DistBlockNet: A Distributed Blockchains-Based Secure SDN Architecture for IoT Networks

Pradip Kumar Sharma, Saurabh Singh, Young-Sik Jeong, and Jong Hyuk Park

The rapid increase in the number and diversity of smart devices connected to the Internet has raised the issues of flexibility, efficiency, availability, security, and scalability within the current IoT network. These issues are caused by key mechanisms being distributed to the IoT network on a large scale, which has motivated the authors to propose DistBlockNet.

ABSTRACT

The rapid increase in the number and diversity of smart devices connected to the Internet has raised the issues of flexibility, efficiency, availability, security, and scalability within the current IoT network. These issues are caused by key mechanisms being distributed to the IoT network on a large scale, which is why a distributed secure SDN architecture for IoT using the blockchain technique (DistBlockNet) is proposed in this research. It follows the principles required for designing a secure, scalable, and efficient network architecture. The DistBlockNet model of IoT architecture combines the advantages of two emerging technologies: SDN and blockchains technology. In a verifiable manner, blockchains allow us to have a distributed peer-to-peer network where non-confident members can interact with each other without a trusted intermediary. A new scheme for updating a flow rule table using a blockchains technique is proposed to securely verify a version of the flow rule table, validate the flow rule table, and download the latest flow rules table for the IoT forwarding devices. In our proposed architecture, security must automatically adapt to the threat landscape, without administrator needs to review and apply thousands of recommendations and opinions manually. We have evaluated the performance of our proposed model architecture and compared it to the existing model with respect to various metrics. The results of our evaluation show that DistBlockNet is capable of detecting attacks in the IoT network in real time with low performance overheads and satisfying the design principles required for the future IoT network.

INTRODUCTION

According to the recent Gartner's report [1], 1 million new Internet of Things (IoT) devices will be sold every hour, and $2.5 million will be spent per minute on IoT by 2021. We believe that the idea of a distributed IoT network is promising. Meanwhile, software defined networking (SDN) empowers easy management and network programmability [2]. Initially, it brings up some issues of security, performance, reliability, and scalability due to the centralized control architecture. Recently, numerous distributed SDN controllers have been introduced to address these issues [3-5]. Most of the existing work emphasizes the issue of state consistency among multiple controllers. The mapping between the controllers and the forwarding devices is statically configured, which can result in uneven distribution of loads between the controllers and bursting packets breaking down the controller. In addition to these issues, we need a low response time and distributed SDN network with high availability. Some methods try to offer a reliable and scalable solution to the distributed network for management [6–10], but none of them have completely solved this problem. On the other hand, blockchains have recently drawn much attention from interested stakeholders in a wide range of industries [11, 12]. The reason behind this explosion of interest is that with the blockchains technique, we can operate the applications in a distributed manner that could previously run through a trusted intermediary. We can accomplish the same functionality with the same assurance without the need for a central authority. The blockchains technique offers a distributed peer-to-peer network where, without a trusted intermediary, untrusted individuals can interact in a verifiable manner with each other [13, 14].

REQUIRE DESIGN PRINCIPLES FOR SECURELY DISTRIBUTED ARCHITECTURE

In order to design high-performance architecture for the IoT network that is securely distributed in order to deal with current and future challenges and satisfy new service requirements, we need to consider the following design principles based on previous work on designing distributed network architecture, research new network technologies, and investigate new service requirements.

Adaptability: Trends are evolving, and the needs of clients are changing. These changing trends require that the network architecture is improved and is able to adapt to the changing environment. Adaptability is vital to ensure its growth and survival. The network architecture should be able to adapt and have its usage broadened with the increase in clients' needs and demands.

High Availability and Fault Tolerance: The high availability of a network control system is important in the actual operation of the network.

Digital Object Identifier:
10.1109/MCOM.2017.1700041

Pradip Kumar Sharma, Saurabh Singh, and Jong Hyuk Park are with Seoul National University of Science and Technology (SeoulTech); Young-Sik Jeong is with Dongguk University.

Thus, the provisioning of a priori redundancies, the detection of failures, and the invocation of mitigation mechanisms are necessary steps for action.

Performance: Ability to adapt performance linearly. In current IoT environments, it is a common challenge to try and achieve linear performance over a large-scale distributed network architecture.

Reliability: When designing the distributed architecture, reliability is ranked as the highest priority. It measures the correlation between the corresponding performance required and the total performance achieved by the system in all environmental conditions of time and space.

Scalability: Scalability is an essential principle in designing a future-proof distributed network architecture, which not only reduces costs, provides the flexibility to extend the network, and supports unexpected services, but also involves the deployment of new services and meets new market requirements.

Security: Security must be everywhere in a distributed network to build a secure distributed architecture that is provided as a service to protect the confidentiality, integrity, and availability of all connected information and resources. Therefore, securing the network must be one of the objectives of designing new distributed architectures.

Research Contributions: On the basis of the discussion above, the main contributions to the research of this work can be summarized as laid out below:

- We are proposing a distributed secure SDN architecture for IoT using the blockchain technique. When using the proposed architecture, security must automatically adapt to the threat landscape without administrators needing to manually review and apply thousands of recommendations and opinions.
- We are proposing a technique for updating the high-performance availability flow rule tables in the distributed blockchain SDN.
- We have evaluated the performance of our proposed technique and compared it to the existing model with respect to various metrics.

