Hindawi
Wireless Communications and Mobile Computing
Volume 2019, Article ID 5741708, 8 pages
https://doi.org/10.1155/2019/5741708

Research Article

An Active Network-Based Open Framework for IoT

Mahwish Amjad and Faisal Iradat

Computer Science Department, Institute of Business Administration, Karachi, Pakistan

Correspondence should be addressed to Mahwish Amjad; mamjad@iba.edu.pk

Received 9 July 2019; Accepted 5 September 2019; Published 10 October 2019

Guest Editor: Shah Nazir

Major benefits of wireless sensor nodes of IoT like low cost and easy deployment are advocating their usage in variety of applications. Some of them are health monitoring, agriculture, environmental and habitant monitoring, and water monitoring. These nodes are autonomous in nature. It follows that they like to operate in a dynamic and adaptive network environment. So, the communication mechanism between IoT nodes must be robust and adaptive with respect to the environmental change. Unfortunately, the traditional networking architecture supports limited and fixed network computations. These limitations inhibit flexible and robust IoT nodes communication. In addition, the energy consumption in communication nodes is high due to limited processing. To address these issues, this paper gives rebirth to the active system. The proposed active network framework brings a novel integration of the active system with recent technologies (software-defined networking and network function virtualization). As a result of integration, the active system runs as a network function virtualization under the control of software-defined networking. In our view, the amalgam of recent technologies with the active system will promote a robust and flexible IoT nodes communication along with reduced energy consumption. Moreover, various design benefits such as security, flexibility, usability, cost, and performance will be added to the system. Additionally, the proposed framework is open and generalized. It can be extended to other networks such as mobile, satellite, and vehicular networks.

1. Problem Statement

Most of the network functionality in traditional networking architecture is implemented in network hardware (switches, router, and firewall) so that the network functionality, network operating system, and API become part of a network device. However, the implementation of network functionality into the device supports enhanced security. At the other end, the user has been restricted to directly program the network. When it comes to customizing the network system, it brings major difficulties for the network operator to make changes in the network system. More specifically, individual devices need to be manually configured for any change in the network system [1].

On a separate discussion, continuous growth of network devices has enabled Internet of things paradigm [2]. The Internet of things (IoT) nodes are small in size and adaptive in nature. Their autonomous behavior makes them to operate in a dynamic environment [3]. As a result of this, nodes communication brings critical challenges for researchers. In particular, energy management is the most highlighting issue. The limitations of traditional networking architecture, discussed above, inhibit robust communication between IoT nodes [4]. Each time when energy requirement changes with respect to the environment, the whole system needs to be upgraded manually [5]. Consequently, there is an emergent need to propose a network architecture that promotes flexible and reliable IoT device communication in future.

In our view, programmable networks in comparison with traditional networking architecture can enable a robust, reduced energy, and cost-effective communication between IoT nodes. Idea behind programmable networks is to separate hardware and software that promotes flexibility along with vast in-network processing [6, 7].

There are various categories of programmable networks. Active network is the major category that allows user to perform network computations. Accordingly, the overall management of the network system becomes distributed. Moreover, the network is permitted to perform vast in-

network processing. So, in our view, the distributed management and vast in-network processing can efficiently manage energy consumption between nodes communication. Each time when energy requirement changes, the system will not be required to upgrade manually. Following this, a robust and reliable communication between IoT nodes can take place.

Active networks came into existence in early nineties. At that time, recent high-performance technologies, software-defined networking (SDN) and network function virtualization (NFV), were not available. So, critical design challenges brought major difficulties for their implementation. Consequently, the system could not be deployed in real life. We present here major design challenges of active systems:

(1) Flexibility: aim of active networks is to permit user to customize network

(2) Usability: users are able to execute the required software

(3) Performance: functionality added by user must not degrade system performance

(4) Cost: code embedded in active packets must be executed via active routers which are high in cost

(5) Security: code added by user might bring security vulnerabilities

It is always being challengeable to make a balance in between all design issues. For instance, if a secured active system allows only trusted software to be downloaded, then, flexibility of a system gets compromised for security. Similarly, other design challenges like cost, performance, and usability might also be effected for a secured active network system.

Various researchers have proposed active network architecture to make a balance in between all design challenges. We like to present here few line descriptions of only selected (well known) active network architectures, ANTS [8], Switchware [9], DAN [10], and Alien [11].

