applied sciences

MDPI

Article

Mitigating DDoS Attacks in SDN-Based IoT Networks Leveraging Secure Control and Data Plane Algorithm

Song Wang [1,*], Karina Gomez [1], Kandeepan Sithamparanathan [1], Muhammad Rizwan Asghar [2], Giovanni Russello [2] and Paul Zanna [3]

1 School of Engineering, RMIT University, Melbourne, VIC 3000, Australia; karina.gomezchavez@rmit.edu.au (K.G.); kandeepan.sithamparanathan@rmit.edu.au (K.S.)
2 Cyber Security Foundry, The University of Auckland, Auckland 1142, New Zealand; r.asghar@auckland.ac.nz (M.R.A.); g.russello@auckland.ac.nz (G.R.)
3 Northbound Networks, Hoppers Crossing, VIC 3029, Australia; paul@northboundnetworks.com
* Correspondence: s3478896@student.rmit.edu.au

Abstract: Software-Defined Networking (SDN) and Internet of Things (IoT) are the trends of network evolution. SDN mainly focuses on the upper level control and management of networks, while IoT aims to bring devices together to enable sharing and monitoring of real-time behaviours through network connectivity. On the one hand, IoT enables us to gather status of devices and networks and to control them remotely. On the other hand, the rapidly growing number of devices challenges the management at the access and backbone layer and raises security concerns of network attacks, such as Distributed Denial of Service (DDoS). The combination of SDN and IoT leads to a promising approach that could alleviate the management issue. Indeed, the flexibility and programmability of SDN could help in simplifying the network setup. However, there is a need to make a security enhancement in the SDN-based IoT network for mitigating attacks involving IoT devices. In this article, we discuss and analyse state-of-the-art DDoS attacks under SDN-based IoT scenarios. Furthermore, we verify our SDN sEcure COntrol and Data plane (SECOD) algorithm to resist DDoS attacks on the real SDN-based IoT testbed. Our results demonstrate that DDoS attacks in the SDN-based IoT network are easier to detect than in the traditional network due to IoT traffic predictability. We observed that random traffic (UDP or TCP) is more affected during DDoS attacks. Our results also show that the probability of a controller becoming halt is 10%, while the probability of a switch getting unresponsive is 40%.

Keywords: DDoS; SDN; IoT; OpenFlow; Zodiac; security; Packet_In message; TCP/UDP

check for updates

Citation: Wang, S.; Gomez, K.; Sithamparanathan, K.; Asghar, M.R.; Russello, G.; Zanna, P. Mitigating DDoS Attacks in SDN-Based IoT Networks Leveraging Secure Control and Data Plane Algorithm. *Appl. Sci.* **2021**, *11*, 929. https://doi.org/10.3390/app11030929

Received: 27 December 2020
Accepted: 18 January 2021
Published: 20 January 2021

1. Introduction

Software-Defined Networking (SDN) introduces an innovative architecture to decouple the control and data plane, which otherwise are intermingled in traditional networks. Basically, SDN divides a network into three layers: application layer, control layer and data layer. SDN switches in the data plane are deprived of the ability of thinking and are managed by a centralised controller in the control plane. The advantage of this revolution is obvious as there is an ease of management. However, the controller can easily become a single point of failure. That is, once the controller is down, all the SDN switches attached might stop working because they lose connection to the controller. Although some SDN switches can be configured to work in the traditional mode when disconnected from the controller, it is no longer SDN, thus providing no flexibility. The link between the control and data plane is defined by the OpenFlow protocol [1] or P4 [2], which consults with the controller about the decision regarding how to process certain packets. The controller generates flow rules according to its running applications, and then sends these rules to the switch to manage the network behaviour. IoT enables machine-to-machine communications and data exchange to broaden the range of coverage area. Through sensors, identification and ubiquitous computing, IoT tries to involve a diverse range of devices and merge different networks altogether to achieve localisation, monitor, management, etc. In outdoor deployments, the number of devices under the network has an exponential growth due to a high demand for environment monitoring and data collection [3]. Recently, new technologies—such as LoRa, DASH7 and Narrowband (NB-IoT)—promise to provide low-power and long-range connectivity solutions for IoT applications, and they also meet the key requirements of low cost, long battery life, extended coverage area, support for a massive number of connected devices (scalability), security and privacy. However, the massive number of IoT devices connected to the same network increases the attack vector, which raises new security issues for IoT networks. These security issues, include, but are not limited to, malicious code attacks, inability to receive security patches, hacking smart meters, eavesdropping, sniffing attacks and Distributed Denial of Service (DDoS) attacks [4].

DDoS attacks are a common threat to the network, although the attacker typically does not aim to steal any data. Basically, DDoS attacks aim at consuming system resources until the target is not available to offer its services. DDoS attacks could be divided into three categories: application layer attack, protocol attack and volumetric attack. For volumetric attack, an attacker can deplete the available resources of the victim or bandwidth towards the target. Not only the data plane in the SDN, but also the controller and southbound interface, could suffer from this kind of attack as well; this is because a client host can trigger inquiry from the data plane to control plane. Although there have been a lot of discussions about DDoS attacks in the SDN and IoT networks, the large number of IoT gadgets is still a good chance to launch attacks, as well as the communication link between controllers and switches in SDN [5,6]. Additionally, more validations in the real network are required. Moreover, the programmability and centralised control in the SDN give users more options to probe into this threat. In this paper, volumetric attack is implemented.

Research Contributions

IoT could benefit from the combination with SDN, particularly in the reliability owing to the global vision of SDN controller. SDN-based IoT networks allow dynamic access control in home networks as well as support authentication and authorisation [7]. However, numerous types of devices and their tremendous numbers in the IoT make them an ideal target for DDoS attackers. In this paper we conduct the following.

- We extend our previous work [8,9] from DoS to DDoS attacks using IoT traffic models and complex scenarios. Moreover, we focus on adapting and improving our previous algorithm in order to detect and block DDoS attacks in the SDN-based IoT networks hinged on several IoT traffic scenarios.
- We also deeply analyse the state-of-the-art DDoS attacks in the SDN and SDN-based IoT networks.
- We replicate IoT realistic traffic models and DDoS attacks by emulations and physical SDN-based IoT hardware and software equipment. From the real implementation, we have authentic results to evaluate the impact and outcome of DDoS attacks in the IoT networks.
- Our results show that the improved version of our SECOD algorithm [8,9] is able to efficiently detect and block DDoS attacks in SDN-based IoT networks.
- We also report observations on the effect of DDoS attacks with periodical and event trigger IoT traffic in the SDN architecture.

The rest of this article is organised as follows. Section 2 reviews related work. Section 3 describes existing IoT traffic model applications. The SDN-based IoT architecture is presented in Section 4. The implementation of DDoS attacks and defence techniques is covered in Section 5. Results and discussion are given in Section 6. Finally, Section 7 concludes this article and gives directions for future work.

2. Related Work

DDoS attacks have been well investigated for SDN networks, where the target of DDoS attackers may be the control plane or data plane, compromising the controller or SDN switches, respectively [10,11]. In this section, we provide a review of DDoS attacks on SDN in Tables 1 and 2 followed by a review of security concerns of SDN-based IoT networks.

Table 1. Existing solutions for detecting and defending Software-Defined Networking (SDN) against Distributed Denial of Service (DDoS) attacks.

Algorithm	Simulation	Real Implementation	Detection or Defence	Control or Data Plane	Advantages	Disadvantages
Dao et al. [12]	✓	✗	Both	Both	Feasible and accurate in the small network.	Resource consumption is high when confronting mature attacks.
Mousavi et al. [13]	✓	✗	Detection	Both	Low resource consumption and short detection time.	Hard to define the threshold of detection for different applications.
Dong et al. [14]	✗	✗	Detection	Control Plane	Prompt and accurate, improvement in false positive and false negative issues.	No simulation or implementation related to SDN.
Yan et al. [15]	✓	✗	Both	Data Plane	Low false positive.	High latency, especially for the users sharing the same port with the attacker.
Dharma et al. [16]	✗	✗	Both	Control Plane	Further inspection to decrease false positive.	Produce delay to legitimate users.
Shoeb et al. [17]	✗	✗	Both	Both	Able to protect the flow table on the switch.	Hard to define key parameters, such as peak time.
Xiao et al. [18]	✓	✗	Detection	Data Plane	High detection rate and low false positive.	Hard to define key parameters, such as abnormal link utilisation. Limited to link flooding attacks.
Kokila et al. [19], Phan et al. [20]	✓	✗	Detection	Control Plane	High accuracy and low false positive.	Performance is based on the training dataset.
Lim et al. [21]	✓	✗	Both	Data Plane	Capable of localising attackers and transformable countermeasures.	Hard to define the metric and threshold.
Chin et al. [22]	✓	✗	Detection	Data Plane	Effective and scalable.	Require extra device and interface.
Macedo et al. [23]	✓	✗	Both	Control Plane	Associate multiple controllers instead of extra devices to mitigate DDoS attacks.	High-frequency attack may result in continual leader controller election.

Table 1. *Cont.*

Algorithm	Simulation	Real Implementation	Detection or Defence	Control or Data Plane	Advantages	Disadvantages
Hameed et al. [24]	✓	✗	Both	Data Plane	Allows inter-domain defence of DDoS attacks.	Attack with spoofed IP address will make controllers block normal users.
Sahay et al. [25]	✓	✓	Defence	Data Plane	The collaboration between the controller on the customer side and ISP side enables quick response to DDoS attacks.	The link between ISP controller and customer controller becomes a new threat in this model.

Table 2. Countermeasures in SDN-based IoT networks.

Algorithm	Simulation	Real Implementation	Detection or Defence	Control or Data Plane	Advantages	Disadvantages
Tortonesi et al. [26]	✓	✗	Defence	Data Plane	Capable of edge defence due to the programmable controller in the data plane, the work load on the control plane is shared by the data plane.	The interface between SPF controller and programmable controller needs to be defined. The gateway devices shall support both SDN and programmable controller.
Özçelik et al. [27]	✓	✗	Both	Data Plane	Fog computing enables the mitigation of DDoS attacks at the ingress of the network.	Defence mainly focuses on Mirai botnet and TCP protocol, other attack types might need validation.
Sarwar et al. [28]	✓	✗	Both	Control Plane	Existing trustful users can still interact with the controller during DDoS attacks.	The controller rejects all the new flows under DDoS attacks. A legitimate user without a high trust value will encounter more delays when the network is busy.
Ravi et al. [29]	✓	✓	Both	Both	The real-time training keeps the flow rules that are generated by machine learning up-to-date.	The performance of LEDEM relies on the training dataset. The link between local and universal controllers needs to be defined.
Sharma et al. [30]	✓	✗	Detection	Both	The proposed mechanism is able to detect not only DDoS attacks but also other types of attacks.	New-flow attacks against one network slice might saturate system resources, as it triggers a new flow record and the update in the detection pattern.

Table 2. *Cont.*

Algorithm	Simulation	Real Implementation	Detection or Defence	Control or Data Plane	Advantages	Disadvantages
Nobakht et al. [31]	✗	✓	Both	Data Plane	Detection is based on the IoT application, which means the behaviour in the network is predictable for a specific application.	The performance of detection might highly rely on the habit of using the application. For a flexible application, it could result to a low accuracy in the detection.

In Tables 1 and 2, we systematically analyse each algorithm. More specifically, we provide a comparative analysis in the light of simulation or real implementation, whether it puts forward a defence mechanism or not, the focus of the algorithm in terms of data and control plane and a summary of the advantages and disadvantages of existing algorithms.

2.1. DDoS Attacks on SDN

As DDoS attacks have different behaviours, several techniques are used to detect and block those attacks.

