ACCEPTED FROM OPEN CALL

SoftEdgeNet: SDN Based Energy-Efficient Distributed Network Architecture for Edge Computing

Pradip Kumar Sharma, Shailendra Rathore, Young-Sik Jeong, and Jong Hyuk Park

The authors are proposing a SoftEdgeNet model, which is a novel SDN-based distributed layered network architecture with a blockchain technique for a sustainable edge computing network. At the fog layer, they introduce an SDN-based secure fog node architecture to mitigate security attacks and provide real-time analytics services. They also propose a flow rule partition, and allocation algorithm at the edge of the network.

Abstract

The volume of data traffic has increased exponentially due to the explosive growth of IoT devices and the arrival of many new IoT applications. Due to the large volume of data generated from IoT devices, limited bandwidth, high latency, and real-time analysis requirements, the conventional centralized network architecture cannot meet users' requirements. Intensive real-time data analysis is one of the major challenges in current state-of-the-art centralized architectures due to the ubiquitous deployment of different types of sensors. To address the current challenges and adhere to the principles of architectural design, we are proposing a SoftEdgeNet model, which is a novel SDN-based distributed layered network architecture with a blockchain technique for a sustainable edge computing network. At the fog layer, we introduce an SDN-based secure fog node architecture to mitigate security attacks and provide real-time analytics services. We are also proposing a flow rule partition, and allocation algorithm at the edge of the network. The evaluation result shows our proposed model allows for a significant improvement of the interactions in real time data transmission. In terms of the ability to mitigate flooding attacks, the bandwidth is maintained above 9 Mb/s until the attack rates exceed 2000 PPS in the hardware environment and the bandwidth remained almost unchanged in the software environment. In the case of scalability of the proposed model, our proposed algorithm has performed better and proceeded linearly with the increase in traffic volume.

Introduction

Nowadays, by offering the most convenience and flexibility in our various daily applications, the advancement and broad deployment of IoT has revolutionized society's way of life. IoT applications, such as smart homes, smart grids, smart transportation, and smart healthcare, have their own unique characteristics in their own respective domains. Although their core is identical, each IoT application has formed its own network by interconnecting a number of IoT devices that collect real-time data in order to achieve better and more intelligent decisions. According to a recent report by Cisco [1], monthly global mobile data traffic will be 49 exabytes, and annual traffic will exceed half a zettabyte by 2021.

Currently, IoT is a big data problem due to the expectation of generating large volumes of sensor data for real-time analysis [2]. The most accurate decision can be taken from the largest real-time data report by IoT devices. It will cost enormous communication resources to compute the real-time data report. This becomes even worse when false data is injected by attackers [3]. This causes an increase in latency, consumes more bandwidth and energy, and causes inaccurate decisions to be made by the controller. On the other hand, with the growing focus on climate change, cloud data centers are also looking for ways to reduce carbon emissions and electricity costs. As a result, desirable techniques should address these issues and challenges to achieve a sustainable network.

Recently, by introducing the concept of fog/edge computing, the computing paradigm has brought computing and storage capabilities closer to the devices that generate data at the edge of the networks. By keeping storage and computing resources closer to the end users, fog/edge computing offers a next generation of applications and services with low communication delay and high bandwidth in geo-distributed public infrastructures. Edge, fog, and cloud computing are usually structured in a layered architecture [4, 5].

Meanwhile, in real-time networking, by providing flexibility in developing new protocols and policies, Software Defined Networking (SDN) is a promising next-generation network architecture. SDN provides programmable functions to dynamically control and manage packet forwarding by moving the control plane to a logically centralized controller [6, 7], whereas in a wide range of industries, stakeholder interests attract the blockchain technique that allows for a distributed peer-to-peer network where untrusted members can interact verifiably without trusted intermediaries [8]. For a sustainable edge computing network, we can use the blockchain technique to provide a secure mesh network and can offer a platform for IoT to reliably interconnect and avoid the threats that affect the centralized architecture model. Recently, there are many industries such as manufacturing and agriculture that use blockchain to

Digital Object Identifier:
10.1109/MCOM.2018.1700822

TPradip Kumar Sharma, Shailendra Rathore, and Jong Hyuk Park are with the Seoul National University of Science and Technology (SeoulTech); Young-Sik Jeong is with the Dongguk University.

0163-6804/18/$25.00 © 2018 IEEE

power IoT systems and enable a secure, low-power network that can remotely manage physical operations without centralized cloud models.

Previously, we proposed DistBlockNet, a distributed mesh network architecture model for securing the IoT network using SDN and blockchain techniques [9]. In the DistBlockNet model, we focused on increasing the scalability of the mesh network using blockchain. Later, we extended our research work and proposed a blockchain based distributed cloud computing architecture with secure fog nodes at the edge of the network [10]. Motivated by the limitations of our previous work, we are designing a new edge computing architecture by pushing computing and storage resources to the extreme edge of the IoT network to manage a large volume of real-time traffic data, overcome resource constraints on the switches, and minimize traffic redirection through an efficient flow rules partition and allocation algorithm for a sustainable edge computing network.

Research Contributions

In this article, we propose an SDN-based distributed layered network architecture with a blockchain technique for a sustainable edge computing network. The main contributions of this research work are summarized below:

- We present architecture design principles for a sustainable edge computing network.
- We propose a novel distributed layered network architecture to meet the design principles required for a sustainable edge computing network.
- We introduce SDN-based secure fog node architecture to mitigate security attacks and provide real-time analytics services.
- We introduce edge node architecture based on SDN and blockchain techniques and propose a flow rule partition and allocation algorithm for a sustainable edge computing network.

