Multi-agent Reinforcement Learning for Intrusion Detection

Arturo Servin and Daniel Kudenko

Department of Computer Science, University of York
Heslington, York. YO10 5DD, United Kingdom
{aservin,kudenko}@cs.york.ac.uk

Abstract. Intrusion Detection Systems (IDS) have been investigated for many years and the field has matured. Nevertheless, there are still important challenges, e.g., how an IDS can detect new and complex distributed attacks. To tackle these problems, we propose a distributed Reinforcement Learning (RL) approach in a hierarchical architecture of network sensor agents. Each network sensor agent learns to interpret local state observations, and communicates them to a central agent higher up in the agent hierarchy. These central agents, in turn, learn to send signals up the hierarchy, based on the signals that they receive. Finally, the agent at the top of the hierarchy learns when to signal an intrusion alarm. We evaluate our approach in an abstract network domain.

1 Introduction

As computer networks and information systems become more critical and complex, researchers are looking for new techniques to protect these assets. In this paper we present our work on the application of distributed reinforcement learning to allow cooperative network devices to identify and categorize faults, attacks and in general, any abnormal state in the network. The use of heterogeneous agents to detect distributed denial of service without central processing or management is an area with very little previous work. The number of studies that use machine learning techniques to scale up the solution to inter-domain networks or to adapt it to changes in traffic and attack behavior is scarce as well.

From a machine learning perspective, network intrusion and fault detection provides challenging scenarios to test and develop new multi-agent reinforcement learning techniques. To achieve reliable intrusion detection, RL will need to deal with noisy inputs and large discrete or continuous state-action spaces. We have chosen a hierarchical architecture of agents to provide a coordination scheme and learning mechanisms using data from distributed sources. The paper is structured as follows. In Section 2 we present a brief overview of the problem of detecting and categorizing Distributed Denial of Service Attacks, a review of IDS and point out some of the challenges these systems are still facing. The last part of this section is an overview of Multi-Agent Reinforcment Learning (MARL) and the problems that it faces as the number of agents and input information grows. In Section 3 we explain our proposed technique and the assumptions

K. Tuyls et al. (Eds.): Adaptive Agents and MAS III, LNAI 4865, pp. 211–223, 2008.

that we made when designing it. In Section 4 we show some results obtained from testing different architectures varying the number of agents, the number of states per sensor agent, the exploration/exploitation strategy, the distribution of attacks as input information and the agent architecture. Finally in Section 5 we point out our conclusions and an outlook to future work.

2 Background

In this section we introduce some concepts and terminology that provide the background to our approach.

2.1 Denial of Service Attacks

Denial of Service (DoS) attacks are very common in today's internet infrastructure. In a DoS, the attacker tries to exhaust key resources of the target to refuse a service to the legitimate users. DoS can be performed directly to attack a target or they can be an effect of other security problems such as the spreading of self replicating code or worms. A more worrisome type of this threats is the Distributed Denial of Service Attack (DDoS). DDoS are launched from several sources attacking one target. The effect would depend on the number of sources, the available bandwidth for each of them and the vulnerability that they are exploiting.

Defenses against DoS and DDoS are complex to design due to several factors. DoS are always accompanied by a heavy use of some kind of resource. If this resource is not heavily used, it is easy to identify the threat comparing normal to abnormal activity. DDoS use a distributed control with thousands of attackers spreading all over the Internet. To accurately identify and stop them it is necessary to coordinate several entities along the path of the DoS attack [10,20]. Under this assumption Mirkovic and Reiher [9] in their Taxonomy of DoS attacks and defenses state: *"the need for a distributed response at many points on the Internet".*

2.2 Intrusion Detection Systems

Intrusion Detection Systems are just one part of the whole collection of technologies and processes needed to protect computer networks and information systems from intruders and attacks. In combination with firewalls, IDSs are the first line of defense in many computer networks. An IDS monitors hosts or networks searching for abnormal or non-authorized activity. When they find attack activity, they record the event and they may perform defensive actions. There are two basic types of IDS: anomaly intrusion detection and misuse/signature intrusion detection. Anomaly IDS uses different methods to detect abnormal activity; they vary from simple statistical methods to more complex AI techniques. Misuse or signature intrusion detection system use rule matching to detect intrusions. These IDSs compare system activity with specific intrusion rules that are

generally hard coded. When the observed activity matches the intrusion pattern an intrusion is detected.

