In Proceedings of Bio-ADIT 2004 (The First International Workshop on Biologically Inspired Approaches to Advanced Information Technology), pp. 96-111, January 29-30, 2004, Lausanne, Switzerland

An Artificial Immune System Approach to Misbehavior Detection in Mobile Ad-Hoc Networks

Jean-Yves Le Boudec and Slaviša Sarafijanović

EPFL/IC/ISC/LCA,
CH-1015 Lausanne, Switzerland
{jean-yves.leboudec, slavisa.sarafijanovic}@epfl.ch

Abstract. In mobile ad-hoc networks, nodes act both as terminals and information relays, and participate in a common routing protocol, such as Dynamic Source Routing (DSR). The network is vulnerable to routing misbehavior, due to faulty or malicious nodes. Misbehavior detection systems aim at removing this vulnerability. In this paper we investigate the use of an Artificial Immune System (AIS) to detect node misbehavior in a mobile ad-hoc network using DSR. The system is inspired by the natural immune system of vertebrates. Our goal is to build a system that, like its natural counterpart, automatically learns and detects new misbehavior. We describe the first step of our design; it employs negative selection, an algorithm used by the natural immune system. We define how we map the natural immune system concepts such as self, antigen and antibody to a mobile ad-hoc network, and give the resulting algorithm for misbehavior detection. We implemented the system in the network simulator Glomosim; we present detection results and discuss how the system parameters impact the results. Further steps will extend the design by using an analogy to the innate system, danger signals, costimulation and memory cells.

1 Introduction

1.1 Problem Statement: Detecting Misbehaving Nodes in DSR

Mobile ad-hoc networks are self organized networks without any infrastructure other than end user terminals equipped with radios. Communication beyond the transmission range is made possible by having all nodes act both as terminals and information relays. This in turn requires that all nodes participate in a common routing protocol, such as Dynamic Source Routing (DSR) [16]. A problem is that DSR works well only if all nodes execute the protocol correctly, which is difficult to guarantee in an open ad-hoc environment.

A possible reason for node misbehavior is faulty software or hardware. In classical (non ad-hoc) networks run by operators, equipment malfunction is known to be an important source of unavailability [17]. In an ad-hoc network, where routing is performed by user provided equipment, we expect the problem to be exacerbated. Another reason for misbehavior stems from the desire to save battery power: some nodes may run a modified code that pretends to participate in DSR but, for example, does not forward

packets. Last, some nodes may also be truly malicious and attempt to bring the network down, as do Internet viruses and worms. An extensive list of such misbehavior is given in [2]. The main operation of DSR is described in Section 2.1. In our simulation, we implement a faulty node that, from time to time, does not forward data or route requests, or does not respond to route requests from its own cache.

We consider the problem of detecting nodes that do not correctly execute the DSR protocol. The actions taken after detecting that a node misbehaves range from forbidding to use the node as a relay [1] to excluding the node entirely from any participation in the network [3]. In this paper we focus on the detection of misbehavior and do not discuss actions taken after detection.

We chose DSR as a concrete example, because it is one of the protocols being considered for standardization for mobile ad-hoc networks. There are other routing protocols, and there are parts of mobile ad-hoc networks other than routing that need misbehavior detection, for example, the medium access control protocol. We believe the main elements of our method would also apply there, but a detailed analysis is for further work.

1.2 Traditional Misbehavior Detection Approaches

Traditional approaches to misbehavior detection [1, 3] use the knowledge of anticipated misbehavior patterns and detect them by looking for specific sequences of events. This is very efficient when the targeted misbehavior is known in advance (at system design) and powerful statistical algorithms can be used [4].

To detect misbehavior in DSR, Buchegger and Le Boudec use a reputation system [3]. Every node calculates the reputation of every other node using its own first hand observations and second hand information obtained from others. The reputation of a node is used to determine whether countermeasures against the node are undertaken or not. A key aspect of the reputation system is how second hand information is used, in order to avoid false accusations [3].

The countermeasures against a misbehaving node are aimed to isolate it, i.e., packets will not be sent over the node and packets sent from the node will be ignored. In this way nodes are stimulated to cooperate in order to get service and maximize their utility, and the network also benefits from the cooperation.

Even if not presented by its authors as an artificial immune system, the reputation system in [3, 4] is an example of (non-bio inspired) immune system. It contains interactions between its healthy elements (well behaving nodes) and detection and exclusion reactions against non-healthy elements (misbehaving nodes). We can compare it to the natural *innate* immune system (Section 2.2), in the sense that it is hardwired in the nodes and changes only with new versions of the protocol.

Traditional approaches miss the ability to learn about and adapt to new misbehavior. Every target misbehavior has to be imagined in advanced and explicitly addressed in the detection system. This is our motivation to use an artificial immune system approach.

1.3 Artificial Immune System (AIS) Approaches

An AIS uses an analogy with the natural Immune System (IS) of vertebrates. As a first approximation, the IS can be described with the “self, non self” model, as follows (we give more details in Section 2.2).

The IS is thought to be able to classify cells that are present in the body as self and non-self cells. The IS is made of two distinct sets of components: the innate IS, and the adaptive IS. The innate IS is hard-wired to detect non self (and destroy) cells that contain, or do not contain, specific patterns on their surface.

