A Machine Learning based Approach for Protecting Wireless Networks Against DoS Attacks

Yeaseen Arafat, Kazi Samin Yeaser, Ashikur Rahman and Arnab Dasgupta
Department of CSE, Bangladesh University of Engineering and Technology (BUET), Dhaka-1205, Bangladesh
Email: yeaseen.arafat96@gmail.com, ksyp2008d@gmail.com, arnabadg1@gmail.com, and ashikur@cse.buet.ac.bd

***Abstract*—Two major security threats for wireless networks are physical jamming and virtual jamming. The inherent openness of the wireless channels exposes the network to the *physical* jamming problem. On the other hand, the virtual carrier-sensing mechanism of IEEE 802.11 based MAC protocols opens up even a less expensive *virtual* jamming problem. A malicious node can effectively launch Denial of Service (DoS) attacks through virtual jamming inhibiting access to a large portion of a wireless network at the minimum expense of power resulting in significant drop in aggregate throughput and traffic carrying capacity of the network. The existing solution(s) can recover to some extent but the attackers are developing new variants of attacks day-by-day. Therefore, novel and more robust mechanisms are needed to combat virtual jamming. In this paper, we propose a novel machine learning based solution that can effectively classify the malicious (i.e., jammer) and non-malicious nodes in order to intelligently ignore any channel allocation requests made by the jammers. Finally, by presenting rigorous simulation results we show the efficacy of the proposed solution and its superiority over other non machine learning based solutions.**

I. Introduction

The wireless channel is broadcast in nature making it highly vulnerable to denial of service (DoS) attacks. Malicious nodes can launch a DoS attack by performing either a *real* jamming or a *virtual* jamming. In real jamming, a malicious node intentionally jams critical packets at the right moment by producing interfering signals affecting signal-to-noise ratio at the receiver end so that the packet gets corrupted. On the other hand, in virtual jamming the virtual carrier-sensing mechanism of the IEEE 802.11 MAC layer is exploited. A malicious node (periodically) misleads other nodes forcing them to update their Network Allocation Vectors so that the channel is marked as busy for a long period of time. Both real and virtual jamming drastically reduce the throughput of the communicating nodes with a very little energy budget.

The main focus of this paper is virtual jamming and its remedy. The collision avoidance mechanism in carrier sense multiple access (CSMA) based MAC protocols leads to virtual jamming problem. In CSMA, the sender-receiver pair exchanges short Ready-To-Send (RTS) and Clear-To-Send (CTS) packet before the actual data packets in order to reserve the channel for the duration of the actual transmission. As the validity of the announcement for channel reservation is not checked, an adversary may easily exploit this mechanism by sending spurious RTS/CTS packet(s) falsely announcing possible reservation of the channel, but not sending any data packet at the end during the reserved period. In this way an adversary may devise sophisticated DoS attacks (possibly) causing partition in the network with a very low energy budget. By conserving energy the lifetime of the virtual jammer is also increased and it remains a threat for a longer period of time. It also degrades the aggregate throughput and the traffic carrying capacity of the network.

To prevent virtual jamming, validating RTS packets through physical carrier sensing has been proposed in the literature. According to this method, at the beginning or at a random slot of the announced data transmission period mentioned in the RTS packets, the neighboring nodes perform a physical carrier sensing of the channel to perceive whether the channel actually goes to busy state or not. This validation refrains attackers from generating false announcement using RTS/CTS packets. However, there are some limitations of this RTS validation method. First of all, although the declined throughput could be recovered to some extent using such RTS validations, there still remains a scope of further improvement and recover more throughput. Secondly, the RTS validation requires physical carrier sensing which consumes more energy.

In this paper, we propose an interesting blend of machine learning and wireless networking to combat virtual jamming and recover aggregate throughput to highest extent. The proposed mechanism applies classification techniques to identify malicious nodes launching virtual jamming and isolates those from well behaving nodes to prevent DoS attacks. Moreover, the proposed solution is *backward compatible* in a sense that any intelligent node equipped with the proposed mechanism might still carry on the communications with the nodes without having the mechanism incorporated within them.

