information

MDPI

Article

Impact of Southbound Expansion on Clustered OpenFlow Software-Defined Network Controller Synchronisation Using ODL and ONOS

Egodahettiarachchige Don Sarada Indumini Hettiarachchi , Nurul I. Sarkar * and Jairo Gutierrez

Computer Science and Software Engineering, Auckland University of Technology, Auckland 1010, New Zealand; spy5359@autuni.ac.nz (E.D.S.I.H.); jairo.gutierrez@aut.ac.nz (J.G.)
* Correspondence: nurul.sarkar@aut.ac.nz; Tel.: +64-211-758390

Abstract: The clustering methods of Software-Defined Networking (SDN) have gained popularity due to their ability to offer improved scalability, consistency, dependability, and load balancing within overlay networks and SDN partitions. This paper delved into the effects of increasing the number of OpenFlow-enabled southbound devices on the establishment and coordination of SDN-controller clusters. Specifically, we examined the volume of east–west cluster packet communications concerning the number of southbound devices within the topology. Many research studies have focused on bandwidth and the number of bytes in east–west communication. While bandwidth refers to the maximum rate at which data can be transferred, and the number of bytes reflects the volume of data being transmitted, the number of packet communications directly influences the efficiency and responsiveness of network operations. Our investigation encompassed the impact of SDN controller-to-controller communication within the cluster concerning the rising number of OpenFlow switches connected to various topologies, including tree (star-bus network), linear, and torus configurations. This study provided data on communication patterns within Open Network Operating Systems (ONOS) and OpenDaylight (ODL) clusters, revealing differing levels of controller communication with southbound network expansions. We evaluated the scalability of ODL and ONOS controllers by scrutinising the effect of increasing the number of southbound devices on the control communication volume. Our analysis revealed varied communication patterns within ONOS and ODL clusters, resulting in different volumes of control communication with southbound expansions. The findings indicated that in small-to-medium-sized SDNs, ODL outperformed ONOS, notably with faster cluster discovery. Conversely, ONOS demonstrated greater efficiency in larger networks owing to its centralised communication architecture. Finally, we provide recommendations for selecting the most suitable controllers based on the size of southbound networks, aiming to provide practical guidelines for optimal network performance.

Keywords: software-defined networking; southbound interface; controller clustering; east–west communication; leader-based models; ODL clustering; ONOS clustering

check for updates

Citation: Hettiarachchi, E.D.S.I.; Sarkar, N.I.; Gutierrez, J. Impact of Southbound Expansion on Clustered OpenFlow Software-Defined Network Controller Synchronisation Using ODL and ONOS. *Information* **2024**, *15*, 440. https://doi.org/10.3390/info15080440

Academic Editor: Ruggero Lanotte

Received: 26 May 2024
Revised: 18 July 2024
Accepted: 19 July 2024
Published: 29 July 2024

1. Introduction

The key concept of Software-Defined Networking (SDN) involves the separation of the control and data planes within a data network. This decoupling enables next-generation networks to gain enhanced flexibility, programmability, and ease of management [1].

In contrast to conventional network infrastructure and methodologies involving switches and routers, the SDN concept advocates for a centrally administered software program to oversee the network's control plane. The OpenFlow protocol serves as a means of communication between the SDN controller and network devices for the management of configurations [2]. The OpenFlow protocol enables switch communication in SDN, relays device flow statistics to the controller, and handles unknown flow identification requests. The southbound interface is essential for SDN's effective network management and control [3]. In the context of SDN, the OpenFlow protocol serves as the widely adopted southbound interface or API that enables control decisions, including device detection, topology management, and flow management [4].

In addition to the southbound interface, there are also northbound and east–west interfaces that play important roles in network management and connectivity. The northbound interface enables communication between the controller and SDN applications, while the east–west interface facilitates traffic exchange between clustered SDN controllers at the control layer. These interfaces enhance scalability, reliability, and programmability within an SDN architecture.

Implementing an architecture that incorporates a centralised control plane or SDN controller operation can introduce the risk of a single point of failure within the SDN network, particularly affecting the southbound interface. The southbound interface is critical as it facilitates communication between the SDN controller and the network devices (southbound devices). In a centralised architecture, the SDN controller manages the entire network's control logic from a single location.

To mitigate these risks, network architects often consider distributed or hybrid control plane architectures. These approaches distribute control logic across multiple controllers, enhancing redundancy and fault tolerance.

However, it is crucial to also consider how these controllers synchronise and communicate with each other. Effective inter-controller communication is essential for maintaining consistency in southbound communication and facilitating seamless failover in case one controller fails. This ensures that the failure of one controller does not incapacitate the entire network and that network operations continue smoothly and efficiently.

Having investigated the influence of expanding numbers of OpenFlow switches integrated into various network topologies, we focused on examining the impact of this communication on controller synchronisation and communication. Specifically, we analysed communication between SDN controllers within a cluster, termed as east–west traffic, across tree (star-bus network), linear, and torus topologies.

ONOS and ODL, chosen for their cutting-edge technology and reputation for providing advanced capabilities in SDN controllers, were utilised in our research. Their esteemed status and value-added features made them crucial tools for conducting comprehensive assessments and analyses in our study.

Consequently, past research has explored using a dispersed or clustered SDN controller architecture to manage dispersed multi-domain SDN systems in data centres, corporations, consumer networks, and other places [5]. Table 1 shows the advantages of a multi-controller clustered architecture in terms of modularity, consistency, and industry readiness. Furthermore, the east–west-bound interface, the third interface in a multi-SDN controller or clustered SDN system, enforces rules and protocols.

Table 1. Single-controller and multi-controller architectures.

Criteria	**Single Controller**		**Multi-Controllers**	
	POX	**Ryu**	**ODL**	**ONOS**
First Release in	2011	2012	2013	2014
Architecture	Centralised	Centralised	Distributed	Distributed
East–West	NA	NA	Available	Available
Language	Python	Python	Java	Java
Modularity	Poor	Medium	Very Good	Very Good
Consistency	No	Yes	Yes	Yes
Updates	Poor	Medium	Very Good	Very Good
Industry Proven	No	No	Yes	Yes

The terms 'distributed' and 'clustered' are used interchangeably to provide simpler explanations in this research [6].

1.1. Research Challenges

In this study, we address the following three key research questions or challenges:

1. What effect does the selection of controller clustering techniques have on the control communication volume among controllers across various network topologies?
   To address this research question, we investigate the effect of controller clustering techniques, particularly ODL cluster and ONOS, on the volume of control communications in three network topologies (torus, linear, and tree). The variation in control communication volume between these two types of controllers for a given network topology is also investigated.
2. What are the key factors that influence the decrease in control communication volume in an ODL cluster leaderless compared to an ONOS leader-based controller cluster?
   We address this research question by examining the factors that contribute to lower control communication volume per second in the ODL cluster compared to the ONOS controller cluster. This investigation helps identify the reasons behind the observed differences and determine what contributes to them.
3. What challenges did the ODL method face with an increase in the number of southbound devices compared to the ONOS method?
   This question focuses on the challenges specific to ODL cluster communication when scaling up the number of southbound devices. It aims to understand the reasons behind the exponential increase in the cluster communication volume observed in ODL compared to the more gradual increase in ONOS.

1.2. Research Contribution

The main contributions of this paper can be summarised as follows:

1. We provide a comprehensive analysis of the communication patterns observed in ODL and ONOS clusters. In doing so, we offer insights into the behaviors of different cluster coordination operations in torus, linear, and tree network topologies.
2. We evaluate the system performance, focusing on the scalability of ODL and ONOS controllers. In doing so, we provide guidelines for selecting the appropriate controller based on the size of the southbound network.
3. We conduct a detailed comparison of the coordination patterns among controller clusters, including the time intervals between each cluster. The differences observed in large-scale network environments and the challenges faced by each controller cluster provide a better understanding of the strengths and limitations of ODL and ONOS SDN clusters.