The adaptive IS is more complex. It produces a large number of randomly created detectors. A "negative selection" mechanism eliminates detectors that match all cells present in a protected environment (bone marrow and the thymus) where only self cells are assumed to be present. Non-eliminated detectors become "naive" detectors; they die after some time, unless they match something (assumed to be a pathogen), in which case they become memory cells. Further, detectors that do match a pathogen are quickly multiplied ("clonal selection"); this is used to accelerate the response to further attacks. Also, since the clones are not exact replicates (they are mutated, the mutation rate being an increasing function of affinity between detector and antigen) this provides a more focused response to the pathogen ("affinity maturation"). This also provides adaptation to a changing non-self environment.

The self-nonself model is only a very crude approximation of the adaptive IS. Another important aspect is the "danger signal" model [11, 12]. With this model, matching by the innate or adaptive mechanism is not sufficient to cause detection; an additional danger signal is required. The danger signal is for example generated by a cell that dies before being old. The danger signal model better explains how the IS adapts not only to a changing non-self, but also to some changes in self. There are many more aspects to the IS, some of which are not yet fully understood (see Section 2.2).

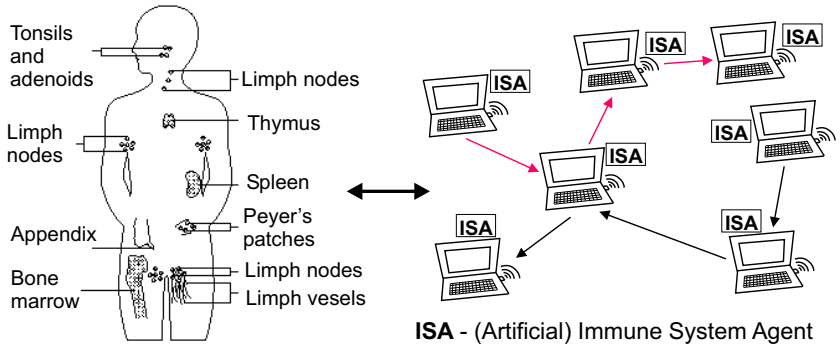


Fig. 1. From the natural IS to an AIS: Making DSR immune to node misbehavior.

1.4 Artificial Immune Systems - Related Work

Hofmeyer and Forrest use an AIS for intrusion detection in wired local area networks [5, 6]. Their work is based on the negative selection part of the self-nonself model and some form of danger signal. TCP connections play the role of self and nonself cells. One connection is represented by a triplet encoding sender's destination address, receiver's destination address and receiver's port number. A detector is a bit sequence of the same

length as the triplet. A detector matches a triplet if both have M contiguous bits equal, where M is a fixed system parameter. Candidate detectors are generated randomly; in a learning phase, detectors that match any correct (i.e. self) triplets are eliminated. This is done offline, by presenting only correct TCP connections. Non-eliminated detectors have a finite lifetime and die unless they match a non-self triplet, as in the IS. The danger signal is also used: it is sent by humans as confirmation in case of potential detection. This is a drawback, since human intervention is required to eliminate false positives, but it allows the system to learn changes in the self. With the terminology of statistical pattern classification, this use of the danger signal can be viewed as some form of supervised training. Similarly, Dasgupta and González [20] use an AIS approach to intrusion detection, based on negative selection and genetic algorithms.

A major difficulty in building an artificial immune system in our framework is the mapping from biological concepts to computer network elements. Kim and Bentley show that straightforward mappings have computational problems and lead to poor performance, and they introduce a more efficient representation of self and nonself than in [5]. They show the computational weakness of negative selection and add clonal selection to address this problem [8]. In their subsequent papers, they examine clonal selection with negative selection as an operator [9], and dynamical clonal selection [10], showing how different parameters impact detection results. For an overview of AIS, see the book by de Castro and Timmis [19] and the paper by de Castro and von Zuben [18].

1.5 Contribution of this Paper and Organization

Our long term goal is to understand whether our previous work, based on the traditional approach [2], can benefit from an AIS approach that introduces learning and adapting mechanisms. This paper is our first step in this direction.

The first problem to solve is mapping the natural IS concepts to our framework. This is a key issue that strongly influences the detection capabilities. We describe our solution in Section 3.1. For the representation of self-nonself and for the matching functions, we start from the general structure proposed by Kim and Bentley [8], which we adapt to our case. Then we define the resulting algorithm, which, in this first step, is based only on negative selection. Our main contribution in this phase is the definition of a mapping and a construction of an AIS adapted to our case, its implementation in the Glomosim simulator, and its performance analysis.

The rest of the paper is organized as follows. Section 2 gives background and terminology on DSR and the natural immune system. Section 3 gives the mapping from the IS to the detection system for DSR misbehavior detection, and the detailed definition of the detection system. Section 4 gives simulation specific assumptions and constraints, simulation results and discussion of the results. Section 5 draws conclusions and describes what we have learned and how we will exploit it in future steps.

2 Background

2.1 DSR: basic operations

Dynamic source routing protocol is one of the candidate standards for routing in mobile ad hoc networks [16]. A “source route” is a list of nodes that can be used as intermediate relays to reach a destination. It is written in the data packet header at the source; intermediate relays simply look it up to determine the next hop.

