Journal of Network and Systems Management (2022) 30:11
https://doi.org/10.1007/s10922-021-09616-6

Human Immune-Based Intrusion Detection and Prevention System for Fog Computing

Farouq Aliyu[1] · **Tarek Sheltami**[1] · **Mohamed Deriche**[2,3] · **Nidal Nasser**[4]

Received: 20 December 2020 / Revised: 4 July 2021 / Accepted: 12 July 2021 /
Published online: 13 October 2021

Abstract

The exponential increase in Internet of Things devices on the Internet causes a deluge of traffic at the cloud. Most of the traffic data is redundant. However, fog computing solves the problems by processing data at the network's edge. Lately, the fog layer is a target of cyberattacks, due to its resource constraints. In this paper, we proposed a lightweight, human immune, and anomaly-based intrusion detection system (IDS) for the fog layer. The proposed system achieves low resource overhead by distributing the IDS functions among the fog nodes and the cloud. We obtained an accuracy of up to 98.8%. Also, we recorded a 10% reduction in the energy consumption of the fog node when compared with deploying a neural network on the fog node.

Keywords Network security · Human-immune system · Intrusion detection and prevention system · Fog computing · Internet of Things · Cloud computing

✉ Farouq Aliyu
g201303650@kfupm.edu.sa

Tarek Sheltami
tarek@kfupm.edu.sa

Mohamed Deriche
mderiche@kfupm.edu.sa

Nidal Nasser
nnasser@alfaisal.edu

[1] Department of Computer Engineering, King Fahd University of Petroleum and Minerals, Dhahran 31261, Saudi Arabia

[2] Department of Electrical Engineering, King Fahd University of Petroleum and Minerals, Dhahran, Saudi Arabia

[3] Artificial Intelligence Research Center (AIRC), College of Eng. and IT, Ajman University, Ajman, UAE

[4] College of Engineering, Alfaisal University, Riyadh, Saudi Arabia

1 Introduction

Lately, there is an exponential rise in the deployment of IoT devices. By 2020, the number of devices connected to the internet may reach 50 billion [1]. Also, the volume and velocity of data received by the cloud have been increasing exponentially. Thus, leading to the degradation of the quality of services provided by the cloud. However, forty percent (40%) of IoT-generated data can be analyzed at the network's edge that is physically close to the IoT nodes [2]. This discovery led to the development of fog computing (FC).

FC (see Fig. 1) is a system where computing devices are placed physically close to the end-user devices (Things) to process data on behalf of the cloud. Thus, reducing latency. Any device with computing, storage, and network connectivity can be deemed a fog node/device [3]. The fog and the cloud layers complement each other. They provide interdependent and mutually beneficial services that make communication, computing, control, and storage possible throughout the system. Therefore, the security of the fog layer cannot be overemphasized. A compromised fog layer can lead to access of users' vital information or denial of service.

For example, Li et al. [4] demonstrated how a Man-in-the-Middle attacker could eavesdrop or compromise an IoT-fog network. The authors show how attackers can place themselves between the cloud and the fog nodes, thereby intercepting any information sent between them. Stojmenovic and Sheng [5] argued that since the fog nodes are physically close to the IoT layer, attackers can physically tamper with the fog nodes. More examples of attacks on FC systems have been extensively discussed in [6].

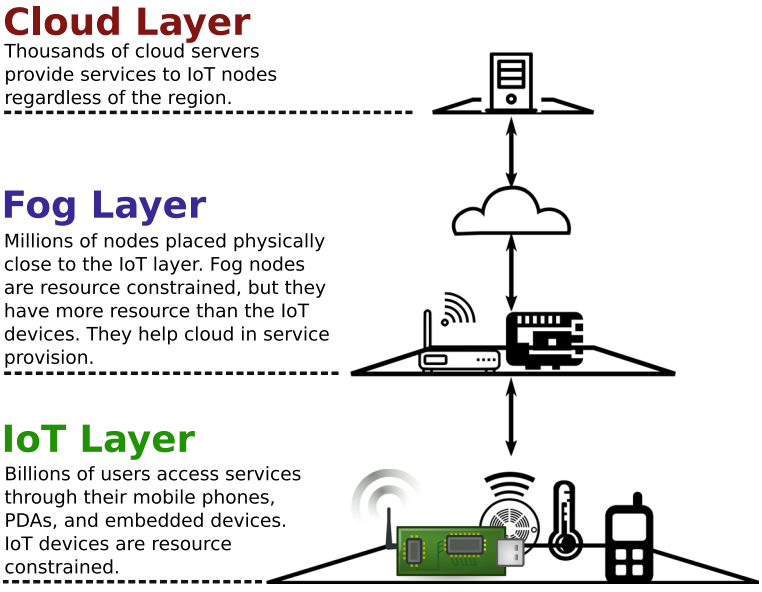


Fig. 1 Fog computing structure

Traditional security systems such as firewalls and anti-viruses have limitations: They only block attacks, but they cannot detect security breaches [7]. Hence, researchers propose intrusion detection systems (IDSs) or intrusion detection and prevention systems (IDPSs). An IDS is a reactive system that can detect behaviors attempting to compromise the confidentiality, integrity, or availability of a resource [8]. An IDPS (also known as Intrusion Prevention System (IPS) [9]) is a proactive system that identifies potential threats and tries to stop them before they occur [8]. An IDPS permits or blocks traffic base on a set of rules [7]. They observe and record events in the network, and they notify the network administrator [10].

In this paper, we propose a network-based, anomaly-based IDS for FC. The system mimics the human immune system. It models the white blood cells (Leukocytes) behavior, which allows it to provide security with minimum overhead. It also achieves lightweight by distributing detection roles to different components of the FC network. Thus, allowing the fog nodes to provide services to the IoT layer with little overhead. The proposed IDS finds application in static distributed FC. An example of such applications are network access points (AP) that serve as fog nodes [11]. The proposed system can also be used in time-critical industrial IoT applications [12]. The main contributions of this paper are as follows;

1. The paper proposes a modified human immune-based IDPS for FC networks using OMNET++.
2. The proposed IDPS mimics both granulocytes and monocytes in the leukocytes to improve the system's accuracy.
3. Also, the paper shows how distributing the functions of the different components of the leukocytes to the different components of an FC network reduces energy consumption and latency of the system.

The remaining parts of this paper are as follows: Sect. 2 presents a literature review of the IDS/IDPS in FC. Section 3 presents our proposed system. Section 4 discusses the experiments and simulations carried out, and the results obtained, while Sect. 5 concludes the paper.

2 Literature Review

Currently, FC is gaining interest among researchers. Fog nodes (FGNs) are physically closer to the IoT layer, making them more vulnerable than the cloud layer. In addition, the FGNs are resource-limited compared to the cloud. As such, it is impossible to apply traditional IDS techniques to them. Table 1 shows a summary of the latest IDS in FC discussed in this section.

Generally, IDS can be classified (base on detection) into two broad categories: anomaly—and signature-based. An anomaly-based IDS detects intrusion by detecting abnormal activities in the system, while signature-based IDS looks for tell-tale signs of a particular attack. Otoum and Nayak [13] proposed an IDS for IoT that combines both techniques. The system achieved a 96.9% detection rate when tested

Table 1 Summary of papers reviewed

References	System	Performance	Strength	Weakness
[13]	The system combines signature-based and anomaly-based IDS for IoT application	Detection rate = 96.9% Sensitivity = 96.6% Specificity = 96.8%	The system is able to achieve high accuracy	The system is resource demanding
[14]	The system cascades two RNN modules. An alarm is generated if any of the modules detects an attack	Accuracy = 92.0%	The system has low false-negative rate	The system has high false-positive rate. Also, the system is resource demanding
[15]	An ANN host-based IDS is developed using ANN-Cluster technique	Accuracy = 97.0%	The proposed system achieves high accuracy due to the use of ANN-C architecture	The system is resource demanding due to the architecture of ANN used and the large number of hardware listener processes
[16]	The authors used Gini Index technique to develop the tree	Accuracy = 96.7% Precision = 97% Recall = 97%	The system is lightweight and accurate	The system is prone to over-fitting and may not adapt to a wide range of attacks
[17]	The author proposed a danger theory host-based intrusion detection system using machine learning	–	Artificial Immune Systems are known for their flexibility	The author failed to capture the analytic power of the T-Cells
[18]	The authors proposed an IDS for industrial cloud storage using self-nonself theory	Detection rate = 99.97%	The proposed system shows high accuracy	However, the system require the storage of large amount of detectors for identifying attacks. This leads to overhead in memory and processing
[19]	The authors proposed a network-based Intrusion Detection System (IDS) by combining NSA and Dendritic Cell Algorithm (DCA) [20]	Accuracy = 77.1% Recall = 58.9% Precision = 98.7%	The proposed system is adaptive to changes in attack surface	The system is resource demanding due to the number of large number of detectors it generates

on the NSL-KDD dataset. Almiani et al. [14] proposed a multi-layered recurrent neural network for FC security. The system reached up to 92% accuracy on the NSL-KDD dataset. They achieved high accuracy because the system has two recurrent neural networks (RNN) cascaded: Whenever the first detects an event as benign, it passes it to the second for further test. The system only considers an event normal when both neural networks declare it harmless. Otherwise, it is an attack. The system is resource-demanding and has high false-positive rates due to the use of the two RNNs. Therefore, it is not ideal for an FC security system.

