Enhancing Security through the use of Load-Balancing Stochastic Algorithms

RODRIGO FONSECA PIRES, Instituto Superior Técnico, Portugal

Computer networks are growing everyday, as are the demands on performance and reliability. People use wireless ad hoc networks everywhere and perform tasks that deal with sensitive data and require fast responses. The wide spread use of wireless networks makes for an increase in the concern of attacks like the Man-In-The-Middle attack or Denial of Service (DoS). These attacks are very successful when routing protocols aim to increase performance, by prioritizing the use of optimal paths with minimal latency, making the interception of packets easier for an attacker by being trivial to predict.

Many security solutions revolve around authentication methods and cryptography to protect intercepted data. What we propose here is the use of stochastic algorithms with reinforcement learning that reduces the chances of interception in the first place.

We approached this topic with a game theory perspective, defining the problem as a game and a learning algorithm that computes the equilibrium for the current scenarios. In order to assess the different algorithms, we built a framework to run simulations of our game to prove the results experimentally. Finally, we compared our results to deterministic approaches used in state-of-the-art network protocols to show increases in security and minimal impact on the performance. With these results we were able to conclude that learning algorithms are effective at protecting routers from targeted attacks, while keeping the network average performance similar to deterministic routing algorithms or even improving performance thought indirect load balancing.

Keywords: Network protocols; Network routing; Stochastic algorithms; Security; Game theory.

1 INTRODUCTION

Throughout the years computer networks have been growing exponentially [31, 32] in size and complexity while, at the same time, the demands in performance and reliability increase as well. This constant growth puts pressure on researchers to continually work on network routing to allow the networks to keep expanding without losing their qualities. As of late, computer networks also have the necessity to be mobile and adaptable, as mobile devices become increasingly more popular[33]. This mobility is assured by wireless networks, which allow and are prepared for constant changes in their topology, with nodes connecting/disconnecting and moving around.

Traditional wired networks have a constant topology, that only occasionally changes when new nodes are added or some are removed/fail. This persistence allows routing protocols to compute optimal paths between nodes and store them as, they either never change, or do so at such slow rate that recalculations are scarce. However, when talking about wireless networks, adaptability is key because optimal paths computed at the present moment may become sub-optimal or even broken due to the volatility of this type of networks. Thus, wireless networks rely on cooperation between nodes to constantly keep every node up to date, but this cooperation with unknown participants brings major security concerns, as they open the doors to attacks such as packet interception, which can be used to either stop packets from reaching their destination or to "snoop" information from them. These attacks can even be more effective if the malicious node is strategically positioned in a optimal path between nodes, as the majority, if not all of the packets transmitted will pass through them.

Nowadays networks have a lot of built-in security measures to prevent malicious activity, like eavesdropping attacks. These measures are usually related with authentication and authorization methods, using techniques like cryptography to encrypt messages. Although this does not prevent the packets from being compromised, it secures the information in them. What we propose here is a way to enhance security by routing means, complementing existing security measures in place, to prevent the packets from being intercepted in the first place.

The approach we propose here is to use stochastic routing techniques to make use of multiple paths between source and destination, in order to spread the packets unpredictably through the network. By using a probability distribution to choose the next hop for a packet instead of precomputed optimal paths, we make it impossible for an attacker to accurately predict which nodes the packets will pass through. On top of this, we also introduce reinforcement learning into the routing algorithm, to constantly update the probabilities and adapt to the current state of the network and possible attackers. By making the nodes constantly update their routing tables we can quickly adjust to changes in the network and guarantee close to optimal performance when the network is being compromised as well as when everything is fine and there are no malicious nodes.

To work on the proposed routing algorithm, we took a game theory approach to the problem. We defined a game that emulates a scenario where there is a network being used my multiple players and an attacker is introduced. This malicious player will target one of the users in the network, the defender, and try its best to intercept packets while the network learns to adjust to the malicious activity. By defining the problem as a stochastic game we will be able to compute its Nash Equilibrium, which allows us to compare multiple algorithms in different conditions. When the game converges we are able to see the worst case scenario for that specific situation and consequently analyse which approaches better respond to an attacker intervention.

1.1 Main Objectives

The main goal of our work is to propose a way to enhance security in computer networks through making use of multiple routes between source and destination, without negatively impacting the performance of the network. We have set some objectives we want to work on in order to achieve this goal:

- Develop a framework where it is possible to generate different network graphs with customizable topology
- Develop the simulation for the game
- Gather data from the simulations for statistical analysis

- Analyse and compare various strategies and algorithms through simulations to find the best suitable to increase security in computer networks
- Understand the benefits and limitations of using stochastic routing algorithms
- Present possible ways of continuing further development for future researchers on the topic

2 NETWORK ROUTING PROTOCOLS

We will first be looking at the evolution of Mobile Ad-Hoc NETworks (MANET) through some examples that have been compiled in [11]. With this analysis we will see that some of the challenges and limitations of MANET have been haunting researchers in the area of network routing since they first started getting developed.

MANET have a lot of advantages when compared with more traditional networks, their infrastructure is easier and cheaper to set up; they offer more fault tolerance due to their decentralized nature; the usage of multiple hops reduces the risk of bottlenecks and, obviously, they offer the mobility that wired networks cannot.

However, it is not all advantages as MANET still lacks in reliability due to its ever changing nodes and connections and, because each router needs to perform routing tasks, they require more memory and computational power, which contradicts their mobility advantage, as we want mobile devices to be small and lightweight, but that limits their computational resources and battery power. MANETs are also hard to scale, because as we increase the number of nodes in the network, the amount of processing power and memory necessary to keep the routing efficient also increases, putting more work in each individual node.