DSR specifies how sources discover, maintain and use source routes. To discover a source route, a node broadcasts a route request packet. Nodes that receive a route request add their own address in the source route collecting field of the packet, and broadcast the packet, except in two cases. The first case is if the same route request was already received by a node; then the node discards the packet. Two received route requests are considered to be the same if they belong to the same route discovery, what is identified by the same value of source, destination and sequence number fields in the request packets. The second case is if the receiving node is destination of the route discovery, or if it already has a route to the destination in its cache; then the node sends a route reply message that contains a completed source route. If links in the network are bidirectional, the route replies are sent over the reversed collected routes. If links are not bidirectional, the route replies are sent to the initiator of the route discovery as included in a new route request generated by answering nodes. The new route requests will have as the destination the source of the initial route request. The node that initiate original route request gets usually more route replies, every containing a different route. The replies that arrive earlier then others are expected to indicate better routes, because for a node to send a route reply, it is required to wait first for a time proportional to the number of hops in the route it has as answer. If a node hears that some neighbor node answers during this waiting time, it supposes that the route it has is worse then the neighbor's one, and it does not answer. This avoids route reply storms and unnecessary overhead.

After the initiator of route discovery gets first route reply, it sends data over obtained route. While packets are sent over the route, the route is maintained, in such a way that every node on the route is responsible for the link over which it sends packets. If some link in the route breaks, the node that detects that it cannot send over that link should send error messages to the source. Additionally it should salvage the packets destined to the broken link, i.e., reroute them over alternate partial routes to the destination.

The mechanisms just described are the basic operation of DSR. There are also some additional mechanisms, such as gratuitous route replies, caching routes from forwarded or overheard packets and DSR flow state extension [16].

2.2 The Natural Immune System

The main function of the IS is to protect the body against different types of pathogens, such as viruses, bacteria and parasites and to clear it from debris. It consists of a large number of different innate and acquired immune cells, which interact in order to provide detection and elimination of the attackers [13]. We present a short overview based on the self-nonself and the danger models [13, 12].

Functional architecture of the IS. The first line of defense of the body consists of physical barriers: skin and mucous membranes of digestive, respiratory and reproductive tracts. It prevents the body from being entered easily by pathogens.

The innate immune system is the second line of defense. It protects the body against common bacteria, worms and some viruses, and clears it from debris. It also interacts with the adaptive immune system, signaling the presence of damage in self cells and activating the adaptive IS.

The adaptive immune system learns about invaders and tunes it's detection mechanisms to better match previously unknown pathogens. It provides an effective protection against viruses even after they enter the body cells. It adapts to newly encountered viruses and memorizes them for more efficient and fast detection in the future.

The innate immune system consists of macrophages cells, complement proteins and natural killer cells. Macrophages are big cells that are attracted by bacteria to engulf them in the process called "phagocytosis". Complement proteins can also destroy some common bacteria. Both macrophages and complement proteins send signals to other immune cells when there is an attack.

The adaptive immune system consists of two main types of lymphocyte cells: B cells and T cells. Both B and T cells are covered with antibodies. Antibodies are proteins capable to chemically bind to nonself antigens. Antigens are proteins that cover the surface of self and nonself cells. Whether chemical binding will happen between an antibody and an antigen depends on the complementarity of their three-dimensional chemical structures. If it does, the antigen and the antibody are said to be "cognate". Because this complementarity does not have to be exact, an antibody may have several different cognate antigens. What happens after binding depends on additional control signals exchanged between different IS cells, as we explain next.

One B cell is covered by only one type of antibody, but two B cells may have very different antibodies. As there are many B cells (about 1 billion fresh cells are created daily by a healthy human), there is also a large number of different antibodies at the same time. How is this diversity of antibodies created and why do antibodies not match self antigens ? The answer is in the process of creating B cells. B cells are created from stem cells in the bone marrow by rearrangement of genes in immature B cells. Stem cells are generic cells from which all immune cells derive. Rearrangement of genes provides diversity of B cells. Before leaving bone marrow, B cells have to survive **negative selection**: if the antibodies of a B cell match any self antigen present in the bone marrow during this phase, the cell dies. The cells that survive are likely to be self tolerant.

B cells are not fully self tolerant, because not all self antigens are presented in bone marrow. **Self tolerance** is provided by T cells that are created in the same way as B cells, but in the thymus, the organ behind the breastbone. T cells are self tolerant because almost all self antigens are presented to these cells during negative selection in the thymus.

After some antibodies of a B cell match antigens of a pathogen or self cell (we call this event "signal 1b") that antigens are processed and presented on the surface of the B cell. For this, Major-Histocompatibility-Complex (MHC) molecules are used and their only function. If antibodies of some T cell bind to these antigens and if the T cell is activated (by some additional control signal) the detection is verified and a confirmation signal sent from T cell to B cell (we call this event "signal 2b"). Signal 2b starts the process of producing new B cells, that will be able to match the pathogen better. This process is called clonal selection. If signal 2b is absent, it means that the detected antigens are probably self antigens for which the T cells are tolerant. In this last case, the B cell will die together with its self reactive antibodies.

B cells can begin clonal selection without confirmation by signal 2b, but only in the case when matching between B cell antibodies and antigens is very strong. This occurs with a high probability only for memory B cells, the cells that were verified in the past to match nonself antigens.

In the **clonal selection** phase, a B cell divides into a number of clones with similar but not strictly identical antibodies. Statistically, some clones will match the pathogen that triggered the clonal selection better than the original B cells and some will match it worse. If the pathogens whose antigens triggered clonal selection are still present, they will continue to trigger new B cells that match pathogen well to clone. The process continues reproducing B cells more and more specific to present pathogens. B cells that are specific enough become memory B cells and do not need **costimulation** by signal 2b. This is a process with positive feedback and it produces a large number of B cells specific to the presented pathogen. Additionally, B cells secrete chemicals that neutralize pathogens. The process stops when pathogens are cleared from the body. Debris produced by the process are cleared by the innate immune system. Memory B cells live long and they are ready to react promptly to the same cognate pathogen in the future. Whereas first time encountered pathogens require a few weeks to be learned and cleared by the IS, the secondary reaction by memory B cells takes usually only a few days.

