Performance of Security Mechanisms in Wireless Ad Hoc Networks

Matthias Becker, Martin Drozda and Sven Schaust
FG Simulation und Modellierung, Institute of Systems Engineering, G. W. Leibniz University of Hannover
Welfengarten 1, 30167 Hannover, Germany.
Email: {xmb,drozda,svs}@sim.uni-hannover.de

***Abstract*— Adding security mechanisms to computer and communications systems without degrading their performance is a difficult task. This holds especially for wireless ad hoc networks that, due to their physical and logical openness, are easier to attack than wired networks. Additionally, such networks are expected to have less resources in terms of computational capabilities and must also rely on batteries for power supply.**

We investigate the impact of two misbehavior detection mechanisms based on Neural Networks and Artificial Immune Systems. In the performance analysis we assume that wireless devices, in many scenarios, must work with extremely limited computational resources. An example of such a scenario are sensor networks. This implicates that the security system of choice has to be very efficient in order not to disturb the normal wireless device operation.

***Index Terms*—Artificial Immune Systems, Neural Networks, Sensor Networks, Misbehavior Detection.**

I. Introduction and Motivation

Adding security mechanism to computer and communication systems results in their decreased performance. The decrease can even result in a complete system failure caused by the security mechanism, e.g. an email server might be forced to stop some services because of its spam filter exhausting the processor.

Since security mechanisms can have an impact on the quality of service of stationary systems with huge computational and power resources, the effect can be much more severe in computationally limited and energy constrained systems.

In this work we consider wireless sensor networks, which are a specific kind of ad hoc wireless networks. Ad hoc networks have no fixed infrastructure (as opposed to wireless clients that connect to a fixed base station). Typical devices employed in an ad hoc network are laptops, PDAs, mobile phones. A sensor network is a collection of small wireless devices, or sensors, that are able to monitor the deployment area. The sensors then transmit all measured information to a collection station (sink). Such measured information is, in general, forwarded by intermediate sensor that lie between the information acquisition sensor and the collection station. Sensor networks are expected to find applications in many engineering as well as military or rescue scenarios.

A typical security problem in sensor networks is to detect and isolate misbehaving nodes. The misbehavior might be a malfunction due to the damage of a sensor node, or a malicious attempt to disturb or misuse the functions of the sensor network. The nature of a misbehavior is rather unforeseeable, thus an intelligent approach is needed that would also respond to new kinds of attacks. This is in contrast to employing hard-wired attack signatures that do not offer the flexibility needed. Additionally, there is usually not enough memory on sensor nodes to include any attack signature database (e.g. virus signatures).

Approaches for misbehavior detection in wireless ad hoc networks based on Artificial Immune Systems (AIS) have been presented in [9], [14] and with focus on sensor networks in [22]. An approach based on Neural Networks (NN) can be found in [23].

Both approaches have been compared in [16]. NN have been successfully used in pattern matching and for detection in various fields, among them also detection of user anomaly [18], [19] and detection of network intrusion [17]. There exist also flavors of NN where their training continues during operation which should be valuable for highly dynamic systems such as ad hoc networks.

In [23], NN have been evaluated for misbehavior detection in sensor networks and a comparison to AIS with respect to the detection quality has been done.

In this work, the scenarios used in [23], [22] and [24] will be the base for this study of the impact of the security mechanisms (AIS and NN) on system performance. We will study the computational costs of the learning phase as well as the computational resources needed in the detection phase, and how both security mechanisms affect the performance of a sensor node.

This document is structured as follows. First, we will shortly review the mechanism of AIS and NN. Second, an outline of the scenario will be given and third, the detection performance of AIS and NN as well as the computational resources needed for the detection will be discussed.

II. Artificial Immune Systems

The Human Immune System is able to distinguish between what is usually is part of the human body (self antigen), and everything else (non-self antigen), which is usually some kind of foreign agent, be it something mechanical (splinter etc.) or a disease (virus, bacteria).

