

A consensus based network intrusion detection system

Michel Toulouse
Faculty of Engineering
Vietnamese-German University
Ho Chi Minh City, Vietnam
Email: michel.toulouse@vgu.edu.vn

Bùi Quang Minh
Faculty of Engineering
Vietnamese-German University
Ho Chi Minh City, Vietnam

Philip Curtis
Department of Computer Science
Oklahoma State University
Stillwater, Oklahoma, US

Abstract—Network intrusion detection is the process of identifying malicious behaviors that target a network and its resources. Current systems implementing intrusion detection processes observe traffic at several data collecting points in the network but analysis is often centralized or partly centralized. These systems are not scalable and suffer from the single point of failure, i.e. attackers only need to target the central node to compromise the whole system. This paper proposes an anomaly-based fully distributed network intrusion detection system where analysis is run at each data collecting point using a naïve Bayes classifier. Probability values computed by each classifier are shared among nodes using an iterative average consensus protocol. The final analysis is performed redundantly and in parallel at the level of each data collecting point, thus avoiding the single point of failure issue. We run simulations focusing on DDoS attacks with several network configurations, comparing the accuracy of our fully distributed system with a hierarchical one. We also analyze communication costs and convergence speed during consensus phases.

Keywords—Anomalie-based network intrusion detection; average consensus protocol; naïve Bayes classifier; DDoS attacks

September 16, 2015

I. INTRODUCTION

Security experts and researchers have proposed and implemented different strategies to defend computer installations against attacks. Among them *intrusion detection systems* (IDSs) seek specifically to identify attacks that could target a computer or a network and its resources. IDSs have two main components: a *data audit* component, sensors or log files, that monitor/collect data on the system behavior; a *detection method* component which analyzes the observed/collected data to detect malicious activities. In terms of the audit component, IDSs are classified as host-based (HIDS) or network-based (NIDS) [1]. Host-based IDSs detect attacks to a computer system by monitoring mainly operating system events. Network-based IDSs detect attacks to nodes connected by a network by monitoring network TCP/IP events. In terms of detection methods, IDSs are further classified as signature-based or anomaly-based detection systems. Signature-based systems monitor traffic for known attack patterns (signatures), similar to virus scanners that protect personal computers. Signature-based IDSs efficiently detect existing threats but always lag behind new threads. Anomaly-based systems [2] detect intrusions by classifying observed traffic as either normal or anomalous based on a profile that characterizes normal

behavior. Anomaly-based systems are better at detecting new types of attacks but they usually experience a high level of false positives (report an attack when there is none).

Increasingly, networks are themselves interconnected and more heterogeneous, which make them the target of sophisticated attacks that could spread over different administrative domains. Various new NIDS architectures have been proposed to protect these networks. In *centralized* NIDS, the audit component of the system is distributed, collected audits are forwarded to one or a small number of nodes where the analysis takes place. *Hierarchical* NIDS are often a network of several local NIDSs, each of them protecting a different sub-network. The local NIDSs decide about the status of their sub-network, decisions that are sent to a centralized node which makes a final decision about the status of the whole network, often using a simple majority rule. This approach is more scalable and it allows to detect attacks that will not be detected using a single NIDS [3]. Hierarchical NIDSs can be degraded substantially by taking down the root node of the system. A *truly distributed* NIDS is one where the data collection component and the analysis component of the IDS are combined in a single component residing at every data collection point. Cooperating security managers (CSM) [4] is one implementation of this type of NIDS. A similar approach for cloud computing is proposed in [5]. Such distributed systems incur communication overheads since, to complete its analysis phase, each node must receive information such as audit data from all the other nodes. *Mobile agent-based* NIDSs [6] address this communication issue through code migration. In this design, mobile NIDS agents migrate between nodes to carry the analysis. The issue with the mobile agent-based approach is the time needed to compute the final analysis may exceed the real time requirements of a NIDS.

The design of NIDSs can be represented as a three phases process [2]: parameterization, training and detection. The main purpose of parameterization is features extraction: identifying those features that separate normal from attack traffic. The outcome of this phase is a features vector f_1, f_2, \dots, f_m , m is the number of features. The training and detection phases depend on the method used to process feature vectors. In [2], anomaly-based detection methods are classified as either statistical based, knowledge based or machine learning based (see [7] for a more recent and different taxonomy). Our proposed NIDS relies on the Naïve Bayes classifier as processing method, which can be considered as a machine

learning approach to anomaly-based detection.