Dao et al. [12] define a table in the controller to track the packets by IP address during a DDoS attack. All the new packets are regarded as suspicious packets and assigned a small timeout value in the flow entry. The number of packets using that connection is also compared with a minimum value to determine if it is a normal request or an attack. From the simulation, this method effectively reduces flow entries in the switch, and the bandwidth of controller-switch channel is still available during DDoS attacks. However, this mechanism consumes a huge amount of resources on the controller if the attacker modifies source address.

Mousavi and St-Hilaire [13] propose to use entropy for DDoS detection due to its ability to measure randomness, where two essential components are time period and threshold. Although it may improve detection accuracy in the real network, the proposed techniques only address detection without providing countermeasures. Similarly, Dong et al. [14] suggest a statistical tool, called Sequential Probability Ratio Test (SPRT), to improve existing false positive and false negative issues. It predefines two boundaries (A and B, $B < A$) related to the probabilities of false positive (a) and false negative (b) (it is suggested $A = b/(1-a)$, $B = (1-b)/a$), and the decision is made from the log-probability ratio. The evaluation of the DARPA Intrusion Detection Data Sets [32] shows its promptness and accuracy. However, the proposed method is evaluated using only mathematical results without simulations, where random variables can be introduced.

Yan et al. [15] propose a "Multislot" strategy to process requests in each time slot so that legitimate users can communicate to each other properly during DDoS attacks. However, if the subscriber and attacker share the same switch port, large flow latency will be introduced because it places the legitimate and malicious request in the same queue. Dharma et al. [16] propose to use a "flow collector", which sits between the switch and the controller. When the number of invalid packets exceeds the threshold within a certain duration, the flow collector is triggered to further inspect those suspicious packets. However, this introduces a delay to legitimate users. Furthermore, there is no mathematical analysis, simulations or real implementation.

Shoeb and Chithralekha [17] define a peak time and establish a trust level to defend control and data planes against DDoS attacks. The node's trust level is used to determine the priority of processes on the controller, where the value is set depending on the behaviour during normal time. During peak time, the controller discards requests from particular nodes whose number of requests already exceeds a certain threshold. Even for the normal nodes, the controller replies switch a new rule with a lower timeout value. However, the authors do not indicate how to define a peak time and the threshold of the peak time. Moreover, the proposed technique is not simulated or tested on the real equipment. Xiao et al. [18] propose a model to use a Bloom filter to deal with the detection of link flooding attack in the SDN. This model contains two subsystems: (i) collector and (ii) detector. When the link utilisation is abnormal, the collector scans the flow table on the switch and finds the abnormal flows from the statistics of flow entries. The detector monitors the entire network using a controller; therefore, it can sniff packets. The classification of these packets are sent to the Bloom filter to determine whether it is abnormal, because relevant IP features are stored in the Bloom filter. However, there is no definition of abnormal link utilisation, and how to detect this issue on the controller is also unmentioned.

Kokila et al. [19] propose to detect DDoS attacks using a Support Vector Machine (SVM) classifier. The SVM learns the pattern with training samples and predicts the unknown traffic sample to be normal or attack. The 2000 DARPA intrusion detection scenario specific dataset is taken to instruct the SVM. The SVM has a higher accuracy and lower false positive comparing with other methods from the simulation. However, the performance of SVM is deeply based on the training dataset. Similarly, Phan et al. [20] propose to use the combination of SVM and SOM to classify DDoS attacks. SVM and SOM are trained by ready-made datasets before the model is used for testing. Each protocol has a dedicated SVM to filter traffic in the control plane. If a specific flow is in the attack region according to the SVM, it is then sent to the classifier. If the flow is in the vogue region, it is sent to SOM to make the decision. The simulations indicate that the combination of SVM and SOM has better performance than deploying them individually.

Lim et al. [21] propose to modify the IP address of the victim to mitigate DDoS attacks. The DDoS Blocking Application (DBA) running on the controller has a secure channel, which is directly connected to the server. Once the server detects DDoS attacks using some metrics, DBA assigns the server a new IP address and asks switches to redirect packets to this new address. After updating the IP address, if a host still sends packets to the previous address and the number exceeds a predefined threshold, the host is blocked as a bot. The simulation results show that DDoS attacks from bots are blocked. However, how to define the metric and threshold to trigger defence and drop action is not mentioned. Chin et al. [22] present an approach to detect DDoS attacks by the coordination of Monitor, Correlator and Controller. The Monitor part observes network for anomaly detection and triggers an Intrusion Detection System (IDS) to further inspect packets. Once the IDS confirms the attack, it delivers relevant information to the Correlator and alerts the Controller to retrieve the flow table in the switch. If any relevant feature in the flow table matches the suspicious element, the controller instructs the switch to behave according to the prevention actions.

Only few works endorse multi-domain networks. Macedo et al. [23] propose a multi-controller cluster model, called Protocol for DDoS Attack miTigation in Multi-contrOller SDN networkS (PATMOS). PATMOS has three phases: (i) find the overloaded controller via delay in the control message or its stability, (ii) elect the best performance controller to coordinate mitigation and (iii) minimise the effects of DDoS attacks. Simulation results show the efficiency of PATMOS in reducing CPU utilisation, increasing throughput and decreasing latency.

Hameed and Khan [24] propose a controller-to-controller (C-to-C) secure protocol to deploy a collaborative DDoS mitigation method. The C-to-C protocol mainly comprises three sections: data, certificate and signature. The Certificate part sets up a chain for authentication between controllers, while the signature part verifies authenticity and integrity. Once a controller detects a DDoS attack, it updates the policy on the data plane and sends a list of malicious IP addresses to the neighbouring controller. Thus, these packets are blocked in different network domains. Simulation results show that this method takes a short time to inform neighbouring controllers and mitigate attacks. Sahay et al. [25] propose a DDoS defence framework, called ArOMA, to mitigate malicious traffic on the data plane. ArOMA uses FlowID to identify flows in the network, and the detection engine on the customer side determines whether the traffic flow going to the ISP is suspicious. The controller on the customer side informs the controller on the ISP side of the flow state. The ISP controller decides which path to send the malicious flow to the filter for a further check. However, the communication between controllers needs protection as well.

2.2. Security in SDN-Based IoT Networks

As explained in the previous section, DDoS attacks in SDN, though a well-investigated topic, still require attention to improve DDoS detection and mitigation. One of the biggest challenges in the IoT network is also security, because of the vulnerabilities in IoT devices [33]. If we combine SDN with IoT networks, it could be a potential solution to improve the security feature in the IoT network, especially against DDoS attacks [34].

Tortonesi et al. [26] propose an SDN-based architecture, called SPF, to alleviate the burst of information in IoT, which is similar to the DDoS attack scenario. Data plane in the SDN is replaced by a processing and dissemination plane, which contains a programmable module. Instructions from the SPF controller are received through this module. The processing module filters requests from user application by the type of service, and then sets the priority as a reference for dissemination. Smart decisions are made on the basis of the knowledge of SDN controllers.

Özçelik et al. [27] utilize a scheme, called Edge-Centric Software-Defined IoT Defence (ECESID), to detect and mitigate DDoS attacks on the edge node of IoT network. The detection phase is based on the hypotheses that (i) a benign connection is more likely to succeed than a malicious one, and (ii) the frequency of connection requests from an infected host is much higher than from a normal host. Thus, from the record of connection attempts and failure counts within a given period, a host can be defined as infected or not. Then, updated flow rules are inserted from the SDN controller to the switch to block all the flows originated from that malicious host.

Sarwar et al. [28] propose a mechanism based on trust level of users to protect SDN controller against DDoS attacks in the IoT paradigm. Each user in the network is assigned a trust value refer to its history record, and a higher trust value means the controller gives a higher priority on the request, while a low trust level could lead to discarding packets from that user. Moreover, the controller has a buffer to queue requests, if this buffer is also overwhelmed, the request with the lowest trust level is dropped to make room for a more trustworthy inquiry.

Ravi et al. [29] involve semi-supervised machine learning algorithms to detect DDoS attacks in the IoT network. The proposed mechanism, called learning-driven detection mitigation (LEDEM), has a hierarchical control plane and divides IoT devices into two categories: fixed (fIoT) and moving (mIoT). A local controller is responsible for a small part of network, and a universal controller manages all the local controllers. LEDEM detects DDoS attacks based on the feature of incoming packets; a machine learning model is trained with both labelled and unlabelled data to make the decision if an IoT device is compromised. Sharma et al. [30] also propose to use machine learning to detect DDoS attacks, they combine both cloud networks and wireless SDN to protect IoT networks against DDoS attacks. Once a new flow enters the network, the so-called OpCloudSec scheme uses deep brief network to analyse if it is an attack flow. Normal flows will be forwarded and known attack flows will be reported to the controller for further instructions. Even if it is a new kind of attack, OpCloudSec allows administrative update in the attack pattern database so that it becomes a known attack next time.

Nobakht et al. [31] propose a smart home-based model, called IoT-IDM, which employs machine learning in the intrusion detection in IoT networks. As smart home IoT devices need to be turned on/off manually, IoT-IDM selects the number of bytes in the command and response packets, along with inter-packet interval to be the detection metrics. By using both linear and nonlinear logistic regression model of machine learning, the precision rate could be over 94% from real implementation.

Comparing to aforementioned works, this article categorises the IoT traffic into four main types by their behaviours in the network, and demonstrates the impact of DDoS attacks and the effect of proposed algorithm on these IoT traffic scenarios. Moreover, we deeply analyse the difference in the output when modifying timeout parameters in the algorithm, as well as the performance when running over powerful and non-powerful SDN controllers.

The proposed mechanism is validated in the real test bed, it does not require auxiliary devices or extra interfaces to work. Due to its simplicity, the resource consumption is low on the controller side when blocking DDoS attacks. Unlike traditional firewalls, DDoS attacks will be mitigated at the ingress of the network, which means the impact on the bandwidth is minimal. As some IoT applications have predictable transmission behaviours, the threshold of DDoS detection is easier to define than other applications.

3. IoT Traffic Models

Among IoT applications, there are some popular real implements, features and traffic models that are discussed in this section [35]. There are two main identified traffic model trends that are considered for IoT applications: (i) periodical and (ii) event trigger traffic as explained below [36].

- **Periodical:** This is a typical IoT traffic model applied to passive monitoring applications. A monitor generates stable traffic every predetermined period to update the state of object. This kind of traffic is always predictable between an IoT end-user device and an IoT data platform collector (IoT-DPC). For example, smart city applications, IoT in agriculture, smart grids, healthcare and farming devices send periodic messages along with statistics to the IoT-DPC for monitoring.
- **Event Trigger:** This is a typical IoT traffic model applied to dynamic monitoring applications. Event trigger happens occasionally to declare that an operation may be required. Although these events add uncertainty to the traffic flow, it is still under control due to its probability of presence. Devices in smart homes, wearables, connected car, industrial IoT and smart retail might randomly generate information to the IoT-DPC. These events only arise in a low frequency; otherwise, real-time monitoring might not be necessary.

Based on periodical, event trigger or both, the IoT traffic model can be characterised for specific applications. In the following, the most popular IoT applications are elaborated.

1. *Smart Home (event trigger)* includes diverse real-time monitor and event triggered actions. Typical scenarios include home condition retrieve, preconfigured intelligent services and remote control of home appliances, achievable through sensors and Internet-connected devices at home [37]. For people who would like to manage from outside home, collected data are sent to the distant application IoT-DPC via a home gateway, such as a modem, and then the IoT-DPC tries to reach the user via mobile or wireless networks.
2. *Wearables (event trigger)* are portable devices that have the capability to save and transmit data. Similar to mobile phones, these wearables may upload or download files and communicate over the Internet. As they are always carried by users and activated erratically, it is an arduous task to predict the access point and traffic utilisation.
3. *Connected Car (event trigger)* is capable of communicating with other internal and external IoT devices, as well as accessing the Internet. Sensors embedded in the vehicle ensure the performance of driving, while modules exchanging data with peer vehicles avert the chance of a crash. As a connected car is able to gather information from devices nearby, it acts as a mobile collector in the IoT [38].