(1) ANTS architecture is based on mobile code and cache techniques.

(2) Switchware followed a layered approach, where different levels of security and performance are provided to each layer of architecture.

(3) DAN contains finite set of predefined functions along with function identifiers and parameters. These functions are daisy chained. Each packet is assigned a subset of functions, depending upon its type and category. Then, the subset of functions calls daisy-nested functions.

(4) Alien allows only set of functions or software with privileges. However, other functions are restricted in privileges.

Reviewing the past literature, we concluded that active networks are unsecured in nature. Design solutions like restricting functions, providing security in different layers of the active system, or making other architectural change in an active system will not provide a solution for the problem. In our view, instead of changing the active network architecture, it has to be integrated with recent technologies (SDN and NFV). Following this, the security will be provided to the whole active system instead of individual entity.

This paper presents a high-level view of an active network architecture that is the novel integration of the active system with recent technologies, software-defined networking and network function virtualization. Software-defined networking (SDN) [12] is a programmable network that supports centralized management. A central server, called controller, is responsible for centralized management. The proposed technique defines all security measures of the active system inside the controller of SDN. Based on the technique, security will be provided to the whole active system instead of an individual entity or a separate layer. Furthermore, the hardware cost of a system will be reduced using virtualization technique. Network function virtualization (NFV) is one of the promising virtualization techniques that execute multiple network functions. Recently, techniques have been proposed to integrate both the technologies software-defined networking and network function virtualization together [13, 14]. So, the network services can be viewed as a virtual function running at the top of SDN framework.

The proposed system executes the active system as a network virtual function (NFV) in SDN-enabled NFV. The controller of SDN will work as a gate keeper for the active system. The incoming packet will initially be verified via the controller. Then, action will be taken either to drop or forward the packet, after applying considerable security measures. There might be cases when packet will be embedded with complex vulnerabilities. In that case, the system will fail to drop the packet and active system might be the victim of malicious code sent by the active node. However, virtualization support will protect the system hardware to damage. In addition to this, other network functions, running parallel to the active system, will be safe because of isolation between each function. Following this, the system failure will not occur, in virtue of integrating SDN-enabled NFV with the active system. In our belief, this technique will bring enormous advantages for today's network, more specifically, for enabling a robust and flexible IoT nodes communication.

Detailed description of the proposed technique is presented in later sections. In our next section, we provide brief overview of technologies used in our system. Then we discussed the proposed architecture in detail, along with the description of design challenges and packet processing. Finally, we concluded the discussion with highlighting some research challenges.

2. Technologies Used in the Proposed Framework

2.1. Programmable Networks. Unlike traditional networks, programmable networks make separation in between hardware and software. The separation enables network engineer to reprogram whole network without rebuilding it manually. In comparison with traditional networking, it has several advantages:

(i) Reduced cost
(ii) Better resource allocation
(iii) Better allocation of bandwidth
(iv) Better traffic management
(v) Improved flexibility
(vi) Support for security and privacy

2.2. Active Networks. Active networks are programmable networks that allow active packets to modify the operation of the network system. They are different from traditional architecture that the user is permitted to perform network computations. User intervention brings several advantages for the network system. For example, real-time changes do not require the system to be updated manually. So, services are easily deployed with no standardization requirement. In addition to this, intelligent services may be deployed to enhance the system performance.

The user code, which will be allowed to modify the network system, might send either in a discrete or integrated approach. The discrete approach sends the user program separately to the active node. However, the integrated approach encapsulates user program along with data packet via some protocol. The ANEP protocol is one of the most commonly used encapsulation protocols in the active system. The structure of the ANEP protocol is defined in Figure 1. The initial 8-bit field "Version" of the ANEP protocol defines the header format. However, the next 8-bit field "Flags" determines what the node should do, according to the header format (version) specified. For instance, if version is defined to be 1, then the node may take the option to discard the packet. 16-bit field "Type ID" defines the environment in which the packet is executed. The next two 16-bits blocks define the length of header and packet. Second last 32-bit "Options" field determines the method by which active node handles the next 32-bit "Payload" encapsulated into the packet.

Each node executing the active packets (that modifies the network) must have an operating system (JNOS), which provides control over network bandwidth consumed, memory, CPU, and other network resources.

