted and the lifetime of useful detectors should be extended.

C. Misbehavior

In a companion paper [11] we have reviewed different types of misbehavior at the MAC, network and transport layers of the OSI protocol stack. We note that solutions to many of these attacks have been already proposed; these are however specific to a given attack. The appeal of AIS based misbehavior detection rests on its simplicity and applicability in an environment that is extremely computationally and bandwidth limited. Misbehavior in sensor networks does not have to be executed by sensors themselves; one or several

computationally more powerful platforms (laptops) can be used for the attack. On the other hand, a protection using such more advanced computational platforms is, due to e.g. the need to supply them continuously with electric power, harder to imagine. It would also create a point of special interest for the possible attackers.

V. Experimental Setup

In [9] we showed that AIS can be efficiently used for detecting misbehavior in sensor networks. In this document we provide an in-depth analysis of relative usefulness of genes, the variability of antigens and the efficiency of the negative selection learning process. The experimental setup is similar to [9]; the focus of performance analysis is however different.

Definitions of input and output parameters: The input parameters for our experiments were: r parameter for the r-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 averages and 95% confidence intervals $ci_{95\%}$ of detection rate, number of false positives, real time to compute detectors, data traffic rate at nodes, number of different antigens in a run, and number of matches for each gene. 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. The number of matches for each gene was evaluated using the r-contiguous matching rule; we considered two cases: i) two bit-strings get matched from the left to the right and the first such a match will get reported (matching gets interrupted), ii) two bit-strings get matched from the left to the right and all possible matches will get reported. There is a notable performance difference[4] between these two approaches. The first approach is exactly what we used when computing the real time necessary for negative selection, the second approach was used when our goal was to evaluate relative usefulness of each gene.

Scenario description: We wanted to capture "self" and "non-self" packet traffic in a large enough synthetic static sensor network and test whether using an AIS we are able to recognize non-self, i.e. misbehavior. 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 [15]. 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.

[4]Both are $O(r(l-r))$ s.t. $r \leq l$, where l is the bitstring length. In the second case, all $l-r$ comparisons must be done.

We used 10 CBR (Constant bit rate) connections; the CBR data traffic may not correspond to "real" sensor network traffic, however, there is no much practical experience with these types of networks, and the data traffic pattern can substantially vary in the future. The connections were chosen so that their length is $\sim$7 hops and so that these connections share some common intermediate nodes; see [9] for a visualization of the network and the connections. 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.

Encoding of self and non-self antigens: Each of the five genes was transformed in a 10-bit signature where each bit defines an interval[5] 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 (the Gray coding is an alternative solution).

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 (implemented at the network layer; data packets[6] that should get dropped will simply not be inserted into the IP queue); 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 (encoding).
3) The 20 "normal" traffic runs were used to compute detectors. Given the 28 windows and 20 runs, the sample size was 20×28 = 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 Fig. 2 was used for detection; we restricted ourselves to nodes that had in both the normal and misbehavior traffic at least a certain number of data packets to forward (packet threshold).

The experiment was then repeated with different r, desired number of detectors and misbehavior level.

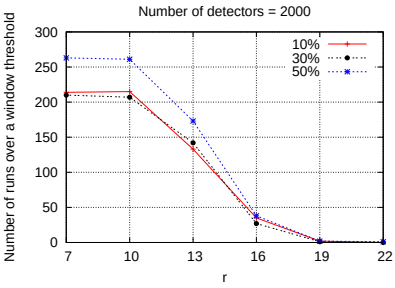
The parameters for this experiment are summarized in Fig. 4. The injection rate and packet sizes were chosen in order to comply with usual data rates of sensors (e.g. 38.4kbps for Mica2; see [1]). We chose the Glomosim simulator over

[5]The interval encoding of genes is adapted from [21]. 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.

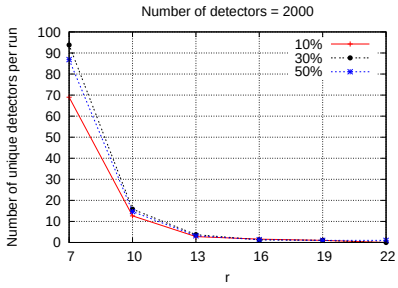
[6]Data packets include both data packets from the transport layer as well as routing protocol packets.