Because Anomaly IDSs compare current activity with a model of normal behavior they can detect unknown attacks when the network state is deviating from normal activity. However, non-malicious activity that does not match normal behavior can also trigger the intrusion mechanism. This results in a high rate of false positives or false alarms in anomaly IDSs. On the other hand, misuse-signature IDS are very reliable and they have low rates of false positives. Nevertheless, they lack the ability to detect new types of attacks. Other dimensions along which IDSs can be categorized are the type of response to detected intrusions (passive or active), the type of data-processing and the data-collection (centralized or distributed), and the source of the audit data (host or network).

As computer networks become more complex systems and threats on them are reaching global magnitudes researchers are looking for novel approaches to adapt IDS to these new needs. Some authors [2,3,11,16,19] point out that the use of a rich diversity of sensor information may achieve the development of more reliable IDS. The rationale behind this is that sensor variety is needed because each sensor perceives different information depending on its capabilities, its function and where it is deployed in the network. The amount of information required to infer malicious activity using distributed heterogeneous sensor architectures would overwhelm any human network manager and automatic processing becomes necessary.

2.3 Reinforcement Learning

Our approach to intrusion and fault detection is based on RL, where each network node is learning to send signals to other nodes in a network hierarchy. Before describing the details of this approach, we briefly introduce the main RL concepts.

In RL, agents or programs sense their environment in discrete time steps and they map those inputs to local state information. RL agents execute actions and observe the feedback from the environment or a trainer in the form of positive or negative rewards. After performing an action and receiving a reward, the agent observes any change in the environment and it updates its policy in order to optimize the reward received for future actions [18]. There are different approaches to calculate the optimal policy and to maximize the obtained reward over the time. One of the most widely used techniques is Q-learning. In Q-learning as in other Temporal-Difference-Learning methods the agent iteratively tries to estimate the value function. To estimate the value function, Q-learning constructs a table (Q-table) whose rows are states and columns are actions. The agent in each state s chooses an action a, observes the reward r and the next state s'. Then it updates the estimated Q-value denoted by $\hat{Q}$ in Equation (1). In this equation α is the learning rate with a value $0 < \alpha < 1$ and γ is a constant with value $0 < \gamma < 1$ that represents the relative value of delayed versus immediate rewards.

$$\hat{Q}(s,a) \leftarrow (1-\alpha)\hat{Q}(s,a) + \alpha(r + \gamma \; max_a \; \hat{Q}(s',a')) \tag{1}$$

The exploration/exploitation problem is a specific challenge that is present in reinforcement learning algorithms. To obtain the best reward, the agent tends to prefer actions that have been proved to provide high rewards. In order to discover these actions the agent needs to try actions that have not been tested. We say that the agent *exploits* actions that lead to better expected rewards but also it needs to *explore* other actions that may lead it to better rewards in the future [18]. In order to converge to the optimal policy, the agent needs to explore and to exploit actions. One simple solution is to use a *random* strategy. While in theory this strategy guarantees convergence; in practice it is very slow. A more subtle alternative is to let the agent explore actions in the beginning of the learning and progressively start choosing those actions that prove to lead to better expected rewards. ϵ*-greedy* and *Boltzmann* use this alternative. ϵ-greedy is a semi-uniform random exploration strategy; it uses a small value as a base probability to choose an action. The downside of e-greedy is that it chooses among all the actions with the same probability. To address this problem Boltzmann strategy, also called *softmax action selection rules*, weights each action with a probability according to their expected value using the given equation Equation.2:

$$P(a) = \frac{e^{Q(s,a_n)/T}}{\sum_0^i e^{Q(s,a_i)/T}} \tag{2}$$

T is a positive number called *temperature*. Under high values of temperature the action selection tends to choose equally between all actions. Low values of temperature favor actions with high expected values. In practice to speed up convergence, the value of the temperature is decreased exponentially.

Reinforcement Learning has been adopted to solve problems where on-line learning is needed and where the construction of a model is difficult or not possible. For more complex problems involving the interaction of several agents, RL becomes an appealing yet challenging alternative due to several factors. The *curse of dimensionality* that affects standalone RL and other machine techniques has an even bigger effect in MARL, as the number of agents and states increase and it becomes difficult to scale these systems to a large number of agents. Different approaches from function approximation techniques [8,17] to hierarchical reinforcement learning [1,5] have been proposed to scale MARL to large number of agents.

