applied sciences

MDPI

Article

An Advanced Strategy for Addressing Heterogeneity in SDN-IoT Networks for Ensuring QoS

Abuzar Zafar [1], Fahad Samad [1], Hassan Jamil Syed [2,*], Ashraf Osman Ibrahim [2], Manar Alohaly [3,*] and Muna Elsadig [3]

1 FAST School of Computing, National University of Computer and Emerging Sciences, Karachi 75030, Pakistan; abuzar.zafar@nu.edu.pk (A.Z.); fahad.samad@nu.edu.pk (F.S.)
2 Faculty of Computing & Informatics, Universiti Malaysia Sabah, Jalan UMS, Kota Kinabalu 88400, Sabah, Malaysia; ashrafosman@ums.edu.my
3 Department of Information Systems, College of Computer and Information Sciences, Princess Nourah Bint Abdulrahman University, P.O. Box 84428, Riyadh 11671, Saudi Arabia; memohamedahmed@pnu.edu.sa
* Correspondence: shjamil@ums.edu.my (H.J.S.); mfalohaly@pnu.edu.sa (M.A.)

Abstract: The internet of things (IoT) is a complex system that includes multiple technologies and services. However, its heterogeneity can result in quality-of-service (QoS) issues, which may lead to security challenges. Software-defined network (SDN) provides unique solutions to handle heterogeneity issues in large-scale IoT networks. Combining SDN with IoT networks has great potential for addressing extreme heterogeneity issues in IoT networks. Numerous researchers are investigating various techniques to resolve heterogeneity issues in IoT networks by integrating SDN. Our study focuses on the SDN-IoT domain to improve QoS by addressing heterogeneity. Heterogeneity in SDN-IoT networks can increase the response time of controllers. We propose a framework that can alleviate heterogeneity while maintaining QoS in SDN-IoT networks. The framework converts m heterogeneous controllers into n homogeneous groups based on their response time. First, we examine the impact of the controller's bandwidth and find that the system throughput decreases when the controller's bandwidth is lowered. Next, we implement a simple strategy that considers both the bandwidth and service time when selecting the peer controller. Our results show some improvement in the framework, indicating its potential to alleviate heterogeneity while maintaining QoS and other metrics.

Keywords: software-defined networks; internet of things; quality of service; security

check for updates

Citation: Zafar, A.; Samad, F.; Syed, H.J.; Ibrahim, A.O.; Alohaly, M.; Elsadig, M. An Advanced Strategy for Addressing Heterogeneity in SDN-IoT Networks for Ensuring QoS. *Appl. Sci.* **2023**, *13*, 7856. https://doi.org/10.3390/app13137856

Academic Editor: Yosoon Choi

Received: 8 May 2023
Revised: 26 June 2023
Accepted: 30 June 2023
Published: 4 July 2023

1. Introduction

Heterogeneous networks allow users to communicate with each other despite having various infrastructures and communication manners that have contrasting properties and characteristics. IoT is one of the largest heterogeneous networks that allows distinct devices to communicate with each other, making an extensive network of various devices. The internet of things (IoT) and the Internet of Everything are now recognized as inevitable concepts in communication technology. It has gained rapid and vast attention from a wide range of applications, services, and industries. It is a global network of machines interconnected with each other and capable of interacting with the real world. In 2014, Gartner predicted that IoT will reach 25 billion units by 2020, up from 0.9 billion in 2009 [1]. In 2021, according to Statista, there were 21.6 interconnected devices around the globe, and it is expected to reach 30.9 billion units in 2025 [2]. IoT provides access to information anytime and anywhere; however, this results in excess traffic. The basic abstract model of IoT is presented in Figure 1 [3]. This model describes the lower layer, referred to as the sensing layer, responsible for collecting data from different applications and services. These collected data are transferred to IoT servers through different paths in a communication network referred to as the network layer. IoT servers provide different services and IoT applications as well. This last upper layer is known as the application layer. This model describes the basic architecture of IoT and its principal layers.

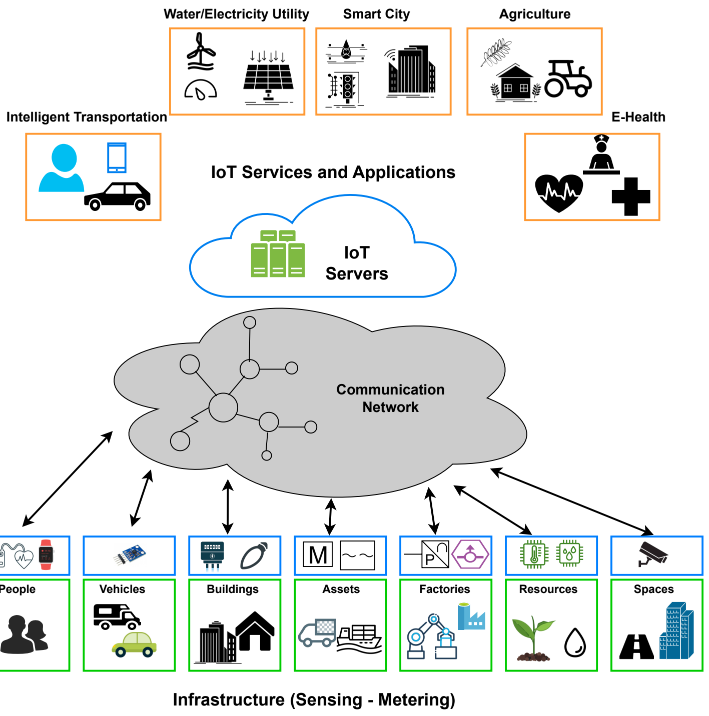

Water/Electricity Utility
Smart City
Agriculture
Intelligent Transportation
E-Health
IoT Services and Applications
IoT
Servers
Communication
Network
People
Vehicles
Buildings
Assets
Factories
Resources
Spaces
Infrastructure (Sensing - Metering)

Figure 1. IoT abstract model.

Smart cities, intelligent transportation systems, smart farming, and smart homes are all concepts that have become a reality due to IoT. Despite all the enormous advancements by IoT in numerous applications, it is still in the development stage. Challenges like security, standard architecture, scalability, management, and heterogeneity still exist, and the research community is dealing with challenges to make IoT more efficient. Hence, a great amount of research is being carried out to improve its technicalities and overcome its issues. Colakovic et al. [4] discussed open and key challenges and issues in IoT. The authors categorized research areas in IoT. Standardization is one of the main research areas in IoT due to the diversity of technology. Currently, there is no single standardization available for IoT despite many standardization bodies being involved to develop such standards.

An open standard is an important facilitator for the development of IoT systems because of its availability to the public. IoT architectures should design in such a way that they provide scalability, interoperability, heterogeneity, and openness. IoT is the complete integration of systems from hardware to software. Hence, the researchers divided architecture into four categories: general architecture, hardware/network architecture, software architecture, and process architecture. The general architecture is a conceptual model for applications, businesses, and services for IoT and operating with other services. Hardware/network architecture enables heterogeneity among different hardware devices and communication protocols and methods. Software architecture provides a common set of services for data collection and processing. IoT application software frameworks are required to provide easy interface with sensors and communication between different devices. Interoperability in IoT systems between different devices is one of the key issues. At all layers of IoT, this interoperability causes a problem, and a layered framework standardization architecture is used to cater to this problem. IoT is developed on various existing communication technologies, and communicating between these different technologies is a complex task.

Ahmed et al. [5] discussed the open challenges in IoT regarding service management. IoT has more diversity in hardware, software, services, and protocols. This diversity raises issues. The concept of web services with a service-oriented solution is introduced to cater to this problem. The main idea was to integrate the Web of Things technology with intelligent devices. The authors in this paper conducted a survey and pointed out some open research challenges in IoT. Challenges in terms of service management in IoT are categorized into four areas. Interoperability challenges include connectivity issues between different devices and protocols, which lead to integrating diverse IoT platforms and how to manage them. The scalability-related challenge in IoT regarding service and management is also discussed. Service management issues regarding scalability include the registration of new devices and upgradation of any software and firmware in a heterogeneous environment.

Various applications are running on the internet. Each application and service has different requirements for QoS (quality of service). Bandwidth, less delay, and packet loss are all parameters that indicate the QoS of any service. These parameters vary according to the QoS with respect to the application's nature. In a traditional network, it is challenging to maintain QoS for various applications and services. Many methods have been proposed to resolve QoS issues, but none of them have had a significant impact due to the nature of traditional networks.

Heterogeneity, directly and indirectly, impacts every other parameter or challenge in IoT. QoS (Quality of Service) is one of the most critical factors in IoT. As multiple domain applications and services are part of the IoT, and their requirements are different, they require different QoS. However, IoT systems consist of heterogeneous devices, in which various network connectivity options, communications modes, and protocols are implemented. Therefore, there is a high degree of heterogeneity in IoT, and there is expected to be more in the future. QoS is one of the burning topics by researchers. There are various issues under the QoS, such as interoperability, response time, reliability availability, mobility, security, and many more defined by researchers [6,7]. All these challenges have direct ties with heterogeneity. Researchers have seen potential in SDN to resolve challenges in IoT. Research communities have presented many methods to cater to different challenges in IoT. The researchers considered the virtualization of networks, implementation of gateways, and SDN-based frameworks to counter the interoperability and heterogeneity problems. The authors also discussed some other challenges in IoT related to QoS (quality of service) that different services require different quality to maintain in large systems. Handling massive amounts of data and keeping them is another complex issue, as is gathering useful information from data and delivering services of IoT to apply AI and ML techniques for better results. Scalability, security, and management are other open research challenges in IoT [4].

In SD-WAN (software defined–wide area network), the controller becomes overwhelmed due to a continuous increase in traffic, and it has limited processing capacity. This impacts the QoS of the network. To cater to this issue, authors in [8,9] presented deep reinforcement learning (DRL) in SD-WAN. They achieved optimization of load-balancing, minimizing the average request delay and increasing the survivability of the network. These studies show the controller capacity's impact on QoS and how modern techniques are being used to mitigate these kinds of challenges.

1.1. Motivation

In this subsection, we discuss a problematic scenario and our motivation behind this topic. As discussed earlier, different IoT applications require different QoS in terms of availability, low latency, high throughput, etc. Kesvan et al. [10] addressed a scenario example regarding QoS in IoT. Consider an industrial application of IoT, where QoS plays a vital role. An intelligent nuclear reactor monitoring system is designed to continuously monitor reactor temperature through thermal imaging. When any anomaly is detected, the system should inform in a timely manner. In this IoT example, tracking in real time is required for safety. In such scenarios, the system should provide data to the station in real time accurately and without any delay. These types of services and applications are time critical, and their traffic is categorized as delay-centric. Issues such as heterogeneity affect the QoS and could have catastrophic results in some applications.

