EURASIP Journal on Wireless Communications and Networking
a SpringerOpen Journal

RESEARCH Open Access

Design and performance evaluation of a lightweight wireless early warning intrusion detection prototype

Alexandros G Fragkiadakis[1*], Elias Z Tragos[1], Theo Tryfonas[2] and Ioannis G Askoxylakis[1]

Abstract

The proliferation of wireless networks has been remarkable during the last decade. The license-free nature of the ISM band along with the rapid proliferation of the Wi-Fi-enabled devices, especially the smart phones, has substantially increased the demand for broadband wireless access. However, due to their open nature, wireless networks are susceptible to a number of attacks. In this work, we present anomaly-based intrusion detection algorithms for the detection of three types of attacks: (i) attacks performed on the same channel legitimate clients use for communication, (ii) attacks on neighbouring channels, and (iii) severe attacks that completely block network's operation. Our detection algorithms are based on the cumulative sum change-point technique and they execute on a real lightweight prototype based on a limited resource mini-ITX node. The performance evaluation shows that even with limited hardware resources, the prototype can detect attacks with high detection rates and a few false alarms.

Keywords: lightweight intrusion detection, jamming, signal-to-interference-plus-noise ratio, cumulative sum algorithms, performance evaluation, prototype

1 Introduction

Wireless networks' proliferation has been remarkable during the last decade as the license-free nature of the ISM band and the rapid proliferation of the Wi-Fi compatible devices, especially the smart phones, have offered ubiquitous broadband wireless internet access to millions of users worldwide. However, due to their open nature, wireless networks are susceptible to a number of attacks. Adversaries can exploit vulnerabilities in the medium access and physical layers and heavily disrupt the network operation (e.g., see [1-5]). The traditional methods of protecting the networks by using firewalls and encryption software are not sufficient, and for this reason, several intrusion detection algorithms have been proposed by the research community in order to address these issues.

In general, intrusion detection techniques fall into two main categories: misuse (or signature-based) detection and anomaly-based detection. The former is based on known signature attacks, it has low false alarm rates (FARs) but it lacks the ability to detect new types of attacks. The latter may have higher FARs but it has the potential ability to detect unknown types of attacks. In this article, we study the performance of anomaly-based intrusion detection.

In our previous studies [6,7], we investigated the performance of several algorithms for the detection of physical-layer jamming attacks. This type of attacks can be launched by adversaries through the generation of interference in neighbouring channels. We proposed intrusion detection algorithms that considered several metrics using two types of algorithms: simple threshold and cumulative sum (Cusum). The performance evaluation, in terms of the detection probability (DP), FAR, and the robustness to different detection thresholds, showed that Cusum Max-Min, a Cusum type of algorithm, has the best performance among all algorithms. The attack model we considered was based on a modified IEEE 802.11 node that violated several mechanisms (backoff, spectrum sensing, etc.), emitting energy on the

* Correspondence: alfrag@ics.forth.gr
[1]Institute of Computer Science of the Foundation for Research and Technology-Hellas (FORTH), P.O. Box 1385, GR 71110 Heraklion, Crete, Greece
Full list of author information is available at the end of the article

neighbouring channel legitimate nodes used for communication.

In this article, we extend our previous contribution in order to detect attackers (jammers) who follow different attack strategies. Such an attacker can for example emit energy on the same channel legitimates nodes use. For the detection of this type of attack, we consider a metric based on the ratio of the corrupted packets over the correctly decoded packets.

Furthermore, more powerful jammers based on software defined radio can completely block wireless network's operation. In this case, a metric based on the SINR or error-based metrics are not useful as no packets are transmitted at all. We detect this type of attack, called as blocking attack, using a metric based on the number of beacon packets transmitted by the access point (AP) in a pre-defined time window.

Based on these metrics we implemented anomaly-based intrusion detection algorithms running in a real limited resource prototype. This work presents in detail the functional blocks of the prototype and shows its operation in a real infrastructure-based IEEE 802.11 wireless network. We evaluate the performance of the algorithms in terms of the DP, the FAR, and their robustness to different detection thresholds. Our main contributions are listed below:

- we consider anomaly-based intrusion detection algorithms for the detection of different types of attacks,
- we develop a real lightweight prototype executing and evaluating the intrusion detection algorithms in realistic conditions,
- we show that even with limited hardware resources, the prototype gives high detection rates and low FARs,
- we introduce the term robustness to describe algorithms' performance stability under different detection threshold values.

The evaluation shows that all types of the attacks can be detected with a high DP and low FARs.

The remainder of this article is organised as follows. In Section 2, we describe the related work. In Section 3, we present the network layout for testing our prototype and the attack models used. The intrusion detection algorithms and their associated metrics are analysed in Section 4. The structure of the prototype and its functionalities are given in Section 5. In Section 6, we describe the evaluation methodology and then we present the performance results. Finally, conclusions appear in Section 7.

2 Related work

There are several significant contributions made by the research community in the area of the intrusion detection in communication networks. The work presented in [8] evaluates two types of algorithms for the detection of SYN attacks. The evaluation shows that the simple detection algorithm has satisfactory performance for the high intensity attacks but it deteriorates for the low intensity attacks. The Cusum algorithm, on the other hand, has robust performance for different types of attacks. This is consistent with the findings of this work; however, we perform measurements at the physical and medium access layers.

The authors of [9] describe and evaluate methods for anomaly detection and distributed intrusion detection in mobile adhoc networks, focusing on two routing protocols. They use a two-layer hierarchical system, where anomaly indexes are combined using an averaging or median scheme, with the averaging scheme having higher performance.

Peng et al. [10] present an information sharing model for distributed intrusion detection. A Cusum algorithm is used to collect statistics at local systems, while a learning algorithm decides when information has to be shared among the nodes, in order to minimise detection delay and reduce the communication overhead. Data are fused using the sum rule.

In [11], the authors describe a distributed change-point detection scheme for the detection of DDoS attacks over multiple network domains. At each router, a Cusum algorithm executes, raising alerts that are sent to a central server. Then, the server creates a subtree displaying a spatiotemporal vision of the attack. In a second hierarchy level, a global picture of the attack is created by merging all subtrees together.

The so-far described related contributions focus on local, distributed or collaborative schemes for attack detection at higher network layers (e.g. IP, TCP), whereas this work focuses on detecting jammers at the physical and medium access layers.

A similar work studying jamming at the physical layer appears in [12], where the authors describe several types of jammers and propose two types of detection algorithms, considering metrics such as the *packet delivery ratio*, the *bad packet ratio* and the *energy consumption amount*. The basic algorithm tries to detect jamming by using multiple if-else statements on the aforementioned metrics, while the advanced algorithm uses a distribution scheme where information is collected from neighbouring nodes. The evaluation shows high detection rates, but trade-offs regarding the FAR versus the DP or the robustness of the algorithms is not presented.

In [13], techniques that detect anomalies at all layers of a wireless sensor network are proposed. The authors show how the DP increases when the number of the nodes running the proposed procedure increases, but they do not show the trade-off with the FAR.

The authors of [14] show how the errors at the physical layer propagate up the network stack, presenting a distributed anomaly detection system based on simple thresholds. A method for combining measurements using the Pearson's product moment correlation coefficient is also presented. A disadvantage of this method is that "raw" RSSI measurements by several sniffers are needed. This could generate a high volume of traffic flowing from the sniffers to a main node where the algorithm executes. In contrast, our proposal is based on passive monitoring performed by a single node.

Several adversarial models are presented in [15], all focusing on RF jamming attacks. One of the proposed algorithms, applies *high order crossings* a spectral discrimination mechanism that distinguishes normal scenarios from two types of the defined jammers. The authors introduce two detection algorithms based on thresholds that use signal strength and location information as a consistency check to avoid false alarms.

The authors of [16] present a cross layer approach to detect jamming attacks. Jamming is performed at the physical layer by using RF signals, and at the MAC layer by targeting the RTS/CTS and NAV mechanisms of the IEEE 802.11 protocol. Jamming detection is split into two phases. In the first phase, simple threshold algorithms are deployed using metrics such as the physical carrier sensing time, the number of RTC/CTS frames, the duration of channel idle period and the average number of retransmissions. The second phase is triggered if there are threshold violations.

The authors of [17] describe ARES, an anti-jamming reinforcement system for 802.11 networks which tunes the parameters of rate adaptation and power control to improve the performance in the presence of jammers. However, ARES should be present in every wireless node in order to regulate rate and power while our system consists of a prototype based on passive measurements and no modifications are needed for the wireless clients. Furthermore, they consider a Jammer that creates interference (so it operates on neighbouring channels), while our prototype can also detect jammers emitting energy on the same channel, as well as detecting blocking attacks performed by powerful jammers that completely block the communication within their transmission range.

Cardenas et al. [5] consider the sequential probability ratio test. However, their work is about detecting MAC-layer misbehaviours and not attacks.

Wood et al. [18] propose DEEJAM, a MAC-layer protocol for defending against stealthy jammers using IEEE 802.15.4-based hardware. Nevertheless, as the authors note, against a powerful and more sophisticated jammer, DEEJAM cannot effectively defend the wireless network.

The authors of [19] propose a lightweight intrusion detection system that is however used for sensor networks and their related attacks (e.g. sinkhole attack), while our prototype is for infrastructure networks and different attack types.

Finally in [20], the authors describe a lightweight intrusion detection system for wireless mesh networks. Nevertheless, they study attacks (port scanning, consumption attacks, spam detection, etc.) that are not wireless-specific as those we studied in this article.

3 Network layout and jamming model

The network layout we use for testing our prototype is shown in Figure 1. This consists of off-the-shelf IEEE 802.11 devices that communicate through a wireless AP. The monitor node (MN) and the display server (DS) comprise our prototype for jamming detection. These two devices are inter-connected through a wired local area network (LAN) over a secure VPN tunnel. Jammer is a device that emits energy at pre-defined intervals, aiming to disrupt network operation.

Regarding the jamming attacks, there is always the trade-off between jamming intelligence and cost. An intelligent jammer can cause severe DoS attacks with a low energy consumption but its cost can be significantly high (e.g. [21]). On the other hand, a less sophisticated Jammer based on off-the-shelf hardware can cause significant performance degradation, although consuming more energy but it costs less and it can also be used by individuals with any specialised knowledge about network protocols and functionalities. We experiment with two types of jammers. The first one is based on a mini-ITX board that carries 512MB of RAM and an 80 GB hard disk (Figure 2a). This board is also equipped with an Atheros CM9-GP mini-PCI card, controlled by Ath5k, an open source IEEE 802.11 driver [22] running on Gentoo Linux. Two types of jamming are performed using this device:

- energy emission on the same channel (we call it as main channel in the rest of the article) legitimate nodes use for communication,
- energy emission on neighbouring channels.