Architectural Design Principles

Due to the rigorous requirements of edge computing networks and the need to address major open challenges, we had to design a sustainable edge computing network architecture that takes into consideration the design principles listed below.

Heterogeneity: Due to the rapid development of IoT and its applications, the heterogeneity of communication, networks, and devices have become an essential design requirement that we must consider in order to avoid problems, such as asynchronism in the edge network.

Real-Time Analytics: Real-time analysis is one of the most critical design requirements for performing real-time analysis and extracting usable knowledge from real-time IoT data streams in the edge network.

Reduce Latency and Bandwidth: The reduction of communication delays and the cost of operational bandwidths both play an important role in the introduction of the edge computing concept. Latency can be significantly reduced and save the cost of operational bandwidths if processing and storage capacities are close to end users.

Fault-Tolerance: To ensure a sustainable edge computing network, the designed architecture needs to be fault-tolerant. The system should continue to function and provide services even if certain devices start to malfunction.

High Energy Efficiency: Due to the rapid growth in IoT applications and data volume, the designed architecture should be energy efficient and reduce energy consumption for a sustainable network.

Scalability: Due to the expeditiously increasing number of IoT devices, scalability is an important design principle that must be provided in a new design network architecture.

Security: To prevent and fight against the false data injection and flooding attacks from external attackers, the fog node is able to identify and mitigate the attack at the edge of the network

Proposed SoftEdgeNet Model Architecture

On the basis of the design principles described above, SoftEdgeNet, a novel network architecture for sustainable edge computing networks, has been proposed. In this section, we discuss the design overview of the proposed SoftEdgeNet model, the fog node and edge node architectures, and the workflow of the SoftEdgeNet model.

SoftEdgeNet Design Overview

Figure 1 illustrates the overview of the proposed SoftEdgeNet model architecture. The SoftEdgeNet model adopts decentralized network control on the system level to meet the required design principles. The architecture is designed in a layered model consisting of five layers: data producer, consumer, edge, fog, and cloud layers. At the edge of the network, the data producer layer is the sensing network that contains many sensory nodes and smart devices. To monitor the changing conditions and environments in various public infrastructures over time, these low-cost and highly reliable sensors can be widely distributed. From these geo-spatially distributed sensors, huge sensing data is produced, which must be processed as a coherent set. The data producer layer sends the filtered data that is consumed locally to an edge layer and consumer layer. With condition data, such as date, temperature, and time, the collected data can also be increased. Due to limited network resources, communicating data to be gathered relies on contextual information and conditions given by data consumers that eliminate a large amount of disinterested data.

In the edge layer, the data layer forwards the raw data to the edge layer, which is composed of some high-performance and low-power SDN based computing controllers at the edge of the network. Each SDN controller at the edge layer is associated with a small local group of sensors that covers a small community and is responsible for performing timely data analysis and service delivery. The edge layer reports on the output of data processing results to the intermediate computing node, which is the fog layer and consumer layer (if needed) in our model. On the basis of output data processing results, it is also responsible for the rapid control of feedback to a small local public infrastructure to respond to threats to the monitored infrastructure components and provide the necessary services.

The reduction of communication delays and the cost of operational bandwidths both play an important role in the introduction of the edge computing concept. Latency can be significantly reduced and save the cost of operational bandwidths if processing and storage capacities are close to end users.