1.3. Structure of the Article

The remainder of this paper is organised as follows: Section 2 presents related work on SDN controller clustering. The distribution and coordination factors of clustered controllers are discussed in Section 3. The impact of southbound expansion on cluster performance is analysed in Section 4. Section 5 focuses on the southbound implementation in leader-based and leaderless controller clusters. Simulation results are presented in Section 6, and practical system implications are discussed in this section. Finally, the paper is concluded in Section 7.

2. Related Work

The SDN controller clustering approach increases network operations' agility, extensibility, and flexibility [6]. Moreover, SDN simplifies configuration and device manageability with vendor-neutral configuration approaches.

The research conducted between 2015 and 2019 has laid significant groundwork for understanding and improving Software-Defined Network (SDN) architectures. Müge Erel et al. [7] utilised a Mininet-based SDN simulation to demonstrate the scalability of OpenDaylight, offering insights into flow admission with a limited number of topologies.

Kim et al. [8] investigated the effectiveness of distributed data stores in OpenDaylight controller clusters, providing valuable findings on load balancing and system performance in large-scale networks. Chaipet and Putthividhya [9] examined scalability issues in single-controller designs, contributing to our understanding of load balancing under various traffic conditions. Abubakar Siddique et al. [10] focused on the scalability and reliability of ONOS clusters, monitoring data transfer rates and volumes, though the study lacked information on detailed topology arrangements. Suh et al. [11] aimed to enhance SDN availability and scalability for service providers, addressing downtime minimisation during reconfigurations. Despite being several years old, these studies have had a lasting impact on the field by highlighting critical aspects of SDN scalability, reliability, and performance, which continue to inform and inspire current research efforts.

Building on these foundational studies, recent research has further advanced our understanding of SDN architectures and their practical applications.

Dharmik et al. [12] identify network bandwidth and connectivity as key constraints affecting performance, focusing on throughput and latency. Their study shows that OpenDaylight's latency is impacted by the number of hosts, while ONOS achieves higher average throughput and lower jitter. However, the evaluation is limited to scenarios using OpenFlow switches within a tree topology.

S. Deepak et al. [13] assert that large-scale SDN networks necessitate the deployment of multiple controllers, which must collaborate effectively to ensure scalability, fault tolerance, and reduced latency. The study identifies propagation delay, dynamic traffic load, and the connection of switches to controllers in a multi-controller environment as critical factors impacting network performance. However, the study would benefit from a focused analysis of the Multi-Domain Partition (MDP) technique, particularly concerning its impact on inter-controller communication.

Shirvar et al. [14] have identified that network throughput, measured as the number of packets per second an SDN controller can handle, significantly impacts the performance of SDN controllers. Throughput can be assessed based on bits per second (bps) or as the number of network packets or flows (e.g., packet-in and packet-out) per second, specifically in the context of SDN. However, the study would be more comprehensive if it included additional scenarios and topologies, such as star and tree topologies.

Niu et al. [15] investigated the scalability and reliability of SDN multi-controller systems. The study revealed that nodes broadcast data through east–west interfaces and collaborate in decision-making processes in both ONOS and ODL. Additionally, the research identified six states in the initial distributed controller communication process. However, the study did not include data pertaining to initial packet communication as an output.

Xu et al. [16] propose a multi-controller deployment strategy for Software-Defined Networking (SDN) using an enhanced clustering algorithm aimed at enhancing network quality within SDN architectures. The study evaluates the clustering algorithm based on factors such as the number of nodes, number of links, controller processing rate, switch request rate, and link propagation rate. However, a more comprehensive analysis could include consideration of the impact of inter-controller communication patterns and packet rates on the clustering algorithm, providing additional insights into its effectiveness.

Our research delves into the effect of expanding southbound communication to encompass multi-SDN controller communication in a data centre environment. The focus is specifically on separate ONOS and ODL cluster environments, examining their impact and implications. Each cluster consists of three SDN controllers. OpenFlow 1.3 [17] was used as the southbound protocol in the experiment. The system impact and analysis of each southbound topology were carried out in Mininet emulation software 2.3.0 [18], utilising up to 255 OpenFlow-enabled virtual switches. Additional hardware resources were used to conduct seven torus topology scenarios to collect different outputs. Captured data communication was plotted onto graphs using the Wireshark protocol analyser [19]. The distribution and coordination factors of clustered controllers are discussed next.

3. Distribution and Coordination Factors of Clustered Controllers

There are two significant qualitative variables in a clustered-controller environment. They are distributed designs and coordination strategies. These two qualitative elements are further split into two kinds of distributed designs (logically centralised and logically distributed) and two types of coordination strategies (leader-based and leaderless) [20].

3.1. Distributed Architectures

The architectural differences or variations in qualitative components of industrial-use SDN controllers are often overlooked in the existing literature when formulating findings on controller east–west operations. Therefore, taking architectural changes into account in distributed controller contexts will lead to more significant arguments and findings for future research.

The initial SDN concept has evolved from a single central controller method to dual or multi-SDN controller designs widely used in enterprise-level networks. Due to scalability, performance, reliability, and other considerations, contemporary technological trends focus on distributed controller architectures [21].

3.1.1. Logically Centralised Architecture

The single SDN controller provides the centralised interface for connecting all SDN-capable southbound devices. SDN's initial premise was logical centralisation. However, scalability issues arise when a single controller manages many flows. Some studies [22,23] have developed literature to find a solution by restricting route requests. In addition to scalability issues, a single controller creates a single point of failure for SDN communication. The controller's failure prevents subsequent southbound flows from being processed.

3.1.2. Logically Distributed Architecture

Multiple controllers work together in a distributed design to overcome the limitations of a single-controller architecture. This method makes it easier for dispersed controller programs to share network burdens.

The ability to share information among dispersed SDN controllers is vital. In recent controller applications, there are two basic coordination strategies available for this purpose [21].

3.2. Coordination Strategies

There are two major coordination strategies, one where a controller (root/master) gathers all network updates (leader) and then distributes them to each controller (non-root/slave), and the other in which a peer-to-peer strategy (flat model) is used to maintain a global topology view of the complete network in each controller [24].

3.2.1. Leader-Based Strategy

In this leader-based controller communication, there will be a leader, root, or master controller who maintains the global network topology perspective. This controller has read/write access to the global topology and can update it.

The non-leader, non-root, or slave controllers will always communicate with the master controller to update the southbound details of the network, as shown in Figure 1.

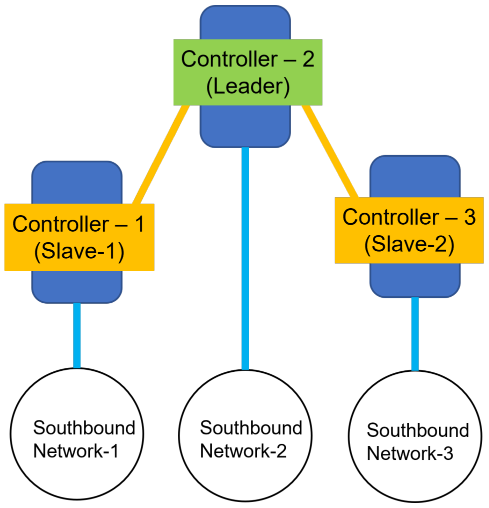

Figure 1. Leader manages the cluster coordination.

3.2.2. Leaderless Strategy

A leaderless or flat approach will connect each controller and share the topology information regularly with each other.

Figure 2 shows that each controller maintains a global network view in local data structures. Subsequently, this information communicates via the east–west interface to another cluster member.