The SDN controller response time has a huge impact on QoS and the security of IoT networks. Heterogeneity also involves increasing the response time of the controller, and in this case, the endpoint security solution will also take time to identify any violation [11]. In an interconnected nodes network, the heterogeneity of the nodes affects the flow, specifically QoS, and also impacts security adoption [12]. Heterogeneous networks are less secure as compared to homogeneous networks because heterogeneity increases the complexity of the system, which makes it harder to differentiate between normal behaviors from attacks by the system. The controller response time is a key metric to meet the specification of QoS. Bandwidth, delay, packet loss, throughput, and others are also important metrics of QoS. Different applications need different QoS key metrics (e.g., multimedia-type applications need to maintain bandwidth, real-time monitoring of critical data needs less delay, etc.). It means that these parameters affect the QoS, and response time plays a crucial part in it.

1.2. Our Contribution

An SDN-based solution has been developed to alleviate heterogeneity in IoT networks. There are multiple solutions to cater to heterogeneity in IoT and improve QoS. Among all these solutions, there is one recent and novel solution proposed by Sood et al. [13] to maintain the QoS by alleviating heterogeneity in the SDN-IoT network, which is also related to security. Their proposed framework output results show improved QoS by alleviating heterogeneity. But there are many aspects from which their framework has not been tested, and they invite the research community to make it more efficient by exploring those aspects. So, we took one of their limitations as our work. Their framework was tested in ideal conditions; however, they did not consider the bandwidth of the controller and network delays, which are the practical parameters that cannot be ignored. Here, we introduce those parameters and find out new results of the system. After that, we will look at how to maintain QoS while handling bandwidth requirements. This will improve the system further.

2. Literature Review

The internet of things (IoT) is a large system of interconnected devices from different vendors that use various communication protocols. The heterogeneity of these devices and protocols is one of the characteristics that make IoT popular but also raises issues and challenges. Colakovic et al. [4] and Ahmed et al. [5] discussed the open challenges of IoT systems related to quality of service (QoS), scalability, management, standardization, architecture, and security, as well as how heterogeneity is linked to all these challenges. Marques et al. [14] discussed open research issues in IoT with some applications and showed how heterogeneity impacts QoS and security.

Different QoS metrics are defined in IoT, which defines specific parameters, such as availability, interoperability, pricing, scalability, and others. Singh et al. [15] defined IoT QoS in terms of communication, computation, and things, with parameters such as jitter, bandwidth, delayed QoS, interoperability, accuracy, efficiency, security, privacy, and response time.

SDN and NFV are two critical technologies that can solve future networking issues and challenges. SDN separates forwarding and control planes, providing a global view of the network, while NFV utilizes virtualization technology to perform network functions by software, separating them from hardware dependency. SDN and NFV are developed independently but can be combined to revolutionize the network domain [16]. Mathias et al. [17] discussed the progression and evolution of SDN, NFV, and their merging, presenting an architecture of SDN-enabled VNF. Soto et al. [18] compared the three most popular SDN controllers in terms of network performance.

SDN has more potential to resolve future network challenges than other solutions, such as name data networking (NDN) [19] and programmable networks [20].

2.1. Quality of Service in SDN

The internet supports various applications and services, such as online gaming, cloud gaming, video streaming, cloud computation, video conferencing, e-commerce, etc., each with its own requirements. Traditional networks based on best-effort models cannot efficiently handle the diverse QoS demands of these applications [21]. To address this issue, different QoS architectures have been proposed, including integrated services (IntServ) [22] and differentiated services (DiffServ) models [23], and multiprotocol label switching (MPLS) technology [24]. However, these methods have their limitations and do not fully support all QoS requirements.

Software-defined networks (SDNs), with their decoupled data and control plane, offer a promising solution to address QoS issues in networks. Researchers have presented various models to monitor QoS parameters in SDN networks, such as FlowSense [25] and PayLess [26]. These models focus on parameters such as bandwidth, delay, throughput, packet loss ratio, jitter, and route tracing to maintain QoS. FlowSense uses a push-based approach to monitor link utilization in flow-based networks, while PayLess provides accurate information about the network in real time with less overhead, using a flexible RESTful API method to collect statistical information at different aggregation levels.

Maintaining quality of service (QoS) in large networks is a significant challenge. Factors such as scalability, traffic management, and bandwidth have a significant impact on network quality. Several SDN-based solutions have been proposed to enhance QoS and address these challenges. The author in [27] presented a reactive approach in the forwarding application of SDN in ISP networks, aiming to control the number of OpenFlow messages. Their efficient quantitative model accurately computes the required number of exchanged OpenFlow messages and bandwidth, as well as the number of flow rules to be installed on each switch.

Another study by [28] highlighted the performance bottleneck of the control channel and introduced a prediction model based on seven functions. Their model achieves a 94% accuracy rate. The load on the SDN controller also affects QoS and other parameters. In [29], the author proposed a novel scheme for load balancing among multiple SDN controllers using the SMCLBRT method. The load balancing is based on the controller response time and load. Additionally, in [30], the author applied a flow redirecting technique to minimize the maximum response time of the SDN controller and improve QoS. Their model takes into account flow-table size and link capacity constraints. The test results demonstrate a 50–80% increase in the maximized controller response time. All these studies highlight the importance of controller response time and capacity in maintaining QoS.

2.2. SDN and IoT

The internet of things (IoT) integrates the physical world with the cyber world, providing better monitoring and control systems for our physical world from around the globe. Traditional internet architecture is unsuitable for IoT, as there are different layered architectures designed for IoT networks. One of them is the simple and easy-to-understand three-layered architecture. The first layer in this architecture is the application layer, which takes commands from the user to control actuators or display information in a meaningful way, gathered from the sensors. The second layer is the network layer, whose main objective is to provide networking services. The last one is the perception or physical layer, which deals with sensors and actuators to gather data and control any device or material thing [31].

The emergence of IoT raises different challenges, like interoperability between devices, efficient and dynamic network management, scalability, and the mobility of nodes. The research community has seen the potential of software-defined networks (SDN) to solve traditional network problems in different technologies. SDN coupled with network functions virtualization (NFV) based IoT architecture can resolve or overcome many issues [32]. SDN and OpenFlow optimization will facilitate the issues of the IoT paradigm. Heterogeneity in IoT is another vast research issue, where different applications require different quality of service (QoS). Implementing intelligent routing, scheduling, and virtualization of the network for slicing can be implemented by SDN to provide QoS according to the IoT application [33].

Combining SDN with NFV in IoT networks offers great opportunities to increase performance [34]. NFV allows many network functions to be stored in the cloud, reducing the load on low-level network devices. The data layer in this architecture has different network resources converted into virtualized resources due to NFV. NFV is also aware of SDN and provides new and unique services due to their coupling. The network operator can easily deploy new services when required. This architecture offers low latency, higher bandwidth, and other features.

SDN-NFV-based IoT systems show positive results in all domains of IoT. SDN coupled with NFV-based IoT architecture is shown in Figure 2 [33]. This architecture is distributed, so no single point of failure is in the system, making it more reliable, secure, and efficient. Ref. [35] proposed an SDN-IoT-based intelligent public transportation system that uses context information and the EI technique in IoT-based systems. The proposed system shows decreased execution time, reduced average delay, increased transportation usage among people, and better transportation management during critical hours.

Adding SDN and NFV layers in the IoT architecture provides more network flexibility, and various vendor devices can operate under one application easily. Virtualization in the network enables many tasks to be automated, reducing operating costs. The addition of SDN in the network also enhances security. The central controller can implement a global security mechanism. Traffic filtration, monitoring, and security policies are also applied in edge switches and routers [36]. Updating security policies is much easier in the SDN-IoT network. WSN is an essential part of IoT. Implementing SDN and NV in WSN will give great benefits to IoT systems [37].

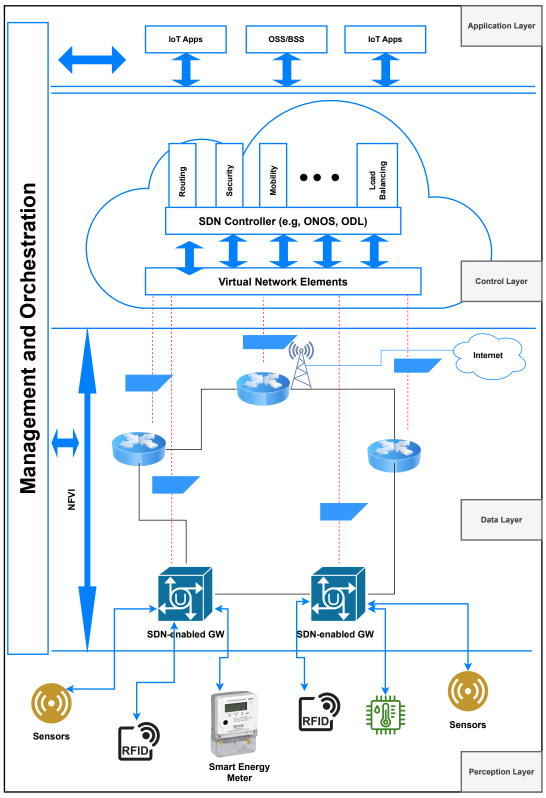


Figure 2. SDN with NFV-based IoT, architecture.

SDN has been applied in WSNs to improve their performance and functionality. Luo et al. [38] embedded VMs in sensor nodes for increased autonomy and utilized on-air programming for central control. Galluccio et al. [39] presented a framework that decouples the control plane from data by utilizing SDN for logic forwarding on nodes. In IoT networks, Thubert et al. [40] used SDN to translate IoT services into the network for a better end-to-end delay. Yiakoumis et al. [41] implemented SDN-orchestrated network virtualization in home network management, using slicing to give full management control over their networks. Batalle et al. [42] suggested SDN-orchestrated network virtualization that relieves core network routers and switches of routing functionality. These architectures demonstrate the potential of SDN in IoT systems while presenting new challenges.

2.3. Recent Work: IoT Systems Using SDN

In the previous subsections, we discussed the heterogeneity issues in IoT and how SDN-based architectures are being applied to IoT networks. In this section, we review recent work performed by the research community to address heterogeneity in IoT systems using SDN-based approaches. We also examine how these approaches handle quality of service (QoS) in SDN-IoT networks.