In another research, Pacheco et al. [15] achieved an accuracy of 97%. The authors proposed an artificial neural network (ANN) host-based IDS that monitors memory, CPU usage, and other hardware resources of the FGNs. Whenever there is a deviation from normal activities, the alarm goes off. This system has two limitations; 1) there are many listener processes for the system to monitor, which increases the energy and the latency overhead, and 2) The system observes the host processes and not the received packets. Therefore, the intrusion is only detected after the attack had begun.

To reduce the overhead associated with using traditional machine learning techniques, the authors in [16] proposed the use of the decision tree technique. The authors used the Gini Index technique to develop their tree. The authors tested the system using UNSW-NB 15 datasets and found that the system has an accuracy of 96.7%. Also, it is lightweight and accurate, but it is prone to over-fitting [21]. It is also unable to adapt to changes to input [22].

An alternative to ANN and decision tree techniques is the Human Immune-based Intrusion Detection System (HI-IDS). Several works have been carried out on HI-IDS security systems because of their adaptability [23]. An artificial immune system is a group of machine learning algorithms that mimic the human immune system. They work base on two distinct theories; self-nonself theory and danger theory [24]. The self-nonself theory consists of a family of algorithms that classify all activities into self (non-malicious) and non-self (malicious). The technique is analogous to the human body's way of differentiating cells into human cells (self) and antigens (non-self).

The danger theory models the signals our body generates in response to the unnatural death of its cells. Ou [17], proposed a danger theory host-based IDPS using machine learning. The system is modeled based on the dendritic cells (DC) and the T cells (TC). The DCs have three states; (1) immature state, which is their initial state, (2) semi-mature state where the system is safe, and (3) mature state where the DCs detect danger, then the body activates the TCs to attack the antigens. Unfortunately, the author failed to capture the analytic power of the TCs. Also, they ignored the memory T-cells, which allow the TCs to attack antigens without alerts from the DCs.

Wang et al. [18], proposed an IDS for industrial cloud storage using the dynamic artificial immune algorithm. The system is base on the self-nonself theory. The authors used an improved version of the Negative Selection Algorithm (NSA) [25]. They mimic the immune system's strategy of filtering leukocytes that are autoimmune. Rather than removing the self detectors, they mutate them into non-self. The authors obtain a detection rate of 99.97%, when tested on the KDD 99 dataset. The

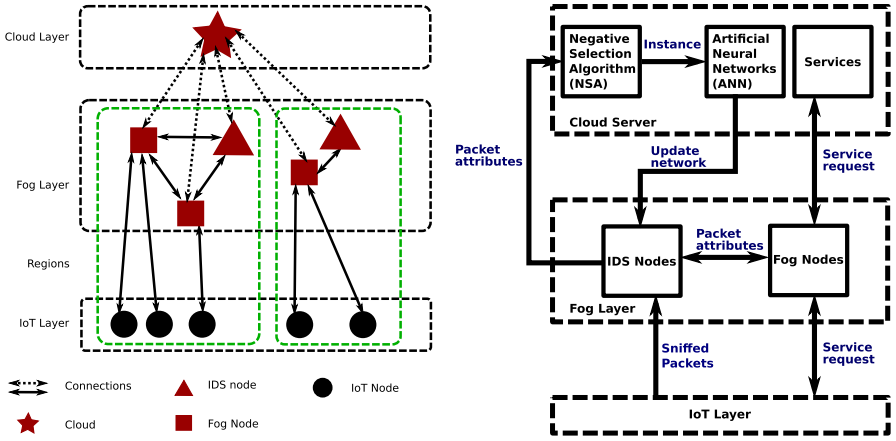
result shows that the immune system can provide efficient intrusion detection in an industrial network. However, NSA generates numerous detectors for future detections. Hence, increasing the memory requirement of the system, which disqualifies it for use in FC.

Hosseinpour et al. [26, 27] proposed a lightweight network-based IDS for an IoT system using Artificial Immune System (AIS). The network consists of IoT devices, edge nodes, FGNs, and the cloud server. The authors point out the large overhead associated with deploying AIS-based meta-heuristics like NSA and DCA. Therefore, the authors proposed a system where; (1) The cloud is responsible for creating and training the non-self detectors using unsupervised clustering techniques (NSA in this case). (2) The fog layer analyzes the intrusion alerts received from the edge nodes and decides whether they false alarms by counting the number of detectors detecting the event. When an attack is properly detected, the FGN creates a memory cell detector capable of identifying the attack in the future. Then, the FGN distributes the memory cells to the edge nodes. (3) The detectors created by the cloud are deployed at the edge layer, where the network traffic is sensed, and attacks are detected. The authors tested the system on KDD 99 dataset, and they obtained an accuracy of 77%. However, the fog and the edge nodes perform the same function; the detectors in the fog layer are duplicated and sent to the edge. This leads to the duplication of memory cell detectors saved in the edge nodes, thus wasting the precious memory. Furthermore, the limited-resourced edge node cannot keep up with the ever-increasing number of memory cell detectors generated in a high traffic network.

Our paper proposes a HI-IDPS similar to that in [26, 27]. However, we consider each component of the network analogous to a component of the white blood cells: A special type of nodes in the fog layer called "IDS nodes" (IDSN) behave like the lymphocytes. They store detectors for future detections. The cloud uses NSA to develop the detectors in the form of a neural network. The use of ANN ensures low memory overhead regardless of the number of FGNs, the frequency –, and the variety of attacks. Finally, the FGNs behave like monocytes. They carry out the innate defense using statistical analysis on the packets before servicing them. Statistical analysis is used because of its low latency and energy overheads. The system achieves lightweight by distributing IDS activities to different parts of the networks. Moreover, the system achieves higher accuracy because each network component carries out detection using a different technique.

3 Proposed System

The proposed system aims at detecting intrusions in the fog layer of a network. The fog layer consists of FGNs and Intrusion Detection System Nodes (IDSN) as shown in Fig. 2. The novelty of the proposed system is its ability to combine innate and adaptive immune systems in one IDS. On the innate immune system side, are the FGNs. In addition to providing services to the IoT layer, the FGNs use statistical techniques to detect malicious packets. Just like the monocytes, they then forward the suspicious packets' attributes to the IDSNs for verification.



(a) The Network Topology of the Proposed System

(b) The proposed IDPS Structure

Fig. 2 The proposed network structure of the IDPS

On the adaptive immune system part of the IDS, are the IDSNs. The IDSNs use a pre-trained neural network classifier they are downloaded from the cloud to detect attacks. When an IDSN detects an attack, it first notifies the neighboring FGNs, who in turn blacklist the culprit IoT node. Then, it notifies the cloud/server. The cloud/server uses the Negative Selection Algorithm (NSA) to verify the event. It then uses the information to update the neural network used by the IDSNs. This allows the IDSN to dynamically learn about new threats, which is analogous to the behavior of a T-cell. Sections 3.1 and 3.2 describe the functions of the different components of the proposed system in details, while Sect. 3.3 describes the attacker model.

3.1 Fog Nodes

The sole function of the FGNs is to provide services to the IoT nodes. Therefore, the IDS added to the FGNs only carry out an innate immune system function, such that it does not affect the primary function of the FGNs. They observe the incoming traffic from the network using statistical techniques. Statistical techniques were used because they consume less memory and computation power [28]. However, the cost of using this low resource-demanding technique is lower accuracy [28]. Hence, the FGNs send a detection alert to the nearest IDSN for further analysis.

Figure 3 shows the activity diagram describing the IDS mechanism for the FGNs. When an FGN receives a packet from the IoT layer, it first checks it against a blacklist. Nodes found in the blacklist are blocked, their packets are discarded. This forms part of the IPS that mitigates attacks at the fog layer.

However, packets that are not from the blacklisted nodes are tested using statistical techniques to determine whether they are malicious. Those that are found to be non-malicious are serviced, and the results are replied to the respective senders. But,

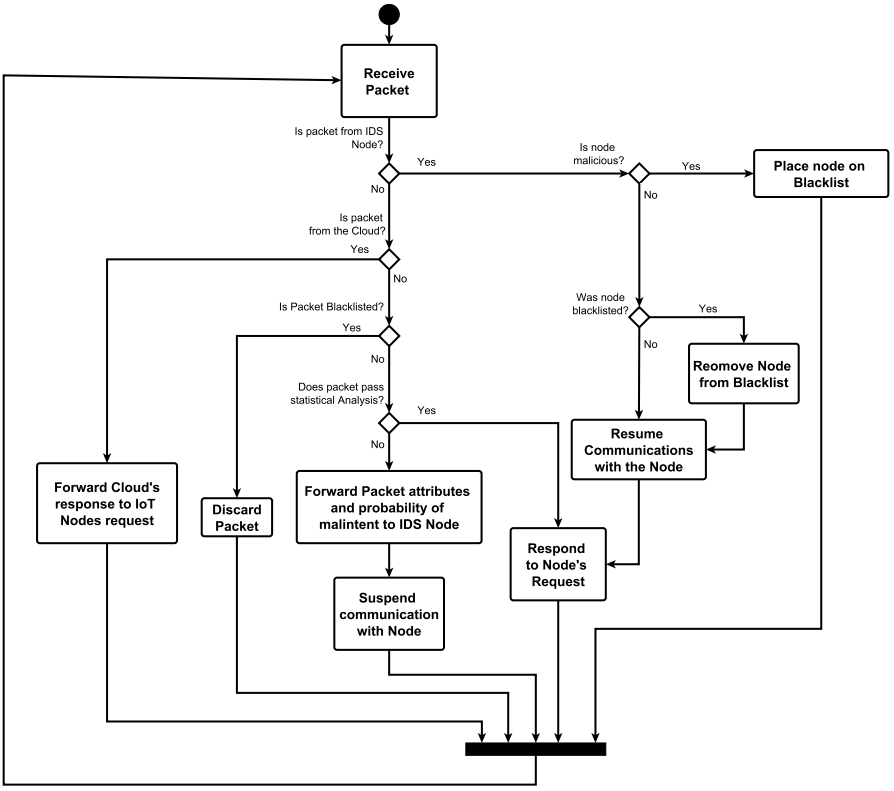


Fig. 3 Activity diagram for the fog nodes

malicious packets are forwarded to the IDSN for further analysis. Also, the FGN suspends communication with the node until the IDSN vindicates it. In addition, the IDSN periodically sniffs packets and tests them. If the IDSN sniffs a malicious packet, it notifies its neighbors so that they can blacklist the culprit. However, if the node is benign, the sniffed packets are discarded.