Naïve Bayes classifiers apply the Bayes rule

$$\mathbf{P}(H|O) = \alpha \mathbf{P}(H) \mathbf{P}(O|H) \quad (1)$$

to infer a probability distribution on a set of explanatory hypotheses about the behavior of a system. In (1), $\mathbf{P}(H)$ denotes the probability distribution for the specific set of hypotheses $H = \{h_a, h_n\}$, traffic is normal h_n or traffic is anomalous h_a . An observation is denoted by O , it is a vector o_1, o_2, \dots, o_m of m values, one value for each feature. The distribution $\mathbf{P}(H)$ is based on prior observations or a training phase, it is identified as “prior” knowledge. Similarly, conditional probabilities $\mathbf{P}(O|H)$ express the “likelihood” the combination of values $o_i \in O$, $i = 1..m$ can occur conditioned by each hypothesis, it is learned during the training phase of the Bayesian system. The Bayes rule computes the *posterior* probability distribution $\mathbf{P}(H|O)$ which, after consideration of the observation O and the priors, gives the probability that the observed traffic is normal and anomalous. For example, the probability of anomalous traffic $\mathbf{P}(h_a|O)$ is computed as $\alpha \mathbf{P}(h_a) \prod_{k=1}^m P(o_k|h_a)$, where α is a normalization constant.

The present paper proposes a fully distributed NIDS. The proposed system uses n NIDS modules each running a Naïve Bayes classifier to compute distributively posterior probably distributions about the state of the network. The detection component of this distributed system has two phases. In the first phase, local likelihood probabilities $\mathbf{P}(O_i|h) = \prod_{k=1}^m P(o_k|h)$ are computed by each module i based on the traffic observation $o_1^i, o_2^i, \dots, o_m^i$ of module i . The second phase computes the joint distribution of the local likelihoods obtained in the first phase. Assuming conditional independence of the n local observations, the joint distribution $\mathbf{P}(O|h)$, $O = \cup_i \{O_i\}$, is the product of the local likelihoods: $\mathbf{P}(O|h) = \prod_{i=1}^n P(O_i|h)$. In this second phase, NIDS modules execute cooperatively an average consensus algorithm [8], [9] which computes the joint distribution $\mathbf{P}(O|h)$ asymptotically and in parallel without the help of any centralized storage or computation.

Our work is based on recent proposals about average consensus algorithms in sensor networks for distributed hypothesis testing, distributed detection, multi-target tracking and others [10]. To the best of our knowledge it is the first time these approaches are adapted to distributed intrusion detection systems. Therefore, this is the main contribution of this research. The paper is organized as follow. Next section introduces average consensus. Section III provides a formulation of the average consensus problem for a fully distributed intrusion detection system. Section IV describes our distributed algorithm and analyzes its behavior with different simulated NIDS networks. Finally, Section V concludes.

II. AVERAGE CONSENSUS

Consensus is the problem of finding an agreement among autonomous entities such as peoples, autonomous software agents or computers in a network. Each agent i has a state variable x_i initialized to some value v_i . Agents must agree on a single output value while communicating directly with only a subset of the other agents, this subset is denoted by \mathcal{N}_i , the neighborhood of agent i . Consensus problems arise for example when multiple sensors observe a same object

or control devices compute the same response to a system behavior. Though they may be provided with different inputs, the computing devices have to agree on a same output. Consensus problems have been studied in computer science [11], physics [12], operations research [13] and control theory [14].

Average consensus is a consensus problem where agents must agree on the average sum of the input values: $(\frac{1}{n}) \sum_{i=1}^n x_i(0)$, for n the number of agents. The distributed averaging problem is solved cooperatively by a network of agents where each agent i iteratively computes a linear weighted sum of $x_i(t)$ and $x_j(t)$, $j \in \mathcal{N}_i$:

$$x_i(t+1) = x_i(t) + \sum_{j \in \mathcal{N}(i)} W_{ij}(x_j(t) - x_i(t)), \quad t = 0, 1, \dots \quad (2)$$

where $x_i(0) = v_i$ and W_{ij} is the weight of the edge connecting agents i and j in the network.