The rest of the article is structured as follows. We discuss the proposed distributed secure architecture used for the IoT using the blockchain technique. We also present the architecture workflow and flow rule tables update technique in the distributed blockchain network. Next, we evaluate the proposed model based on different performance metrics. Finally, we conclude the research.

DistBlockNet Distributed Secure Architecture

According to the analysis in the previous section for rapidly growing IoT networks created by the new communication paradigms, we have observed that the currently distributed network architecture, protocols, and techniques are not designed to meet the required design principles for future challenges and satisfy new service requirements. The speed and complexity of this development exponentially creates new categories of attacks; gathering known and mysterious threats; taking advantage of "zero-day" vulnerabilities; and using malware concealed in websites, documents, networks, and guests. At present, organizations need a single distributed secure architecture that includes powerful network security devices with proactive, real-time protection with high performance to meet the analyzed design principles. In this section, we propose a novel distributed secure SDN architecture called DistBlockNet, its workflow, and a technique for updating high-performance availability flow rule tables in a distributed blockchain network.

DistBlockNet Design Overview

DistBlockNet adopts distributed secure network control in the IoT network by using the blockchain technology concept to improve security, scalability, and flexibility, without the need for a central controller. Figure 1 shows the global and local views of the proposed architecture. In the proposed architecture, all controllers in the IoT network are interconnected in a distributed blockchain network manner so that each IoT forwarding device in the network can easily and efficiently communicate. Each local network view comprises OrchApp, Controller, and Shelter modules. The Shelter and OrchApp modules in each local network handle the security attacks at a different level. OrchApp mainly functions at the management or application layers, the controller-application interface, and the control layer. Shelter operates at the data layer, the controller-data interface, and the control layer. The DistBlockNet architecture provides not only operational flexibility, but also proactive and reactive incident prevention based on the recurring threat landscape by inserting the rapidly changing, dynamic, and high-performance OrchApp and Shelter modules. It offers a network infrastructure that is agile, modular, and secure. Protections must dynamically adapt to the threat landscape without having to include security administrators to manually process a huge number of advisories and approvals. These insurances must coordinate well into the more extensive IoT environment, and the architecture must take on a protective stance that cooperatively leverages both savvy inside and outside sources.

OrchApp: Its prime purpose is to offer programming characterized fortifications and to set out them for execution at the appropriate application layer enforcement points, whether implemented using high-performance as host-based software on mobile devices, in the IoT network or the cloud. Security classifications incorporate access control, data protection, and threat intelligence. Based on the underlying domain knowledge from which security strategy plans are drawn, these methods vary.

Access control implements a security convention model of approved associations among resources and clients in the IoT network, as set up by the management layer. On the other hand, *data protection* focuses on the classification of data rather than on behavior and interaction. The management layer concludes the standards or strategies for data flows in the organization. *Threat intelligence* provides the understanding of threats and their behavior. It is powered by applying collaborative intelligence to real-time threats obtained from different communities.

Protections must dynamically adapt to the threat landscape without having to include security administrators to manually process a huge number of advisories and approvals. These assurances must coordinate well into the more extensive IoT environment, and the architecture must take on a protective stance that cooperatively leverages both savvy inside and outside sources.

To identify the attacks in the security policies resulting from the actual changes made to the system data plan, which is linked to each flowchart and to the topological exchange metadata, the graph builder analyzes the parsed dataset to construct and alter the flow diagrams that are connected to the network traffic.