3.2 Intrusion Detection System (IDS) Nodes

The IDSNs simulate the lymphocytes (i.e., the T Cells, B Cells, and NK Cells). This is achieved with the help of neural networks, which the IDSN downloads from the cloud as an update. One might wonder why a neural network classifier? Neural network is chosen because of its accuracy [15]. In our analogy with HIS, the neurons in the ANN emulates receptors of the lymphocytes. The neural network allows the cloud to compress information about the attack surface. Thus, reducing memory overhead compared to detectors used in [19]. Another merit of this technique is that the cloud obtains information about possible attacks from all IDSNs in the network. As such, the generated neural network has a global knowledge of the attack space.

In addition, the IDSNs are responsible for intrusion prevention by blocking malicious nodes from the network; when an FGN detects anomalous behavior in the network, it informs the nearest IDSN. The IDSN checks the event. If the behavior is malicious, the node is blacklisted. Also, other FGNs in the network are informed to blacklist the node. Otherwise, the FGN that sends the request is informed that the node is non-malicious.

A summary of the IDSNs' activity is shown in Fig. 4. The IDSNs sample the packets in the network. The sniffed packets are tested; if an IDSN detects a malicious packet, the IDSN sends it to the cloud for further verification. If the cloud affirms that the node is malicious, the IDSN blacklists it. Also, the IDSN notifies all neighboring FGNs so that they blacklist the node. However, if the node is not malicious, the IDSN removes it from its blacklist. Then, the IDSN notifies the FGNs to remove the node from their blacklists.

3.3 Attacker Model

Figure 5 shows the attacker model considered in the development of the proposed system. The following are the characteristics of the attackers:

– The attackers resides in the IoT layer and the attack(s) arrive randomly with an inter-arrival time of following an exponential distribution.

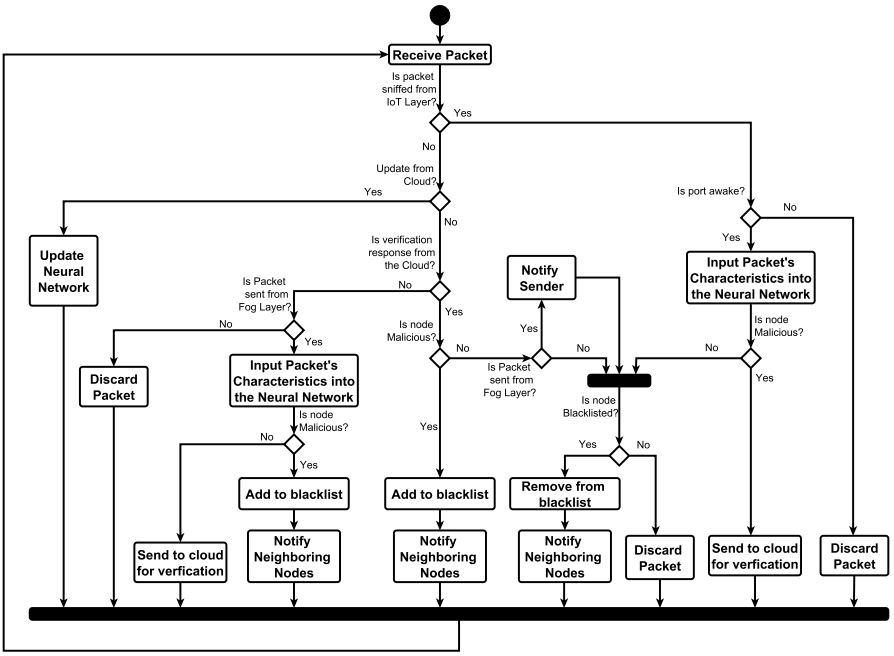


Fig. 4 Activity diagram for the IDS nodes

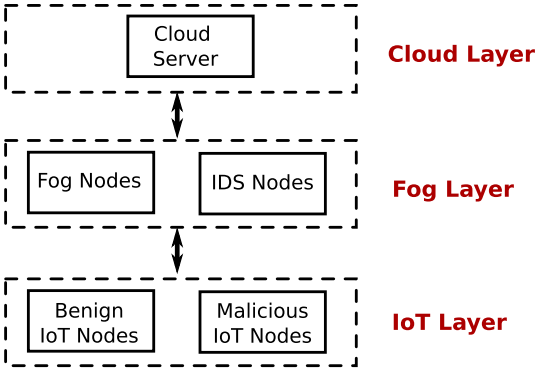


Fig. 5 Block diagram of the attacker model

- The attacker(s) possess large amount of resources, which are more than those possessed by both the FGNs and the IDSNs. However, their resources are less than that of the cloud.
- The attacker(s) can communicate with the FGNs. But they cannot access the IDSNs, because the IDSNs only communicate with the FGNs.
- Also, the attacker(s) and other IoT nodes in the IoT layer cannot communicate directly with the cloud. Rather, all requests are serviced through the FGN

4 Discussion of Results

In this paper, we assume the FGNs are made from Single Board Computers (SBCs), which is in line with the latest trend [29, 30]. As such, we carried out experiments to obtain their transmission and computational characteristics. The data obtained is then used in simulating the proposed system. For this reason, this section is broken into three subsections; Sects. 4.1 and 4.2 discuss the experiments and the simulations respectively, while Sect. 4.3 compares the performance of the proposed system with that of similar systems in the literature.

4.1 Experiment

Here, we obtain the characteristics of the SBCs to enable us to simulate the IoT layer, the Fog layer, and the Cloud layer. The IoT layer is required for the simulation because attacks and service requests come from there. The cloud layer helps the FGNs with services they cannot provide. It also validates the possibility of an intrusion for the IDSNs.

For IoT layer, Orange Pi (OPi) Lite was used [31]. We chose an OPi because its hardware is similar to that of an average smartphone, which makes it a good candidate to simulate the characteristics of the IoT layer. The OPi is an open-source SBC, which can run on either Android 4.4, Ubuntu, Armbian, or Debian Image. It uses the Allwinner H3 1.2 GHz quad-core Cortex A7 SoC as a processor and 512 MB DDR3 SDRAM main memory.

For the fog layer, the Raspberry Pi (RPi) model 3 B+ was used [32]. We chose the RPi because it has more computing power than the OPi. Like OPi, RPi is an SBC that runs on the Linux kernel. The processor is a 64-bit quad-core ARM Cortex-A53 CPU and 1 GB LPDDR2 SDRAM main memory.

The experiments aim to investigate the performance of the devices that we can use in the different layers of an FC system. The results obtained enable us to accurately simulate the proposed system in Sect. 4.2. In the experiments, we investigated the devices'; energy consumption, latency and communication bandwidth, delay due to extracting data from packets, and delay of using ANN for intrusion detection. Figure 6 shows the setup of the proposed experiment.

As mentioned earlier, we investigated the energy and network performance of OPi and RPi. Then the results are used to model the IoT- and the fog layer. The performance of the RPi was investigated as follows; the power consumption of the RPi during booting and when the wireless radio is turned on were investigated. This is carried out by connecting the RPi in series with a (1Ω) shunt resistor. The resistor was connected along the negative line of the RPi's power supply. This allows us to safely connect a voltmeter across the shunt resistor. In the case of our experiment, an Arduino Nano's analog to digital converter (ADC) was used as a voltmeter. Equation 3 is used in calculating the power consumption of the RPi. See Table 2 for definition of terms. The current (I) passing through the shunt resistor (R) is the same current passing through the RPi since the two are connected in series. Therefore, I can be obtained by dividing the voltage measured with the Arduino Nano (V_{ADC}) by R. Also, the voltage across the RPi (V_{RPi}) is given by Eq. 2.

$$I = \frac{V_{ADC}}{R} \tag{1}$$

$$V_{RPi} = Vcc - V_{ADC} \tag{2}$$

$$P = I \times V_{RPi} \tag{3}$$

Figure 7 shows the power consumption of the RPi, which was used to model both FGNs and IDSNs. The power consumption was sampled every 500 ms. Figure 7a