Inspired by the Human Immune System, Artificial Immune Systems have been developed and have already been considered to be a viable means for misbehavior detection in com-

Proceedings of the 2008 High Performance
Computing & Simulation Conference ©ECMS
Waleed W. Smari (Ed.)
ISBN: 978-0-9553018-7-2 / ISBN: 978-0-9553018-6-5 (CD)

munication networks, and especially in flexible environments such as ad hoc and sensor networks.

An Artificial Immune System basically can be divided into two parts, namely a learning phase and a testing phase. In the learning phase the system uses a so called self-set, which represents the expected behavior, to create a set of anomaly detectors. During the testing (or detection) phase, this detector set will be used to detect anomalies. It is necessary that any detector is unable to match a normal behavior. Otherwise *false positives* may occur. A false positive is a detected anomaly which is caused by an incomplete self-set during the learning phase. We will discuss the process of learning in more details in the following subsection. The testing phase is the phase in which the detection actually happens. For an observed time window the system measures different network and traffic features for the neighborhood of a single node and computes an antigen. Each computed antigen is then matched against the complete detector set of the particular single node.

A misbehavior detection system for ad-hoc wireless networks based on AIS has been introduced in [14], [15]. The authors also suggest to use a co-stimulation for misbehavior classification in the form of a danger signal. The function of the danger signal is to inform nodes on a forwarding path about deterioration in terms of a quality of service (QoS) measure. The signal is sent from the connection source to the connection sink, thus propagating QoS information along the connection route. Danger signals were introduced in [3].

One of the earlier proposals of AIS for misbehavior detection was given in [9]. The authors describe an AIS able to detect anomalies at the transport layer of the OSI protocol stack, considering only TCP connections in a stationary wired network. They define the normal network behavior (self) as normal pairwise TCP connections. Each detector is represented as a 49-bit string. The pattern matching is based on r-contiguous bits with a fixed $r = 12$.

In [12] the authors discuss 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 problem occurs because of the specific choice of problem representation and also due to the choice of matching threshold r for r-contiguous bit matching.

In [22] an AIS approach, measuring network features at the link and network layer of the OSI stack, has been studied for sensor networks. The authors use a fixed ad hoc sensor network scenario with approx. 1700 nodes, of which 236 are misbehaving, and 10 concurrent connections with a constant bit rate traffic. The implemented misbehavior in their experiments is probabilistic packet dropping. They conclude that their AIS is capable of detecting such a misbehavior in the described environment with about 80% accuracy.

A. Learning Mechanism of Artificial Immune Systems

The process of T-cells maturation and selection in the thymus is often used as an inspiration for learning in AIS. In AIS, the characteristics of self and non-self can be represented as bit-strings. The role of detectors is to detect non-self antigen. A popular algorithm for matching detectors and non-self antigen, if represented as bit-strings, 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.

In order to generate a set of working detectors, pseudo random detectors are created and tested with self. Only if a detector does not match any self-antigen, the detector is added to the detector set. This generate-and-test approach (*negative selection*) described above is analyzed in [6]. They assume that both self and non-self sets, as well as detectors can be modeled as bit-strings of length l. Let the size of the self set be N_S, the probability that a randomly chosen detector and a string from the self set match be P_m and the probability that a string from the non-self set is not matched by any detector be P_f. Then the time and space complexity of this algorithm for a fixed matching probability P_m is $O(\frac{-ln(P_f)}{P_m(1-P_m)^{N_S}}N_S)$ and $O(lN_S)$, respectively. This algorithm requires that the number of required candidate detectors is exponential to N_S. The advantage of this algorithm is its simplicity and good experimental results in cases when the number of detectors to be produced is fixed and small [14]. A review of other approaches to detector computation can be found in [2].