Some of the main issues surrounding MARL are:

1. In single agent RL, agents need to adapt their behavior in accordance with their own actions and how they change the environment. In addition to this, MARL agents also need to adapt to other agents' learning and actions.
2. MARL agents do not always have a full view of the environment and even if they have, they normally cannot predict the actions of other agents and the changes in the environment [6].
3. The *credit assignment problem* [15] describes the difficulty of deciding which agent is responsible for successes or failures of the multi-agent system. Related to this, the question arises on how to split the reward signal among the

agents. The reward can be the same for all agents *(global reward)* or it can be assigned based on the individual contribution of the agent *(local reward)*.

3 Agent Architecture and Operation

The security of computer networks is provided by devices such as IDS. As previously mentioned IDS monitor the network and detect abnormal or non authorized activity. When it detects suspicious activity, it records the event and in some cases performs defensive actions. The use of a rich diversity of sensor information may achieve more reliable detection of abnormal events in the network. Different network devices can provide diverse information based on their capabilities, their local network state observations, and their location in the network.

To process the information of distributed heterogeneous sensors to infer malicious activity there are multiple choices ranging from central control and management to peer to peer agent interaction; and from flat topologies to hierarchical central management and clustering. Since it is infeasible to assume that agents are able to communicate their complete local state observations (due to bandwidth restrictions), we have chosen an approach that is somewhere between central management and distributed control.

We propose a hierarchical architecture of Distributed Intrusion Detection Systems (DIDS) integrated by remote sensor agent diversity and reinforcement learning to detect and categorize DDoS Attacks. In this approach distributed sensors process the local state information and pass on short signals up a hierarchy of RL-IDS agents. With these signals the RL-IDS agents learn to distinguish abnormal activity from a diversity of sources. The lower the hierarchical level of the agent is; the more local information it is processing. The result is that high-level hierarchical agents have a better overview of the current state of the whole network. Under this consideration the agent on top of the hierarchy learns whether or not to trigger an overall alarm to the network operator.

Our base topology or Basic Cell is shown in Figure 1. It is composed of one central agent and n sensor agents. Sensor agents are in the form of network devices and they vary in capabilities and information that they can process. Each sensor agent receives only partial information about the global state of the network and they map local state information to communication signals which they send to the central agent (RL-IDS) of the cell (the signal constitutes the action of the sensor agent). The RL-IDS agent tries to model the state of the monitored network through these signals and decides in turn on a signal action. If the signal is in accordance with the real state of the monitored network, all the agents receive a positive reward. If the action is inaccurate, all the agents receive a negative reward. The goal is that after a certain number of iterations of the algorithm, every agent would know for each state the action that they need to execute to obtain positive rewards.

To expand the sensor architecture to analogous computer network architectures we created a hierarchical architecture with 2 levels as shown in Figure 2.

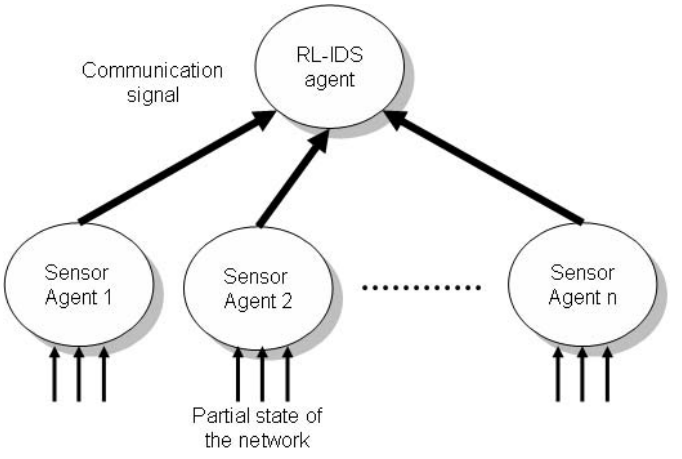

Fig. 1. Basic Cell of Agents. Each Sensor Agent sends communication signals to the RL-IDS agent.