II. Related Work

In this section, we concentrate on some endeavours that are closely related to our work. Acharya et. al. [2] propose a MAC protocol to enable parallel transmission in IEEE 802.11. Although by enabling parallel transmission, the DoS attack can be mitigated to some extent, the protocol requires a rigorous modification of IEEE 802.11. As our approach is backward compatible with the IEEE 802.11 protocol, we only focus on those works that are also backward compatible and require minimal changes in the basic IEEE 802.11 protocol.

There exists some static approaches to overcome DoS attack. The first one in the series requires the *RTS validation* which is suggested by Chen et. al. in [3]. As this approach can not recover the usage of channel bandwidth sufficiently at

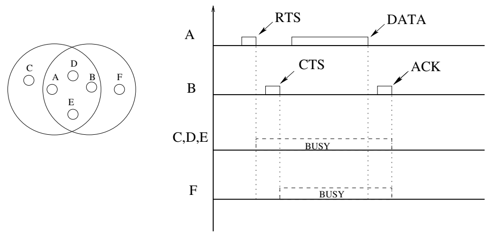


Fig. 1: Four Way Handshake

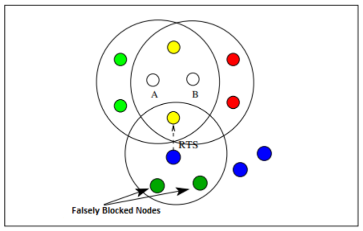


Fig. 2: False Blocking explained.

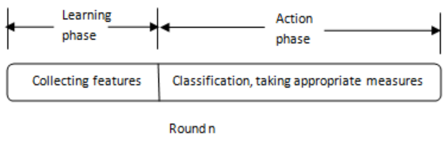


Fig. 3: Learning and action phase

the time of attack, another approach is suggested by Rahman and Gburzynski in [9] which is known as *Random RTS validation*. Though the later one is also a static process, it is better than the first one approach in terms of network performance and has the ability to combat advanced variant of the attack. A fuzzy logic based learning algorithm for MANET was proposed by Devi et. al. [4] based on the results from [5]. A dynamic approach was proposed in [6], [8], [11] to enable and disable IEEE 802.11 DCF's RTS/CTS handshake. In [8], a *regulator* function has been introduced to capture the empirical characteristics of RTS/CTS performance. The function depends on some tunable parameters namely packet size, transmission rate for both data and signaling traffic, as well as network contention. Again, a good static way to detect vulnerable RTS packet was proposed in [7]. But that process can not identify the invalid RTS packet until it receives the DATA packet.

Apart from the above mentioned works, to the best of our knowledge, no other prior work explores the technique to prevent DoS attacks using *machine learning*. Unlike other works, the proposed work extracts some features from the network traces so that a learner can be built to differentiate malicious nodes from non-malicious ones.

III. Virtual Jamming Problem and its remedy

As a background, we review virtual carrier sensing of 802.11 and show how it leads to false blocking and virtual jamming.

A. IEEE 802.11 Protocol: virtual carrier sensing

The virtual carrier sensing mechanism of IEEE 802.11 solves the well-known "hidden node problem" with the help of a four-way handshake of RTS/CTS/DATA/ACK packets. The mechanism is illustrated in Figure 1. At first the sender initiates the handshake by sending a Request-To-Send (RTS) broadcast packet to its neighborhood. In response, the indented recipient replies with a Clear-To-Send (CTS) broadcast packet in the recipient's neighborhood this time. As both RTS and CTS contain the duration to transmit the actual data packet, nodes receiving those packets easily become aware of the transmission and do not interfere with it. Finally the sender transmits the actual data packet, and the receiver responds back with an ACK packet if data is received correctly.

B. Problems with the virtual carrier sensing

The virtual carrier sensing mentioned above leads to *false blocking*, and *virtual jamming*, both of which we describe next.

