Republic of Iraq

Ministry of Higher Education and Scientific Research

University of Babylon

Information Technology College

Information Networks Department

AN EFFICIENT APPROACH FOR ENHANCING SECURITY AND CONSISTENCY IN DISTRIBUTED SDN CONTROLLERS BASED ON THE BLOCKCHAIN TECHNOLOGY

A Dissertation

Submitted to the Council of the College of Information Technology for Postgraduate Studies of University of Babylon in Partial Fulfillment of the Requirements for the Degree of Doctor of Philosophy in Information Technology-Information Networks

BY

WED KADHIM OLEIWI HOSHAN

Supervised by

Assist Prof. Dr. Alharith A. Abdullah

2023A.D. **1444 A.H.**

بسم الله الرحمن الرحيم

﴿الَّذِينَ آمَنُوا وَتَطْمَئِنُّ قُلُوبُهُم بِذِكْرِ اللَّهِ أَلَا بِذِكْرِ اللَّهِ تَطْمَئِنُّ الْقُلُوبُ﴾

صدق الله العلي العظيم

سورة الرعد ٢٨

2023A.D. **1444 A.H.**

Supervisor Certification

I certify that the dissertation entitled (**AN EFFICIENT APPROACH FOR ENHANCING SECURITY AND CONSISTENCY IN DISTRIBUTED SDN CONTROLLERS BASED ON THE BLOCKCHAIN TECHNOLOGY**) was prepared under my supervision at the department of Information Networks / College of Information Technology/ University of Babylon as partial fulfillment of the requirements of the degree of Doctor of Philosophy in Information Technology-Information Networks.

Signature:

Supervisor Name: Assist Prof. Dr. Alharith A. Abdullah

Date: / /2023

The Head of the Department Certification

In view of the available recommendations, I forward the thesis entitled "AN EFFICIENT APPROACH FOR ENHANCING SECURITY AND CONSISTENCY IN DISTRIBUTED SDN CONTROLLERS BASED ON THE BLOCKCHAIN TECHNOLOGY" for debate by the examination committee.

Signature:

Prof. Dr. Saad Talib Al-Jebori

Head of Information Networks Department

Date: / /2023

Certification of the Examination Committee

We hereby certify that we have studied the dissertation entitled (**AN EFFICIENT APPROACH FOR ENHANCING SECURITY AND CONSISTENCY IN DISTRIBUTED SDN CONTROLLERS BASED ON THE BLOCKCHAIN TECHNOLOGY**) presented by the student (Wed Kadhim Oleiwi) and examined her in its content and what is related to it, and that, in our opinion, it is adequate with (**Excellent)** standing as a thesis for the degree of Doctor of Philosophy in Information Technology-Information Networks.

Signature:
Name: Dr. Ali Kadhum Idrees
Title: Professor
Date: / / 2023
(Chairman)

Signature:
Name:Dr.Ahmed M.AL-Salih
Title: Assistant Professor
Date: / / 2023
(Member)

Signature:
Name: Dr.Ghaiddaa A.AL.Sultany
Title: Professor
Date: / / 2023
(Member)

Signature:
Name: Dr.Yasir Amer Abbas
Title: Assistant Professor
Date: / / 2023
(Member)

Signature:
Name: Dr.Mehdi Ebady Manaa
Title: Assistant Professor
Date: / / 2023
(Member)

Signature:
Name:Dr.Alharith A.Abdullah
Title: Assistant Professor
Date: / / 2023
(Member and Supervisor)

Approved by the Dean of the College of Information Technology, University of Babylon.

Signature:
Name: :Dr. Hussein Attia Lafta
Title: Professor
Date: / / 2023
(Dean of Collage of Information Technology)

Dedication

To my creator, my safety, and my tranquillity..... God Almighty

To the man who gave up his youth for us..... my dear father

To the candle that melted to light our life..... my beloved mother

To my support in life..... my brothers and sister

To my soul mate..... my beloved husband

To the reason for my smile and continuity..... my soul Noor Al-Hussein and Yassain

Wed AL-Sherefy

Acknowledgement

The greatest verses of reverence to **God**, his prophet **Muhammad**, and his immaculate family are included in this dissertation, for which I am eternally grateful.

