# MAC Layer Misbehavior Detection Using Time Series Analysis

Maggie X. Cheng
New Jersey Institute of Technology
Newark, NJ 07102
maggie.cheng@njit.edu

Yi Ling
Missouri Univ. of Science & Technology
Rolla, MO 65401
ylz29@mst.edu

Wei Biao Wu
The University of Chicago
Chicago, IL 60637
wbwu@galton.uchicago.edu

***Abstract*—This paper presents a solution to the real-time detection of MAC layer misbehaviors in IEEE 802.11 networks. Among the wide range of misbehaviors, we focus on the sender side selfish behavior that creates a channel-capturing effect by using favorable parameters, and the receiver side selfish behavior that does not respond with CTS and ACK upon receiving RTS and data packets, which clears the channel for itself and causes its sender to waste resources. These misbehaviors are subtle to detect, and yet can undermine the performance of the well-behaved nodes significantly. This paper shows a powerful real-time detection method that can catch these misbehaviors as soon as they have started. The detection method requires collecting delay, throughput, and packet interval data to generate time series and applying a sequential change point detection algorithm on the data streams as soon as new data points come in. All attacks are simulated in ns-3 and the simulation results verified the effectiveness of the detection method.**

## I. Introduction

The IEEE 802.11 MAC protocol is the *de facto* protocol for wireless local area networks (WLANs). The multiple access control mechanism features a distributed coordination function (DCF), which offers random and distributed access to participating nodes [1]. One of the big success of IEEE 802.11 MAC protocol is it addresses the hidden terminal problem by using the Request to Send/ Clear to Send (RTS/CTS) message exchange before data packets are transmitted. Nodes in the vicinity of the sender and receiver will receive the RTS and CTS packets and therefore will refrain from channel access. Once a node has sensed the channel is busy, it enters binary exponential back-off to wait until the channel is clear. This is the key mechanism for collision avoidance.

In order for the protocol to be successfully executed, every node in the same WLAN is expected to conform to the protocol and follow the rules to determine the set of control parameters. These parameters include the carrier sense time Distributed Inter-Frame Space (DIFS) and Short Inter-Frame Space (SIFS), as well as the back-off value in the contention phase. If some nodes do not obey the protocol, it will create unfairness to other nodes, and even cause further damage to the network.

In this paper, we focus on the detection procedure and its application on three types of attacks: 1) manipulation on carrier sense time, 2) manipulation on back-off value, and 3) RTS dropping attack. We use a time-series analysis approach for the detection of MAC layer misbehaviors: we model the network performance measurements taken over time as time series and then apply a change point detection algorithm on time series to detect the abrupt changes caused by the misbehaviors. The objective is to detect misbehaviors as soon as they have started so that there is plenty of time to find the root cause and take effective counter-measures accordingly. In the context of network security, we take a two-step procedure to protect the network: detection and diagnosis. The detection procedure is active all the time. It is set out to hunt for anything that appears to be suspicious. Upon receiving alarms from the detection procedure, the diagnosis procedure will be activated to determine the root cause of the suspicious behavior.

The rest of the paper is organized as follows. In Section II, we survey the previous work in MAC layer attacks and detection methods. In Section III, we provide the background for misbehavior detection using time series analysis approach, and we present the core algorithmic component of the detection method. In Section IV, we show how the algorithm is used for the detection of each type of misbehavior. In Section V, we present the detection performance. Section VI concludes the paper with future research directions.

## II. Related Work

Attacks on the lower layer protocols of wireless networks have been extensively studied in recent years, for instance, attacks on the network layer to disrupt routing [2], attacks on the MAC layer to gain more channel access [3], and attacks on the physical layer to jam the channel [4]. Studies on MAC layer attacks as well as prevention and mitigation techniques constituted the main body of the literature. We further distinguish two types of attacks at the MAC layer: malicious attacks and selfish attacks.

In a malicious attack, an attacker intentionally attacks on a service node by keeping the channel busy near the service node [5], causes denial of service (DoS) to other nodes by injecting enormous amount of traffic into the network [6], or masquerades as a legitimate node by MAC address spoofing in order to disrupt network services [7]. In a selfish attack, a selfish node violates the rules of the protocol in order to gain more channel access to itself at the cost of performance degradation of others [8].

IEEE 802.11 MAC offers distributed and random access to nodes, but also leaves a lot of room for a selfish nodes

978-1-5386-3180-5/18/$31.00 ©2018 IEEE

to exploit. The selfish attacks are mainly targeted at the Distributed Coordinated Function (DCF) of IEEE 802.11 MAC [1], [9]. In the literature, the most addressed attack is cheating on the backoff rule [10]–[15]. The well-behaved nodes would follow the binary exponential back-off rule to decide its waiting time during the backoff stage, whereas a selfish node can just choose a small back-off value and increase its channel access time.