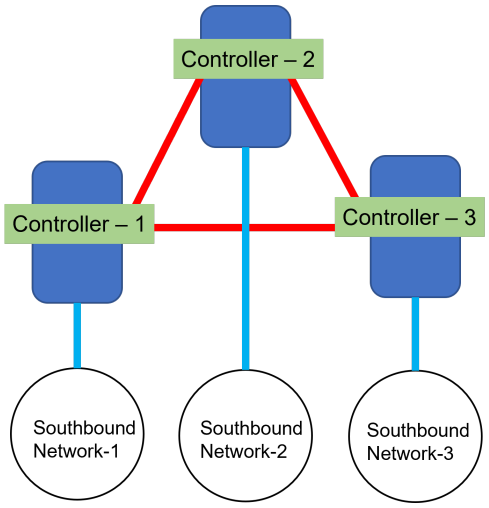

Figure 2. Controller federation manages the cluster coordination.

4. Impact of Southbound Expansion for Cluster Performance Influences

In complex network environments like those found in Software-Defined Networking (SDN), the quest for scalability is met with the need to ensure consistency throughout the system. As networks grow larger and more intricate, it becomes necessary to deploy multiple controllers or clustering mechanisms to efficiently manage increased demands. However, this expansion brings about challenges in maintaining uniformity and coherence across the network.

The main challenge lies in finding the right balance between scalability—ensuring the network can handle growing traffic, devices, and services—and consistency, which requires synchronised operations and coherent policies across all controllers or clustered units. Scalability is crucial for adapting to changing network needs and accommodating expanding workloads, while consistency prevents issues like data discrepancies and operational inefficiencies.

Figure 3 illustrates the various challenges and considerations related to scaling southbound operations in a multi-controller or clustered SDN environment. This visual representation aids in understanding the complex interactions and dependencies involved in managing scalability while maintaining consistency and addressing related concerns such as dependability and load balancing.

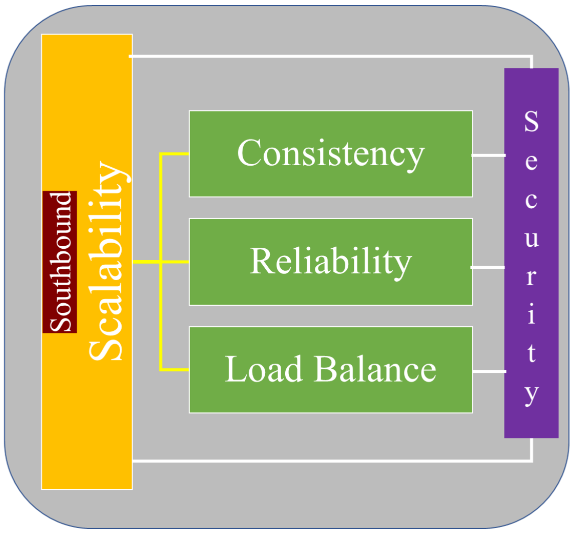

Figure 3. Southbound scalability matters for consistency, reliability, load balance, and security.

The task of managing the enlarged network topology falls to cluster controllers as the network expands southward. Consistency in network topology makes it easier to achieve assured host communication.

The unbalanced distribution of southbound devices across the number of clustered controllers in distributed SDN setups creates load-balancing difficulties.

Furthermore, link failures and security events, such as DDoS attacks in the southbound direction, can cause failures in all mentioned areas, including the reliability of southbound links [25].

The scalability of southbound operations takes several forms, including increasing the number of southbound devices, adding additional functions to the southbound interface, expanding geographic coverage, accommodating increased traffic volume, and incorporating more peer hardware.

When considering southbound scalability in SDN, academic studies often focus on addressing two fundamental research problems: clustered-controller consistency and southbound operational scalability. These challenges are critical for ensuring the stability, efficiency, and effectiveness of SDN deployments, particularly in large-scale or complex network environments.

Clustered-controller consistency refers to ensuring consistency and synchronisation among multiple controllers in a clustered SDN architecture. This is crucial for maintaining network stability and coherence.

Southbound operational scalability focuses on the scalability of the southbound interface, which connects the SDN controllers to the forwarding elements (e.g., switches and routers) in the network. Ensuring that this interface can handle the increasing demands of a growing network while maintaining performance and efficiency is essential for overall network scalability.

Encountering an increase in the number of access-level network hardware or SDN-capable southbound network devices connected to a clustered controller leads to several system performance challenges. These challenges include heightened network utilisation, complexities in load balancing, increased latency, bandwidth limitations, and elevated traffic volumes. Additionally, managing the scalability of SDN networks becomes especially challenging when the southbound network spans multiple geographic locations or network partitions. Furthermore, future scalability and growth of the southbound network will impact various system features and capabilities. Therefore, it is crucial to carefully consider and design operational consistency in the southbound direction [26].

5. Southbound Implementation in Leader-Based/Leaderless Controller Clusters

In OpenFlow-enabled SDN controller clusters, the Open Network Operating System (ONOS) [27] and OpenDaylight (ODL) [28] have achieved the highest 'Technology Readiness Level' (TRL) [29]. Both ODL and ONOS are at the 'Proven system' level.

The ONOS controller is a service provider-focused effort that aims for high availability, high performance, high scalability, and highly resilient application deployment. To handle all functionalities, ONOS runs in Java and the OSGi runtime bundle with Apache Karaf. ONOS assists service provider networks through various features, with the most essential being an easy-to-use GUI for controlling OpenFlow devices. ONOS also offers real-time network setup and switch management, allowing network application developers to install applications that govern the data plane and southbound traffic.

The leader-based ONOS cluster architecture strongly links member controllers to establish an east–west interface. One controller serves as the 'master', while the other two serve as 'slave controllers'. The ONOS master controller in the cluster is selected using periodic polling technology and is responsible for connecting member controllers. Slave controllers maintain all network status information; if the master fails, the slave controllers will select another master.

The Atomix distributed systems architecture is used for the experimental cluster, in which all controllers exchange cluster characteristics and save modularised information in distinct memory regions. All Atomix nodes broadcast their statuses to other known members during the ONOS propagation to establish the cluster [30].

The operational structure of the leaderless ODL cluster architecture consists of three layers: a network device layer, a coordination and control layer, and an application layer.

The network device layer manages southbound activities, such as plugins and protocols. The coordination and control layer is responsible for service abstraction and network services. The application layer handles all northbound APIs and apps. Many programmable network functions, such as topology management and forwarding plane operation, are supported by the ODL cluster. It employs the YANG data modelling language to maintain the network services structure.

The Model Driven-Service Abstraction Layer (MD-SAL) procedure in ODL forms a clustered controller with similar services provided by all member controllers. Furthermore, MD-SAL includes a distributed datastore feature for storing network states. Cluster controllers, in particular, maintain data partitions and duplicate each other due to the distributed datastore.

ODL uses the Raft protocol to replicate information in each partition, providing strong consistency at the cost of inferior read/write performance. Additionally, the ODL cluster enables the establishment of redundant connections, allowing for load sharing among controllers. The ODL and ONOS cluster controllers exchange heartbeat signals regularly to monitor each cluster's stability [31]. Therefore, ONOS and ODL SDN cluster applications are implemented to observe changes in the Mininet environment.

5.1. Evaluation Environment Settings

For the system evaluation, we utilised a QuantaPlex T41S-2U Node server environment comprising three separate physical servers. Each server was designated to host specific virtual machines (VMs) for the installation of ONOS 2.6.0 (with three VMs), OpenDaylight 15.0 (also with three VMs), and Mininet 2.3.0 (with one VM).