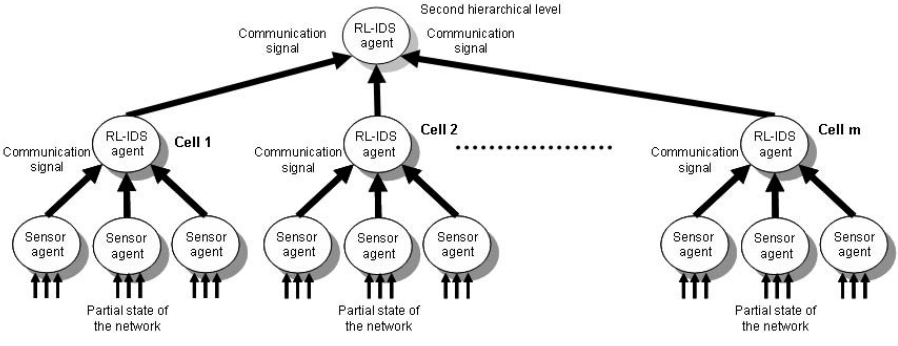

Fig. 2. Hierarchical Architecture: Each RL-IDS agent inside cells communicates with a higher level RL-IDS

This architecture is build from m cells with n agents per cell. In this topology each cell's RL-IDS agents receive local information from sensor agents and learn what signal to trigger to the next RL-IDS higher up in the hierarchy. Then, via the signals from the lower-level RL-IDS agents and the reward function, the high-level hierarchical RL-IDS agent in the topology learns which signal to trigger to the network operator or to the next hierarchical level RL-IDS. If we considered h as the number of hierarchical levels and n as the number of sensor agents per cell and the number of cells that one single RL-IDS can handle, the number of agents in a topology is denoted by Equation (3).

$$TotalNumberOfAgents = \sum_{i=0}^{h} n^{h-1} \tag{3}$$

We have opted to use a hierarchical architecture of agents instead of a flat one because we consider that the former adapts better to the topologies of real computer networks. These systems are constructed by several hierarchical layers where the lower layers perform data access to users and high layers perform high speed packet switching. Our hierarchical architecture is also easily adapted to process intrusion detection between different networks domains similarly as it occurs in real Internet interconnections.

4 Experiments and Results

We applied our algorithm to different agent architectures varying the number of agents, the exploration/exploitation strategy, the number of states per sensor agent, the distribution of attacks as input information and the agent architecture. In these initial experiments, we fell back on an idealized model of a network that nevertheless poses the principal learning and coordination challenges of the real-world case.

Each agent uses a modified version of Q-learning (see Equation (4) below) to learn which action to execute in a specific state. The value of this function is the maximum discounted cumulative reward or the value of executing action a in the state s plus the value of following the optimal policy afterward. The action selection strategy during learning is provided by Boltzmann exploration. Boltzmann exploration uses a decreasing factor (T) known as temperature to slowly decrease exploration over time. To measure the learning performance we used accuracy, precision, recall and specificity (Table 1). These four variables give us more information about the relation between False Positives (FP) and False Negatives (FN) and between FN-FP and the correct categorized events (True Negatives and True Positives).

$$\hat{Q}(s,a) \leftarrow \hat{Q}(s,a) + \alpha(r - \hat{Q}(s,a)) \tag{4}$$

Table 1. Performance Metrics

Measure	*Formula*	*Meaning*
Accuracy	(TP + TN) / (TP + TN + FP + FN)	The percentage of positive predictions that is correct
Precision	TP / (TP + FP)	The percentage of positive labeled instances that was predicted as positive. Also defined as *Intrusion Detection Rate*
Recall	TP / (TP + FN)	The percentage of negative labeled instances that was predicted as negative
Specificity	TN / (TN + FP)	The percentage of predictions that is correct

In the simplest experiment, we created a cell with two sensor agents and one RL-IDS agent. We set up the sensor agents to have 2 states (0 and 1). Note that each sensor agent cannot observe the states of other sensor agents and that the combination of all states of the sensor agents represents the global state of the network. In this simple scenario we have 4 states ([0,0],[0,1],[1,0],[1,1]) where state [1,1] represents an abnormal network state that would require an alarm signal from the RL-IDS. The sensor agents have to learn to produce the right signal action to the RL-IDS agent, while this agent needs to learn to interpret these signals. In our basic scenario there are only two sensor signals A and B. The RL-IDS must learn which signals from the sensor agents represent a normal state of the network or a warning state. As it can be observed in Figure 3, the agents in this test are able to detect and categorize the normal and abnormal states of the network.