To **my adviser**, who provided me with many hours of instruction, supervision, and encouragement, I am eternally grateful.

I would like to express my gratitude to all my wonderful academics at the College of Information Technology for providing us with such an excellent education and inspiring us to achieve our educational objectives.

Lastly, I would want to express my gratitude to all of my classmates in the Ph.D. program, as well as the whole staff of IT department of a Babylon university .

Abstract

Software-Defined Networking (SDN) is a cutting-edge technology for increasing network programmability. Because of its centralized nature, SDN is particularly susceptible to single point of failure (SPoF) and scalability issues. Distributed SDN (DSDN) offers to eliminate the single point of failure; security issue, and scalability that present in the centralize SDN controller. The main concept is to have numerous controllers that can share the burden on the network, and one controller can take over another controller when it breaks. In addition, using blockchain technology can guarantee SDN security and consistency and pave the way for a more scalable and efficient SDN architecture.

In this dissertation, a combination of SDN approach and Blockchain technology are proposed to overcome the weak points the SDN network, Where an Opendaylight-based completely distributed system to deal with issues of uneven consumption of controller's CPU , each controller controlling its own domain and thereby sharing the burden across the network's controllers. while, a blockchain technology proposed as additional consensus mechanism among DSDN controllers where a smart contract, a consensus mechanism and application ledger work on maintains a distributed ledger to update the flow rules and broadcast new rules to all the controllers in the same time , synchronize, ensuring the consistency of the controller's rule set were proposed.

To build the testbed of proposed system a GNS3used as simulator and configurator of the network ,Mininet used to build a custom topologies in the infrastructure , three of OpenDayLight as SDN controller platform, while the wireshark used as packet capture used to capture the packet that send between control plane and data plane, or to capture packet that translate among the SDNcontrollers in control plane.

The result of performance evaluation comparison between central SDN network and DSDN shows that the dropped packet of (SPoF) in central SDN after delete all

the flow table in OVS switches was 100% while the percentage of **Dropped Packet** decrease to 12% ,and 22% as one controller ,two controllers respectively fail in DSDN. The performance evaluation of **Topology Discover** shows that there are a Progressive relationship between the time of Topology Discover and number of nodes in data plane. The **Throughput** of distributed Opendaylight controllers is 45.4 percentage points higher than that of the centralized controller, while the centralized Opendaylight controller has a 3.3% greater **Latency** than distributed SDN controllers; where each of **Throughput** and **Latency** are SDN network **Scalability metrics.** The proposed blockchain-based distributed SDN decreases the **Write Time Overhead** by only about 141ms. The well-known information security paradigm is known as the **Confidentiality-Integrity-Availability (CIA)** triangle which was the basis for our security evaluation of the proposed system; it proves how secure our system is.

Declaration Associated with this Dissertation

Some of the works presented in this Dissertation have been published or accepted as listed below

1. A SURVEY OF THE BLOCKCHAIN CONCEPT AND MITIGATION CHALLENGES IN DIFFERENT NETWORKS/ / Published Q2
2. Design and Implementation of Distributed Controller Clustering for Solving the Issue of Single Failure in SDN Networks / Published Q2
3. Virtual Environment Testbed For DSDN network/Accept (**Springer Conference**)
4. Develop A Blockchain-enabled flow rules as additional consistency mechanism within Distributed Opendaylight Controllers / Under Review
5. Performance Evaluation Comparison Between Central SDN network and DSDN /Under Review