These physical servers were equipped with Intel Xeon E5-2603 V3 6-Core 1.60 GHz processors, each containing 4 $\times$ 16 GB DDR4 2400 MHz RAM modules, and 16.4 TB of SATA storage. Additionally, they were outfitted with four 10 Gbps SFP network interface cards.

The virtual machines were created using VMware ESXi 6.5.0 hypervisor. Each instance of ONOS and OpenDaylight VMs was allocated 8 GB of RAM, 100 GB of storage, and 2 CPUs. The Mininet VM occupied an entire physical server's resources.

In this evaluation setup (Figure 4), the VMs ran Ubuntu 20.04 LTS server OS, and were provisioned with ONOS 2.6.0, OpenDaylight 15.0.0, and Mininet 2.3.0 for conducting the assessments.

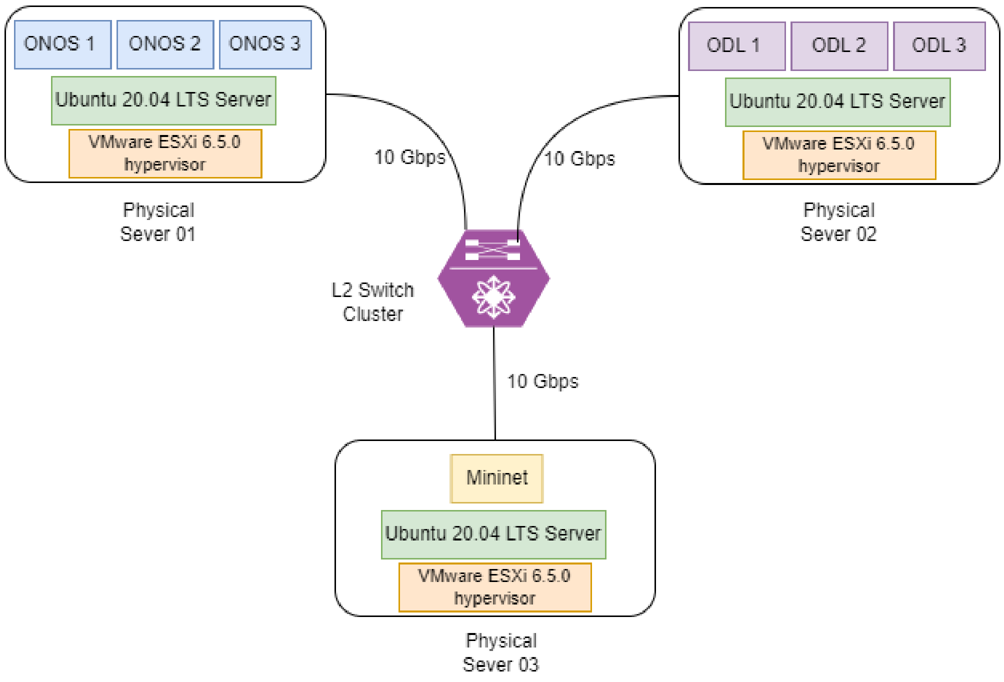

Figure 4. SDN topology in data centre environment.

In the ONOS torus topology configuration, where we employed high-performance cluster hardware with 64 GB RAM, we allocated one physical server for each ONOS instance. Peak level cluster communication used the ONOS torus topology with 64 GB RAM) with a separate physical server for Mininet. This ensured optimal performance and resource utilisation for the ONOS controllers within the evaluation setup.

In addition, each blade server was connected to a clustered switch environment using Cisco Nexus 9300-EX switches, facilitating 10 Gbps fibre channel connections. This setup ensured robust and high-speed communication between the servers, essential for efficient data exchange and coordination within the SDN environment.

5.2. Experimental Topology

We use Mininet to deploy the southbound OpenFlow-Switch topology in each experimental scenario in the simulated environment. Devices in scenarios were incremented with Mininet capabilities [32]. There were eight scenarios tested in the linear and tree types of topology with ONOS and Opendaylight controllers. We only tested seven scenarios for the torus topology due to hardware limitations.

The experimental ODL and ONOS clusters were deployed on Apache Karaf environments, utilising identical hardware and software platforms. The key distinction between the clusters lies in their architectural approaches, with the ODL cluster operating under a leaderless architecture and the ONOS cluster adopting a leader-based architecture.

6. Results and Discussion

We carried out test-bed measurements using the VMware ESXi (6.5) Enterprise platform in a high-end data centre environment. Both leader-based (ONOS) and leaderless (ODL) controller clusters (three controllers per cluster) were tested for three different topologies. For the system performance analysis, we consider packet-level communication and look at various packets to analyse the output data.

6.1. Cluster Initialisation

In the leader-based cluster initialisation process, the ONOS controller cluster exhibited a notable surge in initial communication. As illustrated in Figure 5, the packet transfer rate peaked at over 1200 packets per second during this phase. Notably, during the initial 60 s, the ONOS leader and its members engaged in communication averaging less than 30 packets. The establishment of the first leader–member connection occurred within the time interval of 61–121 s, followed by the formation of the second leader-member connection between 121 and 181 s. This sequential process of member initialisation was consistently observed, with each member initialising one after another. After the initialisation process, the keep-alive communication happened by communicating 200 packets per second communication.

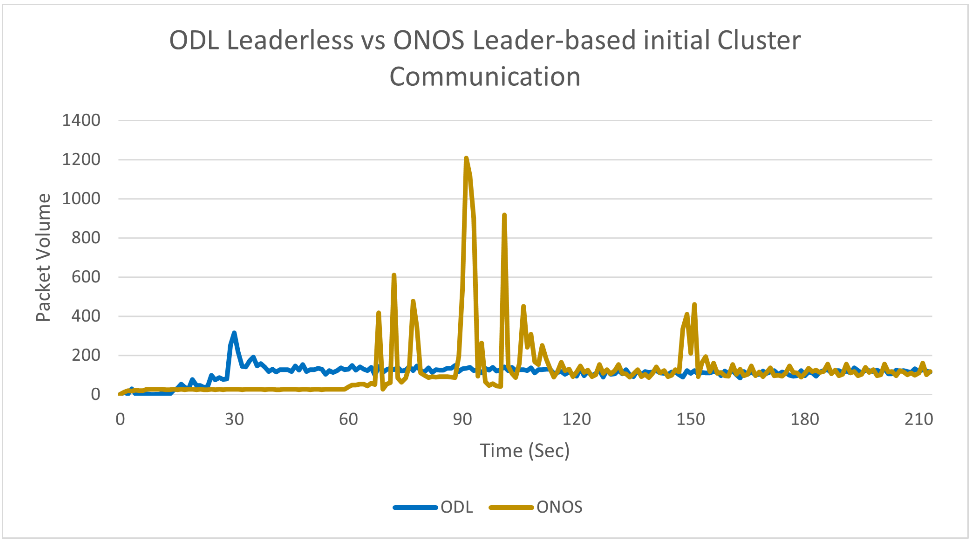

Figure 5. ODL leaderless vs. ONOS leader-based initial cluster communication.

Figure 6 illustrates that during the leaderless open daylight SDN cluster initialisation phase, communication peaks at nearly 170 packets per second, encompassing the initialisation of all three controllers. However, subsequent to the initialisation, ODL members maintain a communication rate of less than 200 packets per second. Moreover, the member initialisation process for all three controllers concludes within the 1–61 s time frame, with a maximum communication rate of 300 packets, significantly lower than that observed in ONOS. Beyond the initial minute, the ODL cluster engages in keep-alive message exchanges, with communication rates remaining below 200 packets. During both ONOS and ODL initialisation, communication between servers utilises a maximum of 5 Mbps data transmission capacity, despite the physical network supporting throughput of up to 10 Gbps. Furthermore, the average packet size for each communication event is less than 200 bytes in both cluster scenarios.