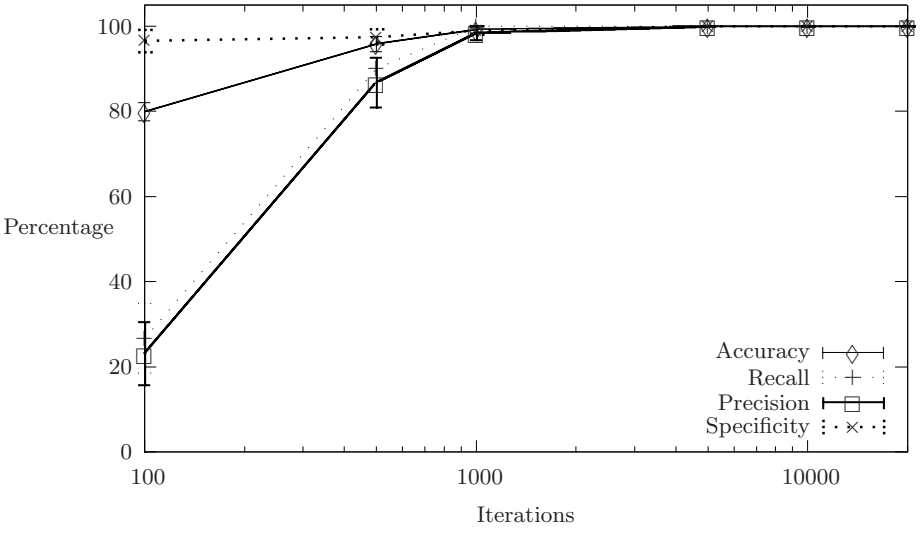

Fig. 3. Two Sensor Agents: After less than 1,000 iterations the RL-IDS agent learns how to identify abnormal activity through the signals from sensor agents

In general the global state of the network is simulated by randomly choosing between normal and abnormal states. Following a uniform distribution of possible states in tests with more than two agents creates a very small number of abnormal states compared with the number of normal states. This distribution of training data biased the agents to learn that the safer action was not to generate any alarm action at all. The result is a low performance in the intrusion detection rate and recall variables as shown in the second row of Table 2 and in Fig.5 To solve this problem we provide a minimum of 25% of abnormal states in the training data. With this new set up, the agents were able to learn to act

correctly with higher levels of accuracy. Still the intrusion detection rate was low. In other words, the agents generated high rates of false positives.

We found that the agents had little time to explore all the no-attack states and to *fix* the Q-values that were miscalculated as result of the credit assignment problem and the partial observation of the environment. To tackle this problem we extended the exploitation phase of the exploration/exploitation strategy to allow the agents to exploit actions and to modify the values of their Q-tables. To carry out this task we divided the exploration/exploitation strategy in two parts. The first part was the initial Boltzmann strategy where agents slowly decrease exploration over time accordingly to a decreasing factor (T). The second part was a total exploitive strategy where agents do not explore actions any more. We denoted this as a *pure exploitive strategy*. The level of *pure exploitation* is given by Equation 5. The results presented in Table 2 and marked as *with pure exploitive strategy* uses a level value of 0.5. In other words the agents explore/exploit 50% of the time accordingly to a Boltzmann strategy and exploit actions the rest of the time. In Figure 4 there is a graphical comparsion between tests with four agents. One of the test was performed with 25% of abnormal activity and Boltzmann strategy (exploitive Level = 1). The other test was performed with 25% of abnormal activity and an exploitive level equal to 0.5. As shown, the use of new strategy (Boltzmann + total exploitive) provided higher values of accuracy (See Table 2 for comparsion between three agents) as well as high values of intrusion detection rates compared with test with only Boltzmann exploration/exploitation.

$$ExploitiveLevel = \frac{Number\ of\ pure\ exploitive\ iterations}{total\ of\ iterations} \tag{5}$$

The next step in our testing was to increase the number of states and review the maximum numbers of agents per cell with acceptable levels of performance. As shown in Figure 5; when we increased the number of agents the levels of precision went down due to high rates of false negatives. Under this assumption we considered that the maximum number of agents per cell is less than 6. As it can be observed in Table 2 the remaining variables had very little effect as the number of agents increased. This is the effect of the previously mentioned problems often found in MARL such as credit assignment, partial observation, curse of dimensionality and mis-coordination penalized with high negative rewards. It is important to note that the number of iterations presented in the graphs of Figure 5 and Figure 6 are per hierarchical level; for a test with 9 agents in 2 hierarchical levels the test needs 10,000 iterations per level (as shown in the figures) or 20,000 iterations in total to reach the performance levels listed.