Table of Contents

List of Tables

List of Figures

List of Abbreviations

Abbreviation	Description
Akka	set of open-source libraries uses the actor model to Meets the Needs Distributed Systems
ARP	Address Resolution Protocol
AD-SAL	API Driven-SAL
AMPQ	Advanced message queuing protocol
API	Application programming interface
BGP	Border Gateway Protocol
Cbench	POX controller average throughput and Latency testing tool
CLI	Command line interface
DISCO	Distributed mutli-domain SDN controller
D-ITG	Distributed internet traffic generator tool
DOS/DDOS	Denial of Service / Distributed Denial of Service
DSDN	Distributed SDN
FTP	File transfer protocol
Gossip	peer-to-peer communication mechanism in which nodes periodically exchange state information
Gnutella	peer-to-peer network protocol
GUI	Graphical User Interface
HTTP	Hyper test transfer protocol
HP	American multinational information technology company
ICMP	Internet Control Message Protocol
IoT	Internet of Things
IP	Internet Protocol
Iperf	Hosts Maximum achievable bandwidth testing tool
IPv4	Internet protocol version 4
ISP	Internet service provider
ISO Image	a disc image of all the installation files
JSON	JavaScript Object Notation

LLDP	Link layer discovery protocol
MAC	Media access control address
MD-SAL	Model Driven - SAL
NAT	Network address translation
NBI	Northbound interface
NEC	Information technology company
NOX	Type of Controllers
NP	Network performance
NVP	Network virtualization platform
ODL	OpenDayLight
ONIX	Distributed control platform for large scale production
ONOS	Open network operating system
OVS	Open Virtual Switch
OVSDB	Open vSwitch Database
OSGi	Open Services Gateway initiative
Packet_In	Packet from OVS to SDN controller
Packet_Out	Packet from SDN controller to OVS
P2P	Peer to Peer
QoS	Quality of service
REST	Representation State Transfer
SAL	Service Abstraction Layer
SBI	Southbound interface
SDN	Software defined network
SPoF	Single Point of Failure
SSL	Secure sockets layer
SOX/DSOX	A Generalized / Extensible Smart Network Open flow Controller(X)
TCP	Transport control protocol
TLS	Transport layer security protocol
WAN	Wide Area Network
WheelIFS	wide-area distributed storage system
XML	Extensible Markup Language

YANG model	data modeling language

1.1 Introduction

The mutual advantages of blockchain and SDN integration keep academics interested in the field. When distributed ledger technology (Blockchain) and software-defined networking (SDN) are compatible, All these advantages add to a potential answer to the most serious problems with SDN, including its consistency, stability, security, and scalability[1][2][3]. Additionally, blockchain's decentralized nature lends assistance to the decentralized design of SDN controllers, supplying methods to effectively deal with single-point failure worries by doing away with a central controller [4][5][6]. Therefore, a distributed SDN built on blockchain technology may reap the benefits of both SDN and blockchain technology including security, scalability, privacy, and reliability [7] [8] [9].

To make the design, monitoring, and management of next-generation networks easier, software-defined networking separates a conventional network into a centralized control plane and a remotely programmable data plane[9]. An intelligent, centralized SDN control plane guides the actions of forwarding devices during packet processing and provides a bird's-eye view of the entire network. As they a result of centralized management, programmable networks are possible, as are the implementations of adaptive and autonomous network control. Traditional networks are hard to manage because they are not centralized and have a lot of moving parts. SDNs are trying to solve this problem[8][9].

As SDN acts as a logically centralized controller in next-generation networks, it allows network administrators to dynamically regulate

traffic flows throughout the network to satisfy fluctuating traffic needs. Centralized SDN control enables the creation of dynamic network topologies in data centers and the implementation of policy-based routing in service provider and enterprise networks. To provide a reliable, consistent, and safe control platform, several methods have been used by different controllers[3].

A major vulnerability of the centralized SDN strategy is that the whole network is managed by a single point of control. In order to scale effectively, a centralized entity has to have enough computing capacity and effective data management methods to process flow requests from forwarding switches[3]. Flow setup requests have the potential to overwhelm the centralized controller and slow down response time if they arrive at a high rate and there are frequent changes to the network. The central SDN controller may become a bottleneck as the network expands in terms of the number of switches and end hosts. In order to address these problems, it has been suggested that many controllers be used to distribute the control plane, with one controller being replaced by another in the event of a failure and the controllers working together to distribute network load [3]. Using horizontal and hierarchical topological models to give tasks to controllers on the control plane[3].