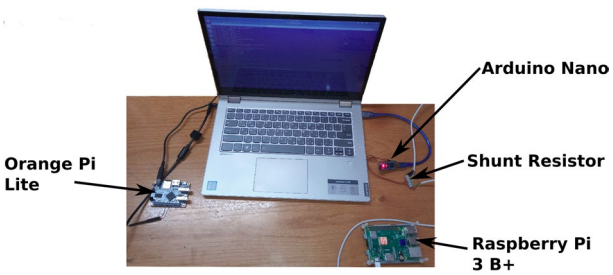
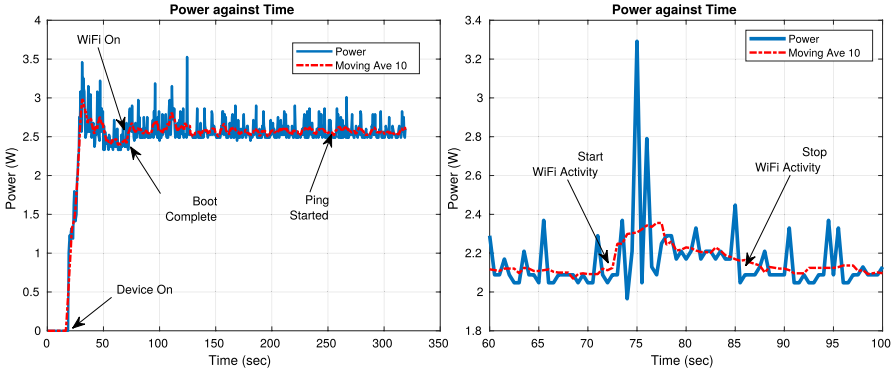


Fig. 6 Experiment setup for the IoT and fog characteristics

Table 2 Table of Notations

Notation	Description
FC	Fog computing
FGN	Fog node
IDSN	IDS node
HIS	Human immune system
R	Resistance of shunt resistor
I	Current passing a shunt resistor as well as the load
V_{ADC}	Voltage measured by the voltmeter (using Arduino ADC)
V_{RPi}	Voltage across the raspberry Pi
λ_{fog}	Time it takes the FGNs to reply to a request from the IoT layer
$\lambda_{i\_f\_comm}$	Latency due to communication between IoT and fog layer
λ_{f_proc}	Latency due to processing request from the IoT layer
α	Hit/Miss factor
λ_{f_proc}	Latency of Fog node during data processing
λ_{ANN}	Time taken by ANN to convert an input to an output
$\lambda_{extract}$	Time taken to extract information from a packet
$\lambda_{service}$	Time taken by the proposed system to service a request from IoT layer
TP	True positive, when an attack event is correctly detected
TN	True negative, when a benign event is correctly detected
FP	False positives; when benign event is detected as malicious
FN	False negative; when malicious event is detected as benign
E_{fog}	Total energy consumption of FGN
$E_{service}$	Energy consumed by FGN when servicing request from IoT
$E_{detection}$	Energy consumed by FGN during detection
E_{IDS}	Total energy consumption of IDSN
$E_{sniffed}$	Energy consumed by IDS when sniffing packets
$E_{fog-IDS}$	Energy consumed due to communication between FGN and IDSN
$E_{cloud-IDS}$	Energy consumed due to communication between cloud and IDSN



(a) Power-Time Graph of Fog - and IDSN **(b)** Power-Time Graph of Onboard WiFi

Fig. 7 Power-time graph of raspberry Pi during different operation

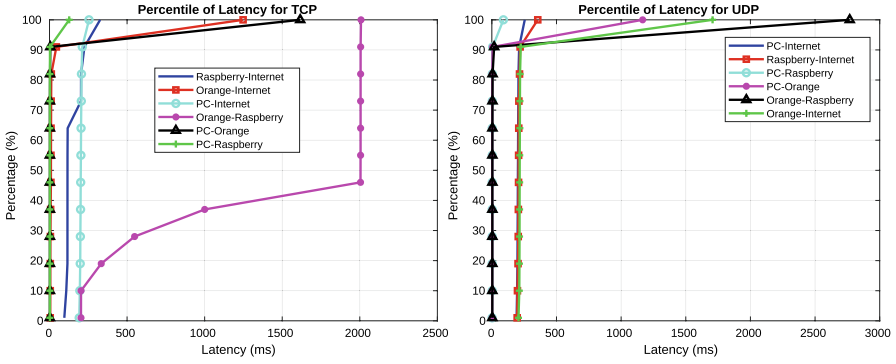
shows the power consumption of the RPi from the time it was switched on until 325 s later. It was found that the power consumption of the RPi peaks at 3.5 W during booting. Afterward, it fell to 2.4 W when the node was idle. However, it should be noted that a display screen was connected to the RPi, which consumes 0.7 W [33]. After the operating system has finished booting, the Wi-Fi was turned on and the power consumption peaks at 3.6 W, but on average the power consumption is just above 2.5 W. The PC was pinged from the RPi, but no much difference in power consumption was observed. The reason is that when the Wi-Fi is listening for data, it is consuming nearly the same amount of power as when it is transmitting data. To closely observe the power consumption of the Wi-Fi, Fig. 7b was obtained. The figure shows a power consumption of 0.2 W on average and a maximum of 1.3 W.

$$\lambda_{fog} = \lambda_{i\_f\_comm} + \alpha\lambda_{f\_proc} + (1-\alpha)(\lambda_{c\_proc} + \lambda_{f\_c\_comm}) \tag{4}$$

$$\alpha = \begin{cases} 1 & \text{if fog hit} \\ 0 & \text{if fog miss} \end{cases} \tag{5}$$

Perhaps the most important parameter in FC is the latency (λ_{fog}). The latency of an FC system is shown in Eq. 4. It consists of; latency due to communication between IoT and fog layer ($\lambda_{i\_f\_comm}$) and latency due to processing requests from the IoT layer (λ_{f_proc}). However, if the FGN cannot process the request, a miss has occurred. Then the request is forwarded to the cloud. In such cases, the λ_{fog} becomes the sum of; $\lambda_{i\_f\_comm}$, the latency of processing the request in the cloud (λ_{c_proc}), and the latency of communication between the cloud and the fog layer ($\lambda_{f\_c\_comm}$).

We used network performance tools to obtain the communication latency between an FGN and an IoT node or a cloud. Matt's Traceroute (MTR) was used to measure the latency of the link between the IoT layer and the fog layer, and between the fog layer and the cloud layer. Figure 8 and Tables 3 and 4 show the experiment results for User Datagram Protocol (UDP) and Transmission Control Protocol (TCP) latency for the links. Figure 8 shows the cumulative frequency curves of the



(a) Latency of TCP Communication **(b)** Latency of UDP Communication

Fig. 8 Cumulative frequency curve for latency of different links in an FC network

Table 3 Latency of TCP communication between PC, internet, Fog- and IDSN's in a LAN

Configuration	Mean (ms)	SD (ms)	Min (ms)	Max (ms)
PC—Internet	202.94	9.09	191.48	254.53
Raspberry Pi—Internet	148.94	56.2	95.44	326.3
Orange Pi—Internet	49.41	183.62	4.6	1247.86
PC—Raspberry Pi	5.41	17.6	2.33	126.94
PC—Orange Pi	81.66	308.17	2.22	1615.85
Orange Pi—Raspberry Pi	1348.84	774.91	202.91	2007.88

Table 4 Latency of UDP communication between PC, internet, fog- and IDSN's in a LAN

Configuration	Mean (ms)	SD (ms)	Min (ms)	Max (ms)
PC—Internet	202.94	9.09	191.48	254.53
Raspberry Pi—Internet	216.81	36.75	195.89	355.23
Orange Pi—Internet	270.95	262.22	206.16	1707.34
PC—Raspberry Pi	7.8	17.43	2.2	89.42
PC—Orange Pi	29.84	165.62	2.23	1166.41
Orange Pi—Raspberry Pi	95.34	430.97	3.46	2768.09

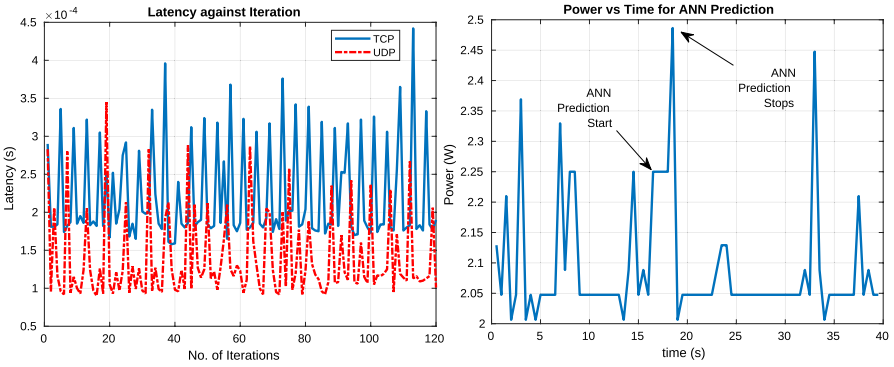
communications, which allows us to investigate whether the data is skewed. This is necessary because the latency of networks is known to be widely skewed, especially for Internet-based networks [34]. From Fig. 8a, it can be seen that up to 90% of the sample data have a little deviation from the mean. However, links involving the OPi (see the highlighted rows) show more deviations due to lower resources, especially memory. Table 3 shows a summary of latency for TCP. Therefore, in our simulation, we shall use the mean for links with low standard deviation, such as; RPi-Internet (i.e. fog-cloud), OPi-Internet (i.e. IoT-cloud), and RPi-RPi (i.e. fog-fog) links. However, the maximum latency will be used to describe the TCP latency of the OPi-RPi (i.e. IoT-fog) link.

Also, similar latency experiments were carried out for the UDP. This will allow users to simulate UDP-based FC systems. Figure 8b shows the cumulative frequency for a sample size of 50. Table 4 shows that the communications involving OPi (see the highlighted rows) have a standard deviation greater than 150. This is attributed to the fact that OPi is resource-constrained. Therefore, all links that are connected to the OPi must be modeled using the maximum latency values, while other communications can be modeled with the mean value.

