# Simulation of Misbehaviour Detection in Ad Hoc Wireless Networks*

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

**Abstract**

The goal was to test performance and suitability of Artificial immune systems for detecting misbehavior in ad hoc wireless networks. We have used a realistic simulation setup consisting of a medium sized ad hoc wireless network. Additionally, we have tested scalability of this approach using synthetic input sets.

## 1 Introduction and Motivation

Ad hoc wireless networks lack a centralized authority that controls the flow of packets. Instead, each node in an ad hoc network serves as a routing device. Each node is able to forward packets to its neighbors[1], and vice-versa each node is able to receive packets only from its neighbors. Node movement is allowed, nodes can also be switched off and on at any time. Such networks are extremely vulnerable to user misbehavior. Since nodes within an ad hoc network 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 ad hoc networks are expected to operate autonomously with no or spare maintenance. Classical intrusion detection approaches, many of which are based on intrusion signatures, are therefore not suitable for such a task.

Artificial immune systems (AIS) on the other hand seem capable of handling such demands. AIS are inspired by the *Human immune system* (HIS) using selected features of this defense system. The basic feature of an HIS is the ability to discriminate between *self* and *non-self* antigen;[2] 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. In case of ad hoc networks non-self is any kind of user behavior that impacts the network in a negative way. The purpose of our simulation based experiments is to show that AIS are a suitable approach for detecting misbehavior in ad hoc networks.

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

[1]Nodes that lie within radio range of the sending node.

[2]Self and non-self in short.

## 2 Learning in Artificial Immune Systems

The process of T-cells maturation in thymus is used as an inspiration for learning in AIS. T-cells are covered by receptors that are able to bind antigens. The creation of T-cells (detectors) in thymus is a result of a pseudo-random process. After a T-cell is created, it undergoes a censoring process called *negative selection*. During negative selection T-cells that bind self are destroyed (random generate and test). 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. 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*.

## 3 Experimental Setup

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 means of negative selection when detectors are created randomly and tested against a set of self strings. Similar to [2] we have collapsed different sorts of T-cells into a single entity called detector.

*Growing and shrinking of detectors:* We implemented two enhancements to negative selection. These were motivated by the results of Ji and Dasgupta in [1]. In each case, a range for the $r$ parameter is chosen, i.e. $r = \{r_1, r_2, ..., r_i, r_{i+1}, ..., r_k\}$, $k$ is the size of the range, $r_{i+1} = r_i + 1$. The binding ability of a detector doubles when $r$ is set to $r_{i-1}$. This range can be chosen experimentally as in our case, or a heuristic can be used. In the first enhancement, when a valid detector was produced, i.e. a detector that matches no self string, we tried to increase his binding ability by decrementing $r$. This requires that the initial value of $r$ is chosen at the upper end the range. We keep decrementing $r$ until we either reach the lower limit of the range or the detector matches a self string. In the latter case we used the last $r$ that produced a valid detector. In the second enhancement we try to decrease the binding ability of a detector. The initial value of $r$ is set to the lower limit of the range, then either the detector is valid in which case we keep it, or in other case we decrease its binding ability by choosing the next larger $r$ value from the range. If this new larger value produces a valid detector, we keep it, otherwise we keep increasing its value until we reach the upper limit of the range. If $r$ at the upper limit of the range does not produce a valid detector then this detector is rejected. These two ideas can be concisely described as growing and shrinking binding (matching) power of detectors. In [1] only growing of detectors is considered; their setup uses real-valued detectors and not bit-strings as in our case. Moreover, their experiments are not done in the context of wireless networks.