In order to adapt our architecture to detect abnormal activity on inter-domain networks or in intra-domain networks with geographical zones we develop a hierarchical architecture of agents (See Figure 2). In this new architecture sensor agents and RL-IDS inside a cell learn how to identify local normal and abnormal activity. Once they have learned this, the RL-IDS agents inside the cells send communication signals to the next RL-IDS in the hierarchy which is learning

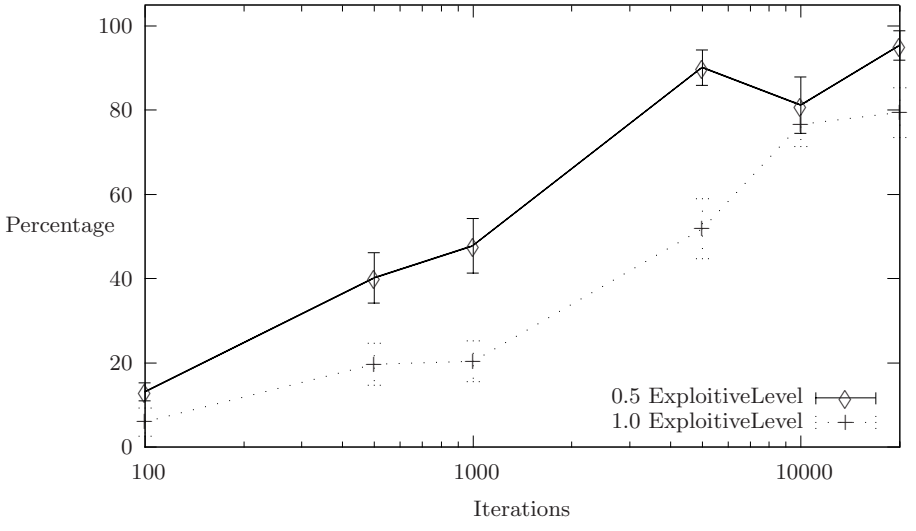

Fig. 4. Intrusion Detection Rate Four Agents with different Exploitive levels of Exploration/Exploitation and 25% abnormal activity

Table 2. Performace (Percentages)

Test	*Accu.*	*Error*	*Prec.*	*Error*	*Recall*	*Error*	*Spec.*	*Error*
Two sensor agents (3)	98.9	1.1	90.0	10.0	90.0	10.0	100.0	0.0
Three sensor agents	96.1	0.8	10.0	10.0	10.0	10.0	100.0	0.0
Three sensor agents (1)(2)	99.9	0.0	92.0	5.0	90.0	7.5	100.0	0.0
Six sensor agents (1)(2)	99.5	0.2	37.9	11.3	100.0	0.0	99.5	0.2
Six sensor agents Hierarchical (9 total) (1)(2)	99.9	0.0	90.0	10.0	90.0	10.0	100.0	0.0
9 sensor agents Hierarchical (13 total) (1)(2)	99.9	0.0	85.0	7.5	100.0	0.0	100.0	0.0
27 sensor agents Hierarchical (40 total) (1)(2)(3)	99.9	0.0	83.0	8.0	100.0	0.0	100.0	0.0

(1) 25 abnormal training, (2) Pure exploitive strategy, (3) 2 states per sensor agent 10,000 iterations in all tests.