There are some other factors that negatively affect the performance of MANETs like the limited range of wireless transmission and the constant flux of in and out of nodes in the network due to its mobile characteristics, but here our main focus is security. Due to the mobile and wireless characteristics of MANETs, malicious nodes can enter the network at any time and launch attacks such as man-in-the-middle and Denial of Service(DoS). The security of the network is usually maintained by security layers, like authentication [12] and cryptography [13], but what we propose is a way to increase security through routing methods.

Some of first MANET algorithms were proposed in the 90's like the Destination Sequenced Distance Vector(DSDV) [14] and the Dynamic Source Routing(DSR) [15]. Both algorithms worked with precalculated routes, which means that when a packet is sent, the whole route is already determined. In order to compute and maintain paths from and to multiple nodes, these algorithms used a lot of memory (to keep the routes cached in routing tables) and needed a lot of bandwidth every time recalculations of paths were necessary. The main difference between the two is in the way that they compute new routes when a change happens in the network topology, DSDV [14] being proactive by constantly exchanging updates between nodes and DSR [15] being reactive by making use of acknowledgements and triggering rediscovery of paths when a lot of error messages are received.

After learning from the previous algorithms researchers came up with better performing protocols, one of which was the Ad-hoc On-demand Distance Vector(AODV) protocol [16], improving upon the DSDV protocol [14] and taking the route discovery through using on-demand route requests from DSR [15]. But it still lacked a major feature, multicast support, that would be later implemented on a protocol that was an extension of AODV, the Multicast Ad-hoc On-demand Distance Vector(MAODV) [17]. This new functionality improves the protocol by enhancing communication with multiple nodes and increasing routing knowledge while also reducing control traffic overheads [18].

After all these developments made in order to improve the reliability and performance of MANETs there was still a major issue to be addressed, which was security. Wireless networks are more vulnerable to a wide variety of network attacks than wired networks due to its non physical way of transmission. The original AODV protocol had no security measures in place and as such, was extremely vulnerable to malicious activity, like tempering with the control headers that were used by the nodes to exchange network knowledge. To try and mitigate this problems, researchers developed numerous security and authentication methods for MANETs [12] as well as continuing to iterate over previous protocols, such as the Security-aware Ad-hoc On-demand Distance Vector(SAODV) [19] and Adaptive Secure Ad-hoc On-demand Distance Vector(A-SAODV) [20], both designed after the AODV protocol, but with security in mind.

Although a lot of progress has been made to improve security in wireless networks [21], the solutions mostly revolve around authentication and authorization methods, that prevent attackers from forging or changing messages and inject them into the network. What we want to propose here is a way to improve security through routing methods, making it harder for a malicious node to have the chance of intercepting packets in the network or having to expend a lot more resources to do so.

3 REINFORCEMENT LEARNING

Now we will take a look at works developed in the machine learning area, more specifically, reinforcement learning methods and techniques. Agents using this form of learning rely on getting rewards (or penalties) for their actions, resulting in them adjusting their strategies to try to maximize some sort of score or minimize some cost [22].

3.1 Learning algorithms

Xiaosong Lu and Howard M. Schwartz designed and presented a decentralized learning algorithm [4] that makes use of the Lagging Anchor algorithm [5] [6] to converge to a Nash-Equilibrium in two-player zero-sum matrix games, be it a Pure Strategy or a Mixed one. They also proved that the algorithm converges into NA with only the knowledge of the action and its reward. To achieve this, the authors studied other algorithms that served as inspiration for the development of their own and divided them into two groups. Of the four presented here, the first two are based on learning automata, which have the objective of learning the optimal strategy by updating its action probability distribution based on the environment response [4]. The latter two are based in gradient ascent learning

methods that consist in updating the player strategy in the direction of the current gradient with some small step size [8].

(1) Linear Reward-Inaction Algorithm. The Linear Reward-Inaction algorithm is defined in [7] and consists on a reinforcement learning method that attributes a probability to each player's actions and after each play, if the reward was positive, the probability of choosing the same action in the future increases, otherwise nothing changes. It is also proven in [7] that if all players in a matrix game use this algorithm, then it "guarantees the convergence to a Nash Equilibrium under the assumption that the game only has strict Nash equilibria in pure strategies" [4].

(2) Linear Reward-Penalty Algorithm. The Linear Reward-Penalty algorithm is similar to the previous Linear Reward-Inaction algorithm in the sense that it gives each of the player's actions some probability of being executed and after each play, if the reward is positive, it will increase the respective action's probability to be picked. However, if the action fails to give a positive reward, instead of doing nothing, we decrease the probability of using the same action in the future. This algorithm can only be applied to two-player zero-sum games that have Nash Equilibrium in fully mixed strategies [7]. It is also important to note that the Linear Reward-Penalty algorithm "can guarantee the convergence to the Nash equilibrium in the sense of expected value, but not the player's strategy itself" [4].

(3) WoLF-IGA Algorithm. Win or learn fast-infinitesimal gradient ascent algorithm [9] can be applied in two-player two-action matrix games. The algorithm constantly updates the player's strategy based on the current gradient and some variable learning rate. This learning rate is smaller when the player is winning when compared with the value when the player is losing. This algorithm "can guarantee the convergence to a Nash equilibrium in fully mixed or pure strategies for two-player two-action matrix games" [4], but is not decentralized because the player needs to know its own reward matrix and the opponent's selected action.

