AIS-Lib: An AIS Library for Ad Hoc and Sensor Wireless Networks

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

***Abstract*— We present and discuss the AIS Library (AIS-Lib) that is based on the Jist/SWANS network simulation platform. It implements basic features of Artificial immune systems (AIS). AIS are one of the most recent approaches in computational intelligence; they are motivated by the efficiency of the Human immune system.**

I. Introduction and Motivation

Ad hoc wireless networks do not need any infrastructure in the form of base stations or wireline to operate. Data packets are forwarded by intermediate wireless devices until the destination is reached. Wireless devices in this setting are expected to be small and therefore battery powered.

Sensor networks are a specialized flavor of ad hoc wireless networks. A sensor network is understood to be a collection of small wireless devices (sensors) that are able to monitor environmental or physical conditions such as humidity, temperature, motion or noise. These sensors are suitably spatially distributed in the monitored area.

An ad hoc or sensor network can be subject to user misbehavior or malicious attacks. A non-exhaustive list of different types of misbehavior includes: (data) packet dropping, selective forwarding of routing control packets, deliberate skewing of network's topology, creating multiple identities of nodes,[1] or simply any behavior that could negatively impact the network's overall performance and/or lead to some undeserved advantage for one of the users; see [1] for a more complete list. In addition to user misbehavior, ad hoc and sensor networks can be subject to hardware or software failures. Such a failure can have similar or identical effects as misbehavior.

Since computational power and battery life of wireless devices in an ad hoc network are, in many scenarios, expected to be extremely limited, there has been an on-going interest in providing them with a protection solution that would fulfill several basic criteria. The first criterion is the ability of self-learning and self-tuning. Because maintenance of ad hoc networks by a human operator is expected to be very sporadic, they have to have a built-in *autonomous* mechanism for identifying user behavior that could be potentially damaging to them. This learning mechanism should itself minimize the need for a human intervention, therefore it should be self-tuning to the maximum extent. It must also be computationally extremely conservative and meet the usual condition of high detection rate. The second criterion is the ability to undertake an action against one or several misbehaving users. This should be understood in a wider context of co-operating wireless devices acting in collusion [2], [3] in order to suppress or minimize the adverse impact of such misbehavior. Such a co-operation should have a low message complexity because both the bandwidth and the battery life are of scarce nature. The third and last criterion requires that the newly introduced protection system does not itself introduce new weaknesses to the systems that it should protect.

The general goal of a protection system meeting the above criteria should be to impose a certain degree of survivability [4] on an ad hoc network. The degree heavily depends on the mission of a given ad hoc network. This immediately implies that the mission of the ad hoc or sensor network must be well defined since it is hard to expect that such a protection system will have the flexibility of human operated systems as we know them from wired networks (spam, virus and intrusion detection systems).

Artificial immune systems (AIS) [5] are an emerging approach for detecting anomaly in ad hoc networks. They are based on the well-known properties of the Human immune system (HIS). The reason behind a recent increased interest in AIS is the *simplicity* of the underlying mechanisms; this makes it an ideal solution for technical systems that need to be protected to some extent and that operate with extremely limited resources.

Considering the severe *implementation and scaling* problems of our previous experiments that integrate sensor networks with AIS [6], [7], we felt a need for a general purpose AIS library aimed at ad hoc and sensor networks. Our goal was to allow for a fast and efficient scenario prototyping. Since, to our best knowledge, there is no such library available, we decided to write one. It works under SWANS [8], a Java based network simulator.

II. Medium Access in Ad Hoc and Sensor Networks

We shortly review some well-known properties of MAC protocols that are often used for wireless medium resolution in ad hoc and sensor networks.

The medium reservation is often contention based. In order to transmit a data packet, the IEEE 802.11 MAC protocol uses carrier sensing with an RTS-CTS-DATA-ACK handshake.[2] Should the medium not be available or the handshake fails, an exponential back-off algorithm is used. This is combined with a mechanism that makes it easier for neighboring nodes to estimate transmission durations. With the goal to save battery power, researchers suggested, a sleep-wake-up schedule for nodes would be appropriate. This means that nodes do not

[1]We use the term *node*, when a wireless device and/or a sensor is meant.