In order to make the off-the-shelf node operate as Jammer, we modified the values of several hardware registers (through Ath5k) that are part of the Atheros wireless card, disabling the back-off and the clear channel assessment (CCA) mechanisms of IEEE 802.11. By

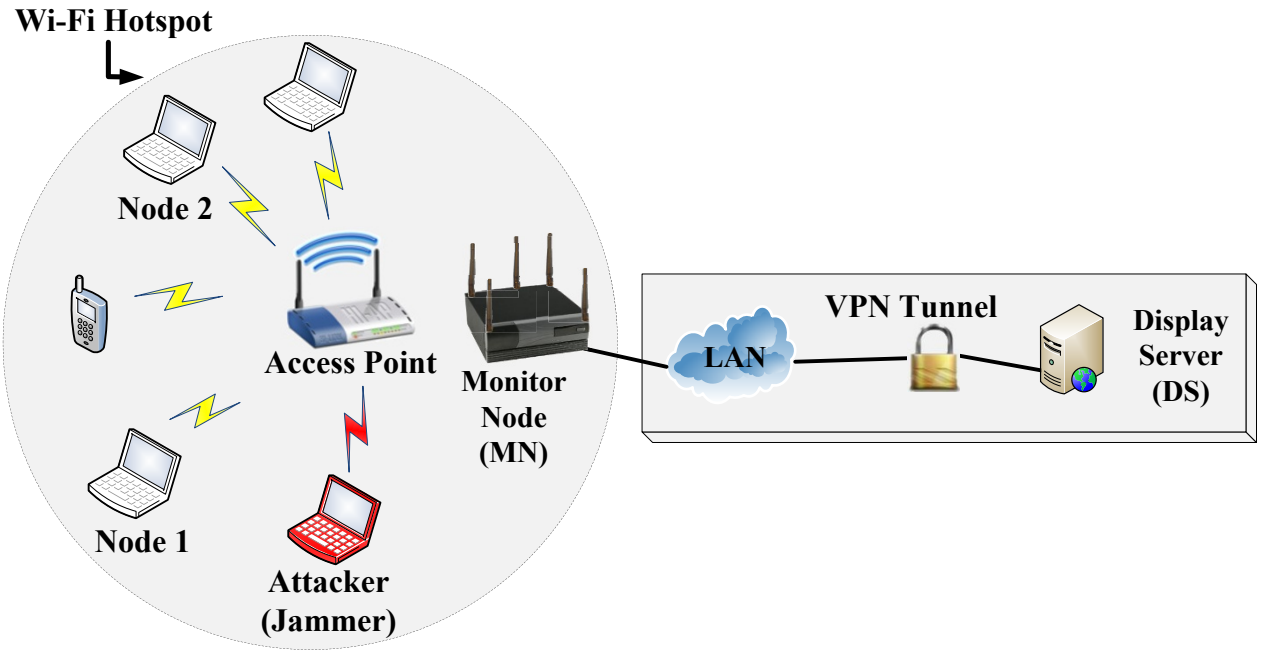

Figure 1 Experimental network layout.

disabling these mechanisms, Jammer becomes a non-compliant IEEE 802.11 node that is immune to the energy radiated by the legitimate nodes, thus it can freely perform jamming.

The second type of Jammer we use is based on the universal software radio peripheral (USRP), a family of hardware for making software radios (Figure 2b). The term software radio refers to re-programmable devices that can change their radio-frequency (RF) characteristics (e.g. carrier frequency, modulation, etc.) through software means. Popular software for modifying the USRP RF characteristics is GNU Radio [23] and Matlab [24]. This type of Jammer has several enhanced characteristics compared to the off-the-shelf type as: (i) it can emit signals in any carrier frequency, (ii) the transmission power granularity is smaller and more stable, and (iii) energy emission is possible without following any MAC-layer protocol (e.g. IEEE 802.11). We use this type of Jammer in order to launch blocking attacks, making the network completely inoperable. The rest of the attacks are launched using the off-the-shelf Jammer, as we want to demonstrate that this device can also cause severe network performance degradation. Nevertheless, our prototype can detect jamming regardless the type of the Jammer used.

Depending on the spectrum distance from the main channel Jammer operates, we define a number of different attacks. Table 1 shows these attacks (and subsequently the attacks our prototype can detect), and the hardware used (column 4 is discussed in Section 5).

In order to demonstrate how network performance deteriorates by Jammer's presence, we conduct an experiment using the network layout shown in Figure 1. The off-the-shelf Jammer broadcasts UDP traffic with a transmission rate of 5 Mbps on channel 40, in a periodic fashion (10 s of traffic transmission followed by 20s of inactivity). Furthermore, on channel 44 Node 1 continuously transmits UDP traffic (using iperf [25]) with a transmission rate of 27 Mbps to Node 2 (we used UDP as the transport protocol to avoid TCP's congestion control mechanism). MN is set to promiscuous mode recording the SINR in a per packet basis, only for the packets transmitted by the AP. The packet loss and throughput (for the flow between Nodes 1 and 2) are provided by iperf. Figure 3 shows how the SINR, throughput, and packet loss are affected during the jamming attacks (these are depicted by the orthogonal boxes). SINR drops about 50%, throughput degradation is over 85%, while the packet loss increases more than 50%.

4 Jamming detection

In our previous works ([6,7]), we investigated several algorithms for jamming detection, all based on the SINR. Among all, Cusum Max-Min (C_{mm}) has the best performance in terms of the DP, FAR, and robustness (the term robustness is analysed in Section 6). Cusum belongs to the category of the Cusum algorithms, detecting changes of a certain distribution (change-point detection), and it has been widely used in the literature

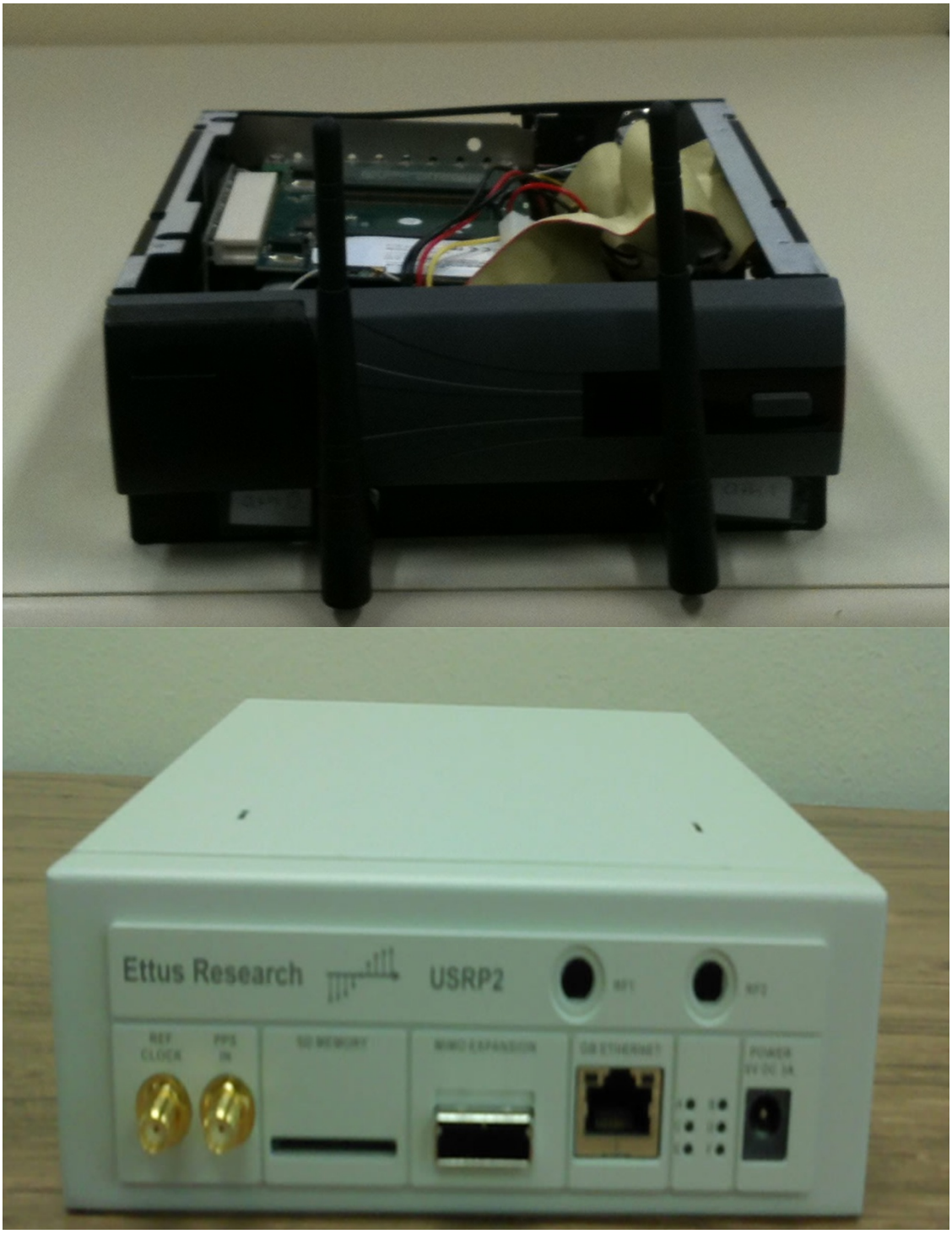

Figure 2 Types of jammers.

for anomaly-based intrusion detection (e.g. [5,11,26-28]). In general, there are two types of Cusum algorithms: (i) parametric and (ii) non-parametric. Parametric Cusums are used when a parametric model for $\{x\}$, where x is an independent and identically distributed (i.i.d) random variable, is known. Using the parametric model, a Cusum algorithm can detect whenever a change to $\{x\}$ takes place. On the other hand, non-parametric Cusums

Table 1 Jamming attacks

Symbolic name	Description	Type of jammer	Metric for detection
Jam1	Jammer emits energy on neighbouring channels of the main channel	Off-the-shelf	SINR
Jam2	Jammer emits energy on the main channel or neighbouring channels completely blocking the network operation (blocking attack)	Software defined radio	Beacons loss
Jam3	Jammer emits energy on the main channel	Off-the-shelf	Ratio of the corrupted packets over the correctly decoded packets

are used when the model of $\{x\}$ cannot be known. This is the case for our jamming detection techniques and their associated metrics, where the distribution of $\{x\}$ cannot be known in advance. Therefore, C_{mm} is a non-parametric Cusum algorithm defined using the following formula:

$$y_n = \begin{cases} y_{n-1} + Z_n - a & \text{if } y_n \geq 0 \\ 0 & \text{if } y_n < 0 \end{cases} \quad (1)$$

Z_n is the expectation of a specific metric that changes whenever jamming takes place, and $a \in R^+$ controls its drift. C_{mm} that executes on MN aims to detect these changes signalling the appropriate alarms whenever y_n exceeds a pre-defined detection threshold h. As an example, Figure 4 shows how the metrics expectations (Z_n) are affected during each different attack using an experimental test-bed with six wireless nodes, the AP and the Jammer.