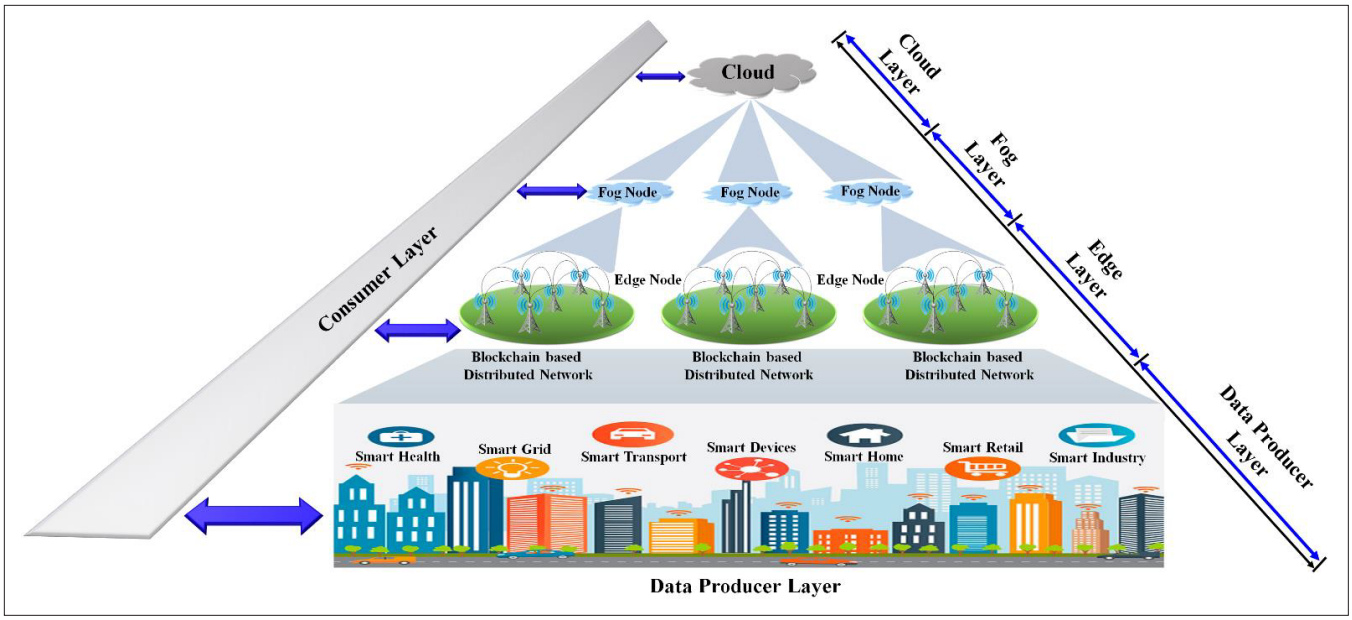

Figure 1. Overview of the proposed SoftEdgeNet model architecture.

The fog layer is comprised of a number of intermediate SDN based computing controllers. To identify potential hazardous events and provide the necessary services, each SDN computing controller at the fog layer is connected to a group of edge nodes at the edge layer and associates temporal and spatial data. When hazardous events are identified, it provides a quick response to control the infrastructure. It provides quick feedback control and services to the edge layer, consumer layer, and data layer. In order to monitor long-term and large-scale conditions and behavioral analysis, it also reports the data processing results to the cloud layer.

To enable context awareness and low latency, the edge and fog layers provide localization, while the cloud layer provides city-wide monitoring and centralized control. To perform long-term pattern recognition, behavioral analysis, and large-scale event detection, the cloud layer provides efficient distributed computing and storage. In the consumer layer, data consumers, such as research organizations, companies, healthcare, educational institutions, transportation, and individual users, can be considered for IoT services. Depending on the requirements, data consumers can request four layers to get services.

SoftEdgeNet Fog Node Architecture

To meet the challenges of large data analysis and provide fast responses across a wide area network, fog computing's hierarchical architecture provides a powerful computing and communications system. To avoid potential bottlenecks in computing, fog computing's massive parallelism offers high-performance computing and can easily balance the load and throughput between all computing nodes and edge devices. Generally, without the inadequate provisioning of resources, applications should be seamlessly sized, which creates new categories of attacks that combine known and mysterious threats, be able to exploit "zero-day" vulnerabilities and use malicious software hidden in documents and networks.

Keeping all of these aspects in mind, we are proposing a fog computing node architecture for a sustainable edge computing network. Rather than transmitting the enormous raw data produced by the data producer layer to the cloud layer, the distributed multi-layer edge and the fog computing nodes at the edge layer and the fog layer only provide representations of data during computing tasks, resulting in a significant reduction in the amount of data transmitted to the cloud. The architecture of the SDN based fog node at the fog layer is shown in Fig. 2. Note that we leveraged the strengths of the FS-OpenSecurity SDN pragmatic security architecture model based on our previous work [7]. The SDN fog node consists of four modules: the attack mitigation module, service management, the context awareness module, and distributed database.

Attack Mitigation Module: This security module is grouped into three phases: packet parser, graph builder, and a validation module. In the packet parser phase, to identify critical OpenFlow messages and create a global network view from incoming OpenFlow packets, it monitors and parses all the packets. To retrieve relevant metadata, we dynamically monitored these OpenFlow messages. Other messages are simply transmitted through this module without further processing. To identify the policy violations or security attacks, we built incremental topological flow graphs associated with the network flow based on the information gathered from the phase of the packet parser phase in the graph builder phase. In order to recognize malicious metadata updates, it maintains the flows of physical and logical topology, such as IP-MAC binding and the host MAC-Port, when the switches generate PACKET_IN messages and the status of the transmitted messages. When the controller generates an FLOW_MOD for the switches, the preferred paths to be taken by the stream and subsequent updates are resolved. To extract the statistical level of flows in the data plane, including bytes/packets transferred, and collect switching configurations, such as port sta-