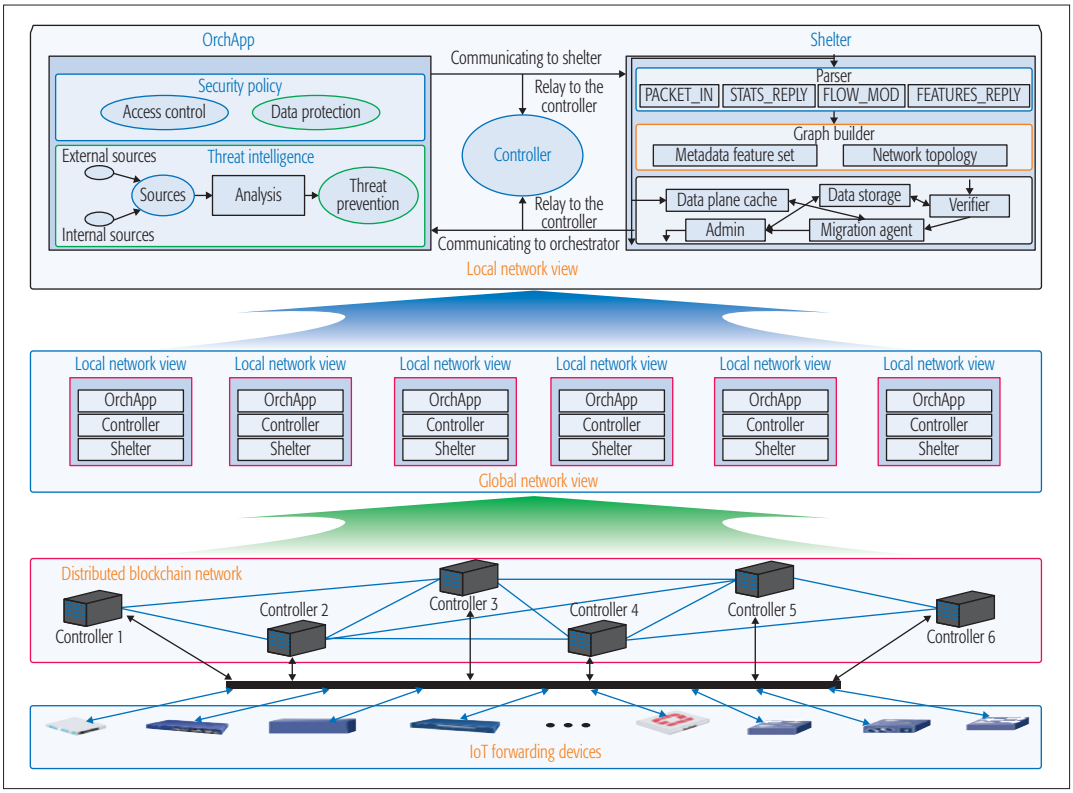

Figure 1. Overview architecture of DistBlockNet.

OrchApp provides the level of adaptability sought to adapt to the new and dynamic threats and modifies the enterprise network configurations. The application layer provides a solid platform that can execute assurances at the application points all throughout the enterprise. Because the protections are software-controlled, critical hardware deployed at these points of the application does not need to be exchanged when a new threat or attack technique is exposed or when new technologies are introduced in the industry. Protections should automatically take place in the threat landscape without requiring manual monitoring of the analysis of a large number of opinions and endorsements. This is accomplished by using an automated threat counteractive action control that works together with the management layer that is only essential for human decision-making for when the threat indicators offer less assurance about recognizing an attack or threat.

Shelter: Attackers often rank in the network to take advantage of the insider's advantage points, and then launch attacks on the internal network. Given that our objective is to assert the appearance of attacks on the network topology and the data plane and the aggression of identity of the flow rules or the strategies within the SDN, our threat system perfectly identifies the scenarios where the antagonist initiates attacks within the SDN. Thus, we designed the SDNs as a non-open system. Removing restrictions on unidentified external communications helps focus our analysis only on OpenFlow control packets or messages within the SDN because the OrchApp handles all of these issues in the DistBlockNet model.

Shelter is composed of a flow control analyzer and packet migration components. The analyzer component takes care of the main functionality of the network infrastructure as soon as the saturation attack has occurred. Whereas, the packet migration component sends a benign network stream to the OpenFlow controller without overloading. As shown in Fig. 1, the module units define the flow analyzer as a control application on the controller platform. Furthermore, the migration agent of the migration component is applied to a controller application between the control plane, the data plane, and an element of the cache data plane.

Parser: The attackers use the subset of OpenFlow messages, such as Packet_in, Flow_Mod, Features_Reply, and Stats_Reply, in order to change the network's view of the controller. Thus, to identify abnormal behavior, we extracted the important metadata by monitoring and parsing incoming packets.

Graph Builder: To identify the attacks in the security policies resulting from the actual changes made to the system data plan, which is linked to each flowchart and to the topological exchange metadata, the graph builder analyzes the parsed dataset to construct and alter the flow diagrams that are connected to the network traffic. Our model retains the flows of logical and physical topologies and Flow_Mod transmission status messages to identify malicious update metadata.

Verifier: We generated path conditions offline and reactive rules online. In order to reduce the overhead at runtime, we processed the path condition generator to navigate the possible paths and to collect all path conditions offline. Online reactive rule generation monitors and assigns the current value of the global variables to the status path. The input variables are symbolized in the path conditions, and the reactive flow rule

dispatcher components are used to parse each status path. It is only with the paths that the final decision is taken into account in processing a small set of modifications to generate a status message. Finally, the reactive flow rules we need are established.

Migration Agent: The migration agent detects attacks and makes the appropriate decisions based on the type of alarms received. In order to generate new rules and migrate the missing table packet in the data cache, it triggers the flow rules of the parser during saturation attacks. It migrates all missing packets to the data plan cache during the generation of flow rules and the update stage. As a result, the controller does not overload itself with the flooding packets. Finally, it processes all the missed packets stored in the cache after the flow rules are updated.

Data Plan Cache: During a saturation attack, it temporarily caches the missing packets. During flooding attacks, most flood packages are redirected to the data plan cache to avoid flooding the controller. By using the classifier, Packet$_i$n generator, and buffer queue, it parses the header of the migrated packets and stores them in the appropriate queue.

Shelter Workflow

As shown in Fig. 1, the Shelter module has three different stages. In the first stage, in order to build a complete network view, Shelter monitors and parses all of the packets communicating with the controller and identifies the appropriate OpenFlow packets. In the second stage, to build an incremental graph network with traffic flow, Shelter analyzes all of these parsed OpenFlow packets to obtain the topological metadata and status of the transmission. Shelter mainly maintains the metadata feature set, the network topological state that is obtained from the OpenFlow packet headers, actual measurements of traffic flow within network connections, and outbound flow path configuration directives, respectively. In the third stage, Shelter allows this metadata to flow against a set of acceptable metadata values collected during the flow period, administrative rules, and strategies. Shelter identifies known attacks through policies specified by the administrator, although it uses precise flow activities obtained over time to detect unplanned and possibly malicious activity.