At this point, we provide the rationale behind the metrics we consider for jamming detection (shown in Table 1). For the detection of Jam1, where the Jammer emits energy on neighbouring channels, we consider the SINR. SINR drops when Jammer is on, as intereference (and/or the noise) increases. MN computes the SINR for the beacon packets transmitted by the AP.

Jam2 takes place when the Jammer manages to completely block the communication between the wireless nodes. As we have verified from several experiments, the USRP Jammer (Figure 2b) can easily block the wireless communications when its carrier frequency is close to that of the main channel. This happens because when Jammer is on, all nodes (including the AP) defer from transmission either because the channel is continuously occupied, or the noise level is above their CCA level. During this attack, SINR or any metric based on the received packets cannot be used, as no packets are

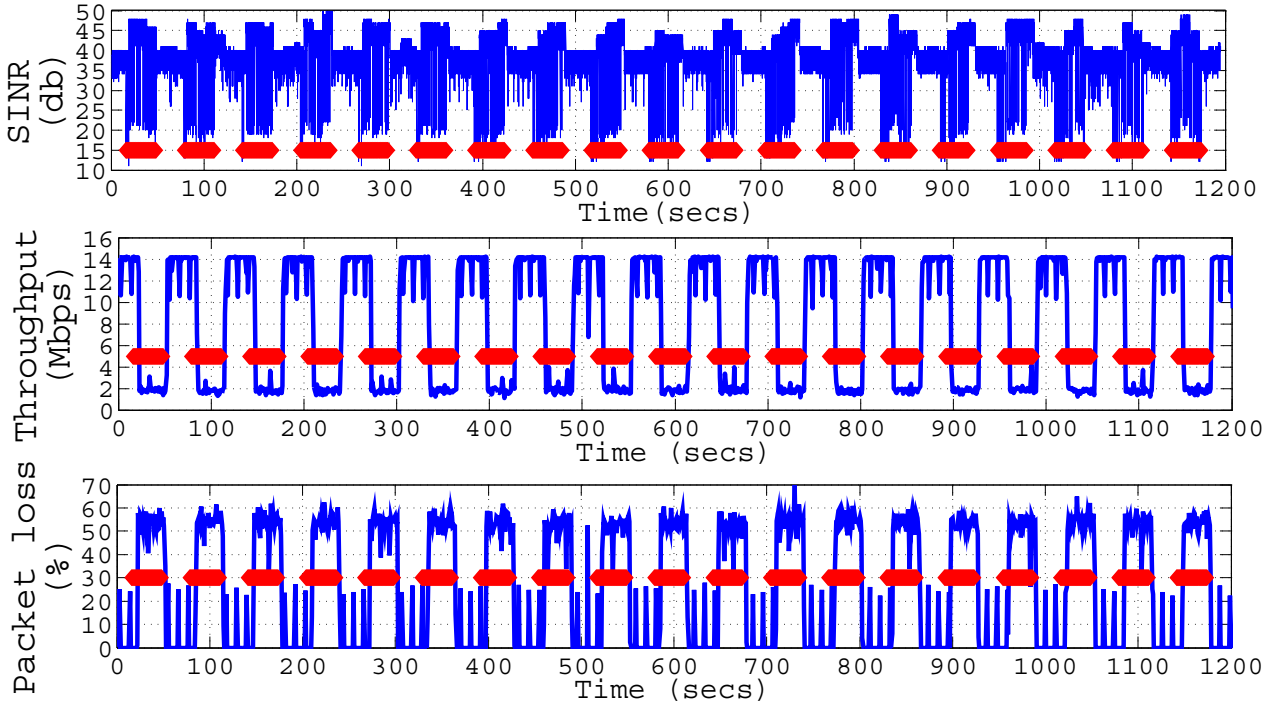

Figure 3 Jamming effect on the SINR, throughput, and packet loss.

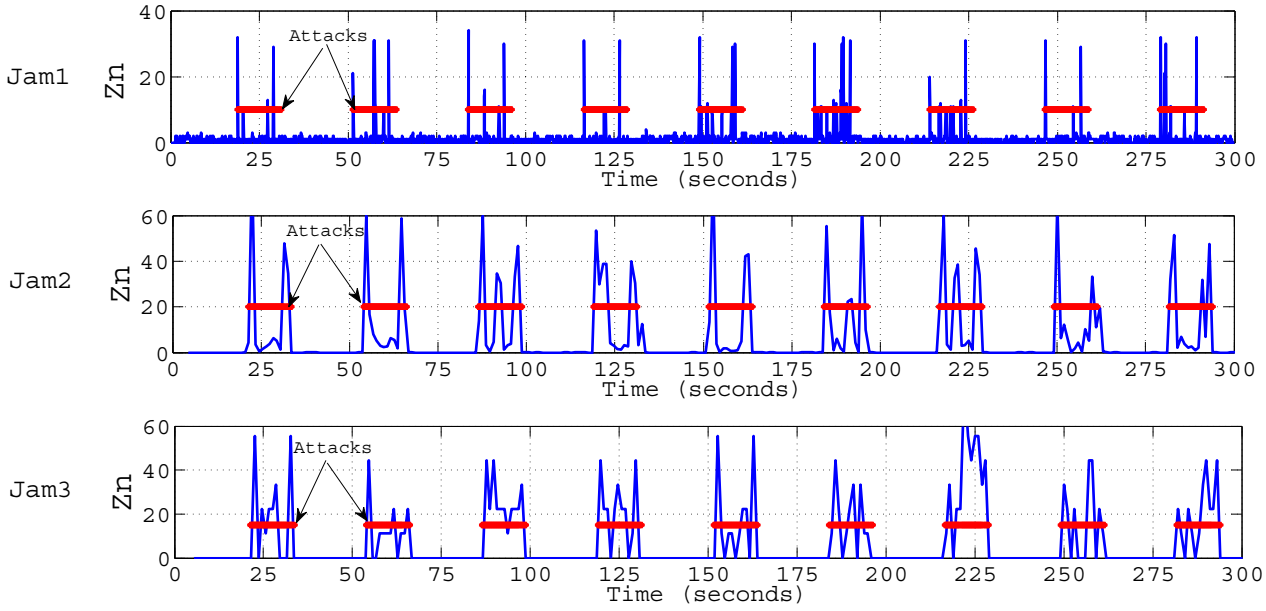

Figure 4 Metric expectations for the different attacks.

transmitted; therefore, no packets can be recorded by MN. To overcome this limitation we consider as metric the beacon loss. Beacon loss is estimated using the number of the received beacons within a time window, and the number of the beacons that should have been received within that period (AP transmits beacons in a pre-defined interval). During the blocking attacks beacon loss can reach 100%.

When Jammer operates on the main channel (Jam3), SINR does not drop because Jammer's signal is not regarded as interference. However in this case, the number of corrupted packets increases as Jammer does not perform any spectrum sensing and/or backoff, hence the probability of collision substantially increases. For this reason, we use as metric the ratio of the corrupted packets over the correctly received packets. Both types of packets are measured in the wireless interface of MN that is configured for the main channel.

In total, there are three different metrics one for the detection of the three types of attacks. MN uses three different threads applying the C_{mm} algorithm independently for each metric, signalling the appropriate alarm. Each C_{mm}'s functionality is based on two sliding windows: a short one and a long one. For the measured value x_n for sampling n, the maximum-minus-minimum value of the x is computed within the short window as

$$D(n) = \max_{n-K+1<i\leq n} x_i - \min_{n-K+1<i\leq n} x_i,$$

where K is the length of the short window. Next, the average maximum-minus-minimum x is estimated in the long time window as:

$$\bar{D}(n) = \frac{\sum_{i=n-M+1}^{n} D(i)}{M},$$

where M is the length of the long window. Now, Z_n is given by $Z_n = D(n) - \bar{D}(n)$ and finally from Equation (1), an alarm is raised if $y_n \geq h$.

For the different attacks we consider different values for the long and short windows. For Jam1, we choose K = 10 and M = 100. As our AP transmits beacons every 100 ms, the short and long windows are equivalent to 1 and 10 s, respectively. For Jam2 and Jam3, we use K = 1 and M = 10, as the monitoring periods for the corrupted/correct packets and the beacon packets are set to 1 s.

5 Prototype implementation

This section describes the prototype implementation for attack detection (Figure 1). MN performs passive measurements in the wireless network, it executes C_{mm} for the metrics described in Section 4 (shown in Table 1), and reports its findings (metric values, output of the detection algorithms) to DS that is exclusively used for displaying purposes.

5.1 The monitor node

Monitor node is implemented using a mini-ITX board with a VIA Esther processor 1,300 MHz and 512MB of RAM (Figure 2a) with Gentoo Linux and Ath5k as the wireless driver. It splits into two main software parts: the kernel module and the user-space module (Figure 5). The kernel module contains the code of the Ath5k