[2]RTS = Ready to send, CTS = Clear to send, ACK = Acknowledgment.

listen continuously to the medium, but switch themselves off and wake up again after a predetermined period of time. Such a sleep and wake-up schedule is similarly to duration values exchanged among nodes. An example of a MAC protocol, designed specifically for sensor networks, that uses such a schedule is the S-MAC [9]. A sleep and wake-up schedule can severely limit operation of a node in *promiscuous mode*. In promiscuous mode, a node listens to the on-going traffic in the neighborhood and collects information from the overheard packets.

III. Artificial Immune Systems

A. The Human Immune System

The HIS [5] is able to protect humans against an amazing set of pathogens. Pathogens are cells that are non-self to the body: bacteria, viruses or parasites. The HIS is non-reactive with respect to other cells, most notable to cells that are self to the body e.g. blood cells.

B-cells and T-cells are created in lymphoid organs: B-cells in the bone marrow, T-cells in the thymus. The thymus is protected by a blood-thymic barrier that is able to keep this organ pathogen-free. This is an important property as T-cells are created through a pseudo-random process called *negative selection*. Within this process, it is tested whether a T-cell can recognize a self cell. If this is true, such a T-cell is destroyed, allowing only T-cells that were not able to match any self cell to be released into the body.

After pathogens enter the body, many of them get processed or killed by phagocytes (eating cells). Such cells are able to display pathogen fragments on their surface, which can be recognized by T-cells. Unlike T-cells, B-cells are able to recognize pathogens without the extra help of a displayed pathogen fragment. If a recognition happens, the B-cell gives rise to many plasma cells producing *antibodies*.

T- and B-cells that prove to be especially useful in dealing with pathogens become mature. Those T- and B-cells recognizing a pathogen can be subject to increased division (cloning). While cloning of a T-cell creates an exact copy of the cell, cloning of B-cells allows for some “error”. This is believed to be inversely proportional to the matching ability of a given B-cell. T- and B-cells that were not able to recognize any pathogen are subject to death by neglect (apoptosis). This happens to the most T-cells and is an instance of *positive selection*; see [10].

B. AIS for Ad Hoc and Sensor Networks

An important design issue is translation of the HIS functionality to AIS. The basic question is whether the HIS should be mapped to a single wireless device or to the whole network. This means that either each wireless device has to mimic the creation of B-, T-cells, negative/positive selection etc. or these tasks get distributed over the whole network. Our treatment assumes the former approach; we follow the architecture presented in the seminal work by Hofmeyr and Forrest [11] and other works motivated by it; see [6], [12].

The process of T-cells priming in thymus and their subsequent maturing is used as an inspiration for learning in AIS. T-cells (detectors) are frequently represented as bit-strings; other liked option are real-valued vectors [5]. Popular matching rules that mimic the ability of T-cells to match self or non-self are Hamming distance and r-contiguous bits matching rule; see [2] for a review of other applicable matching rules and their in-depth comparison. 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.

The goal of negative selection is in the case of AIS to produce *detectors* that are able to identify behavior that is unusual or directly damaging to the network. For this purpose it is necessary to observe the network for some period of time and decide what constitutes the “normal” (usual) behavior. This normal behavior then gets represented as bit-strings; usually there is one bit-string computed per window, where a window is a shorter period of time in which it is expected that some or many features of the underlying network become observable. A set of bit-strings that encode normal behavior is called the set of self strings. Then, a random bit-string gets generated and is compared against the set of self strings. If the randomly generated bit-string matches anything in the set of self strings, it is deleted. Otherwise, the random bit-string becomes a detector.