IoT devices constantly exchange traffic from the network to the server. Improper traffic distribution can lead to resource shortages and server overloads [3]. To solve this, Montazerolghaem et al. [3] proposed a solution that distributes traffic among servers based on their capacity to maintain the QoS of various IoT services. They implemented an SDN-based framework using a predictive and proactive heuristic approach based on time-series analysis. In the controller, they used fuzzy logic rules based on the ratio between the server memory and CPU capacity. They divided the complex problem into two sub-problems to handle time complexity.

The controller architecture has a modular structure that clearly distinguishes between data logic and management. The controller has a global view of all resources, traffic, and the IoT network. An appropriate path and server are selected for each type of traffic using the OpenFlow protocol. The first server is selected, and then the QoS path for that server is chosen. The CPU and memory are essential factors in selecting a server. This is performed through Flow Agent in the controller, which takes CPU usage and memory capacity sampling for statistics and stores it in the database for analysis. Time-series analysis is used for the predictive approach for the controller; they used the normalized least mean square (NLMS) algorithm. The fuzzy logic control load window is provided to every server with the help of the NLMS output. The authors observed that the traditional Kaa load distribution among servers becomes overloaded when traffic increases and the throughput becomes zero. However, their proposed method improved throughput and reduced response time. They tested different traffic classes and compared their results with the integer linear programming (ILP) model in terms of execution time and objective function. The results show that the SDN-based framework's execution time is less than that of the ILP model, and other parameters are very close to it [3].

Tran et al. [43] proposed another SDN-based framework to resolve heterogeneity issues in IoT. This solution focuses on the dynamic environment with an open distributed infrastructure. The central controller can self-observe and adapt. This layered controller is a middleware between IoT devices and the user application level. Observing the user request ontology helps the controller adapt to new conditions [43]. They used the open-source SDN controller Floodlight based on Java. As the IoT network is dynamic, every state information, such as joining nodes, leaving nodes, and network topology, is stored in the database as state info. The user request is translated into the lower level, and according to the service, the path and IoT devices are selected. The lower layer is responsible for selecting the path.

Peros et al. [44] proposed an SDN-based architecture for IoT networks that supports dynamic QoS, unlike previous solutions that focused on static QoS. The architecture has four subsystems, including a non-IP network, a control layer, and a management layer. The IoT devices are differentiated based on their communication capabilities, and the control layer manages the data plane, while the management layer detects node status, assigns QoS profiles, calculates QoS paths, allows reconfiguration, and specifies dynamic QoS rules. The system provides a friendly GUI for users to optimize QoS based on their needs. The author compared the work with other SDN-based QoS solutions and claimed that their work provides dynamic QoS in IoT systems and an easy-to-use interface.

Theodorou et al. [45] proposed MINOS, a middleware-based framework for managing IoT networks using SDN that caters to heterogeneity. It is an operating system that controls the SDN and provides a GUI interface. Sood et al. [13] proposed an SDN framework for managing large IoT systems and maintaining QoS. The framework transforms heterogeneous controllers into homogeneous groups based on their service rate and provides a centralized SDN controller that divides the controllers into groups. The authors used RA and FCFS algorithms to test the approximation model and validated the proof of concept using the Mininet tool and ONOS as the SDN central controller. Figure 3 [45] shows the POC test bed view.

The experimentation results showed a direct relation between the controller response time and the level of heterogeneity, and the throughput of the system was tested by increasing the number of controllers and IoT flows on each controller using both TCP and UDP traffic. The author also discussed some limitations of the approach, including network delay and controller bandwidth.

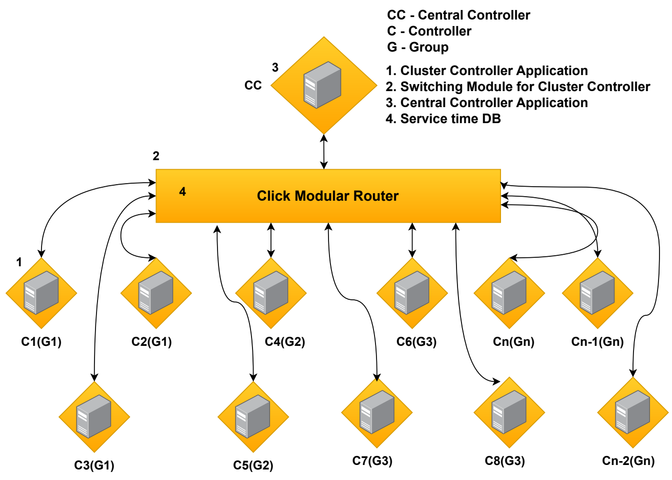


Figure 3. POC test bed view.

In this section, we discuss recent studies on alleviating heterogeneity in SDN-IoT and improving QoS in SDN. Several researchers proposed various and unique techniques in their study. We provide a brief description of related studies and their limitation in Table 1.

Table 1. Summary of related studies.

S.No.	Existing Work	Summary	Limitations
1	[8]	DRL is applied on SD-WAN for load-balancing optimization. It combines the deep Q network (DQN) and deep deterministic reinforcement learning (DDRL) algorithm to learn and divide the load among controllers. This strategy shows significant improvement in controller performance.	They only consider load-balancing among controllers, and migrate flow is not considered in this work. Also, the author identified that we need to study a large number of controller scenarios. This approach is designed for SD-WAN; further research is required for another scenario.
2	[9]	In this system, DRL with multi-agent Q-Network algorithm is applied in SD-WAN to minimize the average request delay and increase the network life. Improvement to these parameters results in better QoS of the network.	In this study, other parameters of the controller need to be investigated. This solution works on the control layer and is tested on intra-domain routing. The model is only implemented in the SD-WAN architecture.

Table 1. *Cont.*

S.No.	Existing Work	Summary	Limitations
3	[13]	Impact of heterogeneity on QoS is showed by author. The service time of the controller and flow arrival time relationship with heterogeneity is presented. A mechanism is developed to reduce the response time of the controller and alleviate heterogeneity by making a cluster of controllers so it can be treated as homogeneous.	Tested in very ideal conditions, as bandwidth controller and network delay parameters are not considered. Other algorithms of schedule are applied to see any improvement. Combining security methods with this framework is not tested.
4	[3]	To maintain QoS in the SDN-IoT network, the best path is selected between the source and server. This method makes decisions on server capacity, available bandwidth, and the classification of traffic. Fuzzy logic is implemented on the SDN controller.	This method only focuses on load balance on the basis of server capacity. It is only applicable for centralized single controllers, not for distributed ones.
5	[43]	To overcome heterogeneity in SDN-IoT, a centralized SDN controller can self-adapt by self-observing the user request. This approach is applied by the author to improve QoS. It uses heuristic routing based on the Lagrange relaxation theory and ontology created for analyzing user requests.	This system adapts according to the user's current needs. The approach is only tested in a simulation; the real-world implementation has not yet been performed. Parameters like controller response time and capacity are not considered.

3. Methodology

IoT is a heterogeneous network, as it consists of many different devices, and there can be a change in the number of devices in the network. The interoperability between these devices is a challenge due to the heterogeneous nature of the IoT network. Security, QoS, and network performance are all factors that are affected due to heterogeneity in the IoT network. The implementation of SDN in the IoT network is able to cater to many of these problems with the right approach. In the future, the degree of heterogeneity will increase as the size of IoT networks increases. This motivates the research community to focus on heterogeneity issues and propose methods to resolve them using the SDN-IoT control plane. In this study, we also focused on this problem with the QoS of SDN-IoT and its security perspective.

Transferring information on time in a secure manner to the end user is important. This makes end-to-end performance parameters significantly important in multi-domain SDN-IoT networks. The controller response time of a flow is one of the key metrics to maintain QoS requirements and improve security at the same time.

Let us discuss some examples and see how QoS and security are interconnected and the controller response time has a direct impact on it. Security mechanisms applied on SDN can be seriously hampered through a huge volume of traffic [46]. In the IoT network, the response time of the controller may be increased due to the heterogeneous nature of the network. This impacts the endpoint security solutions, which result in not identifying violations on time and late responses [47]. It is the responsibility of the controller in an SDN-IoT-driven network to maintain the resources timely; otherwise, the load on the controller will increase, and it will affect controller performance and may lead to security failure [48]. The interconnected node in the network and their nature of heterogeneity may affect the flow-specific QoS. Also, if the service time of the controller is too high, then packet loss will be considered in the network, and duplicate packets will flow into the network upon the request of the switch. This will lead to identical packets in the network and cause flow update problems [13,49].

Our methodology to alleviate heterogeneity in the SDN-IoT network to maintain QoS and enhance security is very much inspired by [13]. The proposed method is simple and feasible. The centralized SDN controller has a universal view of the underlying IoT network. Therefore, it will receive the service rate of all distributed peer controllers. On the basis of these service rates, the central controller divides these distributed heterogeneous controllers into homogeneous groups. In this approach, the controller belonging to the same group is treated as one homogeneous controller. Any mathematical module or theory on homogeneous networks can be implemented here. We implement the pull-based scheduling approach to collect statistics from the peer controllers. To fetch the statistics of the service rate from the peer controller, the NBI parser is used in the central controller. These collected statistics are then forwarded into a sub-module known as the state-collection module and database. From the collected data in the state-collection module, the peer controllers are divided into groups. Grouping is performed by group formation and queuing modules. After forming the groups, the group formation and queuing module fetch the QoS requirement from the incoming flow. This offloads the flow request towards a group that satisfies the application or flow-specific QoS requirements. The proposed framework architecture [13] is presented in Figure 4.

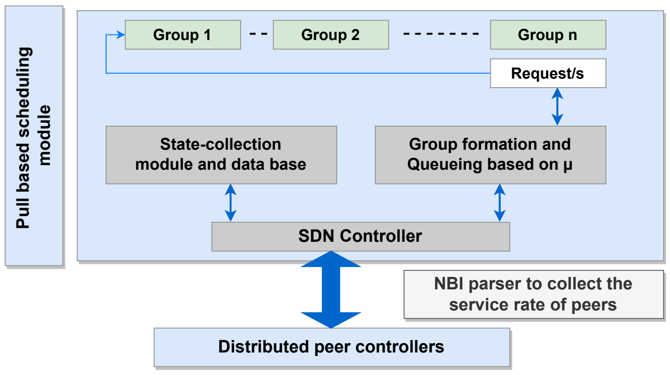

Pull based scheduling module
Group 1
Group 2
Group n
Request/s
State-collection module and data base
Group formation and Queueing based on μ
SDN Controller
NBI parser to collect the service rate of peers
Distributed peer controllers

Figure 4. The high-level architecture of SDN-IoT to alleviate heterogeneity.