To detect the misbehavior of cheating on the back-off rule, many studies used the comparison of a selfish node and the well-behaved nodes on their performance measures. Without knowing whether there is a selfish node and who is the selfish node, the access point would periodically collect data from all nodes to get a center line (or average). A general principle of these detection methods is that if a node's data deviate from the center line too much, it is considered selfish. Performance measures used as detection statistics include throughput [8], [13], the mean time between received packets [13], RTS and data packets retransmission rates [16], and node's channel access time [17], etc. This approach involves determining the average from all nodes, and setting a detection threshold. The detection threshold is either a constant shift from the center line, or a constant fraction of the center line, where the constant shift or constant coefficient are manually set or calculated from historical data.

Another category of detection methods involve more sophisticated statistical methods, such as the sequential probability ration test (SPRT) method [3], or the statistical process control (SPC) method [11]. The SPRT method in [3] determines if a selfish node exists in a network by observing a sequence of $n$ data points representing packet inter-arrival time. The distribution of packet inter-arrival time is estimated, and then SPRT is performed on each new data point using the estimated probability density functions under both hypotheses. It stops when either of the stopping criteria is met, i.e., either accepting the null hypothesis that there is no selfish behavior, or accepting the alternative hypothesis that there is selfish behavior. Although it is a sequential method, it is not designed to detect the selfish behavior at the moment it starts; it is rather used to determine whether the $n$ data points are generated by a selfish node or a well-behaved node. It is assumed that the node behavior has been consistent in the sequence of $n$ data points, therefore it is not a change point detection method. Section III provides more detailed description of the method.

## III. Time Series Analysis Methods

### A. Network Measurements

The detection procedure requires that we sample round-trip time from successfully received packets starting from sending RTS until receiving ACK, and also monitor the data packets received over time. Attack detection is running on each active node. A sender node should sample the round-trip time over time, and a receiver node should record the received data packets from a particular sender over time. The number of packets received over time is bursty even in the non-attack scenario due to the random nature of IEEE 802.11 MAC. If we use this time series for attack detection, it will generate too many false alarms. Therefore, instead of using the instantaneous throughput, we use the cumulative throughput over time as the time series to perform attack detection. Detection algorithm is performed at each node independently. Each node run a sequential change point detection algorithm on the time series it generates, some may requires using derived data instead of raw data to perform. If a node has detected a change point, alarms are sent to the designated node in the network. If it is a sensor network, such designated node could be the base station; if it is a wireless ad hoc network with no base station, a clustered structure is sufficient. Then all alarms will be sent to the cluster heads, and cluster heads can make decision whether a true attack has happened depends on the alarms it has received.

### B. Problem Statement

In mathematical abstraction, detecting abrupt changes caused by the misbehaving nodes is casted as a change point detection problem in time series. A time series is a sequence of data points $\{x_1, x_2, \ldots, x_t, \ldots, \}$ representing some measurements/observations taken from a stochastic process over a continuous time interval, and a change point in the time series is a time instance starting from which the probability distribution of the stochastic process has changed. It is assumed that before the change point, the time series follows one distribution, and after the change point, and the time series starts to follow another distribution. Suppose the pre-change and post-change density functions are $f(\cdot)$ and $g(\cdot)$, respectively, and the change point is $\mu$. The alternative hypothesis is then formulated as:

$$\begin{aligned} \mathcal{H}_\mu: \quad & \{x_1, x_2, \ldots, x_\mu\} \sim f \\ & \{x_{\mu+1}, x_{\mu+2}, \ldots, x_n\} \sim g \end{aligned}$$

while the null hypothesis is formulated as:

$$\mathcal{H}_0: \quad \{x_1, x_2, \ldots, x_n\} \sim f$$

If no change has occurred, then $\mathcal{H}_0$ is true; if a change has occurred, then $\mathcal{H}_\mu$ is true. The detection algorithm is tasked with deciding which hypothesis is true, and if the alternative hypothesis is true, the algorithm reports the change point. It is desired that the algorithm have high detection rate and low false alarm rate, and a low detection latency, which is the time interval from the true value of $\mu$ to the time a change is detected.

### C. Detection Algorithm

In this paper, we use a new sequential change point detection algorithm that requires no knowledge about the data characteristics. The algorithm compares its detection statistics with a detection threshold that is decided based on the current data, not the preset values.

The algorithm works as follows.