Important issues are whether the iterates converge to the consensus value and how fast they converge (how many iterations are required for convergence). Convergence and the speed of convergence are analyzed through the weight matrix \mathbf{W} (also called the consensus matrix) which is formed from the row vectors in W_{ij} of each agent i . Proofs of convergence rely on the network connectivity assumption and on properties of the graph Laplacian. A network is connected if there is path in the network between each pair of agents. If a network is connected then it has a graph Laplacian. The graph Laplacian L of a network is a $n \times n$ matrix where $L[i, j]$ is given as:

$$L[ij] = \begin{cases} -1 & \text{if } j \in \mathcal{N}_i \\ |\mathcal{N}_i| & \text{if } i = j \text{ } (|\mathcal{N}_i| \text{ is the number of neighbors} \\ & \text{to process } i) \\ 0 & \text{otherwise} \end{cases}$$

The conditions for convergence are the following [15]:

- 1) \mathbf{W} has the same sparsity as the graph Laplacian
- 2) $\mathbf{W}^t = \mathbf{W}$
- 3) $\mathbf{W}\mathbf{1} = \mathbf{1}$
- 4) The norm $\|\mathbf{W} - \mathbf{J}\| < 1$

in which \mathbf{W}^t denotes the transpose matrix of \mathbf{W} , $\mathbf{1}$ denotes the vector of all ones and $\mathbf{J} = \frac{1}{n} \mathbf{1}\mathbf{1}^t$. It is relatively easy to find a weight matrix that satisfies these condition. For example, $W = \mathbf{I} - \alpha L$, with $0 < \alpha < \frac{1}{\max_{i,j} |\mathcal{N}_i|}$ is such weight matrix where $\max |\mathcal{N}_i|$ is the neighborhood with the largest cardinality and \mathbf{I} is the identity matrix. A second example is the Metropolis-Hasting matrix:

$$W_{ij} = \begin{cases} \frac{1}{1 + \max(d_i, d_j)} & \text{if } i \neq j \text{ and } j \in \mathcal{N}_i \\ 1 - \sum_{k \in \mathcal{N}_i} W_{ik} & \text{if } i = j \\ 0 & \text{if } i \neq j \text{ and } j \notin \mathcal{N}_i \end{cases} \quad (3)$$

where $d_i = |\mathcal{N}_i|$, i.e. the number of processes adjacent to process i .

However, selecting the coefficients of weight matrix \mathbf{W} such to optimize the speed of convergence is still very much a research issue. Interested readers can consult [9], [16].

III. A CONSENSUS BASED NETWORK INTRUSION DETECTION SYSTEM

We formulate the average consensus problem for a fully distributed network-based intrusion detection system. It is composed of n NIDS modules connected by an independent and secure network. Without loss of generality, we assume that the links connecting pairs of NIDS modules are direct physical links. It is assumed that this network is connected, though it does not fully connect pairwise all the n NIDS modules.

Each NIDS module i has two state variables x_i^a and x_i^n . The initialization of the state variables is performed in the first phase of the system as follows: Once a module i has completed a traffic observation, it computes its local likelihood

$$P(O_i|h_a) = P(o_1^i, o_2^i, \dots, o_m^i|h_a) = \prod_{k=1}^m (o_k^i|h_a) \quad (4)$$

(respectively $P(O_i|h_n)$). The state variables are initialized using the logarithm of the local likelihoods: $x_i^a = \log(P(O_i|h_a))$ and $x_i^n = \log(P(O_i|h_n))$.

The second phase computes the joint distribution of the local likelihoods $(O|h_a) = \prod_{i=1}^n P(O_i|h_a)$. The computation of this product of probabilities is transformed into the computation of an average sum using the laws of logarithms: the log of a product is equal to the sum of the log of the terms in the product. Therefore, $\log(P(O|h)) = \log(\prod_{i=1}^n P(O_i|h_a)) = \sum_{i=1}^n \log(P(O_i|h_a))$. Taking the average of the last sum we obtain

$$Q^a = \frac{1}{n} \log(P(O|h_a)) = \frac{1}{n} \sum_{i=1}^n \log(P(O_i|h_a)) \quad (5)$$

(respectively Q^n). Formulated as an average sum, the terms Q^a and Q^n can be computed distributively and asymptotically using the average consensus iterate in (2). Upon convergence of its iterate, i.e. when $|x(t+1) - x(t)| < \epsilon$ (for ϵ sufficiently small), a NIDS module makes use of the intermediate results Q^a and Q^n to compute locally the joint distribution of the local likelihoods: $P(O|h_a) = \exp(nQ^a) \approx \prod_{i=1}^n P(O_i|h_a)$ (respectively $P(O|h_n) = \exp(nQ^n)$). At this point, a module has all the information needed to execute naïve Bayesian inferences on the state of the overall network, i.e. to compute the posterior probability $\mathbf{P}(h|O) = \alpha \mathbf{P}(h) \mathbf{P}(O|h)$ that the network traffic is normal or anomalous.