Whole architecture of the active system is shown in Figure 2. According to the figure, it consists of four main layers: hardware (Layer 1), operating system (Layer 2), execution environment (Layer 3), and application layer (Layer 4). All legacy and active routers are residing at the bottom layer called hardware or Layer 1. However, Layer 2, which is called the operating system layer, is responsible for different tasks like resource management, scheduling, interfacing, and, most importantly, support for the execution environment. An execution environment is similar to the Unix shell, and it is actually responsible for executing active packets. In diagram, it is shown at the top of the JNOS operating system, a well-known active network operating system. However, all active applications (AA) are running on top of the execution environment.

2.3. Software-Defined Networking. Software-defined network is a programmable network that enables dynamic and

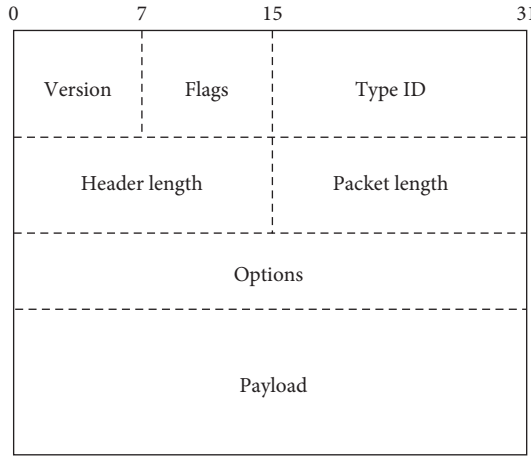

Figure 1: ANEP packet format.

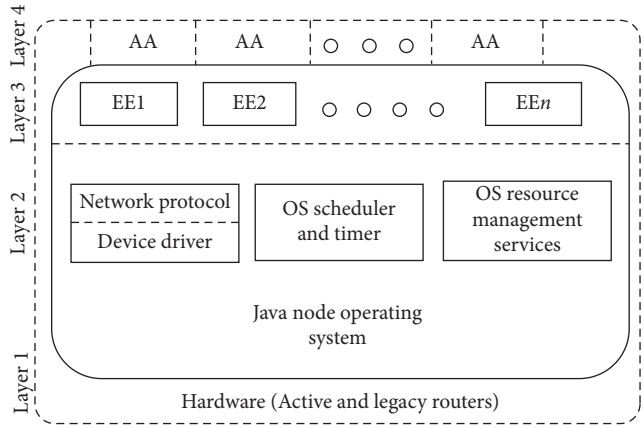

Figure 2: Active network architecture.

feasible networking environment [12]. Unlike the active system, it does not allow every user to modify the network system. Instead, it permits only central server to manage the whole network system. So, all intelligence of a network comes at the central server, which is also called controller. The controller acts as a middle ware between the hardware devices and software applications. All hardware devices are connected to the controller via southbound API. However, software applications are connected to the controller via northbound API. Figure 3 represents the whole architecture of software-defined networking. It is clearly evident from figure that SDN mainly constitutes of three main layers, namely, hardware layer (Layer 1), controller layer (Layer 2), and network application layer (Layer 3). All hardware appliances (routers, switches, etc.) are residing at Layer 1. However, all network applications are running at Layer 3. Layer 2 connects both network applications and hardware appliances with controller via southbound and northbound API, respectively. There are different protocols for southbound API. Openflow is one of the most commonly used southbound API [15, 16]. Any device like to communicate via SDN must support the Openflow protocol so that each switch may be guided how to handle the packets.

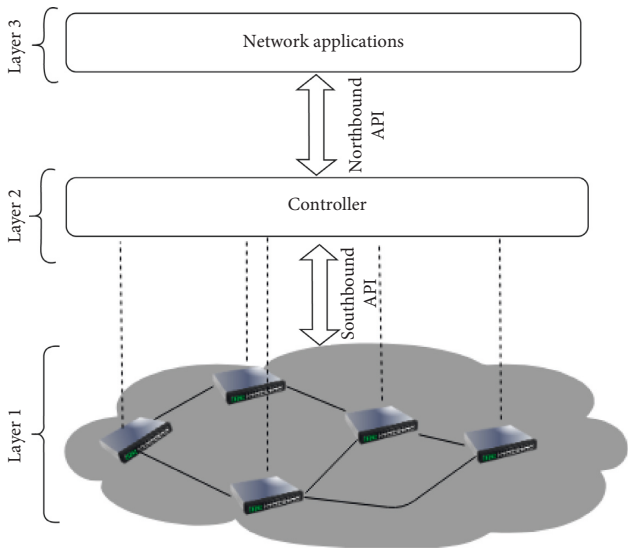

Figure 3: Software-defined networking.