(4) The Lagging Anchor Algorithm. Introduced by Dahl [5] [6], the Lagging Anchor algorithm, like the previous one, updates the player's strategy according to the gradient but, unlike the WoLF-IGA algorithm, in this one "the limitation on each player's actions to be two actions is removed and the convergence to the Nash equilibrium in fully mixed strategies is guaranteed" [4], but it still requires the knowledge of the reward matrices of the players, thus this algorithm is also not decentralized.

After studying the previous algorithms, the authors in [4] designed the L_{R-I} Lagging Anchor algorithm, focusing on "both the player's current strategy and the long-term average of the player's previous strategies at the same time" [4]. They basically took the way in which the player strategy is updated in the Linear Reward-Inaction algorithm and added the lagging anchor term from the Lagging Anchor algorithm [4]. They also proved that this new algorithm guarantees the convergence to NA in both pure and fully mixed strategies in two-player two-action zero-sum games.

What makes this algorithm especially useful is the fact that it is decentralized, which is the type of learning we want to apply to our network so that each node can be independent from all others. Also, because we will be working with an incomplete information game, only needing to know the action and consequent reward is a really good reason to try to make use of this algorithm.

3.2 Artificial Barriers

Anis Yazidi, Daniel Silvestre and B. John Oommen discuss and propose the use of Artificial Barriers in Learning Automata to solve Two-Person Zero-sum Stochastic Games [1]. LA had previously been used [2] to compute the Nash-Equilibrium of this type of games under limited information. However, they noted that these existing algorithms often focused in the cases where the Saddle Point existed in a Pure Strategy. This can be a problem, especially on the game we are trying to solve in our research, because if the game's strategy converges into a single action, it becomes easy to predict the other player's move. So, they proposed a solution that makes it possible to converge into an optimal mixed Nash-Equilibrium even if the game has no Saddle Point when a Pure Strategy is invoked. This was accomplished with the use of Artificial Barriers that prevent the game from being absorbed into a Pure Strategy. Although a similar method had already been proposed by Lakshmivarahan and Narendra [3] 40 years ago, this new approach is, as described by the three researchers, "more elegant and required less parameter tuning".

In section "IV.Simulations D.Real-life Application Scenarios" [1] they suggest the use of their leaning algorithm in security games and describe a very similar, although simpler, version of the game we are trying to investigate in our proposal. The main reason for this suggestion is that a mixed Nash-Equilibrium is preferred over Pure Strategies in security games, because it makes it a lot harder to predict what the other player will do.

4 NETWORK GAME

In this chapter we will go over the problem at hand, describing it and identifying the elements and rules that compose it. Then we will present the software solution that was developed to help us study the problem.

4.1 Scenario Overview

First, we will present an example scenario to help illustrate the problem. As shown below, in Figure 1, our problem consists in a computer network where there are multiple users exchanging packets. At some point, an Attacker infects a router in the network. This malicious user has the objective of intercepting packets from a specific user, we call it the Defender, from reaching their destination, the Target User.

What we want to study in this scenario is how can the routing algorithm adapt to help mitigate the negative impact that the Attacker creates on the Defender; without needing to know who the Attacker and Defender are, or even if there is a malicious user in the network.

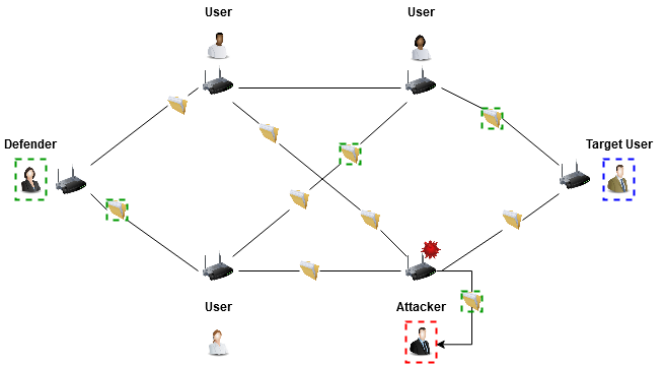

Fig. 1. Scenario example

4.2 Environment

First let us describe the game environment in which the players will compete. Since we are trying to simulate what would happen to network packets P in a computer wireless network, what makes the most sense is representing the game world as a graph $G = (V, E)$, in which the vertex V_i represents the network node number i and the edge $E_{i,j}$ represents the network link connecting nodes i and j. Although this is a simulation, we want to apply some restrictions on our game's network components to closer reassemble a real life scenario, thus making the game more realistic and the obtained results more reliable.

4.2.1 Network Node. As stated before, a network node is represented by the vertex V_i of the graph G and works both as a redistribution point and a communication endpoint. This means that all vertices are capable of redirecting incoming packets as well as generating new ones and injecting them into the network.

A node can only send one packet per each unit of time and has a waiting queue that can have a limited capacity. If a node has a full waiting queue it will discard some packets following some policy, for example:

- Discards the latest packets that arrived at the node (gives priority to the packets that arrived first)
- Discards the oldest packets (the ones with lower TTL, meaning that they have been circulating for more time)
- Discards the packets that are further away from their destination node (because they could be more likely no not make it)

Aside from this discard policy, each node will also have a routing table that will determine which link are the packets sent to. We will talk more about this in a following section.

4.2.2 Network Link. A network link that connects node i and node j is represented by the edge $E_{i,j}$ and it works as the medium in which packets can be transmitted from one node to another. Every link has a positive integer length l that represents the number of units of time that a packet takes to be transmitted from node i to node j and vice-versa.

4.2.3 Network Packet. Network packets P are the units of data that transverse our network and carry the data [1] to be transmitted from some node V_i to some node V_j and metadata that takes part in helping to deliver the packet to its destination.