IV. EXPERIMENTAL ANALYSIS

This section compares our intrusion detection system based on average consensus with one based on a hierarchical approach. In contrast to average consensus, the network wide detection phase of the hierarchical approach is implemented by sending the local likelihoods to a central node which computes directly the joint distribution.

All comparisons between the two approaches are based on simulations of NIDS networks of different sizes and topologies. Each simulation takes in input a test set and a graph representing an NIDS network topology. Network connections for the test set come from the NLS-KDD data set [17], an improved version of KDD'99 data set. The KDD'99 data set has been generated by the MIT Lincoln Laboratory for the evaluation of computer network intrusion detection systems

under the sponsorships of the Defense Advanced Research projects Agency (DARPA) and the Air force Research Laboratory (AFRL) [18], [19]. As for the KDD'99 data set, connections in the NLS-KDD data set are described in terms of 41 features.

We have run tests with four categories of NIDS network topologies represented by four types of non-oriented input graphs: rings, 2-dimensional torus, the Petersen graph (10 nodes 15 edges) and several random graphs having the same number of vertices and edges as in the Petersen graph. Ring, torus and the Petersen graph are regular graphs, each vertex in a given graph has the same degree: two for rings, three for the Petersen graph and four for the 2-dimensional torus. While the number of vertices and edges is constant among random graphs, in a same graph the vertex degree may vary from one vertex to another. Each vertex in a graph represents a NIDS module. The degree of a vertex stands for the size of the neighborhood \mathcal{N} of the associated NIDS module. Each graph type represents a different NIDS network topology. Finally, the different numbers of vertices in the ring and torus graphs represent networks of different sizes, respectively networks with 9, 25, 49, 81 and 121 NIDS modules.

Algorithm 1

- Step 0 **Training phase**; SimulationLoop = 0;
- Step 1 **First phase** Read values o_1, o_2, \dots, o_m corresponding to m features;
 $P(O|h_a) = \prod_{k=1}^m P(o_k|h_a)$; Compute local likelihood
 $x^a(0) = \log(P(O|h_a))$;
- Step 2 **Second phase (consensus loop)**
 $x^a(1) = x^a(0) + \sum_{j \in \mathcal{N}} w_j (x_j^a(0) - x^a(0))$;
 $t = 1$;
while $(|x(t) - x(t-1)| < \epsilon)$
 $x^a(t+1) = x^a(t) + \sum_{j \in \mathcal{N}} w_j (x_j^a(t) - x^a(t))$;
 $t = t + 1$;
 $P(O|h_a) = \exp(nx^a(t))$;
- Step 3 **Compute network-wide posterior probabilities**
 $p(h_a|O) = \alpha p(h_a) P(O|h_a)$;
- Step 4 **Compute decision** If $\frac{p(h_a|O)}{p(h_n|O)} > \tau$ raises alert;
- Step 5 **Simulation termination test** SimulationLoop++;
 If SimulationLoop < 1000, goto to Step 1;

Fig. 1. Average consensus based algorithm for network intrusion detection

As our solution to intrusion detection is fully distributed, a simulation consists to execute independently the code of each NIDS module in a given network topology. The code is the same for each module, it is summarized in Algorithm 1, which has been implemented in Java.

Algorithm 1 takes in input a training set, a weight matrix and some control parameters. The control parameters are the *convergence parameter* (ϵ) and the *decision parameter* (τ). The convergence parameter is the stopping criterion of the consensus loop. When $|x(t) - x(t-1)| < \epsilon$, i.e. when the change in the consensus value is smaller than ϵ , the NIDS module stops receiving/sending updates from/to its neighboring NIDS modules. This parameter is set 0.001 in our tests. The decision parameter τ set the threshold needed to raise an attack alert

based on network-wide posterior probabilities. This parameter is used for both the hierarchical and the consensus approaches, with the same value in both cases.

We have run tests with the following three input weight matrices [9]:

- The *Best-constant edge weight* scheme

$$W_{ij} = \frac{2}{\lambda_1(L) + \lambda_{n-1}(L)}$$

where L is the Laplacian matrix of the NIDS network, λ_1, λ_{n-1} are the first and $n - 1$ eigenvalues of L .

- The *Local-degree weights* scheme where the weight of an edge is the largest degree of its two adjacent vertices

$$W_{ij} = \frac{1}{\max\{d_i, d_j\}}.$$

- The *Max-degree weight* where d_{max} is the largest degree of the vertices in the network is a constant weight scheme

$$W_{ij} = \frac{1}{d_{max}}.$$

All three matrices satisfy the convergence conditions described in Section II.