Shelter does not issue an alarm signal when it detects a new flow behavior. Alternatively, Shelter prompts an alarm signal when it detects untrusted entities that invoke modifiers to the existing flow behavior or where the flow resists any rules or security rules specified by the administrator. Also, Shelter will not raise any alerts on flow reroutes because they are generated by FLOW_MOD messages from the trusted controller. This drastically reduces the alerts that can occur if the recognition of each new behavior is signaled, which is possible in growing networks. However, malicious activity will be noticed by looking back when Shelter later reports authentic activities as being dubious, only to be deemed illegal by the administrator. Shelter can identify such false links by allowing the flow metadata data plan transfer, which collects the flow charts of valid network traffic along a path in the flow graph. Specifically, Shelter applies a custom algorithm to monitor and perceive the bytes of stream statistics by collecting STATS_REPLY messages at each switch in the flow path and determines whether the switches are diverging values of the transmitted byte account.

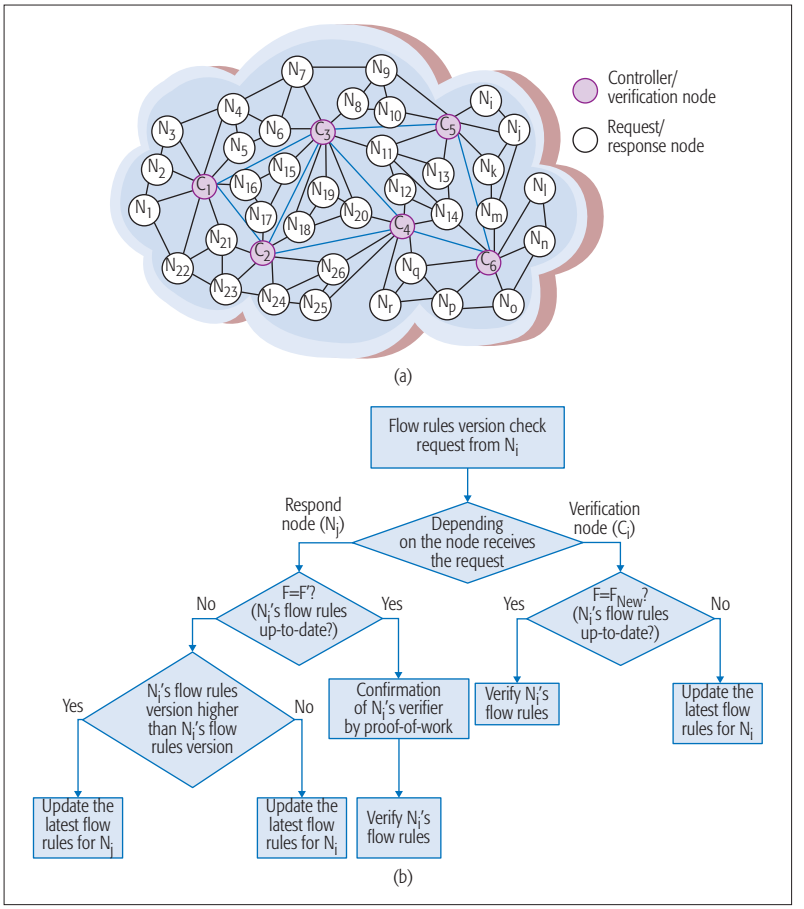

Figure 2. Updating scheme of flow rules table: a) distributed blockchain network; b) flowchart of flow rules table update.

The Updating of Flow Rules Table in the Distributed Blockchain Network

Figure 2a shows the overall DistBlockNet distributed blockchain network. The distributed blockchain network includes the controller/verification and request/response nodes. The verification node denotes the controller in the blockchain network, which maintains the updated flow rules table information in its own database. Request/response nodes are the IoT forwarding devices, which update its flow rules table in a blockchain network. IoT forwarding devices can be a request node or a response node. If a node requests its flow rules table, the node becomes a request node. At the point when a node sends a request message to update its flow rules table, the rest of the other normal nodes are considered to be response nodes from the viewpoint of the requesting node.

Figure 2b shows the DistBlockNet architecture model flow rules update in the distributed blockchain network. When an IoT forwarding device starts its flow rules table update by broadcasting a request packet with a version check, it views it as a request node. Once the version verification request packet is broadcasted in the