Meta data that packets contain:

- Source/Destination node number
- TTL
- Route taken thus far

4.3 Agents

Our game is a competitive game, where two agents are trying to maximize their own score. On one hand we have what we will call the defender and it represents a "normal" user of the network that has access to a node and wants to send packets to some other node; on the other hand we have the attacker which represents a "malicious" user of the network that is trying to sabotage the communication of the other agent.

4.3.1 Defender. The defender has access to some node $V_{defender}$ in the network where he can inject packets. In each unit of time, the defender can either send one packet or do nothing. The defender has no idea that is being attacked and as such considers everything that happens to its packets a consequence of the environment.

4.3.2 Attacker. The attacker infects one node in the network that is neither the node that the defender has access to or the node that the defender wants to send packets to (as that would make the attacker have an overwhelming advantage). The attacker has control over the infected node and can:

- Generate packets and inject them in any link connected to the infected node
- Control the waiting queue of the node and the next packet to be sent
- Drop packets from the defender

4.3.3 Normal user. Normal users perform the same actions as the defender, but are not being targeted by the attacker. Their role is to send packets to populate the network and make for a more realistic and complex scenario. These agents could be seen as a part of the environment.

4.4 Game

A game has two phases: the "preparation phase"(4.4.2) and the "attack phase" (4.4.3). A game is separated in multiple "rounds" and each round represents what happens in a unit of time.

4.4.1 Round. A round represents everything that happens in one unit of time. In a round:

- Every network user (including the defender and the attacker) can send one packet through one of its links
- Every packet that is currently on a link moves one step forward
- Every packet that either reaches its destination or gets dropped (by running out of TTL or, in the case of the packets sent by

[1] The data part of the packet is irrelevant in our simulations; we will assume that every packet has "valid" data, because the point here is to increase security through routing means and not data verification.

the defender packets, gets attacked) sends an acknowledgment[2] to all the nodes it passed through

4.4.2 Preparation phase. At the start, there is no attacker and the nodes have empty routing tables. In this phase the goal is to populate these tables and establish the best routes in a normal situation. This is done by generating packets in every node and send them to random destinations.

In this case, without the disruption of the attacker we expect the routing tables to tend to favor the shortest paths if the number of packets sent is not too high. Otherwise, if the network is crowded with packets we expect the routing tables to tend to favor splitting them between multiple links to take advantage of multiple paths.

After some arbitrary number of iterations the tables will reach some equilibrium and won't change much anymore (if the rate at which packets are generated is constant). When this equilibrium is achieved we can introduce the attacker.

4.4.3 Attack phase. In this phase the attacker starts by choosing a node to infect (that is not the source/destination of the packets sent by the defender) and some link connected to the same node. Then the attacker starts redirecting packets that reach the infected node and dropping packets that come from the defender.

In a similar fashion to the previous phase, the network will eventually reach an equilibrium in which it wont change its routing tables much anymore, we can end the game at this point and collect relevant data to compute both player's "scores".

4.5 Scoring

At the end of a game we want to attribute a score to both the attacker and the network (represented by the defender). This will let us compare different strategies and algorithms across different games. On one hand, the attacker will be rewarded by how much he was able to disrupt the packets sent by the defender, on the other hand the network will be rewarded by how much it was able to "protect" the defender from the disruption caused by the attacker.

To attribute these scores we will use multiple metrics, like the packet loss experienced by the defender (both from direct attacks and timeouts) and the time, paths and number of hops that packets took to arrive to the destination (minimum/maximum, average, standard deviation, etc.).

4.6 Formal solution

Now we will define the proposed solution as a Game Theory game. To define a game we need to specify the Players, their available Actions, the Payoffs for each action and the Information that they have at each point, known as PAPI (players, actions, payoffs, and information) [10]. We will be referring to the proposed game as "Stochastic Routing Game". The game only truly starts when we have both the $attacker$ and the $defender$ in the network, which we refer to as the "Attack phase" described in Section 4.4.3.

In the Routing Game we will define the players as attacker and defender. Both of the players will occupy and play as a router (node in the graph), but there are many more routers in the network. These other routers will have the same actions and information as the defender, but will not be targeted by the attacker and will not have the objective of maximising some score/utility. As such, for simplicity of the model, we will treat all other routers as the "Nature" [10]. They will be simply following the routing algorithm controlled by the defender and inject packets into the network randomly.

4.6.1 Defender. The defender action set will be dependent on how many links the node he is occupying is connected to, let us call it $V_{defender}$. If the node $V_{defender}$ has n links, then the defender action set is defined as $A_{defender} = \{a_i\}$, $(i = 1, ..., n)$, where each action a_i represents sending a packet through the ith link of node $V_{defender}$. We also define the target node for the packets sent by the defender as $V_{target} \neq V_{defender}$.

Each action a_i will also have some probability p_i of being executed by the defender and the set of probabilities is defined as $P_{defender} = \{p_i\}$, $(i = 1, ..., n)$, where each probability p_i represents the likelihood of sending the packet through link i of node $V_{defender}$.

As for the payoff of each action, we need to first make the distinction between the expected payoff and the actual payoff. Not only the defender cannot know the exact payoff some action will give when it is played out, he also does not know right after executing the action. This is due to the fact that we cannot know how successful or unsuccessful the network was at delivering a packet before it either reaches the destination or is dropped. Because of this delay in receiving the reward, the defender needs to have a set of expected payoffs for each action a_i, defined as $E_{defender} = \{e_i\}$, $(i = 1, ..., n)$, where each expected payoff e_i represents an estimate of how successful the network will be at delivering the packet through link i of node $V_{defender}$, which directly correlates with the probability p_i.