Testing a network on unusual behavior is similar. In each window, a bit-string, that encodes observable behavior of the network, gets created. This bit-string is matched against the set of detectors. If a match exists, then a node (or a group of nodes) has been tested positive on previously unseen behavior. Is is up to the designer of the AIS to decide, whether after a match an action should be undertaken, or whether some statistical analysis on the positive tests will be undertaken. The pros and cons of the above approach are as follows. The number of candidate detectors that must be tested against the set of self string is, in general, exponential to the size of the set of self strings [13]. However, there have been very positive results reported when the desired number of detectors has been made small and fixed; see [6], [7], [11]. Another problem is the need for a “misbehavior-free” period of time during which the set of self strings is computed. The *danger signal* theory [14] acknowledges that misbehavior that causes no damage is not dangerous. Damage in our setting would be increased data packet loss rate, increased end-to-end delay or, in general, deterioration in terms of any of the key global performance measures. Therefore, it has been suggested that as long as there is no danger observable in the network, the network can be assumed to be misbehavior-free. In [12] a simple form of danger signal has been used. A source of any TCP connection observes whether data packets get correctly and timely acknowledged.[3] If not, a danger signal is sent downstream the routing path. Since all the nodes in the network execute the negative selection process, they are able to correlate the danger signal with a detected anomaly.

Once detectors get produced, it is not guaranteed that they will ever be useful for identifying an anomaly. In [11] there has been a notion of usefulness attached to detectors. Those that prove to be useful would become *memory detectors*; remaining ones would be deleted and substituted by fresh ones coming from the negative selection process.

To apply the above described learning process, it is necessary that each node observes and evaluates data and control

[3]Misbehavior was in their experiments modeled as probabilistic data packet dropping.

traffic that he forwards or that he overhears in the neighborhood. This traffic can be characterized by performance measures; a popular term in the AIS community for such a performance measure is *gene*. It is naturally important that genes are easy to compute locally. Each used gene is then converted into a bit-string. These bit-strings get then concatenated and become basis for the negative selection process.

A key to the performance of AIS is the choice of correct genes; see [3], [6], [12], [15] for ideas on genes for ad hoc and sensor networks. 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 $s_s \neq s_d$. Consider the following two genes:

1) 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. It needs some adjustment, when s_{i+1} is a connection destination node.
2) 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}.

The first gene was adapted from the watchdog idea presented by Marti et al. in [16]. It requires that the node s_i operates in promiscuous mode. Promiscuous mode (indirectly) requires that the node s_i stays on most of the time. This could, however, be very power demanding, especially for battery powered sensor networks. On the other hand, this gene is very efficient when the misbehavior of nodes is an instance of data packet dropping (we assume omnidirectional antennas).

The second gene [7] indirectly measures the level of medium contention around the node s_i. If there are many incomplete handshakes, this signals a severe competition for the medium. Any data packet dropping in its nature decreases the medium contention. Less data packets to forward implies less attempted MAC layer handshakes. The advantage of this gene is that a node can get occasionally switched on and off as it is assumed by e.g. the S-MAC protocol. The interesting fact is its generality. It can be applied when a misbehavior influences the medium contention resolution. Therefore, we assume, this or similar gene can be used against a wide range of misbehaviors from packet dropping to wormholes [17].

As the above two genes suggest, in order to compute a gene, the node has to first record its own events, observable events of its neighbors and often also their casual relationships. An implementation of this recording process is straightforward, but usually very *time consuming*.

IV. AIS-Lib

A. Basic Features

AIS-Lib is based on SWANS [8], a Java based network simulation tool. AIS-Lib is an add-on library to SWANS. We chose to use SWANS due to its high simulation events processing throughput and low memory requirements. The other simulation tools that we considered were Glomosim and ns2. We opted not to use Glomosim due to the lack of the necessary documentation and also because its development ended in 2002 with the introduction of its commercial version, Qualnet. We decided not to use ns2 as it, in our opinion, performs worse than Glomosim. Our previous AIS studies [6], [7] were based on Glomosim.When implementing the library, our goal was to offer the user the following features:

1) An easy access to events that happen at a node or in the observable neighborhood of the node. This resulted in a data structure attached to each node. This data structure allows for an easy acquisition of many observable events that are necessary for a straightforward implementation of genes
2) Efficient creation, testing and deletion of genes. This is important for the ease of testing as most of our effort, when still working with Glomosim, was devoted to finding specialized solutions to genes' implementation. The genes have an attached window size, i.e. a time period over which observable events get summed up and averaged. The currently implemented genes are those introduced in [6]. However, it was not our goal to implement a multitude of different genes, but to offer the user an easy way to implement any gene that he considers to be worth of investigation.
3) Support for the negative selection with bit-strings of fixed length. Our design goal was to allow for several instances of the negative selection at a node. Each of these instances can work with a different set of genes and/or be based on observable events at different neighboring nodes. Computed detectors get stored in a data file for use in possible later simulation runs.
4) Support for bit-strings with three choices of encoding: Gray coding, interval coding [6], [12] and conventional binary coding.
5) Support for bit-string matching with the r-contiguous bits matching rule and the Hamming distance.
6) Support for several basic types of misbehavior. Currently, packet dropping with probability p and wormholes are implemented. The packet dropping rate can be different for different types of data and control packets.
7) Additionally, we included support for an easy tracking of relative utility of genes and detectors, and data packet forwarding activity at nodes. The user can thus easily compute the related detection rate, rate of false positives or similar statistics. It was not our goal to compute e.g. the detection rate directly, rather we decided to provide the user with plenty of output statistics from which any measure can be computed. The reason was to give the user the freedom to define e.g. the formula for detection rate to his liking.

B. Modular Architecture

The SWANS simulator uses independent components which can be combined to create different kinds of network simulations. SWANS offers components for applications, networking, routing and media access, radio transmission, reception and noise models, signal propagation and fading models, and node mobility models.

The AIS-Lib was designed as SWANS-application with access to different layers of the OSI protocol stack (most important are the access to network and link layer for monitoring and access to the transport layer for misbehavior influence). This architecture offers high flexibility to be used for different kinds of artificial immune system approaches. AIS-Lib consists of five key component packages: the *core*, *matching*, *misbehavior*, *monitoring*, and *utils* package. The core package defines the main AIS features, e.g. whether the detector generation task or the detection and observation task should be started, the gene library, and it furthermore defines which monitoring and matching abilities should be used. A factory pattern is used to allow the creation of different kinds of AIS cores. The matching and misbehavior packages define the offered implementations for matching rules (such as Hamming distance matching or *r*-contiguous bits matching) and misbehavior (such as packet dropping or wormhole-based misbehavior). Both packages can be easily extended. The monitoring package defines a *NodeDatabase* and a *NodeProperty* class which handle the monitoring and storage of observed node behavior. This node behavior represents detected misbehavior which can later, during an evaluation of the traffic traces and other stored information, be used to define more precisely which nodes were misbehaving and which were wrongly accused of misbehaving. The utils package offers helper classes for gene encoding and detector generation.

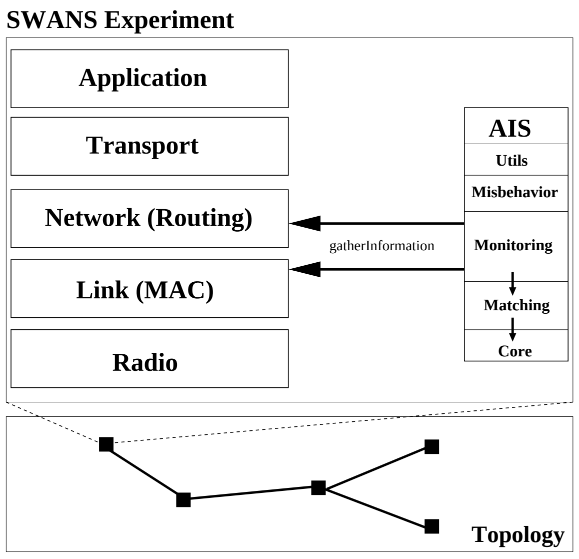

Fig. 1. Modular setup of an AIS-Lib experiment. The AIS currently uses data from the Network and Link layers.

V. Related Work

To our best knowledge there is no general AIS tool available that would be specifically aimed at ad hoc or sensor networks. The approaches presented in [6], [7], [12], [18] are specific to the needs of given performance evaluation.

VI. Conclusions

We introduced and discussed AIS-Lib, an AIS library aimed at ad hoc and sensor wireless networks. This library is an add-on to SWANS, a Java based network simulation tool. Our current goal is to implement various types of danger signals; see [12], [14]. It is our intention to make the AIS-Lib open-source [19].