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, detection rate, rate of non-valid detectors, data traffic rate at nodes, number of different antigens in a run, number of matches for each gene; 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.

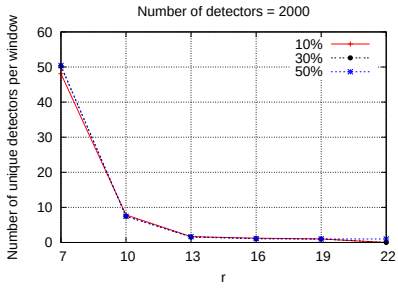
Fig. 4. Parameters used in the experiment.



(a) Total number of runs with window threshold ≥ 14.

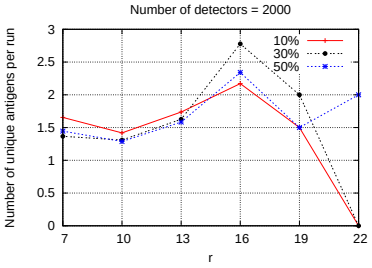


(b) The number of unique detectors that matched an antigen in a run. Conf. interval range for $7 \leq r \leq 13$ is $ci_{95\%}$ = 6.5-10.1%.

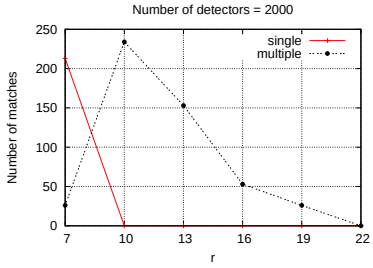


(c) The number of unique detectors that matched an antigen in a window; each run has 28 windows. Conf. interval ranges: $ci_{95\%} < 0.16\%$.

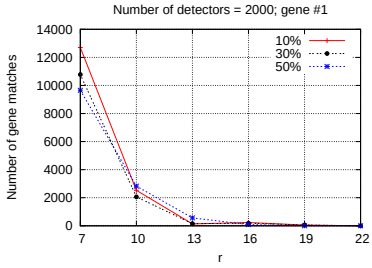
Fig. 5. Window threshold and detector related performance measures.



(a) Number of unique antigens per run. Conf. interval range for $7 \leq r \leq 13$ is $ci_{95\%}$ = 5.3-8.9%.

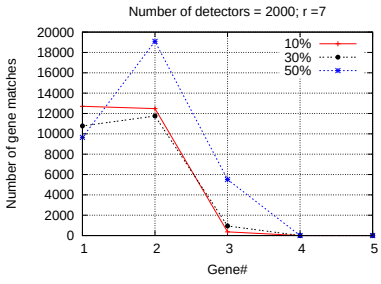


(b) Total number of matches; $single$ = one gene got matched, $multiple$ = more than one gene got matched.

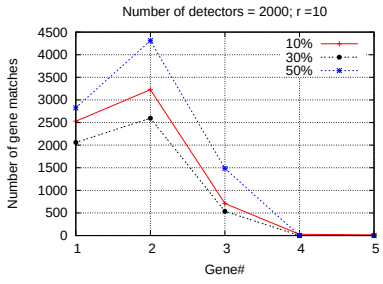


(c) Total number of matches for Gene #1.

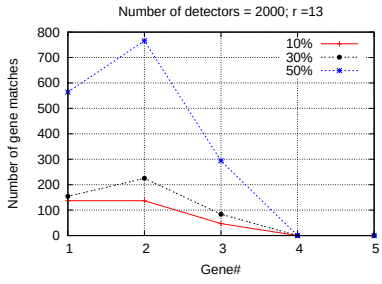
Fig. 6. Antigen and gene related performance measures.



(a)



(b)



(c)

Fig. 7. Performance of Genes #1 through #5 for the number of detectors = 2000 and (a) $r = 7$, (b) $r = 10$, (c) $r = 13$.

other options (most notably ns2) because of its better scaling characteristics and our familiarity with the tool.

VI. Results Evaluation

When evaluating our results we define two additional constraints:

C1. We define a node to be detected as misbehaving if it gets flagged in at least 14 out of the 28 possible windows. This notion indirectly defines the time until a node is pronounced to be misbehaving. We call this a *window threshold*.