2.4. Network Function Virtualization. Network function virtualization (NFV) is a way to decouple the software services from hardware. It may consist of one or more virtual machines that run different software and process on top of hardware infrastructure (switches, routers, high-volume servers, etc.). Figure 4 represents the NFV architecture. According to the figure, there are three main layers of NFV:

(1) NFVI (network function virtualization infrastructure) layer is responsible to provide virtualization (container, hypervisor, etc.), physical compute, storage, and network components

(2) VNF (virtual network function) is a software application that is responsible to provide network services like routing, firewall, and security

(3) MANO (management and network orchestration) layer is responsible for management and orchestration of VNF in NFVI

3. Proposed Active Network Architecture

3.1. Challenges to Address in the Proposed Architecture. The proposed system uses active networks for IoT nodes communication. Using the active system allows the programmable open nodes to perform network computations at user behalf. User intervention in the network system enables vast network processing. Moreover, the deployment of network services will become easier because hardware will not be required to change for the deployment of network service. With respect to the performance, it is envisioned that end to end performance of a system will be enhanced due to varieties of intelligent services deployed by the user [17]. In addition to this, the overall network management system will become distributed due to the user intervention. Hence, the flexibility, usability, and performance of a system will be enhanced. However, with respect to security and hardware cost, major challenges exist to address.

With respect to security, the active packet and node must be verified to ensure it is not malicious. In our view, following might be the vulnerabilities attached with the code sent by the user:

(i) An active packet may contain a vulnerable code that may destroy the resources of a system.

(ii) An attacker may send a code in active packets that may reconfigure the network services of a system.

(iii) Vulnerable code that may exist in active packet may misuse the resources of a system.

(iv) Since active nodes are responsible to manage the overall network resources, resource fairness and load balancing might be the issue in active systems.

(v) Each active node of a system is allowed to send unlimited number of packets to other nodes. In that case, the network system may be prone to denial-of-service attacks. In which, attacker sends series of network packets to deprive the system from network resources.

With respect to the hardware cost, active routers are major concern. Since active packets require complex and high computational processing, the active routers, which are computationally powerful and expensive, are required to execute active packets [18]. So, challenges exist to reduce active router cost.

3.2. System Architecture. In order to meet with abovementioned challenges of security and hardware, we propose following architectural changes in active network infrastructure:

(i) Place a central server in between active nodes communication for security measurement

(ii) Execute whole active system under virtualization

The two abovementioned changes in the active systems are the critical challenges to be addressed in the proposed system. As we have explained earlier that we would like to build security policies outside the active system rather than inside, it implies that there should be another network system that works as central server. Whenever nodes communicate, that central server can work as middle ware between them. In addition to this, it must have a support for virtualization. In order to meet abovementioned challenges, we propose to use software-defined networks as a central server for nodes communication. Software-defined networking is a centralized programmable network along with a virtualization support (network function virtualization). All security measures of the active system can be defined inside the centralized server. However, the virtualization support will execute the active system as the virtual active system. This implies that the active system will not run on bare hardware. Instead, it will be allowed to work as a virtual network function. So, the overall physical hardware will be protected from malicious activity.

Figure 5 presents the whole view of the proposed architecture. The infrastructure mainly uses the software-