The issues of centralized SDN may be mitigated when several controllers are used in SDN architecture to manage scalability, reliability, performance, and single-point failure. The distributed SDN control plane is more responsive, scalable, and reliable than conventional SDN control planes, allowing it to quickly respond to a variety of networking events such as connection failures, requests for new flow configurations, intrusions, etc. Distributed control plane architecture can handle the millions of networking events that happen in

dynamic places like multi-tenant data centers because its solutions can be scaled up and changed[3][4].

Numerous publications have highlighted the potential of blockchain technology [7][8] . It is a brilliant combination of several technologies including P2P networks , encryption protocols , distributed databases, and smart contracts[9][10][11][12]. Blockchains are distributed ledgers that store data similar to databases but they are not managed by a single entity but rather by a network of computers that may be placed anywhere in the world and operated by anybody with access to the Internet [13][14][15][16] . Each data exchange transaction is verified in a P2P network using a time stamp, creating a secure chain of previously traded information. The ledger may be updated by adding a new "block" for each confirmed transaction[19][20].

A smart contract is an electronic protocol or computer program that executes the conditions of a contract or agreement and it is meant to function in the blockchain system, which is a regulatory or legally important move[5][21]. Smart contracts may initially be implemented on Ethereum[22][23].

The main goal of this dissertation is offering a solution for handling security problems in SDN controllers using a decentralized approach. The mechanism's intended purpose is to decentralize the control plane in order to preserve a network-wide perspective, protect against single-point failure, address security and scalability concerns, and improve the controller's own dependability. Instead of a single (central), logical controller regulating all of the network nodes, the suggested system uses three controllers, one for each node. In this setup,

the fewest possible number of devices will all be linked to a single controller at once.

Moreover, a unique approach was provided in the dissertation , which relied on a consensus mechanism that was supported by a private blockchain network that included a smart contract. One of the most crucial aspects of guaranteeing SDN's effective functioning is maintaining consistency in flow rules. Therefore, it is crucial to the secure functioning of DSDN to ensure flow rule consistency, prevent the proliferation of malicious strategies, and guarantees the correct delivery and execution of different flow rules, since forwarding devices blindly believe in them.

1.2 Study Challenges

1.2.1 Single Point of Failure

The loss of the controller renders the network inoperable, unable to react to the failure of other components or even typical operating adjustments[22][23]. As a result, failure of the centralized controller might possibly endanger the entire network. The SDN controller is prone to hardware and software failures, as well as malicious assaults. It should be noted that, in addition to negative stimuli such as failures or assaults, a sudden rush in flow entry changes caused by an increase in network traffic might potentially produce a bottleneck at the controller[24][25].

1.2.2 Scalability

One of the key considerations for developing more scalable networks in SDN is controlling performance and investigating the performance of controllers in relation to various network workloads,

implementations, designs, and so on[23][24].Although studies analyze the scalability performance of controllers they suggest in terms of numerous performance metrics, such as path installation time, link usage, and so on, depending on their target problem, the most prominent and regarded metrics are control plane throughput, which refers to the number of flow requests processed per second, and latency, flow setup, which refers to the delay to reply to flow requests[22][23].

1.2.3 SDN controllers response time

The distributed SDN controller design is not just concerned with managing networks with several SDN controllers operating simultaneously. It also covers how these controllers interact with one another and share the information needed to run the network, where they share the same viewpoint and function as a single unit, one of the most crucial elements in guaranteeing the effective operation of SDN is the consistency of flow rules[23][24]. Consistency algorithms employed in the SDN control work to increase the response time, as is the case in the Raft algorithm, where the leader controller is responsible for the update, while the rest do not have the ability, even if the request is sent to it[24].

1.2.4 Controller –to- Controller Communication overhead

Domain-specific controllers solve data plane failures or traffic flow bottlenecks in their own domains on the distributed SDN control plane. Such modifications should be promptly notified to all other controller instances within a cluster in order to maintain a consistent global view[24][25]. Achieving such a time-sensitive degree of consistency while retaining high performance, however, is a challenging issue;which cause in resource consumer[24][25] .