Let $m$ be the window size. Let $t$ be an integer with $0 \leq t \leq n - 2m$. As the algorithm moves from $t = 0$ to $t = n - 2m$,

it takes $m$ consecutive data points from the time series—call it window 1, and the next $m$ data points— call it window 2. Then we compute the sum of the $x$'s in the two windows.

$$Y_1(t) = \sum_{i=t+1}^{t+m} x_i \quad \text{and} \quad Y_2(t) = \sum_{i=t+m+1}^{t+2m} x_i$$

$$D(t) = \mid Y_1(t) - Y_2(t) \mid$$

We compare the difference between the sums of the two windows with a threshold $D_{Th}$, which is determined by the characteristics of the data and the desired upper bound for false alarm rate. The detection threshold $D_{Th}$ is computed by using the following procedure:

Let $\Phi(\cdot)$ be the cumulative distribution function (CDF) for standard normal distribution, defined as:

$$\Phi(z) = \mathbb{P}(a \leq z)$$

For a given false alarm rate $\epsilon$, we can solve the cutoff value $z$ from the following equation:

$$1 - \Phi(z) = \epsilon$$

Then the solution $z$ for the above equation is scaled to be used as the detection threshold:

$$D_{Th} = z\sqrt{2m}\sigma$$

where $\sigma$ is the square root of the sample variance.

As the two windows move from the low end to the high end of the time series, the following condition is checked:

$$D(t) \geq z\sqrt{2m}\sigma$$

If the condition is satisfied, the alternative hypothesis is true. The algorithm then reports that a change point has occurred at $\tilde{\mu} = t + m$.

## IV. Misbehavior Detection in IEEE 802.11 MAC

In IEEE 802.11 MAC, ordinary asynchronous traffic all uses the distributed coordination function for medium access. Before the sender transmits a frame, it performs carrier sense. The carrier sense duration is dependent upon the Inter-Frame Space (IFS) value at the present time. If the medium is idle for IFS amount of time, it transmits a frame. Each atomic unit for frame transmission is a four-frame sequence: RTS-CTS-DATA-ACK. If the medium is busy or becomes busy before the IFS amount of time counting down to zero, it waits until the current transmission ends and waits for IFS amount of time again before going to contention. During the contention period, each node performs binary exponential back-off to select a waiting time. The node that selects a smaller back-off time among the contenders will be the winner and will transmit a frame. For asynchronous traffic, there are two different IFS values: Distributed coordination function IFS (DIFS) and Short IFS (SIFS). DIFS is used before sending RTS, while SIFS is used in the later part of the sequence before CTS, DATA, and ACK.

In this paper, we address three types of attacks targeted at IEEE 802.11 MAC protocol.

### A. Type I Attack: Shorter DIFS

Based on the carrier sense protocol, a node refrain from transmission if the medium is busy. If a node uses a shorter DIFS value than its peers, it gains a lot of advantage. It is essentially a priority to use the transmission medium. If two nodes both have a frame to transmit and they start to perform carrier sense at the same time, the node with shorter DIFS will start to transmit ahead of the other one. Carrier sense prevents the other node from interrupting the ongoing transmission so the other node will have to wait until the medium is cleared. If a node always uses shorter DIFS values, it will have a larger share of the channel resource than other nodes, so the throughput and delay performance will be improved, meanwhile the throughput and delay performance of other flows will be degraded.

To detect the selfish behavior, we cannot rely on the selfish node to perform attack detection on its own data and report itself. We can only detect the misbehavior from the performance degradation of other flows. The delay time series in an attack scenario won't be a constant shift from its normal state. As delay increases, the jitter or variance of delay also increases. If we put the delay data in a time series, it is expected that the distribution of the cumulative means of delay and variance of delay have a change point at the moment of attack. Throughput time series will show similar increase in both cumulative means and variance. Applying the change point detection algorithm on any one of the time series will be able to detect the attack.

### B. Type II Attack: Shorter DIFS and Smaller Back-off Value

In the carrier sense period, if a sender sensed that the medium is busy, it will refrain from channel access. After the medium becomes free for DIFS or SIFS amount of time, if all senders start to transmit a frame right away, there is potential collision among the senders in the back-off stage. Therefore it is necessary to make each sender wait for a random number of slots before transmitting a frame. This period is called contention. The random number is chosen between zero and a power of 2 (not inclusive) based on the binary exponential back-off algorithm. In the non-attack case, every node chooses its random number independently, the one that chooses the smallest number will start to transmit first and deter others from transmission. In the attack case, the attacker either chooses a random number from a smaller window or uses a fixed back-off value, $\gamma$, instead of following the exponential back-off algorithm during the contention period. The attacker will have higher probability to gain channel access than the rest of nodes.

It is observed that when the attacker uses a smaller $\gamma$, it gains more channel access than using a larger $\gamma$. Whether the combined use of shorter DIFS and smaller back-off value further create more detrimental effect on other flows, this is interesting to find out. Intuitively, a more aggressive attacker will cause more damage to the network. However the data from simulation show that this is not the case.