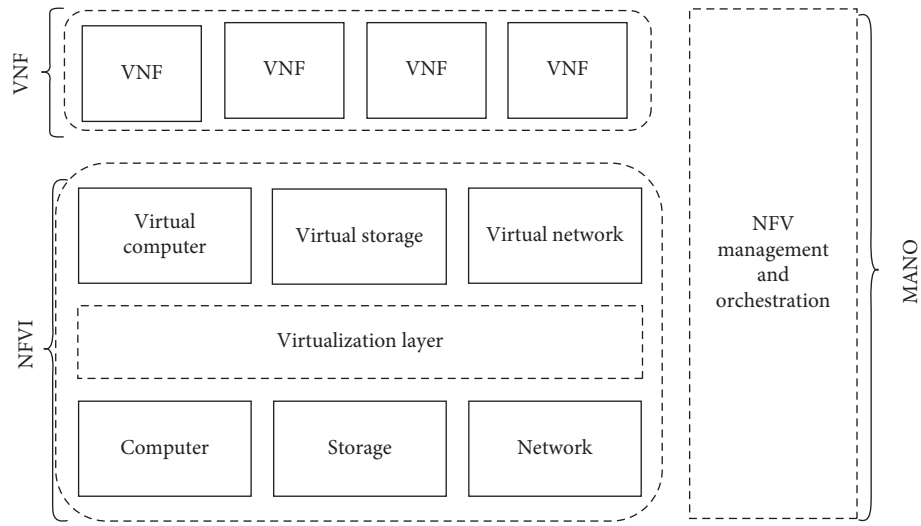

Figure 4: Network function virtualization infrastructure.

defined networking technology. It has three layers: application layer, control layer, and hardware/infrastructure layer. IoT nodes and other hardware devices will remain inside the infrastructure layer. However, the control layer has a support of virtualization, where active network is running as a network function virtualization. All active applications along with SDN applications are running at the application layer, which is at the top of model.

In our view, presence of controller and virtualization support can enable the system to meet the abovementioned challenges. Following sections describe how controller and virtualization can benefit the proposed system:

3.2.1. Controller for Security Vulnerabilities. A controller, placed in between hardware devices and software applications, is responsible for all network-related tasks (redirecting traffic, intrusion detection, firewall, etc.). This central server, controller, might be reconfigured to assure the authenticity of each active packet before forwarding these packets to the neighbor node. The proposed framework reprograms the controller to ensure the following authenticity mechanisms for each active packet:

(i) Identifies the valid sending source.

(ii) Assigns limit on number of packets that sender will be allowed for preventing the network resourcing from malicious use.

(iii) An active packet may stuck in an endless loop. Packet stuck in an infinite loop will occupy the system resource for an endless time. So, the controller in our proposed framework will place the time limitations on packet processing to prevent the unnecessary usage of system resources.

(iv) Controller will allow limited number of resources for each node to ensure the resource fairness issue.

3.2.2. Controller for Hardware Cost. Switches in SDN are dumb devices. Controller of SDN is responsible to instruct these switches how to handle the packet using some matching criteria. The matching criteria may be as follows:

(i) Destination MAC address which defines that the controller may behave as L2 switch

(ii) Destination IP which defines that the controller may behave as a router

(iii) Any header

(iv) It may determine any action using the application layer

Above functionalities of matching criteria define that the SDN controller has the capability to do more than a legacy router. They have a great potential for providing the improved security, reliability, energy management, etc. As a result, they can replace the active router cost.

3.2.3. NFV for Security Vulnerabilities. Network function virtualization is an integral part of an Internet architecture which promises flexibility and security and increases management in network systems [19]. It decouples network functions from proprietary hardware. So, multiple heterogeneous networks run in isolation over virtual hardware. Since each network function executes in their own virtual space, security breaches in one function do not hamper other virtual machines running in parallel.

The proposed framework executes the active system as NFV. This implies that the active system will be a network function running in its own virtual space. So, if the active packet contains the bug or network misconfiguration code, then it can only effect the virtual space where it is running. The other virtual spaces will remain safe from the malicious activity.

3.2.4. NFV for Hardware Cost. The network function virtualization runs on software instead of proprietary hardware. Using the technique, the hardware requirements (firewall, routers, etc.) can be replaced by appropriate software [19]. Since active codes are complex in nature, they