Both the ONOS and ODL clusters were implemented under identical environments and technologies, ensuring a consistent basis for comparison between leaderless and leader-based architectures in SDN controller clusters. The deployment environments for both clusters utilised similar setups, including Apache Karaf frameworks and standardised hardware and software platforms. The distinction between the clusters lay in their architectural approaches: ONOS adopted a leader-based architecture where a designated leader node coordinated communication and initialisation among cluster members, whereas ODL implemented a leaderless architecture where all nodes participated in synchronising the network state with minimal overhead. This setup allowed for a direct evaluation of how architectural differences impact cluster initialisation and communication patterns in leaderless and leader-based SDN environments.

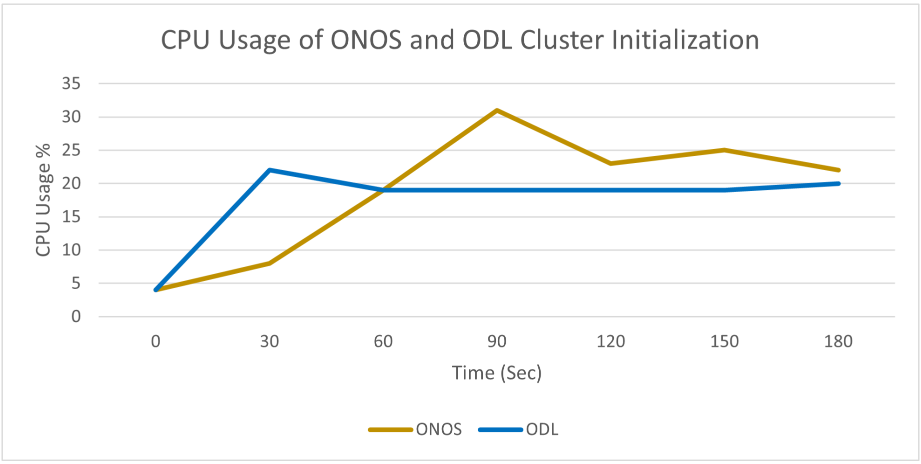

Figure 6. CPU usage of ONOS and ODL cluster initialisation.

Based on these readings, it is evident that there was higher CPU utilisation during periods of high-volume packet communication in both ONOS and ODL environments. Notably, the ODL environment exhibited a maximum CPU utilisation of 22 percent, while the peak CPU utilisation observed in the ONOS environment was higher at 31 percent. This suggests that ONOS may have experienced more intensive processing or resource demands during certain stages of cluster initialisation compared to ODL, potentially due to differences in internal packet processing mechanisms or workload distribution.

6.2. Impact of Southbound Expansion in Tree Topology

Eight scenarios were tested using a tree topology to test the expandability of a single OpenFlow switch to 255. Table 2 shows details of the host count and network links simulated in each experiment.

Table 2. Total number of devices and connections in simulated environment (tree topology scenarios).

Scenario	OpenFlow Switch Count	Host Count	Link Count
01	01	02	02
02	03	04	06
03	07	08	14
04	15	16	30
05	31	32	62
06	63	64	126
07	127	128	254
08	255	256	510

In Mininet, the tree topology is a preset topology option that offers a hierarchical structure resembling a binary tree. This topology distinguishes itself from other topologies by its method of generation, which relies on configuring depth and fan-out values within the Mininet setup. Depth in Mininet refers to the number of hierarchical stages, akin to the levels in a binary tree, while fan-out denotes the number of connections or branches at each stage. Mininet provides users with the flexibility to choose from five different depth and fan-out combinations. In our experiment, we expanded our investigation to include eight distinct depth levels to gather comprehensive results.

ONOS and ODL graphs (Figures 7 and 8) indicate that beyond the fifth scenario, there are significant deviations highlighting one of the limitations of Mininet.

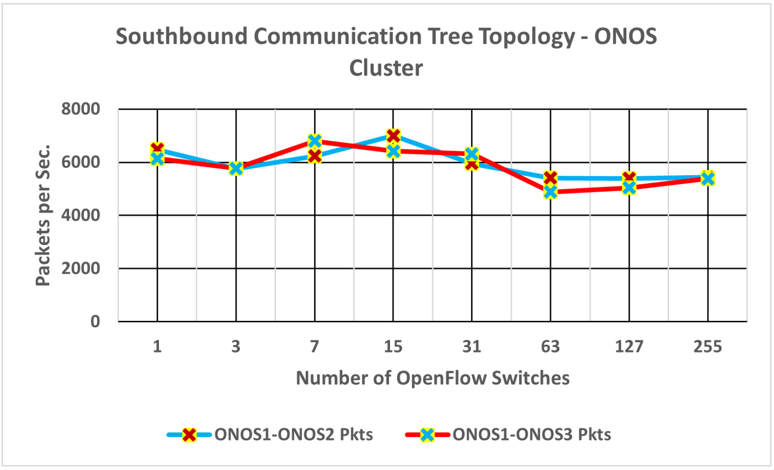

Figure 7. Peak-level cluster communication in ONOS tree topology.

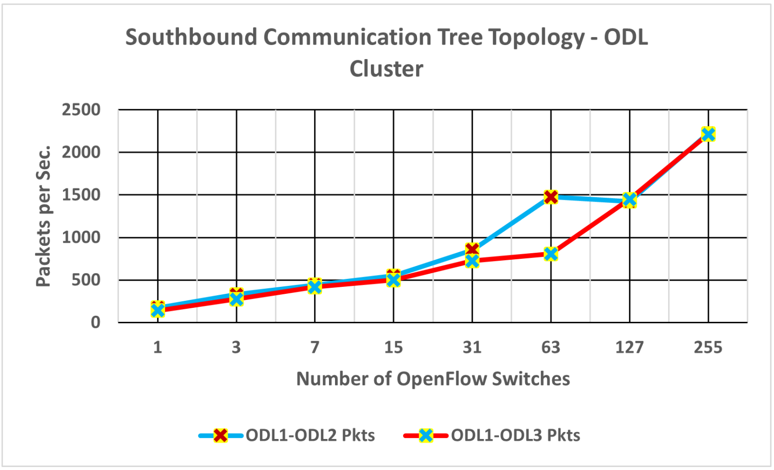

Figure 8. Peak-level cluster communication in ODL tree topology.

However, leader-based ONOS cluster communication has more than five times higher packet communication in southbound node expansion for a tree topology. It implies increased resource utilisation and potential delays in cluster coordination and processing power required to efficiently manage the additional data flow. Moreover, if this heightened packet communication persists as the network expands, it may present scalability challenges for leader-based ONOS clusters, necessitating adjustments or optimisations to support larger deployments.

6.3. Impact of Southbound Expansion on Linear Topology

A single host will connect to a single switch in a linear topology, and Mininet has just one connection between OpenFlow switches. As shown in Table 3, there are eight scenarios that we have tested to identify the cluster synchronisation behaviours in ONOS and ODL clusters.

Figure 9 shows the leader-based ONOS cluster produced more than 2000 packet communications in the first three scenarios where the number of OpenFlow switches was minimal. However, Figure 10 shows the leaderless ODL cluster recorded less than 2000 packet counts even in the last few scenarios. Furthermore, the leaderless scenarios indicate a gradual increment of maximum packet communication, while the leader-based approach has significant stability of the increment after the seventh scenario.

Table 3. Total number of devices and connections in simulated environment (linear topology scenarios).