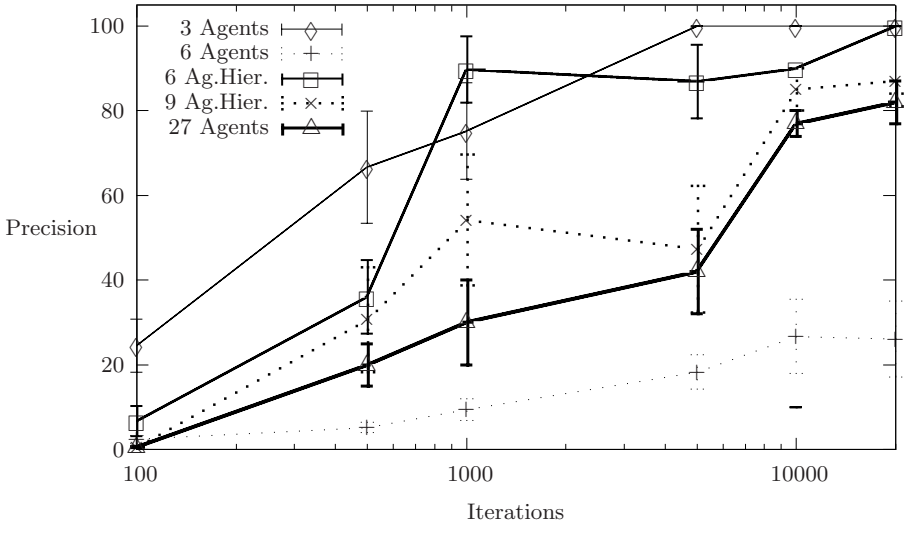

Fig. 5. Comparison of Precision (Intrusion Detection Rate) by number of agents

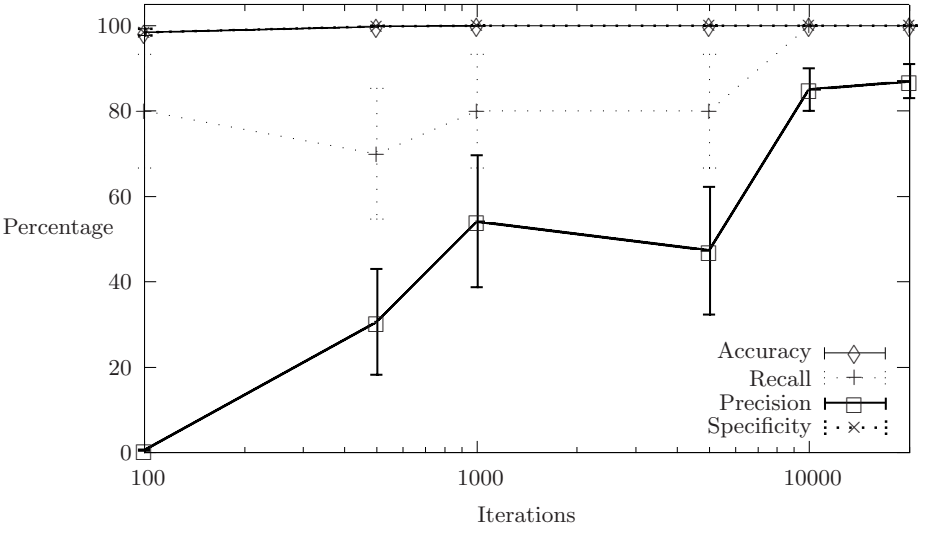

Fig. 6. Performance Metrics with Nine Agents

to signal in turn. This procedure is repeated iteratively until it reaches the last RL-IDS in the topology, i.e. the agent responsible for determining the state of the whole system. We compared the performance of 6 agents using this hierarchical design with 2 levels against the flat approach of 6 agents in one cell. We found

that this architecture presents high levels of performance on all of our metrics than the flat approach.

Additionally, this approach permits the use of more than 6 agents arranged in various hierarchical levels. We expanded the architecture up to 9 sensor agents and 4 RL-IDS agents (13 agents in total) in 2 hierarchical levels and up to 27 sensor agents and 13 RL-IDS agents (40 agents in total) in three hierarchical levels. These tests shown (Figure 6) very acceptable levels of performance on all of our metrics.

5 Conclusion and Further Work

This paper presented RL experiments in an abstract network model where distributed network sensor agents learn to send signals up a hierarchy of agents. Higher agents in the hierarchy learn how to interpret local collected information from these signals and signal an overall abnormal state to the network operator when it is necessary. We presented solutions that enable the agents to learn an accurate signal policy and we have shown that the approach scales up to a large number of agents. In future work we plan to port the abstract network model to a realistic network simulation.

In our work, we used a fairly straightforward Q-update function and simple exploration/exploitation strategy. We intended to use this simple approach and to focus on the hierarchical mechanism to expand our proposal to several connected cells resembling real computer network environments. We do believe that a more complex approach to exploration/exploitation strategy or to calculating the value function may yield similar results but with fewer iterations, more input features and more agents per cell. We also plan to apply some techniques from single-state games [7], hierarchical reinforcement learning [1,5] function approximation [8,13,17] techniques and others [4,14].