Next, the bandwidth of OPi and RPi were investigated. The bandwidth of the nodes is important because it helps in calculating the energy consumed during the transmission. The network performance tool iperf was used to measure the bandwidth. We investigated the bandwidth between the PC and the OPi, and between the PC and RPi. The tests were carried out between the SBCs and the PC because the PC has abundant resources. Thus, the maximum performance of each SBC is

Table 5 Bandwidth for transmission

	Orange Pi			Raspberry Pi		
	TCP	UDP		TCP	UDP	
	BW (Mbps)	BW (Mbps)	Jitter (ms)	BW (Mbps)	BW (Mbps)	Jitter (ms)
Mean	46.840	1.050	0.150	49.370	1.050	3.610
Std	4.270	0.000	0.060	5.130	0.010	6.690
Min	35.900	1.050	0.085	39.400	1.030	0.195
Max	49.400	1.050	0.279	54.000	1.050	18.161



(a) Latency of Extracting Data from Packet **(b)** Power Consumption for ANN Prediction

Fig. 9 Performance of raspberry Pi on the basic operations of both fog- and IDSNs

obtained. A sample size of 50 was used. As can be seen from the highlighted numbers in Table 5, the standard deviation of the data is small (less than 10). Therefore, the mean values can be used.

$$\lambda_{f\_proc} = \lambda_{extract} + \lambda_{ANN} + \lambda_{service} \tag{6}$$

It is known from Eq. 4 that in some cases, the latency of the fog layer depends on the data processing latency of the FGNs (λ_{f_proc}). Therefore, an experiment was carried out to measure the latency (λ_{ANN}) of using the artificial neural network (ANN) by the IDSN to detect intrusion as discussed in Sect. 3.2. As shown by Eq. 6, the latency of detecting an intrusion is the sum of the latency of extracting information from a packet ($\lambda_{extract}$), the latency of using that information to detect intrusion using ANN (λ_{ANN}), and the latency to service the request for the IoT layer ($\lambda_{service}$).

OpenNN [35] was used to model and train a single layer Multi-layer Perceptron (MLP) for the IDSN. The MLP used for this experiment has 100 neurons in the hidden layer, 121 input neurons, and 23 output neurons. OpenNN was chosen because it is flexible and lightweight. Figure 9 shows the delay and energy consumption of applying ANN intrusion detection on the RPi using KDD99 dataset. For the NSL-KDD dataset, 126 input neurons, 3 hidden layers (504, 252, and 126 neurons), and

2 output neurons were used on the Scikit-learn library [36]. We experimented with this database, to compare the accuracy of the proposed system with some selected papers that carried out similar work in the literature (see Sect. 4.3).

We developed C-programming language-based packet sniffing code to obtain the time (i.e., $\lambda_{extract}$) it takes the IDSN to extract information from a packet. The program measures $\lambda_{extract}$ for each received packet. Then store it in an external file. The program is a modification of [37]. Figure 9a shows the distribution of the data obtained for both the TCP and UDP. A summary of the experiment is shown in Table 6. The table shows that the standard deviation is low compared to the mean. Therefore, the mean value can be used in the simulation.

Lastly, we investigated the latency of ANN classification (λ_{ANN}). The λ_{ANN} is the time taken from when an input is given to the ANN to when an output classifying the packet as malicious or benign is obtained. Table 6 shows λ_{ANN} for both KDD99 and NSL-KDD on RPi. The results were obtained from a sample size of 120. The samples were uniformly randomly selected from the dataset. This allows us to obtain a statistically accurate λ_{ANN}. The ANN code was manually executed 120 times to avoid experimental errors. As highlighted in the "Std" column of Table 6, the sample has a small standard deviation compared to the mean. Therefore, the mean of the samples can be used in the simulation. Also, the power consumption for running the ANN using a loop with 1000 iteration is shown in Fig. 9b. Numerous iterations were used because the ANN classification has a latency of 0.94 ms $\leq \lambda_{ANN} \leq$ 1.08 ms. This makes it difficult for the Arduino Nano to monitor a single execution. The power-time graph in Fig. 9b shows that the power consumption due to ANN classification ranges from 2.25 to 2.45 W. To get the worst case of ANN impact on the system, 2.45 W was chosen.

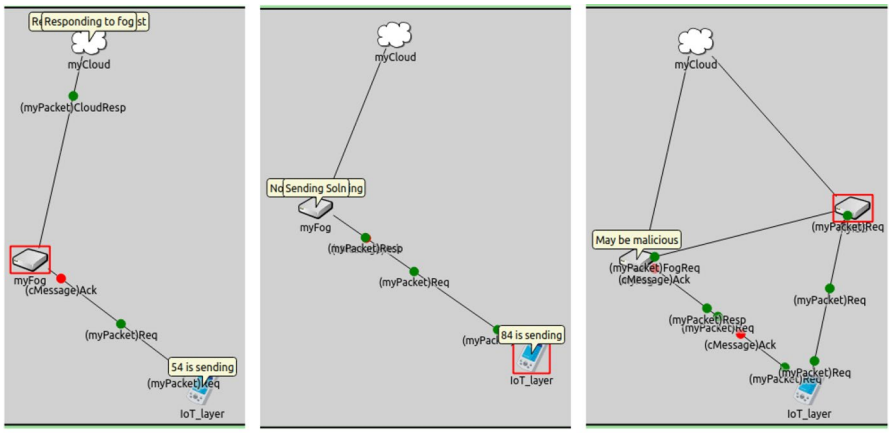
4.2 Simulations

OMNET++ was used to simulate the proposed system. OMNET++ is a C++ based simulator that allows for modular development of network simulations [38]. The simulator allowed us to investigate different aspects of the proposed system and allowed for easy modification where necessary.

In this paper, four (4) simulation scenarios were investigated as shown in Table 7. Also, Fig. 10 shows the different network scenarios simulated. The simulations were aimed at investigating the performance of the FC system.

Table 6 Latency of fog- and IDSN when extracting data from packet

Protocol	Min (ms)	Max (ms)	Mean (ms)	Std (ms)
UDP	0.09	0.34	0.14	0.06
TCP	0.16	0.44	0.22	0.07
ANN classification (KDD99)	0.94	1.08	0.96	0.02
ANN classification (NSL-KDD)	1.57	4.65	2.21	0.28



(a) No Security **(b)** ANN Secured **(c)** Proposed System

Fig. 10 Comparison of different security setups for the FC system

To get accurate simulations, the results obtained from the experiments in Sect. 4.1 were used as simulation parameters. However, we obtained the accuracy of the negative selection algorithm (NSA), ANN, and statistical techniques from the literature [18, 39, 40] respectively. Tables 8 and 9 shows the parameters used in the simulation of the proposed system. The first column contains the components of the network, which includes the channels connecting the different parts of the network. The second column is the variable names as used in the simulation. This will come in handy for those studying our simulation code. Our simulation code is available on GitHub [41]. The third column shows the assigned values used in the simulation. These values were used in all experiments carried out in this paper unless mentioned otherwise. Finally, the fourth column gives a brief description/comment on the variables and how the values were obtained.

The simulations were carried out to investigate the performance of the proposed system. Three main performance metrics were investigated, these are; accuracy, energy consumption, and latency. Figure 11 shows the accuracy of the different components of the proposed system, as well as the accuracy of the system as a whole. Figure 11a, b show the results obtained when the KDD99 and the NSL-KDD datasets were used respectively. The results were recorded with a 95% confidence interval (CI). We found that the accuracy of the proposed system Fog_IDS_29 is 93.96%, which is close to the IDSN accuracy of 95.0%. This is akin to using ANN directly on the FGN for anomaly detection without deploying the IDSN. As such, experiments were carried out to investigate the overhead of using only ANN on the FGN later in this section. When the duty cycle was increased to 67%, the accuracy increased to $\approx$ 96.71%. However, this will increase the energy consumption of the IDSN since its wake-up period is longer. Moreover, there are overlaps in the CI for Fog_Naive_10, Fog_Naive_26, and Fog_Naive_67, in both datasets. It shows that there is no significant difference in the setup. As such, we take Fog_Naive_10 and Fog_Naive_26, for further analysis.