4. *Industrial IoT (event trigger)* is a future project that pushes remarkable changes in both application and technology in the manufacturing system, the objective is to achieve service-oriented industry [39]. The key point is that users hand over their behaviour and feedback to the factory in real time, and the factory reacts to these statistics by reconfiguring manufacturing process and regulating orders from suppliers [40]. As there are too many possibilities and combinations of customer feedback in this area, IoT traffic is unpredictable in these applications unless manufacturers predefine a duration for the comments.
5. *Smart City (periodical/event trigger)* employs sensors all over the city to monitor the real-time state from environment to parking space. The architecture consists of three tiers: IoT sensor, IoT gateway and server. In the IoT sensor tier, devices are deployed in the facilities for dedicated tasks, which means the traffic is stable for a kind of service under a given gateway [41].
6. *IoT in Agriculture (periodical/event trigger)* manages to read the status of farmland, such as soil moisture and temperature, to optimise the growing conditions without manual intervention [42]. As this kind of application has no requirement to share information with peers from time to time, traffic generated online is hard to foresee.
7. *Smart Retail (event trigger)* saves customer interest and shopping history in the database. Once a customer walks into the store, a scan of member card causes the end device on the trolley to download personal details from the server. Traffic between the gateway in the branch and server reflects customer flows, which could be studied from statistics in the past.
8. *Smart Grid (periodical/event trigger)* collects the usage information during the day so as to formulate a dynamic strategy for power supply. Various sensors on the transmission line monitor the change of environment and impact on the power line, they can access to the public network via mobile or optical networks [43]. As the number of sensors is solid in a region and the coverage of the server is constant, traffic flow received on the server is normally steady.
9. *Healthcare (periodical/event trigger)* via IoT is achieved by sensors attached to the user to monitor real-time body functions. The results are sent to the server through a mobile network or broadband [44]. As these devices may connect to the server via different access points, traffic is unpredictable in this case.
10. *IoT in Farming (periodical/event trigger)* fetches data from sensors and cameras in the poultry house to maintain a comfortable environment. A notification is sent to the farmer via the smartphone, and the operation is triggered due to exceeding certain thresholds [45]. Data collection is similar to IoT in agriculture, it can be accomplished without Internet access. Traffic is predictable due to the limited number of sensors and switches.

4. Characterising SDN-Based IoT Architecture

4.1. The Architecture of SDN Based IoT Networks

As summarised in Table 3, IoT traffic messages are generated either periodically or occasionally by the IoT applications. This could be regarded as periodical access and random access in the SDN infrastructure.

The diversity of IoT applications demands a well-directed transmission route, or so-called user-centric services, to achieve a better performance. In order to process these requests separately, SDN slicing provides an effective solution. As the entire network is centrally controlled, the SDN controller could logically split a physical underlying network tunnel into multiple Virtual Tunnels (VTs). Each VT is independent and has its optimisation of a specific use case. Note that an issue in one VT will not impact other VTs [46]. As there will be various management policies over the same channel, SDN has an inherent advantage to do that due to its decoupled architecture [47]. To improve the efficiency of utilisation, the SDN controller also needs to generate and adjust flow entries for different services on the common physical tunnel. Meanwhile, unnecessary features are removed and essential factors are enhanced according to the service type [48]. To simplify, a basic data plane of SDN supporting IoT consists of three entities:

- IoT Data Aggregators (IoT-DAG): There are various IoT subscribers including attackers. A client collects data from IoT devices (wired or wireless sensors). These data could be originating from different applications. The client behaves as an upper level node accessing the SDN switches. In this article, we use Raspberry Pi (RPi) as the IoT-DAG, which is a mini-computer running an operating system similar to Linux. Although portable in terms of size, RPi is capable of processing data having graphical interface and could connect to the Internet via Ethernet cable or WiFi. Therefore, users can reprogram or configure it remotely, which is crucial to the IoT applications [49]. Another one is Odroid that supports Ubuntu and Android system with a more powerful RAM compared to other devices with the same size, it is even equipped with a heat sink [50]. These devices have some common points that meet IoT requirements: cheap, small size, easy-to-use, low energy consumption, open-source and adequate processing ability.
- SDN Switches: The support from a controller to the SDN switch is essential, although the switch can still work with its last configuration. As a legacy switch without a controller, SDN loses its power. Thus, the controller acts as a coordinator to instruct a switch and redirect different kinds of data to the destination via SDN. Zodiac-FX is the one we use in the test. It is a small OpenFlow switch that is designed to be used in the laboratory providing real traffic on the physical hardware. It has open-source firmware, simple setup procedure and CLI access. Zodiac-FX supports OpenFlow 1.0 and 1.3 and can obtain flow entries from controllers, such as Ryu [51]. This is a suitable choice to emulate SDN-based IoT networks to meet the aforementioned IoT requirements.
- IoT Data Servers (IoT-DS): Each IoT application has an independent server, which only processes its type of service. Both scheduled and random access share the same server. Google smart home is a typical random access service, it stores devices per room first and uses this information to create a database, called Home Graph (HG) [52]. Then, the Google Assistant (GA) on the server side will process the user's request according to the HG. Google Home collects command from the user and sends it to the GA. The GA then tries to recognise the request and find relevant devices in the HG. Finally, the GA executes the operation via Google Home. Another instance is Amazon Website Service (AWS) IoT; with the aid of a cloud network, it simplifies the connection and management of enormous amount of IoT devices supporting data collection and complex analytics for reacting to the real-time actions [53].

Table 3. Traffic nature of IoT applications.

IoT Application	Periodical Message	Event Trigger
Smart Home	✗	✓
Wearables	✗	✓
Connected Car	✗	✓
Industrial IoT	✗	✓
Smart City	✓	✓
Agriculture	✓	✓
Smart Retail	✗	✓
Smart Grid	✓	✓
Healthcare	✓	✓
Farming	✓	✓

Flow entries in the SDN switches are generally provisioned in two modes, reactive and proactive, as explained below.

- Reactive: Initially, there is no flow entry provisioned in the switch, except a default entry whose operation is to inquire the controller. Therefore, flow entries are generated according to the algorithm in the controller. It is fully flexible, yet the control plane is vulnerable in this scenario. As a mismatched customer packet can trigger a request to the controller, DDoS attacks could impact both the control and data plane.
- Proactive: Flow entries are preconfigured in the switch, the role of controller varies according to the default operation given by the administrator.
  - *Drop when mismatch:* No new flow entry is generated by the controller. Customer packets either follow the existing flow entries or be dropped. This kind of configuration sacrifices flexibility to higher security. As a host, the device is no longer capable of initialising a request to the controller. DDoS attacks from hosts can only take effect in the data plane.
  - *Ask controller when mismatch:* This is similar to reactive flow entries. However, DDoS attacks triggering on the controller could be more sensitive due to the preconfigured flow entries. Because most of the popular routes must have been covered by the proactive flow entries, there should not be many requests to the controller within a short period.

In order to explain the SDN-based IoT network under attack, Figure 1 depicts a basic scenario used for IoT applications under the SDN architecture:

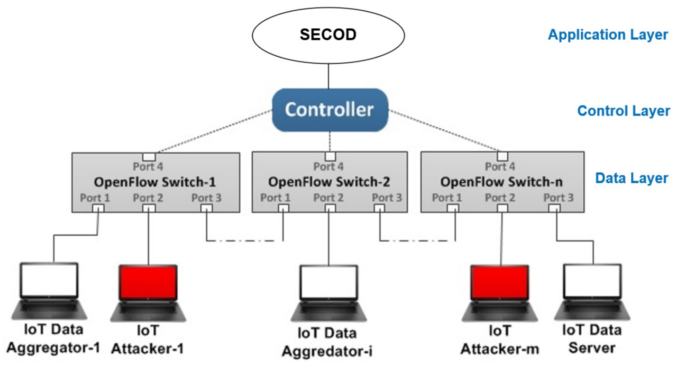


Figure 1. The basic topology of SDN-based IoT architecture under a DDoS attack.

- Each end-user device behaves as an IoT-gateway or IoT-DAG that aggregates data from IoT-sensors (wired or wireless) and sends the IoT traffic to related IoT-DS or IoT-DPC.
- The IoT attacker could be a compromised gateway or IoT sensor, which creates malicious traffic in the network. The traffic could be divided into two categories depending on the destination IP address: (i) varied IP address (targeted to the controller) and (ii) fixed IP address (targeted to the server).
- The IoT-DS is an IoT server processing both TCP and UDP traffic sent by IoT-gateways. The IoT-Servers can be any computing platform providing services for IoT data collection or management.

In the SDN network with multiple SDN switches (OpenFlow-based), a single attacker could not only impact the SDN-switch that is directly connected, but also other SDN switches under the same controller as well. This is because OpenFlow switches (OF-switch) forward these malicious packets to the controller by default. As the destination is unknown to the controller, it instructs the OF-switch to flood out all the ports except the incoming port (In-port in OpenFlow). When the next OF-switch receives this packet via switch-to-switch link, it repeats the same procedure as the first OF-switch. This operation spreads the vicious packets across the network.

4.2. The Proposed Algorithm for Mitigating DDoS

The proposed SECOD algorithm is running in the application layer of SDN architecture, it makes the rule to manage the network, as well as protects the network against DDoS attacks. Predefined policies in the SECOD are converted to flow entries via the controller, and then these policies are inserted to the OpenFlow switches to allow or block the access in the data plane. Note that SECOD and controller are running within the same device in our test, which means an internal link between the control and application layer is built within the same device.

The main workflow of SECOD is given in Figure 2. SECOD monitors the network and compares the real-time counter with threshold every fixed time period, and an excess of Packet_In messages could lead to DDoS detection or threshold update. Threshold update is used to handle burst traffic, while malicious traffic triggers detection function. In the DDoS detection, SECOD figures out where is the attack coming from—host or switch—so as to execute relevant mitigation process. The mitigation function keeps running until the attack is over, and then the threshold is reset. Finally, SECOD goes back to the monitor phase under normal state. For high-demand benign traffic, threshold will adjust dynamically to adapt to the environment. However, if the demand is too high to handle, the traffic will be labelled as malicious. Because it has reached the limit of device, either switch or controller, the traffic will be dropped automatically even without any algorithm to do so. Some key parameters in SECOD are defined in Table 4, and more details in [8,9]. There are five main modules:

1. **Monitor Function:** SECOD counts the number of *Packet_In* messages γ_{ij} generated from ith port in jth switch, as well as per IP address γ_{ijk}. Then, it sums up the total number of *Packet_In* messages per switch $\Sigma_j \gamma_{ij}$, and compares with ω. Once $\Sigma_j \gamma_{ij}$ exceeds ω, it means the controller is overloaded and the algorithm enters *DDoS Detection* module. Apart from ω, if one or multiple γ_{ij} are greater than α_{ij} and $\Sigma_j \gamma_{ij}$ is still under ω, then the algorithm enters *Threshold Update* and only updates the α_{ij}. Thus, each port of switch has a different threshold due to its past workload after the network running for some time.
2. **Threshold Update Function:** This function is mainly used to update α_{ij}. During network initialisation, α'_{ij} is used as a reference for DDoS detection, and later this threshold is updated to α_{ij} which reflects the network operation. α_{ij} keeps updating according to history statistics when the network is in normal state. If the network is just recovered from DDoS attack, α_{ij} is reset to α'_{ij}, as the statistics during DDoS attacks cannot reveal the normal behaviour of network.
3. **DDoS Detection:** This is the attack localisation phase before entering mitigation phase. The controller first sends a flow entry to all the suspicious switches, and this flow entry asks the switch to drop all the new flows that cannot match the existing flow table within a extremely short period. If the controller receives no *Packet_In* messages during that period, then the attack is originated from hosts; otherwise, the switch is compromised. Depending on the result, it goes to *Host/Switch Mitigation*.
4. **Host Mitigation:** SECOD checks γ_{ijk} and compares with ρ. If there is only one abnormal γ_{ijk}, the controller inserts a flow entry to switch j so as to block packets from the suspicious IP address. If there are multiple anomalous γ_{ijk}, the controller asks the switch to drop all the packets from that port. The flow entry to block packets has idle timeout enabled, so that it is removed as soon as the attack stops.
5. **Switch Mitigation:** As the switch is compromised, the controller is unable to manage that switch. Thus, SECOD instructs the controller not to process the *Packet_In* messages from that malicious switch, but to keep counting the number of requests from that switch. When its $\Sigma_j \gamma_{ij}$ is lower than ω, we regard it as DDoS attack is over, and then the flow entry which blocks the switch is removed.