This methodology improves QoS in IoT networks by reducing controller response time and increasing real-time attack detection while maintaining security. The approach divides controllers into homogeneous groups based on service rate, transforming heterogeneous controllers into homogeneous ones. This experimental study builds on [13] by introducing real-world parameters, such as bandwidth, using a network from the Internet Zoo Topology. Various simulations were conducted under different conditions, including changes in the number of hosts and protocols, with results compared to the base paper.

4. Proposed Solution

There have been various approaches proposed to address the challenges faced by IoT networks. One popular approach involves leveraging SDN to address the issue of heterogeneity. However, there are still several open research gaps in SDN-IoT networks. In this study, we propose an extension to an existing method that aims to alleviate heterogeneity while maintaining QoS in SDN-IoT networks. We identified a research gap in the method proposed by [13], which demonstrated the effectiveness of their framework in addressing heterogeneity and maintaining QoS, as well as improving security. However, their approach had certain limitations. To address this, we introduce the concept of the controller bandwidth in the network and evaluate the performance of the improved approach. We believe that our proposed approach will help fill the identified research gap and provide valuable insights into the effectiveness of this approach in addressing the challenges faced by SDN-IoT networks.

In the previous section, we discussed the impact of the service rate of controllers on the SDN-IoT network's QoS and security mechanism. To alleviate heterogeneity while maintaining QoS, controllers with different service rates are grouped together. In our experimentation, we considered two algorithms: the random algorithm (RA) [50] and the first come first serve (FCFS) algorithm. The RA algorithm selects a controller randomly to serve the incoming flow, while the FCFS algorithm prioritizes the first request in the queue [51].

There is a central controller and to achieve the closed-form expression of many metrics of interest, the M/M/1 queuing discipline is selected. It also aligns with the previous research conducted in this domain. The M/M/1 queue represents a basic system with a single controller, where the queue length corresponds to the number of incoming IoT flows following a Poisson distribution. IoT flows are considered two events in our model that occur in different time periods. Incoming IoT flows follow the Poisson distribution, and the service rate of each SDN controller follows an exponential distribution. These are the basics of queuing theory, and M/M/1 is the most suitable choice. To analyze the system's performance, we treated the entire IoT network as an M/M/1 queuing discipline [52]. The arrival rate of packets is denoted by λ, while the control flow rate of each individual controller is denoted by λ. In our heterogeneous network, m controllers are divided into n homogeneous groups, resulting in a service rate of m.μ for the whole system and n.μ for each individual controller.

The degree of heterogeneity is determined by the service rate of the controllers, and we define the system as heterogeneous when h > 2; otherwise, it is homogeneous. Previous research work related to SDN-IoT networks provided us with important variables and equations for this heterogeneous system.

Overall, our proposed solution aims to fill the research gaps in the SDN-IoT network's heterogeneity issues, while maintaining the QoS and increasing the security performance and mechanism:

$$h = \left[\frac{\mu_{max}}{\mu_{min}}\right] \tag{1}$$

The service rate is the average time of the packet or flow spent in the queue in waiting and time spent in utilizing resources of the controller. The average time flow, waiting time, and service rate of the controller are used to represent the complete network service rate. These general formulas are given below, and the changes in these formulas are also presented below with respect to the RA algorithm and the first come first serve algorithm (FCFS). Table 2 represent a description of variables that are used in formulas.

Generalized formulas of average service rate, the average time of flow remaining in the system, and per-flow average waiting time:

$$\mu_c = \sum_{i=1}^{m} p(C_i) \times \mu_i \tag{2}$$

$$T_q = \frac{\frac{1}{\mu}}{1-p} = \frac{1}{\mu_c - \lambda_c} \tag{3}$$

$$T_w = T_q\left(\frac{\lambda_c}{\mu_c}\right) \tag{4}$$

Again, these formulas are repeated below with modifications according to the RA (random algorithm):

$$\mu_(cr) = \frac{1}{2}\mu(h+1) \tag{5}$$

$$T_{qr} = \frac{1}{\frac{1}{2}\mu(h+a) - \lambda_c} \tag{6}$$

$$T_{wr} = T_{rq}\frac{\lambda_c}{\mu_{cr}} = \frac{\lambda_c}{\frac{1}{2}\mu(h+1)[\frac{1}{2}\mu(h+1)] - \lambda_c} \tag{7}$$

Below are, again, these formulas according to the FCFS algorithm (first come first serve):

$$\mu_{cf} = \sum_{i=1}^{m} P_f(C_i)\mu_i = \frac{1}{3}\mu(2h+1) \tag{8}$$

$$T_{qf} = \frac{1}{\frac{1}{3}\mu(2h+1) - \lambda_c} \tag{9}$$

$$T_{wf} = T_{qf}\frac{\lambda_c}{\mu_{cf}} = \frac{\lambda_c}{\frac{1}{3}\mu(2h+1)[\frac{1}{3}\mu(2h+1) - \lambda_c]} \tag{10}$$

Table 2. Important notations used in formulas and their description.

Notation	Description
m	Number of heterogeneous controllers
n	Number of homogeneous groups
μ	Service rate of whole system
$\mu_m in$	$min(\mu_1, \mu_2, \mu_3....\mu_m)$ minimum service rate among controllers
$\mu_m ax$	$max(\mu_1, \mu_2, \mu_3....\mu_m)$ maximum service rate among controllers
μ_i	Service rate of C_i th controller
h	Degree of heterogeneous
p	Stability of the system
μ_c	Service rate of controller
μ_{cr}	Service rate of system in RA
μ_{cf}	Service rate of system in FCFS algorithm
T_q	Average time flow remain in system
T_{pr}	Average time flow remain in system using RA
T_{qf}	Average time flow remain in system using FCFS algorithm
T_w	Average waiting time
T_{wr}	Average waiting time in the system using RA
T_{wf}	Average waiting time in the system using FCFS algorithm
C_i	Group of h heterogeneous controller
P_r	Probability of one controller chosen in a system when RA is used
P_f	Probability of one controller chosen in a system when FCFS algorithm is used
λ	Packet or flow arrival rate of the whole system
λ_c	Packet or flow arrival rate of individual controller

In the average service rate formula, the probability of the controller to be assessed is denoted by $p(C_i)$. In our model, as we discussed above, m heterogeneous controllers are divided into n groups based on their service rate, where each group consists of $G_n(n = 1, 2, 3, \ldots h)$ controllers. The service rate of each controller is represented by $n\mu$, thereby changing the system of heterogeneous controllers into an approximate homogeneous system.

Our model is based on M/M/1 queue discipline. To compute the average time of flow that remains in the system, Equation (3) is used.This equation is derived from the basic formulas of the M/M/1 queue model and the use of Little's law. The whole system incoming flow arrival rate is represented by λ, and for individual controllers, it is $\lambda_c = \lambda(1/m)$, where λ is a heterogeneous controller. For a system to remain in a stable state, the stability factor is $p = \lambda_c/\mu_c$. In any situation we need, $\lambda_c \leq \mu_c$ or $0 \leq p \geq 1$. The per-flow average waiting time is a product of the stability factor, and the average time of flow that remains in the system, as shown in Equation (4).

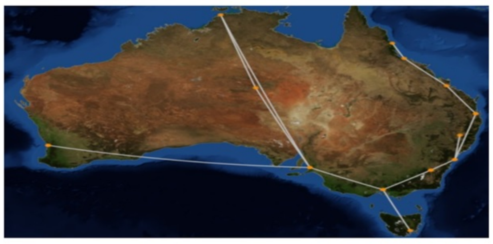
After analyzing the whole system, we implement this framework to fill some of its gaps. The framework is not tested using controller bandwidth and network delay. We introduce the bandwidth parameter and analyze the result. The term controller bandwidth refers to the bandwidth of the links between the central controller and peer controllers. Finally, we apply a simple mechanism to the known available bandwidth of the link, and if the selected controller's available bandwidth is more than the threshold point, then another controller of that same group is selected to serve the incoming flow. It is used to see whether this approach has an impact on the system or not.

5. Results and Discussion

This section comprises three sub-sections. The initial sub-section pertains to the experimental setup and execution, encompassing an extensive discussion of all the tools and software utilized. The second sub-section presents graphical results, while the final sub-section compasses a detailed analysis and interpretation of the aforementioned results.

5.1. Experimental Details

The complete detail of the experimental setup and tools are discussed in this section. This section covers the details of the tools and why they were selected in this experiment. First, as we know, the main objective of our proposed framework is to convert m heterogeneous controllers into n homogeneous groups in an SDN-IoT-based network. First, we must construct a network with multiple hosts and open-flow switches. In the experiment, two real-world topologies are created. Figure 5a,b show these networks' graphical representations on the map. The data of these networks are available on Internet Topology Zoo. The ITZ is a public online source, where data of different network real-world topologies are stored. The ITZ has many different network topologies which are publicly available from various network providers around the globe. Around 250 different network topologies' data are presented as undirected graphs in GraphML format. Extensible markup language (XML) is used to represent all topologies' data [53].


(**a**) ARRNET Network Topology Australia

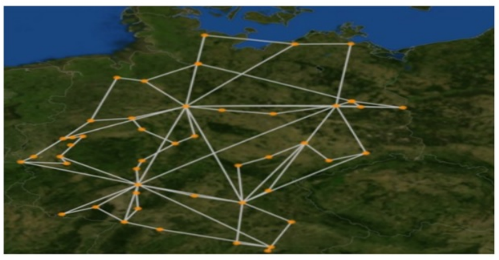

(**b**) DFN Network Topology Germany

Figure 5. AARNET/DFN network topologies.

The ITZ provides detailed data about each topology in its database. The details about nodes, their connectivity, bandwidth, and their location in terms of longitude and latitude are essential to construct the network. The two topologies which we used are AARNET and DFN. The details about how these data are used to construct the topology in Mininet are discussed in the next paragraph. Listing 1 shows important topology data in XML format.

There is much information in these structures, but to construct a topology, you need some important values from this XML structure. We exclude the meta-data and pass the important values to Mininet to generate the topology through the parser. Every node in the structure represents a switch, and edges represent a link between these nodes (switches). Each node has a unique ID number, and every node has some information. The important information about every node is its longitude and latitude. These are used to calculate the delay between two connected switches. The information in the edges tells us about the connection to two switches and their link bandwidth. The topology in Mininet is constructed through these important values of network topology. We are interested in these values in parsing Listing 2.