Table 7 Simulation scenarios

SN	Scenario	Topology	Variations	Comment
1	Control Experiment	See Fig. 10a	Fog_Baseline	An FC system without any IDS
2	Fog + ANN	See Fig. 10b	Fog_ANN	Fog computing system with an ANN classifier installed in the FGN
3	Fog +IDS	See Fig. 10c	Fog_IDS_29	Proposed system where the IDSN has a wake-up to sleep ratio of 2:5 (i.e. 29% Duty cycle)
			Fog_IDS_67	Proposed system where the IDSN has a wake-up to sleep ratio of 2:1 (i.e. 67% Duty cycle)
4	Naive Fog +IDS	See Fig. 10c	Fog_Naive_10	Proposed system with no IDS the FGN has no security and the IDSN has a wake-up to sleep ratio of 1:9(i.e. 10% Duty cycle)
			Fog_Naive_29	Proposed system with no IDS on the FGN has no security and the IDSN has a wake-up to sleep ratio of 2:5 (i.e. 29% Duty cycle)
			Fog_Naive_67	Proposed system with no IDS on the FGN and the IDSN has a wake-up to sleep ratio of 2:1 (i.e. 67% Duty cycle)

Table 8 Simulation parameters continued

Component	Variable	Value	Comment
Cloud Layer	cloud_accuracy	0.98	Detection accuracy of the cloud server. Obtained from [18]
	cloud_service_time	0.01 s	The latency of service provision. It depends on application
	cloud_security_time	0.0013 s	The latency due to IDS detection. Obtained from experiment
Channel	ch_IoT_fog	2007.88 ms	Channel latency. Obtained from experiment
	ch_fog_cloud	150.09 ms	Channel latency. Obtained from experiment
	ch_fog_ids	89.73 ms	Channel latency. Obtained from experiment

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{7}$$

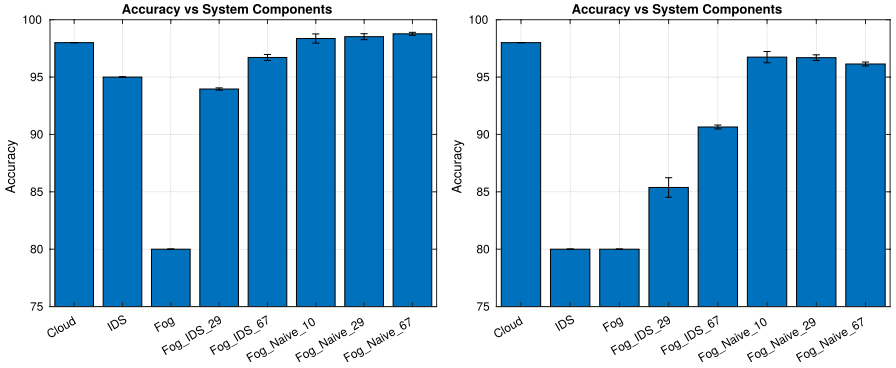
Surprisingly, the accuracy of the proposed system improved when the FGN does not carry out any intrusion detection. As shown in Fig. 11a, b, the accuracy of the proposed system, Fog_Naive_29 and Fog_Naive_10, improved to $\approx 98.77\%$ on KDD99 and $\approx 96.74\%$ on NSL-KDD respectively. Thus, there is a need for further investigation. We carried out a diagnostic test on the systems using the KDD99 dataset. We choose one dataset since both systems show a similar trend. The diagnostic test parameter investigated are; True Negative (TN), True Positive (TP), False Negative (FN), and False Positive (FP) (see Table 2). As shown by Eq. 7, the diagnostic test parameters gave us an insight into why we obtain the accuracy values aforementioned.

Figure 12 shows a diagnostic test of Fog_IDS_29, Fog_IDS_67, Fog_Naive_10 and Fog_Naive_29 as Fig. 12a–c respectively. In each figure, the performance of the cloud, the FGN, the IDSN, and the whole system investigated. The IDSN receives packets from the FGN for double-checking, and it also sniffs packets independently to test them. Hence, we used RX_IDS and Sniffed_IDS to show the IDSN's performances when it receives packets from the FGN and sniffed ones. Furthermore, the y-axis is on a logarithmic scale to make the variation in the values more visible. It is vivid that the values from Fig. 12a, b show the same pattern. But the IDSN sniffed more packets in Fog_IDS_67. Therefore, the Fog_IDS_67 has slightly larger values because it stays up longer than the Fog_IDS_29.

Similar behavior is observed between Fig. 12c, d. Also, there is no reading for the FGN (labeled"Fog" in the charts) and packets received by the IDSN (labeled "RX_IDS"). It is because the FGN neither tests the packets nor forwards them to the IDSN for testing. We learn from the performance of the system in Fig. 12 that the improvement in accuracy seen in Fig. 11 is due to a sharp fall in False Positive cases after removing the statistics-based IDS from the FGN, which is known for high false alarm rates [40]. It also leads to a reduction in the false-positive values at the cloud and the system in general.

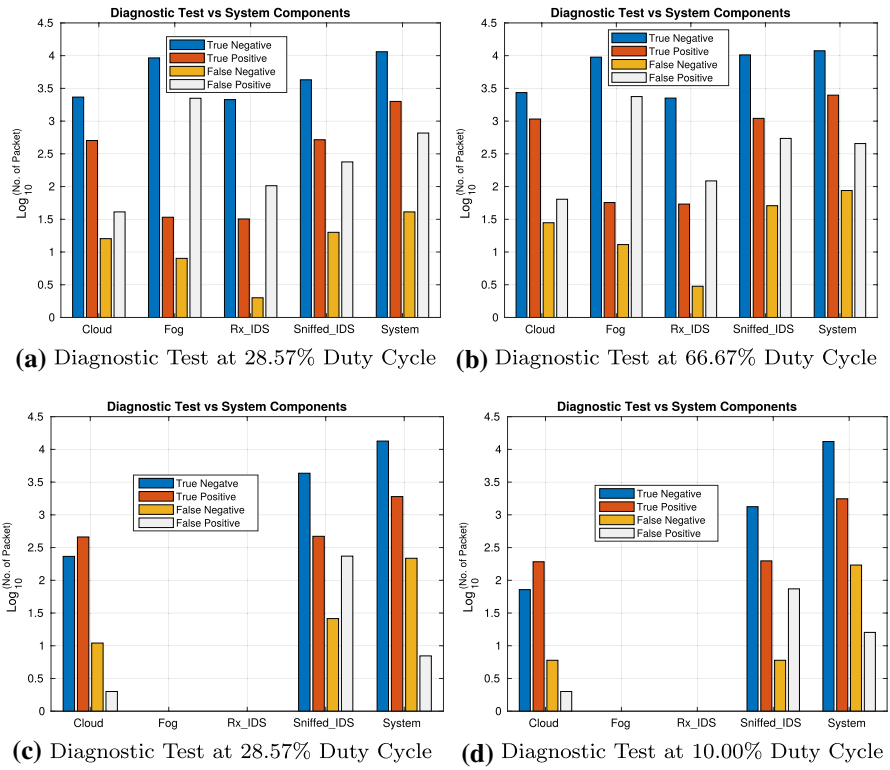
Table 9 Simulation parameters

Component	Variable	Value	Comment
IoT layer	arrival_rate	1.0 s	The mean arrival rate of service requests. This variable is application specific
	users	100	The number of nodes in the IoT layer
	attackers	0.1	The percentage of nodes that are possible attackers
IDS layer	ids_accuracy	0.95	The combination of true positive and true negative. Obtained from [39]
	ids_security_time	0.0013 s	The latency due to IDS detection. Obtained from experiment
	T_sleep	5 s	The time (sec) until node starts sniffing again. It depends on application
	T_wake	2 s	Duration (sec) for sniffing time. It depends on application
	handover_time	0.0002 s	Duration (sec) to process and handover packet. It depends on application
	ids_wifi_power	2.3 W	The average transmission energy in watts. Obtained from experiment
	ids_idle_power	2.1 W	The average idle power consumption in watts. Obtained from experiment
	ids_bw	49.370 Mbps	TCP bandwidth (Mbps) of the fog node. Obtained from experiment
Fog layer	fog_accuracy	0.8	The combination of true positive and true negative. Obtained from [40]
	service_time	0.01 s	The time until node responds with solution to request. It depends on application
	fog_security_time	0.0013 s	The latency due to IDS detection. Obtained from experiment
	fog_security_power	2.45 W	Power consumption due to IDS detection. Obtained from experiment
	fog_hit	0.7	The fog nodes hit rate. It depends on application
	cloud_2_iot	0.02 s	The time needed by fog to handover packets from cloud to IoT. It depends on application
	fog_wifi_power	2.3 W	The average transmission energy in watts. Obtained from experiment
	fog_idle_power	2.1 W	The average idle power consumption in watts. Obtained from experiment
	fog_bw	49.370 Mbps	TCP bandwidth of the fog node. Obtained from experiment



(a) Accuracy on KDD99 **(b)** Accuracy on NSL-KDD

Fig. 11 Accuracy of different setup of the IDS system



(a) Diagnostic Test at 28.57% Duty Cycle **(b)** Diagnostic Test at 66.67% Duty Cycle

(c) Diagnostic Test at 28.57% Duty Cycle **(d)** Diagnostic Test at 10.00% Duty Cycle

Fig. 12 Results for the diagnostic test in the different components of the proposed system

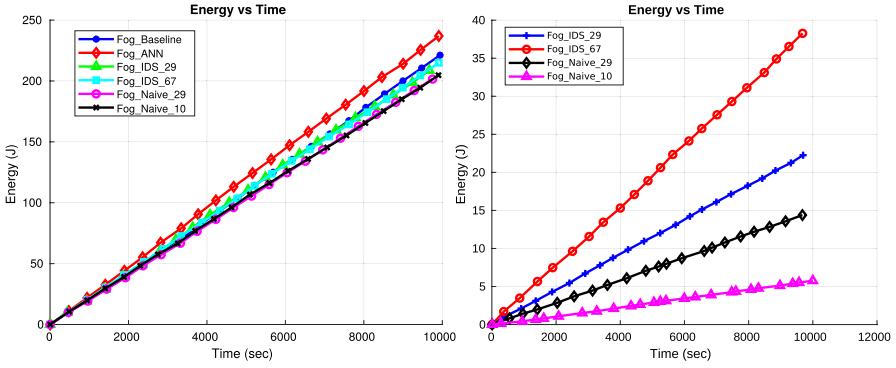
$$E_{fog} = \sum E_{service} + \sum E_{detection} \tag{8}$$

$$E_{IDS} = \sum E_{sniffed} + \sum E_{fog-IDS} + \sum E_{cloud-IDS} \tag{9}$$

Another important factor, especially in resource-limited FC, is energy. We use the results obtained from Fig. 9b to investigate the energy consumption of the system. Figure 13a, b show the energy consumption of the FGNs and the IDSNs for the different simulation scenarios in Table 7 respectively. However, we ignore the energy consumption of the cloud and the IoT layer because they are beyond the scope of this paper. The energy consumption of the FGN in the Fog_ANN system shows the most energy consumption. Equation 8 supports the result. It shows that the total energy consumption of the FNG is the sum of energy consumption due to services it provides to the IoT layer and energy consumption due to intrusion detection. The second energy consumer is the system without any intrusion detection technique, Fog_Baseline. It consumes energy because it processes more packets than when it does not deploy an IDS. When it deploys an IDS, it only processes benign packets, while without IDS, it processes both malicious and non-malicious packets.