References

1. Barto, A.G., Mahadevan, S.: Recent Advances in Hierarchical Reinforcement Learning. Discrete Event Dynamic Systems 13(4), 341–379 (2003)
2. Barford, P., Jha, S., Yegneswaran, V.: Fusion and Filtering in Distributed Intrusion Detection Systems. In: Proceedings of the 42nd Annual Allerton Conference on Communication, Control and Computing (September 2004)
3. Bass, T.: Intrusion Detection Systems and Multisensor Data Fusion. Communications of the ACM 43(4), 99–105 (2000)
4. Chang, T.H., Kaelbling, L.: All learning is local: Multi-agent learning in global reward games. In: Advances in NIPS, vol. 14 (2004)
5. Elfwing, S., Uchibe, E., Doya, K., Christensen, H.I.: Multi-agent reinforcement learning: using macro actions to learn a mating task. In: IROS 2004. Intelligent Robots and Systems (2004)
6. Jennings, N., Sycara, K., Wooldridge, M.: A roadmap of agents research and development. Autonomous Agents and Multi-Agent Systems 1, 7–38 (1998) In: [12]

7. Kapetanakis, S., Kudenko, D., Strens, M.: Learning to coordinate using commitment sequences in cooperative multi-agent systems. In: AISB 2003. Proceedings of the Third Symposium on Adaptive Agents and Multi-agent Systems, Society for the study of Artificial Intelligence and Simulation of Behaviour (2003)
8. Kostiadis, K., Hu, H.: KaBaGe-RL: Kanerva-based generalisation and reinforcement learning for possession football. In: IROS 2001. Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (2001)
9. Mirkovic, J., Reiher, P.: A taxonomy of DDoS attack and DDoS defense mechanisms. ACM SIGCOMM Computer Communication Review 34(2) (April 2004)
10. Moore, D., Shannon, C., Voelker, G.M., Savage, S.: Internet Quarantine: Requirements for Containing Self-Propagating Code. In: INFOCOM 2003. 22th Joint Conference of the IEEE Computer and Communications Societies, March 30- April 3, 2003, vol. 3, pp. 1901–1910 (2003)
11. Neumann, P.G., Porras, P.A.: Experience with EMERALD to DATE. In: 1st USENIX Workshop on Intrusion Detection and Network Monitoring, Santa Clara, California (April 11-12, 1999)
12. Panait, L., Luke, S.: Cooperative Multi-Agent Learning: The State of the Art. Autonomous Agents and Multi-Agent Systems 11(3), 387–434 (2005)
13. Porta, J., Celaya, E.: Reinforcement Learning for Agents with Many Sensors and Actuators Acting in Categorizable Environments. Journal of Artificial Intelligence Research 23, 79–122 (2005)
14. Powers, R., Shoham, Y.: New criteria and a new algorithm for learning in multi-agent systems. In: Advances in Neural Information Processing Systems (forthcoming), Rubinstein, A.: Modeling Bounded Rationality. MIT Press, Washington (1998)
15. Sen, S., Weiss, G.: Learning in Multiagent Systems. In: Weiss, G. (ed.) Multiagent Systems, A Modern Approach to Distributed Artificial Intelligence, pp. 259–298. MIT Press, Cambridge (1999)
16. Siaterlis, C., Maglaris, B.: Towards multisensor data fusion for DoS detection. In: Proceedings of the 2004 ACM Symposium on Applied Computing, pp. 439–446 (2004)
17. Stone, P., Sutton, R.S., Singh, S.: Reinforcement Learning for 3 vs. 2 Keepaway. In: Stone, P., Balch, T., Kreatzschmarr, G. (eds.) RoboCup-2000: Robot Soccer World Cup IV, Springer, Berlin (2001)
18. Sutton, R., Barto, A.: Reinforcement Learning, An Introduction. MIT Press, Cambridge (1998)
19. Wasniowski, R.A.: Multisensor Agent Based Intrusion Detection. Transactions on Engineering, Computing and Technology 5, 110–113 (2005)
20. Yegneswaran, V., Barford, P., Jha, S.: Global Intrusion Detection in the DOMINO Overlay System. In: Proceedings of the Network and Distributed System Security Symposium (2004)