### C. Type III Attack: RTS Dropping

In the RTS dropping attack, the receiver may not response with a CTS upon receiving a RTS. It selectively drops RTS packets, which causes the transmitter to retransmit several times, wasting time and channel resources. If the sender does not receive a CTS in the specified time, it will time out and go into the back-off state. The attacker thus effectively deters the sender from channel access. In some case, the sender may perceive the absence of CTS as a result of congestion and tell the upper layer to change route in order to avoid the congested area. The ultimate benefit for the attacker is less requests from neighbors to access the medium. The attacker thus *clears* the channel for itself so that it can have more channel access time.

Absence of CTS is often regarded as a result of collision, or poor wireless channel condition. It may not be immediately taken as a sign of attack. The distinction between congestion and selfish misbehavior is possible when more flows are observed. In the congestion case, all flows show performance degradation along with high retransmission rates of RTS packets, whereas in the attack case, only the flow that has the attacker as a relay node or receiver will suffer, and the attacker as a sender actually has improved throughput due to increased channel access time.

## V. Simulation

To simulate the misbehaviors, we hacked into IEEE 802.11 WiFi MAC in ns-3 to create three attacker classes different from well-behaved nodes. The well-behaved nodes will follow the IEEE 802.11 MAC protocol and use default parameters, while the attacker classes can deviate from the rules of the protocol and use different parameters.

We use ns-3 to generate network traffic, and then retrieve performance measures such as delay, throughput and inter-packet interval from the trace file. To collect delay data, the sender needs to time-stamp the packet at the MAC layer when RTS is sent. receiver can recollect end-to-end delay, throughput, and inter-packet interval from the received packets.

Fig. 1 shows the network used for simulation. Nodes are deployed in a $150m \times 150m$ terrain area. Every sender is one hop away from its receiver. Node 2 is the selfish node in all cases.

### A. Case 1: Shorter DIFS Attack

*1) Simulation Setup:* The IEEE 802.11 family of standards describe the DCF protocol, which controls access to the physical medium. A station must sense the status of the wireless medium before transmitting. If it finds that the medium is continuously idle for DIFS amount of time, it is then permitted to transmit a frame. If the channel is found busy during the DIFS interval, the station should defer its transmission. The DIFS plays an important role in contention-based medium access process. If the attacker can modify the value of DIFS and get a smaller one, it can get more chance to access the channel and let other legitimate nodes suffer various delay. In IEEE 802.11b, the default value of DIFS is SIFS + 2 $\times$ slot-time, with SIFS=$10\mu s$ and a slot-time=$20\mu s$. In this simulation, the selfish node uses DIFS=SIFS.

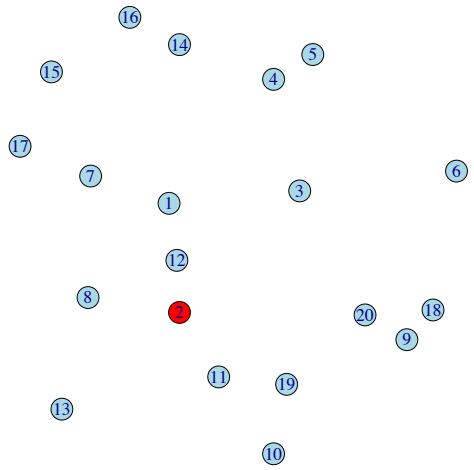



Fig. 1: The 802.11 network under selfish attack. Node 2 (red) is the attacker in all cases.

There are five flows from the well-behaved nodes: $19 \rightarrow 1$, $14 \rightarrow 1$, $12 \rightarrow 1$, $10 \rightarrow 1$, $6 \rightarrow 1$.

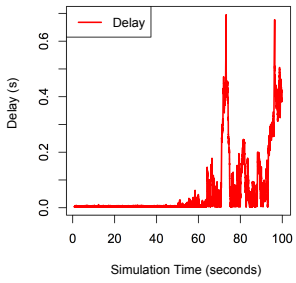
The attacker starts to behave selfishly at simulation time 50 second by switching to a shorter DIFS. The attacker is near the sink node (node 1 in purple color), and for some flows it is also near the sender. Due to the shorter DIFS, the attacker has better chance to send out RTS and reserve the channel, thus deter the other flows from transmission. We expect that all other flows will experience a performance degradation.

*2) Results:* With the attacker using a shorter DIFS value, other flows are given less channel access time. Fig. 2 shows the delay and throughput as well as the mean and variance of them for the flow from node 14 to node 1. Although the throughput shows oscillation, the cumulative mean of throughput is decreasing therefore the throughput is still decreased over time. Other flows all show similar performance degradation. As we can see, the delay of a normal flow increases and the throughput drops. All victim flows show similar pattern.