III. Neural Networks

Since Neural Networks and the learning via back-propagation algorithm are well known, we only shortly review the way Neural Networks work here. Details can be found in standard textbooks such as [20].

A. Sketch of NN Algorithm

NN learn a function from an input vector to an output vector. In the so called learning phase input vectors are presented to the NN and an output vector is calculated. Depending on whether the output is correct or not, the weights in the NN are corrected using the back-propagation algorithm in order to achieve the right output. These procedure is repeated until for all input vectors, the right output is calculated (within some epsilon error interval).

The data is divided in a training set and a test set. In the learning phase, only the training set is used. The test set is used in order to evaluate the ability of the NN for *prediction* of unknown input.

Without these two sets there is the danger of creating a NN that works perfect on the set used for training (100 percent correct classification of input vectors), but fails when unknown input is presented. A similar behavior is often caused by so called over-fitting, see e.g. [21] for a mathematical analysis of the problem.

IV. Sensor Networks

The sensor network simulation scenario used here is described in detail in [22], so we only give a sketch of the scenario here.

The sensor network consists of 1718 nodes with a radio radius of 100m. The distribution of the nodes over a square area of 2,900m×2,950m is a snapshot of the nodes moving by the random waypoint mobility model. The motivation in using this movement model and then creating a snapshot are based on results on structural robustness of sensor networks [5]. The traffic in the network is generated by 10 CBR (Constant Bit Rate) connections. The connections were chosen so that their length is ∼7 hops and so that these connections share some common intermediate nodes. For each packet received or sent by a node the following information has been captured: IP header type (UDP, 802.11 or DSR [10] 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. We refer the reader to [11] for more information on sensor networks. 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. Glomosim [7] was used as simulator, with 4-hours of simulated traffic. We choose to collect traffic information within 28 non-overlapping 500-second windows in our simulation. In each 500-second window self and non-self antigens were computed for each node. The experiment was repeated 20 times with independent Glomosim runs using different random seeds.

A. Encoding of genes

From the captured simulation data the following measures have been calculated and then coded into genes:

MAC Layer:

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

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

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

Routing Layer:

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

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

Encoding of self and non-self antigens was done as follows. Each gene value was transformed in a 10-bit signature where each bit defines an interval[1] of a gene specific value range. We created self and non-self antigen strings by concatenation of the defined genes. Each self and non-self antigen has therefore a size of 50 bits. The interval representation was chosen in order to avoid carry-bits that make the binary representation less compact.

Simulation runs were done for one of $\{10, 30, 50\%\}$ misbehavior levels (packet dropping) and "normal" traffic with no misbehavior, so that 'self' could be learned from the normal behavior and then the quality of the misbehavior detection could be evaluated either with NN or AIS, using the genes extracted for the misbehaving nodes for the simulation runs with misbehavior.

V. Training and Detection Quality of AIS

As described in section II-A, for each node in the network an AIS has been set up with detectors obtained via the negative selection algorithm using the genes that encode the traffic characteristics met around this node during normal network operation. Only data from nodes with enough traffic has been used. The experiments in [22] showed that the detection rate is rather independent of the packet thresholds. Packet threshold of e.g. 500 means that a node had at least 500 packets to forward in both the learning and misbehavior (test) phases; this number is measured over the whole 4-hour simulation period. Except for some extremely low threshold values the detection rate stays constant.

Then the genes from the sensor network that includes misbehaving nodes have been presented to the AIS of the single nodes for misbehavior detection. A node is defined to be detected as misbehaving, if it gets flagged in at least 14 out of 28 possible windows. This definition is equivalent (under reasonable assumptions) to saying that the time to detection is double the size of the window, i.e. 1000 seconds in this case. In [22] it is shown that if the misbehavior level is set very low, i.e. 10%, the AIS usually struggles to detect misbehaving nodes, because the traffic pattern of misbehavior is not distinct enough from the noise appearing in normal traffic, where also packets are dropped sometimes. At the 30 and 50% misbehaving levels the detection rate stays solid at about 70-85%.