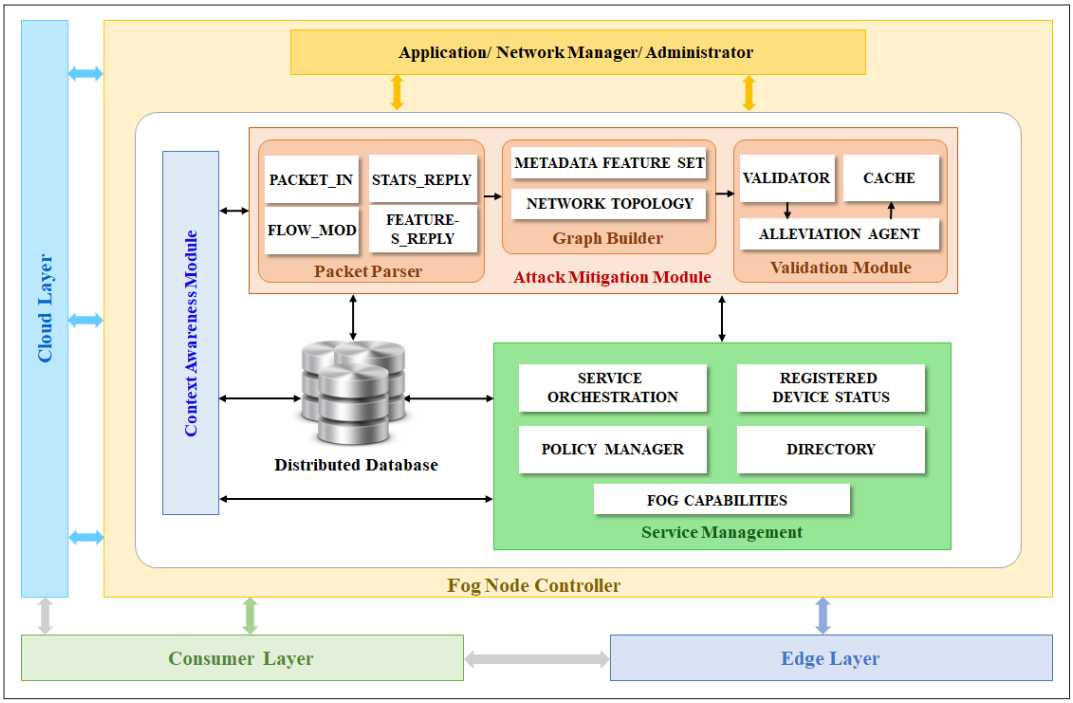

Figure 2. SDN based fog node architecture at edge network.

tus, when a switch first connects to the controller, we use STATS_REPLY and FEATURES_REPLY. The validation phase consists of three components: validator, alleviation agent, and cache. To validate the flow, we generated path conditions in active (offline) and reactive (online) modes. To reduce the overhead of the controller at runtime, we generated path conditions using a conventional symbolic algorithm to navigate all possible paths offline. We used a reactive flow dispatcher to analyze each status path at runtime and generated the reactive flow rules at runtime. The alleviation agent component receives an altered message from the validator component, identifies attacks, and makes a decision. During the saturation attack, the alleviation agent migrates the flows to the data cache and triggers the flow rule dispatcher to generate new rules. As a result, the flood packets will not flood the fog controller. The cache component is used to store missing packets during saturation attacks. Once the new flow rules are generated, the missed packets are fetched from the cache and processed again. The cache uses the packet classifier and buffer queue to store and retrieve the missing packets based on the types. When the cache receives the packets, it parses the packet header and stores it in the appropriate buffer queue. When the module recognizes unreliable entities that cause changes in the behavior of an existing flow or flow that disputes a specified security policy, it triggers alerts and further appropriate steps are taken by the system based on the type of alert.

Service Management: This module offers policy-driven dynamic fog services. Similar to fog infrastructure and services, the management functionality is distributed. The service management module consists of service orchestration, the status of the registered device, policy manager, and directory. Service orchestration provides software-defined protection and allows execution to the application of the appropriate application layer. It offers the level of flexibility needed to handle different types of threats and changes in network configurations. The policy manager component allows the node to provide policy-driven services.

Context Awareness Module and Distributed Database: As the number of sensors deployed in public infrastructure continue to rapidly increase, the volume of data is also increasing. In order to add value to raw sensor data, contextual computing has proved effective in understanding sensor data. Context-awareness, such as activity-awareness and location-awareness, can recognize different activities, and can also do so when the device moves into a specific region and moves far from a specific territory. The distributed database offers faster data storage and retrieval compared to centralized storage, resulting in increased scalability and fault tolerance. It stores all of the metadata, policies, and application data needed to facilitate fog management.

Edge Node Architecture and Workflow

Figure 3 shows the architecture of the SDN based edge node in the sustainable edge network. All base stations under a fog node are connected in a distributed manner using the blockchain technique that provides services to a particular range of public infrastructure. Each base station node is embedded with an SDN based local edge controller. The local SDN controller node is comprised of various modules: radio resources, traffic monitoring, network resources, channel monitoring, switch information, and cache management. The local SDN controller translates the network policies into specific rules in the flow tables, which are implemented using the ternary content addressable memory (TCAM) of each network switch. The flow tables cannot exceed a few hundred entries due to the limitation of TCAM. As a

Service orchestration provides software-defined protection and allows execution to the application of the appropriate application layer. It offers the level of flexibility needed to handle different types of threats and changes in network configurations. The policy manager component allows the node to provide policy-driven services.

As the number of sensors deployed in public infrastructure continue to rapidly increase, the volume of data is also increasing. In order to add value to raw sensor data, contextual computing has proved effective in understanding sensor data.