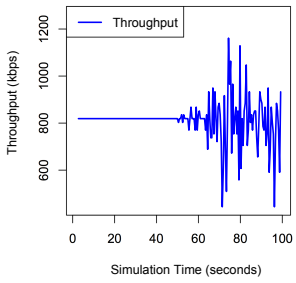
### B. Case 2: Shorter DIFS and Smaller Back-off Value Attack

In 802.11 MAC, the contention window size is measured in slot-time, with each slot-time=$20\mu s$. A well-behaved node follows the binary exponential back-off rule to select a random number $\gamma$ between [0, CW-1], where CW is the contention window size. The minimum contention window size is 32 and the maximum contention window size is 1024. A node would double its contention window size each time it enters the contention state without a successful transmission until it reaches the maximum value. Upon successful transmission, the contention window size is reset to the minimum value 32.

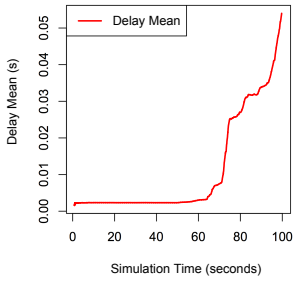
The selfish node can cheat on the back-off rule by using a smaller back-off value. In this simulation, we choose a fixed $\gamma = 2$. In the contention period, when the channel is sensed



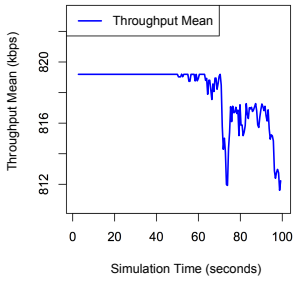

(a) Delay



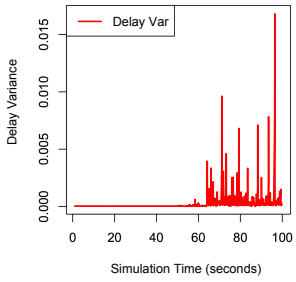

(b) Throughput



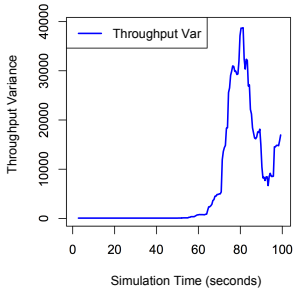

(c) Delay Mean




(d) Throughput Mean




(e) Delay Variance

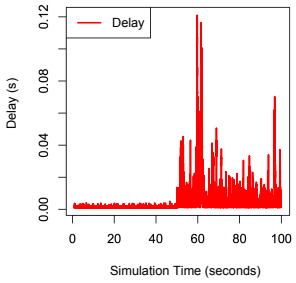



(f) Throughput Variance

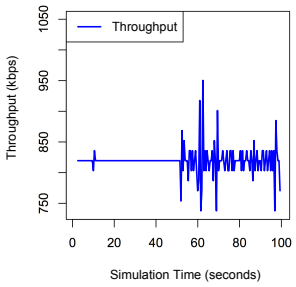
Fig. 2: The delay and throughput as well as their mean and variance for one flow (14-1).

idle for DIFS amount of time, the back-off counter starts to count down. When the counter reaches zero, a node transmits a RTS frame. Since the selfish node uses DIFS=SIFS and a fixed back-off value $\gamma = 2$, whereas the normal nodes have DIFS=SIFS+2 $\times$ slot-time, it is equivalent to say that the selfish node waits no time after a normal DIFS amount of time to transmit a new RTS packet, while the normal nodes have to wait an additional random number of time-slot.

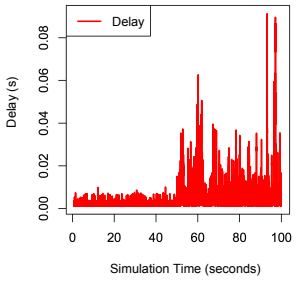
The network setup is the same as in case 1. The results show that all five flows experience significant throughput loss, companioned by increased delay. Fig. 3 shows the delay and throughput of four flows. The fifth flow from node 6 to node 1 is shown with more detail in Fig. 4. Applying the change point detection algorithm on the cumulative means time series can easily detect the performance change caused by the selfish node at near 50 seconds.