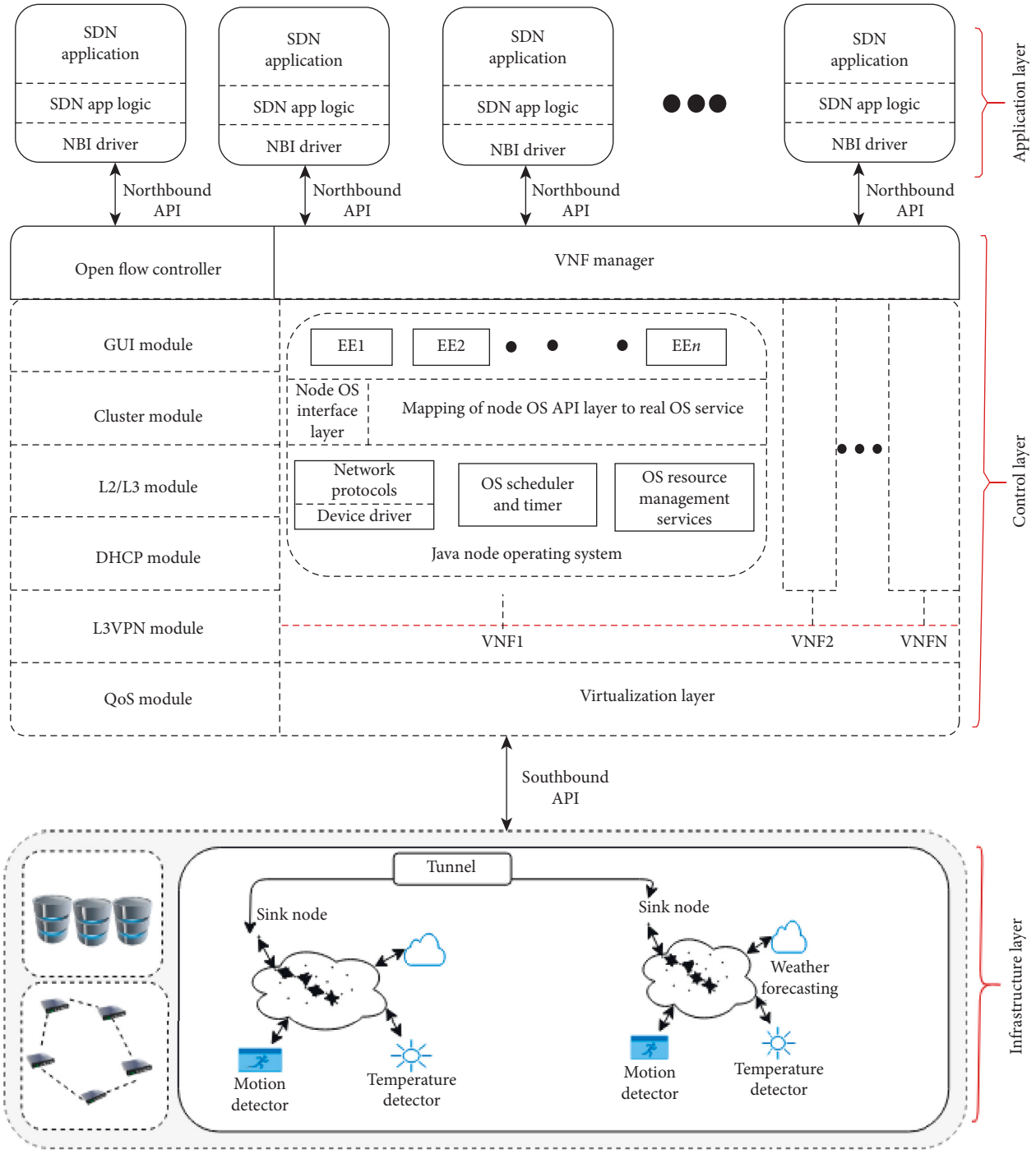

Figure 5: Proposed methodology.

cannot be executed on legacy routers. Instead, they have to be executed on high-performance routers called active routers. Unfortunately, the active routes are high in cost. In addition, in case of security breaches, these routers are expected to be damaged by the vulnerable code. However, the network function virtualization technique does not allow active code execution on bare hardware. So, if security breaches occur, then it does not cause any damage to the physical hardware of a system. Hence, the NFV technology can bring a cost-effective active system.

3.3. Packet Processing in the Proposed Framework. The infrastructure layer of the proposed framework (Figure 5) will be responsible to receive packets from network devices. Following this, packet details such as packet header and payload will be added using the IP protocol. The IP protocol (Figure 6) of the proposed system is inspired from [20] as it has a support for active packets along with legacy packets. Figure 6 represents the structure of IP datagram proposed in [20]. It is clearly evident from figure that the IP datagram reserves some extra bits for IP options. These IP options might

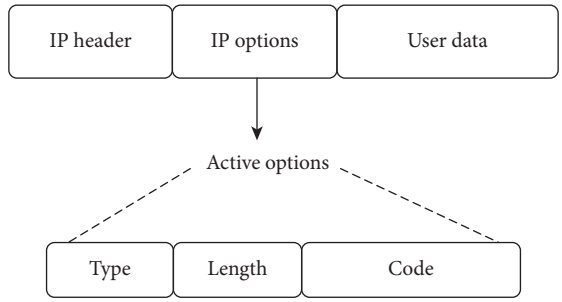

Figure 6: IP protocol for active packets.