1) False Blocking Problem: The False Blocking occurs when some nodes become blocked for a non-existent transmission. To understand the details consider the scenario shown in Fig. 2. During the data exchange between *A* and *B*, all red, yellow, and green nodes are temporarily blocked by the RTS-CTS packets exchanged between the communicating pair A and B. However, blue nodes, being outside the range of both *A* and *B*, are allowed to transmit and receive. Now if a blue node *D* wants to transmit to a yellow node *C*, *D* will initiate with a RTS packet addressed to *C*, but *C* will fail to respond with a CTS as it is blocked due to the transmission between *A* and *B*. *D* will try later assuming *C* is busy. But the ineffective RTS packet will make all nodes within *D*'s transmission range blocked for the entire duration of the non-existent transmission, as announced by *D*. This false blocking [10] may further propagate if some other blue node tries to send RTS packets to any of the newly-painted green nodes, thus hindering possible transmission.

2) Virtual Jamming Problem: False blocking opens up an opportunity for malicious nodes to launch a DoS attack. During this attack, the malicious node can deliberately send short RTS packets periodically announcing long transmissions never to occur. Hearing the RTS, some part of the network will remain blocked and the blocking can be propagated like false blocking. Thus a large segment of the network can be effectively jammed using a trivially small amount of power as RTS is a small packet. Virtual jamming is shown in Fig. 4 where *A* sends false RTS packets to *B* inserting a large but legitimate value in the duration field. All nodes within A and B's transmission range (i.e., *C,D,E* and *F*) will become blocked after receiving such a packet for the duration requested by *A* and *B*. Once the waiting time of *C,D,E* and *F* will be over, *A* will send false RTS packet again to continue the attack.

C. Solutions

As a solution of both virtual jamming and false blocking problem RTS validation has been proposed in [10] [3].

1) RTS Validation: To prevent virtual jamming, validating RTS packets through physical carrier sensing has been proposed [3]. According to this method, at the beginning of the announced data transmission period in the RTS packets, the neighboring nodes perform a physical carrier sensing of the channel to realize whether the channel actually goes to busy state or not. If the channel is found idle other sources who are waiting to reserve the channel for possible data transmission are allowed to access it. On the contrary, if the channel is really busy then it is a valid announcement and the neighboring nodes

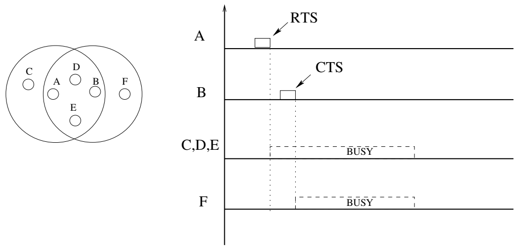


Fig. 4: Virtual Jamming

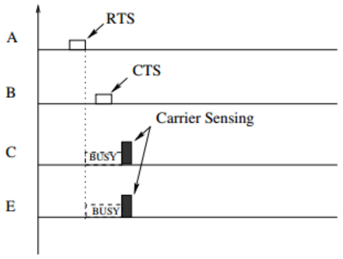


Fig. 5: RTS Validation

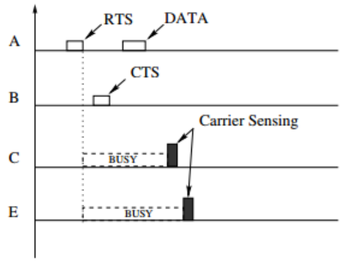


Fig. 6: Random RTS Validation

defer until the ongoing transmission is over.

2) Random RTS Validation: RTS validation does not work for **advanced** variant of DoS attacks. If an attacker already knows that all its neighbors will validate its transmission request after waiting for a CTS packet transmission time, then it can intelligently send a very small packet at the beginning of the data transmission time to create an impression that the channel is busy. Transmission of this small packet will force other nodes to (falsely) assume that the transmission has taken place and force them to remain blocked for the entire time. To prevent the advanced attacker a process called Random RTS Validation is proposed in [9]. In Random RTS Validation the neighboring nodes check for transmission randomly at any part of the data transmission period. As the validation is made at random, the advanced attacker will be detected in most of the cases and almost 50% of the data transmission period can be recovered on the average. The process is illustrated in Fig. 6.