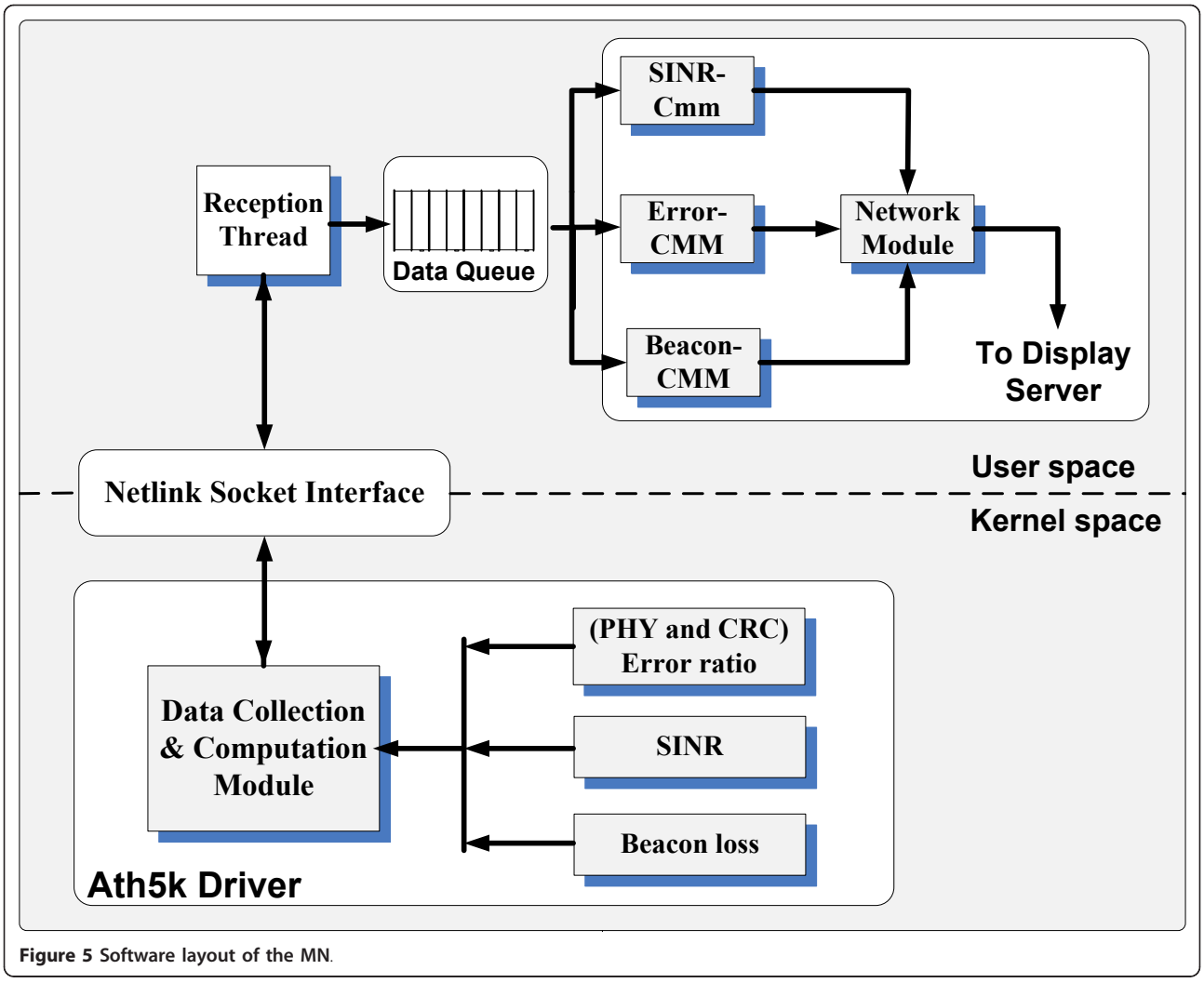

Figure 5 Software layout of the MN.

driver and several software functions we have implemented within it for the collection of information regarding the SINR values, the PHY (physical) and cyclic-redundancy-check (CRC) errors, and the beacon loss. PHY errors are related to errors monitored at the physical layer of the medium, while CRC are the errors at the MAC layer. Packets with these errors are characterised as corrupted. The required information is collected with the aid of Ath5k using our three kernel threads, each assigned with a specific task.

The first thread monitors the number of the CRC and PHY errors (these are reported by the driver), as well as the correctly decoded packets captured by the interface that is configured for the main channel and set to promiscuous mode. Every 1s, it reports to the error-CMM thread, residing in user-space, the ratio of the corrupted packets (due to CRC or PHY errors) over the correctly decoded packets.

The second thread computes the SINR every time a beacon packet is captured by MN's main interface (this is the wireless card configured to listen to the main channel in promiscuous mode). Ath5k reports signal-to-noise ratio (SNR) values and not SINR. However, as we have verified from several experiments, SNR drops when the USRP Jammer is used, emitting energy without following the IEEE 802.11 protocol, but it does not drop when energy is emitted following it. This is also the case when the off-the-shelf Jammer is used. On the other hand, SINR should drop in all cases, as either the noise, or the interference level increase when a Jammer is present. In order to measure the SINR based on Ath5k's reporting, we use the following method.

Ath5k can report signal and noise values independently for each configured wireless interface. MN is equipped with five wireless interfaces so we are firstly able to measure the signal S_m and the noise N_m in the

main interface. Moreover, by setting the rest four interfaces in the immediately adjacent and next adjacent channels, and taking into account that we experiment in an IEEE 802.11a network where the channel separation is 20 MHz, we can monitor the signal levels in five consecutive channels. Therefore, we compute the SINR in the main channel based on the measurements from these five interfaces. However, no more than five interfaces are required because as our experiments (and the related literature) have shown that energy emitted using a channel 60 MHz or more far from the main channel does not have any impact on network's performance.

According to [29] the leakage of the immediately adjacent channels (IACs) to the main channel is X_1 = 22.04 dB, while for the next adjacent channels (NACs) is X_2 = 39.67 dB. Based on these values, we assume that the interference generated by a signal of S dBm in the IACs is S - X_1 dB, while for the NACs *is* S - X_2 dB. Based on these, the total interference-plus-noise power on the main channel is calculated as follows:

$$I\_N_m(\text{dB}) = 10\times\left[\log_{10}\left(10^{\frac{S_{+1}-X_1}{10}}+10^{\frac{S_{-1}-X_1}{10}}+10^{\frac{S_{+2}-X_2}{10}}+10^{\frac{S_{-2}-X_2}{10}}+N_m\right)\right] \quad (2)$$

The signals referred to the four neighbouring channels (NACs+IACs) S_i, $i \in$ [-2, -1, +1, +2], are the average signals estimated on each channel within a pre-defined time window (5 ms). N_m is the noise measured in the main interface. Finally, SINR is given by

$$\text{SINR} = S_m - I\_N_m, \quad (3)$$

where S_m is the signal measured in the main interface (per received beacon). The SINR values are then reported to the SINR-CMM thread that resides in user-space.

The last kernel thread counts the number of the beacon packets received within a time period (1s) by MN's main interface. Then, it computes the beacon loss based on this number and the known expected number of beacons. For example, our AP broadcasts 1 beacon every 100 ms, captured by the wireless main interface. The estimated beacon loss is then reported to the Beacon-CMM thread.

The above information is collected in kernel-space with the use of several functions of the Ath5k driver and then it is asynchronously transmitted to the user-space modules through the netlink socket interface. In user-space, the reception thread receives the acquired data and stores them into a software queue. Data are de-multiplexed out by the associated threads for further processing. SINR-CMM, Error-CMM and Beacon-CMM are independent threads executing in the multi-threaded environment of Linux, reading data from the data queue, so as the reception of the new data coming from the kernel-space is not blocked by the user-space operations.

Each of the threads executes the C_{mm} algorithm using its associated metric and then, through the network module, sends the output of the algorithm along with the current value of the metric to DS for displaying. Consequently, MN provides the following information to DS:

- the SINR values in a per (received beacon) packet basis,
- the output of the SINR-CMM thread that can signal the Jam1 attack detection,
- the beacon loss percentage estimated within a predefined time window,
- the output of the Beacon-CMM thread that can signal the Jam2 attack detection,
- the ratio of the corrupted packets over the correctly decoded packets estimated within a predefined time window,
- the output of the Error-CMM thread that can signal the Jam3 attack detection.

The communication between MN and DS is performed using a protocol we have designed that runs over UDP. Furthermore, we use a VPN tunnel between these hosts in order to provide enhanced authentication and encryption functionalities, securing the data flowing between them.

5.2 The display server

The DS mainly provides display functionalities to a network administrator and it consists of two threads. The first thread opens a UDP socket waiting for data from the MN. Any received information is stored into a software queue. The second thread reads data from this queue and displays them through the Gtk interface [30].

Figure 6 shows a snapshot of the DS monitor. The upper three boxes show the metrics computed in MN's kernel-space with the aid of Ath5k's functions, while the lower boxes show the outputs of the C_{mm} algorithms. For demonstration purposes, we launched the three types of possible attacks (Table 1) one after the other. During Jam1, Figure 6 shows that SINR reduces from 40 to 15 db, while the output of the corresponding C_{mm} algorithm significantly increases (Equation 1). Similarly for Jam2, the beacon loss goes to 100%, and for Jam3 the error ratio increases up to 35%, while the outputs of the corresponding C_{mm}s substantially increase.

6 Experimental results and performance evaluation

In the previous section, we described the layout of our lightweight prototype and the associated information

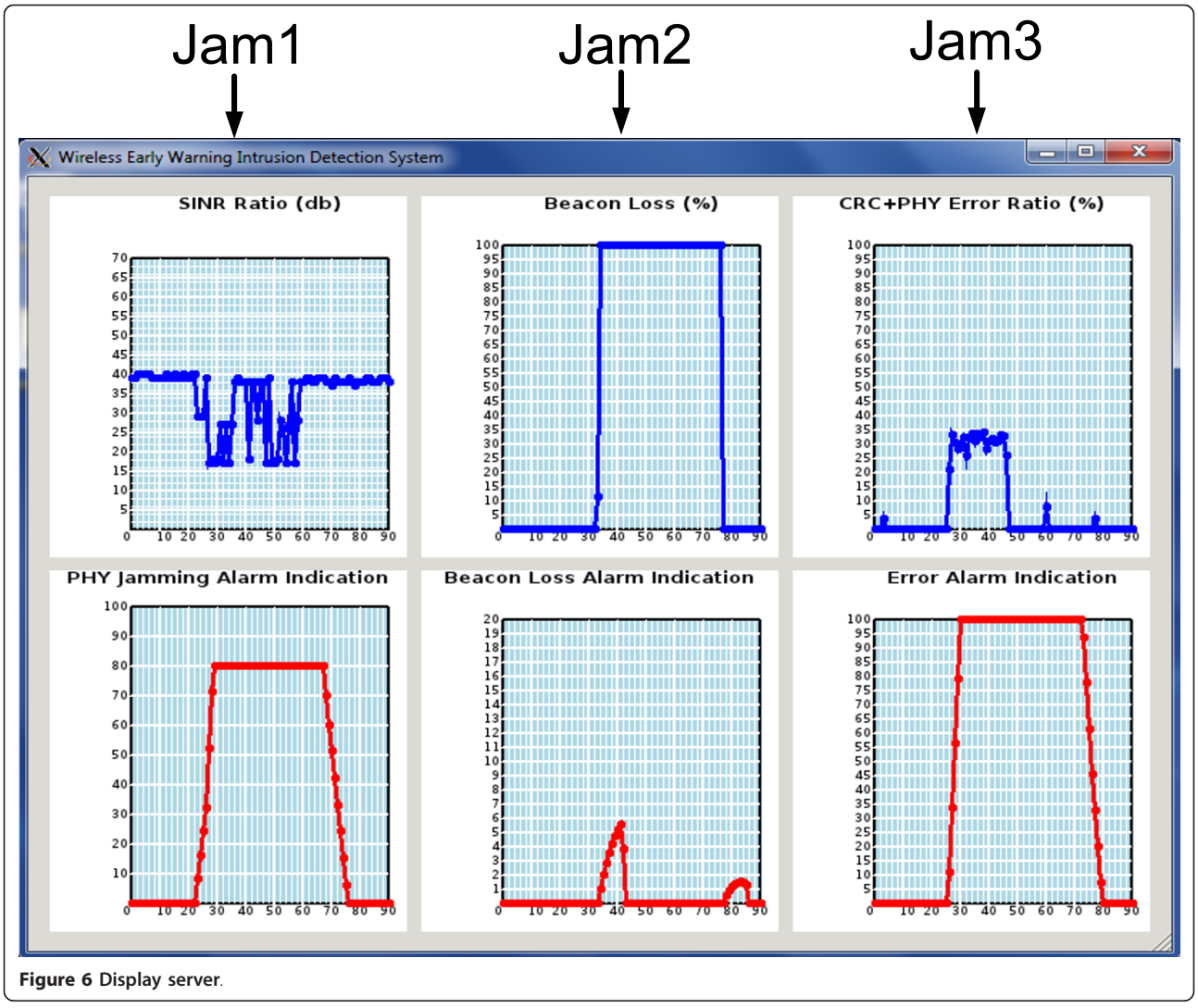

Figure 6 Display server.

displayed through DS; however, DS does not generate or display alarms. An alarm is signalled whenever the output y_n of a C_{mm} algorithm (Equation 1) exceeds a predefined detection threshold h. The challenge is to select the optimum value of h as well as of a, the value that controls the drift of expectation Z_n (Figure 4). We name a as *drift coefficient* throughout the rest of the article.