The Danger Signal is an additional control used for activating T cells. After T cell antibodies bind to antigens presented by MHC of a B cell (signal 1t), the T cell is activated and sends signal 2b to a B cell only if it receives a confirmation signal (signal 2t) from an Antigen Presenting Cell (APC). The APC will give signal 2t to a T cell only if it engulfed the same nonself antigens, which happens only when the APC receive a danger signal from self cells or from the innate immune system. The danger signal is generated when there is some damage to self cells, which is usually due to pathogens. As an example, the danger signal is generated when a cell dies before being old; the cell debris are different when a cell dies out of old age or when it is killed by a pathogen.

There are many other subtle mechanisms in the IS, and not all of them are fully understood. In particular, time constants of the regulation system (lifetime of B and T cells, probability of reproduction) seem to play an important role in the performance of the IS [14]. We expect that we have to tune similar parameters carefully in an AIS.

3 Design of Our Detection System

3.1 Mapping of Natural IS Elements to Our Detection System

The elements of the natural IS used in our detection system are mapped as follows.

Body: the entire mobile ad-hoc network
Self-Cells: well behaving nodes
Non-Self Cells: misbehaving nodes
Antigen: Sequence of observed DSR protocol events recognized in sequence of packet headers. Examples of events are “data packet sent”, “data packet received”, “data packet received followed by data packet sent”, “route request packet received followed by route reply sent”. The sequence is mapped to a compact representation as explained in Section 3.2.

Antibody: A pattern with the same format as the compact representation of antigen (Section 3.2).

Chemical Binding: binding of antibody to antigen is mapped to a "matching function", as explained in Section 3.2.

Negative Selection and Bone Marrow: Antibodies are created during an offline learning phase. The bone marrow (protected environment) is mapped to a network with only certified nodes. In a deployed system this would be a testbed with nodes deployed by an operator; in our simulation environment, this is a preliminary simulation phase.

3.2 Antigen, Antibody and Matching Function

Antigens could be represented as traces of observed protocol events. Protocol events and their timings constitute the behavior of a node, and our goal is to detect this behavior if it is wrong. However, even in low bit rate networks, this rapidly generates sequences that are very long (for a 100 seconds observation interval, a single sequence may be up to 1 Gbit long), thus making generation of a large number of patterns prohibitive. This was recognized and analyzed by Kim and Bentley in [8] and we follow the conclusions, which we adapt to our case, as we describe now.

A node in our system monitors its neighbors and collects one protocol trace per monitored neighbor. A protocol trace consists of a series of data sets, collected on non-overlapping intervals during activity time of a monitored neighbor. One data set consists of protocol events recorded during one time interval of duration Δt ($\Delta t = 10$s by default), with an additional constraint to maximum N_s events per a data set ($N_s = 40$ by default).

Data sets are then transformed as follows. First, protocol events are mapped to a finite set of primitives, identified with labels. In the simulation, we use the following list.

A= RREQ sent
B= RREP sent
C= RERR sent
D= DATA sent and IP source address is not of monitored node
E= RREQ received
F= RREP received
G= RERR received
H= DATA received and IP destination address is not of the monitored node

A data set is then represented as a sequence of labels from the alphabet defined above, for example

$$l_1 = \texttt{(EAFBHHEDEBHDHDHHDHD,..)}$$

Second, a number of "genes" are defined. A gene is an atomic pattern used for matching. We use the following list.

Gene 1 = #E in sequence
Gene 2 = #(E*(A **or** B)) in sequence
Gene 3 = #H in sequence
Gene 4 = #(H*D) in sequence

where #('sub-pattern') is the number of the sub-patterns 'sub-pattern' contained in a sequence such is l_1, with * representing one arbitrarily label or no label at all. For

example, #(E*(A **or** B)) is the number of sub-patterns that are two or three labels long, and that begin with E and end with A or B. The genes are used to map a sequence such as l_1 to an intermediate representation that gives the values of the different genes in one data set. For example, l_1 is mapped to an antigen that consists of the four genes:

$$l_2 = \big(3\ 2\ 7\ 6\big)$$

The genes are defined with the intention to translate raw protocol sequences into more meaningful descriptions. Concretely, we define them in such a way to capture correlation between protocol events. For a normal DSR operation (Section 2.1), the values of genes 1 and 2 are correlated, as well as the values of genes 3 and 4. In the case of misbehavior, this correlation will change. This is a manual way to define the genes, and we discus alternatives in Section 5.

Finally, a gene value is encoded on N_g bits ($N_g = 10$ by default) as follows. A range of values of a gene, that are bellow some threshold value, is uniformly divided on N_g intervals. The position of the interval to whom the gene value belongs gives the position of the bit that is set to 1 for the gene in the final representation. The threshold is expected to be reached or exceeded rarely. The values above the threshold are encoded as if they belong to the last interval. Other bits are set to 0. For example, if N_g=10 and if the threshold value for all the four defined genes is equal to 20, l_2 is mapped to the final antigen format:

$$l_3 = \big(0000000010\ 0000000010\ 0000001000\ 0000001000\big)$$

There is one antigen such as l_3 every Δt seconds, for every monitored node, during activity time of the monitored node. Every bit in this representation is called a "nucleotide".