Listing 1. Important values in XML format of DFN topology.

```
<?xml version=" 1.0" encoding="utf -8"?>
<graphml ...>
  <key attr.name="key" attr.type="int" for
    = "edge" id="d36" />
    ....
    <graph edgedefault= "undirected">
    <data key="d0">2/01/11</data>
    <data key="d1">Germany</data>
    <data key="d2">Country</data>
    <data key="d3">DFN</data>
    ...
    <data key="d28">1</data>
    <node id="0">
    <data key="d29">1</data>
    <data key="d30">50.83333</data>
    <data key="d31">Germany</data>
    <data key="d32">0</data>
    <data key="d33">12.91667</data>
  <data key="d34">CHE</data>
  </node>
  ...
  <edge source="0" target="1">
    <data key="d35">e52</data>
    <data key="d36">0</data>
  </edge>
  ...
  </graph>
</graphml>
```

Listing 2. Mininet Python API transforming listing 1 to construct network topology.

```
#!/usr/bin/python
from mininet.topo import Topo
...
class GeneratedTop(Top):
  def __init__(self, **opts):
  #Initialize Topology
  Topo.__init__(self, **opts)
  # swithces first
  CHE = self.addSwithc('s0')
  ...
  # add new hosts
  CHE_host = self.addHost('h0')
  ...
  # add edges between switches and corresponding hosts
  self.addLink (CHE, CHE_host)
  ...   # add edges between switches
  self.addLink (CHE, LEI, bw=10, delay='0.348009503ms')
  ...
topos =  'generated' ; (lambda: GeneratedTopo())
...
if __name__ == '__main__':
  sshd(setupNetwork())
```

Mininet is a lightweight emulation tool built for the emulation of networks with the support of an SDN controller. The bandwidth and delay parameters of the links can be set in Mininet. In our scenario, the delay between switches is the time a packet takes between two connected nodes. Latitude and longitude data are used to calculate the delay between nodes. The following formulas [54] are used to find the delay time between two nodes. Table 3 shows the important notations used in Equations (11) and (12):

$$dist(SF, DP) = cos^{-1}\{sin(La_{DP})\cdot sin(La_{SP}) + cos(La_{DP})\cdot cos(La_{SP})\cdot cos(La_{DP} - La_{SP})\}\cdot r \tag{11}$$

$$t_l = \frac{dst(SP, DP)}{v_l} \tag{12}$$

Table 3. Important notations used to calculate delay and their description.

Notation	**Descriptions**
$Dist$	Distance
SP	Source point (source node)
DP	Destination Point (end node)
La	Latitude in radians
Lo	Longitude in radians
r	Radius (6,378,137 m)
v_l	Velocity of signal (1.97×10^8) m/s

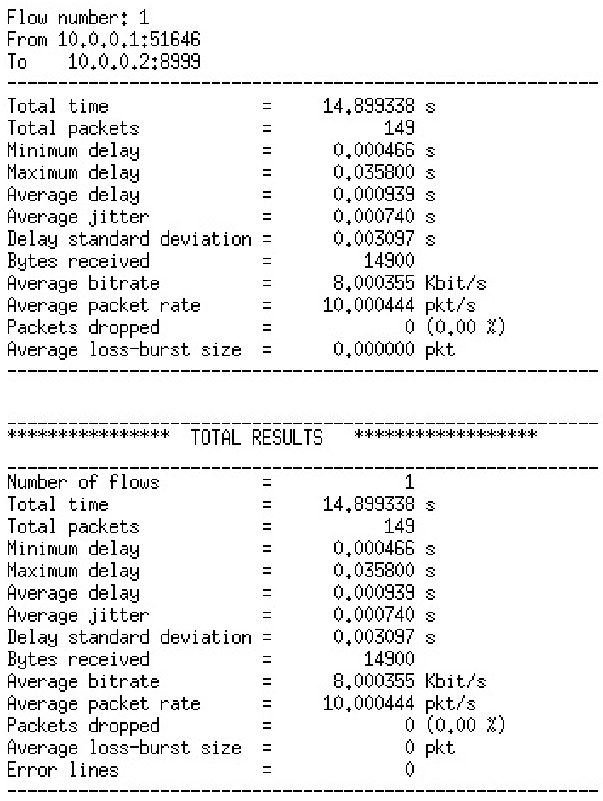
The spherical law of cosine is used to calculate the distance of topology in Equation (11). The radius value is assumed. The nodes are connected via a fiber optic link, so the signal speed is taken as the speed of light and divided by the reflective factor of 1.52. After the successful construction of topology in Mininet, a tool is needed to evaluate the topology through real distributed network traffic.D-ITG (distributed internet traffic generator) is one tool that replicates real packets of various protocols. Through D-ITG, different packet streams can be generated, and statistics are collected on the logging server. In D-ITG, there are different modules responsible for their function.

In the network, one host has to act as a log to collect all the data regarding D-ITG traffic. The D-ITG log sample information is shown in Figure 6. One host sends a UDP packet to another host, and in the end, the decoder analyzes the data collected by log. D-ITG also provides the feature of analyzing information in real time. In our case, we wrote bash files for TCP and UDP traffic. When Mininet topology is created, every host also accesses the shell. On another terminal, we ran the bash file of D-ITG so traffic can be generated in the network.

Finally, let us discuss the SDN controller, the crucial component of the system. The ONOS controller is selected for our study. The reason is that the author of our base paper also uses this controller, and it is an open-source controller. The second ONOS controller is compatible with Mininet. It is an open-source controller providing next-generation SDN/NFV solutions. There is no major challenge in connecting the ONOS controller with the Mininet topology. It is recommended for clustering or making groups of controllers, so in our method we have to create groups of controllers, so this will be the best choice for this experiment.

In [13], they created VM for every controller, and one VM machine is used as a click modular router [55], which is used to develop the communication fabric of controllers. Figure 7 shows the concept; they use Core i7—7700k @4:20 GHz with 64 GB of RAM for the experiments. Virtual machines are configured with 4-core CPUs and 12 GB of RAM. Also, the whole experimental network is constructed over a Dell Power Edge M640 Server consisting of two Intel Xeon Gold 6126 @ 2.6 GHz processors and 384 GB (6 × 64) by [13]. Due to limited computational resources, we changed some experimental strategies.

It is similar to the original one with a bit of change. We created 20 peer controllers for the experiment and one central controller. Despite assigning every controller a complete resource equal to a computer, we made all controllers on one system. All controllers are connected in topology; between these peer controllers and the central controller, one OpenFlow switch is connected to provide communication. Figure 7 shows our concept. This switch is the replacement of the click modular router.



```
Flow number: 1
From 10.0.0.1:51646
To    10.0.0.2:8999
----------------------------------------------------------
Total time                =     14.899338 s
Total packets             =           149
Minimum delay             =      0.000466 s
Maximum delay             =      0.035800 s
Average delay             =      0.000939 s
Average jitter            =      0.000740 s
Delay standard deviation =      0.003097 s
Bytes received            =         14900
Average bitrate           =      8.000355 Kbit/s
Average packet rate       =     10.000444 pkt/s
Packets dropped           =             0 (0.00 %)
Average loss-burst size   =      0.000000 pkt
----------------------------------------------------------

----------------------------------------------------------
****************  TOTAL RESULTS   ******************
----------------------------------------------------------
Number of flows           =             1
Total time                =     14.899338 s
Total packets             =           149
Minimum delay             =      0.000466 s
Maximum delay             =      0.035800 s
Average delay             =      0.000939 s
Average jitter            =      0.000740 s
Delay standard deviation =      0.003097 s
Bytes received            =         14900
Average bitrate           =      8.000355 Kbit/s
Average packet rate       =     10.000444 pkt/s
Packets dropped           =             0 (0.00 %)
Average loss-burst size   =             0 pkt
Error lines               =             0
----------------------------------------------------------
```

Figure 6. D-ITG log showing information about UDP packet transfer from H1 to H2.

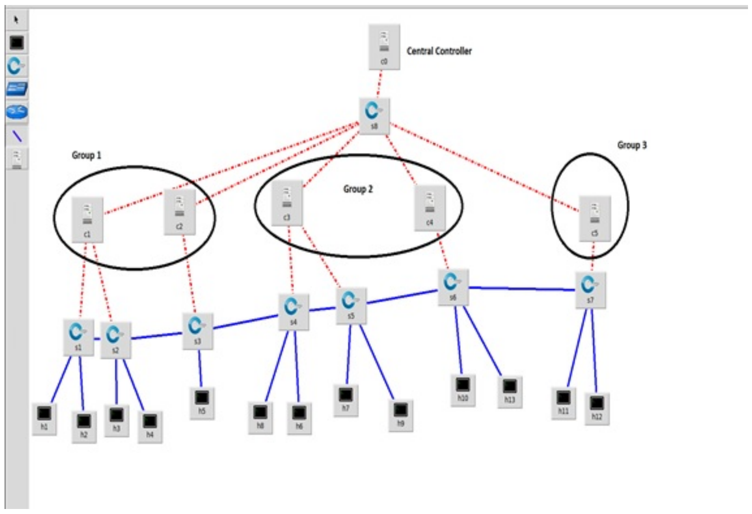


Figure 7. Peer controllers forming groups on the basis of service rate and connected to the central controller via OpenFlow switch.

At first, the SDN controller collects the service rate of peer controllers. The service rate to peer controllers is assigned from 1 k to 30 k/s. After collecting the service rate, the central controller decides whether the network is heterogeneous. It is determined based on the service rate of peer controllers. If the ratio of maximum and minimum service rates is less than two, the network is considered a homogeneous network and heterogeneous otherwise. If the system is homogeneous, the central controller randomly selects any controller to fulfill the flow request. In case the system is heterogeneous, then central control forms groups of peer controllers on the basis of their service rate. Similar service rate controllers are combined to form a homogeneous group. The central controller first receives the flow of each packet, then according to the algorithm (RA or FCFS), it selects a peer controller to process the packet flow. The service rate data of the peer controller are collected through a parser using JSON format. The service rate is collected after a specific interval of time. The interval between collections of peer service rate can also depend upon the average throughput of the system or degree of heterogeneity. We applied both strategies to set the "service rate collection interval time" in this study. When the interval time of service rate is dependent upon throughput or degree of heterogeneity, it will vary according to these parameters. Consider that after collecting the service rate and forming groups, after a specific interval, it again collects the service rate; now it checks if the system's throughput is increased, decreased, or remains the same.In the case of the same time interval, if the throughput increases, then the interval of time is increased to collect the service rate, being either doubled or increased by a constant factor. Finally, if the throughput is decreased, then the interval time is cut down to half of its current value.