In this section, we investigate the performance of the prototype for the three different types of jamming attacks in terms of the DP, FAR and its robustness to different detection threshold values. DP is defined as the number of the detected attacks over the total number of the attacks. FAR is the ratio of the number of false alarms over the total duration of the experiment, in minutes.

6.1 Robustness

Traditionally, performance evaluation is presented by showing the trade-off points between FAR and DP (e.g. [8,11,31-33]). The performance of an algorithm increases when its associated trade-off points are closer to the left top corner of each graph (higher DP with a lower FAR). Although this is a significant method for performance evaluation, it is not complete as it provides no information regarding the robustness of the algorithms. By robustness we mean how the performance in terms of the DP and FAR varies, when the detection threshold changes. Moreover, this simplistic approach is not appropriate when the number of the algorithms under evaluation, or the experimental data increase, as it is predicated on subjective criteria. We define that an algorithm is robust if the metric we consider that combines the DP and FAR changes no more than 20%, when the detection threshold changes by more than 20%. This metric (we name it as score) is given by:

$$S = b * (c - d), \tag{4}$$

where $d = \sqrt{\mathrm{FAR}^2 + (1 - \mathrm{DP})^2}$ is the distance of a trade-off point (for a specific threshold h) from the optimum point (DP = 1 and FAR = 0), and $b,c \in \mathbb{R}^+$.

We evaluate the performance of the C_{mm} algorithms in two steps: (i) we filter the robust DP, FAR trade-off points and we calculate the maximum scores for the different values of a using Equation (4) and (ii) based on the maximum scores we select the optimum value of a, then we re-evaluate the algorithm for the different detection threshold values for the specific value of a, and finally we present the associated trade-off points. Algorithm 1 describes how the maximum score values are computed.

6.2 Experiments

For the evaluation, we conducted a number of experiments using the layout shown in Figure 1, varying the number of the wireless clients, the channel Jammer is operating and the attack intensity. We also use the Airlive WLA-5000AP as AP. For Jam1 and Jam3 we use the off-the-shelf Jammer, while for Jam2 the USRP. In all experiments Jammer emits energy for 10s followed by 20s of inactivity. Jamming intensity changes by varying the throughput of the Jammer. Based on the throughput degradation, delay, delay jitter and loss increase in the wireless network, we define three attack intensities: high, medium and low. Both legitimate and jamming traffic are transmitted using iperf, and UDP as the transport protocol. Legitimate nodes transmit traffic of 1.5 Mbps and logs are collected in DS for further processing. Table 2 shows the conducted experiments.

6.3 Performance evaluation

A key issue for the performance evaluation is the selection of the optimum drift coefficient a. This is performed running Algorithm 1 for the three types of C_{mm} algorithm, and for all experiments. Selecting b = 5 and c = 4 in Equation (4), the maximum score an algorithm can have (when DP = 1 and FAR = 0) is 20. We make the selection of the optimum value of a using several criteria. SINR_CMM is used to detect attacks when Jammer operates on NACs or IACs, while ERROR_CMM is for jamming detection when Jammer operates on the main channel. BEACON_CMM is for the detection of blocking attacks. For these reasons, SINR_CMM should have high detection and low FARs in NACs and IACs, while at least low FARs if Jammer is present on the main channel. Similarly, ERROR_CMM should have high detection and low FARs when Jammer is on the main channel, while at least low FARs if it is on NACs and IACs. Finally, BEACON_CMM should have high detection and low FARs when Jammer completely blocks the communication, regardless the channel Jammer operates on, while it should have at least low FAR in all other situations.

Using Algorithm 1 we compute the maximum scores against a for all algorithms, and for all the experiments. Indicatively, Figure 7 shows the maximum score values for ERROR_CMM when Jammer emits energy on the main channel. Based on these graphs, we select a for each Cusum algorithm according to our pre-defined criteria (we finally select a = 3 for the SINR_CMM, a = 8 for the ERROR_CMM, and a = 0.5 for the BEACON_CMM). Further work can include the automated selection of a based on more criteria. For example, Equation (4) assigns the same score to DP and FAR. However, a network operator could classify FAR as more important than DP; hence, different weights should be used for the score computation. Nevertheless, our results show that all algorithms have high DP and low FAR.

6.3.1. Performance evaluation of the ERROR_CMM algorithm

We begin by presenting the evaluation of ERROR_CMM; the algorithm that aims to detect attacks launched by jammers operating on the main channel. Having selected a = 8, Figure 8 shows DP versus FAR, for different values of the detection threshold, and when the (off-the-shelf) Jammer operates on the main channel. Recall that these are the robust trade-off points selected using Algorithm 1. The three upper graphs show that for the high intensity attack (Jammer's throughput is 5 Mbps), ERROR_CMM detects all attacks with zero false alarms. For the medium intensity attack (middle graphs), all attacks are detected with zero false alarms, except for Exp_2_1 where FAR = 0.2 (false alarms per minute) that is however a low FAR. For the low intensity attack, for Exp_3_2 and Exp_3_3, ERROR_CMM detects no attacks and has no false alarms; therefore, the corresponding graphs are blank. For Exp_3_1, both DP and FAR are low. The miss detections for the low intensity attacks cannot be regarded as a significant issue, as the impact of these attacks on network's performance is negligible. When Jammer operates on the NAC or IAC, ERROR_CMM detects no attacks and has zero false alarms.

Summarizing, based on these experiments and by choosing h = 10 and a = 8, ERROR_CMM can detect all attacks launched by the Jammer operating on the main channel with a maximum FAR of 0.2 false alarms per minute (Figure 8). For the rest two attacks (Jam1, Jam2) it has zero false alarms and zero attacks are detected.

6.3.2. Performance evaluation of the SINR_CMM algorithm

Next we present the evaluation of SINR_CMM, the algorithm aiming to detect attacks caused by jammers operating on NACs or IACs. Repeating the procedure

Table 2 Experiments

Experiment id	Number of clients	Channel of jammer	Jammer throughput (Mbps)	Attack intensity
Exp1_1	6			
Exp1_2	4	Main	5	High
Exp1_3	2			
Exp2_1	6			
Exp2_2	4	Main	3	Medium
Exp2_3	2			
Exp3_1	6			
Exp3_2	4	Main	1.5	Low
Exp3_3	2			
Exp4_1	6			
Exp4_2	4	Immediately adjacent	5	High
Exp4_3	2			
Exp5_1	6			
Exp5_2	4	Immediately adjacent	3	Medium
Exp5_3	2			
Exp6_1	6			
Exp6_2	4	Immediately adjacent	1.5	Low
Exp6_3	2			
Exp7_1	6			
Exp7_2	4	Next adjacent	5	High
Exp7_3	2			
Exp8_1	6			
Exp8_2	4	Next adjacent	3	Medium
Exp8_3	2			
Exp9_1	6			
Exp9_2	4	Next adjacent	1.5	Low
Exp9_3	2			

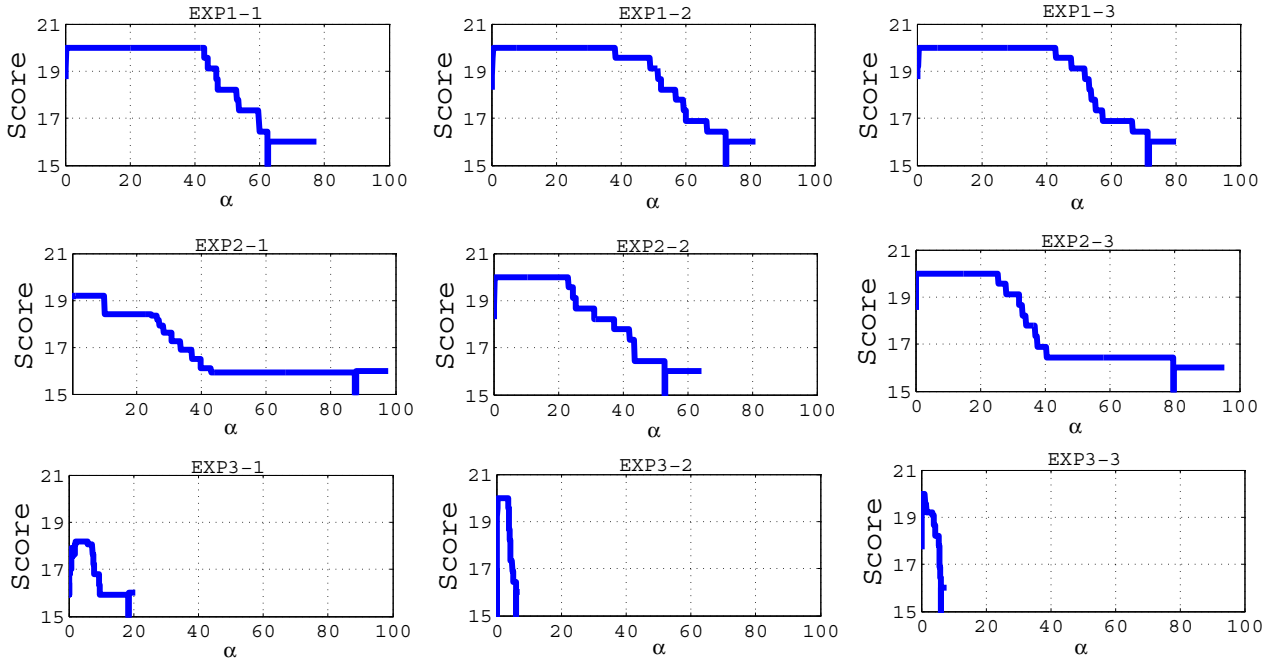

Figure 7 Maximum score versus the drift coefficient for the ERROR_CMM when jammer operates on the main channel.