Antibody and Matching Function. Antibodies have the same format as antigens (such as l_3), except that they may have any number of nucleotides equal to 1 (whereas an antigen has exactly one per gene). An antibody matches an antigen (i.e. they are cognate) if the antibody has a 1 in every position where the antigen has a 1. This is the same as in [9] and is advocated there as a method that allows a detection system to have good coverage of a large set of possible nonself antigens with a relatively small number of antibodies.

Antibodies are created randomly, uniformly over the set of possible antibodies. During negative selection, antibodies that match any self antigen are discarded.

3.3 Node Detection and Classification

Matching an antigen is not enough to decide that the monitored node misbehaves, since we expect, as in any AIS, that false positives occur. We say that a monitored node is **detected** (or "suspicious") in one data set (i.e. in one interval of duration Δt) if the corresponding antigen is matching any antibody. So, detection is done per interval of duration Δt. A monitored node is **classified as "misbehaving"** if the probability that the node is suspicious, estimated over a sufficiently large number of data sets, is above a threshold. The threshold is computed as follows.

Assume we have processed n data sets for this monitored node. Let M_n be the number of data sets (among n) for which this node is *detected* (i.e. is suspicious). Let $\theta_{\max}$

be a bound on the probability of false positive detection (detection of well behaving nodes, as if they are misbehaving) that we are willing to accept, i.e. we consider that a node that is detected with a probability $\leq \theta_{\max}$ is a correct node (we take by default we take $\theta_{\max} = 0.08$). Let α (=0.01 by default) be the classification error that we are willing to accept. We classify the monitored node as misbehaving if

$$\frac{M_n}{n} > \theta_{\max}(1 + \frac{\xi(\alpha)}{\sqrt{n}}\sqrt{\frac{1-\theta_{\max}}{\theta_{\max}}}) \quad (1)$$

where $\xi(\alpha)$ is the $1-\alpha$-quantile of the normal distribution (for example, $\xi(0.01) = 2.33$). As long as Equation (1) is not true, the node is classified as well behaving. With default parameter values, the condition is $\frac{M_n}{n} > 0.08 + 0.27\frac{\xi(\alpha)}{\sqrt{n}}$. The derivation of Equation (1) is given in Appendix 1.

The detailed detection and classification algorithm that is executed in every node (in addition to the standard DSR code) is given in Appendix 2.

3.4 Detection System Parameters

In this implementation, the default values of the parameters (Table 1) are chosen from extensive pilot simulation runs, as a compromise between good detection results and a small memory and computation usage of the detection system.

Table 1. Detection System Parameters

Parameters	Default values
maximal number of self antigens collected for learning	80
maximal time for collecting self antigens for learning	200 s
maximal time for collecting a data set (an antigen)	10 s
max. number of protocol events recorded in a data set	40
number of detectors (i.e. antibodies)	300
number of genes in an antigen	4
number of nucleotides per gene	10
max. accepted misdetection probability $\theta_{\max}$	0.08
targeted classification error ratio α	0.01

4 Simulation Results

4.1 Description of Simulation

We have implemented this first pass of our system in Glomosim [15], a simulated environment for mobile ad-hoc networks. We simulate a network of 40 nodes. The simulation area is a rectangle with the dimensions of 800 m x 600 m. The nodes are initially deployed on a grid with the distance between neighbors equal to 100 m. Mobility model is random way-point with fixed speed of 2 m/s. The radio range is 450 m. Traffic is generated as constant bit-rate, with packets of length 512 bytes sent every 0.2-1 s. We inject one misbehaving node.

For all simulations, the parameters have the default value, except for the number of self antigens collected for learning, the number of antibodies, and misbehavior probability, which are taken as parameters.

Misbehavior is implemented by modifying the DSR implementation in Glomosim. We implemented two misbehavior: (1) non-forwarding route requests and non-answering from its route cache and (2) non-forwarding data packets. The misbehaving node does not always misbehave. In contrast, it does so with a fixed probability for both types of misbehavior, which is also a simulation parameter (default value is 0.6).

Performance Metrics: We show simulation results with the following metrics:

True Positive Detection: percentage of successfully detected nonself antigens

False Positive Detection: percentage of mistakenly detected self antigens

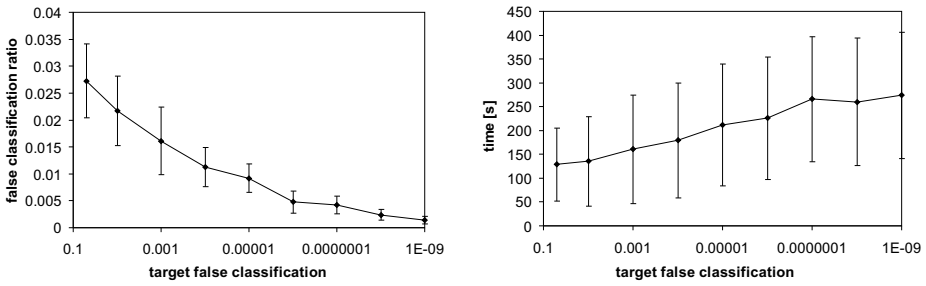
Correct Classification Time: time until the misbehaving node is classified as such (after a sufficiently large number of positive detections occurred, see Equation (1))

False Classification: the percentage of well behaving nodes that are mistakenly classified as misbehaving. This is much less than false positive detection since classification uses evidence from several antigens.

We performed 10 independent replications of all experiments, in order to obtain confidence intervals. All graphs are with 90% confidence intervals.