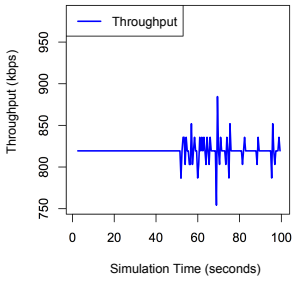

(a) Delay of flow 19-1



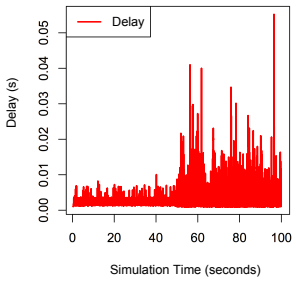

(b) Throughput of flow 19-1



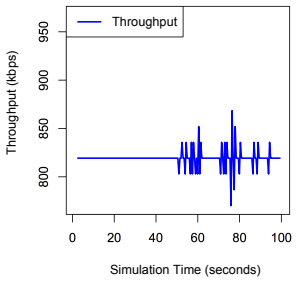

(c) Delay of flow 14-1



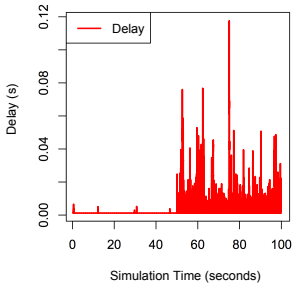

(d) Throughput of flow 14-1



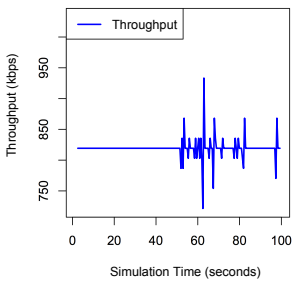

(e) Delay of flow 12-1




(f) Throughput of flow 12-1



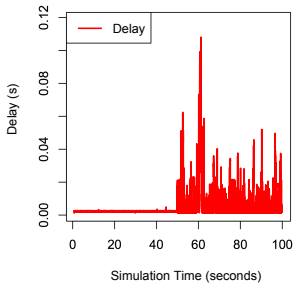

(g) Delay of flow 10-1



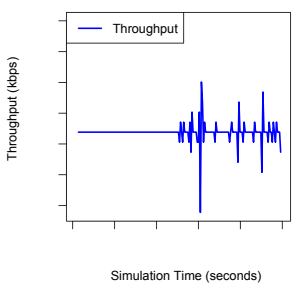

(h) Throughput of flow 10-1

Fig. 3: The delay and throughput of four flows in the combined shorter DIFS and smaller back-off value attack.

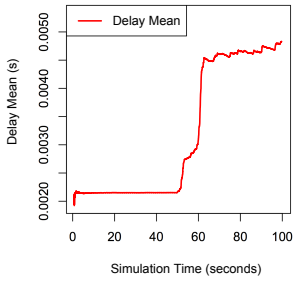
A counter-intuitive observation is that the performance degradation of normal flows are not as severe as in the first case where the attacker only uses shorter DIFS but still follows



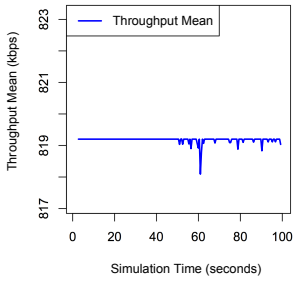

(a) Delay of flow 6-1




(b) Throughput of flow 6-1




(c) Mean Delay of flow 6-1




(d) Mean Throughput of flow 6-1

Fig. 4: The delay and throughput of flow 6-1 in the combined shorter DIFS and smaller back-off value attack. (c) and (d) show the cumulative means of delay and throughput, respectively.

the same rule for contention. Table I shows the cumulative means of delay and throughput of each victim flow. This is due to the fact that the selfish node no longer participates in contention when the well-behaved nodes start to contend, and therefore the number of nodes in the contention period is reduced. As a result, all other flows experience lighter traffic load in the contention period.

| | 19-1 | 14-1 | 12-1 | 10-1 | 6-1 |
|---|---|---|---|---|---|
| Case 1 Delay | 0.0283 | 0.0541 | 0.0064 | 0.0360 | 0.0521 |
| Case 2 Delay | 0.0042 | 0.0044 | 0.0035 | 0.0038 | 0.0048 |
| Case 1 Throughput | 818.95 | 812.22 | 819.12 | 817.44 | 816.26 |
| Case 2 Throughput | 818.95 | 819.20 | 819.20 | 819.20 | 819.03 |

TABLE I: Cumulative means of delay and throughput of all five flows in case 1 and case 2.

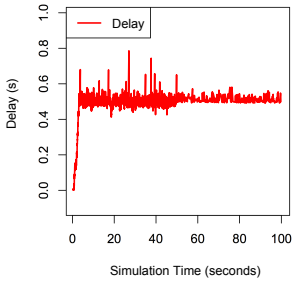
### C. Case 3: RTS-dropping Attack

*1) Simulation Setup:* In the RTS-dropping attack, the selfish receiver may not response with a CTS by selectively dropping a number of RTS packets, which will cause the transmitter node to retransmit serval times, deterring other nodes from transmission while the selfish node can ignore it and go ahead with its own transmission. Since the transmitter does not receive CTS in the specified time, it will time out on CTS and use binary exponential time to back off. And it definitely make the network latency higher.