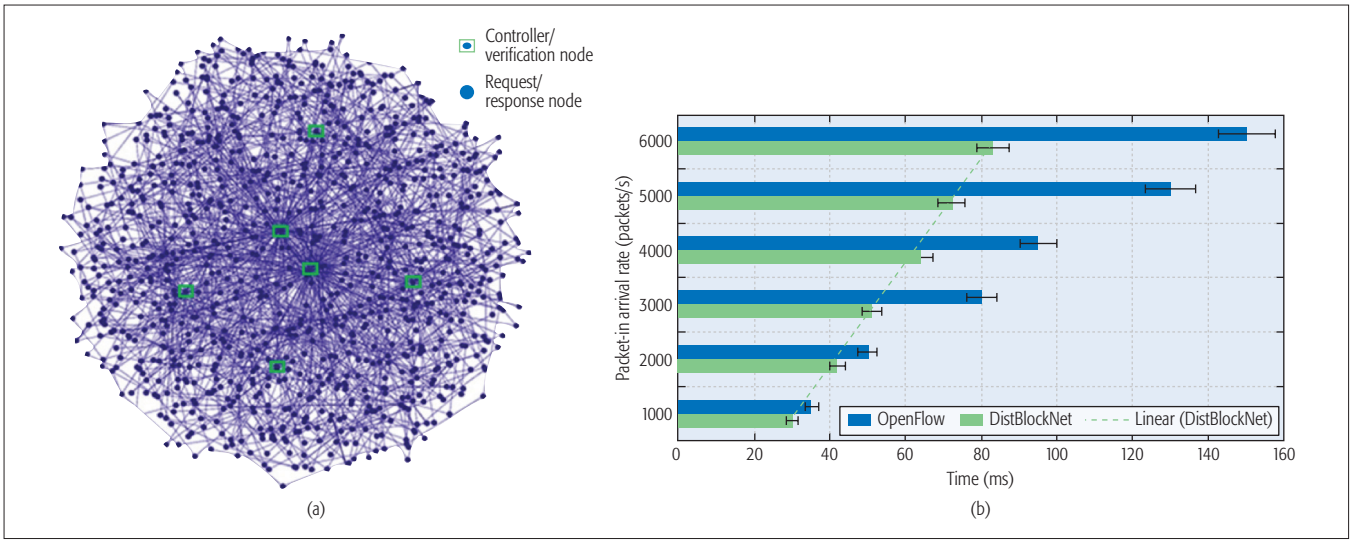

Figure 3. DistBlockNet performance on a large-scale network: a) distributed blockchain network with 6 controllers and 6000 nodes; b) flow table update time vs. packet-in arrival rate.

distributed blockchain IoT network, the rest of the IoT forwarding devices (i.e., response node and all controller/verification nodes) will respond to the request packet of the version verification. The response process varies depending on the node type. In the case of the controller/verification node, it checks whether the request packet node has the up-to-date flow rules table or not. The controller/verification node also checks the integrity of the flow rules table if the requesting node has the up-to-date flow rules table. Otherwise, the controller/verification node sends a response packet with the latest version of the flow rules table to the requested node.

In another case, when the response node receives the request, it checks the request's node version of the flow rules table with its own flow rules table version. If both the request and response nodes have the same version of the flow rules table, the response node requests the other nodes in the distributed blockchain network to verify the hash value of the flow rules table of the requested node. If the response node gets the confirmation of the hash value from the other nodes in the network (i.e., proof-of-work), the response node believes that the flow rules table is correct and sends the responding packet to the requested node. In another case, when the request and response nodes have a different version of flow rules tables, the response node checks whose flow rules table is the latest version. If the response node has the latest version, it will send the response packet to the requesting node with the latest version of the flow rules table. Otherwise, when the response node has a lower version of the flow rules table, it updates its own flow rules table from the request node packet.

Performance Evaluation

In this section, we present the details of the implementation, experimental environment, and evaluation of DistBlockNet. We carried out different experiments to evaluate the scalability, defense effects, accuracy, and efficiency of our proposed DistBlockNet architecture model.

Scalability

To assess the scalability of the DistBlockNet model, large-scale experiments are presented in this subsection with a cluster of 6 Intel i7 3.40 GHz with 16 GB RAM servers. We built a distributed blockchain network with 6 controllers/verifications and 6000 request/response nodes, as shown in Fig. 3a. We used the OpenFlow software switch instead of the OpenVSwitch because when a large number of switches are emulated, OpenVSwitch does not scale well. To compare the performance of the flow rules table update scheme of our proposed DistBlockNet model in a large-scale network, we also built a normally distributed SDN network. Figure 3b shows the result of the flow rules table update time with respect to the packet-in arrival rate in both the DistBlockNet model distributed blockchain network and distributed SDN network. In this experimental result, we observed that our proposed DistBlockNet model constantly performed superior to the distributed SDN network as the rate of the packet-in arrival increased.

Defense Effects

To assess the defense effects of our DistBlockNet model, we evaluated and compared it with an existing OpenFlow network by considering both software and hardware test environments [15]. We used the MININET SDN emulation tool for the software environment. We used the POX controller, OpenFLow switch, and server machines to implement clients and data plane caches in the hardware environment. We used some clients to dispatch a UDP floating attack to the switches. We measured the bandwidth of clients without and with flooding attacks generated by some clients at different speeds to the switch. We evaluated the impact on the bandwidth with and without the DistBlockNet model in both software and hardware environments separately because both environments have different capabilities.

In the software test environment, as shown in Fig.4a, we noticed that the bandwidth starts at 1.9 Gb/s without the presence of any attacks. When we started dispatching flooding attacks, the band-