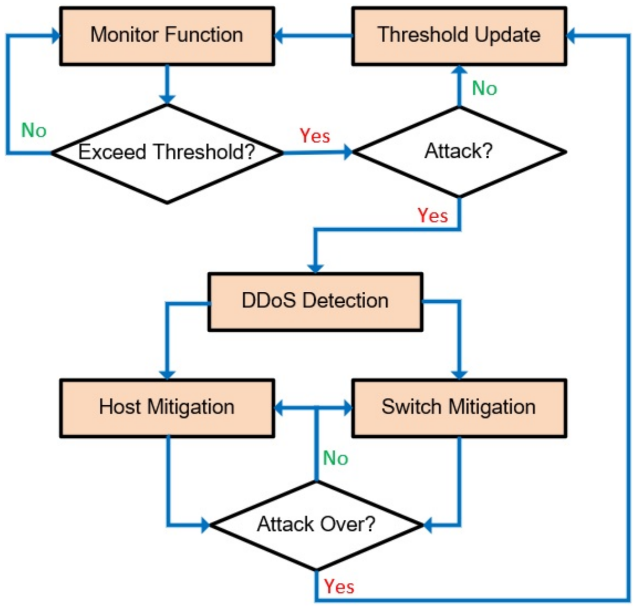


Figure 2. Workflow of SECOD algorithm.

Table 4. Parameters in the SECOD algorithm.

Symbol	**Definition**
i	Port ID on a switch, where i = 1, 2. . . n
j	Switch ID in the network, where j = 1, 2. . . m
k	IP address ID under a specific switch port, if an IP address is the first IP who generates a request towards the controller under a switch port, then k = 1 for this IP address under switch m port n.
γ	Count of *Packet_In* messages, where γ_{ijk} indicates the kth IP address from ith port in jth switch.
ω	The maximum number of *Packet_In* messages that is allowed from one switch within a time period. It is the switch threshold. As the target of DDoS attack is to exhaust resources, ω is defined based on the maximum system resources that users would like to assign to the switch for new flow processing, i.e., a switch can generate up to 1000 *Packet_In* messages per minute, the administrator can set ω to 500 per minute, which means the switch is allowed to use up to 50% of its resources to generate *Packet_In* messages. This value varies according to the application and customer requirement. ω is defined before a switch joins the network, and this threshold will not change over time.
α	The maximum number of *Packet_In* messages that is allowed from one switch port within a time period. It is the port threshold. The initial port threshold is α', it is defined by ω and the number of ports under a switch, i.e., a switch has 5 ports and ω is 500 per minute, then α' is 100 per minute. α will update according to the network behaviour.
ρ	The maximum number of *Packet_In* messages that is allowed from one IP address under a specific switch port within a time period. It is the IP threshold. ρ is also dependent on the performance of the switch, because a more powerful switch could allow a user to make more requests. This threshold is predefined by the administrator and will not change over time.

SECOD aims to mitigate DDoS attacks without involving extra components, such as middleware, and we try to automatically localise the source of attack, either from the host or switch, via the control plane. Through the validation in the SDN-based IoT network, we also verified its performance which will be discussed in Section 6.

4.3. Explanation of Key Operations in SECOD

In order to explain the operation of SECOD, we refer to Figure 1. Basically, SECOD uses the following techniques.

4.3.1. Blocking Ports

The controller receives requests from port-2 of switch-1, port-1 of switch-2 and port-1 of switch-3 when IoT attacker-1 is running. This means all the OF-switches within the network seem to have a subordinate IoT attacker from the controller′s point of view. If the controller filters packets by port, the connection between IoT-DAG that are under different switches may suffer from an interruption within a period. This period depends on the timeout value of the flow entry created by the controller. As IoT attacker-1 is the root cause of the chaos in the network, as long as packets from IoT attacker-1 are blocked, port-1 of switch-2 and switch-3 will lose the source of attack, so that network is able to recover connectivity soon. Nevertheless, this creates an out-of-service time period leading to multiple IoT-DAG disconnection in a large network. Therefore, an IP address-based filter is proposed to avoid this network outage, as connections between switches are always available during defence.

4.3.2. Keep Counting

In the SECOD algorithm [9], counters per IP address are running in the controller to monitor the behaviour in the network. Each IP address represents one or multiple residential users or business customers. Generally, the threshold of each IP address is different according to the service fee that is paid to the vendor. The vendor has various IP address pools for different level of services, so a single IP address must map to a specific service level for the sake of easy management. This means a single IP address should have a fixed threshold, which is decided by the available bandwidth and number of users. As a single IP address could trigger requests from multiple switches, once an IP address exceeds its threshold, a drop rule shall be deployed in all the switches under the same controller.

4.3.3. Controlling Packet_In

Another characteristic of the attack towards control plane is that most of Packet_In messages are Address Resolution Protocol (ARP). ARP is used to discover the MAC address associated with an IP address; it is essential to the data transmission. However, the Packet_In message generated by the user is usually non-ARP due to the existence of ARP cache in the user itself. In particular for Linux system, an ARP entry will not be erased immediately after a time period, it only updates the state (from REACHABLE to STALE) to wait for a further period. Once it is hit in the state of STALE, it changes back to REACHABLE, and an official ARP request is sent out 5 s later by default. Otherwise, it is removed after a long period without any hit [54]. In this case, most of the Packet_In messages already contain the MAC address of the destination. All they want is a flow entry in the OpenFlow switch. When the official ARP is generated after 5 s, as the flow entry is already created on the switch, there is no need to inquire the controller any more, which means normal Packet_In messages are generally non-ARP. Furthermore, the IP address on the user port could be a public address, which represents multiple private IP addresses. If a suspicious IP address is banned, all the normal users using this public address will lose connections to the network. Thus, filtering packets by both IP address and ARP is a better way to block DDoS attacks, while normal connections are still permitted.

For periodical IoT data, the configuration of flow entry can be predefined on the controller depends on the period to save system resources. Request and reply between the switch and controller are no longer iterating, because the Time-To-Live (TTL) value of the flow entry will be refreshed before timeout. For random IoT data, reducing TTL value could be a better choice due to its unpredictability.

5. DDoS Attacks Emulation on SDN-Based IoT Architecture

5.1. Effect of SECOD in the Simulation

Before validating SECOD in the real testbed, we first verify its performance using Mininet (Network emulator, available at: http://mininet.org/), and the simulation is running over Ubuntu 16.04 OS with Intel i5 CPU and 8 G RAM. The setup of the virtual network is shown in Figure 3. There are 30 switches (s1–s30) attached to the same controller, and each switch has two subordinate hosts, which means there are 60 hosts (h1–h60) in this network. The bandwidth of each link between switches and hosts is limited to 1 GB. iPerf tool (Traffic generator, available at: https://iperf.fr) is used to measure the available bandwidth from h1 to h60, so that we can compare the output when the network is normal, under attack and protected by SECOD. Here, h1 behaves as a benign user and h60 acts as a server. Among the rest 58 hosts, there are 10 IoT attackers, these compromised hosts generate malicious traffic to consume the network resources.

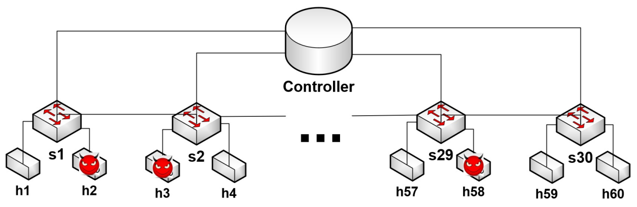


Figure 3. Network architecture in the Mininet.

We compare the available bandwidth during normal state and DDoS attack state to certify the performance of SECOD, each scenario has 10 runs and results are illustrated in Figure 4. When the network is in the normal state, the average available bandwidth is 953 Mbps, and this value drops to 631 Mbps under DDoS attacks. If SECOD is activated under DDoS attack, the average bandwidth is 939 Mbps. From the result, it can be seen that SECOD can protect the network against DDoS attacks under multiple attackers.

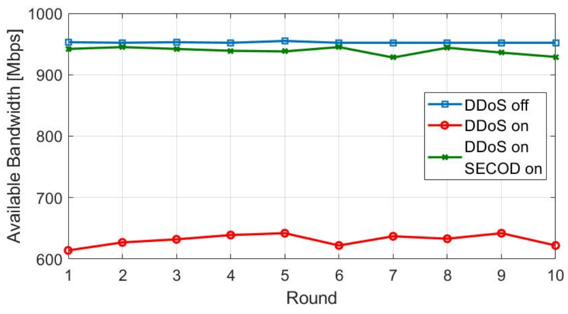


Figure 4. Comparison of available bandwidth.

In order to emulate IoT traffic in an SDN-based network, we replicate as close to a realistic IoT scenario as possible, described as follows.

5.2. SDN-Based IoT Testbed

The physical testbed consists of two IoT-DAG, two IoT Attackers, one IoT-DS, three OpenFlow switches and one SDN Controller as shown in Figure 5. In order to have a realistic setup for IoT applications, IoT-DAG-1, IoT-DAG-2, IoT Attackers, IoT-DS and SDN Controller use Raspberry Pi 3; this is because IoT devices are expected to be low cost, low battery consumption and low CPU capacity.

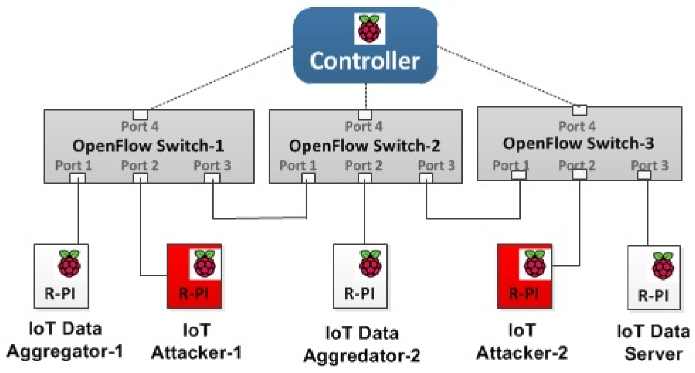


Figure 5. The SDN-based IoT architecture of the testbed used for the emulation of DDoS attacks.

The OpenFlow switch is Zodiac-FX, a low-cost simple device, which has 4 physical ports: port 4 for a connection between the controller and the switch, ports 1 to 3 for peer connections between switches. In order to manage multiple switches via a single controller, a traditional router is used to link the controller to all the three switches. IoT-DAG tries to send data to the IoT server, while IoT attackers will try to impact the transmission from IoT-DAG to the server by depleting bandwidth resources. IoT attackers have been deployed under two switches as shown in Figure 5, so that both IoT-DAG-1 and IoT-DAG-2 will be impacted. iPerf is adopted to generate traffic in the network as explained in the next subsection, where two scenarios are considered:

- *In scenario-1,* IoT-DAG-1 generates periodical and random TCP or UDP traffic one after another to verify the nature of each traffic type in the network.
- *In scenario-2,* IoT-DAG-1 generates periodical UDP and periodical TCP traffic, and IoT-DAG-2 generates random UDP and random TCP traffic.