Beyond the pure detection abilities, in this work we have a closer look at how parameters related to the computational complexity of the AIS algorithm influence the detection rate. Figure 1 shows the impact of r on detection rate. When $r = \{7, 10\}$ the AIS performs well, for $r > 10$ the detection rate decreases. This is caused by the inadequate numbers of detectors used; in general the number of detectors should be doubled when r is increased by one.

Besides r, the number of detectors is the crucial parameter regarding the detection rate and especially computational

[1]The interval encoding of genes is adapted from [14]. 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.

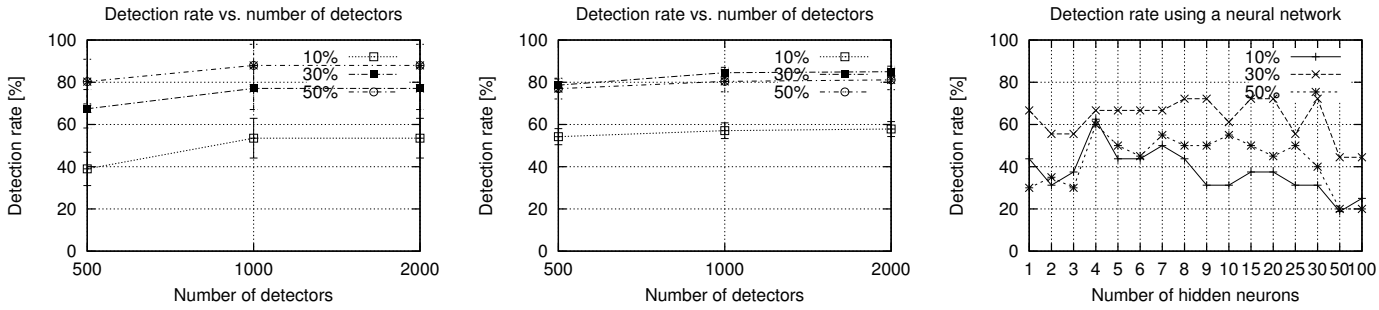


Fig. 2. Detection rate vs (a) number of detectors using 10 connections, (b) number of detectors using 50 connections, (c) the mean number of hidden neurons. All scenarios use a Poisson based traffic model. Packet threshold was 1000 packets.

complexity. Figures 2 (a) and (b) show that for the actual problem, less than 1000 detectors are not sufficient for a good detection rate, 2000 detectors show best success, while more detectors do not improve the detection rate anymore and only increase computational complexity and memory requirements. The figures are based on two scenarios using Poisson traffic [24], one having 10 connections the other 50 connections. We used the same packet threshold value of 1000 packets for both scenarios in order to verify whether a node qualifies for detection or not.

VI. Training and Detection Quality of NN

Back-propagation networks with three layers using the FANN library [8] have been deployed here for misbehavior detection. We used the same representation of good/bad behavior, i.e. a binary vector of length 50. Therefore the Neural Networks had 50 input neurons and only one output neuron (zero indicating good, one indicating bad behavior). The data from the simulation experiments has been divided into a training and test set (approx. $\frac{2}{3}$ training, $\frac{1}{3}$ test). Only data from experiments/nodes has been used, where enough data was present (a bad node cannot develop a bad node pattern, if he is outside the traffic), and where the data was unambiguous.

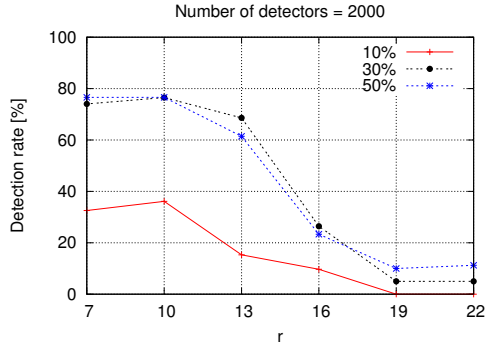


Fig. 1. Detection rate vs r-value of the matching algorithm. Packet threshold was 1000 packets.