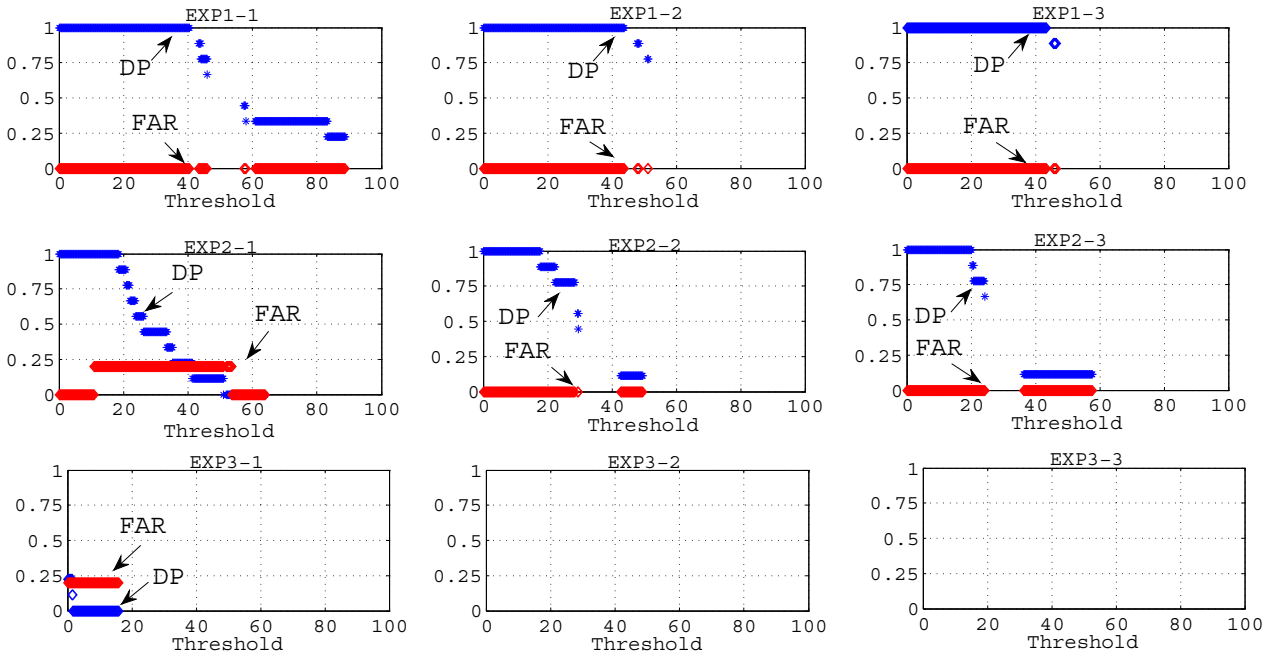

Figure 8 Performance evaluation of the ERROR_CMM when Jammer operates on the main channel.

with Algorithm 1, we select $a = 3$. Based on this value, Figure 9 shows the evaluation for the experiments Jammer emits energy in the IAC. For all the experiments and attack intensities, SINR_CMM detects all attacks with zero false alarms, except in the case of Exp_5_3 and Exp_6_3, where all attacks are detected with a maximum FAR of 0.2 false alarms per minute.

In Figure 10, we show the evaluation when Jammer operates on the NAC. Again, it can detect all attacks

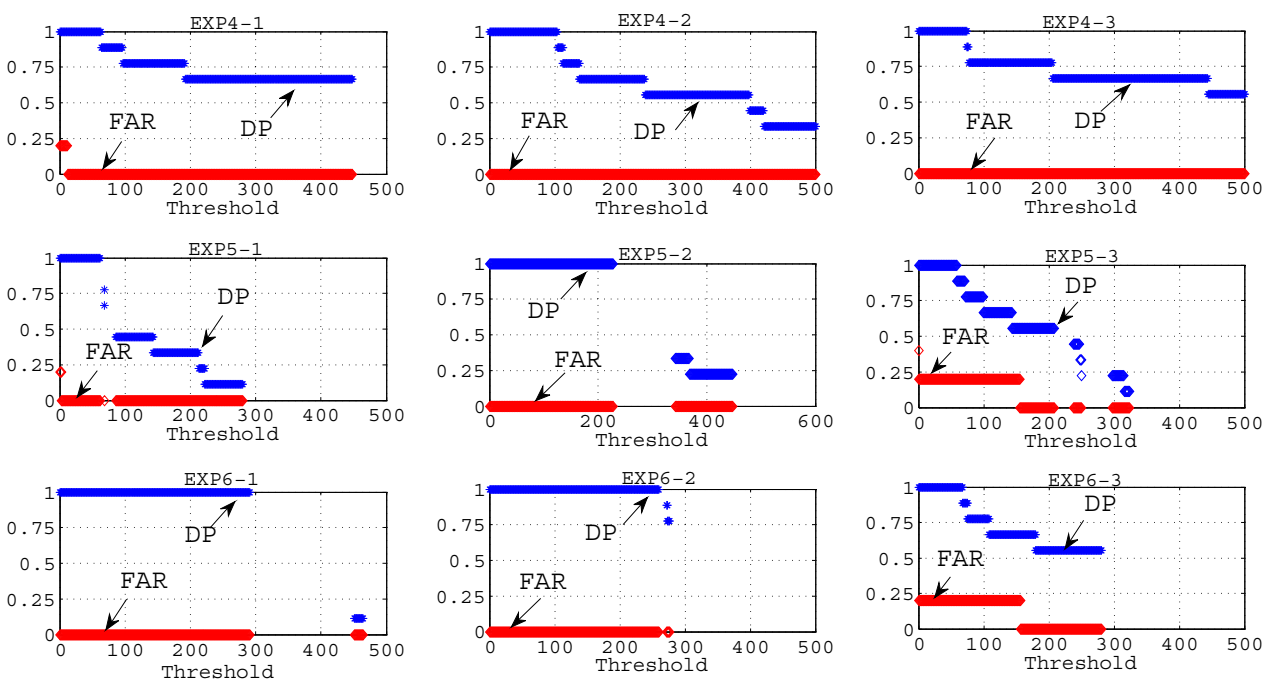

Figure 9 Performance evaluation of the SINR_CMM when Jammer operates on the immediately adjacent channel.

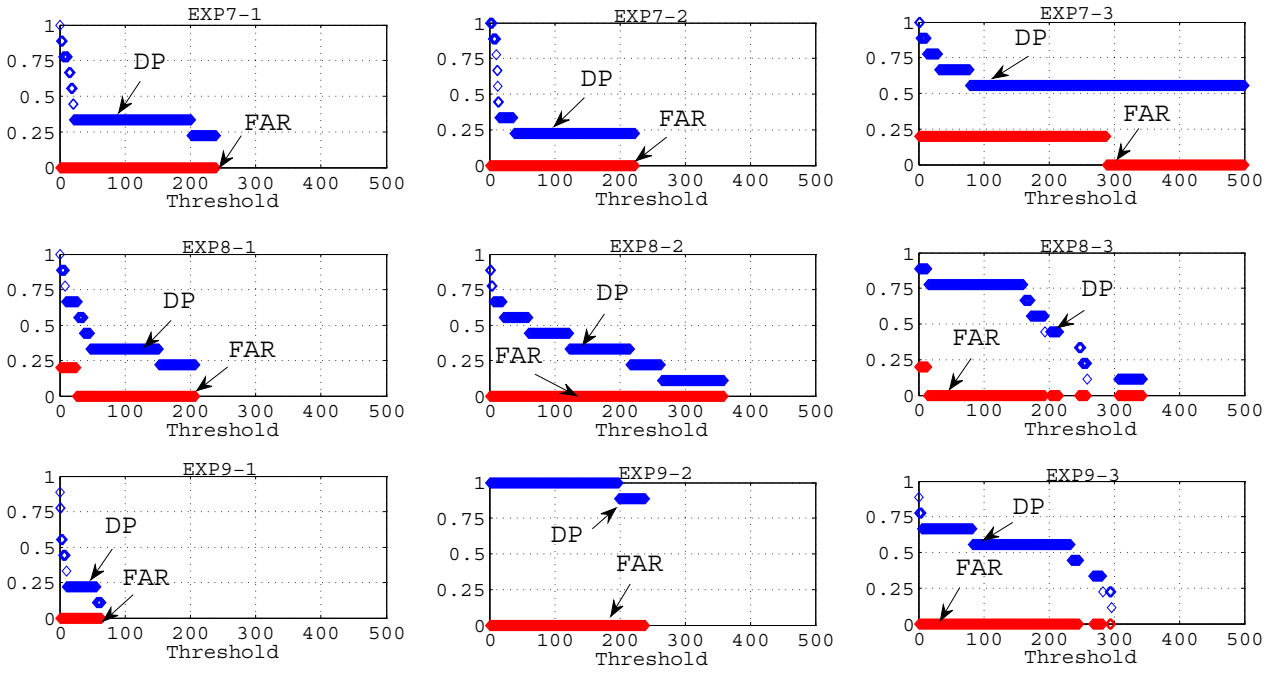

Figure 10 Performance evaluation of the SINR_CMM when Jammer operates on the next adjacent channel.

with zero false alarms, except in three cases (Exp_7_3, Exp_8_3, Exp_9_1), where FAR can increase up to 0.2.

These results show that the SINR-based metric we consider can achieve high detection rates with low false alarms when Jammer emits energy in neighbouring channels. However, it is important to investigate its performance when Jammer emits energy on the main channel. As Figure 11 shows, SINR_CMM has low FAR (less than 0.2). Also, DP is low; however, this is not an issue, as the ERROR_CMM can detect this type of attack.

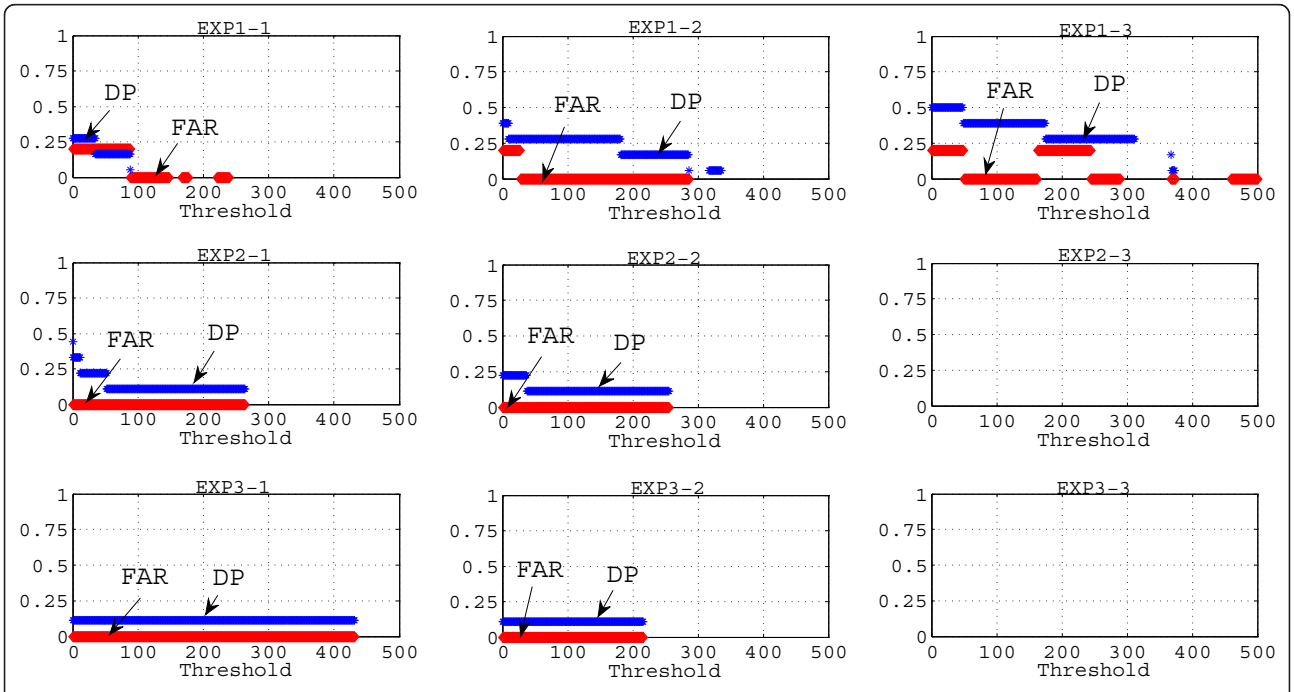

Figure 11 Performance evaluation of the SINR_CMM when Jammer operates on the main channel.