C2. A node s_i has to forward in average at least m packets over the 20 runs in both the "normal" and misbehavior cases in order to be included into our statistics. This constraint was set in order to make the detection process more reliable. It is dubious to flag a neighboring node of s_i as misbehaving, if it is based on "normal" runs or runs with misbehavior, in which node s_i had no data packets to forward (he was not on a routing path). We call this a *packet threshold*; m was in our simulations chosen from $\{500, 1000, 2000, 4000\}$. *Example:* for a fixed set of input parameters, a node forwarded in the "normal" runs in average 1,250 packets and in the misbehavior runs (with e.g. level 30%) 750 packets. The node s_i would be considered for misbehavior detection if $m = 500$, but not if $m \geq 1000$. In other words, a node has to get a chance to learns what is "normal" and then to use this knowledge on a non-empty packet stream.

In Fig. 5(a) we show the total number of runs in which a node was identified as misbehaving (for this and remaining figures, the C1 constraint had to be met). The steep decline for values $r > 10$ (in this and other figures) documents that in these cases it was necessary to produce a higher number of detectors in order to cover the non-self antigen space. The higher the r, the higher is the specificity of a detector, this means that it is able to match a smaller set of non-self antigens. In Fig. 5(b) and (c) we show the number of detectors that got matched during the detection phase (see Fig. 2). Fig. (b) shows the number of detectors matched per run, Fig. (c) shows the number of detectors matched per window. Fig. (b) is an upper estimate on the number of unique detectors needed in a single run. Given that the total number of detectors was 2,000, there were less than 5% detectors that would get used in the detection phase. The confidence intervals[7] for the number of unique detectors matched per window (see Fig. (c)) is a direct consequence of the small variability of antigens as shown in Fig. 6(a).

Fig. 6(a) shows the number of unique antigens that were subject to classification into self or non-self. The average for $r = \{7, 10\}$ is about 1.5. This fact does not directly imply that the variability of the data traffic would be inadequate. It is rather a direct consequence of our choice of genes and their encoding (we only used 10 value levels for encoding). Fig. 6(b) shows the number of matches between a detector and an antigen in the following way. When a detector under the r-contiguous matching rule matches only a single gene within an antigen, we would increment the "single" counter. Otherwise, we would increment the "multiple" counter (for this and remaining figures both the C1 and C2 constrain had to be met). It is obvious that with increasing r, it gets more and more probable that a detector would match more than a single gene. The interesting fact is that the detection rate for both $r = 7$ and $r = 10$ is in both cases at about 80% (see Fig. 3(a)). This means the increased r did not substantially help to increase the detection rate. The main difference is the real time to compute the set of 2,000 detectors (see Fig. 3(b)) and the rate of non-valid detectors [9] (detectors that got rejected because they matched a self string; see Fig. 1).

Fig. 6(c) shows the performance of Gene #1. The number of matches shows that this gene contributed to the overall detection performance of our AIS. Figs. 7(a-c) sum up performance of the five genes for different values of r. Again, an interesting fact is the contribution of Gene #1 to the overall detection performance. The usefulness of Gene #2 was largely expected as this gene was tailored for the kind of misbehavior that we implemented. The other three genes came out as marginally useful. The importance of the somewhat surprising performance of Gene #1 is that it can be computed in a simplistic way and does not require continuous operation of a node.

The detailed results regarding the (general) performance of our AIS were already discussed in [9]. In short, we found out that a detection rate of above 80% at a moderate rate of false positives is possible for sensor networks. We also concluded that the real time needed to compute a detector set may not be prohibitively high for sensor networks.

VII. Related Work

In [21], [19] 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 [13] the authors describe an AIS able to detect anomalies at the transport layer of the OSI protocol stack; only a wired TCP/IP network is considered. Self is defined as normal pairwise 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. [17] 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 this is due to the choice of problem representation and due to the choice of matching threshold r for r-contiguous bits matching.

To overcome the deficiencies of the generate-and-test approach a different approach is outlined in [18]. Several signals each having a different function are employed in order to detect a specific misbehavior in sensor wireless networks.

[7] For practical reasons we show $ci_{95\%}$ only for $7 \leq r \leq 13$.

Unfortunately, no proper performance analysis was presented and the properties of these signals were not evaluated with respect to their misuse.

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 [21], [19] we showed in [9] that in case of static sensor networks it is reasonable to expect the detection rate to be above 80%.