Step 0 of Algorithm 1 represents the training phase of an NIDS module. This phase uses the training data from NLS-KDD. Steps 1 to 5 drive the simulation loop. In Step 1, a connection from the NLS-KDD data set is read. Next, the state variable $x^a(0)$ is initialized with the logarithm of the local likelihood (to shorten Algorithm 1, we only shows the computation of the anomalous hypothesis). Step 1 returns the likelihood that the corresponding connection is normal and the likelihood that it is anomalous. Note that step 1 has been implemented using the naïve Bayes Classifier from the weka library http://www.cs.waikato.ac.nz/~remco/weka_bn, develop by the Machine Learning Group project at the University of Waikato. Step 2 drives the consensus loop. After computing the value of $x^a(1)$ and initializing the loop iterate variable t , each iteration of the consensus loop sends the value of each NIDS module state variable to its neighbors in the network topology, wait to receive the corresponding values from its neighbors and then computes a weighted sum of the differences between the value of the neighboring state variables and the value of its own state variables. This communication/computation sequence is repeated until convergence. The variable w_j in step 2 is the weight of the edge from a NIDS module to neighbor j in the network, this value is provided by the weight matrix. Once the consensus loop stops, the inverse of the log and average functions are applied to $x^a(t)$ to get an approximation $P(O|h_a)$ of the joint distribution of the local likelihoods. Step 3 takes this approximation and computes an approximation of the network-wide posterior probability of each hypothesis. In Step 4, a decision is made whether the observed network behavior is normal or anomalous.

Algorithm 1 describes the behavior of each NIDS module. We now describe the behavior of simulations, where one simulation consists to iterate several times the execution of all the NIDS modules in a given network topology. Simulations have a few control parameters. For example, NIDS modules

in a given simulation can be trained with a same training set or with different training sets and can be given the same or different weight matrices. Considering the objectives of this research, we have selected to train the NIDS modules with the same data set, and in a given simulation, the NIDS modules all take in input the same weight matrix. In a given simulation, there is a predefined ratio of normal and anomalous connections in the NIDS network. For the tests reported in this paper, this ratio is 60%, i.e. 60% of the NIDS modules at a given iteration receive anomalous connections. This ratio is kept constant during a simulation. The number of iterations of the simulation loop is fixed to 1000 (termination criterion in Step 5). In one simulation each module executes 1000 connection readings, therefore a simulated NIDS network with 9 NIDS modules analyzes 9000 connections, a simulated NIDS network with 81 NIDS modules analyzes 81000 connections. We simulate synchronous NIDS networks, i.e. a new iteration of the simulation loop can only start once the consensus phase of all the modules is completed. All the simulations detect only Distributed Denial of Service (DDoS) attacks.

A. Test samples and results

Tests first analyze the impact of the weight matrices and NIDS network topologies on the convergence speed during the consensus phases. Both network topologies and weight matrices are known to impact the convergence speed of average consensus algorithms, and therefore their communication cost. Next, we compare the communication cost and the accuracy of the consensus versus hierarchical approaches.

Fig. 2 reports the average number of iterations of the consensus loop for the 3 different weight matrices and torus networks of respectively 9, 25, 49, 81 and 121 NIDS modules. Clearly, consensus with the best-constant weight matrix converges faster than the two others, which have very similar convergence speeds.

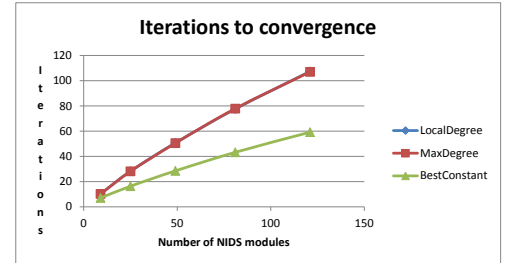


Fig. 2. Average number of consensus iterations to convergence

Using the best-constant weight matrix, Fig. 3 and Fig. 4 report the average number of iterations of the consensus loop for all the network topologies. Fig. 3 compares ring and torus topologies, showing that consensus convergence is much slower for ring networks. Fig. 4 compares the Petersen network (column 1) with 10 other random networks of same size (same number of NIDS modules and same number of connections in the networks). The Petersen graph is an instance of Ramanujan graphs, they are graphs known to have very good convergence speed for the average consensus algorithm [20]. In Fig. 4, the Petersen network has a faster convergence compared to the 10 other random networks.