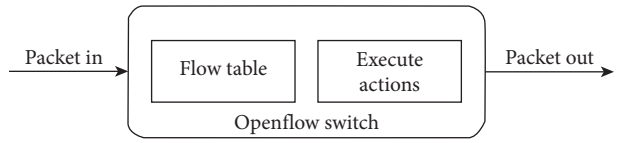

Figure 7: Openflow.

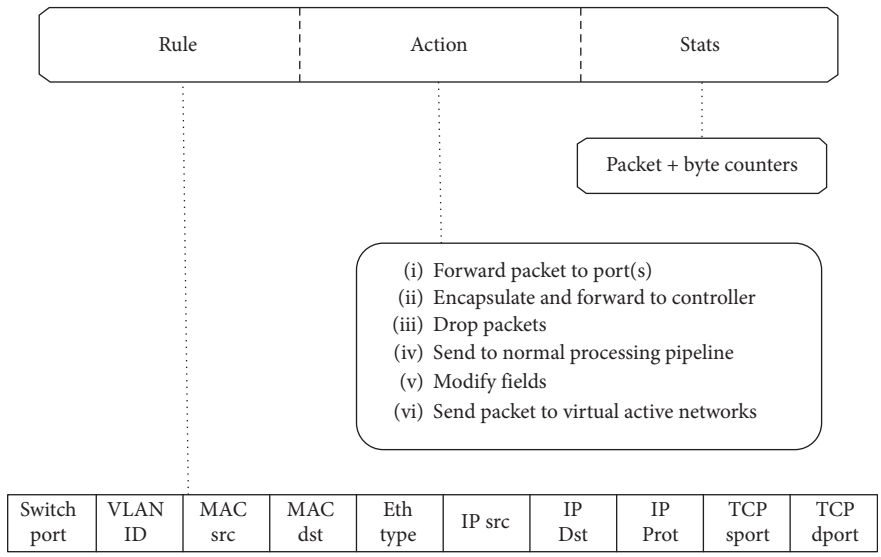

Figure 8: Open flow entries for the proposed system.

have active options, if packet belongs to the active system. The active options include information like the type of active packet, its length, and code fragment to modify the network. A central server, controller, maintains a direct control over hardware infrastructure via a programming interface (API) OpenFlow. Openflow [15], shown in Figure 7, contains one or more flow tables in it.

Flow table in Openflow switch plays a critical role. They define certain rules like forward, drop, or modify the packets. Some more rules will be added, in the proposed system, to support active packets along with legacy packets. Figure 8 shows the structure of each flow entry installed in the Openflow switch. The structure mainly constitutes of rules (source/destination information), actions to take for each rule, and finally the statistics for matching packets. It is clearly visible that an additional entry "Send packet to virtual active networks" has been included in the open flow entries. That instructs the controller how to handle active packets.

The controller is responsible to evaluate the IP packets received from the infrastructure layer with the help of the Openflow protocol. When the packet comes from the infrastructure layer, it can see the active option in IP header. The Openflow entry (vi) shown in Figure 8 instructs the controller that active packet has to be executed via active system. So, the security measures will be applied on incoming packet. Following this, the packets will be delivered to the virtual active system. After that, the active packets can modify the network system. In this way, the user may deploy the intelligent services to improve the system performance. In addition, energy can be efficiently managed with respect to the environment. Since, all security measures have taken place before its entry into the virtual system. Therefore, it is expected that the system will be protected from the malicious active code.

However, there might be exceptional cases of security breaches. So, in those cases, it is assumed that the system is not running at bare hardware. Which protects the system hardware as well as other functions running in parallel to the active system.

4. Conclusion and Challenges

In this paper, we have proposed an active network architecture for flexible IoT nodes communication. The proposed active network architecture is the novel integration of the active system with the recent technologies, software-defined networking, and network function virtualization. It executes active network as NFV under SDN. In our view, this technique will bring considerable design benefits for the active system. Furthermore, the energy utilization between IoT nodes communication will be reduced due to the distributed network management of the active system. Also, it is envisioned that the network management complexities caused by communication overhead might be reduced using the proposed technique. Along with several benefits, the proposed technique widens the space for several research areas. Firstly and mainly, wireless sensor nodes of IoT are one-time programmable. Challenges exist to convert these one-time programmable nodes to reprogrammable nodes. Secondly, some challenges exist with respect to the system performance. At one end, we see that the performance of the system will be enhanced due to intelligent services deployed by active nodes. On the other end, considerable actions must be taken to reduce the scrutinising delay of active packets. Last but not the least, the code embedded in the active system must be verified that at how much extent it can improve system performance.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

References

[1] "Traditional network infrastructure model and problems associated with it," 2016.