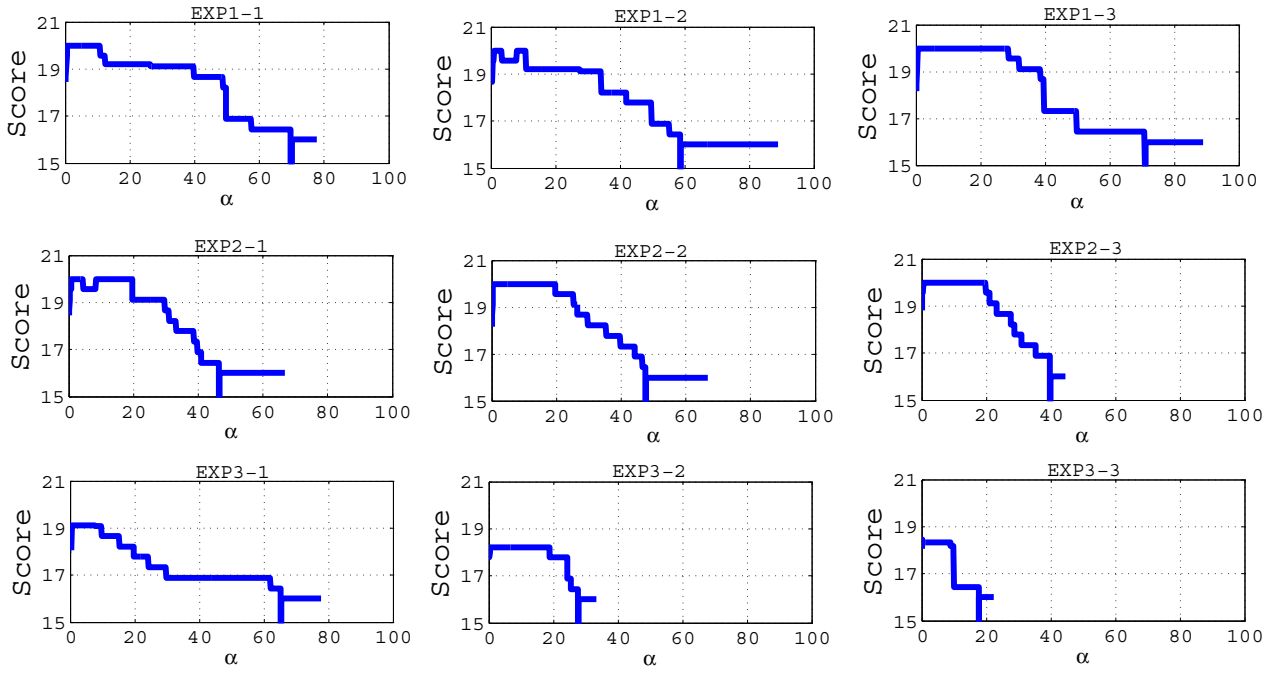

Figure 12 Score versus the drift coefficient for BEACON_CMM when Jammer operates on the main channel.

6.3.3. Performance evaluation of the BEACON_CMM algorithm

BEACON_CMM is used for the detection of blocking attacks; attacks launched by powerful jammers that completely block network's operation. During a blocking attack, AP is unable to transmit packets, hence beacon loss reaches 100%. This algorithm uses as metric the estimated beacon loss within a time window.

Figure 12 shows BEACON_CMM's score versus the drift coefficient a when the off-the-shelf Jammer emits energy on the main channel. From this figure it is obvious that with a proper selection of a, BEACON_CMM can detect attacks in the main channel of high or medium intensity, as it can reach the maximum score (20). For the low intensity attack, its performance deteriorates. However, when Jammer operates on the IAC, BEACON_CMM's performance deteriorates for the low intensity attacks (Figure 13), and it further deteriorates when Jammer is on the NAC for all intensities (Figure 14). Low scores can indicate low DP and/or high FAR. As we aim to detect blocking attacks, the selection of a should satisfy two requirements: (i) BEACON_CMM shall have high DP and FAR for the blocking attacks and (ii) BEACON_CMM shall not give high FAR for the rest of the attacks. The second requirement can be satisfied by selecting $a > 70$ as, Figures 12, 13 and 14 show that BEACON_CMM will not trigger any alerts, so no false alarms will be generated. Using our USRP Jammer (Figure 2b) we conducted a number of blocking attacks, aiming to select the optimum value for a. When $a = 70$, BEACON_CMM can detect all blocking attacks with zero false alarms. For the rest two attacks it has zero false alarms and zero DP.

7 Conclusions-Further work

In this work, we described and evaluated anomaly-based intrusion detection algorithms executing on real light-weight prototype. The algorithms use the Cusum change-point detection technique seeking for changes using three different metrics: SINR, ratio of corrupted over correctly decoded packets and beacon loss; trying to detect three different types of jamming attacks.

We introduced the term of robustness that shows if DP and FAR remain relatively stable as the detection threshold increases. We also proposed an algorithm that filters the robust trade-off points and assists a network administrator to select the optimum value of the drift coefficient.

We evaluated the algorithms collecting traces from a real experimental network. The evaluation of SINR_CMM shows that it can detect all Jam1 attacks (attacks on neighbouring channels) with a maximum FAR of 0.2. For Jam3 it detects no attacks and it has a maximum FAR of 0.2. BEACON_CMM can detect all Jam2 attacks (blocking attacks) with zero false alarms. For Jam1 and Jam3 it detects no attacks and has zero false alarms. ERROR_CMM can detect all Jam3 attacks (attacks on the main channel) with zero false alarms.

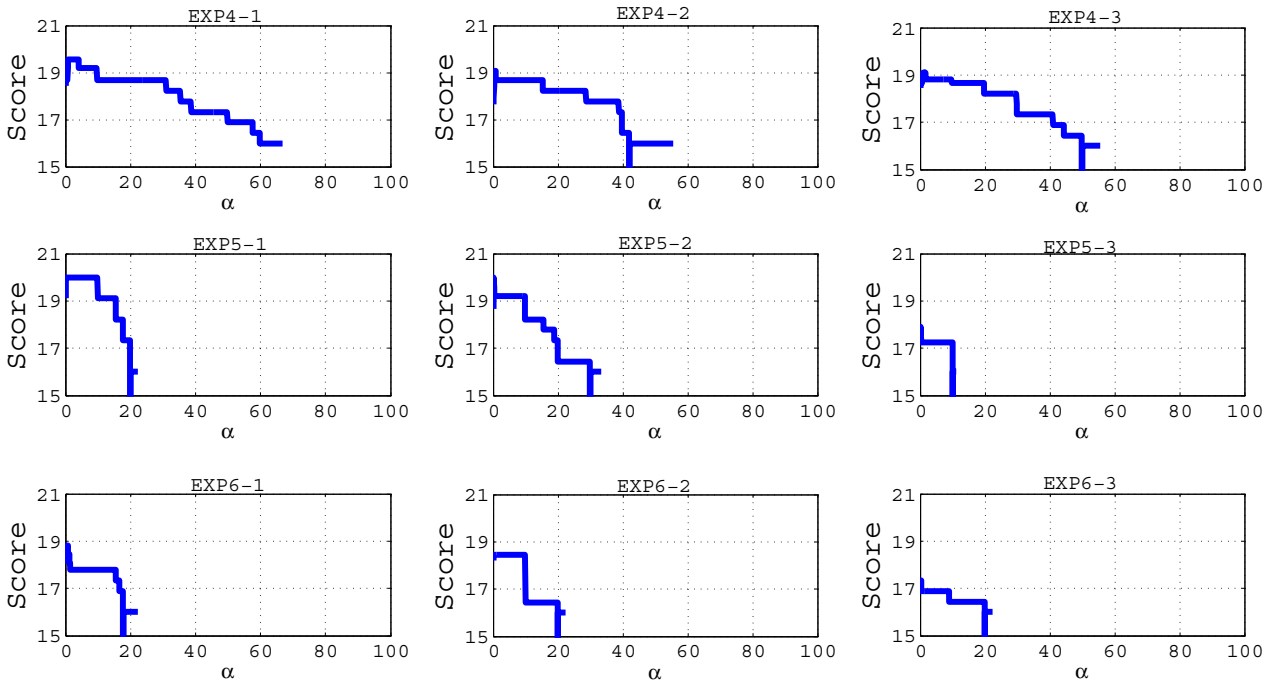

Figure 13 Score versus the drift coefficient for BEACON_CMM when Jammer operates on the immediately adjacent channel.

For Jam1 and Jam2 it detects no attacks and has zero false alarms.

Further work includes the selection of the drift coefficient based on more criteria that will also allow a network administrator to assign different weights to DP and FAR. Furthermore, we will build a notification mechanism (e.g. through emails) so as a network administrator is alerted when the outputs of the C_{mm} algorithms exceed the detection thresholds.