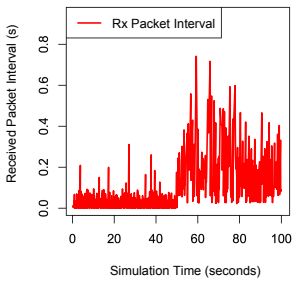
We use the same network as shown in Fig. 1 to simulate the attack. There are five flows in the network: $20 \rightarrow 2$, $11 \rightarrow 2$, $8 \rightarrow 2$, $7 \rightarrow 2$, $5 \rightarrow 2$.

Node 2 is the selfish node that selectively drops RTS. In the simulation, the ratio of CTS to RTS is 1/20, i.e., it drops the first 20 RTS packets and responds to the $21^{st}$ with CTS, and repeats.

*2) Results:* Fig. 5 shows the delay and throughput for one flow. Other flows show similar patterns. It is interesting to observe that the delay does not show significant increase. This is due to the fact that the RTS retransmission limit is set to 7. With a high drop rate of $20/21$, the RTS packets including the retransmitted RTS are mostly dropped, and only a small portion can get through. The ones that did get through did not experience much delay since the network traffic load is lighter. This is clearly observed on the delay chart where the received packets are sparser after 50 seconds. Note that the x-axis shows the receiving time of the packets. Fig. 5(b) shows that the inter-packet interval experiences significant increase after the attack starts.




(a) Delay




(b) Rx Inter-packet Interval

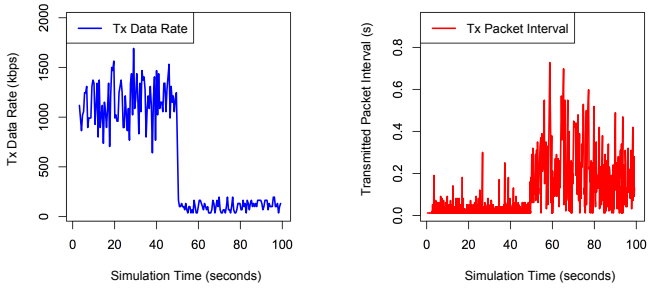
Fig. 5: The delay and inter-packet interval of a victim flow (20-2) under RTS dropping attack. Time series are taken from the receiver side.

In this case, instead of using the delay time series, the detection is performed on the inter-packet interval and throughput time series. Both data are available at the receiver, however, we cannot rely on the receiver to perform attack detection since the receiver is the attacker. The sender has to keep track of successfully transmitted data packets and record the packet interval based on the sending time and the total number of bytes transmitted. Fig. 6 show the data rate and the inter-packet interval for the successfully transmitted packets.

To directly apply the change point detection algorithm on the transmitter data rate and packet interval time series will generate many false alarms, however when we use the cumulative average of the data rate and packet interval, the time series are smoothed out, and the "change point" corresponding to the attack time is obvious, as shown in Fig. 7.

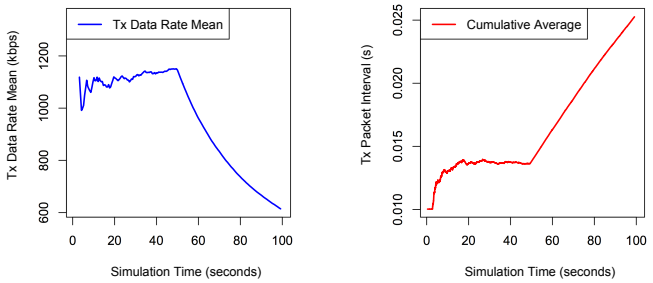
### D. Detection Results Summary

- Shorter DIFS attack: applying the new change point detection algorithm on delay, throughput, and cumulative




(a) Tx Data Rate (b) Tx Inter-packet Interval

Fig. 6: The transmitter data rate and inter-packet interval of a victim flow (20-2) under RTS dropping attack. Time series are taken from the sender side based on the sending time.




(a) Tx Data Rate (b) Tx Inter-packet Interval

Fig. 7: The cumulative means of the transmitter data rate and inter-packet interval of a victim flow (20-2) under RTS dropping attack. Time series are taken from the sender side based on the sending time.

means of throughput series can detect the attack at 50 seconds. Applying it on other time series will have larger detection latency. The cumulative means of throughput provides the overall best detection result— the smallest false alarm rate and the lowest detection latency.

- Shorter DIFS and smaller back-off value attack: applying the change point detection algorithm on the cumulative means of delay and throughput can detect the attack at 50 seconds, with delay series having higher false alarm rate.
- RTS Dropping attack: although the change point is visible on the transmitter data rate and transmitted packet interval time series, applying change point detection on these two time series generates high false alarms. Applying change point detection on the cumulative means of data rate and packet interval generates fewer false alarms, and the inter-packet interval series provides shorter detection latency.