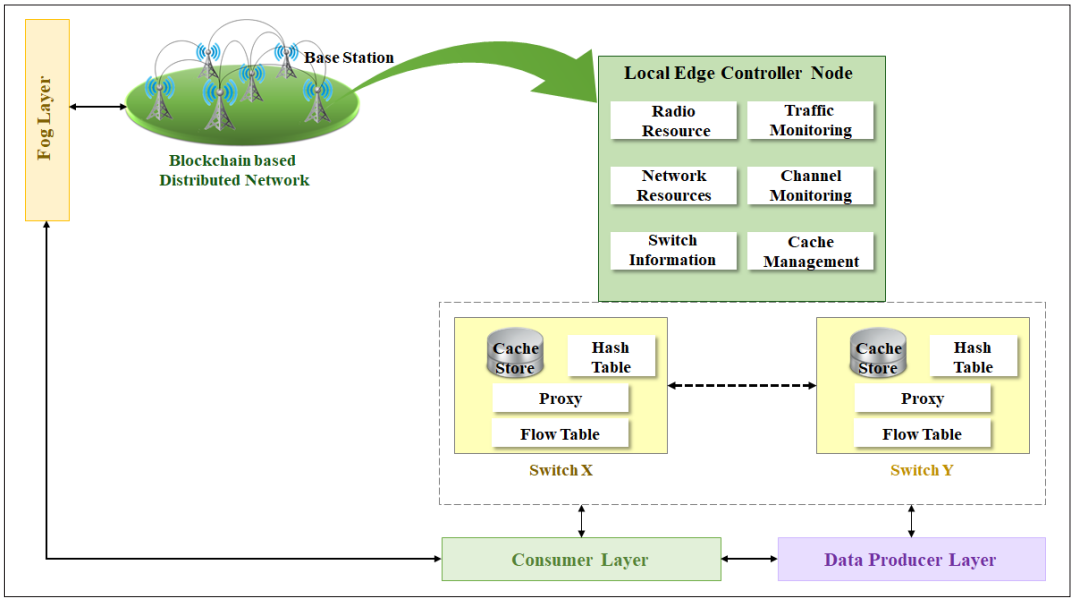

Figure 3. Architecture of the SDN based edge node in the sustainable edge network.

Input: S ← Maximum available flow rules entry space size of the switches
Begin
  FL ← Divide all the flow rules into multiple dependency set groups
  For Each FL_i do
    If $FL_i > S$ **Then**
      $FL_{split} \leftarrow CBD(FL_i)$
      $FL \leftarrow$ REMOVE (FL_i)
      $FL \leftarrow$ ADD (FL_{split})
  For Each FL_i do
    SELECT $Target_{device} \leftarrow vCRIB(FL_i)$
    If SIZEOF($Target_{device}$) < SIZEOF(FL_i) **Then**
      $FL_{split} \leftarrow CBD(FL_i)$
      $FL_{remain} \leftarrow FL_{split} - FL_j$
      FL ← REPLACE (FL_i, FL_j)
      SELECT $Target_{device} \leftarrow vCRIB(FL_j)$
      FL ← APPEND (FL_{remain})
  List of allocated flow rules
End

Algorithm 1. RPAL algorithm.

result, switches must reactively cache rules that cause large buffers and packet delays when cache misses occur. Huang *et al.* [11] proposed heterogeneous flow table distribution in SDN networks considering the properties of policy, dependency, and popularity. They have introduced a rule partitioning algorithm to put the flow rules which have the same policy, dependency and allocation algorithm for load balance the switch. To overcome resource constraints on switches and minimize traffic redirection, a vCRIB algorithm is proposed by Mosherf *et al.* [12]. By keeping all the aspects in mind and to address the above issue and efficiently distribute flow rules across network switches, we have proposed the flow rule partition and the allocation algorithm named "RPAL" in Algorithm 1 by leveraging the strength of the policy and dependency based partition algorithm introduced by Huang *et al.* [11], and the resource and traffic aware allocation scheme presented by Mosherf *et al.* [12].

RPAL Algorithm: The RPAL algorithm is based on two properties: policy, which is one of the strategies that flow rules must execute; and dependency, which is the dependency between flow rules. The steps of the RPAL algorithm are listed below.

Step 1: The different switches have a different flow rules entry space size. We collected all the switch information from the switch information module in the local SDN controller. Let S be the maximum flow rules entry space size available in the current states of the switches.

Step 2: Divide all the flow rules into multiple dependency set groups. It may be that one dependency set is using multiple policies. Distinguished dependency sets must not be dependent on flow rules and may vary in terms of the size of the sets. If the size of an individual dependency set is greater than S, we used the Cut-Based Decomposition (CBD) algorithm [13] to break the dependency between the rules and divide it into multiple dependency sets.

Step 3: For each disjoint set *i*, we selected the target device that maximizes the benefits and the best feasible target device for partition *i* using the vCRIB technique [12].

Step 4: If the size of the available flow table entries *k* of the selected device is less than the size of the disjoint set *i*, we used the CBD algorithm to break the dependency, updated the selected target device with *k* rules, and created another disjointed dependency set with the remaining rules.

Step 5: We repeated Steps 3 and 4 for all the disjointed sets.

Flow Rule Table Validation Workflow at the Edge of the Network: At the edge network, all SDN based base station controllers are connected using the blockchain technique. In order to check and validate the flow rule table version at the base station, we describe step by step the validation scheme of the flow rules as below [9].

Step 1: Base station node B_i sends a flow version check and update request to all the neighboring connected base station nodes.

Step 2: Each request receive node will check the received flow rule version with its own version of flow rules. If it finds that its flow rule version is higher than the requested version received, it will respond with the updated version of the flow rule. The version of the received request may be more up to date than their version. In this case, the receiving base station node generates the flow version check, updates the request, and sends it to all neighboring nodes.

Step 3: Once the requesting node has received the response from its neighbor node, it goes for the Proof-of-Rule approach and takes the appropriate action. With the Proof-of-Rule approach, if the requesting node receives the same responses from neighboring nodes and a total number of same responses is greater than the threshold value of the proof of work, the flow rule table will be updated. The threshold value of the proof of work varies and depends on the shape of the network. Here we considered the threshold values of the proof of work as being 2/3 of the same response received in relation to the total responses.

Step 4: If the requesting node does not meet the proof of work requirement, then it will send its request to the intermediate fog node, and update it accordingly.