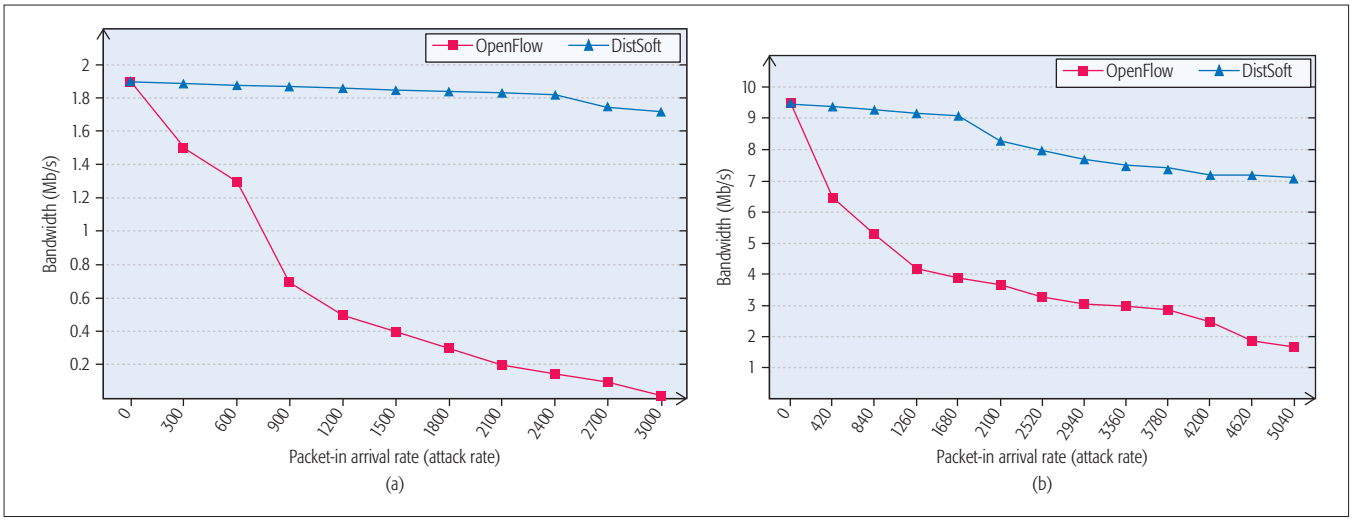

Figure 4. Effects on bandwidth during different attack rate in: a) software environment; b) hardware environment.

width decreased rapidly with an increase in the attack rate. The bandwidth went down to almost half when the packet-in arrival rate reached 800 packets/s. The entire network started malfunctioning when the packet-in arrival rate reached 3000 packets/s. On the other hand, using the DistBlockNet model, the bandwidth started at 1.9 Gb/s without the presence of an attack, and after the packet-in arrival rate reached 3000 packets per second, the bandwidth remained practically unchanged.

Figure 4b shows the results in the hardware test environment. In the hardware test environment, the bandwidth started at 9.5 Mb/s both with and without using the DistBlockNet model with any attack. In this experiment, we noticed that the bandwidth without using the DistBlockNet model went up to half when the attack rate of the packet-in arrival rate reached 1000 packets/s and started malfunctioning when the attack rate reached 5000 packets/s. While using the DistBlockNet model, the bandwidth was maintained above 9 Mb/s until the packet-in arrival reached 1600 packets/s. After that, the bandwidth started to go down because the ternary content addressable memory was not available in our switch. We used the OpenWRT software tool in place of the ternary content addressable memory to execute a flow rule table. Although a software-based flow rule table is not able to achieve a similar level of performance, we still noticed that DistBlockNet conserves resources and provides significant protection.

Accuracy

We evaluated the accuracy rate of the detection of DistBlockNet under two different parameters with one in the real-time identification of attacks and another in the presence various traffic and many distinctive defects in the system. The DistBlockNet model has the ability to identify every attack quickly. In the case of real-time identification, synthetic faults were used in parallel with the suitable traffic with 6K for the Mininet emulsified hosts on our physical testbed. Here we viewed the detection time as the time required for issuing of an alert from the moment when the DistBlockNet model received the offending packet. We used the custom traffic generator, which generates 1500 FLOW_MOD/sec. ARP attacks and fake topology are easily identified when the PACKET$_i$N messages are processed. The detection times may fluctuate because in order to recognize DDoS/DoS attacks, DistBlockNet occasionally runs the flowchart validator and, as a result, the flow diagram size increases. In another case, we used Mininet to increase the number of hosts to 30K. Then we propelled DDoS, ARP poising, and fake topology attacks throughout the distributed blockchain network. We reiterated each examination more than 15 times and noticed that under the distinctive topologies, DistBlockNet effectively recognized each of the issues.

We ran a pessimistic scenario on the false alerts raised for a given δ using conflicting TCP iperf streams. The fair nature of the TCP will create fluctuations in flow to cause changes in the switches along the flow path, which would raise some precautions. As shown in Fig. 5a, we observed that with the increase of δ, the probability of false alarms occurring decreases.

The recall and precision are zero due to the absence of a true positive. In these experiment results, we observed that at the default value of δ = 1.06, there were 7 alarms out of 10 competing flows over 6 min. We also performed this experiment on our physical testbed and obtained comparable results.

To assess the absence of real alerts for a given δ, we defined the ratio between the number of checks that did not raise alerts to the total number of checks that raised alerts during verification. We evaluated the above metric among the Mininet hosts for controlled flows, which are eight hops apart. As shown in Fig. 5b, we observed that the absence of real alerts during verification increases as δ increases. For a given δ, the recall and precision are identical, which is equal to one minus the probability of the absence of real alerts at every data point.

Overhead Analysis

To evaluate the performance overhead of our DistBlockNet model, we used l2 learning and l3 learning applications and recorded CPU utilization during a flooding attack. We simultaneously