We have undertaken two series of experiments. The first series of experiments uses

1. Negative selection algorithm: random generate and test.
2. **Input parameters:** 1. r-contiguous matching rule with $r = \{7, 12, 16\}$. 2. Encoding: 5 genes each 10 bits long = 50 bits. 3. Number of detectors 500. 4. Window size 500 seconds.
3. **Performance measures:** real time to compute detectors, number of iterations to compute detectors, detection rate, rate of non-valid detectors, number of duplicate detectors and their arithmetic averages.
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 3km×3km.
5. **Number of connections:** 10 connections.
6. **MAC protocol**: IEEE 802.11b DCF. **Routing protocol**: DSR.
7. Other parameters: (i) Propagation path-loss model: two ray (ii) Channel bandwidth: 2 Mb (iii) Channel frequency: 2.4 GHz (iv) Topography: Line-of-sight (v) Radio type: Accnoise (vi) Network protocol: IP (vii) Connection type: UDP. The transmission range of transceivers was 100 meters.
8. **Simulator used:** GlomoSim 2.03; hardware used: several Linux (SuSE 10.0) PCs with 2GB RAM memory and Pentium 4 3GHz microprocessor.
9. **Injection rate:** 1 packet/second. 43,200 packets per connection we re injected. Packet size was 512 bytes.
10. The simulation time was 12 hours.

Figure 1: Parameters used in Experiment 1.

a synthetic static ad hoc network. The second series of experiments uses purely synthetic input data.

**Definitions of input and output parameters:** The input parameters for our experiments were: $r$ parameter for the r-contiguous matching rule, ratio of the self set size and non-self set size $R_S$, and the (desired) number of detectors $N_D$.[3]
The performance (output) measures were arithmetic averages (and the associated standard deviations) of real time to compute detectors, number of iterations to compute detectors (number of random tries), detection rate, rate of non-valid detectors, number of false positives, number of duplicate detectors (detectors that were randomly produced but are identical). 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.
**Experiment 1:** The purpose of this experiment was to capture "self" and "non-self" packet traffic in a synthetic static ad hoc network and test whether using an AIS we are able to recognize non-self, i.e. misbehavior. We only considered packet

[3]It may not always be possible to compute a given number of detectors for a predetermined $r$. This happens when (smaller) $r$ causes that almost all detectors and self strings match under the r-contiguous matching rule. We have set the maximum number of random tries to $10^7$ in order to prevent the negative selection process to loop forever.

traffic at the MAC and routing 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 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.

In this experiment we used a static ad hoc network. The topology was determined by a snapshot of 1,718 mobile nodes moving in a square area of 3×3km as prescribed by the random waypoint mobility model. We have used 10 CBR (Constant bit rate) connections. The connections were chosen so that their length is about 8 hops and so that these connections share some common intermediate nodes. 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), 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 $s_s \neq s_d$. Motivated by [3] we have used the following *genes* to capture some 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 $\infty$ (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 $\infty$ (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 [4].

#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:* Each gene value was transformed in a 10-bit signature where each bit defines an interval[4] of a gene specific value range. We

[4]The interval encoding of genes is adapted from [3].

1. Negative selection algorithm: random generate and test, random generate and test with growing detectors, random generate and test with shrinking detectors.
2. **Input parameters:** 1. r-contiguous matching rule with $r = \{6, 7, 8, 9, 10\}$. 2. Number of detectors 25–600 in 25 increments. 3. Size of self population $R_S$: 30%, 50%, 70%, 90% of 1,000.[a] Size of non-self population: app. 2% chosen from the complement to the self set i.e. there were $20 \pm 5$ non-self bit-string to be detected.
3. **Performance measures:** real time to compute detectors, number of iterations to compute detectors, detection rate, rate of non-valid detectors, number of duplicate detectors and their arithmetic averages.
4. Number of runs: 20 for each combination of input parameters.
5. Non-self generation method: chosen randomly and uniformly from the complement to the self set.
6. Simulation time: until the desired number of detectors is computed and tested.
7. Encoding: self and non-self strings were encoded as 30-bit bit-strings.
8. Hardware used: several Linux (SuSE 10.0) PCs with 2GB RAM memory and Pentium 4 3GHz microprocessor. Implementation in C++; compiled with GNU gcc/g++ v4.0.