Experimental Analysis

In this section, we first evaluate and compare the performance of the different architectures based on the log of the amount of data to the cloud and response time to an event. Then we analyze the defense ability of the fog node during saturation attacks, and then evaluate the performance of the proposed rule partition and allocation approach at the edge of the network.

To emulate open vSwichtes, we used Mininet on a Linux server with 10 desktops, where each desktop had an Intel i7 processor and 64 GB DDR3 RAM. For fog and edge nodes, we ran controllers in separate VMs hosted on a Linux server. To compare the performance of our proposed model, we used the Amazon EC2 cloud data center.

Performance Comparison Between Different Architectures

To collect real-time data, we built a prototype to monitor the environment both inside and around our university campus using sensors and CCTV. By using a 1 km diameter area, as shown in Fig. 4a, the size of the data to be sent to the cloud per second was compared. We looked at three different architectures: the core model, the SoftEdgeNet model with a fog node, and the SoftEdgeNet model with fog and edge nodes. For the core model, we used traditional architecture using the Amazon EC2 cloud data center. To evaluate the efficiency of our proposed model, we plotted the log values of the data size of three different architectures. Figure 4a shows that the size of the transmitted data decreased considerably, which reduced transmission bandwidth and energy consumption.

Due to there being high loads of data transmission, in the case of real-time interactions, it is very difficult to provide real-time control in the core architecture. Keeping these aspects in mind, we analyzed the response time for hazardous events using three different computing architectures. In our experiment, we analyzed the response time for hazardous events under fixed and variable bandwidths. Figure 4a also demonstrates the log response time of different architectures during hazardous events under fixed (1.5 Mb/s) and variable bandwidths. Our results show that our proposed model allows for a significant improvement of the interactions in real time.

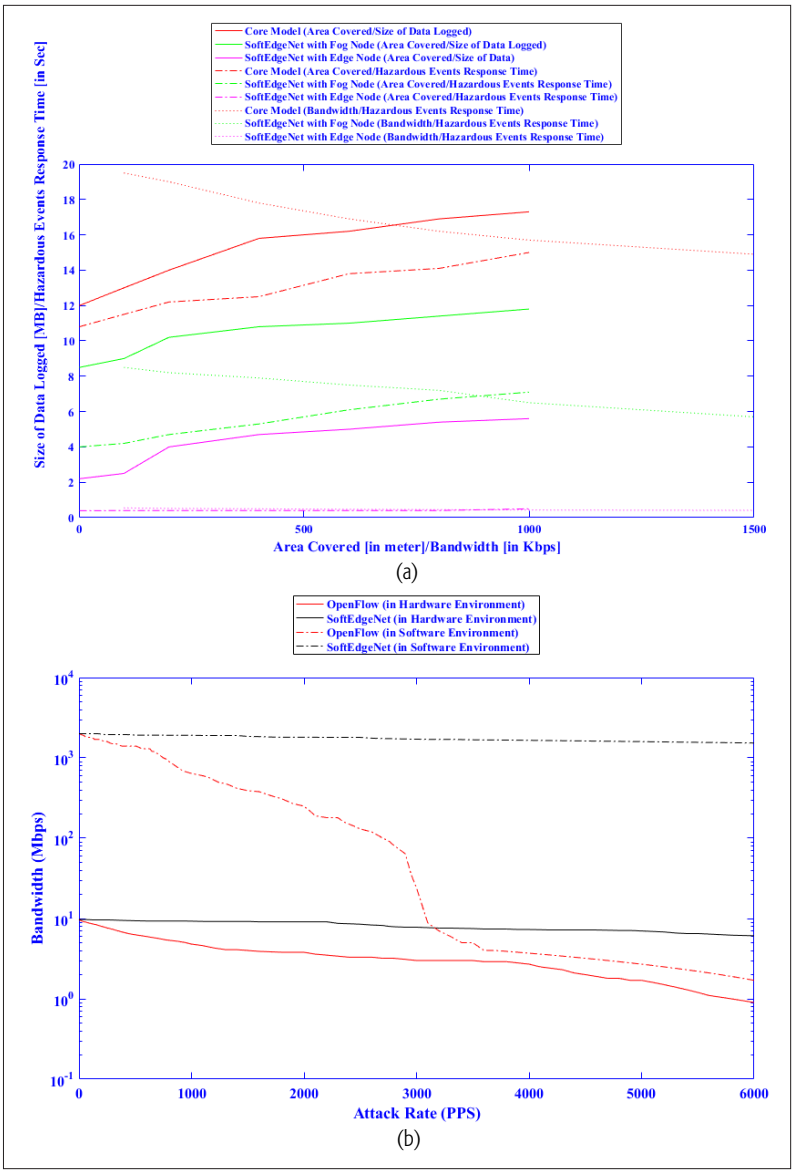

Figure 4. a) Comparison of performance between different architectures; b) Defense effects of fog node during saturation attacks under different environments.