The IoT-DS is responsible for collecting TCP and UDP traffic and generating test results for each scenario. In order to distinguish periodical and random TCP/UDP traffic on the server side, different port numbers have been assigned to these four types of traffic.

iPerf is also used on the IoT attacker to emulate the DDoS attack in the testbed. It is worth noting that we are using UDP flooding attack against the control plane. The attack frequency can be modified through the interval of transmission. The attacker sends a flow to a new destination each time to trigger a request to the controller. Thus, as long as it produces a huge amount of requests within a short period, the controller has to consume enormous resources to process these fake packets so that DDoS attacks can take effect on the controller. Meanwhile, the switch has to process these requests as well, so that it may not be able to process normal packets. Therefore, the impact is on both the control and data planes. Each experiment runs 10 times for 20 min.

We use the topology in Figure 1 to verify the behaviour of each scenario under normal circumstance, under DDoS attacks and running SECOD to resist DDoS attacks.

5.3. IoT Traffic Emulation

We use iPerf synthetic traffic for network measurements in order to create TCP and UDP traffic between IoT users and the IoT-DS. The UDP traffic is generally preferred in wireless communication, such as wireless sensors, because it is connectionless and has less overhead, which means lower power consumption and a longer lifespan. Another reason is that UDP is faster than TCP, thus making it an ideal carrier for real-time streaming for IoT applications, whereas most of the applications, especially for interaction with the website or other people, such as email, adopt TCP as the solution. Because the transmission is connection-oriented and guaranteed, which ensures the reliability of message you received, TCP traffic is not the preferred for majority of IoT applications. Thus, we divide traffic into four categories:

- **Periodical UDP Traffic:** IoT-DAG transmits 100 bytes UDP packets every 10 s (α = 10 s). It is usually employed in smart city applications to collect information from sensors every period.
- **Periodical TCP Traffic:** IoT-DAG transmits TCP traffic with maximum available bandwidth every 60 s (β = 60 s). It is usually employed in the smart grid to read meters remotely every period.
- **Random UDP Traffic:** IoT-DAG transmits 100 UDP bytes every 60–120 s (γ = 60 s, δ = 120 s). It usually happens on the wireless sensor devices when an event is triggered.
- **Random TCP Traffic:** IoT-DAG transmits TCP traffic with maximum available bandwidth every 60–120 s (ϵ = 60 s, ζ = 120 s). This usually happens on the personal device trying to connect to IoT applications, such as remote control of smart home.

iPerf is running on both the sender (IoT-DAG), as well as the receiver (IoT-DS) to test network performance by generating TCP/UDP traffic. For UDP traffic, packet loss is the metric used to measure the performance, and available bandwidth is the metric for TCP traffic. These four types of traffic are implemented at the same time under different environments to emulate a real IoT traffic model, while showcasing the impact of DDoS attack and effect of SECOD.

5.4. Flow Operation Setup

As only new packets generate Packet_In messages, theoretically the number of Packet_In message is related to the TTL of flow entries. There are two kinds of timeout: idle_timeout and hard_timeout. Idle_timeout removes flow entry if the entry has not been used for that period, while hard_timeout removes flow entry after that period, regardless of any other condition. Originally, we set the idle_timeout to 10 s, and hard_timeout to 0 which is disabled.

If the period of UDP traffic is α, the period of TCP traffic is β, and the random period of UDP traffic ranges from γ to δ ($\gamma < \delta$), the random period of TCP traffic is between ϵ and ζ ($\epsilon < \zeta$), we have a group of time slots which contains from α to ζ. Then, we try different idle_timeout (η) and hard_timeout (θ) value. Enabling idle_timeout is able to avoid sending requests from the switch to the controller repeatedly, so that the resource of control plane and the bandwidth between control and data plane are saved, while enabling hard_timeout aims to release the resource of route table on the switch especially during DDoS attacks.

5.5. SECOD Operations

SECOD uses statistics of Packet_In messages within a predefined duration to decide if the control or data plane is suffering from DoS attacks. Thresholds on the switch and controller are predefined manually according to the network requirement and can be adjusted according to the real-time network operations. The definition of monitoring duration relies on the allocated system resource. That is, if administrator assigns more resources to monitoring and detection, the duration can be set to a small period to make early detection, though at the cost of consuming more resources. Alternatively, the detection time can be made longer, thereby saving resources, though there is a sacrifice. Once the attacker is localised, its incoming packets will be dropped based on its IP address. SECOD mainly operates based on the traffic behaviour according to the following principles.

- *In order to stop the malicious packets,* SECOD blocks suspicious requests at the first node, which is the ingress port of SDN. A filter based on the IP address tries to find the source of attack and drop its packets once received on the ingress SDN switch. The reason that we choose IP address instead of MAC address as the metric is because the MAC address of a packet will always be the MAC address of the port that is directly connected to the SDN switch. However, the IP address could be the original address of the sender (or address of a network address translation router), so filtering by IP will allow a lower rate of false positives. Note that if the attacker is able to modify source IP address, then SECOD will have to block the whole port of SDN switch, because if these malicious packets already impact on the normal users attached to the same SDN switch port, it can hardly figure out where is the source of attack if the hacker is sophisticated.

- *For IoT traffic,* the time period of periodical traffic is normally predefined, such as smart grids. Even for random traffic, the behaviour is predictable within an acceptable range, such as healthcare, it would not be infinite demand as in the traditional network. Random traffic that triggered by the event could be burst requests due to emergency situation, but they shall still under control because of the limit of original application. This feature makes SDN-based IoT network more efficient in detecting DDoS attacks, because the threshold of detection can be set more precisely according to different applications. Moreover, IoT applications can easily reach the user to inform any event or change. This helps the server to send out an alert to the abnormal device to ask for the involvement of user to check if the device is still working properly. Otherwise, it is separated from the network as a result of infection.

Based on the reason above, DDoS detection in the SDN-based IoT network has advantages in the early detection, because the definition of threshold can be strict by reason of limited options of IoT requests. In some applications, such as connected car, the active time is generally during daytime, which means the network could cut down relevant resources and reduce threshold as well. Thus, SECOD can be modified to better adapt to IoT network, especially in early detection, we decide to use counter based time measurement to determine DDoS attacks. We measure the duration of threshold excess to detect DDoS attacks, if the duration is long enough, it is normal traffic. However, if the duration is shorter than the predefined value, it is regarded as under a DDoS attack. The predefined value depends on the service requirement and network performance. To compare the detection time in SECOD with SECOD Modified (SECOM), we define that the monitoring period is t seconds, attack rate is r packets/second, threshold is h packets/monitoring period, attack begins at the sth second ($s < t$) of the monitoring period and detection time period is d seconds. Then, the detection time in SECOD is

$$\begin{cases} d = t - s, & r * (t - s) \geq h, \\ d = 2t - s, & r * (t - s) < h, \end{cases} \tag{1}$$

While in the IoT scenario, if we apply new threshold definition that checks the length of time consumed to have a specific number of packets, the detection time is fixed to

$$d = \frac{h}{r} \tag{2}$$

When the attack rate is fast enough, the detection time is $(t - s)$ seconds refer to the top formula in (1), and the smallest possible value is the one in (2), in which the statistic happens to hit the threshold value h at the end of a monitoring period. Furthermore, if the attack rate is not fast, the detection time is more than a monitoring period t, because once the statistic is lower than h, it has to wait till the end of next period. Thus, early detection is much easier to achieve in the SDN-based IoT traffic.

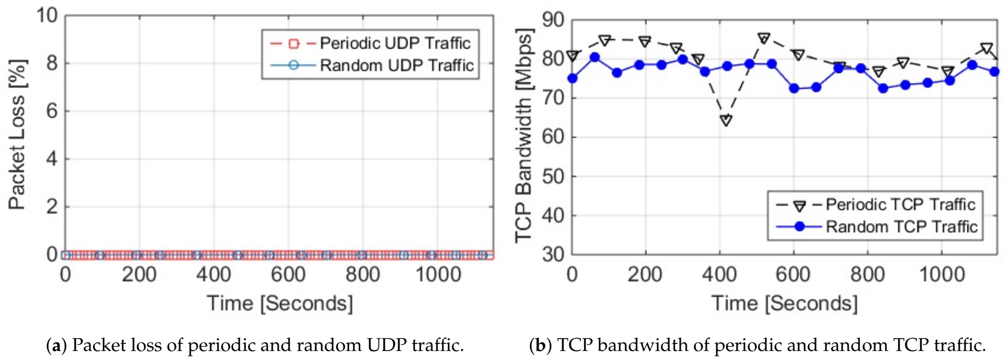
6. Results and Discussions

In this section, we report network performance under normal and DDoS attacks scenarios, as well as the difference in detection time using SECOD and SECOM.

6.1. Network Performance Behaviour in Normal Operations

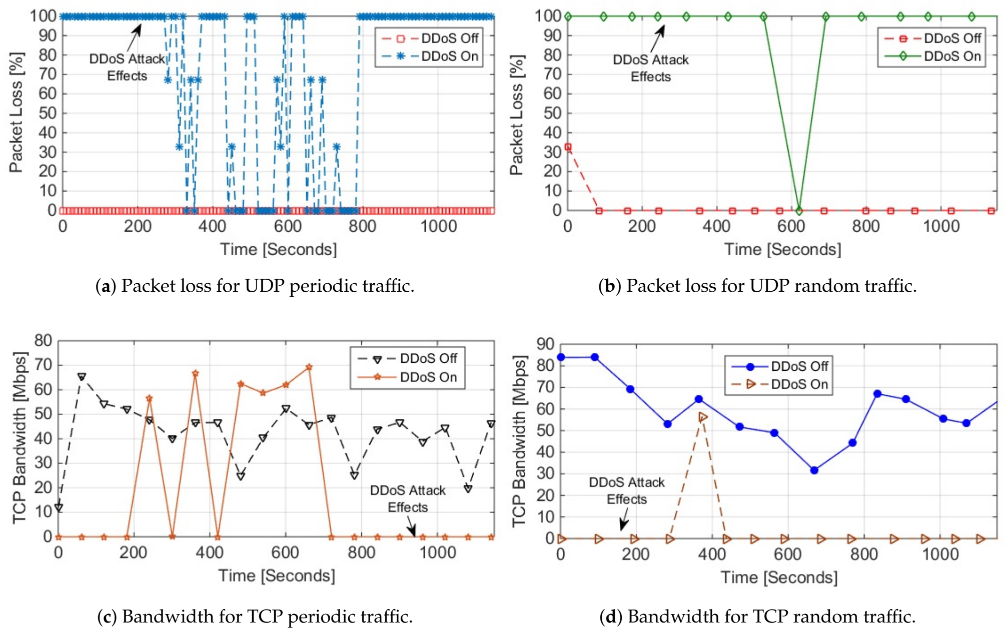
We first analyse the network performance under normal operations using periodical and random UDP and TCP traffic depicted in Figure 6, in which various types of requests happen under the same network, respectively. It is noted that TCP traffic tries to utilise the maximum bandwidth of the network, which leads to a higher packet loss ratio in the UDP traffic transmitting at the same time, because the tunnel deteriorates for UDP traffic.