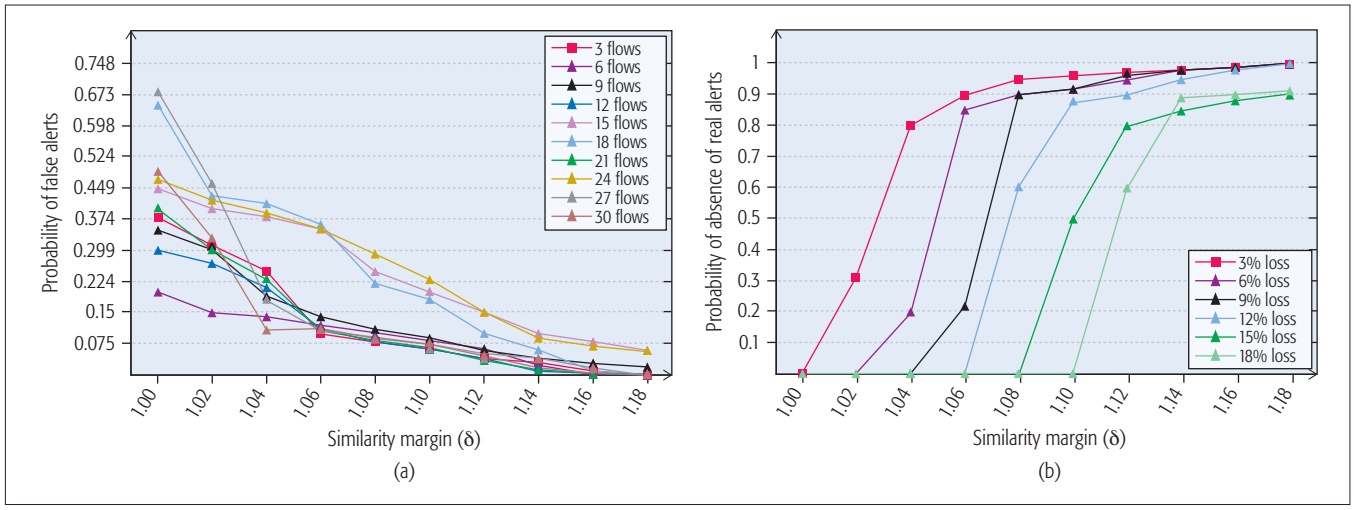

Figure 5. DistBlockNet accuracy rate: a) probability of false alerts with variation in flows and δ; b) probability of absence of real alerts vs. loss rate and δ.

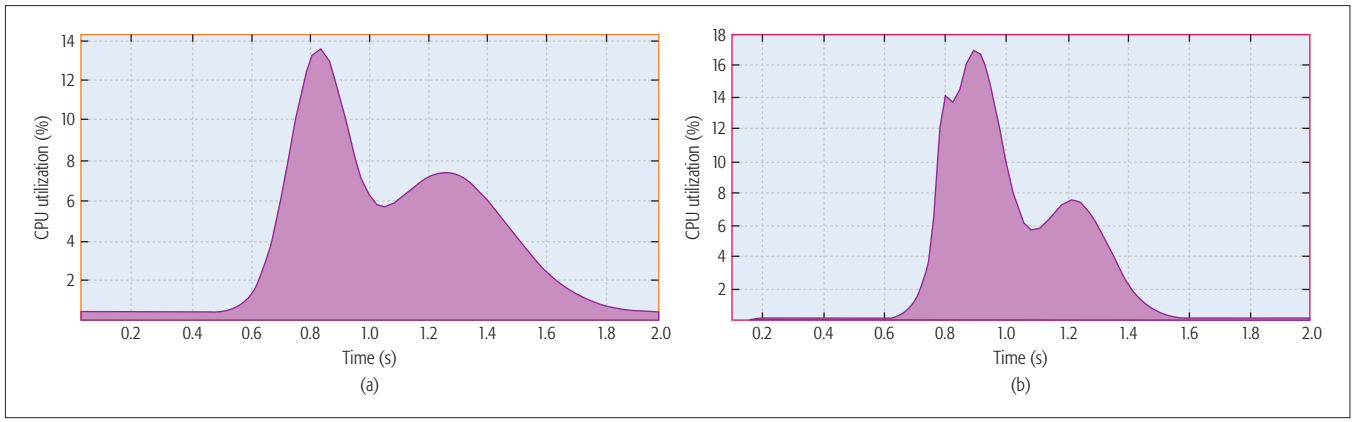

Figure 6. CPU utilization during flooding attack: a) running l2 learning application; b) running l3 learning application.

executed these two applications and used some clients to act as attackers and propelled the saturation attack with a rate of 500 packets/s with a DistBlockNet model in the hardware environment. For each application, we monitored the consumption of the resources. Figure 6 shows the average CPU utilization for all the controllers for different applications with the DistBlockNet model during flooding attacks. The flooding attacks began at about 0.5 s, and we noticed that CPU utilization quickly increased for each application. Then CPU usage started to slowly decrease after we installed the migration rules of flow. Based on the results, we observed that DistBlockNet provides effective protection and creates a more secure distributed network without consuming many resources during a saturation attack.

Conclusion

In this article, based on an analysis of the challenges that large-scale IoT networks face due to new communication paradigms, DistBlockNet, a new distributed secure IoT network architecture consisting of an SDN base network using the blockchains technique, has been proposed to address the current and future challenges and to satisfy new service requirements. DistBlockNet improves a system's performance and capacity. The core role of the DistBlockNet model is to generate and deploy protections, including threat prevention, data protection, and access control, and mitigate network attacks such as cache poising/ARP spoofing, DDoS/DoS attacks, and detect security threats. The DistBlockNet model also focuses on reducing the attack window time by allowing IoT forwarding devices to quickly check and download the latest table of flow rules if necessary. The performance evaluation is based on scalability, defense effects, accuracy rates, and the performance overheads of the proposed model. The evaluation results show the efficiency and effectiveness of the DistBlockNet model and have met the required design principles with minimal overhead.

In the future, we will extend our research work to build a distributed cloud computing architecture with secure fog nodes at the edge of the IoT network.