Defense Effects of the Fog Node

To evaluate the performance of our SoftEdgeNet model during saturation attacks, we tested it in hardware and software environments. We configured three clients to launch a UDP flood attack at the edge of the network. To provide forwarding services and discover the topology, we used the POX controller. Under different attack rates, we evaluated the bandwidth with and without using the SoftEdgeNet model in both environ-

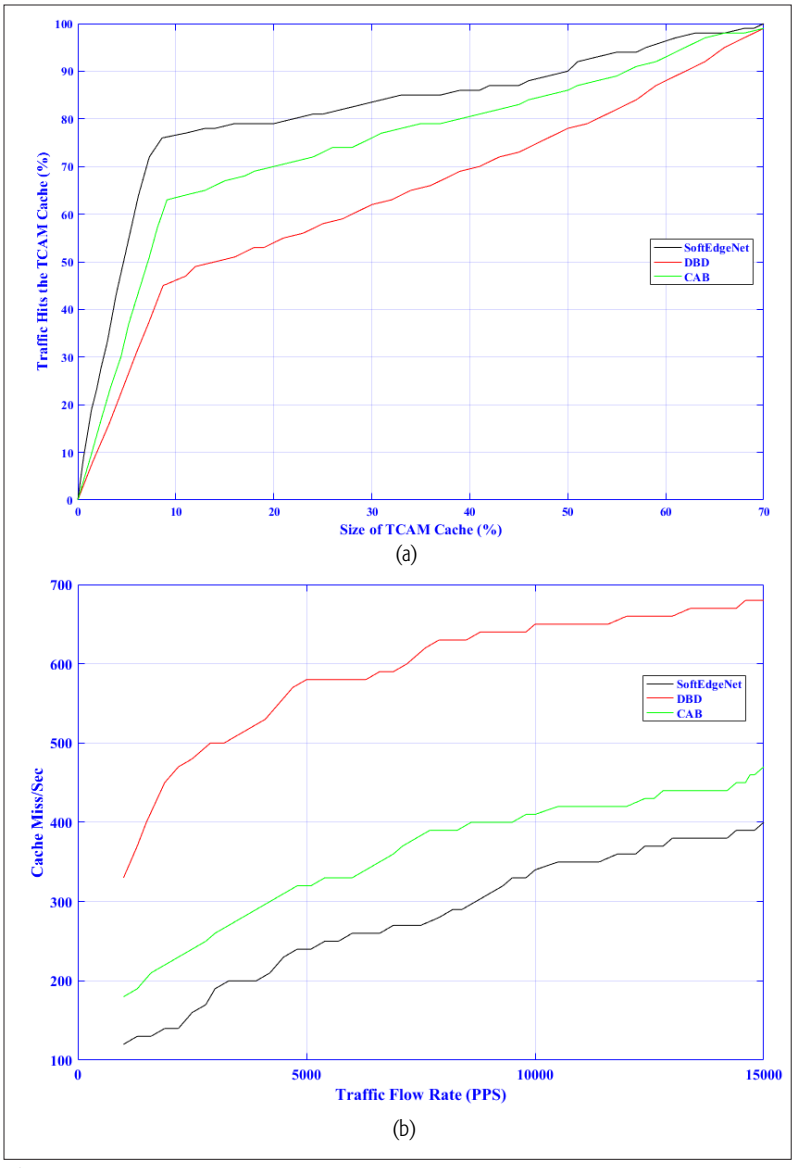

Figure 5. a) Percent of Hits the TCAM cache vs. size of TCAM cache; b) Cache miss/sec vs. traffic flow rate (PPS).

ments. Fig. 4b shows the bandwidth results in hardware and software environments under different attack rates. In the hardware environment, the results show that with an increase in attack rates, the bandwidth decreases rapidly without the Soft- EdgeNet model. As shown in Fig. 4b, the bandwidth started at 9.7 Mb/s in both cases and decreased to almost half when the attack rate reached 1200 packets per second (PPS) without the SoftEdgeNet model and started malfunctioning when the attack rate reached 6000 PPS without the SoftEdgeNet model. With the SoftEdgeNet model, the bandwidth is maintained above 9 Mb/s until the attack rates exceed 2000 PPS. On the other hand, in a software environment, bandwidth started at 2 Gb/s, and in both cases and with an increase in attack rates, it went down rapidly without the SoftEdgeNet model. As shown in Fig. 4b, when the attack rates reached 3600 PPS, the whole network started malfunctioning without the SoftEdgeNet model. However, the bandwidth remained almost unchanged with the SoftEdgeNet model.

Performance Analysis of the Rule Partition and Allocation Scheme

To measure the performance of the proposed rule partition and allocation scheme at the edge of the network, we configured 120 switches, where each switch had eight entries to store sub-tables. Using a 6-bit binary string, each sub-table can store 30 random rules, which generated a dependency among the random rules. We randomly generated the flows and topology of the network. To compare the performance of our proposed algorithm, we implemented the Dependency-based Dispatching (DBD) [14] and CAB algorithms [15].

We used the synthetic firewall policy. Furthermore, by using Zipf distribution, the rules were randomly assigned traffic volume. The percentage of traffic hit the TCAM cache with respect to the size of the TCAM cache, as shown in Fig. 5a. The results in Fig. 5a illustrate that compared to DBD and CAB, the proposed algorithm consistently outperforms and provides better cache capability.

To evaluate the scalability performance of our proposed algorithm during massive volumes of traffic, we measured the cache miss ratio with respect to the varying traffic flow rate PPS. Fig. 5b shows the results of the cache miss ratio vs. traffic flow rate.

On the basis of these end results, we concluded that by pushing computational resources to the extreme edge of the network and with the efficient flow rules partition and allocation algorithm, our proposed algorithm has performed better compared to our previously fog node based proposed model [10] and proceeded linearly with the increase in traffic volume.

Conclusion

We conclude that IoT is very beneficial to our everyday lives. However, we must address some challenges in IoT to take full advantage of it.