Even for TCP traffic itself, if a random TCP traffic comes with a periodical TCP traffic, they will have to share the bandwidth. This phenomenon is obvious in the first time slot of traffic which can be seen in Figure 7c,d with DDoS off, as both of the TCP traffic begin to transmit in the first second. As the random TCP traffic already utilises over 80 Mbps resource, the periodic TCP traffic has to compromise and occupy much less bandwidth than normal. Meanwhile, the random UDP traffic also suffers from the lack of bandwidth resource. However, the network behaves normal, and the packet loss and bandwidth are as expected. 100% and 99% of TCP and UDP requests have been successfully transmitted to the destination, respectively. This is mainly due to the sufficient bandwidth and the small-sized IoT traffic that we emulated. We also run Wireshark, which is a network protocol analyser, on the RPis to monitor packet delivery rate. The packet delivery rate is the ratio of received packets to the total number of sent packets. This average rate of TCP packets is around 97%, and for UDP is 99%. A TCP or UDP request usually consists of multiple packets; we consider a request to be successfully delivered only if all the relevant packets are received. Although TCP packets have a lower delivery rate than UDP in the test, the TCP request is guaranteed to be sent to the destination, thanks to its re-attempt feature.



(**a**) Packet loss of periodic and random UDP traffic. (**b**) TCP bandwidth of periodic and random TCP traffic.

Figure 6. SDN-based IoT network performance under normal operation conditions.



(**a**) Packet loss for UDP periodic traffic. (**b**) Packet loss for UDP random traffic.

(**c**) Bandwidth for TCP periodic traffic. (**d**) Bandwidth for TCP random traffic.

Figure 7. SDN-based IoT network performance comparison under normal and DDoS attack conditions.

6.2. Network Performance Behaviour under DDoS Attacks

Now, we analyse the network under DDoS attack conditions. For emulating a DDoS attack, IoT-DAG-1 transmits TCP and UDP traffic to the server, while IoT-DAG-2 transmits random TCP and UDP traffic to the server. The result is depicted in Figure 7. As observed in the results, during the DDoS attack with only two attackers, both TCP and UDP traffic suffer from the impact. For TCP traffic, the tunnel is out-of-service at most of the test period. For UDP traffic, even though it only requires extremely low bandwidth to transmit, a large proportion of traffic is discarded. Only 21% of periodic and 14% of random TCP requests, and 4% of periodic and 1% of random UDP requests are successfully transmitted to the far end. From the result in Wireshark, a RPi tries to resolve the far end MAC address using ARP protocol prior to packet forwarding, and it gives up after six attempts if fails to learn the destination MAC address. Therefore, packet delivery rate is not considered here, because most of the TCP/UDP requests are not even sent out from RPis during DDoS attacks due to the absence of MAC address information. It is important to note that random transmissions are more affected during a DDoS attack, so we can conclude that IoT event trigger applications are more vulnerable to attacks than periodical ones.

6.3. Network Performance Behaviour under DDoS Attacks with Different Idle_timeout

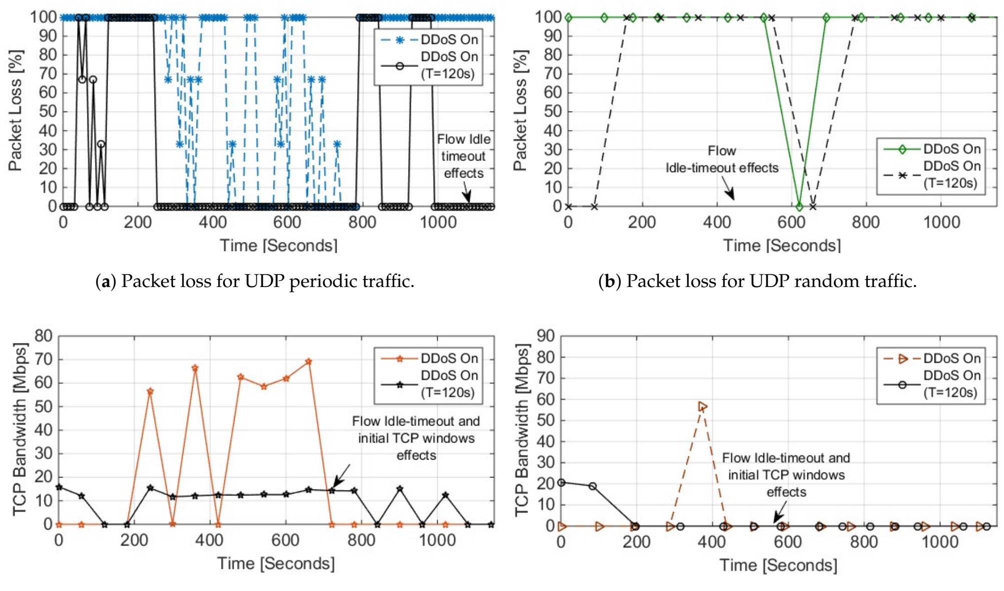
In the next set of experiment, we change some important metrics of OpenFlow (Idle_timeout) and TCP operations (initial TCP windows) in order to investigate their effect. The default idle_timeout value is 10 s, while the transmission interval of TCP traffic is over 10 s, i.e., 60 s for periodical TCP traffic. Thus, an IoT-DAG has to enquire the controller every time a forwarding request is received, because the existing flow entry already expired before a new request comes in. As the impact on the traffic could be resulted from the switch, the controller, or the link between them, we increase idle_timeout value to 120 s (the biggest transmission interval is 120 s in this test) so that the switch has no need to contact the controller once a flow entry is made. Furthermore, we adjust the window size of TCP traffic to limit the bandwidth in each transmission. Here, we add "-w 5000" in the iPerf tool configuration to reduce the bandwidth of a single TCP traffic to around 16 Mbps. Thus, the switch is still capable of handling DDoS attacks if performance improves. Otherwise, the switch is the root cause of network deterioration.

The result is depicted in Figure 8. We can see that successful transmission ratio of periodical TCP/UDP traffic has remarkably increased after the extension of timeout value, which proves that the switch is still working under DDoS attacks. However, from the output of random UDP and TCP traffic, it shows that (i) packet loss in UDP is still very high and (ii) the bandwidth of TCP is not the reason for inferior random TCP performance, because the network has adequate bandwidth resource to transit multiple TCP traffic simultaneously. Therefore, we conclude that metrics of OpenFlow and TCP connections can play a crucial part in keeping at least a minimum performance under a DDoS attack.

In order to find out the importance of flow entries during a DDoS attack, we start attack after a first successful transmission of each type of traffic. Therefore, switches have the required flow entry already, they only need to send out traffic according to the entry. However, the result is even worse than the one in Figure 8, which means the performance of switches is unstable under DDoS attacks.

6.4. Network Performance Behaviour under DDoS Attacks Using a Powerful Controller

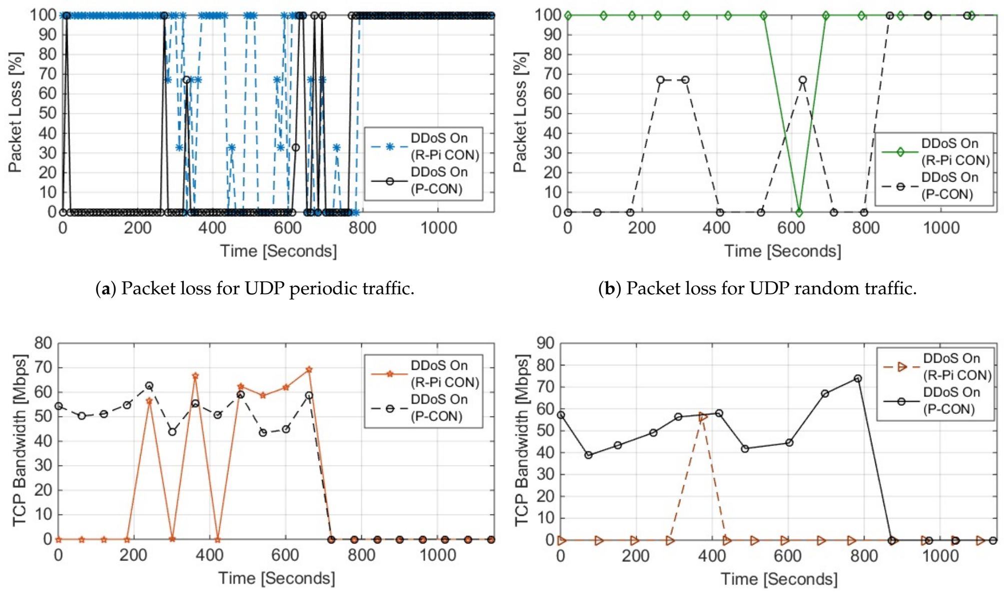
Next, we use a powerful laptop (i5-6300U 2.4GHz CPU, 8G-RAM, Ubuntu OS) to work as a powerful controller (P-CON) to see if the network performance rises with a superior controller. From the results in Figure 9, we can see that a stronger controller, which has additional control plane resources, improves a little in the request processing even though the hardware upgrade is tremendous. As a result of these outputs, link saturation between the switch (data plane) and the controller (control plane) is the bottleneck of this network. It should be noted that the whole network may become stuck, either on switch or controller, during DDoS attacks. This results in 100% packet loss in UDP traffic, as well as 0 Mbps bandwidth in TCP traffic. From a 10-round test run, the probability of a switch gets stuck is 40%, while the probability of a controller turns into stuck is 10%. Thus, we conclude that the resources of control plane (controllers) are unable to turn the tables on the compromise of data plane (switches) resources during DDoS attacks.



(**a**) Packet loss for UDP periodic traffic. (**b**) Packet loss for UDP random traffic.

(**c**) Bandwidth for TCP periodic traffic. (**d**) Bandwidth for TCP random traffic.

Figure 8. SDN-based IoT network performance under DDoS attack conditions with Idle_Timeout (120 s) and initial TCP window size (5000).



(**a**) Packet loss for UDP periodic traffic. (**b**) Packet loss for UDP random traffic.

(**c**) Bandwidth for TCP periodic traffic. (**d**) Bandwidth for TCP random traffic.

Figure 9. Performance of SDN-based IoT network under DDoS attack conditions with P-CON and RPi Controller (R-PI CON).

6.5. Network Performance Behaviour under DDoS Attacks with SECOD Activated

Finally, we activated the SECOD algorithm in order to investigate the network performance. The Receiver Operating Characteristic (ROC) curve is shown in Figure 10; both normal and attack states have been repeated for 100 times, the predefined threshold of a port is 100, where positive is normal state in the test. From the results in Figure 11, it can be seen that both TCP and UDP traffic are able to reach the IoT-DS during a DDoS attack. With the aid of SECOD, 100% of TCP requests and 99% of periodic UDP requests are received by the receiver; however, 27% of random UDP requests are dropped in the middle of transmission. The average delivery rate of TCP packets and periodic UDP packets are 95% and 99%, respectively, which is almost the same as normal state. For random UDP packets, the delivery rate is 76%. The performance of SECOD in a large network topology is verified in the Mininet, it is working properly with more users, more attackers and a higher bandwidth than in the hardware emulation. Although the physical testbed consists of fewer devices, it poses more issues, such as the unexpected output of UDP traffic, from real implementation. The available bandwidth in the Mininet in Figure 4 is relatively more smooth than in the real testbed in Figure 11, because the network environment in the simulator is stable, and all the network resources are managed as a whole by the laptop to balance and meet the requirement from each component. Nevertheless, the results of running SECOD in both the simulator and physical devices build confidence in protecting a large real network. Note that packet loss happens frequently in the random UDP traffic. However, the performance of periodical UDP traffic is perfect. Furthermore, we find that random traffic has worse performance than periodical traffic during DDoS attack. Even though periodical traffic has to pass through a longer route to the server (from IoT-DAG-1 to server), the successful transmission of periodical traffic outweighs random traffic. This may because of the timeout configuration in the SECOD algorithm. Thus, we decided to tune some of the parameters of SECOD in order to improve our algorithm as explained in the following.