Regarding the information the defender (or any other node that is not the attacker) has available, he only knows the final reward (score) of every packet he sent. This means that when a packet sent by the defender reaches its end, successfully or not, the defender gets an acknowledgement[3] with the reward and associates it with the action a_i that was executed to send the respective packet. This also means that the order in which the defender gets the rewards for the actions he executes is not guaranteed to be the same as the actions were played out. Because of this limited amount of information, we can even consider the attacker as part of the "Nature" from the perspective of the defender, as he is unaware that he is being targeted.

4.6.2 Attacker. The attacker has a decision to make before the start of the game that will influence the game until the end. After the "Preparation phase", described in 4.4.2, the attacker has to pick a node to occupy, (that is neither the defender or the target of the packets sent by the defender), let us call it $V_{attacker} \neq V_{defender} \neq V_{target}$.

After the making this initial decision, the game begins (start of the "Attack phase", see 4.4.3) and the attacker has one action set dependent on how many links the node $V_{attacker}$ is connected to. If the node $V_{attacker}$ has m links, then the attacker action set is defined

[2]We are not going to simulate acknowledgment packets (might be something to experiment on in the future)

[3]As explained in note 1 in 4.4.1, this acknowledgements are made "magically" in our game for the sake of simplicity.

as $A_{attacker} = \{b_i\}$, $(i = 1, ..., m)$, where each action b_i represents sending a packet through the ith link of node $V_{attacker}$.

Each action b_i will have some probability q_i or r_i of being executed by the attacker depending on the origin of the packet at play.

If the packet has as its source the node $V_{defender}$, the set of probabilities is defined as $P_{attackerD} = \{q_i\}$, $(i = 1, ..., m)$, where each probability q_i represents the likelihood of sending the packet through link i of node $V_{attacker}$.

If the packet has as its source some node $V_i \neq V_{defender}$, the set of probabilities is defined as $P_{attackerN} = \{r_i\}$, $(i = 1, ..., m)$, where each probability r_i represents the likelihood of sending the packet through link i of node $V_{attacker}$. (This separation gives the attacker means to strategize in a way that he can maximize the damage caused to the defender while "doing its job" as a normal router so that the other routers do not start avoiding sending packets through $V_{attacker}$.)

Similarly to the defender, the attacker has to make his decisions based on an expected payoff, because he can only know the actual payoff of executing an action after the affected packet reaches its end. As such, the attacker needs to have two sets of expected payoffs for each action b_i, dependent on the origin of the packet at play.

If the packet has as its source the node $V_{defender}$, the set of expected payoffs is defined as $E_{attackerD} = \{f_i\}$, $(i = 1, ..., m)$, where each expected payoff f_i represents an estimate of how unsuccessful the network will be at delivering the packet through link i of node $V_{attacker}$.

If the packet has as its source some node $V_i \neq V_{defender}$, the set of expected payoffs is defined as $E_{attackerN} = \{g_i\}$, $(i = 1, ..., m)$, where each expected payoff g_i represents an estimate of how successful the network will be at delivering the packet through link i of node $V_{attacker}$.

In regards to the information available to the attacker, he only has knowledge of the final reward of every packet he sent (same as all other nodes). This means that he has the same amount of information has the defender, except for the fact that he knows who the defender is (otherwise he would not be able to target him) and can differentiate between packets sent from the defender and all other nodes.

4.6.3 Game example. Let us now present an example of the described game. In Figure 2 we can observe a game in the Attack Phase (Section 4.4.3) that has been running for some time. The network has 6 routers, where V_D represents the defender, V_T the destination for the packets created by the defender, V_A the attacker and V_1, V_2, V_3 are normal users. The values in each extremity of a link represent the probability that the node closest to the value has to send a packet through the link, when the destination of said packet is V_T. As an example, the routing table for V_1 is shown in Table 1 for this moment in the game.

Table 1. Routing table for node V_1

	Link to V_D	Link to V_A	Link to V_2
Router ?	...	...	...
Router V_T	0.1	0.3	0.6
Router ?	...	...	...

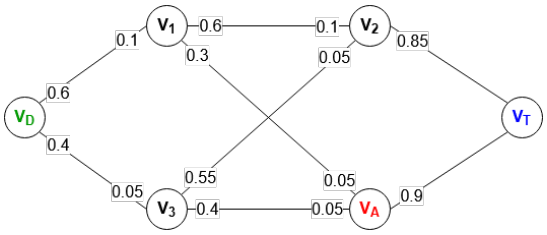

Fig. 2. Game example

5 STOCHASTIC ROUTING GAME DEVELOPMENT

In this section we will present the software project developed[4] to facilitate the study of different network and agent configurations to test and evaluate the proposed stochastic routing game. We will be going through what was developed and how it works, as well as suggestions for improving the project and further development ideas.

5.1 Used Technologies

First let us look at what technologies were used to develop the solution. The solution is written for the most part in C# and developed in a visual studio environment. The program interface is a Windows Presentation Foundation(WPF) project, which can only be ran on Windows systems, but all other supporting projects are written in .NET Standard 2.0 which can be compiled to run in any system. This means that if you want to run the solution in another operating system, you will need to implement a new interface for it.

Other than standard Microsoft libraries, the WPF project also makes use of the ScottPlot.WPF[28] library to display all the graphs used to present the simulations' data.

There is also an optional feature that makes used of the Python3 library NetworkX[29] to generate graphs given some parameters. This feature is integrated in the main solution, but you will need Python3 installed on the machine where you are running the program.

5.2 Architecture Design

Let's us analyse the overall architecture design of the solution.