4.2 Simulation results

Classification capabilities: For all simulation runs, the misbehaving node is detected and classified as misbehaving by all other nodes. The main effect on other classification metrics is by the parameter α, the targeted classification error ratio (Figure 2). By decreasing the value of α, the false positive classification ratio may be decreased to very small values (Figure 2(a)), but there is some increase of the time needed for true positive classification (Figure 2(b)). Some value of α may be chosen as a compromise between false classification ratio and time until correct classification. For the range of



(a) (b)

Fig. 2. The AIS main metrics: (a) False classification ratio and (b) time until correct classification of misbehaving nodes versus target false positive classification probability parameter α. Comment: true positive classification ratio is equal to 1, what is not shown on the graphs.

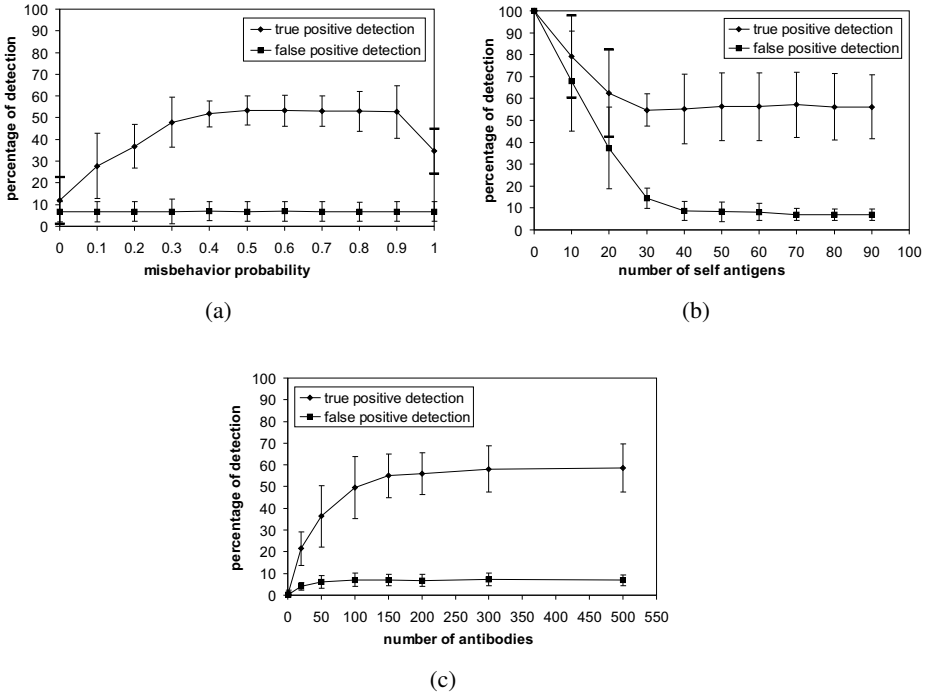


Fig. 3. Impact of misbehavior and parameters tuning: Probability of correct detection of misbehaving nodes (true positive) and misdetection of well behaving nodes (false positive) from a data set collected during an interval $\Delta t = 10$s versus (a) misbehavior probability for the misbehaving node, (b) number of self antigens collected for learning and (c) number of antibodies.

α from 0.05 to 0.03, the values of the false classification ratio are below targeted values, but for the smaller values of α this is not the case (Figure 2(a)). A reason for this may be correlation in detection results in consecutive detection intervals.

Impact of misbehavior and parameters tuning: Effects of some parameters other than α and effect of the misbehavior probability on detection are shown on Figure 3.

Figure 3(a) shows that for a very small probability of misbehavior, distinguishing between good and misbehaving nodes is difficult, but in this case, the impact of misbehavior on the network is also small. If a node misbehaves with the probability equal to or greater than 10 %, it is well distinguished from well behaving nodes, in the sense that it is detected in a percentage that is significantly different than that of well behaving nodes. Even if the percentage of detection is not very high (between 25% and 60 %), this distinguishing allows good classification results (Figure 2). For a very high probability of misbehavior, the percentage of true positive detection is slightly decreased, because the neighbors of the misbehaving node can collect less data about its behavior, as it does not send packets except its own.

Figure 3(b) shows the effect of the number of self antigens collected for learning. If the number of self antigens collected for learning is too small, both self and nonself antigens are mainly detected by antibodies. Antibodies are mainly random and have good coverage of both self and nonself sets of antigens. False positive detection decreases very fast with increasing the number of self antigens used for learning, and it reaches saturation at about 40 self antigens. As an unwanted effect, true positive detection is decreased to the value of about 60 %, because some nonself antigens that are similar to self antigens will not be detected in some cases.

Figure 3(c) shows the impact of the number of antibodies. Increasing this number increases coverage, which increases true positive detection. False positive detection remains small because antibodies are mainly self tolerant. After the saturation of curves is reached, further increasing of the number of antibodies only increases processing cost and does not improve detection. So, as compromise, some optimal value can be chosen.

5 Conclusions

We have designed a method to apply an artificial immune system approach for misbehavior detection in a mobile ad-hoc network. Our simulations show good detection and classification capabilities but, in this early phase, it is premature to draw conclusions about the performance of the AIS approach. We need more experiments to extend our initial work to more misbehavior and traffic patterns. Instead, we would like to focus now on what the experience of designing this first phase tells us for the future.

Mapping IS to AIS The most difficult problem we encountered was the mapping from the IS to the concrete problem of DSR misbehavior detection. We have followed the approaches in [5, 7] but a large number of fundamental issues remain unclear. At the highest level, we still wonder what is the best choice for a target unit to be detected: the node, or sequence of messages, or a message itself. This choice could have a large impact on the design challenges and possible use of the detection system. Even if we stay with the mapping we have designed here, things remain vastly open.