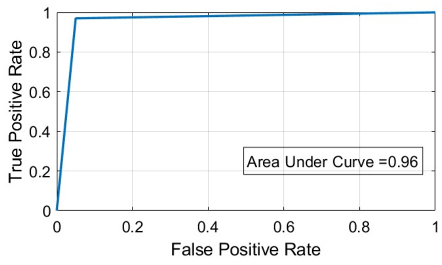


Figure 10. The Receiver Operating Characteristic (ROC) curve of SECOD.

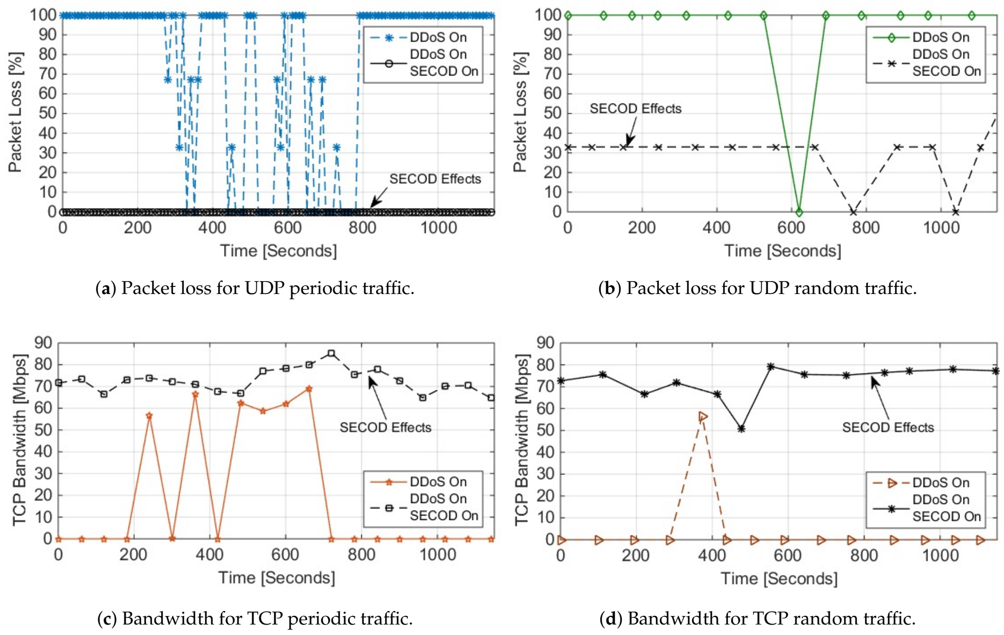


Figure 11. Performance of SDN-based IoT network under DDoS attack conditions with SECOD activated to detect and resist DDoS attacks.

Improvements of SECOD for SDN-Based IoT Networks

To explain the improvement of detection time in the SECOD modified (SECOM), we set the same threshold value to 100 requests per port per monitoring period in the SECOD and per IP address in the SECOM, because if this value is set too small, it is so hard to catch the detection time in the SECOM. Meanwhile, as SECOD makes a decision using switch threshold, we set the switch threshold to 500 requests per switch per monitoring period so that the attacker could trigger the alarm after SECOD dynamically adjust the threshold per port. Then, we launch IoT Attacker-1 and -2 in Figure 5 and set the interval of malicious packets transmission to 1 ms (normally have 60 Packet_In message requests per second due to the hardware performance limit of Zodaic-FX switch). With 10 rounds test, we illustrate the result in Figure 12.

By default, the monitoring period of SECOD is 10 s, while SECOM only needs to check the time duration when the counter of Packet_In requests hit 500. The average detection time of SECOD is 33.58 s, while SECOM only needs 1.65 s. In Table 5, we compare the detection time of SECOD and SECOM with some existing solutions against DDoS attacks in SDN-based IoT networks. Note that this is just a high-level comparison, as the detection time may vary based on the types of attack, test scenarios or testbed hardware. If we reduce the monitoring period of SECOD to 3 s, the average detection time becomes 9.65 s. Therefore, we find SECOD usually requires triple times of monitoring period in this test bed and threshold combination. This could come from the dynamic update of the threshold per port, and the stuck state of switch from time to time during monitoring period. Thus, the longer monitoring period in the SECOD, the more fluctuation in the detection time. However, the detection time for SECOM is always less than 2 s and quite stable as well. If reduce the monitoring period of SECOD further to 1 or 2 s, the attack frequency cannot hit the threshold trigger due to the hardware limit of Zodiac-FX switches.

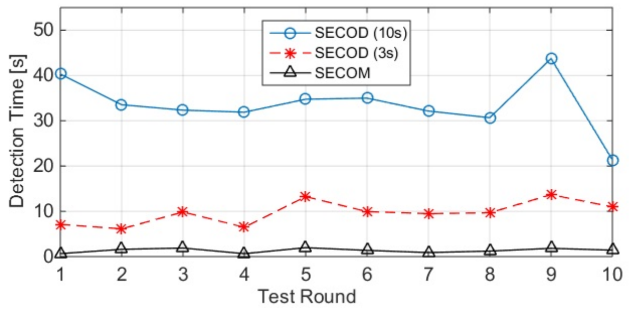


Figure 12. Comparison of detection time using both SECOD (with different monitoring durations) and SECOM algorithms.

Table 5. Comparison of Average Detection Time.

Methodology	Average Detection Time (s)
ESESID [27]	6.02
FlowJustifier [28]	<20
SDIoT-DDoS-DA [55]	5
Yu et al. [56]	3
SECOD	9.65
SECOM	1.65

Both SECOD and SECOM provide lightweight countermeasures against DDoS attacks, they are easy to implement and maintain, and no extra device is required. The dynamic threshold allows auto-adjustments to the network behaviours, so that legitimate burst traffic can pass through without triggering alerts. Once a DDoS attack is detected, the root cause of attack can also be localised so as to give tips to the administrator at once. As SECOM runs more checks than SECOD in anomaly detection, which means a higher resource consumption, the network operator can deploy SECOM if early detection has a high priority, and resources, such as power, is not a problem. Moreover, SECOD is preferred if the long lifespan of device outweighs detection time.

7. Conclusions

In this article, we test and analyse the SECOD algorithm to protect SDN-based IoT network in the real testbed. As IoT has a vast number of applications across different areas, this makes IoT prone to DDoS attacks, which have a huge impact on the SDN-based IoT networks. In order to define the metric of DDoS attacks in the IoT network, studying different features of IoT traffic is the first step. Another challenge is to find out the bottleneck of defence in particular if the control plane or data plane suffers. Periodic and random IoT traffic are emulated in the real test bed, and the results show the following.

- Random traffic (UDP or TCP) is more affected during DDoS attacks. Therefore, we can conclude that IoT event-trigger applications are more vulnerable to attacks than periodical ones.
- Metrics of OpenFlow and TCP connections can play an essential part in keeping at least a minimum performance under a DDoS attack. Therefore, this metric can be tuned based on the characteristics of IoT traffic that the network is supplying.
- Resources of control plane (controllers) play a minor role during DDoS attacks, if the data plane (switches) resources are already compromised.
- The SECOD algorithm is able to block DDoS attacks in the SDN-based IoT network, where traffic behaves like normal circumstances even under DDoS attacks; however, the performance in protecting random UDP traffic still requires improvements.

For the sake of the feature of IoT applications, DDoS attacks can be detected much easier than in the pure traditional or SDN networks, especially on the dedicated IoT network slices. We demonstrate that algorithms for DDoS detection and defence like SECOD can be regulated and improved, similar to SECOM, in order to better adapt to the predictable dynamics of IoT traffic. SECOD and SECOM provide solutions for administrators under different environments; SECOD is more energy efficient than SECOM, but SECOM has a quicker response to DDoS attacks. Both of them are easy to implement and maintain.

The limitation of the proposed algorithm is that it is only able to work under reactive mode, because it relies on the Packet_In message. Furthermore, if the attacker changes or spoofs source IP address, it will be difficult to track. It is also worth noting that if we consider a network topology with core, aggregation and access layers, SECOD best suits the access layer as a lightweight solution against DDoS attacks, while for aggregation or core layer deployment, more features and modifications are required. As future work, we will study the behaviour of IoT subscribers to refine the definition of threshold in different applications, and verify the output with real IoT devices and traffic based on smart city applications.

Author Contributions: Conceptualization, S.W., K.G. and K.S.; methodology, S.W. and K.G.; software, S.W.; formal analysis, S.W.; investigation, S.W.; resources, P.Z.; data curation, S.W.; writing—original draft preparation, S.W.; writing—review and editing, K.G., K.S., M.R.A. and G.R.; supervision, K.G. and K.S. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data presented in this study are available on request from the corresponding author. The data are not publicly available due to privacy.

Acknowledgments: Song Wang is supported by the Australian Government Research Training Program Scholarship.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

SDN	Software-Defined Networking
IoT	Internet of Things
DDoS	Distributed Denial of Service
RPi	Raspberry Pi
IoT-DAG	IoT Data Aggregator
IoT-DS	IoT Data Server
IoT-DPC	IoT Data Platform Collector
ARP	Address Resolution Protocol
P-CON	Powerful Controller
R-PI CON	Raspberry Pi Controller
ROC	Receiver Operating Characteristic

References

1. Open Networking Foundation. *OpenFlow Switch Specification*; ONF TS-024, v.1.4.1; Open Networking Foundation: Menlo Park, CA, USA, 2015.
2. Dargahi, T.; Caponi, A.; Ambrosin, M.; Bianchi, G.; Conti, M. A Survey on the Security of Stateful SDN Data Planes. *IEEE Commun. Surv. Tutor.* **2017**, *19*, 1701–1725. [CrossRef]
3. Lenka, R.K.; Rath, A.K.; Tan, Z.; Sharma, S.; Puthal, D.; Simha, N.V.R.; Prasad, M.; Raja, R.; Tripathi, S.S. Building Scalable Cyber-Physical-Social Networking Infrastructure Using IoT and Low Power Sensors. *IEEE Access* **2018**, *6*, 30162–30173. [CrossRef]
4. Akpakwu, G.A.; Silva, B.J.; Hancke, G.P.; Abu-Mahfouz, A.M. A Survey on 5G Networks for the Internet of Things: Communication Technologies and Challenges. *IEEE Access* **2018**, *6*, 3619–3647. [CrossRef]
5. Galeano-Brajones, J.; Carmona-Murillo, J.; Valenzuela-Valdés, J.F.; Luna-Valero, F. Detection and Mitigation of DoS and DDoS Attacks in IoT-Based Stateful SDN: An Experimental Approach. *Sensors* **2020**, *20*, 816. [CrossRef] [PubMed]
6. Singh, J.; Behal, S. Detection and mitigation of DDoS attacks in SDN: A comprehensive review, research challenges and future directions. *Comput. Sci. Rev.* **2020**, *37*, 100279. [CrossRef]
7. Valdivieso Caraguay, Á.L.; Benito Peral, A.; Barona López, L.I.; García Villalba, L.J. SDN: Evolution and opportunities in the development IoT applications. *J. Distrib. Sens. Netw.* **2014**, *10*, 735142. [CrossRef]
8. Wang, S.; Gomez, K.; Kandeepan, S. SECO: SDN sEcure COntroller Algorithm for Detecting and Defending Denial-of-Service Attacks. In Proceedings of the 2017 5th International Conference on Information and Communication Technology (ICoIC7), Melaka, Malaysia, 17–19 May 2017.
9. Wang, S.; Chandrasekharan, S.; Gomez, K.; Kandeepan, S.; Al-Hourani, A.; Asghar, M.R.; Russello, G.; Zanna, P. SECOD: SDN sEcure control and data plane algorithm for detecting and defending against DoS attacks. In Proceedings of the NOMS 2018—2018 IEEE/IFIP Network Operations and Management Symposium, Taipei, Taiwan, 23–27 April 2018; pp. 1–5.
10. Sahay, R.; Meng, W.; Jensen, C.D. The application of Software Defined Networking on securing computer networks: A survey. *J. Netw. Comput. Appl.* **2019**, *131*, 89–108. [CrossRef]
11. Dayal, N.; Srivastava, S. Analyzing behavior of DDoS attacks to identify DDoS detection features in SDN. In Proceedings of the 2017 9th International Conference on Communication Systems and Networks (COMSNETS), Bengaluru, India, 4–8 January 2017; pp. 274–281. [CrossRef]
12. Dao, N.N.; Park, J.; Park, M.; Cho, S. A feasible method to combat against DDoS attack in SDN network. In Proceedings of the 2015 International Conference on Information Networking (ICOIN), Siem Reap, Cambodia, 12–14 January 2015; pp. 309–311. [CrossRef]
13. Mousavi, S.M.; St-Hilaire, M. Early detection of DDoS attacks against SDN controllers. In Proceedings of the 2015 International Conference on Computing, Networking and Communications (ICNC), Anaheim, CA, USA, 16–19 February 2015; pp. 77–81. [CrossRef]
14. Dong, P.; Du, X.; Zhang, H.; Xu, T. A detection method for a novel DDoS attack against SDN controllers by vast new low-traffic flows. In Proceedings of the 2016 IEEE International Conference on Communications (ICC), Kuala Lumpur, Malaysia, 23–27 May 2016; pp. 1–6. [CrossRef]
15. Yan, Q.; Gong, Q.; Yu, F.R. Effective software-defined networking controller scheduling method to mitigate DDoS attacks. *Electron. Lett.* **2017**, *53*, 469–471. [CrossRef]
16. Dharma, N.I.G.; Muthohar, M.F.; Prayuda, J.D.A.; Priagung, K.; Choi, D. Time-based DDoS detection and mitigation for SDN controller. In Proceedings of the 2015 17th Asia-Pacific Network Operations and Management Symposium (APNOMS), Busan, Korea, 19–21 August 2015; pp. 550–553. [CrossRef]
17. Shoeb, A.; Chithralekha, T. Resource management of switches and Controller during saturation time to avoid DDoS in SDN. In Proceedings of the 2016 IEEE International Conference on Engineering and Technology (ICETECH), Coimbatore, India, 17–18 March 2016; pp. 152–157. [CrossRef]