The solution is composed of seven projects and can be divided in three major parts:

- User interface: NetworkGameFrontend
- Backend services: NetworkGameBackend, NetworkGameDataCollector and NetworkGenerator
- Libraries: Network, NetworkUtils and PythonIntegration

The user interface contains the executable part of the solution, where the user can see, interact and control the application; the backend services have the logic to execute the main functionalities of the application and the libraries consists of data structures and useful functions used across all backend services.

[4]You can view the code at https://github.com/RodrigoPires190776/NetworkGame

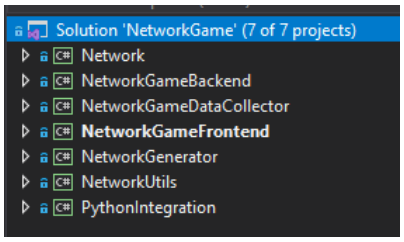

Fig. 3. Visual Studio Solution

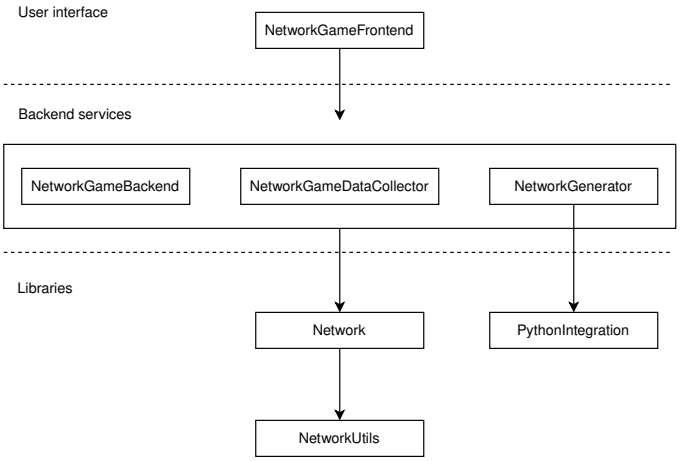

Fig. 4. Architectural design

5.3 Software solution overview

In order to investigate our proposition, we developed an application that allows us to simulate networks and test the behavior of different algorithms by running multiple simulations as games in Game Theory. The simulations ran in our application have 3 main characteristics that facilitate the study of our proposition:

- Instead of running a single game at the time, our application allows us to run any number of games in parallel with the same configurations and combine the results, in order to reduce the uncertainty that comes from using stochastic algorithms.
- It is possible to configure multiple behaviors of the network, not only the routing algorithms, but others like how the routers create packets and how the routers choose which packet to send next. It is also easy to expand these behaviors and create more as all the code of the application is accessible.
- Data collection and presentation are deeply integrated in the simulations, which allows us to visualize just about any data we need from the games. This feature is also expandable with new graphs and types of data that we might want to study.

6 SIMULATIONS AND RESULTS

After developing the our application we performed multiple simulations, where we tested and analysed various scenarios. We started by doing 2 tests using the random routing strategy and variant packet creation probability to serve as benchmarks for the other simulations:

- No route discovery : Allows us to analyse how the network would perform in complete randomness.
- Best path only discovery : Lets us see how the network performs using a deterministic approach that only makes use of the shortest paths.

Following these tests, we ran network game simulations using the linear reward penalty routing algorithm under different circumstances and compared the results between the different runs and the previous 2 benchmark tests:

- Low packet creation probability : Shows the differences between learning rates in a network with low amount of traffic.
- Medium packet creation probability : Shows the differences between learning rates in a network with medium amount of traffic.
- Medium packet creation probability with high penalty rate : Lets us analyse the differences in changing only the reward rate, while keeping a high penalty rate in a network with medium amount of traffic.
- High packet creation probability : Allows us to see the differences between packet picking strategies in a network with high amount of traffic.

The simulations produced data presented in the form of graphs by the application we developed. The data collected was then compiled in tables, with simulations with different configurations side by side for easier comparison. By analyzing this results we were able to draw some conclusions, like:

- The rate at which the network is able to "learn" greatly correlates with how many packets are being sent (how many chances it has to learn) and the value of the learning rates (how much do the values in the routing tables change each time it learns).
- When the network is not crowded (the probability of a router to create a packet is low), the values in the routing tables converge to mainly prefer the shortest paths, becoming almost a form of deterministic routing. However, as the network becomes more crowded, the values in the routing tables tend to be more distributed; because the majority of shortest paths contain a small set of central routers that quickly become overwhelmed if all packets in the network are sent through them, making it a better option to make use of longer paths to avoid spending a lot of time in waiting queues.
- In less crowded networks, the use of stochastic routing tends to perform worse than simply using the shortest paths when there is no attacker. As the network becomes more crowded, the stochastic routing approaches the deterministic routing in performance and even surpasses it, as it provides a form of load balancing, preventing more central routers of being clogged.

- Overall, the use of stochastic routing is a positive for a router that is being targeted by an attacker, even when the network average performance is worse.
- Other than the routing, other decisions, like the order in which the packets are sent from the waiting queues, can greatly affect the performance of the network and its ability to "protect" the defender.

7 CONCLUSION

With computer networks expanding in both size and complexity, as well as being used in more critical areas, like online banking, it is important to keep reinforcing the security of these systems, without taking a toll on the performance. Networks nowadays are very secure against attacks that want to steal sensitive data, with the use of authentication methods and cryptography, but are still vulnerable to attacks that attempt to disrupt their normal operation like packet dropping attacks and denial of service (DoS).

With our work we aim to increase security in computer networks, not by making messages harder to decrypt, but by making it harder to intercept them in the first place. To achieve that, we proposed the use of stochastic routing protocols that make use of multiple paths in the network, decreasing the chance of all packets getting intercepted by an attacker, while trying to maintain or even increase the performance of the network.