IV. Proposed Machine Learning based approach

In this section we propose a novel machine learning based approach to combat the "advanced" variant of virtual jamming.

A. Problem with RTS validation

In RTS validation, every RTS packet coming from a node is validated by all recipients although a malicious node is likely to repeatedly send RTCS packets with the intentions to jam the network. Thus, an optimal approach is to classify a node's surrounding neighbors into *malicious*, and *non-malicious* categories and (safely) ignore all subsequent channel reservation requests coming from already identified malicious node(s) without performing any RTS validations. As the RTS validation is not performed, a significant amount of processing time and energy can be saved. This is the basis of the proposed machine learning based approach. However, there are many challenges. First of all, a node needs to clearly distinguish false blocking and virtual jamming because false blocking is a legitimate naturally evolved phenomenon whereas virtual jamming is a deliberately introduced phenomenon. The classification task becomes more challenging in congested environments because there could be many false blocking mimicking similar traffic patterns of virtual jamming which might lead to frequent mis-classifications and false positives. If a well-behaving node is ever mis-classified as a malicious node, all its future transmission requests will be ignored by all its neighbors. To tackle this problem, the behaviour of all neighboring nodes needs to be reevaluated periodically.

B. Machine learning based approach

The operation of machine learning based remedy of DoS attack is broken into rounds. Each round starts with a *learning* phase during which data values reflecting the behaviour pattern of each node's neighbors are collected. The learning phase is followed by an *action* phase during which neighbors of each node are classified into malicious, and non-malicious categories. In action phase appropriate measures are taken against the malicious nodes. The details are given below:

1) Learning phase: During this phase a node observes the behaviour pattern of all neighbours and collects data of some predetermined features about them. Despite being in learning phase, the node still continues to run *Random RTS Validation* to nullify the detrimental effect of false RTS packets. On the average 50% throughput loss can be recovered in this phase.

2) Action phase: At the beginning of this phase, each node classifies its surrounding neighbours as malicious or non-malicious based on the selected features gathered during the last *Learning Phase*, last *Action Phase* and latest measured values of those selected features. Once classified, each node ignores all subsequent RTS packets coming from the malicious nodes for the remaining period of the phase. For non-malicious nodes it validates their RTS packets by using *Random RTS Validation*. It also observes their behaviour and collects data values for determining their behavioural patterns. Thus, in the action phase nearly 100% throughput can be recovered. Usually the action phase is much longer than the learning phase as shown in Fig. 3. After the action phase the next round starts. The ratio of the duration of learning phase, T_L and duration of action phase, T_A in a round is called *Learning to Action Ratio* or **LAR** and can be tuned for optimal performance. Mathematically, $LAR = T_L/T_A$.

C. Selected Features

Feature Selection is one of the core concepts in machine learning which greatly impacts the performance of the derived models. Different random scenarios were studied first to select suitable features for accurate classification, Also some of the scenarios were generated without presence of any attacker to better understand the usual traffic pattern of the networks. After careful study the following three features were selected.

- Moving Average of Invalid RTS packet Ratio (IRR).
- Deviation of Moving Average of IRR.
- Moving Average of Inter Arrival Time of RTS packets

To understand the effectiveness of the selected features we generated random scenario having 25 nodes and 6 packet

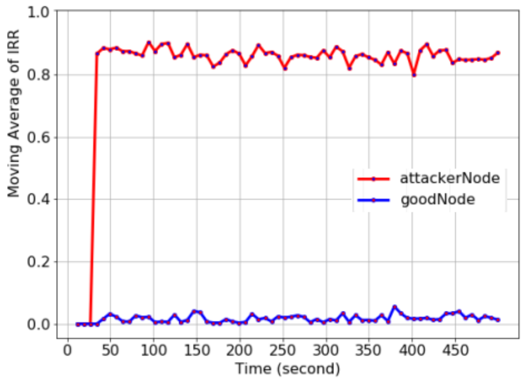


Fig. 7: Effect in Moving average of IRR over Time

sources. Out of these six sources, two sources were attackers and remaining four sources are non-malicious (i.e., regular sources) taken at random.