In summary, Fig. 13a shows that the energy consumption of the FGNs is linear. It is also clear that the duty cycle of the IDSN has no impact on the energy consumption of the FGNs in the proposed system. Therefore, the energy consumption rate is directly proportional to the ability of the proposed system to accurately detect true-negative values. This notion is plausible because the system only deepens investigation on getting a positive result (i.e., TP or FP). But, deeper investigations mean more energy consumption.

However, the energy consumption of the IDSNs shown in Fig. 13b mainly depends on their duty cycle, albeit it partly depends on the ability of the nodes to accurately detect true-negative packets too. We can see that the energy consumption of the system is linear. Also, the energy consumption is directly proportional to the



(a) Energy Consumption of FGNs **(b)** Energy Consumption of IDSNs

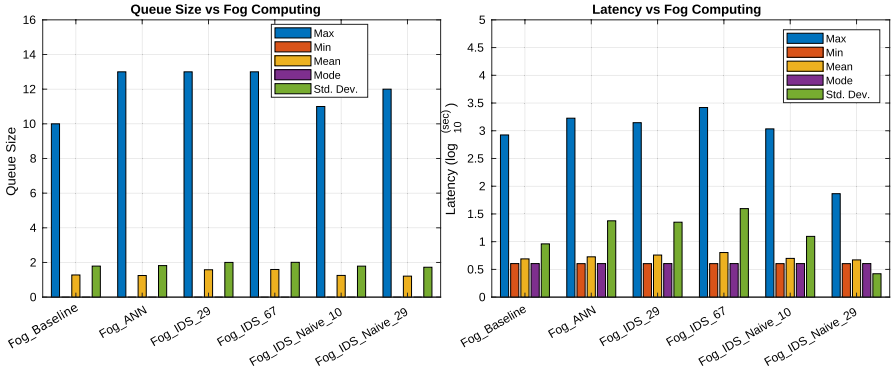
Fig. 13 Energy consumption for different FC systems

duty cycle in each of the results. As the duty cycle of the IDSN increased from 29% to 67%, the energy consumption of the system doubled. Furthermore, the energy consumption increased by a factor of 3 when the duty cycle of the IDSN increased from 10 to 29%. Moreover, the IDSN in Fog_Naive_29 saves 33% more energy than Fog_IDS_29, even though they have the same duty cycle. It is because, in the case of Fog_Naive_29, Eq. 9 has the term $\sum E_{fog-IDS} = 0$ since there are no communications between the FGN and the IDSN.

We also investigated the latency of the proposed system. A queue was added to the IoT layer. When the FGN is busy, it replies with a negative acknowledgment. The request is then added to the queue because it is assumed that the IoT node will continue trying until they are serviced. In this paper, latency is defined as the time it takes to service a request—from the time an IoT node generates it until an FGN services it. Figure 14a shows the performance of the queue. Figure 14b is the latency of service provided by the Fog layer.

A study of Fig. 14a shows the minimum queue size and the mode are zero (0), which means most of the time, there is no congestion in the queue. As expected, the Fog_Baseline scenario has the smallest value for the maximum queue size (MQS) at any time because it has no IDS. The two naive simulations of the proposed system have the second least MQS because their FGNs do not carry out any intrusion detection. Only their IDSNs do. Moreover, Fog_Naive_10 has less MQS than Fog_Naive_26 because its IDSN sleeps more. Thus, reducing the frequency of detection, which in turn reduces the notification messages sent to the FGN. Hence, the FGN has more time to service more IoT nodes. The highest MQS are found in Fog_ANN, Fog_IDS_29, and Fog_IDS_67 because their FGNs perform two activities; intrusion detection and servicing the IoT layer.

To investigate the latency, Fig. 14b is used. Notice that Fog_Naive_29 has lower latency than all the other systems. Intuitively, the Fog_Baseline should have the least latency, since it did not deploy IDS. As such, it has no IDS-related overhead like the other scenarios. But this is not the case; the FGN in the Fog_Baseline services every node, both malicious and non-malicious, while the FGN in Fog_Naive_29 only



(a) IoT Layer's Queue Performance **(b)** Latency of Different Fog Systems

Fig. 14 Performance different FC systems

services those nodes that are non-malicious and drops packets from those that are malicious. In addition, it does not carry out any intrusion detection—it lets its IDSN do the detection. Thus, allowing it to save more time.

Also, the figure shows that the latency of Fog_Naive_10 higher than that of Fog_Naive_29. The reason is that Fog_Naive_10 wakes up for only 10% of the time. As such, Fog_Naive_29 discovers more malicious nodes per unit time compared to Fog_Naive_10. Therefore, it reaches a steady-state where all malicious nodes are blacklisted earlier. Consequently, its FGN saves more time by dropping packets from the blacklisted malicious nodes. Thus, increasing the service rate of its FGN.

4.3 Comparison

In this paper, we propose a HIS-based IDS. The proposed IDPS detects intrusions in the fog layer and mitigates the attack by blacklisting the attacking node. In this section we compare the proposed system with [13, 27, 39]. These papers were discussed in Sect. 2. Furthermore, we compared the performance of the proposed system using both KDD99 and NSL-KDD.

Table 10 shows the accuracy of the proposed system along with the accuracy of other systems in the literature. The results show that the KDD99 dataset outperforms NSL-KDD. The reason is that the NSL-KDD test set contains some classes/labels excluded from the training set. It can be seen that our proposed system (at its best configuration) outperforms [13] and [39]. Its improved performance is due to the collaboration of the FNGs, the IDSNs, and the cloud in detecting attacks. The FGN asks the IDSN, and the IDSN asks the cloud to validate detection results whenever the results are positive.

Table 10 Comparison of the proposed system with papers in literature

IDS	Accuracy (%)	Latency (s)	Ave queue size	Dataset	Source
Fog_IDS_29	93.96	6.27	1.57	KDD99	Proposed system
Fog_IDS_67	96.71	6.18	1.58		
Fog_Naive_10	98.36	5.49	1.23		
Fog_Naive_29	98.52	5.06	1.23		
Fog_Naive_67	98.77	4.97	1.22		
Fog_IDS_29	85.38	6.07	1.50	NSL-KDD	Proposed system
Fog_IDS_67	90.65	6.00	1.53		
Fog_Naive_10	96.74	5.18	1.21		
Fog_Naive_29	96.69	4.98	1.25		
Fog_Naive_67	96.14	5.19	1.22		
[13]	96.90[a]	–	–	NSL-KDD	Literature
[39]	94.00	–	–	ADFA-LD	
[39]	74.00	–	–	ADFA-WD	
[27]	98.40	6.19	1.4	KDD99	

[a]The authors provided data rate, not accuracy

On the contrary, a lower accuracy is recorded, when the statistics-based IDS is deployed in the FGNs. As in Fog_IDS_29 and Fog_IDS_67, the FGNs generate more false-negative results. Thus, increasing the false alarm rates in the system. As for [13] and [39], the detection is only carried out on the FGN, which lowers the performance of the IDS.

Consequently, when we deployed the system in [27] in the setup in Fig. 10b, we found that our system is slightly faster. The proposed system has similar performance with [27], because it deploys a similar strategy. The system carries out detection with the help of both the cloud, the fog layer, and the edge nodes. However, [27] is not scalable, because the system generates detectors in the resources-limited fog layer and IoT layer.

5 Conclusion

In this paper, an Intrusion detection system that mimics the human immune system is proposed. Just like the human immune system, the proposed system breaks down the intrusion detection technique into smaller functions. It allocates the functions to the different parts of the FC system: the FGNs carry out innate immune by using the statistics-based IDS, while specialized IDSNs carry out adaptive detection using ANN. This mechanism achieves an accuracy of 98.8% in KDD99 and 96.7% in NSL-KDD datasets. Also, it reduces the energy consumption by 10%.

Also, it is found that the system's accuracy improves with an increase in the duty cycle. However, this causes an increase in energy consumption and latency. Moreover, it is found that higher accuracy is achieved when the statistics-based IDS is removed from the FGN. This also reduces the energy overhead of both the IDS and the FGN and reduces the system's latency.

Acknowledgements The authors would like to thank the Computer Engineering Department, King Fahd University of Petroleum and Minerals for their support. We would like to thank Dr. Mustapha Aliyu Muhammad (M.D) and Dr. Aliyu Aliyu Muhammad (M.D) for their suggestions and recommendations in the course of this research.

Declarations

Conflict of interest The authors declare that they have no conflict of interest.