In this article, we proposed a distributed layered architecture, called SoftEdgeNet. With the deployment of the SDN-based fog node at the edge of the network, the SoftEdgeNet model cannot only filter inaccurate, false data early and mitigate attacks by external attackers, but also support fault-tolerance. By introducing the edge node architecture to the edge layer with SDN and the blockchain technique, we can meet the required design principles that are needed for a sustainable edge computing network. In addition, extensive performance evaluation was carried out based on various parameter metrics, and the end result demonstrated that our proposed model is superior in performance.

In the future, we will extend our work to meet the limitations of our proposed model by reducing the number of messages in the network and the latency of data access by effectively placing the edge node in the edge network.

Acknowledgments

This work was supported by an Institute for Information & Communications Technology Promotion (IITP) grant funded by the Korean Government (MSIT) (no. 2017-0-01788, Intelligence Integrated Security Solution for Secure IoT Communication based on Software Defined Radio).

References

[1] Cisco Visual Networking Index: Global Mobile Data Traffic-Forecast Update, 2016–2021 White Paper," http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/mobile-white-paper-c11-520862.pdf, accessed 16 Jan. 2018.

[2] Y. Qin *et al.*, "When Things Matter: A Survey on Data-Centric Internet of Things," *J. Network and Computer Applications*, vol. 64, Apr. 2016, pp. 137–53.

[3] R. Lu *et al.*, "BECAN: A Bandwidth-Efficient Cooperative Authentication Scheme for Filtering Injected False Data in Wireless Sensor Networks," *IEEE Trans. Parallel Distributed Systems*, vol. 23, no. 1, Jan. 2012, pp. 32–43.

[4] X. Sun and N. Ansari, "EdgeIOT: Mobile Edge Computing for the Internet of Things," *IEEE Commun. Mag.*, vol. 54, no. 12, Dec. 2016, pp. 22–29.

[5] M. Chiang *et al.*, "Clarifying Fog Computing and Networking: 10 Questions and Answers," *IEEE Commun. Mag.*, vol. 55, no. 4, Apr. 2017, pp. 18–20.

[6] K. Sood, S. Yu, and Y. Xiang, "Software-Defined Wireless Networking Opportunities and Challenges for Internet-of-Things: A Review," *IEEE Internet of Things J.*, vol. 3, no. 4, Aug. 2016, pp. 453–63.

[7] 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.

[8] K. Christidis and M. Devetsikiotis, "Blockchains and Smart Contracts for the Internet of Things," *IEEE Access*, vol. 4, 2016, pp. 2292–2303.

[9] P. K. Sharma *et al.*, "DistBlockNet: A Distributed Blockchains-Based Secure SDN Architecture for IoT Networks," *IEEE Commun. Mag.*, vol. 55, no. 9, Sept. 2017, pp. 78–85.

[10] P. K. Sharma, M. Y. Chen, and J. H. Park, "A Software Defined Fog Node based Distributed Blockchain Cloud Architecture for IoT," *IEEE Access*, vol. 6, Sept. 2017, pp. 115–24.

[11] J. F.Huang *et al.*, "Heterogeneous Flow Table Distribution in Software-Defined Networks," *IEEE Trans. Emerging Topics in Computing*, vol. 4, no. 2, 2016, pp. 252–61.

[12] M. Moshref *et al.*, "Scalable Rule Management for Data Centers," *Proc. NSDI*, vol. 13, Apr. 2013, pp. 157–70.

[13] Y. Kanizo, D. Hay, and I. Keslassy, "Palette: Distributing Tables in Software-Defined Networks," *Proc. IEEE INFOCOM*, 2013, Apr. 2013, pp. 545–49.

[14] N. Katta *et al.*, "Infinite Cacheflow in Software-Defined Networks," *Proc. Third Workshop on Hot Topics in Software Defined Networking*, ACM, Aug. 2014, pp. 175–80.

[15] B. Yan *et al.*, "CAB: A Reactive Wildcard Rule Caching System for Software-Defined Networks," *Proc. Third Workshop on Hot Topics in Software Defined Networking*, ACM, Aug. 2014, pp. 163–68.

Biographies

Pradip Kumar Sharma (pradip@seoultech.ac.kr) is a Ph.D. scholar at the 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 the Thapar University (2014), and Tezpur Univerity (2012), India. His current research interests are focused on security, SDN, SNS, and IoT.

Shailendra Rathore (rathoreshailendra@seoultech.ac.kr) is a Ph.D. student in the Department of Computer Science at Seoul National University of Science and Technology (SeoulTech.), Seoul, South Korea. Currently, he is working in the Ubiquitous Computing Security (UCS) Lab under the supervision of Prof. Jong Hyuk Park. His research interests include information and cyber security, SNS, AI, and IoT. Previous to joining the Ph.D. program at SeoulTech, he received his M.E. in information security from Thapar University, Patiala, India.

Young-Sik Jeong (ysjeong@dongguk.edu) is a professor in the Department of Multimedia Engineering, Dongguk University, Korea. He is the president of the Korea Information Processing Society. His research interests include multimedia cloud computing, information security of 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 in Seoul, Korea in 1987, 1989, and 1993, respectively.

James J. (Jong Hyuk) Park (jhpark1@seoultech.ac.kr) received Ph.D. degrees from the Graduate School of Information Security, Korea University, Korea and the Graduate School of Human Sciences, Waseda University, Japan. He is now a professor with 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 been serving as a chair and on the program committee for many international conferences and workshops.