7.1 Summary

In summary, we developed an application to simulate computer networks as games and present the data from those simulations. With the results we were able to analyze the impact that different strategies and algorithms have on the performance of the network and the protection that the network is able offer to a node that is being targeted by an attacker.

We saw that, utilizing learning algorithms to dynamically change the probabilities of choosing the next hop significantly increases the amount of packets that a network is able to deliver when a user is being targeted by an attacker that aims to remove its packets from the network; more than doubling it in some cases.

We also noticed that the impact on the performance varies, mostly depending on how busy the network is; when the network does not have many packets in circulation, deterministic routing prioritizing the shortest routes yields faster deliveries in general; however, in networks with a high amount of traffic, utilizing stochastic routing to make use of different paths that may not be optimal, increases the effectiveness of delivering packets when compared to more deterministic approaches.

With our results we were able conclude that, if security is the main concern in a network, utilizing stochastic routing is a valid option to consider, because it increases the robustness of the network while, in the worst case, having a minimal impact on the performance and, in the best case, even increasing the network performance through the indirect loading balancing that making use of multiple paths provides.

8 FUTURE WORK

In this section we will suggest what could possibly be done in the future to continue the work in this area, with the help of the work that was done.

8.1 Improving the application

Concerning the application that was developed, some points that could be worked on are:

- Documentation; not much has been written about the code solution itself, apart from the main document and some comments on the code.
- Application UI; the UI was developed enough to be usable, but some work can be done to make it more user friendly and a better experience overall to use.
- In the main document we brought up a possible memory concern regarding the way the routing table is implemented, as its size grows exponentially with the number of nodes in the network. One way to solve this problem is to group nodes that are close together in a region. The routers then have a table where each entry is a router in their region and another table where each entry is a region that is not their own. This way it is possible to massively reduce the number of entries each routing table needs. This solution can also be implemented with any number of levels, meaning that a router can be part of a region, sub-region, sub-sub-region...
- Implementing more strategies and functionalities that future investigators might find useful to study
- Implementing new graphs and collect different types of data to better study the results of changing strategies and configurations
- Make the simulations more realistic; like for example attributing a more realistic amount of time(cycles) to different actions (travel time between routers, time to send a packet, time it takes for the router to make decisions, etc.).

8.2 Further investigation

In regards to the matter of the study, the use of stochastic routing algorithms to enhance network security, a lot more could be explored with the help of the implemented solution:

- Simulations with different networks; varying in both size and topology.
- Studying new routing strategies using other learning algorithms; like the Lagging Anchor Algorithm.
- Studying new packet creation strategies; like creating packets following some sort of distribution or even a way to simulate dividing and replicating packets so that some packets can be intercepted but the message still gets fully transmitted (see hamming codes [30])
- Studying new packet picking strategies; like some "smart" way of choosing the packets to send in order to minimize the quantity of packets dropped.

The study of all this possibilities could lead to a real world implementation of a stochastic routing algorithm in certain scenarios where the concerns with packet interception are high and security is the highest priority.

ACKNOWLEDGMENTS

I have finally made it to end of my academic journey (at least for now). I have spent the last 5 years studying at Instituto Superior Técnico and, although it was not an easy task, I really enjoyed my time here. Throughout the course of both my Bachelor's and Master's degree I was able to acquire not only a lot of technical knowledge in the computer science field but also friendships and connections that I will cherish for life. For allowing me to achieve all of this, I would like to thank the following people.

I would like to thank my family for the support they gave me my whole life and especially the past 5 years. A special thanks to my parents who made it possible for me to move to Lisbon and attend university, although it being a massive financial and mental burden, they never made me worry about those problems and allowed me to focus on my studies.

Thank you to all my friends who accompanied me in this journey, we joined IST in the same year and remained close until the end. I would also like to thank all the teachers that helped me grow in all this years and gave me the knowledge for me to be where I am at today.

To each and every one of you – Thank you.

REFERENCES

[1] A. Yazidi, D. Silvestre and B. J. Oommen, "Solving Two-Person Zero-Sum Stochastic Games With Incomplete Information Using Learning Automata With Artificial Barriers," in IEEE Transactions on Neural Networks and Learning Systems, doi: 10.1109/TNNLS.2021.3099095.

[2] P. Sastry, V. Phansalkar, and M. Thathachar, "Decentralized learning of nash equilibria in multi-person stochastic games with incomplete information," IEEE Transactions on systems, man, and cybernetics, vol. 24, no. 5, pp. 769–777, 1994.

[3] S. Lakshmivarahan and K. S. Narendra, "Learning algorithms for twoperson zero-sum stochastic games with incomplete information: A unified approach," SIAM Journal on Control and Optimization, vol. 20, no. 4, pp. 541–552, 1982

[4] Lu, Xiaosong and Howard M. Schwartz. "Decentralized learning in two-player zero-sum games: A LR-I lagging anchor algorithm." Proceedings of the 2011 American Control Conference (2011): 107-112.

[5] F. A. Dahl, "The lagging anchor model for game learning — a solution to the crawford puzzle," Journal of Economic Behavior & Organization, vol. 57, pp. 287–303, 2005

[6] F. A. Dahl, "The lagging anchor algorithm: reinforcement learning in two-player zero-sum games with imperfect information," Machine Learning, vol. 49, pp. 5–37, 2002.

[7] M. Thathachar and P. Sastry, Networks of Learning Automata: Techniques for Online Stochastic Optimization. Boston, Massachusetts: Kluwer Academic Publishers, 2004.

[8] S. P. Singh, M. J. Kearns, and Y. Mansour, "Nash convergence of gradient dynamics in general-sum games," in Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence, San Francisco, CA, USA, 2000, pp. 541–548.