References

1. Prabhu, C.: Fog Computing. Springer, Deep learning and big data analytics-research directions (2019)
2. Turner, V., MacGillivray, C., Gaw, J., Clarke, R., Morales, M., Kraus, B.: IDC futurescape: worldwide internet of things 2015 predictions. In: IDC (2014)
3. Computing, F.: The internet of things: extend the cloud to where the things are (2016)
4. Li, C., Qin, Z., Novak, E., Li, Q.: Securing SDN infrastructure of IoT-fog networks from MITM attacks. IEEE Internet Things J. **4**(5), 1156–1164 (2017)

5. Stojmenovic, I., Wen, S.: The fog computing paradigm: scenarios and security issues. In: 2014 federated conference on computer science and information systems, pp. 1–8 (2014). https://doi.org/10.15439/2014F503
6. Hu, P., Dhelim, S., Ning, H., Qiu, T.: Survey on fog computing: architecture, key technologies, applications and open issues. J. Netw. Comput. Appl. **98**, 27–42 (2017)
7. Sequeira, D.: Intrusion prevention systems: securitys silver bullet? Bus. Commun. Rev. **33**(3), 36–41 (2003)
8. Mauritian Computer Emergency Response Team: guideline on intrusion detection and prevention systems (2011). https://ncb.govmu.org/portal/sites/ncb/downloads.html
9. Scarfone, K., Mell, P.: Special Publication 800–94: Guide to Intrusion Detection and Prevention Systems. National Institute Standard and Technology, Gaithersburg (2012)
10. Khraisat, A., Gondal, I., Vamplew, P., Kamruzzaman, J.: Survey of intrusion detection systems: techniques, datasets and challenges. Cybersecurity **2**(1), 20 (2019)
11. Aloqaily, M., Balasubramanian, V., Zaman, F., Al Ridhawi, I., Jararweh, Y.: Congestion mitigation in densely crowded environments for augmenting qos in vehicular clouds. In: Proceedings of the 8th ACM Symposium on Design and Analysis of Intelligent Vehicular Networks and Applications, DIVANet'18, pp. 49–56. Association for Computing Machinery, New York (2018). https://doi.org/10.1145/3272036.3272038
12. Balasubramanian, V., Aloqaily, M., Reisslein, M.: An SDN architecture for time sensitive industrial IoT. Comput. Netw. **186**, 107739 (2021). https://doi.org/10.1016/j.comnet.2020.107739
13. Otoum, Y., Nayak, A.: As-ids: anomaly and signature based ids for the internet of things. J. Netw. Syst. Manag. **29**(3), 1–26 (2021)
14. Almiani, M., AbuGhazleh, A., Al-Rahayfeh, A., Atiewi, S., Razaque, A.: Deep recurrent neural network for IoT intrusion detection system. Simul. Model. Pract. Theory **101**, 102031 (2020). https://doi.org/10.1016/j.simpat.2019.102031
15. Pacheco, J., Benitez, V.H., Félix-Herrán, L.C., Satam, P.: Artificial neural networks-based intrusion detection system for internet of things fog nodes. IEEE Access **8**, 73907–73918 (2020)
16. Al-Omari, M., Rawashdeh, M., Qutaishat, F., Mohammad, A., Ababneh, N.: An intelligent tree-based intrusion detection model for cyber security. J. Netw. Syst. Manag. **29**(2), 1–18 (2021)
17. Ou, C.M.: Host-based intrusion detection systems inspired by machine learning of agent-based artificial immune systems. In: 2019 IEEE International Symposium on INnovations in Intelligent SysTems and Applications (INISTA), pp. 1–5. IEEE (2019)
18. Wang, W., Ren, L., Chen, L., Ding, Y.: Intrusion detection and security calculation in industrial cloud storage based on an improved dynamic immune algorithm. Inf. Sci. **501**, 543–557 (2019)
19. Igbe, O., Saadawi, T., Darwish, I.: Digital immune system for intrusion detection on data processing systems and networks (2017). US Patent App. 15/633,056
20. Greensmith, J., Aickelin, U.: The deterministic dendritic cell algorithm. In: International Conference on Artificial Immune Systems, pp. 291–302. Springer (2008)
21. Rhys, H.: Classifying with decision trees. Manning Publications (2020). https://books.google.com.sa/books?id=jRzYDwAAQBAJ
22. Jansen, S.: Chapter 10: decision trees and random forests. Packt Publishing (2018). https://books.google.com.sa/books?id=tx2CDwAAQBAJ
23. Pump, R., Ahlers, V., Koschel, A.: State of the art in artificial immune-based intrusion detection systems for smart grids. In: 2018 Second World Conference on Smart Trends in Systems, Security and Sustainability (WorldS4), pp. 119–126. IEEE (2018)
24. Matzinger, P.: Tolerance, danger, and the extended family. Ann. Rev. Immunol. **12**(1), 991–1045 (1994)
25. Brownlee, J.: Clever algorithms: nature-inspired programming recipes. Lulu.com (2011). https://books.google.com.sa/books?id=SESWXQphCUkC
26. Hosseinpour, F., Amoli, P.V., Farahnakian, F., Plosila, J., Hämäläinen, T.: Artificial immune system based intrusion detection: innate immunity using an unsupervised learning approach. Int. J. Digital Content Technol. Appl. **8**(5), 1 (2014)
27. Hosseinpour, F., Vahdani Amoli, P., Plosila, J., Hämäläinen, T., Tenhunen, H.: An intrusion detection system for fog computing and IoT based logistic systems using a smart data approach. Int. J. Digital Content Technol. Appl. **10** (2016)
28. Ye, N., Chen, Q.: An anomaly detection technique based on a chi-square statistic for detecting intrusions into information systems. Qual. Reliab. Eng. Int. **17**(2), 105–112 (2001). https://doi.org/10.1002/qre.392

29. Hegde, C., Jiang, Z., Suresha, P.B., Zelko, J., Seyedi, S., Smith, M.A., Wright, D.W., Kamaleswaran, R., Reyna, M.A., Clifford, G.D.: Autotriage—an open source edge computing raspberry pi-based clinical screening system. medRxiv (2020). https://doi.org/10.1101/2020.04.09.20059840
30. Xhafa, F., Kilic, B., Krause, P.: Evaluation of IoT stream processing at edge computing layer for semantic data enrichment. Fut. Gener. Comput. Syst. **105**, 730–736 (2020). https://doi.org/10.1016/j.future.2019.12.031
31. Xunlong Software CO., Limited: orange pi lite—orange pi (2016). http://www.orangepi.org/orangepilite/. Accessed May, 2020
32. Nath, O.: Review on raspberry pi 3b+ and its scope. Int. J. Eng. Appl. Sci. Technol. **4**(9), 157–159 (2020)
33. LCD wiki: 3.5inch rpi display - lcd wiki (2020). http://www.lcdwiki.com/3.5inch_RPi_Display. Accessed 17th Aug 2020
34. Crovella, M.E., Carter, R.L.: Dynamic server selection in the internet. In: Third IEEE workshop on the architecture and implementation of high performance communication subsystems (HPCS) (1995)
35. OpenNN.net: Opennn: open neural networks library (2020). https://www.opennn.net/
36. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., et al.: Scikit-learn: machine learning in python. J. Mach. Learn. Res. **12**, 2825–2830 (2011)
37. Long, J.: Interesting native code examples (2012). https://bit.ly/3fYmfkN. Accessed 25 May 2020
38. What is omnet++? (2019). https://omnetpp.org/intro/. Accessed 6 June 2020
39. Sudqi Khater, B., Abdul Wahab, A., Idris, M., Abdulla Hussain, M., Ahmed Ibrahim, A.: A lightweight perceptron-based intrusion detection system for fog computing. Appl. Sci. **9**(1), 178 (2019)
40. Krügel, C., Toth, T., Kirda, E.: Service specific anomaly detection for network intrusion detection. In: Proceedings of the 2002 ACM symposium on applied computing, pp. 201–208 (2002)
41. Farouq, A., Tarek, S., Mohamed, D.: farooouq/idps_omnet: Intrusion detection and prevention system for fog computing using omnet++ (2020). https://github.com/farooouq/IDPS_OMNET

Farouq Aliyu received the B.S. in computer engineering from the Bayero University Kano (BUK), Kano, Nigeria, in 2010 and M.S. in computer engineering from King Fahd University of Petroleum and Minerals (KFUPM), Dhahran, Saudi Arabia, in 2015. Currently, he is a Ph.D. candidate in Department Computer Engineering at KFUPM, Dhahran, Saudi Arabia.

Tarek R. Sheltami received his Ph.D. in Electrical and Computer Engineering from the Electrical and Computer Engineering Department at Queen's University, Kingston, Ontario, Canada on April 2003. Dr. Sheltami is currently a Professor at the Computer Engineering Department at King Fahd University of Petroleum and Minerals (KFUPM), Dhahran, Kingdom of Saudi Arabia.

Mohamed Deriche received his undergraduate degree from the National Polytechnic School of Algeria in 1985. He then joined University of Minnesota, USA, where he completed his MS and PhD in 1988, and 1992 respectively. In 2001, he joined the EE Department at King Fahd University of Petroleum & Minerals, Saudi Arabia, where he is currently leading the signal processing group.

Nidal Nasser received the Ph.D. degree from the School of Computing, Queen's University, Kingston, ON, Canada, in 2004. He worked at the School of Computer Science, University of Guelph, Guelph, ON, Canada, where he was the Founder and the Director of the Wireless Networking and Mobile Computing Research Lab. He is currently a Professor of Software Engineering with the College of Engineering, Alfaisal University, Saudi Arabia.