The very definition of **genes** is one of them. We have defined them in an ad-hoc way, trying to guess the definitions that would have the best detection capabilities. We made sure to have at least two correlated genes per misbehavior, in order to capture it efficiently. Indeed, we propose to use the correlation between genes as an important criterion in selecting the genes defined in the antigen structure. In a next phase, we are planning to automate the process of selecting genes. Correlation between genes from an offered set of genes can be computed from experimental data in a normal operation of the network without misbehaving nodes. Good pairs, triples or k-tuples of genes, which score high cross-correlation, can be selected automatically. The final selection of candidate genes would still require to be screened by a human expert intervention, but this would be considerably simpler than designing genes from scratch, as we did. One can observe that such a gene selection process is not part of the natural IS, but one can view it as an accelerated *selection* process. An alternative is to define genes as arbitrary low level bit patterns, and let negative and clonal selection do the job of keeping only the relevant antibodies. This would be more in line with the original motivation for using an AIS. A problem with this approach is that it would require many genes, and the processing effort needed to generate good detectors increases exponentially with

the number of genes. A possible angle of attack is based on the observation that **the IS is also a resource management system**. Indeed, the IS has mechanisms to multiply IS cells and send them to the parts of the body under attack, thus mobilizing resources where and when needed. The analog here would be to use randomization: in a steady state, only a small, random subset of protocol event sequences is used to create antigen. When an attack is on (signalled by a danger signal, see below), more events are analyzed in the regions that are in danger.

Innate System and Danger Signal: The model we implemented in this first step is not able to learn about a changing self, which is not a problem if it is used for a constant DSR protocol and in the case of mobility and traffic patterns similar to those presented during the learning phase. In order to be able to adapt to mobility model and traffic pattern changes, and eventually changes to DSR, the model must be fortified with the additional mechanisms of the danger signals [12]. Danger signals could be defined as network performance indicators (packet loss, delay). In the natural IS, the danger signal is intimately linked to the innate system. Here, the innate system could be mapped to the traditional approach, i.e. a set of pre-defined detection mechanisms as we developed in [4]. It is likely that many new attacks are accompanied with symptoms that are not new. Thus the innate system could be used as a source of the danger signal as well. This would free resources to focus the adaptive immune system on the detection of truly new, unexpected misbehavior.

Clonal Selection and Memory Effects are not implemented in this step, but it will be straightforward to do so. The expected benefit is a better reaction to a misbehavior that repeats itself.

Regulation: Translation of non-self antigen matching to misbehaviour detection is done in a classical, statistical way. This should be compared to the regulation of B-cell and T-cell clonal division [14], which is algorithmically very different.

Parameter Tuning: Even if an appropriate mapping of IS to AIS is found, it remains that the performance is very sensitive to some parameters; the parameters have to be carefully tuned. It is unclear today whether this dependency exists in the IS, and if natural selection takes care of choosing good values, or whether there are inherently stable control mechanisms in the IS that make accurate tuning less important. Understanding this is key to designing not only an AIS as here, but also a large class of controlled systems.

References

1. Sergio Marti, T.J. Giuli, Kevin Lai, and Mary Baker. Mitigating routing misbehavior in mobile ad hoc networks. In *Proceedings of MOBICOM 2000*, pages 255–265, 2000.
2. S. Buchegger and J.-Y. Le Boudec.A Robust Reputation System for Mobile ad hoc Networks. Technical Report, IC/2003/50, EPFL-DI-ICA, Lausanne, Switzerland, July 2003.
3. S. Buchegger and J.-Y. Le Boudec. Performance Analysis of the CONFIDANT protocol: Cooperation of nodes - Fairness In Distributed Ad-Hoc Networks. In *Proceedings of MobiHOC, IEEE/ACM*, Lausanne, CH, June 2002.
4. S. Buchegger and J.-Y. Le Boudec. The Effect of Rumor Spreading in Reputation Systems for Mobile Ad-hoc Networks. In *Proceedings of WiOpt '03: Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks*, Sophia-Antipolis, France, March 2003.