Scenario	OpenFlow Switch Count	Host Count	Link Count
01	02	02	03
02	04	04	07
03	08	08	15
04	16	16	31
05	32	32	63
06	64	64	127
07	128	128	255
08	256	256	511

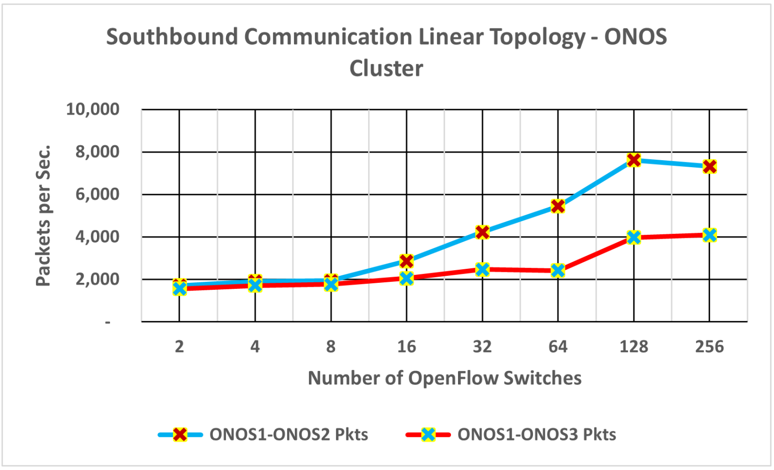

Figure 9. Peak-level cluster communication in ONOS linear topology.

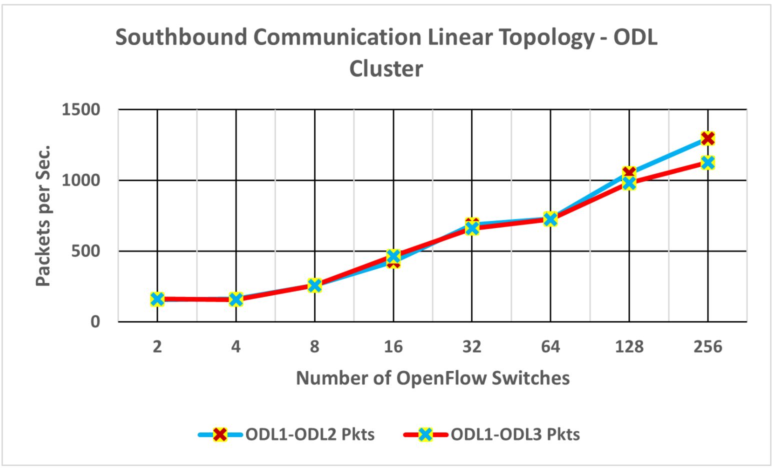

Figure 10. Peak-level cluster communication in ODL linear topology.

The steady escalation of packet communications in the leaderless technique is accessible in all three controllers in the ODL cluster. However, the ONOS cluster creates a high packet rate between the master and first slave controller. The second slave controller receives a 50 per cent lesser packet rate than the master and first slave connection. The hardware resource utilisation in the leaderless cluster approach is lower than in the leader-based approach. It is significant compared to other southbound topology scenarios. The notable discrepancy in packet count increments between ODL and ONOS is significant, particularly when ONOS reaches a count of 8000 packets compared to ODL's 1000 packets. This disparity suggests a substantial difference in the rate of network traffic handling and processing efficiency between the two controller types. The significantly higher packet count reached by ONOS implies a potentially heavier workload or more extensive data processing tasks, whereas the lower count in ODL indicates a comparatively lighter workload or more streamlined operations.

6.4. Impact of Southbound Expansion on Torus Topology

The torus topology is a mesh-type topology that can be created in the Mininet emulator environment. This topology requires more hardware resources to operate appropriately in a cluster environment. In the Mininet environment, we need more than nine OpenFlow switches to initiate the southbound topology connections. Table 4 lists the scenarios with 12 to 768 OpenFlow switches.

Table 4. Total number of devices and connections in simulated environment (torus topology scenarios).

Scenario	OpenFlow Switch Count	Host Count	Link Count
01	12	12	36
02	24	24	72
03	48	48	144
04	96	96	288
05	192	192	576
06	384	384	1152
07	768	768	2304

All VMs have 8GB of RAM and three logical CPU cores in our initial data centre environment. Up to the fifth scenario, where 192 switches and 192 virtual hosts were connected in a torus topology with 576 virtual links, the maximum CPU utilisation reached up to 65 percent. However, in Scenario 6, when the experiment linked 384 OpenFlow switches, the ONOS cluster used more than 95 per cent of its RAM capacity (Figure 11). The seventh scenario (Figure 11) indicates an erroneous output due to the high utilisation of CPU and RAM in Mininet and ONOS VMs. Therefore, the same scenarios were re-initiated in another high-performance environment with 64GB RAM plus six logical cores.

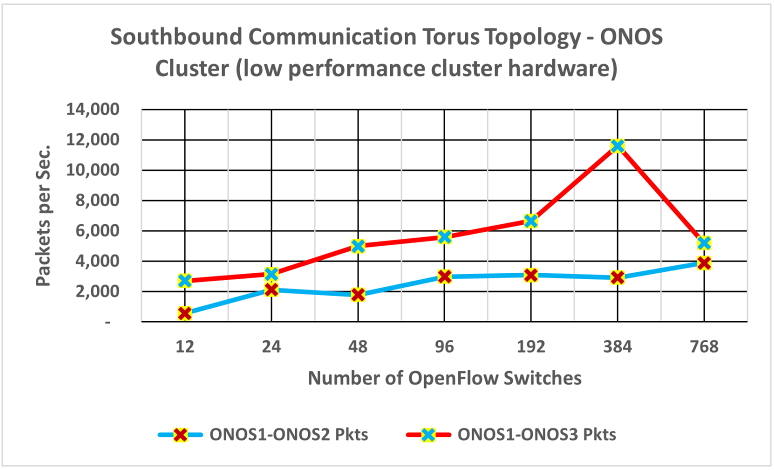

Figure 11. Peak-level cluster communication in ONOS torus topology with 8 GB RAM.

Figure 12 shows the maximum packet communication in the ONOS cluster after increasing hardware resources. The southbound expansions in each experiment scenario (Table 2) produced significant deviation in scenarios 6 and 7 as compared to those shown in Figure 11. Additionally, 4000 packets were communicated between controllers to fulfil synchronisation. Furthermore, the same pattern was identified after scenario 6, similar to scenarios with the linear and tree topologies. However, the slave cluster member-2 achieved a lower packet communication, close to 25 percent.

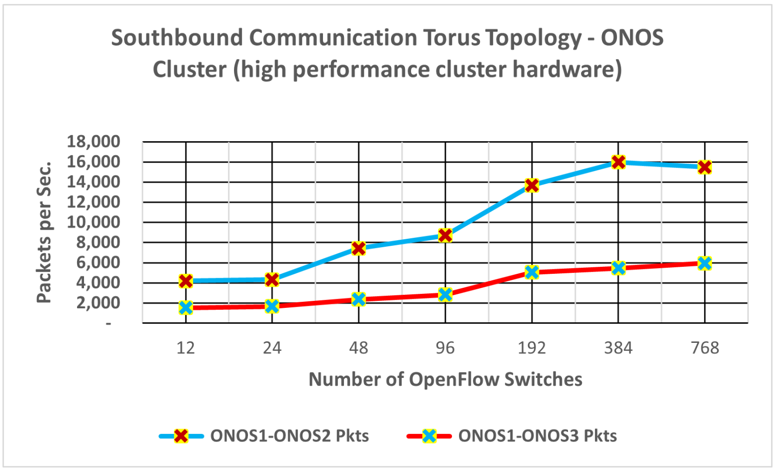

Figure 12. Peak-level cluster communication in ONOS torus topology with 64 GB RAM.