We include a minor change in the central controller algorithm to see improvement in the system. In the last case, the selection of a peer controller also depends upon the available bandwidth of that controller. The central controller also collects link utilization and takes the difference between the bandwidth of that link and current link utilization. The result is the available bandwidth of that link. In this experiment, we only took links of the peer controller connected with a switch which provides a connection to the central controller. Link information is gathered through OpenFlow functions, which discover the topology and provide information.

Last, the controller default forwarding rules are used in the data plane. Most of the work is on the control plane. The Cbench tool is used to measure the throughput of controllers. It is a tool designed for OpenFlow switch and SDN controller to emulate their performance through some benchmarks. Test cases for the experiment are discussed in the result subsection. Table 4 discusses our experimental setup machine and tools languages. In the following steps, the central controller Algorithm 1 is described.

Algorithm 1 Controller selection algorithm.

```
Input: Peer controller service rates $\mu_i$ in JSON format
Output: Selected controller to fulfill incoming flow request
1. Calculate degree of heterogeneity $h$ using Equation (1)
2. If $h < 2$, then treat the whole system as homogeneous and select any controller to fulfill incoming flow request
3. Else, form groups on the basis of their service rate and choose the controller using RA or FCFS algorithm to fulfill the request
4. Use default forwarding rules of the controller to send packets from source to destination
```

Table 4. Details of hardware and software used in experiment.

Items	Description
Computer System	Inter Core i7-10700K 11th generation, 3.8 GHz, 32 GB of ram, 500 GB SSD
Operating System	Linux (Ubuntu 21.04)
Software Tools	Mininet, Cbench
Controller	ONOS
Languages, script	Python, JSON, XML, Bash
Data collection source	Internet Topology ZOO

5.2. Results: Experimental Setup

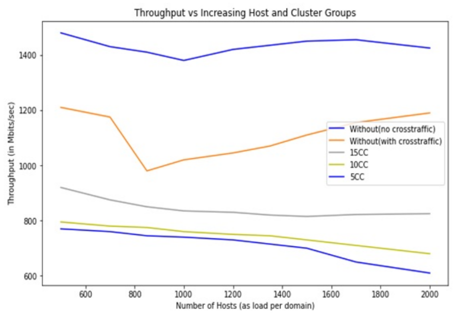
In this experiment, we use two real-world topologies, AARNET and DFN. Both topologies are tested in the proposed framework. TCP and UDP protocols are used in the experiment. Various test cases are applied in this experiment. The results are based on various test cases. We have 20 peer controllers in the network. We make their group according to their service rate. In the experiment, both topologies are tested under different test cases. First, we divide our result and test case according to the protocol. We use TCP- and UDP-based protocol traffic. Another variable that we change in our test case is the number of hosts starting from 500 and reaching 2000 hosts. Cross-traffic and without cross-traffic conditions are also tested. Another test case we test is forming permanent groups without changing their service rate. For that, 5, 10, 15 groups are formed to perform this test case. Without fixing the cluster group number, we also run experiments. In this scenario, the service rate of the controllers are not fixed, so forming cluster groups can vary. Finally, we test this framework under different bandwidths of the controller. The bandwidth of controllers is set at 2 Gbps, 5 Gbps 10 Gbps. We also test a case where random bandwidths are assigned to different controllers. The last test case is about bandwidth utilization when a simple mechanism is implemented to test AARNET at a 10 Gbps bandwidth. Below are the results of the experiment, and an analysis of the result is presented in the next subsection of this paper.

5.3. Discussion of Experimental Result

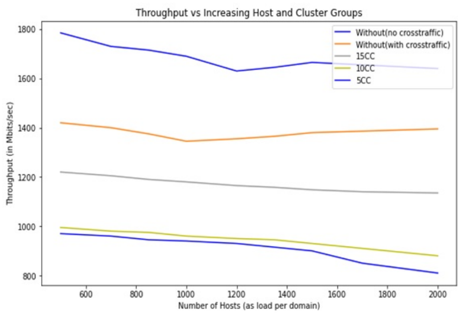
Based on the simulation, the experiment's findings were made. Changing the host count and bandwidth allowed us to obtain different outcomes. Additionally, a simulation with a fixed homogeneous group count was conducted. Results of AARNET on both protocols are shown in Figures 8 and 9. In Figures 10 and 11, the DFN network topology results for both protocols are shown. First, we analyzed the result of both topologies in both protocols at 10 GB bandwidth. The results in Figures 8c, 9c, 10c and 11c are very similar to those obtained by [13]. They consider the unlimited bandwidth of the controller, but in Mininet, if the controller bandwidth link is not defined, it takes around 10 GB bandwidth by default. The throughput of both topologies decreases when their controller bandwidth is decreased. The important point is that in every bandwidth case, the throughput follows the same linearity means; as the number of hosts increases, the throughput decreases. The throughput is reduced from 30% to 35% when a 2 Gbps bandwidth is set. For 5 Gbps, the bandwidth is decreased by 16% to around 20% as compared to the results of Sood et al. [13]. The 10 Gbps results are approximately the same. Based on the results, it seems that heterogeneity affects the QoS.

The decrease in throughput is linear, which means that the throughput does not deviate too much as the number of hosts increases. The other thing we observed is that as the number of cluster groups is increased in the system, the throughput is greater. When we fix the service rate and form constant groups, for example, 5, 10 15, the throughput is increased when the number of groups is greater, which means that this strategy impacts the heterogeneous network performance.

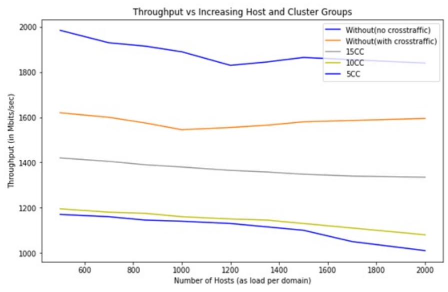
When we do not fix the number of groups, the throughput is maximal in every test case. It indicates that grouping based on the service rate and, after some interval time, changing groups based on their current service rate is the most efficient way. We also implement one method based on throughput to change the interval time of the service rate, but it does not show any significant changes in the average throughput of the system. We need further investigation to see this method's effectiveness on other controller resources. When the bandwidth consideration technique is applied, we see some increase in the throughput when the hosts increase and and decrease in the available bandwidth value due to more traffic. In this scenario, the throughput is increased from 2% to 4% as shown in Figure 12 compared to the results of K. Sood et al. [13]. It shows that there is potential in this framework to become more efficient in terms of QoS while alleviating heterogeneity. Our proposed model focuses on all traffic, not only on control traffic, as presented in [28,29].



(**a**)

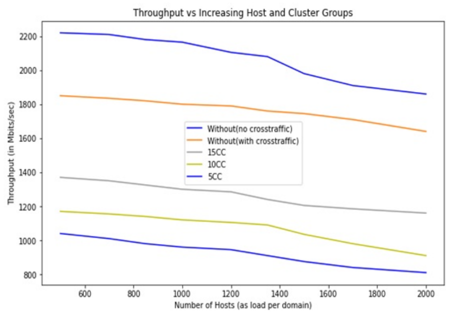


(**b**)

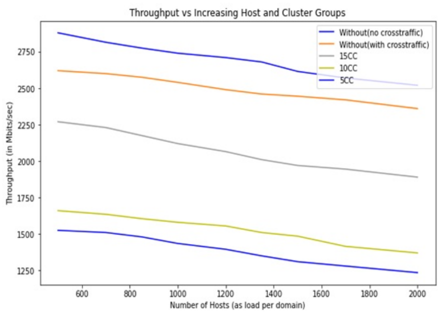


(**c**)

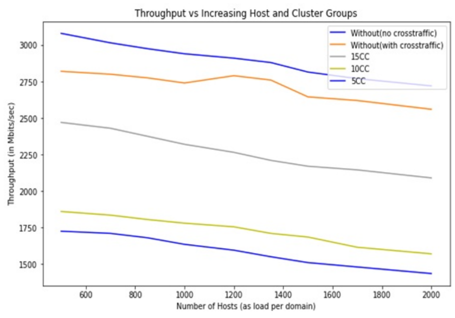
Figure 8. AARNET throughput at different bandwidths TCP. (**a**) AARNET throughput at 2 GB bandwidth TCP. (**b**) AARNET throughput at 5 GB bandwidth TCP. (**c**) AARNET throughput at 10 GB bandwidth TCP.



(**a**)

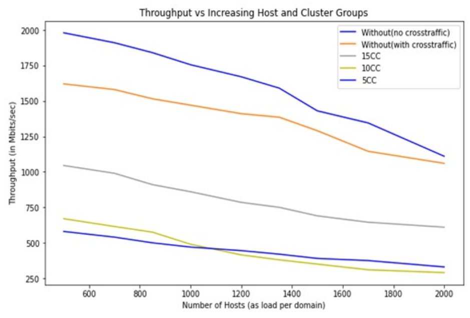


(**b**)

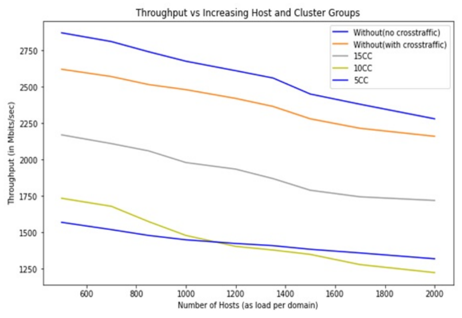


(**c**)

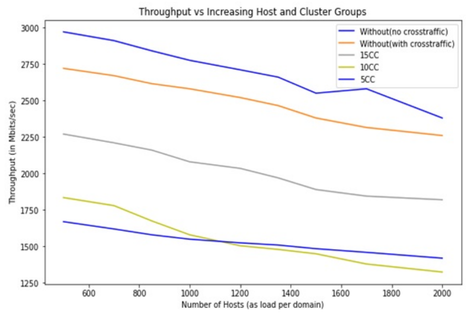
Figure 9. AARNET throughput at different bandwidths UDP. (**a**) AARNET throughput at 2 GB bandwidth UDP. (**b**) AARNET throughput at 5 GB bandwidth UDP. (**c**) AARNET throughput at 10 GB bandwidth UDP.