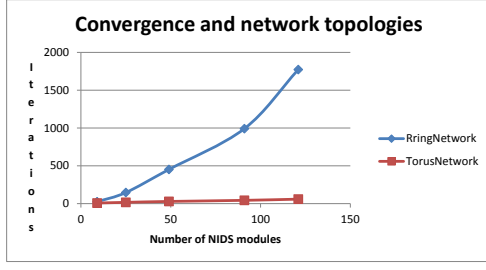


Fig. 3. Convergence speed for ring and torus network topologies

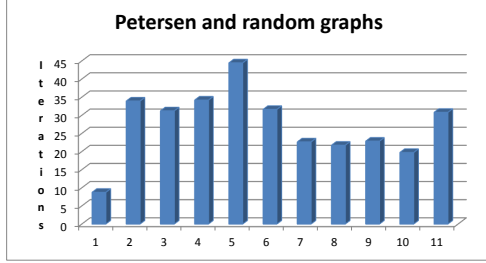


Fig. 4. Convergence speed for Petersen and random graphs

The posterior values computed in Step 3 of Algorithm 1 depend on joint likelihood distributions that are computed approximately in Step 2. The next tests determine whether these approximations are detrimental to the capacity of a consensus based system to detect anomalous activities by comparing the accuracy of the consensus and hierarchical approaches. Accuracy measures how often decisions made like in Step 4 of Algorithm 1 are the correct one. It is defined as follows:

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

where TP (*True positive*) is the number of attacks detected when it is actually an attack; TN (*True negative*) is the number of normals detected when it is actually normal; FP (*False positive*) is the number of attacks detected when it is actually normal; FN (*False negative*) is the number of normals detected when it is actually an attack.

Fig. 5 reports the results of 63 tests. Tests 0 to 29 concern ring (even numbers) and torus (odd numbers) NIDS networks. Tests from 0 to 9, 10 to 19 and 20 to 29 report respectively the results for local-degree, max-degree and best-constant weight matrices. Tests 0-1, 2-3, 4-5, 6-7, 8-9 report results for networks having respectively 9, 25, 49, 81 and 121 NIDS modules and using the local-degree weight matrix (similarly for max-degree and best-constant weight matrices). Tests 30, 41 and 52 report results for Petersen graphs respectively for the local-degree, max-degree and best-constant weight matrices. Tests 31 to 40, 42 to 51 and 53 to 62 report results for random graphs respectively for the local-degree, max-degree and best-constant weight matrices. Fig. 5 shows that the accuracy of the hierarchical approach is slightly better than the consensus approach, but it is clear that approximating the posterior values with consensus is not detrimental to the accuracy of the system.

The consensus approach to network intrusion detection is more scalable because computation is fully distributed. On

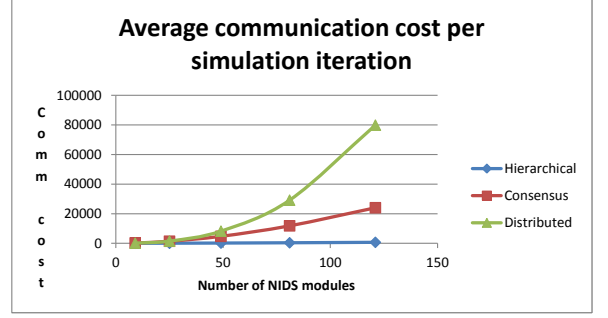


Fig. 6. Comparing the average communication cost of simulation iterations

the other hand, sharing information to support distributed computation incurs communication costs during the consensus phases. We now compare the communication cost of the fastest consensus approach with the hierarchical approach and with a second fully distributed approach to network intrusion detection. Communication costs are measured in the number of hops used in a message. A message is a communication between a pair of NIDS modules, a hop is a direct link between two adjacent NIDS modules. For consensus, a message cost a single hop as all messages take place between adjacent NIDS modules. For hierarchical, the cost of a message is equal to the length of the shortest path between a given NIDS module and the central module computing the posterior probabilities.