Acknowledgments

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

References

[1] M. Drozda and H. Szczerbicka., "Artificial immune systems: Survey and applications in ad hoc wireless networks." *Proc. 2006 International Symposium on Performance Evaluation of Computer and Telecommunication Systems (SPECTS'06)*, pp. 485–492, 2006.

[2] P. Harmer, P. Williams, G. Gunsch, and G. Lamont, "An artificial immune system architecture for computer security applications," *Evolutionary Computation, IEEE Transactions on*, vol. 6, no. 3, pp. 252–280, 2002.

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

[4] J. Sterbenz, R. Krishnan, R. Hain, A. Jackson, D. Levin, R. Ramanathan, and J. Zao, "Survivable mobile wireless networks: issues, challenges, and research directions," *Proceedings of the ACM workshop on Wireless security*, pp. 31–40, 2002.

[5] L. de Castro, *Fundamentals of Natural Computing*. Chapman & Hall/CRC, 2006.

[6] M. Drozda, S. Schaust, and H. Szczerbicka, "Is AIS Based Misbehavior Detection Suitable for Wireless Sensor Networks?" *Proc. IEEE Wireless Communications and Networking Conference (WCNC'07)*, 2007.

[7] ——, "AIS for Misbehavior Detection in Wireless Sensor Networks: Performance and Design Principles," *IEEE Congress on Evolutionary Computation (CEC'2007)*, pp. 3719–3726, 2007.

[8] R. Barr, Z. Haas, and R. Renesse, "Scalable wireless ad hoc network simulation," in *Handbook on Theoretical and Algorithmic Aspects of Sensor, Ad hoc Wireless, and Peer-to-Peer Networks*. CRC Press, 2005, pp. 297–311.

[9] W. Ye and J. Heidemann, "Medium Access Control in Wireless Sensor Networks," *Wireless Sensor Networks*, pp. 73–91, 2004.

[10] C. Janeway Jr, "How the immune system works to protect the host from infection: A personal view," *Proceedings of the National Academy of Sciences*, vol. 98, no. 13, pp. 7461–7468, 2001.

[11] S. Hofmeyr and S. Forrest, "Immunity by design: An artificial immune system," *Proceedings of the Genetic and Evolutionary Computation Conference*, vol. 2, pp. 1289–1296, 1999.

[12] S. Sarafijanovic and J. Le Boudec, "An artificial immune system for misbehavior detection in mobile ad-hoc networks with virtual thymus, clustering, danger signal and memory detectors," *Proc. International Conference on Artificial Immune Systems (ICARIS)*, pp. 342–356, 2004.

[13] P. D'haeseleer, S. Forrest, and P. Helman, "An Immunological Approach to Change Detection: Algorithms, Analysis and Implications," *IEEE Symposium on Security and Privacy*, vol. 95, 1996.

[14] U. Aickelin, P. Bentley, S. Cayzer, J. Kim, and J. McLeod, "Danger Theory: The Link between AIS and IDS?" *Proc. of the Second Internation Conference on Artificial Immune Systems (ICARIS-03)*, pp. 147–155, 2003.

[15] K. Bhargavan, C. Gunter, M. Kim, I. Lee, D. Obradovic, O. Sokolsky, and M. Viswanathan, "Verisim: Formal analysis of network simulations," *IEEE Transactions on Software Engineering*, vol. 28, no. 2, pp. 129–145, 2002.

[16] S. Marti, T. Giuli, K. Lai, and M. Baker, "Mitigating routing misbehavior in mobile ad hoc networks," *Proceedings of the 6th annual international conference on Mobile computing and networking*, pp. 255–265, 2000.

[17] Y. Hu, A. Perrig, and D. Johnson, "Wormhole attacks in wireless networks," *IEEE Journal on Selected Areas in Communications*, vol. 24, no. 2, pp. 370–380, 2006.

[18] N. Mazhar and M. Farooq, "BeeAIS: Artificial Immune System Security for Nature Inspired, MANET Routing Protocol, BeeAdHoc," *Proc. International Conference on Artificial Immune Systems (ICARIS)*, pp. 370–381, 2007.

[19] AIS-Lib. An AIS library for ad hoc and sensor networks. `www.sim.uni-hannover.de/ais-lib/`.