[9] M. Bowling and M. Veloso, "Multiagent learning using a variable learning rate," Artificial Intelligence, vol. 136, no. 2, pp. 215–250, 2002.

[10] Rasmusen, Eric (2007). Games and Information (4th ed.). ISBN 978-1-4051-3666-2.

[11] Alex Hinds, Michael Ngulube, Shaoying Zhu, and Hussain Al-Aqrabi, "A Review of Routing Protocols for Mobile Ad-Hoc NETworks (MANET)," International Journal of Information and Education Technology vol. 3, no. 1, pp. 1-5, 2013.

[12] R. Akbani, T. Korkmaz, and G .V. S. Raju, "HEAP: A packet authentication scheme for mobile ad hoc networks," Ad Hoc Networks, vol. 6, no. 7, pp. 1134–1150, 2008.

[13] Shyam Nandan Kumar, "Review on Network Security and Cryptography ." International Transaction of Electrical and Computer Engineers System, vol. 3, no. 1 (2015): 1-11. doi: 10.12691/iteces-3-1-1.

[14] C.Perkins and P.Bhagwat,"Highly Dynamic Destination-Sequenced Distance-Vector Routing (DSDV) for Mobile Computers," in Proc. of Sigcomm conference on Communications architectures, protocols and applications, London, England, UK, 1994, pp. 234-244

[15] D. B. Johnson and D. A.Maltz,"Dynamic Source Routing in Ad Hoc Wireless Networks," Mobile Computing, T. Imielinski and H. Korth, Ed. Kluwer Academic Publishers, 1996, vol. 5, pp. 153-181.

[16] C. E. Perkins and E. M.Royer,"Ad-hoc On-Demand Distance Vector Routing," in Proc. of the 2nd IEEE workshop on mobile computing systems and applications, 1997, pp. 1-11.

[17] W. A. Mobaideen, H. M. Mimi, F. A. Masoud, and E. Qaddoura, "Performance evaluation of multicast ad hoc on-demand distance vector protocol," Computer Communications, vol. 30, no. 9, pp. 1931–1941, 200

[18] C. E. Perkins and E. M. Royer,"Multicast operation of the ad-hoc on-demand distance vector routing protocol," in Proc. of 5th annual ACM/IEEE international conference on Mobile computing and networking, Seattle, Washington, USA, August 15-20, pp. 207-218.

[19] M.G. Zapata, Secure Ad hoc On-Demand Distance Vector Routing, ACM SIGMOBILE Mobile Computing and Communications Review. Jun, (2002), vol, 6(3), pp.106-107.

[20] D. Cerri and A. Ghioni, "Securing AODV: the A-SAODV secure routing prototype," IEEE Communications Magazine,vol.46 no. 2, pp. 120-125, 2008.

[21] Di Pietro, R.; Guarino, S.; Verde, N.V.; Domingo-Ferrer, J. (2014). Security in wireless ad-hoc networks – A survey. Computer Communications, 51(), 1–20. doi:10.1016/j.comcom.2014.06.003

[22] Kaelbling, Leslie P.; Littman, Michael L.; Moore, Andrew W. (1996). "Reinforcement Learning: A Survey". Journal of Artificial Intelligence Research. 4: 237–285. arXiv:cs/9605103. doi:10.1613/jair.301. S2CID 1708582.

[23] A. Colorni, M. Dorigo et V. Maniezzo, Distributed Optimization by Ant Colonies, actes de la première conférence européenne sur la vie artificielle, Paris, France, Elsevier Publishing, 134-142, 1991.

[24] M. Dorigo, Optimization, Learning and Natural Algorithms, PhD thesis, Politecnico di Milano, Italy, 1992.

[25] Kwang Mong Sim, ; Weng Hong Sun, (2002). [IEEE Comput. Soc 2002 International Symposium Cyber Worlds: Theory and Practices. CW2002 - Tokyo, Japan (6-8 Nov. 2002)] First International Symposium on Cyber Worlds, 2002. Proceedings. - Multiple ant-colony optimization for network routing. , (), 277–281. doi:10.1109/cw.2002.1180890

[26] Zhao, Dongming; Luo, Liang; Zhang, Kai (2009). [IEEE 2009 Fourth International Conference on Bio-Inspired Computing (BIC-TA) - Beijing, China (2009.10.16-2009.10.19)] 2009 Fourth International on Conference on Bio-Inspired Computing - An improved ant colony optimization for communication network routing problem. , (), 1–4. doi:10.1109/BICTA.2009.5338074

[27] B. Chandra Mohan; R. Baskaran (2012). A survey: Ant Colony Optimization based recent research and implementation on several engineering domain. , 39(4), 4618–4627. doi:10.1016/j.eswa.2011.09.076

[28] ScottPlot (2022, October 19) https://scottplot.net/

[29] NetworkX (2022, October 19) https://networkx.org/

[30] R. W. Hamming, "Error detecting and error correcting codes," in The Bell System Technical Journal, vol. 29, no. 2, pp. 147-160, April 1950, doi: 10.1002/j.1538-7305.1950.tb00463.x.

[31] Max Roser, Hannah Ritchie and Esteban Ortiz-Ospina (2015) - "Internet". (2022, October 27) 'https://ourworldindata.org/internet'

[32] ICT Statistics (2022, October 27) https://www.itu.int/ITU-D/ict/statistics/ict/

[33] Number of smartphone subscriptions worldwide from 2016 to 2021, with forecasts from 2022 to 2027 (2022, October 27) https://www.statista.com/statistics/330695/number-of-smartphone-users-worldwide/