Fig. 6 reports the average communication cost per iteration of the simulations with the best-constant weight matrix and the torus network with respectively 9, 25, 49, 81 and 121 NIDS modules. For consensus, the communication cost of one simulation iteration is measured directly from the tests. For hierarchical, the communication cost h_{ce} of one simulation iteration is $h_{ce} = \sum_{i=1}^{n-1} l_i$ where l_i is the length of the shortest path between module i and the central module and n is the number of NIDS modules in the NIDS network. The “distributed” item in Fig. 6 is the communication cost of typical fully distributed approaches where information produced by one NIDS module, such as local likelihoods, is sent to all the other modules in the NIDS network. Communication cost for one simulation iteration is computed as $h_{co} = h_{ce} \times n - 1$ where h_{ce} is communication cost of one simulation iteration for the hierarchical approach. As expected, Fig. 6 shows that the communication cost of the consensus approach grows faster than for hierarchical. This figure also shows that the communication cost of fully distributed approaches grow very rapidly, and that consensus is quite successful at addressing this communication cost issue for distributed systems.

V. CONCLUSION

This paper has described a fully distributed network intrusion detection system based on an average consensus algorithm. Our work is more a proof of concepts than an actual system. Nonetheless, in our opinion and based on our preliminary results, consensus seems a viable alternative to implement distributed network intrusion detection systems. In future works, we will seek to improve the convergence speed of consensus phases by studying a broader range of network topologies and weight matrices. We also consider

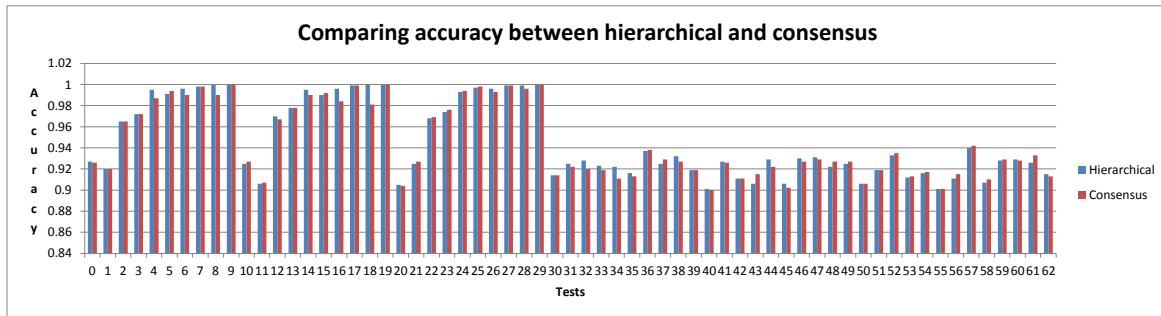


Fig. 5. Comparing accuracy between consensus and hierarchical approaches

to use this approach for intrusion detection in wireless ad hoc networks such as MANETS, which have no centralized control. Distributed consensus protocols are quite prevalent in these networks for addressing diverse coordination problems. A group of consensus protocols, called Byzantine consensus, address security issues such as intrusion in ad hoc networks through fault-tolerance mechanisms, they are designed to achieve consensus in the presence of nodes with unspecified, potentially malicious behaviors. Intrusion detection systems are also very much part of the defense mechanisms for ad hoc networks, and as expected, several of them are distributed IDSs. However, to the best of our knowledge, none of them make use of consensus protocols similar to the one proposed in this paper, therefore we intend to explore this avenue as future work.