(**a**)

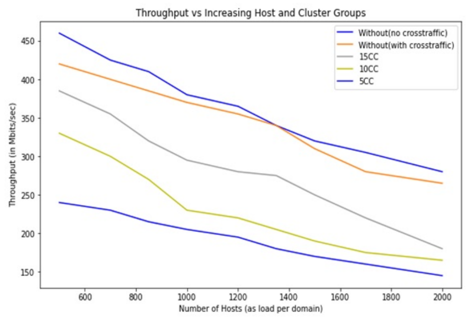


(**b**)

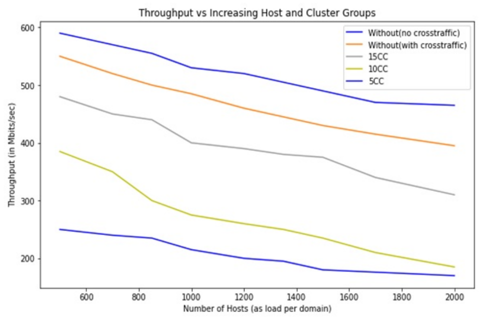


(**c**)

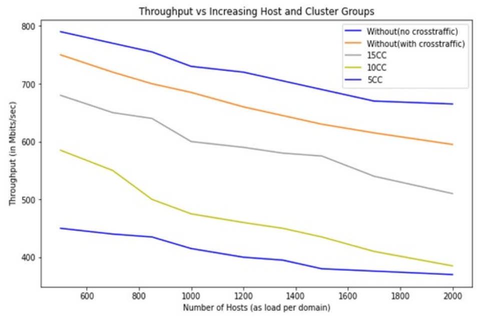
Figure 10. DFN throughput at different bandwidths TCP. (**a**) DFN throughput at 2 GB bandwidth TCP. (**b**) DFN throughput at 5 GB bandwidth TCP. (**c**) DFN throughput at 10 GB bandwidth TCP.



(**a**)

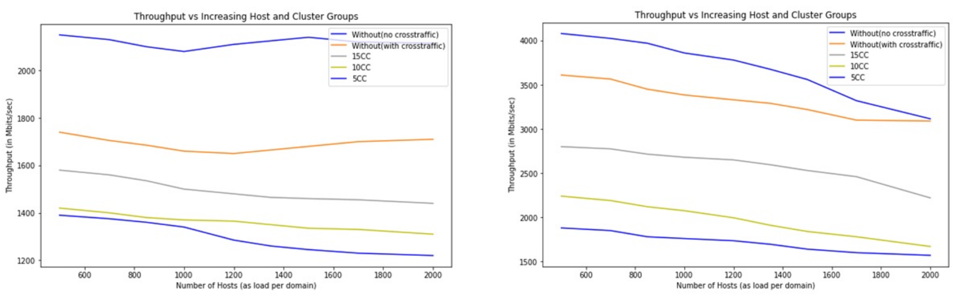


(**b**)



(**c**)

Figure 11. DFN throughput at different bandwidths UDP. (**a**) DFN throughput at 2 GB bandwidth UDP. (**b**) DFN throughput at 5 GB bandwidth UDP. (**c**) DFN throughput at 10 GB bandwidth UDP.



(**a**) (**b**)

Figure 12. AARNET throughput at 10 GB bandwidth TCP/UDP with bandwidth utilization technique. (**a**) AARNET throughput at 10 GB bandwidth TCP with bandwidth utilization technique. (**b**) AARNET throughput at 10 GB bandwidth UDP with bandwidth utilization technique.

In the last test case, we do not fix the same bandwidth of each controller. We randomly assign different bandwidths to each controller. The result generated in this test case is changed every time, and there is much variation in the throughput compared to the fixed bandwidth test cases. It shows a need to understand this situation further and find methods to stabilize the system's throughput. After analyzing the whole system for the delay between controllers, we can say that controller placement is critical in these topologies.

6. Conclusions

In conclusion, the use of SDN to address heterogeneity the in IoT networks shows great potential for improving QoS and enhancing security. In this study, we developed a framework that converts heterogeneous controllers into homogeneous ones based on their service rate, which was tested using the RA FCFS algorithm. Our experiments showed that when the bandwidth is shared equally across all controllers, there is less throughput variation. However, further study is required to investigate different bandwidth allocations to controllers, stability of throughput, and other affected parameters. The inclusion of delay and bandwidth as metrics may also enhance the framework's performance.

As for future work, we suggest investigating the delay between controllers, testing the framework's performance in actual IoT networks, comparing it with other similar SDN-based methods, developing bandwidth-on-demand solutions, addressing the central point of failure issue, implementing traffic-specific flow, and studying the framework for other parameters of QoS. These efforts would help advance the framework and its potential applications in the field.

Author Contributions: Conceptualization, A.Z. and F.S.; methodology, A.Z. and F.S.; software, H.J.S.; validation, A.Z., F.S., M.A. and H.J.S.; formal analysis, A.O.I., M.A. and M.E.; investigation, M.A. and M.E.; resources, A.O.I., M.A. and M.E.; writing—original draft preparation, A.Z. and F.S.; writing—review and editing, A.Z., F.S. and H.J.S.; visualization, A.O.I., M.A. and M.E.; supervision, F.S.; project administration, A.O.I., M.A. and M.E.; funding acquisition, M.A. and M.E. All authors have read and agreed to the published version of the manuscript.

Funding: This project is funded by Princess Nourah bint Abdulrahman University Researchers Supporting Project number (PNURSP2023R383), Princess Nourah bint Abdulrahman University, Riyadh, Saudi Arabia.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Acknowledgments: This work was supported through Princess Nourah bint Abdulrahman University Researchers Supporting Project number (PNURSP2023R383), Princess Nourah bint Abdulrahman University, Riyadh, Saudi Arabia.

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

References

1. van der Meulen, R. *Gartner Says 4.9 Billion Connected "Things" Will Be in Use in 2015*; Technical report; Gartner: Singapore, 2014.
2. Vailshery, L.S. *Number of Internet of Things (IoT) Connected Devices Worldwide from 2019 to 2021, with Forecasts from 2022 to 2030*; Technical report; Statista: Hamburg, Germany, 2022.
3. Montazerolghaem, A.; Yaghmaee, M.H. Load-balanced and QoS-aware software-defined Internet of Things. *IEEE Internet Things J.* **2020**, *7*, 3323–3337. [CrossRef]
4. Čolaković, A.; Hadžialić, M. Internet of Things (IoT): A review of enabling technologies, challenges, and open research issues. *Comput. Netw.* **2018**, *144*, 17–39. [CrossRef]
5. Ahmed, A.I.A.; Gani, A.; Ab Hamid, S.H.; Abdelmaboud, A.; Syed, H.J.; Mohamed, R.A.A.H.; Ali, I. Service management for IoT: Requirements, taxonomy, recent advances and open research challenges. *IEEE Access* **2019**, *7*, 155472–155488. [CrossRef]
6. Khan, R.; Khan, S.U.; Zaheer, R.; Khan, S. Future internet: The internet of things architecture, possible applications and key challenges. In Proceedings of the 2012 10th International Conference on Frontiers of Information Technology, Islamabad, Pakistan, 17–19 December 2012; IEEE: Piscataway, NJ, USA, 2012; pp. 257–260.
7. Ungurean, I.; Gaitan, N.C. Data distribution service for real-time systems—A solution for the internet of things environments. *Ann. Univ. Dunarea Jos Galati Fascicle Ii Math. Phys. Theor. Mech.* **2015**, *38*, 72–76.
8. Ouamri, M.A.; Barb, G.; Singh, D.; Alexa, F. Load Balancing Optimization in Software-Defined Wide Area Networking (SD-WAN) using Deep Reinforcement Learning. In Proceedings of the 2022 International Symposium on Electronics and Telecommunications (ISETC), Timisoara, Romania, 10–11 November 2022; pp. 1–6. [CrossRef]
9. Ouamri, M.A.; Azni, M.; Singh, D.; Almughalles, W.; Muthanna, M. Request delay and survivability optimization for software defined-wide area networking (SD-WAN) using multi-agent deep reinforcement learning. *Trans. Emerg. Telecommun. Technol.* **2023**, *early view*. [CrossRef]
10. Kesavan, M.; Prabhu, J. A survey, design and analysis of IoT security and QoS challenges. *Int. J. Inf. Syst. Model. Des.* **2018**, *9*, 48–66. [CrossRef]
11. Farris, I.; Taleb, T.; Khettab, Y.; Song, J. A Survey on Emerging SDN and NFV Security Mechanisms for IoT Systems. *IEEE Commun. Surv. Tutor.* **2019**, *21*, 812–837. [CrossRef]
12. Yang, Z.; Lui, J.C. Security adoption and influence of cyber-insurance markets in heterogeneous networks. *Perform. Eval.* **2014**, *74*, 1–17. [CrossRef]
13. Sood, K.; Karmakar, K.K.; Yu, S.; Varadharajan, V.; Pokhrel, S.R.; Xiang, Y. Alleviating heterogeneity in SDN-IoT networks to maintain QoS and enhance security. *IEEE Internet Things J.* **2019**, *7*, 5964–5975. [CrossRef]
14. Marques, G.; Garcia, N.; Pombo, N. A survey on IoT: Architectures, elements, applications, QoS, platforms and security concepts. In *Advances in Mobile Cloud Computing and Big Data in the 5G Era*; Springer: Cham, Switzerland, 2017; pp. 115–130.
15. Singh, M.; Baranwal, G. Quality of service (qos) in internet of things. In Proceedings of the 2018 3rd International Conference on Internet of Things: Smart Innovation and Usages (IoT-SIU), Bhimtal, India, 23–24 February 2018; IEEE: Piscataway, NJ, USA, 2018; pp. 1–6.
16. Duan, Q.; Ansari, N.; Toy, M. Software-defined network virtualization: An architectural framework for integrating SDN and NFV for service provisioning in future networks. *IEEE Netw.* **2016**, *30*, 10–16. [CrossRef]
17. Matias, J.; Garay, J.; Toledo, N.; Unzilla, J.; Jacob, E. Toward an SDN-enabled NFV architecture. *IEEE Commun. Mag.* **2015**, *53*, 187–193. [CrossRef]
18. Soto-Cordova, M.M.; Chavez-Hidalgo, G.; Ñiquen-Ortega, R. Approach of Performance Analysis for Controllers of Software Defined Networking. In Proceedings of the 2019 Congreso Internacional de Innovación y Tendencias en Ingenieria (CONIITI), Bogota, Colombia, 2–4 October 2019; IEEE: Piscataway, NJ, USA, 2019; pp. 1–6.
19. Zhang, L.; Estrin, D.; Burke, J.; Jacobson, V.; Thornton, J.D.; Smetters, D.K.; Zhang, B.; Tsudik, G.; Massey, D.; Papadopoulos, C.; et al. *Named Data Networking (NDN) Project*; NDN Technical Report NDN-0001: Hong Kong, China, 2010; Volume 157, p. 158.
20. Campbell, A.T.; De Meer, H.G.; Kounavis, M.E.; Miki, K.; Vicente, J.B.; Villela, D. A survey of programmable networks. *ACM SIGCOMM Comput. Commun. Rev.* **1999**, *29*, 7–23. [CrossRef]
21. Karakus, M.; Durresi, A. Quality of service (QoS) in software defined networking (SDN): A survey. *J. Netw. Comput. Appl.* **2017**, *80*, 200–218. [CrossRef]
22. Braden, R.; Zhang, L.; Berson, S.; Herzog, S.; Jamin, S. Resource Reservation Protocol (RSVP)–Version 1 Functional Specification. Technical Report, 1997. https://datatracker.ietf.org/doc/html/rfc2205 (accessed on 24 June 2023).
23. Blake, S.; Black, D.; Carlson, M.; Davies, E.; Wang, Z.; Weiss, W. *Rfc2475: An Architecture for Differentiated Service*; ACM: New York, NY, USA, 1998.
24. Rosen, E.; Viswanathan, A.; Callon, R. *Multiprotocol Label Switching Architecture*; Technical report; The Internet Society: Reston, VA, USA, 2001.
25. Yu, C.; Lumezanu, C.; Zhang, Y.; Singh, V.; Jiang, G.; Madhyastha, H.V. Flowsense: Monitoring network utilization with zero measurement cost. In Proceedings of the Passive and Active Measurement: 14th International Conference, PAM 2013, Hong Kong, China, 18–19 March 2013; Proceedings 14; Springer: Berlin/Heidelberg, Germany, 2013; pp. 31–41.