18. Xiao, P.; Li, Z.; Qi, H.; Qu, W.; Yu, H. An Efficient DDoS Detection with Bloom Filter in SDN. In Proceedings of the 2016 IEEE Trustcom/BigDataSE/ISPA, Tianjin, China, 23–26 August 2016; pp. 1–6. [CrossRef]
19. RT, K.; Selvi, S.T.; Govindarajan, K. DDoS detection and analysis in SDN-based environment using support vector machine classifier. In Proceedings of the 2014 Sixth International Conference on Advanced Computing (ICoAC), Chennai, India, 17–19 December 2014; pp. 205–210.
20. T.Phan.; Bao, N.; Park, M. A Novel Hybrid Flow-Based Handler with DDoS Attacks in Software-Defined Networking. In Proceedings of the IEEE Conferences on UIC/ATC/ScalCom/CBDCom/IoP/SmartWorld, Toulouse, France, 18–21 July 2016; pp. 350–357.
21. Lim, S.; Ha, J.; Kim, H.; Kim, Y.; Yang, S. A SDN-oriented DDoS blocking scheme for botnet-based attacks. In Proceedings of the Conference on Ubiquitous and Future Networks, Shanghai, China, 8–11 July 2014; pp. 63–68. [CrossRef]
22. Chin, T.; Mountrouidou, X.; Li, X.; Xiong, K. An SDN-supported collaborative approach for DDoS flooding detection and containment. In Proceedings of the IEEE Military Communications Conference, Tampa, FL, USA, 26–28 October 2015; pp. 659–664. [CrossRef]
23. Macedo, R.; de Castro, R.; Santos, A.; Ghamri-Doudane, Y.; Nogueira, M. Self-Organized SDN Controller Cluster Conformations against DDoS Attacks Effects. In Proceedings of the IEEE Global Communications Conference, Washington, DC, USA, 4–8 December 2016; pp. 1–6. [CrossRef]
24. Hameed, S.; Khan, H.A. Leveraging SDN for collaborative DDoS mitigation. In Proceedings of the 2017 International Conference on Networked Systems, Gottingen, Germany, 13–16 March 2017; pp. 1–6. [CrossRef]
25. Sahay, R.; Blanc, G.; Zhang, Z.; Debar, H. ArOMA: An SDN based autonomic DDoS mitigation framework. *Comput. Secur.* **2017**, *70*, 482–499. [CrossRef]
26. Tortonesi, M.; Michaelis, J.; Morelli, A.; Suri, N.; Baker, M.A. SPF: An SDN-based middleware solution to mitigate the IoT information explosion. In Proceedings of the IEEE Symposium on Computers and Communication, Messina, Italy, 27–30 June 2016; pp. 435–442. [CrossRef]
27. Özçelik, M.; Chalabianloo, N.; Gür, G. Software-defined edge defense against IoT-based DDoS. In Proceedings of the 2017 IEEE International Conference on Computer and Information Technology (CIT), Helsinki, Finland, 21–23 August 2017; pp. 308–313.
28. Sarwar, M.A.; Hussain, M.; Anwar, M.U.; Ahmad, M. FlowJustifier: An optimized trust-based request prioritization approach for mitigation of SDN controller DDoS attacks in the IoT paradigm. In Proceedings of the 3rd International Conference on Future Networks and Distributed Systems, Paris, France, 1–2 July 2019; pp. 1–9.
29. Ravi, N.; Shalinie, S.M. Learning-driven detection and mitigation of DDoS attack in IoT via SDN-cloud architecture. *IEEE Internet Things J.* **2020**, *7*, 3559–3570. [CrossRef]
30. Sharma, P.K.; Singh, S.; Park, J.H. OpCloudSec: Open cloud software defined wireless network security for the Internet of Things. *Comput. Commun.* **2018**, *122*, 1–8. [CrossRef]
31. Nobakht, M.; Sivaraman, V.; Boreli, R. A host-based intrusion detection and mitigation framework for smart home IoT using OpenFlow. In Proceedings of the 2016 11th International Conference on Availability, Reliability and Security (ARES), Salzburg, Austria, 31 August–2 September 2016; pp. 147–156.
32. Laboratory, M. Intrusion Detection Attacks Database. Available online: https://archive.ll.mit.edu/ideval/data/index.html (accessed on 6 October 2020)
33. Al Shuhaimi, F.; Jose, M.; Singh, A.V. Software defined network as solution to overcome security challenges in IoT. In Proceedings of the IEEE Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions), Noida, India, 7–9 September 2016; pp. 491–496.
34. Ahmed, M.E.; Kim, H. DDoS Attack Mitigation in Internet of Things Using Software Defined Networking. In Proceedings of the IEEE Conference on Big Data Computing Service and Applications, San Francisco, CA, USA, 6–9 April 2017; pp. 271–276. [CrossRef]
35. Lee, I.; Lee, K. The Internet of Things (IoT): Applications, investments, and challenges for enterprises. *Bus. Horiz.* **2015**, *58*, 431–440. [CrossRef]
36. Sivanathan, A.; Sherratt, D.; Gharakheili, H.H.; Radford, A.; Wijenayake, C.; Vishwanath, A.; Sivaraman, V. Characterizing and classifying IoT traffic in smart cities and campuses. In Proceedings of the 2017 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), Atlanta, GA, USA, 1–4 May 2017; pp. 559–564.
37. Chong, G.; Zhihao, L.; Yifeng, Y. The research and implement of smart home system based on Internet of Things. In Proceedings of the Conference on Electronics, Communications and Control, Ningbo, China, 9–11 September 2011; pp. 2944–2947. [CrossRef]
38. Coppola, R.; Morisio, M. Connected car: technologies, issues, future trends. *ACM Comput. Surv.* **2016**, *49*, 46. [CrossRef]
39. Lasi, H.; Fettke, P.; Kemper, H.G.; Feld, T.; Hoffmann, M. Industry 4.0. *Bus. Inf. Syst. Eng.* **2014**, *6*, 239–242. [CrossRef]
40. Shrouf, F.; Ordieres, J.; Miragliotta, G. Smart factories in Industry 4.0: A review of the concept and of energy management approached in production based on the Internet of Things paradigm. In Proceedings of the IEEE Conference on Industrial Engineering and Engineering Management, Selangor Darul Ehsan, Malaysia, 9–12 December 2014; pp. 697–701. [CrossRef]
41. Theodoridis, E.; Mylonas, G.; Chatzigiannakis, I. Developing an IoT Smart City Framework. In *IISA 2013*; IEEE: Piraeus, Greece, 2013; pp. 1–6. [CrossRef]
42. Mohanraj, I.; Ashokumar, K.; Naren, J. Field Monitoring and Automation Using IoT in Agriculture Domain. *Procedia Comput. Sci.* **2016**, *93*, 931–939. [CrossRef]

43. Ou, Q.; Zhen, Y.; Li, X.; Zhang, Y.; Zeng, L. Application of Internet of Things in Smart Grid Power Transmission. In Proceedings of the Conference on Mobile, Ubiquitous, and Intelligent Computing, Vancouver, BC, Canada, 26–28 June 2012; pp. 96–100. [CrossRef]
44. Gope, P.; Hwang, T. BSN-Care: A secure IoT-based modern healthcare system using body sensor network. *IEEE Sens. J.* **2016**, *16*, 1368–1376. [CrossRef]
45. Mahale, R.B.; Sonavane, S. Smart Poultry Farm Monitoring Using IoT and Wireless Sensor Networks. *J. Adv. Res. Comput. Sci.* **2016**, *7*. [CrossRef]
46. Ordonez-Lucena, J.; Ameigeiras, P.; Lopez, D.; Ramos-Munoz, J.J.; Lorca, J.; Folgueira, J. Network Slicing for 5G with SDN/NFV: Concepts, Architectures, and Challenges. *IEEE Commun. Mag.* **2017**, *55*, 80–87. [CrossRef]
47. Wang, S.; Wu, X.; Chen, H.; Wang, Y.; Li, D. An optimal slicing strategy for SDN based smart home network. In Proceedings of the 2014 International Conference on Smart Computing, Hong Kong, China, 3–5 November 2014.
48. Xu, X.; Zhang, H.; Dai, X.; Hou, Y.; Tao, X.; Zhang, P. SDN based next generation Mobile Network with Service Slicing and trials. *China Commun.* **2014**, *11*, 65–77. [CrossRef]
49. Raspberry Pi. Raspberry Pi Foundation. 2020. Available online: https://www.raspberrypi.org/ (accessed on 6 March 2020)
50. Odriod. Hardkernel Co., Ltd. 2020. Available online: http://www.hardkernel.com/ (accessed on 7 March 2020)
51. Zodiac. Northbound Networks Pty. Ltd. 2019. Available online: http://northboundnetworks.com/collections/zodiac-fx/products/zodiac-fx (accessed on 20 December 2019)
52. Google. 2020. Available online: https://developers.google.com/actions/smarthome (accessed on 12 March 2020)
53. Amazon. 2020. Available online: https://aws.amazon.com/iot (accessed on 12 March 2020)
54. Benvenuti, C. *Understanding Linux Network Internals*; O'Reilly Media, Inc.: Sebastopol, CA, USA, 2006.
55. Wani, A.; Revathi, S. DDoS Detection and Alleviation in IoT using SDN (SDIoT-DDoS-DA). *JIEIB* **2020**, *101*, 117–128.
56. Yu, Y.; Guo, L.; Liu, Y.; Zheng, J.; Zong, Y. An efficient SDN-based DDoS attack detection and rapid response platform in vehicular networks. *IEEE Access* **2018**, *6*, 44570–44579. [CrossRef]