The main parameters of a Neural Network is the number of hidden layers and the number of neurons in the hidden layers. For the problem complexity considered here, one hidden layer is appropriate. The question is, how many neurons the hidden layer should have, in order to be able to distinguish good and bad behavior, but also avoiding over-fitting (when a Neural Network learns the data too exactly, it is not able to succeed on new, unknown data any more, i.e. the ability to abstract/predict is lost). Moreover the number of neurons in the hidden layer is the only free parameter in the Neural Network that has a direct effect on the computational complexity, since the number of input and output neurons are predetermined by the input vectors (genes) and the number of different outcomes. Thus the detection rates have been evaluated for an increasing number of hidden neurons.

A. Experiments

Since each training of a Neural Net is an individual process (depending in the random initial weights and the presentation order of the vectors during the learning phase) and results in a different Neural Net finally, several experiments have to be made in order to get a valid result with mean outcome and mean deviation. Thus for each configuration (consisting of a certain number of hidden neurons and a data set) 20 Neural Nets have been trained and evaluated, figure 2 (c) thus showing mean values. The variance of the exact value of the output neuron was very low so that the classification resulting from the exact value of the output neuron (good/bad) showed no variance any more. Figure 2 (c) shows how many percent of the **test** vectors have been identified correctly as good or bad. This shows the prediction power of the Neural Net, that is the correct response to **new** attacks. The training set has been learned almost perfectly (Training stopped when over 95% of the training set had been learned correctly).

If, for comparability with the AIS approach, the training set would also be considered, then the detection rate would be around 90 percent. Note that the training data is recognized correctly with 100 percent.

Regarding the number of hidden neurons (representing the complexity of the Neural Net) we observe, that a medium

number of hidden neurons (between four and nine) shows best overall success. This corresponds to the experience, that a certain number of hidden neurons is necessary for the correct analysis of the data, but on the other side, if there are too many hidden neurons, the net overfits the data. That means the training set is recognized correctly (100 percent), but the net does not have the ability to judge about unknown data, since it has just learned exactly the training set. As could be expected, a low level of misbehavior makes it harder recognize the difference between good and bad behavior.

VII. Complexity and Performance of NN and AIS Algorithm

Comparing the abilities of both AIS and NN in terms of detection rate for unknown misbehavior, which is between 40 and nearly 80% depending on the level of misbehavior and other parameters, it can be said that both algorithms perform well fulfilling this task. Both algorithms are suited for misbehavior detection in sensor networks.

For the design of a Neural Network, the number of hidden neurons is the key parameter. We showed that not many hidden neurons are required for a decent detection rate.

The performance concerning detection rate of NN and AIS is comparable. Nevertheless both methods are different in some key aspects when thinking of an application for a real sensor network. An AIS (with no danger signals utilized) can be only trained using the normal operation traffic. A Neural Network however needs also bad behavior to be represented in its training phase. Otherwise the Neural Network might learn, that it should always output 'self' constantly, disregarding the input. A solution to the described problem would be to generate non-self patterns, either by injecting errors in the sensor network or by generating artificial 'non-self' training vectors.

A. Computational Requirements of AIS and NN

Once the AIS and NN are trained, an AIS needs nearly six times more memory resources for its task than a NN (see [25], [26] for complexity issues in NN). In our setup, good detection performance was achieved with $d = 2,000$, where d is the desired number of detectors, and with $h \leq 9$, where h is the number of hidden neurons. Within this setup (bit-string length of 50 bits), the memory needed for AIS is approx. $50 \times d = 100$ kbits $= 12,500$ Bytes. A NN is represented by a weight matrix of size $50 \times h + h \times 1 = 459$ weight values, for $h = 9$. When the weights are coded as `float` data types of size 4 Bytes, the memory requirements is $1,800$ Bytes. We remind the reader, Crossbow Mica2 sensors [1] have 4kB RAM and 512kB EEPROM data memory available; these resources are however shared among all applications and the operating system.