[2] K. V. Arya and R. Gore, "Internet of things using software-defined network and cognitive radio network," *Advances in Wireless Technologies and Telecommunication*, vol. 14, pp. 312–328, 2019.

[3] K. Sohraby, D. Minoli, and T. Znati, *Wireless Sensor Networks: Technology, Protocols, and Applications*, John Wiley & Sons, Hoboken, NJ, USA, 2007.

[4] C.-W. Tseng, P.-H. Lai, B.-S. Huang, L.-D. Chou, and M.-C. Wu, "Nfv deployment strategies in sdn network," *International Journal of High Performance Computing and Networking*, vol. 14, no. 2, pp. 237–248, 2019.

[5] H. Kim and N. Feamster, "Improving network management with software defined networking," *IEEE Communications Magazine*, vol. 51, no. 2, pp. 114–119, 2013.

[6] N. Feamster, J. Rexford, and E. Zegura, "The road to SDN," *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 2, pp. 87–98, 2014.

[7] A. T. Campbell, H. G. De Meer, M. E. Kounavis, K. Miki, J. B. Vicente, and D. Villela, "A survey of programmable networks," *ACM SIGCOMM Computer Communication Review*, vol. 29, no. 2, pp. 7–23, 1999.

[8] D. Wetherall, J. V. Guttag, and D. L. Tennenhouse, "Ants: a toolkit for building and dynamically deploying network protocols," in *Proceedings of the 1998 IEEE Open Architectures and Network Programming*, pp. 117–129, San Francisco, CA, USA, April 1998.

[9] D. S. Alexander, W. A. Arbaugh, M. W. Hicks et al., *The Switchware Active Network Architecture*, University of Pennsylvania, Philadelphia, PA, USA, 1998.

[10] D. Decasper and B. Plattner, "Dan: distributed code caching for active networks," in *Proceedings of the IEEE INFOCOM'98*, vol. 2, pp. 609–616, Citeseer, San Francisco, CA, USA, March-April 1998.

[11] D. S. Alexander and J. M. Smith, "The architecture of alien," in *Active Networks*, S. Covaci, Ed., pp. 1–12, Springer Berlin Heidelberg, Berlin, Germany, 1999.

[12] H. Farhady, H. Lee, and A. Nakao, "Software-defined networking: a survey," *Computer Networks*, vol. 81, pp. 79–95, 2015.

[13] K. Kaur, S. Garg, G. Kaddoum, F. Gagnon, N. Kumar, and S. H. Ahmed, "An energy-driven network function virtualization for multi-domain software defined networks," 2019, https://arxiv.org/abs/1903.09924.

[14] O. S. Brief, "Openflow-enabled sdn and network functions virtualization," *Open Networking Foundation*, vol. 17, pp. 1–12, 2014.

[15] P. T. Congdon and C. A. Black, "Controller for software defined network," US Patent App. 16/105,402, 2019.

[16] A.-C. Anadiotis, L. Galluccio, S. Milardo, G. Morabito, and S. Palazzo, "SD-wise: a software-defined wireless sensor network," *Computer Networks*, vol. 159, pp. 84–95, 2019.

[17] K. L. Calvert, S. Bhattacharjee, E. Zegura, and J. Sterbenz, "Directions in active networks," *IEEE Communications Magazine*, vol. 36, no. 10, pp. 72–78, 1998.

[18] T. Wolf and J. Turner, "Design issues for high performance active routers," *IEEE Journal on Selected Areas in Communications*, vol. 19, no. 3, pp. 404–409, 2000.

[19] N. M. M. K. Chowdhury and R. Boutaba, "A survey of network virtualization," *Computer Networks*, vol. 54, no. 5, pp. 862–876, 2010.

[20] D. J. Wetherall and D. L. Tennenhouse, "The active Ip option," in *Proceedings of the 7th Workshop on ACM SIGOPS European Workshop: Systems Support for Worldwide Applications*, pp. 33–40, ACM, Connemara, Ireland, September 1996.