26. Chowdhury, S.R.; Bari, M.F.; Ahmed, R.; Boutaba, R. Payless: A low cost network monitoring framework for software defined networks. In Proceedings of the 2014 IEEE Network Operations and Management Symposium (NOMS), Krakow, Poland, 5–9 May 2014; IEEE: Piscataway, NJ, USA, 2014; pp. 1–9.
27. Bianco, A.; Giaccone, P.; Mashayekhi, R.; Ullio, M.; Vercellone, V. Scalability of ONOS reactive forwarding applications in ISP networks. *Comput. Commun.* **2016**, *102*, 130–138. [CrossRef]
28. Yu, B.; Yang, G.; Yoo, C. Comprehensive Prediction Models of Control Traffic for SDN Controllers. In Proceedings of the 2018 4th IEEE Conference on Network Softwarization and Workshops (NetSoft), Montreal, QC, Canada, 25–29 June 2018; pp. 262–266. [CrossRef]
29. Cui, J.; Lu, Q.; Zhong, H.; Tian, M.; Liu, L. A Load-Balancing Mechanism for Distributed SDN Control Plane Using Response Time. *IEEE Trans. Netw. Serv. Manag.* **2018**, *15*, 1197–1206. [CrossRef]
30. Wang, P.; Xu, H.; Huang, L.; Qian, C.; Wang, S.; Sun, Y. Minimizing Controller Response Time Through Flow Redirecting in SDNs. *IEEE/ACM Trans. Netw.* **2018**, *26*, 562–575. [CrossRef]
31. Zhong, C.; Zhu, Z.; Huang, R.G. Study on the IoT architecture and access technology. In Proceedings of the 2017 16th International Symposium on Distributed Computing and Applications to Business, Engineering and Science (DCABES), Anyang, China, 13–16 October 2017; IEEE: Piscataway, NJ, USA, 2017; pp. 113–116.
32. Bizanis, N.; Kuipers, F.A. SDN and virtualization solutions for the Internet of Things: A survey. *IEEE Access* **2016**, *4*, 5591–5606. [CrossRef]
33. Ojo, M.; Adami, D.; Giordano, S. A SDN-IoT architecture with NFV implementation. In Proceedings of the 2016 IEEE Globecom Workshops (GC Wkshps), Washington, DC, USA, 4–8 December 2016; IEEE: Piscataway, NJ, USA, 2016; pp. 1–6.
34. Islam, M.J.; Mahin, M.; Roy, S.; Debnath, B.C.; Khatun, A. Distblacknet: A distributed secure black sdn-iot architecture with nfv implementation for smart cities. In Proceedings of the 2019 International Conference on Electrical, Computer and Communication Engineering (ECCE), Cox'sBazar, Bangladesh, 7–9 February 2019; IEEE: Piscataway, NJ, USA, 2019; pp. 1–6.
35. Chavhan, S.; Gupta, D.; Chandana, B.; Khanna, A.; Rodrigues, J.J. IoT-based context-aware intelligent public transport system in a metropolitan area. *IEEE Internet Things J.* **2019**, *7*, 6023–6034. [CrossRef]
36. De Gante, A.; Aslan, M.; Matrawy, A. Smart wireless sensor network management based on software-defined networking. In Proceedings of the 2014 27th Biennial Symposium on Communications (QBSC), Kingston, ON, Canada, 1–4 June 2014; IEEE: Piscataway, NJ, USA, 2014; pp. 71–75.
37. Xu, Q.; Zhao, J. A WSN architecture based on SDN. In Proceedings of the 4th International Conference on Information Systems and Computing Technology, Shanghai, China, 22–23 December 2016; Atlantis Press: Amsterdam, The Netherlands, 2016; pp. 159–163.
38. Luo, T.; Tan, H.P.; Quek, T.Q. Sensor OpenFlow: Enabling software-defined wireless sensor networks. *IEEE Commun. Lett.* **2012**, *16*, 1896–1899. [CrossRef]
39. Galluccio, L.; Milardo, S.; Morabito, G.; Palazzo, S. SDN-WISE: Design, prototyping and experimentation of a stateful SDN solution for WIreless SEnsor networks. In Proceedings of the 2015 IEEE Conference on Computer Communications (INFOCOM), Hong Kong, China, 26 April–1 May 2015; IEEE: Piscataway, NJ, USA, 2015; pp. 513–521.
40. Thubert, P.; Palattella, M.R.; Engel, T. 6TiSCH centralized scheduling: When SDN meet IoT. In Proceedings of the 2015 IEEE Conference on Standards for Communications and Networking (CSCN), Tokyo, Japan, 28–30 October 2015; IEEE: Piscataway, NJ, USA, 2015; pp. 42–47.
41. Yiakoumis, Y.; Yap, K.K.; Katti, S.; Parulkar, G.; McKeown, N. Slicing home networks. In Proceedings of the 2nd ACM SIGCOMM Workshop on Home Networks, Toronto, ON, Canada, 15 August 2011; pp. 1–6.
42. Batalle, J.; Riera, J.F.; Escalona, E.; Garcia-Espin, J.A. On the implementation of NFV over an OpenFlow infrastructure: Routing function virtualization. In Proceedings of the 2013 IEEE SDN for Future Networks and Services (SDN4FNS), Trento, Italy, 11–13 November 2013; IEEE: Piscataway, NJ, USA, 2013; pp. 1–6.
43. Tran, H.A.; Tran, D.; Nguyen, L.G.; Ha, Q.T.; Tong, V.; Mellouk, A. SHIOT: A novel SDN-based framework for the heterogeneous Internet of Things. *Informatica* **2018**, *42*, 313–323. [CrossRef]
44. Peros, S.; Janjua, H.; Akkermans, S.; Joosen, W.; Hughes, D. Dynamic QoS support for IoT backhaul networks through SDN. In Proceedings of the 2018 Third International Conference on Fog and Mobile Edge Computing (FMEC), Barcelona, Spain, 23–26 April 2018; IEEE: Piscataway, NJ, USA, 2018; pp. 187–192.
45. Theodorou, T.; Violettas, G.; Valsamas, P.; Petridou, S.; Mamatas, L. A multi-protocol software-defined networking solution for the Internet of Things. *IEEE Commun. Mag.* **2019**, *57*, 42–48. [CrossRef]
46. Deng, S.; Gao, X.; Lu, Z.; Gao, X. Packet injection attack and its defense in software-defined networks. *IEEE Trans. Inf. Forensics Secur.* **2017**, *13*, 695–705. [CrossRef]
47. Poremba, S. *Endpoint Security: Preventing Threats on Devices Connected to your Network.*
48. Yao, L.; Hong, P.; Zhou, W. Evaluating the controller capacity in software defined networking. In Proceedings of the 2014 23rd International Conference on Computer Communication and Networks (ICCCN), Shanghai, China, 4–7 August 2014; IEEE: Piscataway, NJ, USA, 2014; pp. 1–6.
49. Yao, G.; Bi, J.; Li, Y.; Guo, L. On the capacitated controller placement problem in software defined networks. *IEEE Commun. Lett.* **2014**, *18*, 1339–1342. [CrossRef]

50. Yu, S.; Doss, R.; Thapngam, T.; Qian, D. A transformation model for heterogeneous servers. In Proceedings of the 2008 10th IEEE International Conference on High Performance Computing and Communications, Dalian, China, 25–27 September 2008; IEEE: Piscataway, NJ, USA, 2008; pp. 665–671.
51. Si, L.K. *Queueing Systems*; Volume I: Theory; 1976.
52. Mahmood, K.; Chilwan, A.; Østerbø, O.; Jarschel, M. Modelling of OpenFlow-based software-defined networks: The multiple node case. *IET Netw.* **2015**, *4*, 278–284. [CrossRef]
53. Knight, S.; Nguyen, H.X.; Falkner, N.; Bowden, R.; Roughan, M. The internet topology zoo. *IEEE J. Sel. Areas Commun.* **2011**, *29*, 1765–1775. [CrossRef]
54. Großmann, M.; Schuberth, S.J. Auto-Mininet: Assessing the Internet topology zoo in a software-defined network emulator. In *Messung, Mellierung un Bewertung von Rechensystemen (MMBnet)*; Springer: Berlin/Heidelberg, Germany, 2013; Volume 7, pp. 1–10.
55. Kohler, E.; Morris, R.; Chen, B.; Jannotti, J.; Kaashoek, M.F. The Click modular router. *ACM Trans. Comput. Syst.* **2000**, *18*, 263–297. [CrossRef]