REFERENCES

- [1] H. Debar, M. Dacier, and A. Wespi, "Towards a taxonomy of intrusion-detection systems," *Comput. Netw.*, vol. 31, no. 9, pp. 805–822, Apr. 1999. [Online]. Available: <http://dl.acm.org/citation.cfm?id=324119.324126>
- [2] P. Garca-Teodoro, J. Daz-Verdejo, G. Maci-Fernandez, and E. Vazquez, "Anomaly-based network intrusion detection: Techniques, systems and challenges," *Computers & Security*, vol. 28, no. 12, pp. 18–28, 2009. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S0167404808000692>
- [3] R. Gopalakrishna and E. H. Spafford, "A framework for distributed intrusion detection using interest driven cooperating agents," in *The 4th International Symposium on recent Advances in Intrusion Detection (RAID 2001)*, 2001.
- [4] G. White, E. Fisch, and U. Pooch, "Cooperating security managers: a peer-based intrusion detection system," *Network, IEEE*, vol. 10, no. 1, pp. 20–23, Jan 1996.
- [5] C.-C. Lo, C.-C. Huang, and J. Ku, "A cooperative intrusion detection system framework for cloud computing networks," in *Proceedings of the 2010 39th International Conference on Parallel Processing Workshops*, ser. ICPPW '10. Washington, DC, USA: IEEE Computer Society, 2010, pp. 280–284. [Online]. Available: <http://dx.doi.org/10.1109/ICPPW.2010.46>
- [6] W. A. Jansen, "Intrusion detection with mobile agents," *Comput. Commun.*, vol. 25, no. 15, pp. 1392–1401, Sep. 2002. [Online]. Available: [http://dx.doi.org/10.1016/S0140-3664\(02\)00040-3](http://dx.doi.org/10.1016/S0140-3664(02)00040-3)
- [7] Y. Yu, "A survey of anomaly intrusion detection techniques," *J. Comput. Sci. Coll.*, vol. 28, no. 1, pp. 9–17, Oct. 2012. [Online]. Available: <http://dl.acm.org/citation.cfm?id=2379703.2379707>
- [8] L. Xiao, S. Boyd, and S. Lall, "A scheme for robust distributed sensor fusion based on average consensus," in *Proceedings of the 4th international symposium on Information processing in sensor networks*, ser. IPSN '05. Piscataway, NJ, USA: IEEE Press, 2005. [Online]. Available: <http://dl.acm.org/citation.cfm?id=1147685.1147698>
- [9] L. Xiao and S. Boyd, "Fast linear iterations for distributed averaging," *Systems and Control Letters*, vol. 53, pp. 65–78, 2003.
- [10] R. Olfati-Saber, E. Franco, E. Frazzoli, and J. Shamma, "Belief consensus and distributed hypothesis testing in sensor networks," in *Networked Embedded Sensing and Control*, ser. Lecture Notes in Control and Information Science, P. Antsaklis and P. Tabuada, Eds. Springer Berlin Heidelberg, 2006, vol. 331, pp. 169–182. [Online]. Available: http://dx.doi.org/10.1007/11533382_11
- [11] N. A. Lynch, *Distributed Algorithms*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1996.
- [12] T. Vicsek, A. Czirók, E. Ben-Jacob, I. Cohen, and O. Shochet, "Novel type of phase transition in a system of self-driven particles," *Phys. Rev. Lett.*, vol. 75, pp. 1226–1229, Aug 1995. [Online]. Available: <http://link.aps.org/doi/10.1103/PhysRevLett.75.1226>
- [13] J. Tsitsiklis, D. Bertsekas, and M. Athans, "Distributed asynchronous deterministic and stochastic gradient optimization algorithms," *Automatic Control, IEEE Transactions on*, vol. 31, no. 9, pp. 803–812, Sep. 1986.
- [14] R. Saber and R. Murray, "Consensus protocols for networks of dynamic agents," in *American Control Conference, 2003. Proceedings of the 2003*, vol. 2, June 2003, pp. 951–956.
- [15] L. Xiao, S. Boyd, and S.-J. Kim, "Distributed average consensus with least-mean-square deviation," *J. Parallel Distrib. Comput.*, vol. 67, no. 1, pp. 33–46, Jan. 2007. [Online]. Available: <http://dx.doi.org/10.1016/j.jpdc.2006.08.010>
- [16] A. Olshevsky and J. N. Tsitsiklis, "Convergence speed in distributed consensus and averaging," *SIAM J. Control Optim.*, vol. 48, no. 1, pp. 33–55, Feb. 2009. [Online]. Available: <http://dx.doi.org/10.1137/060678324>
- [17] M. Tavallae, E. Bagheri, W. Lu, and A. Ghorbani, "A detailed analysis of the kdd cup 99 data set," in *Computational Intelligence for Security and Defense Applications, 2009. CISDA 2009. IEEE Symposium on*, July, pp. 1–6.
- [18] R. Lippmann, J. Haines, D. Fried, J. Korba, and K. Das, "Analysis and results of the 1999 darpa off-line intrusion detection evaluation," in *Recent Advances in Intrusion Detection*, ser. Lecture Notes in Computer Science, H. Debar, L. M., and S. Wu, Eds. Springer Berlin Heidelberg, 2000, vol. 1907, pp. 162–182. [Online]. Available: http://dx.doi.org/10.1007/3-540-39945-3_11
- [19] R. Lippmann, D. Fried, I. Graf, J. Haines, K. Kendall, D. McClung, D. Weber, S. Webster, D. Wyschogrod, R. Cunningham, and M. Zissman, "Evaluating intrusion detection systems: the 1998 darpa off-line intrusion detection evaluation," in *DARPA Information Survivability Conference and Exposition, 2000. DISCEX '00. Proceedings*, vol. 2, 2000, pp. 12–26 vol.2.
- [20] S. Kar and M. F. Moura, "Topology for global average consensus," in *Signals, Systems and Computers, 2006. ACSSC '06. Fortieth Asilomar Conference on*, Oct 2006, pp. 176–181.