The leaderless ODL cluster exhibited nearly 3500 packet synchronisations in scenarios involving 768 OpenFlow switches using the torus southbound topology. In addition to the observations regarding packet synchronisations, the CPU usage of the leaderless ODL cluster remained below 50 percent even in the scenario with the highest number of OpenFlow switches connected to the environment. Conversely, the ONOS cluster required approximately 16,000 packets for synchronisation. This notable difference underscored efficiency levels in managing network communication and coordination between the two clusters. Despite observing a gradual increment in packet communication between cluster members as the southbound expanded, ODL demonstrated a more streamlined approach compared to ONOS, suggesting potentially superior performance and resource utilisation in certain scenarios (Figure 13).

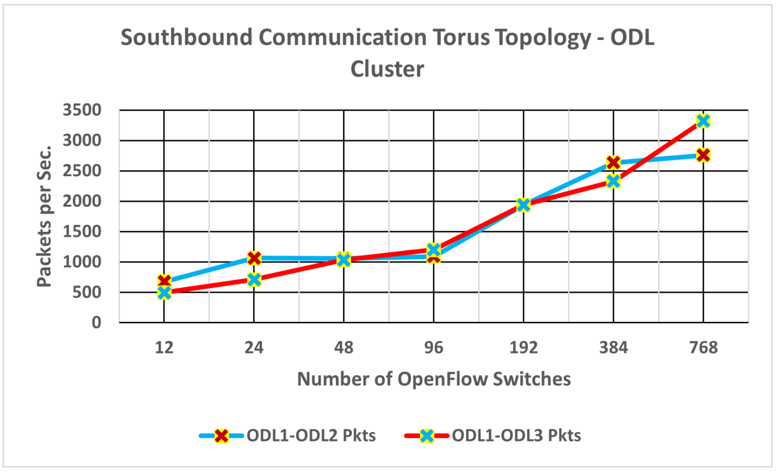

Figure 13. Peak-level cluster communication in ODL torus topology.

7. Conclusions and Future Directions

The impact of increasing the number of southbound devices on system performance was investigated. The leader-based clusters offered higher control communication between the leader and slave members than the leaderless cluster coordination. However, the leader-based architecture is more stable than the leaderless architecture. We observed that the leaderless clusters produced high communication fluctuations even after initialising with zero southbound devices. The results obtained have shown that the different southbound topologies resulted in deviations between the leader-based and leaderless cluster controller environments. We measured the system performance by considering 35 scenarios across three different southbound topologies. We found that both leader-based and leaderless clusters exhibited a consistent increase in cluster communication. The leader-based cluster scenarios produced more cluster communication TCP packets than the leaderless ones. For a small-to-medium SDN environment, the leaderless cluster (ODL) offered superior performance with less topology discovery and flow installation time than the leader-based (ONOS). The ONOS has a granular communication architecture between the leader and follower that allows it to govern all synchronisation from a central location. This produces stable and consistent cluster coordination in large-scale, wide-area networks

For a tree topology, the leaderless cluster communication consistently increased with the number of southbound OpenFlow switches. The leader-based clusters produced 6000 TCP packets even with a single OpenFlow switch topology. However, there was no continuous increment of cluster communication even though we scaled the southbound up to 255 switches. Therefore, leader-based ONOS clusters exhibit excellent stability in wide-area networks.

For a linear topology, there were eight scenarios in each cluster architecture. The ONOS cluster communication showed constant communication rates after reaching 128 OpenFlow switches in the southbound direction. ODL exhibited low TCP cluster control communications in small-scale topologies; however, gradual increment is indicated in large-scale networks. Furthermore, the leader-based architecture produced a significantly lower rate of cluster communication between the second slave member and the master controller in the cluster after the seventh scenario. We found that the higher southbound volume produces consistent communication in the ONOS cluster. In the ONOS leader-based design, it was shown that the second follower has a substantially lower rate of communication than the first follower.

While we focus on utilising readily available resources in Mininet, ONOS, and ODL controller versions, we aim to carry out system evaluation by incorporating diverse and representative network topologies and cluster algorithm communication patterns. This will help us to develop a deeper understanding of SDN technologies and their practical applications in various settings.

ONOS and ODL employ distinct architectural approaches in their implementations of Software-Defined Networking (SDN). ONOS implements a distributed clustering mechanism where the cluster service interface manages roles and nodes across the network. In contrast, ODL adopts a centralised clustering model, treating the cluster as a single logical entity with members represented through the cluster member interface. This architectural difference extends to packet processing, where ONOS utilises distributed packet processing across multiple instances, facilitated by the packet processor interface. ODL, on the other hand, employs a flat model SDN approach for distributed controller operations.

Furthermore, ONOS manages workload distribution through the LeadershipService, which ensures balanced leadership and distributed processing among controllers. In contrast, ODL employs a flat model approach managed by the WorkloadManager. These architectural distinctions significantly impact CPU utilisation and cluster communication patterns within each system. While these differences provide insights into SDN controller performance, further research involving other controller architectures is essential to comprehensively understand their impact in varying network environments.

Author Contributions: Conceptualisation, E.D.S.I.H.; methodology, E.D.S.I.H. and N.I.S.; software, E.D.S.I.H. and N.I.S.; validation, E.D.S.I.H. and N.I.S.; formal analysis, E.D.S.I.H. and N.I.S.; investigation, E.D.S.I.H. and N.I.S.; resources, N.I.S. and J.G.; data curation, E.D.S.I.H. and N.I.S.; writing—original draft preparation, E.D.S.I.H.; writing—review and editing, N.I.S. and J.G.; visualisation, E.D.S.I.H. and N.I.S.; supervision, N.I.S. and J.G.; project administration, N.I.S. and J.G. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The original contributions presented in the study are included in the article, further inquiries can be directed to the corresponding author.