## VI. Conclusion

This paper addresses the MAC layer misbehavior detection problem in wireless ad hoc networks. Three types of misbehaviors are studied, including sender side and receiver side misbehaviors. While all misbehaviors cause the disruption of innocent flows, it is observed that among the three types of misbehaviors, the shorter DIFS attack has the most detrimental effect on the wireless network.

We take a two-step procedure for the defense of wireless networks. First, a fast detection algorithm is used to hunt for any suspicious behavior in real time, and then detailed analytics is launched to localize the offense. In this paper, we have addressed the first step. Future work will address the identification and localization of the misbehaving node, and design effective counter measures to mitigate the damage caused by the misbehaving node.

## References

[1] G. Bianchi, "Performance analysis of the ieee 802.11 distributed coordination function," *IEEE Journal on selected areas in communications*, vol. 18, no. 3, pp. 535–547, 2000.

[2] M. X. Cheng, Y. Ling, and W. B. Wu, "In-band wormhole detection in wireless *ad hoc* networks using change point detection method," in *IEEE ICC 2016*, 2016, pp. 1–6.

[3] Y. Rong, S. K. Lee, and H.-A. Choi, "Detecting stations cheating on backoff rules in 802.11 networks using sequential analysis." in *INFOCOM*. Citeseer, 2006.

[4] N. Lyamin, A. Vinel, M. Jonsson, and J. Loo, "Real-time detection of denial-of-service attacks in ieee 802.11p vehicular networks," *IEEE Communications Letters*, vol. 18, no. 1, pp. 110–113, January 2014.

[5] V. Gupta, S. Krishnamurthy, and M. Faloutsos, "Denial of service attacks at the mac layer in wireless ad hoc networks," in *MILCOM 2002. Proceedings*, vol. 2. IEEE, 2002, pp. 1118–1123.

[6] Y. Zhou, D. Wu, and S. M. Nettles, "On mac-layer denial of service attacks in ieee 802.11 ad hoc networks: Analysis and counter measures," *Int. J. Wire. Mob. Comput.*, vol. 1, no. 3/4, pp. 268–275, 2006.

[7] Y. Sheng, K. Tan, G. Chen, D. Kotz, and A. Campbell, "Detecting 802.11 mac layer spoofing using received signal strength," in *INFOCOM 2008. The 27th Conference on Computer Communications. IEEE*, April 2008.

[8] V. R. Giri and N. Jaggi, "Mac layer misbehavior effectiveness and collective aggressive reaction approach," in *Sarnoff Symposium, 2010 IEEE*. IEEE, 2010, pp. 1–5.

[9] I. C. S. L. M. S. Committee *et al.*, "Wireless lan medium access control (mac) and physical layer (phy) specifications," 1997.

[10] T. Hayajneh, G. Almashaqbeh, and S. Ullah, "A green approach for selfish misbehavior detection in 802.11-based wireless networks," *Mobile Networks and Applications*, vol. 20, no. 5, pp. 623–635, Oct. 2015. [Online]. Available: http://dx.doi.org/10.1007/s11036-015-0605-4

[11] A. Aaroud, M.-A. El Houssaini, A. El Hore, and J. Ben-Othman, "Real-time detection of mac layer misbehavior in mobile ad hoc networks," *Applied Computing and Informatics*, 2015.

[12] M. Li, S. Salinas, P. Li, J. Sun, and X. Huang, "Mac-layer selfish misbehavior in ieee 802.11 ad hoc networks: Detection and defense," *IEEE Transactions on Mobile Computing*, vol. 14, no. 6, pp. 1203–1217, June 2015.

[13] M.-A. El Houssaini, A. Aaroud, A. El Hore, and J. Ben-Othman, "Analysis and simulation of mac layer misbehavior in mobile ad-hoc networks," in *Codes, Cryptography and Communication Systems (WCCCS), 2014 5th Workshop on*. IEEE, 2014, pp. 50–54.

[14] P. Kyasanur and N. H. Vaidya, "Selfish mac layer misbehavior in wireless networks," *IEEE transactions on mobile computing*, vol. 4, no. 5, pp. 502–516, 2005.

[15] A. A. Cardenas, S. Radosavac, and J. S. Baras, "Detection and prevention of mac layer misbehavior in ad hoc networks," in *Proceedings of the 2nd ACM workshop on Security of ad hoc and sensor networks*. ACM, 2004, pp. 17–22.

[16] O. N. Tiwary, "Detection of misbehaviour at mac layer in wireless networks," *proceedings of International Journal of Scientific and Engineering Research*, vol. 3, no. 5, 2012.

[17] A. Hamieh, J. Ben-Othman, A. Gueroui, and F. Naït-Abdesselam, "Detecting greedy behaviors by linear regression in wireless ad hoc networks," in *2009 IEEE International Conference on Communications*. IEEE, 2009, pp. 1–6.