1) Moving Average of IRR: An attacker node is likely to produce a large number of false RTS packets without any data packets to follow. On the other hand non-malicious nodes RTS packets will be followed by data packets as those are legitimate. Thus first feature to distinguish a non-malicious node from attackers is to look into the proportion of invalid RTS packets generated by a node. *IRR* stands for *Invalid RTS ratio* which captures this feature. We call a RTS packet is invalid if no data packet follows it. So the IRR refers to the ratio of Invalid RTS count to the Total RTS count.

$$IRR = \frac{\text{Invalid RTS count}}{\text{Total RTS count}}$$

In every round, during Learning phase and Action phase whenever a RTS packet is received Total RTS count is incremented and if no data packet is sensed during *Random RTS Validation* only then the Invalid RTS count is incremented. It is obvious that for an attacker the IRR will be very high. However, the actual value of the parameter heavily depends on the sender's data transmission rate and its start time. In order to nullify these effects we calculate the moving average of IRR instead of total counts. The moving average at a particular round is calculated from IRR_L of the Learning phase and IRR_A of the Action phase in that round. Then the new value IRR_{new} (of the current round) is calculated by using IRR_{old} of the previous round using the following equation:

$$IRR_{new} = (1 - cf_L - cf_A) * IRR_{old} + cf_L * IRR_L + cf_A * IRR_A$$

where cf_L and cf_A are coefficient of Learning and Action phase used to prioritize them and act as hyperparameters. Their value ranges between 0.0 to 1.0.

Figure 7 shows the moving Average of IRR of two senders, one is the attacker (out of two) and the other is a non-malicious sender (out of four). Note that, the moving average of an attacker and a non-malicious node is clearly separable.

2) Deviation of Moving Average of IRR: When a receiver has a mix of attackers and non-attackers in its neighbourhood, the attacker's moving average of IRR must have much higher value compared to that of a non-attacker node. This is because an attacker is likely to produce more and more deliberate false RTS packets to launch virtual jamming attack. The non-attacker's false RTS packets are only due to the limitation of the protocol (i.e., an RTS does not get any CTS from a node if it is in the middle of an ongoing transmission). This observation leads to pick up a second feature about a neighboring node:– *the deviation of moving average of IRR*. The deviation is calculated for each neighbouring sender at each receivers. When a receiving node has a mix of attackers and non-attackers, it is obvious that for an attacking neighbor the deviation is always positive whereas for a non-malicious sending neighbor the deviation is always negative. The value could be either positive or negative only when a receiver has either all attacking nodes or all non-malicious nodes in its neighborhood. And if its value becomes positive for a non-attacker its value will be negligibly low. Figure 8 shows the deviation of moving Average of IRR of some attackers and non-attackers. Figure 8a shows the deviation of an attacker and a non-malicious sender of a receiver. Clearly the deviation for the attacker is positive and for the non-malicious node it is negative. Figure 8b shows the deviation of moving average of IRR of a non-malicious sender for a scenario containing no malicious nodes in the network. Although the deviation is positive on different occasions, its magnitude is negligible.

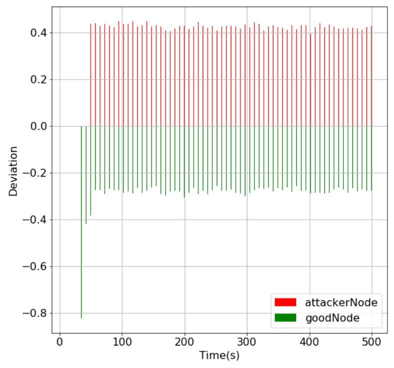
3) Moving Average of Inter Arrival Time of RTS packets: A non-malicious node sends data packets following exact rules of the IEEE 802.11 protocol. The protocol may incur large delays caused by other transmissions and re-transmissions of corrupted data packets (due to the collisions and noises of the shared channel). It also needs to wait sometime for receiver's ACK packets to come. According to the protocol it also back offs before every new transmission of the RTS packet. Therefore, the perceived average inter arrival time of RTS packets by a receiver sent from a non-malicious node will be very high. But an attacker is not enforced to follow the rules of the protocol. So it does not face these delays and consequently the average inter arrival time of RTS packets perceived by a receiver in an attacker's neighborhood will be very low. Therefore the last and final feature to look at for classification is the inter arrival time of RTS packets generated by any sender. For every RTS packet received by a node, the moving average of Inter Arrival Time of RTS packets is calculated using the following equation:

$$IAT_{new} = (1 - cf_{IAT}) \times IAT_{old} + cf_{IAT} \times (CT - LT)$$

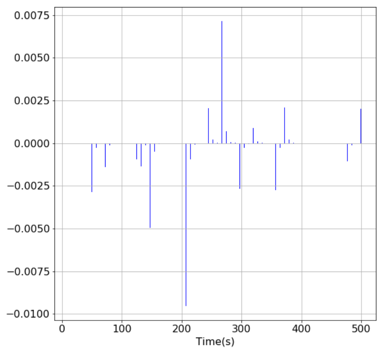
Here CT refers to the time when the new RTS packet is received, LT refers to the time of the last RTS packet received and cf_{IAT} is the coefficient of the RTS arrival interval acting as an hyperparameter. The value of cf_{IAT} ranges between 0.0 to 1.0. Figure 9 shows the effect in moving Average of Inter Arrival Time of RTS packets of one attacker node and one non-malicious. The Inter Arrival Time of RTS packets of a non-malicious node is much higher than that of an attacker.

V. Data set Preparation

We need to create a data set to feed in the machine learning algorithm so that it can learn. In this section, we cover the data



(a) In Scenario with attacker.



(b) In scenario with no attacker.

Fig. 8: Deviation of Moving average of IRR over Time

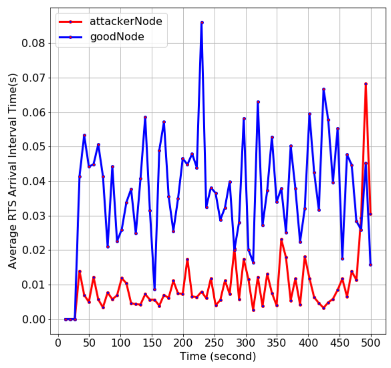


Fig. 9: Effect in Moving average of Inter arrival Time

TABLE I: Dataset

Moving average of IRR	Deviation of moving average of IRR	Average RTS Interval	Is it an attacker?
0.0411155	-0.0462200	0.0550714	0
0.3691936	0.2818581	0.0039915	1
0.0458663	-0.0414691	0.0892042	0
0.0250579	-0.0622775	0.0899507	0
0.3288425	0.2734656	0.0086242	1

TABLE II: SVM coefficients

Axis name	Coefficients
movingAverage	2.59804693
deviation	2.5296639
averageRTSInterval	-0.36001139
bias	-1.09308601

TABLE III: Conf. Matrix

Property	value
True Positive	85,083
False Positive	10,081
False Negative	17,436
True Negative	351,490

collection procedure, the machine learning model being used, the learner's properties and learner's result on test data set.

A. *Data set Properties*

At first we randomly generate some scenarios in NS-2.Packet sending interval is 0.08s. Each packet size is of 1024 bits. For each scenario, we run the simulation for 500 seconds. There were four Attributes in the dataset containing a total of $2,320,449$ rows. The total dataset size was $75.4MB$. Table I shows random five rows of the data set.

B. *Model selection: Support Vector Machine*

As we know the data set is linearly separable, we can feed the data set into a SVM machine for learning. Once trained, it returns a model with vector of coefficients and a bias. Inherently, those coefficients are the coefficients in a hyperplane as SVM separates the data set using a hyperplane. Next we perform a train-test splitting of the data set where SVM learns from training set and validates on the test set. We split the data set on a $80 : 20$ ratio of training set and test set.For learning and testing, we use scikit-learn API.