Acknowledgments

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIP) (No 2016R1A2B4011069)

References

[1] "Top Strategic Predictions for 2017 and Beyond: Surviving the Storm Winds of Digital Disruption," https://www.gartner.com/doc/3471568? ref=unauthreader, accessed May 5, 2017

[2] J. M. Batalla *et al.*, "On Cohabitating Networking Technologies with Common Wireless Access for Home Automation System Purposes," *IEEE Wireless Commun.*, vol. 23, no. 5, Oct. 2016, pp. 76–83.
[3] D. Levin *et al.*, "Logically Centralized?: State Distribution Trade-offs in Software Defined Networks," *Proc. 1st ACM SIGCOMM Wksp . Hot Topics in Software Defined Networks*, Aug. 2012, pp. 1–6.
[4] S. Stefan and J. Suomela, "Exploiting Locality in Distributed SDN Control," *Proc. 2nd ACM SIGCOMM Wksp. Hot Topics in Software Defined Networking*, Aug. 2013, pp. 121–26.
[5] X. Wu *et al.*, "A Multipath Resource Updating Approach for Distributed Controllers in the Software-Defined Network.," *Science China Info. Sciences*, vol. 59, no. 9, Sept. 2016, pp. 92,301–10.
[6] H. Lu *et al.*, "Hybnet: Network Manager for A Hybrid Network Infrastructure," *Proc. Industrial Track, 13th ACM/IFIP/USENIX Int'l. Middleware Conf.*, Dec. 2013, pp. 1–6
[7] D. Drutskoy, K. Eric, and J. Rexford, "Scalable Network Virtualization in Software-Defined Networks," *IEEE Internet Computing*, vol. 17, no. 2, Mar. 2013, pp. 20–27.
[8] Z. Qingyun *et al.*, "On Generally of the Data Plane and Scalability of the Control Plane in Software-Defined Networking," *China Commun.*, vol. 11, no. 2, Feb. 2014, pp. 55–64
[9] Y. Sung *et al.*, "FS-OpenSecurity: A Taxonomic Modeling of Security Threats in SDN for Future Sustainable Computing," *Sustainability*, vol. 8, no 9, Sept. 2016, pp. 919–44.
[10] Q. Vuong, H. M. Tran, and S. T. Le, "Distributed Event Monitoring for Software Defined Networks." *Proc. 2015 Int'l. Conf. Advanced Computing and Applications*, IEEE, Nov. 2015, pp. 90–97.
[11] K. Christidis and M. Devetsikiotis, "Blockchains and Smart Contracts for the Internet of Things," *IEEE Access*, vol. 4, May 2016, pp. 2292–303.
[12] F. Tschorsch and B. Scheuermann, "Bitcoin and Beyond: A Technical Survey on Decentralized Digital Currencies," *IEEE Commun. Surveys & Tutorials*, vol. 18, Mar. 2016, pp. 2084–2123.
[13] X. Xu *et al.*, "The Blockchain as a Software Connector," *Proc. 13th Working IEEE/IFIP Conf. Software Architecture*, Apr. 2016, pp. 1–10.
[14] K. Ahmed *et al.*, "Hawk: The Blockchain Model of Cryptography and Privacy-Preserving Smart Contracts." Univ. MD and Cornell Univ., May. 2015, pp. 1–32.
[15] J. M. Batalla *et al.*, "A Novel Methodology for Efficient Throughput Evaluation in Virtualized Routers," *2015 IEEE ICC*, June 2015, pp. 6899–905.

Biographies

Pradip Kumar Sharma (pradip@seoultech.ac.kr) is a Ph.D. scholar at Seoul National University of Science and Technology. He works in the Ubiquitous Computing & Security Research Group. Prior to beginning the Ph.D. program, he worked as a software engineer at MAQ Software, India. He received his dual Master's degree in computer science from Thapar University (2014) and Tezpur Univerity (2012), India. His current research interests are focused on security, SDN, SNS, and IoT.

Saurabh Singh (singh1989@seoultech.ac.kr) is a Ph.D. scholar at Seoul National University of Science and Technology carrying out his research in the field of ubiquitous security. He holds a strong academic record. He received his Bachelor's degree from Utter Pradesh Technical University and holds a Master's degree in Information Security from Thapar University. His research interests include cloud security, IoT, and cryptography. Finally, he has the experience of being a lab leader of the UCS lab, SeoulTech Korea.

Young-Sik Jeong (ysjeong@dongguk.edu) is a professor in the Department of Multimedia Engineering at Dongguk University, Korea. His research interests include cloud computing, mobile computing, IoT, and wireless sensor network applications. He received his B.S. degree in mathematics and his M.S. and Ph.D. degrees in computer science and engineering from Korea University, Seoul, in 1987, 1989, and 1993, respectively.

James J. (Jong Hyuk) Park (jhpark1@seoultech.ac.kr) received his Ph.D. degrees from the Graduate School of Information Security, Korea University, and the Graduate School of Human Sciences, Waseda University, Japan. He is now a professor at the Department of Computer Science and Engineering, Seoul National University of Science and Technology, Korea. He has published about 200 research papers in international journals/conferences. He has served as Chair and Program Committee member for many international conferences and workshops.

The performance evaluation is based on scalability, defense effects, accuracy rates and the performance overheads of proposed model. The evaluation results show the efficiency and effectiveness of the DistBlockNet model and have met the required design principles with minimal overhead.