5. S.A. Hofmeyr. An Immunological Model of Distributed Detection and it's Application to Computer Security. *PhD thesis*, Department of Computer Sciences, University of New Mexico, April 1999.
6. S. A Hofmeyr and S. Forrest "Architecture for an Artificial Immune System". Evolutionary Computation 7(1):45-68. 2000.
7. J. Kim and P.J. Bentley. The Artificial Immune Model for Network Intrusion Detection: *7th European Conference on Intelligent Techniques and Soft Computing* (EUFIT'99), Aachen, Germany.
8. J. Kim and P.J. Bentley. Evaluating Negative Selection in an Artificial Immune System for Network Intrusion Detection: *Genetic and Evolutionary Computation Conference 2001* (GECCO-2001), San Francisko, pp. 1330-1337, July 7-11.
9. J. Kim and P.J. Bentley. The Artificial Immune System for Network Intrusion Detection: An Investigation of Clonal Selection with Negative Selection Operator. *The Congres on Evolutionary Computation (CEC-2001)*, Seoul, Korea, pp. 1244-1252, May 27-30.
10. J. Kim and P.J. Bentley. Towards an Artificial Immune System for Network Intrusion Detection: An Investigation of Dynamic Clonal Selection. *The Congress on Evolutionary Computation (CEC-2002)*, Honolulu, pp.1015 - 1020, May 12-17, 2002
11. P. Matzinger. Tolerance, Danger and the Extended Family. *Annual Review of Immunology*, 12:991-1045, 1994.
12. P. Matzinger. The Danger Model in it's Historical Contex. *Scandinavian Journal of Immunology*, 54:4-9, 2001.
13. L.M. Sompayrac. How the Immune System Works, 2nd Edition. Blackwell Publishing, 2003.
14. Tak W. Mak. Order from disorder sprung: recognition and regulation in the immune system. in *Phil. Trans. R. Soc. Lond. A (2003) 361, 1235–1250*, May 2003.
15. Xiang Zeng, Rajive Bagrodia, and Mario Gerla. Glomosim: A library for parallel simulation of large scale wireless networks. *Proceedings of the 12th workshop on Parallel and Distributed Simulations-PDAS'98*, May 26-29, in Banff, Alberta, Canada, 1998.
16. D.B. Johnson and D.A. Maltz. The dynamic source routing protocol for mobile ad hoc networks. *Internet draft, Mobile Ad Hoc Network (MANET) Working Group*, IETF, February 2003.
17. G. Iannaccone C.-N. Chuah, R. Mortier, S. Bhattacharyya, C. Diot. *Analysis of Link Failures in an IP Backbone*. Proceeding of IMW 2002. ACM Press. Marseille, France. November 2002
18. De Castro, L. N. and Von Zuben, F. J. (1999), Artificial Immune Systems: Part I Basic Theory and Application, Technical Report RT DCA 01/99
19. Leandro N. de Castro and Jonathan Timmis, Artificial Immune Systems: A New Computational Intelligence Approach, Springer Verlag, Berlin, 2002
20. Dipankar Dasgupta and Fabio González, An Immunity-Based Technique to Characterize Intrusions in Computer Networks, IEEE Trans. Evol. Comput., (6)9, pp. 1081–1088, June 2002

Appendix 1: Derivation of Equation (1)

We model the outcome of the behavior of a node as a random generator, such that with unknown but fixed probability θ a data set is interpreted as suspicious. We assume the outcome of this fictitious generator is iid. We use a classical hypothesis framework. The null hypothesis is $\theta \leq \theta_{\max}$, i.e., the node behaves well. The maximum likelihood ratio test has a rejection region of the form $\{M_n > K(n)\}$ for some function $K(n)$. The function $K(n)$ is found by the type-I error probability condition: $\mathbb{P}\{M_n > K(n)\}|\theta) \leq \alpha$, for all $\theta \leq \theta_{\max}$, thus the best $K(n)$ is obtained by solving the equation

$$\mathbb{P}(\{M_n > K(n)\}|\theta_{\max}) = \alpha$$

The distribution of M_n is binomial, which is well approximated by a normal distribution with mean $\mu = n\theta$ and variance $n\theta(1-\theta)$. After some algebra this gives $K(n) = \sqrt{n}\xi\sqrt{\theta_{\max}(1-\theta_{\max})} + n\theta_{\max}$, from which Equation (1) derives immediately.

Appendix 2: Detailed detection and classification algorithm

```
phase=LEARNING;
switch(phase){
  case LEARNING{
    phaseTimer=maximalLearningTime;
    numberOfCollectedDataSets=0;collectingDataSetsTimer=0;
    SetOfDetectors=CreateAnEmptySetOfDetectors();numberOfDetectors=0;
    while(phase==LEARNING){
      if(aPacketSentReceivedOrOverheared){
      createOrUpdateNeighborsList();
        if(collectingDataSetsTimer==0 &&
           numberOfCollectedDataSets<maxNumOfCSD){
          openNewCollectingDataSets();collectingDataSetsTimer==deltaT;
        }//end if
        UpdateCurrentDataSetsIfTheyAreOpen();
      }//end if
      if(collectingDataSetsTimer==0){
        closeCurrentCollectingDataSets();
      }//end if
      if(phase!=LEARNING) break;
      if(phaseTimer==0){
        //create detectors by negative selection
        setOfSelfAntigens=createSelfAntigensFromCollectedDataSets();
        while(numberOfDetectors<TargetedNumDet){
          generateANewDetectorByRandom();
          if(isNewDetMatchingAnySelfAntigen()){
            deleteNewDetector();
          }//end if
          else{
            addNewDetectorToSetOfDetectors();numberOfDetectors++;
          }//end if else
        }//end while
        phase=DETECTINGandCLASSIFYING;
      }//end if
    }// end while
    break;
  }//end case
  case DETECTINGandCLASSIFYING{
    while(phase==DETECTINGandCLASSIFYING){
      if(isPacketSentReceivedOrOverheared()){
        createOrUpdateNeighborsList();
        if(collectingDataSetsTimer==0 &&
          numberOfCollectedDataSets<maxNumOfCSD){
          openNewCollectingDataSets();collectingDataSetsTimer==deltaT;
        }//end if
        UpdateCurrentDataSetsIfTheyAreOpen();
      }//end if
      if(phase!=DETECTINGandCLASSIFYING) break;
      if(collectingDataSetsTimer==0){
        //do detection and classification
        createAntigensFromCurrentDataSets();
        deleteCurrentDataSets();
        checkAntigensFromLastDeltaTByDetectors();
        updateListOfDetectedNodes();
        updateDetectionResultsForObservedNodes();
        deleteAntigensFromLastDeltaT();
        checkDetectionResultsForClassification();
      }//end if
    }//end while
    break;
  }//end case
}//end switch
```