Table II shows the coefficients along with the feature axes of the data set.

C. *Learner's effectiveness on Test Data*

We activate the learner model on the test data set and the results are quite promising. Below are some results.

1) Confusion Matrix: Table III shows the confusion matrix on test data set. True positive and true negative values are quite high compared to false positive and false negative values.

2) False Positive Ratio and Accuracy: The accuracy on test data is 94%. Also the false positive rate is only 2.78%.

After cross validation, we have incorporated the trained SVM model in NS-2 simulator [1] to capture the effectiveness of the model. We describe the performance of the model next.

VI. Experimental Results

We run simulation experiments using NS-2 simulator [1] to verify the effectiveness of the built model and compare its performance with other non-machine learning based approaches.

A. *Performance Metrics*

1) Throughput: Throughput is the rate of successful message delivery over a communication channel. The goal of every communication channel is to maximize the throughput. We measure both *instantaneous throughput* which is measured over a short time period and the *average throughput* which is measured over a long time period.

2) Delay: The delay of a network is the time taken by a bit of data to travel across the network from one node to another. Both *instantaneous delay* and *average delay* were calculated to evaluate built model's performance. Instantaneous delay is the delay of measured over a short period of time. On the other hand *average delay* is measured over full simulation time.

B. *Simulation Environment*

We run the simulation on the topology shown in Figure 10. Exactly same scenario was used in to generate the results presented in [2] [9] The deployment area is 625×625 sq unit where eight nodes are carefully deployed. Green nodes are non-malicious sender and only attacker is the red node. Inner four nodes 1, 2, 3 and 4 form a clique. The outer ring contains four other nodes 5,6,7, and 8. The nodes in outer

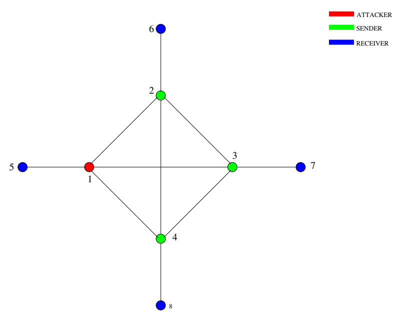


Fig. 10: Static Scenario

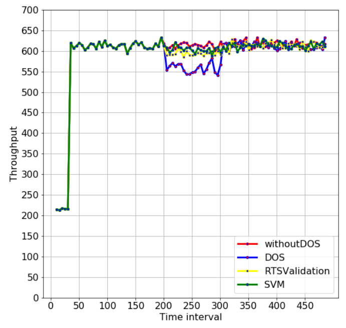


Fig. 11: Throughput: Different approaches

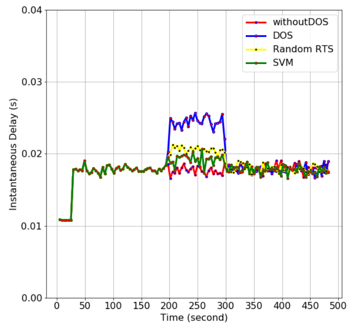


Fig. 12: Delay: Different approaches

ring are connected to exactly one node in the inner ring. The communication pairs are $2 \rightarrow 6$, $3 \rightarrow 7$, $4 \rightarrow 8$ and $1 \rightarrow 5$. The transmission range is 250 units. Packet size is 1024 bit. A sender is allowed to send a maximum of $10,000$ packets. The channel bandwidth is set to 1 Mbps. Interval time between consecutive packets is 0.04s that boils down to a sending rate of 25 packets per second. The attacker sends the false RTS packets at the same rate. We run the simulation for 500 seconds. Both machine learning based approach and the random RTS validation approach were run on this scenario.