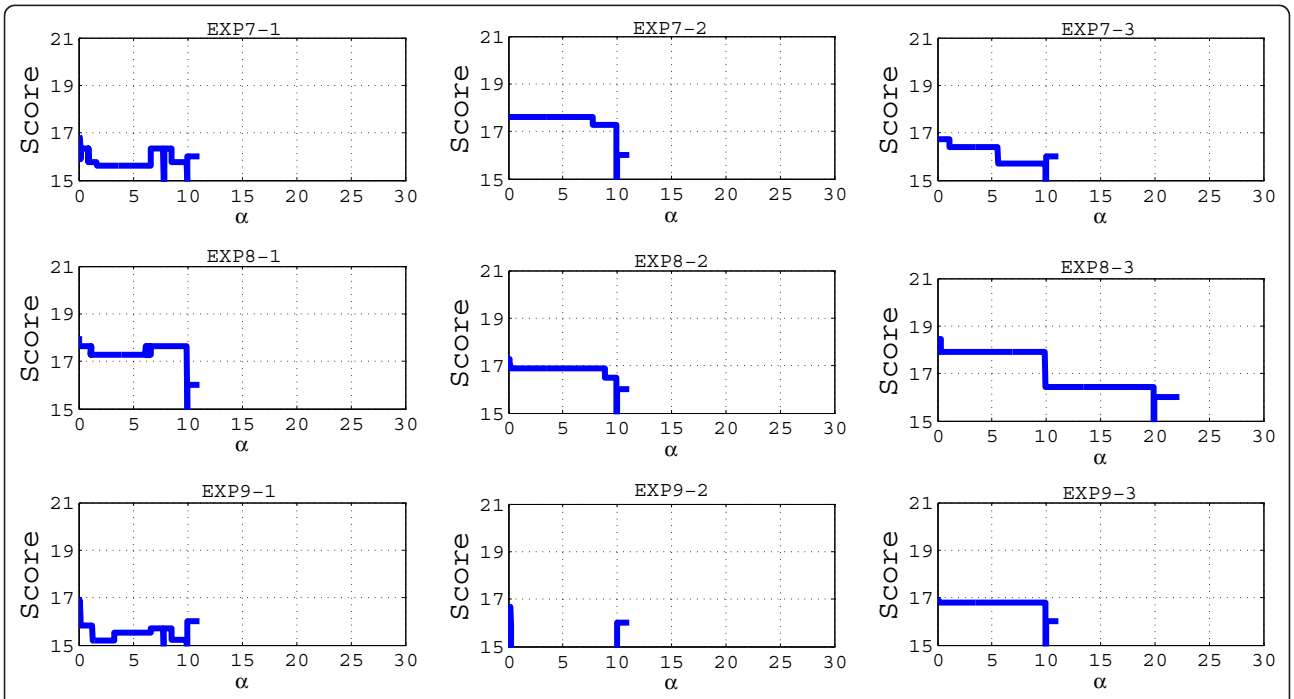

Figure 14 Score versus the drift coefficient for BEACON_CMM when Jammer operates on the next adjacent channel.

Algorithm 1

Max-score computation

Variables:

a_{min} the minimum value of drift coefficient a

a_{max} the maximum value of drift coefficient a

a_{step}: the step that evaluation process proceeds from a_{min} to a_{max}

thr_{min}: the minimum value of the detection threshold

thr_{max}: the maximum value of the detection threshold

thr_{step}: the step that evaluation process proceeds from thr_{min} to thr_{max}

n: the number of the robust trade-off points

METRIC: the metric under consideration (Table 1)

score: array that contains the scores of the robust trade-off points

DP: array that contains the detection probability for the different detection thresholds

FAR: array that contains the false alarm rate for the different detection thresholds

DP_rob: array that contains the detection probability for the different detection thresholds that the algorithm is robust

FAR_rob: array that contains the false alarm rate for the different detection thresholds that the algorithm is robust

S: the final score assigned to the algorithm

INIT_EVAL: the function that evaluates the algorithm based on the detection threshold h and expectation a

Filter_Robust: the function that filters the robust trade-off points

Compute_Score: the function that computes the scores (e.q. 4)

Algorithm:

1:

2: **for** $j = a_{min}$ to $j = a_{max}$ with step = a_{step} **do**

3:

4: **for** $i = thr_{min}$ to $i = thr_{max}$ with step = thr_{step} **do**

5: $k = k+1$

6: $DP(k)$, FAR(k) = INIT_EVAL(i,j)

7: **end for**

8: DP_rob, FAR_rob = Filter_Robust(DP, FAR)

9:

10: **for** $i = 1$ to $i = n$ **do**

11: score(n) = Compute_Score(DP_rob(n), FAR_rob(n))

12: **end for**

13: $S(j)$= max(score)

14: **end for**

Author details

[1]Institute of Computer Science of the Foundation for Research and Technology-Hellas (FORTH), P.O. Box 1385, GR 71110 Heraklion, Crete, Greece

[2]Faculty of Engineering, University of Bristol, Queens Building University Walk, Clifton, Bristol BS8 1TR, UK

Competing interests

The authors declare that they have no competing interests.

Received: 14 September 2011 Accepted: 1 March 2012
Published: 1 March 2012

References

1. K Bicacki, B Tavli, Denial-of-Service attacks and countermeasures. IEEE 802.11 wireless networks Comput Stand Interfaces (Elsevier). **31**(5), 931–941 (2008)
2. D Thuente, B Newlin, M Acharya, Jamming vulnerabilities of IEEE 802.11e, in *Proc of MilCom 2007, IEEE*, Los Alamitos, pp. 1–7 (October 2007)
3. W Xu, K Ma, W Trappe, Y Zhang, Jamming sensor networks: attack and defense strategies. IEEE Netw. **20**(3), 41–47 (2006). doi:10.1109/MNET.2006.1637931
4. M Hall, A Silvennoinen, S Haggman, Effect of pulse jamming on IEEE 802.11 wireless LAN performance, in *Proc of MilCom 2005, IEEE*, Los Angeles, pp. 2301–2306 (October 2005)
5. A Cardenas, S Radosavac, J Baras, Evaluation of detection algorithms for mac layer misbehavior: theory and experiments. IEEE/ACM Trans Netw. **17**(2), 605–617 (2009)
6. A Fragkiadakis, V Siris, A Traganitis, Effective and robust detection of jamming attacks, in *Proc of the Future Network and Mobile Summit 2010*, Florence, Italy, pp. 1–8 (June 2010)
7. A Fragkiadakis, V Siris, N Petroulakis, Anomaly-based intrusion detection algorithms for wireless networks, in *Proc of the 8th International Conference on Wired/Wireless Internet Communications 2010*, Lulea, Sweden, pp. 192–203 (June 2010)
8. V Siris, F Papagalou, Application of anomaly detection algorithms for detecting syn flooding attacks. Comput Commun. **29**(9), 1433–1442 (2006)
9. J Cabrera, C Gutierez, R Mehra, Ensemble methods for anomaly detection and distributed intrusion detection in mobile ad-hoc networks. Inf Fusion (Elsevier). **9**(1), 96–119 (2008)
10. T Peng, C Leckie, K Ramamohanarao, Information sharing for distributed intrusion detection systems. J Netw Comput Appl (Elsevier). **30**(3), 877–899 (2007)
11. Y Chen, K Hwang, W Ku, Distributed change-point detection of ddos attacks: experimental results on deter testbed, in *Proc of USENIX Security Symposium*, Boston, USA, pp. 6–7 (August 2007)
12. M Cakiroglou, T Ozcerit, Jamming detection mechanisms for wireless sensor networks, in *Proc of 3rd Int Conference on Scalable Information Systems*, Napoli, Italy, pp. 1–8 (June 2008)
13. V Bhuse, A Gupta, Anomaly intrusion detection in wireless sensor networks. J High Speed Netw. **15**(1), 33–51 (2006)
14. A Sheth, C Doerr, D Grunwald, R Han, D Sicker, MOJO: a distributed physical layer anomaly detection system for 802.11 WLANs, in *ACM MobiSys*, Uppsala, Sweden, pp. 191–204 (19–22 June 2006)
15. W Xu, W Trappe, Y Zhang, T Wood, The feasibility of launching and detecting jamming attacks in wireless networks, in *Proc of ACM MobiHoc*, Urbana-Champaign, pp. 46–57 (May 2005)
16. M Thamilarasu, S Mishra, R Sridhar, A cross-layer approach to detect jamming attacks in wireless ad hoc networks, in *Proc of Milcom 2006*, Washington DC, USA, pp. 1–7 (October 2006)
17. K Pelechrinis, I Broustis, S Krishnamurthy, C Gkantsidis, Ares: an anti-jamming reinforcement system for 802.11 networks, in *Proc of CoNEXT 2009*, University of California, Riverside, pp. 181–192 (2009)
18. A Wood, J Stankovic, G Zhou, Deejam: defeating energy-efficient jamming in ieee 802.15.4-based wireless networks, in *Proc of the 4th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON)*, San Diego, California, USA, pp. 60–69 (2007)
19. I Krontiris, T Giannetsos, T Dimitriou, LIDeA: a distributed lightweight intrusion detection architecture for sensor networks, in *Proc of SecureComm 2008*, Istanbul, Turkey, pp. 1–10 (Sept 2008)
20. F Hugelschofer, P Smith, N Race, OpenLIDS: a lightweight intrusion detection system for wireless mesh networks, in *Proc of MobiCom 2009*, Beijing, China, pp. 309–320 (Sept 20–25 2009)

21. A Sampath, H Dai, H Zheng, B Zhao, Multi-channel jamming attacks using cognitive radios, in *Proc of ICCCN 2007*, University of California, Santa Barbara, pp. 352–357 (2007)
22. Linux wireless drivers, ath5k http://linuxwireless.org/en/users/Drivers/ath5k
23. Gnu radio homepage http://gnuradio.org/redmine/projects/gnuradio/wiki
24. Matlab homepage http://www.mathworks.com/
25. Iperf homepage http://iperf.sourceforge.net
26. M Nadgir, K Premkumar, A Kumar, J Kuri, Cusum based distributed detection in wsns, in *MCDES 2008*, Bangalore, India, pp. 134–140 (2008)
27. G Yan, Z Xiao, S Eidenbenz, Catching instant messaging worms with change-point detection techniques, in *Proc of the 1st Usenix Workshop on Large-Scale Exploits and Emergent Threats*, San Francisco, CA, pp. 1–10 (April 2008)
28. G Verdier, N Hilgert, J Vila, Adaptive threshold computation for cusum-type procedures in change detection and isolation problems. Elsevier Comput Stat Data Anal. **52**(9), 4161–4174 (2008). doi:10.1016/j.csda.2008.01.026
29. V Angelakis, S Papadakis, V Siris, A Traganitis, Channel interference in 802.11a is harmful. Testbed validation of a simple quantification model. IEEE Commun Mag. **49**, 160–166 (2011)
30. The gtk+ project http://www.gtk.org/
31. K Lu, D Wu, J Fan, Todorovic S, Nucci A, Robust and efficient detection of ddos attacks for large-scale internet. Comput Netw. **51**(18), 5036–5056 (2007). doi:10.1016/j.comnet.2007.08.008
32. Y Chen, K Hwang, WS Ku, Collaborative detection of ddos attacks over multiple network domains. IEEE Trans Parallel Distrib Syst. **18**(12), 1649–1662 (2007)
33. Y Sheng, K Tan, G Chen, D Kotz, A Campbell, Detecting 802.11 mac layer spoofing using received signal strength, in *Proc of INFOCOM'08*, Phoenix, AZ, pp. 1768–1776 (April 2008)

doi:10.1186/1687-1499-2012-73
Cite this article as: Fragkiadakis *et al.*: **Design and performance evaluation of a lightweight wireless early warning intrusion detection prototype.** *EURASIP Journal on Wireless Communications and Networking* 2012 **2012**:73.