VIII. Conclusions and Future Work

We showed the influence and usefulness of certain genes in order to detect misbehavior and the impact of the r parameter on the detection process. In general, the results in Fig. 7 show that Gene #1 and #2 obtained of all genes the best results, with Gene #2 showing always the best results. The contribution of Gene #1 suggests that observing the MAC layer and the ratio of complete handshakes to the number of RTS packets sent is useful for the implemented misbehaviour. Gene #2 fits perfectly for the implemented misbehavior. It therefore comes as no surprise that this gene showed the best results in the detection process. The question which remains open is whether the two genes are still as useful when exposed to different attack patterns.

We conclude that the random-generate-and-test process, with no knowledge of the used protocols and their behavior, creates many detectors which might show to be superfluous in detecting misbehavior. A process with some basic knowledge of protocol limitations might lead to improved quality of detectors.

In [22] the authors stated that the random-generate-and-test process *"is inefficient, since a vast number of randomly generated detectors need to be discarded, before the required number of the suitable ones are obtained"*. Our results show that at $r = 10$, the rate of discarded detectors is less than 4%; see [9] for details. Hence, at least in our setting we could not confirm the above statement. A disturbing fact is, however, that the size of the self set in our setting was probably too small in order to justify the use of negative selection. A counter-balancing argument is here the realistic setup of our simulations and a decent detection rate.

We would like to point out that the Fisher iris and biomedical data sets used in [22] could be very different from data sets generated by our simulations. Our experiments show that anomaly (misbehavior) data sets based on sensor networks could be in general very sparse. This effect can be due to the limiting nature of communications protocols. Since the Fisher iris and biomedical data sets were in [22] not evaluated with respect to some basic properties e.g. degree of clustering, it is hard to compare our results with the results presented therein.

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

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

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

[6] S. Cayzer, J. Smith, J.A.R. Marshall, T. Kovacs. What have Gene Libraries done for AIS? *Proc. the 4th International Conference on Artificial Immune Systems (ICARIS)*, 2005.

[7] D. Dasgupta, F. Gonzalez. An immunity-based technique to characterize intrusions in computer networks. *IEEE Trans. Evolutionary Computation*, vol. 6, no. 3, pp. 281–291, 2002.

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

[9] M. Drozda, S. Schaust, H. Szczerbicka. Is AIS Based Misbehavior Detection Suitable for Wireless Sensor Networks? *Proc. IEEE Wireless Communications and Networking Conference (WCNC)*, 2007.

[10] M. Drozda, S. Schaust, H. Szczerbicka. Simulation of Misbehaviour Detection in Wireless Ad Hoc Networks. *Proc. 19th Symposium on Simulation Technique (ASIM)*, 2006.

[11] 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)*, 2006.

[12] F. Gonzalez, D. Dasgupta, J. Gomez. The effect of binary matching rules in negative selection. *Proc. Genetic and Evolutionary Computation Conference (GECCO)*, 2003.

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

[14] Z. Ji, D. Dasgupta. Real-Valued Negative Selection Algorithm with Variable-Sized Detectors. *Proc. Genetic and Evolutionary Computation Conference (GECCO)*, 2004.

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

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

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

[18] J. Kim, P. Bentley, Ch. Wallenta, M. Ahmed and S. Hailes. Danger Is Ubiquitous: Detecting Malicious Activities in Sensor Networks Using the Dendritic Cell Algorithm. *Proc. International Conference on Artificial Immune Systems (ICARIS)*, 2006.

[19] J.-Y. Le Boudec, S. Sarafijanović. An Artificial Immune System Approach to Misbehavior Detection in Mobile Ad-Hoc Networks. *Proc. Bio-ADIT'04*, 2004.

[20] S. Marti, T. J. Giuli, K. Lai, M. Baker. Mitigating routing misbehavior in mobile ad hoc networks. *Proc. the 6th annual international conference on Mobile Computing and Networking (MobiCom)*, 2000.

[21] S. Sarafijanović, J.-Y. 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.

[22] T. Stibor, P. Mohr, J. Timmis and C. Eckert. Is negative selection appropriate for anomaly detection? *Proc. Conference on Genetic and evolutionary computation (GECCO)*, 2005.

[23] W. Ye, J. Heidemann, D. Estrin. Medium access control with coordinated adaptive sleeping for wireless sensor networks. *IEEE/ACM Transactions on Networking*, vol. 12, no. 3, pp. 493–506, 2004.

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