Figure 3 (a) shows the memory requirements in Kilobytes for a given problem size. The problem size is normalized to the case discussed above, that is problem size of one means $h = 9$ number of hidden neurons in the Neural Network case, and $d = 2000$ detectors in the AIS case. It can be seen that for larger problem sizes, the memory requirements for AIS grow to a size which might not be suitable for small sensors anymore, while the memory requirements stay moderate even for a large problem size in the NN case.

The computational requirements for detection in AIS is in worst case $d \times r \times (50 - r)$ operations (bit comparisons) (r being the value from the r-contiguous bits matching). That is $800,000$ comparisons for $r = 10$. The reason is, when an antigen is constructed, there are possibly up to d comparisons with the detectors necessary. The detection in NN can be measured in multiplication and additions: $50 \times 10 + 10 \times 1 = 510$ multiplications plus approx. other 510 additions.

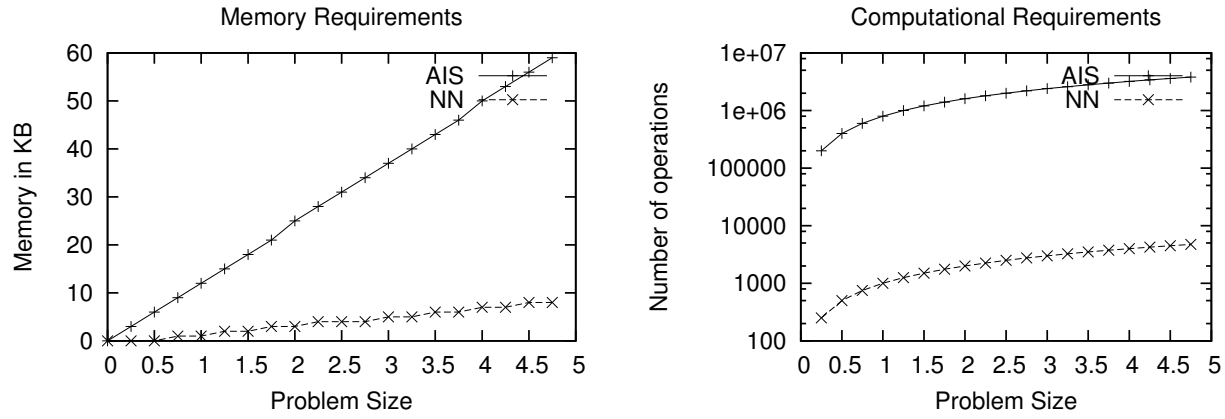
Figure 3 (b) shows the growing number of operations with higher problems size (normalized to the problem above, with constant $r = 10$ in the AIS case). The operations in the AIS are mainly bit comparisons, while in the NN the operations are multiplications and additions. Note that we chose a logarithmic scale on y-axis, since the operational requirements differ by about three orders of magnitude. Since the computational requirements are also proportional to the energy consumption this fact clearly indicates that NN have an advantage in energy constrained sensor networks.

VIII. Conclusions and Future Work

We compared performance and complexity of two approaches known from computational intelligence, when used as a means for misbehavior detection in wireless sensor networks.

We conclude that they meet the memory requirements of to-date sensor platforms which have a memory in the order of hundreds of kilobytes, thus a Neural Net as described above would need less than 1 percent of the memory. Also the computational effort of doing a few hundred operations (additions, multiplications or comparisons) is negligible on sensors with a processor-speed in the order of MHz.