C. *Performance comparison of different approaches*

1) Throughput comparison: At first we run the simulation without any virtual jamming by turning off the $1 \rightarrow 5$ communication. The other three communication pairs $2 \rightarrow 6$, $3 \rightarrow 7$, $4 \rightarrow 8$ were still actively sending packets with an interval of 0.04 seconds. The aggregate average throughput was found to be 593.643 kbps. Then we perform the advanced variant of virtual jamming attack by turning on the malicious node 1 initiating communication between $1 \rightarrow 5$ at 200th second of simulation and continue it for another 100 seconds. Consequently the RTS receiver node 5 becomes blocked during the attack. As the inner nodes $1, 2, 3$, and 4 forms a clique node 1's virtual jamming affects other three communications. Hence both instantaneous and aggregate throughput during the jamming interval decreased by a noticeable amount. The average aggregate throughput while jamming dropped to 580.998 kbps from 593.643 kbps.

Next, to overcome the virtual jamming at first we applied the existing Random RTS validation method and was able to bring back the average throughput up to 588.418 kbps.

Next, we applied the proposed machine learning based approach in place of Random RTS validation method; the average throughput became higher producing a throughput of 590.534 kbps. Clearly machine learning approach outperforms the Random RTS validation approach. Figure 11 compares instantaneous throughput of both approaches and shows how much improvement can be made on throughput.

2) Delay comparison: Finally we plot the instantaneous delay curves for normal scenario (without any attack), virtual jamming scenario (without any measures taken against the attack), with random RTS validation and with the proposed machine learning based approach. Both the random RTS validation approach and the machine learning based approach were able to significantly reduce the delay (roughly between 10 to 15%), the machine learning approach outperforming the random RTS validation.

VII. Conclusions and Future Works

In wireless networks, virtual jamming can be used for launching DoS attacks to significantly waste valuable channel bandwidth, decrease average throughput and increase delay. We propose a machine learning based approach that runs in rounds—each round consisting of a learning phase followed by an action phase. It collects data during learning phase and effectively use them during the action phase to classify neighbors into malicious and non-malicious category. Once successfully classified, the proposed method ignores any channel reservation requests coming from the malicious nodes. The experimental results shows that the proposed system is capable of nullifying the effect of DoS attack and substantially increase the average throughput and decrease the delay.

In future, the Learning to Action Ratio (LAR) and other hyper-parameters can be tuned for optimum performance.

References

[1] "The Network Simulator: NS-2: notes and documentation: http://www.isi.edu/nsnam/ns/."
[2] A. Acharya, A. Misra, and S. Bansal, "MACA-P: a MAC for concurrent transmissions in multi-hop wireless networks," *PERCOM*, 2003.
[3] D. Chen, J. Deng, and P. Varshney, "Protecting wireless networks against a denial of service attack based on virtual jamming," *MobiCom*, 2003.
[4] M. S. Devi and N. S. Gill, "Novel algorithm for enhancing manet protocol in smart environment," vol. 8, no. 10, 2019.
[5] M. Devi and N. Gill, "Performance evaluation of dynamic source routing protocol in smart environment," *IJATCSE*, vol. 8, pp. 333–338, 03 2019.
[6] Y. Edalat, J.-S. Ahn, and K. Obraczka, "Smart experts for network state estimation," *IEEE Trans. on Net. and Serv. Manag.*, vol. 13, 2016.
[7] T. Jamal, M. Alam, and M. M. Umair, "Detection and prevention against RTS attacks in wireless LANs," in *International Conference on Communication, Computing and Digital Systems (C-CODE)*, 2017.
[8] K. Obraczka, Y. Edalat, and B. Amiri, "A machine learning approach for dynamic control of RTS/CTS in wlans," *MobiQuitous*, July 2018.
[9] A. Rahman and P. Gburzynski, "Hidden problems with the hidden node problem." *23rd Biennial Symp. on Comm.*, pp. 270–273, 2006.
[10] S. Ray, J. B. Carruthers, and D. Starobinski, "RTS/CTS-induced congestion in ad hoc wireless LANs." in *WCNC 2003*, vol. 3, March 2003.
[11] T. Shigeyasu, M. Akimoto, and H. Matsuno, "Throughput improvement of IEEE802.11DCF with adaptive RTS/CTS control on the basis of existence of hidden terminals," June 2011, pp. 46–52.