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

References

1. Anerousis, N.; Chemouil, P.; Lazar, A.A.; Mihai, N.; Weinstein, S.B. The Origin and Evolution of Open Programmable Networks and SDN. *IEEE Commun. Surv. Tutor.* **2021**, *23*, 1956–1971. [CrossRef]
2. McKeown, N.; Anderson, T.; Balakrishnan, H.; Parulkar, G.; Peterson, L.; Rexford, J.; Shenker, S.; Turner, J. OpenFlow: Enabling Innovation in Campus Networks. *SIGCOMM Comput. Commun. Rev.* **2008**, *38*, 69–74. [CrossRef]
3. Kreutz, D.; Ramos, F.M.V.; Veríssimo, P.E.; Rothenberg, C.E.; Azodolmolky, S.; Uhlig, S. Software-Defined Networking: A Comprehensive Survey. *Proc. IEEE* **2015**, *103*, 14–76. [CrossRef]
4. Abu Abd-Allah, A.G.; Aya Sedky Adly, A.Z.G. Scalability between Flow Tables & Multiple Controllers in Software Defined Networking. *J. Comput. Sci. IJCSIS* **2019**, *17*, 23–44.
5. Phemius, K.; Bouet, M.; Leguay, J. DISCO: Distributed multi-domain SDN controllers. In Proceedings of the 2014 IEEE Network Operations and Management Symposium (NOMS), Krakow, Poland, 5–9 May 2014; pp. 1–4. [CrossRef]
6. Cox, J.H.; Chung, J.; Donovan, S.; Ivey, J.; Clark, R.J.; Riley, G.; Owen, H.L. Advancing Software-Defined Networks: A Survey. *IEEE Access* **2017**, *5*, 25487–25526. [CrossRef]
7. Erel, M.; Teoman, E.; Özçevik, Y.; Seçinti, G.; Canberk, B. Scalability analysis and flow admission control in mininet-based SDN environment. In Proceedings of the 2015 IEEE Conference on Network Function Virtualization and Software Defined Network (NFV-SDN), San Francisco, CA, USA, 18–21 November 2015; pp. 18–19. [CrossRef]
8. Kim, T.; Choi, S.G.; Myung, J.; Lim, C.G. Load balancing on distributed datastore in opendaylight SDN controller cluster. In Proceedings of the 2017 IEEE Conference on Network Softwarization (NetSoft), Bologna, Italy, 3–7 July 2017; pp. 1–3. [CrossRef]
9. Chaipet, S.; Putthividhya, W. On Studying of Scalability in Single-Controller Software-Defined Networks. In Proceedings of the 2019 11th International Conference on Knowledge and Smart Technology (KST), Phuket, Thailand, 23–26 January 2019; pp. 158–163. [CrossRef]
10. Muqaddas, A.S.; Giaccone, P.; Bianco, A.; Maier, G. Inter-Controller Traffic to Support Consistency in ONOS Clusters. *IEEE Trans. Netw. Serv. Manag.* **2017**, *14*, 1018–1031. [CrossRef]
11. Suh, D.; Jang, S.; Han, S.; Pack, S.; Kim, M.S.; Kim, T.; Lim, C.G. Toward Highly Available and Scalable Software Defined Networks for Service Providers. *IEEE Commun. Mag.* **2017**, *55*, 100–107. [CrossRef]
12. Lunagariya, D.; Goswami, B. A Comparative Performance Analysis of Stellar SDN Controllers using Emulators. In Proceedings of the 2021 International Conference on Advances in Electrical, Computing, Communication and Sustainable Technologies (ICAECT), Bhilai, India, 19–20 February 2021; pp. 1–9. [CrossRef]
13. Sri Deepak Phaneendra, Y.; Prabu, U.; Yasmine, S. A Study on Multi-Controller Placement Problem (MCPP) in Software-Defined Networks. In Proceedings of the 2023 International Conference on Sustainable Computing and Data Communication Systems (ICSCDS), Erode, India, 23–25 March 2023; pp. 1454–1458. [CrossRef]
14. Shirvar, A.; Goswami, B. Performance Comparison of Software-Defined Network Controllers. In Proceedings of the 2021 International Conference on Advances in Electrical, Computing, Communication and Sustainable Technologies (ICAECT), Bhilai, India, 19–20 February 2021; pp. 1–13. [CrossRef]
15. Niu, X.; Guan, J.; Gao, X.; Jiang, S. Scalable and Reliable SDN Multi-Controller System Based on Trusted Multi-Chain. In Proceedings of the 2022 IEEE 33rd Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC), Virtually, 12–15 September 2022; pp. 758–763. [CrossRef]
16. Xu, H.; Li, Q. SDN Multi Controller Deployment Strategy Based on Improved Spectral Clustering Algorithm. In Proceedings of the 2022 7th International Conference on Intelligent Computing and Signal Processing (ICSP), Virtually, 15–17 April 2022; pp. 117–120. [CrossRef]
17. Hu, F.; Hao, Q.; Bao, K. A Survey on Software-Defined Network and OpenFlow: From Concept to Implementation. *IEEE Commun. Surv. Tutor.* **2014**, *16*, 2181–2206. [CrossRef]
18. de Oliveira, R.L.S.; Schweitzer, C.M.; Shinoda, A.A.; Prete, L.R. Using Mininet for emulation and prototyping Software-Defined Networks. In Proceedings of the 2014 IEEE Colombian Conference on Communications and Computing (COLCOM), Bogota, Colombia, 4–6 June 2014; pp. 1–6. [CrossRef]
19. Goyal, P.; Goyal, A. Comparative study of two most popular packet sniffing tools-Tcpdump and Wireshark. In Proceedings of the 2017 9th International Conference on Computational Intelligence and Communication Networks (CICN), Cyprus, Turkey, 16–17 September 2017; pp. 77–81. [CrossRef]
20. Espinel Sarmiento, D.; Lebre, A.; Nussbaum, L.; Chari, A. Decentralized SDN Control Plane for a Distributed Cloud-Edge Infrastructure: A Survey. *IEEE Commun. Surv. Tutor.* **2021**, *23*, 256–281. [CrossRef]
21. Bannour, F.; Souihi, S.; Mellouk, A. Distributed SDN Control: Survey, Taxonomy, and Challenges. *IEEE Commun. Surv. Tutor.* **2018**, *20*, 333–354. [CrossRef]
22. Yan, B.; Xu, Y.; Chao, H.J. BigMaC: Reactive Network-Wide Policy Caching for SDN Policy Enforcement. *IEEE J. Sel. Areas Commun.* **2018**, *36*, 2675–2687. [CrossRef]

23. Görkemli, B.; Tatlıcıoğlu, S.; Tekalp, A.M.; Civanlar, S.; Lokman, E. Dynamic Control Plane for SDN at Scale. *IEEE J. Sel. Areas Commun.* **2018**, *36*, 2688–2701. [CrossRef]
24. Amiri, E.; Alizadeh, E.; Raeisi, K. An Efficient Hierarchical Distributed SDN Controller Model. In Proceedings of the 2019 5th Conference on Knowledge Based Engineering and Innovation (KBEI), Tehran, Iran, 28 February–1 March 2019; pp. 553–557. [CrossRef]
25. Hu, T.; Guo, Z.; Yi, P.; Baker, T.; Lan, J. Multi-controller Based Software-Defined Networking: A Survey. *IEEE Access* **2018**, *6*, 15980–15996. [CrossRef]
26. Nguyen-Ngoc, A.; Lange, S.; Zinner, T.; Seufert, M.; Tran-Gia, P.; Aerts, N.; Hock, D. Performance evaluation of selective flow monitoring in the ONOS controller. In Proceedings of the 2017 13th International Conference on Network and Service Management (CNSM), Tokyo, Japan, 26–30 November 2017; pp. 1–6. [CrossRef]
27. Open Networking Foundation. Open Network Operating System (ONOS®) Is the Leading Open Source SDN Controller for Building Next-Generation SDN/NFV Solutions. October 2021. Available online: https://www.opendaylight.org/ (accessed on 18 July 2024).
28. OpenDaylight. OpenDaylight (ODL) Is a Modular Open Platform for Customizing and Automating Networks of Any Size and Scale. October 2021. Available online: https://www.opendaylight.org/ (accessed on 18 July 2024).
29. Lord, P.; Roy, A.; Keys, C.; Ratnaparkhi, A.; Goebel, D.M.; Hart, W.; Lai, P.; Solish, B.; Snyder, S. Beyond TRL 9: Achieving the Dream of Better, Faster, Cheaper Through Matured TRL 10 Commercial Technologies. In Proceedings of the 2019 IEEE Aerospace Conference, Big Sky, MT, USA, 2–9 March 2019; pp. 1–17. [CrossRef]
30. Septian, K.A.; Istikmal.; Ginting, I. Analysis of ONOS Clustering Performance on Software Defined Network. In Proceedings of the 2021 IEEE International Conference on Internet of Things and Intelligence Systems (IoTaIS), Bandung, Indonesia, 23–24 November 2021; pp. 117–122. [CrossRef]
31. Suh, D.; Jang, S.; Han, S.; Pack, S.; Kim, T.; Kwak, J. On performance of OpenDaylight clustering. In Proceedings of the 2016 IEEE NetSoft Conference and Workshops (NetSoft), Virtually, 23–24 November 2016; pp. 407–410. [CrossRef]
32. Arahunashi, A.K.; Neethu, S.; Ravish Aradhya, H.V. Performance Analysis of Various SDN Controllers in Mininet Emulator. In Proceedings of the 2019 4th International Conference on Recent Trends on Electronics, Information, Communication Technology (RTEICT), Khulna, Bangladesh, 20–22 December 2019; pp. 752–756. [CrossRef]