The results in this work suggest that Neural Networks are better suited for sensor networks with restricted resources, because NN use less memory and need far less operations for the detection. While the need of memory of both NN and AIS is negligible compared to the memory of actual sensors, the issue of operations in the detection phase is more crucial. NN need approximately three orders of magnitude less operations for a classification. This means that each detection in an AIS needs much more time, slowing down the operation of the sensor network. While this slow down might be negligible in sensor networks whose operation is usually not time critical, the resulting bigger energy consumption of AIS is more important, and might tip the scale in favor of NN. However both approaches are different regarding the length of the preprocessing phase, memory requirements, speed of computation and the rate of false positives. Both approaches are suitable for misbehavior detection in sensor networks, the decision which approach to choose for a specific sensor network depends on the details of the scenario.



(a) (b)

Fig. 3. (a) Memory requirements vs problem size (normalized, problem size $1 \sim h = 9$, $d = 2000$), (b) Number of operations vs problem size (normalized, problem size $1 \sim h = 9$, $d = 2000$)

ACKNOWLEDGMENTS

This work was supported by the German Research Foundation (DFG) under the grant no. SZ 51/24-2 (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)*, 2004.

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

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

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

[6] 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.

[7] L. Bajaj, M. Takai, R. Ahuja, K. Tang, R. Bagrodia, and M. Gerla. GloMoSim: A Scalable Network Simulation Environment. UCLA Computer Science Department Technical Report 990027, May 1999.

[8] FANN Library. http://leenissen.dk/fann/

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

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

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

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

[13] 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*, 2000.

[14] 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)*, 2004.

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

[16] D. Dasgupta. Artificial Neural Networks and Artificial Immune Systems: Similarities and Differences. *Proc. IEEE Systems, Man, and Cybernetics Conference (SMC)*, pp. 873-878, 1997.

[17] M. Moradi, M. Zulkernine. A Neural Network Based System for Intrusion Detection and Classification of Attacks. *Proc. 2004 IEEE International Conference on Advances in Intelligent Systems-Theory and Applications*, 2004.

[18] J. Ryan, M.J. Lin, R. Miikkulainen. Intrusion Detection with Neural Networks. *Advances in Neural Information Processing Systems*, vol.10, pp. 943–949, MIT Press, 1998.

[19] H. Debar, M. Becker, D. Siboni. A Neural network component for an intrusion detection system. *Proc. IEEE Computer Society Symposium on Research in Security and Privacy*, pp. 240-250, 1992.

[20] S. Haykin. *Neural networks: a comprehensive foundation.* Prentice Hall, 1999.

[21] A.B. Owen. Overfitting in Neural networks. In *Computing Science and Statistics. Proc. 26th Symposium on the Interface*, pp. 57–62, Interface Foundation of North America, 1994.

[22] M. Drozda, S. Schaust, H. Szczerbicka. Is AIS Based Misbehaviour Detection Suitable for Wireless Networks? *Proc. IEEE Wireless Communications and Networking Conference (WCNC '07)*, pp. 3130-3135, Hong Kong, 2007.

[23] M. Becker, M. Drozda, S. Jaschke, S. Schaust. Comparing Performance of Misbehavior Detection Based on Neural Networks and AIS. IEEE Systems, Man, and Cybernetics Conference (SMC'08), Singapore, submitted.

[24] S. Schaust, M. Drozda, H. Szczerbicka. Impact of Packet Injection Models on Misbehaviour Detection Performance in Wireless Sensor Networks. *Proc. 3rd IEEE International Workshop on Wireless and Sensor Networks Security (WSNS)*, 2007.

[25] A. Engel. Complexity of learning in artificial Neural networks. *Theor. Comput. Sci.*, vol. 265, no. 1, pp. 285-306, 2001.

[26] M. Roisenberg, J.M. Barreto, F.M. De Azevedo. Neural network complexity classification based on the problem. *Proc. IJCNN - IEEE International Joint Conference on Neural Networks*, vol.3, pp. 2413-2418, 1998.