[a]The *absolute* size of the self set was 1,000 bit-strings. $R_S$ was set to either of these values $\{30\%, 50\%, 70\%, 90\%\}$. In an interval $(0.0, 100.0)$ we have chosen three mid-points $\{15.0, 45.0, 75.0\}$. If the ratio is 30% it means that the 1,000 self string were randomly (uniformly) chosen from three intervals $\{15.0 \pm 5.0, 45.0 \pm 5.0, 75.0 \pm 5.0\}$.

Figure 2: Parameters used in Experiment 2.

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 preserve locality information and thus improve efficiency of the r-contiguous bits matching rule.

*Constructing the self and non-self sets:* We have randomly chosen a 500-second window in our 12-hour simulation. In this 500-second window a self antigen is computed for each node. Misbehavior was modeled as random data packet dropping; we have randomly chosen 236 nodes and these were forced to drop $\{10, 20, 30, 50\%\}$ of data packets (the impact of this misbehavior is smaller than it might appear because many of these 236 nodes had no data packets to drop). We have avoided the first minute in the simulation during which the routing protocol tries to establish valid routes by flooding the network. We have run these experiments for different values of the $r$ parameter for r-contiguous bits matching and for different desired numbers of detectors. The parameters for this experiment are summarized in Figure 1.

**Experiment 2:** The purpose of this experiment was to test scalability of AIS based on the negative selection process. The experimental setup is summed up in Figure 2. Due to lack of space, we leave out detailed experiment description; the results are discussed in the next section.

# 4 Results

**Experiment 1:** Results can be summed up as follows: (i) The effect of misbehavior on the network is hard to predict. Due to misbehavior, new routes will get established by the routing protocol. The consequence of this is detected anomaly at nodes that do not misbehave; this problem is solved in e.g. [3] by introduction of a danger signal - only detected anomaly that coincides with a danger signal being emitted is classified as misbehavior. (ii) Increasing the $r$ parameter increases the detection reliability. (iii) To produce a suitable set of genes is a lengthy process requiring plenty of experimentation; to automate this process will require formal analysis of communication protocols.
**Experiment 2:** Results are as follows: (i) There is a sharp phase transition when producing detectors with a given $r$ parameter; as $r$ becomes smaller producing detectors becomes quickly impossible. (ii) The detection rate rises quickly with the desired number of detectors. (iii) Real time needed to compute detectors is feasible; we have obtained a wireless sensor kit from Crossbow in order to test computational feasibility also on smaller devices. (iv) Growing and shrinking detectors does not offer any advantage as compared to uniformly distributed detectors (in contrast to [1]).

# 5 Conclusions

The purpose of our experiments was to verify whether misbehavior detection based on AIS is a viable approach. The results seem to hint that this is indeed the case. Based on our experimentation we believe the most important challenges of any AIS based approach are: (i) constructing a suitable set of genes, (ii) implementation of a danger signal mechanism.

# References


[1] Zhou Ji, Dipankar Dasgupta. Real-Valued Negative Selection Algorithm with Variable-Sized Detectors. *Proc. Genetic and Evolutionary Computation Conference 2004 (GECCO-2004),* 2004: 287-298 .

[2] S. Hofmeyr and S. Forrest. Immunity by Design: An Artificial Immune System. *Proc. Genetic and Evolutionary Computation Conference (GECCO)*, Morgan-Kaufmann, San Francisco, CA, pp. 1289-1296 (1999).

[3] Slaviša Sarafijanović and 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. ICARIS (Third international conference on artificial immune systems)*, 2004.

[4] Sergio Marti and T. J. Giuli and Kevin Lai and Mary Baker. Mitigating routing misbehavior in mobile ad hoc networks. *Mobile Computing and Networking*, pp. 255-265, 2000.