1.3 Literature review

This section with two parts the first , review of the related research studies that focus on the SDN controllers and approaches that boosted the distributed Software Defined Networks , Second, review of the related research studies that attention on the blockchain_based distributed Software Defined Networks:

1.3.1 Distributed SDN Controllers / Approaches

The authors A. Tootnchian and Y. Ganjali proposed a Hyperflow [26**]** the control plane which depends on the NOX OpenFlow controllers. It is SDN NOX-based, whose design ensures that the application of the network works consistently. The controller topology design has a physical distribution but a logical centralization. All controllers operate the logical domains. They are used as a general form of communication between controllers, which result in performance penalties.

The authors M. Luo and Y Tian proposed a SOX/DSOX [27] *Generalized and Extensible Smart Network Openflow Controller*. The control plane in some studies can be set up as a combination of other approaches (controller) for acquiring the benefits of each and evading its limits, as hybrid controllers could experience computational complexity problems. In order to operate a (sub)network of "significant-size," it is desirable to have a cluster of controllers that are centralized in their operation and function in equal mode with automated failover and load balancing. The software-services defined networking technology allows for the controller clusters to be geographically dispersed managing several(sub)networks while maintaining the requisite level of synchronization and consistency.

While in the work of S.H. Yeganeh and Y. Ganjali [28], they proposed Kandoo. This can be described as a hierarchical control, implement leader-based architecture whereby the root SDN controller takes the role of the leader over the sub-ordinate SDN controller. The platform is designed in form of a tree structure with roots and it leaves constructed. The root is responsible for sorting a network global view. Regarding the scalability problem of the controller, the root tier is one of the obvious issues with Kandoo.

As for Berde and others in [29], they proposed the Open Network Operating System (ONOS) which uses different distributed system techniques to maintain a global view of the network, thereby synchronizing the network topology view. Such a form of control provides a strongly consistent protocol. The ONOS controller supports a distributed SDN network that is logically centralized but physically distributed, similar to the master-slave model. It provides the ability for each member in cluster to play a master role in the same time.

Medved and other researchers in [30] proposed OpenDayLight, which is another study that supports a logically centralized but physically distributed system similar to the master-slave model. The distributed architecture found in the OpenDayLight controller relies on the Akka components [31]. Akka Remoting is used for p2p communication among project components. Akka Clustering provides fault-tolerance, and Akka Persistence enables stateful actors. As compared to ONOS, one of the cluster members at a time takes the role of leader, whereas the rest are followers.

Both of the last related works present a flat architecture for distributed SDN controllers that support a logically centralized yet

physically distributed architecture deployed as the master-slave model. Both use the RAFT consensus algorithm and the cluster deals with persistence according to the Raft consensus-based model. Transactions are only dedicated if the majority of the members in the cluster approve.

The researchers Tran and Ha Manh in [32] proposed a hybrid controller approach of networks using the Gnutella and the peer-to-peer network protocols for distributing network data among controllers. This approach provides logically centered and physically distributed SDNs with multiple clusters. The Gnutella protocol is used to form a group of dedicated controllers that belong to various clusters and provide data synchronization among clusters.

Amiri, Esmaeil and others in [33] worked on a hierarchically distributed SDN model to deal with the problem of heavy load on the Root controller through the elimination of intra-domain information. The schema is the root controller's management of the inter-domain traffic, whereas the leaf controllers operate the intra-domain traffic. Their results show the elimination of unnecessary packets like LLDP and ARP.

Other work proposed by Liu, Waixi [34] presents a model for controllers of physical distribution and logical centralization to Adaptively Adjust and Map controllers (AAMcon). They deployed the ideal distance between switch and controller to construct the domain of each controller. The latter replies to the switch request using the shortest distance possible for reducing any delays with the switch. AAMcon has the ability to achieve the load balance among controllers.

Table 1.1: The Comparison of past Distributed SDN Controllers/Approaches to the proposed Distributed SDN System

Approaches	Programing Language	Controller Platform	Network Distribution	Coordination Strategy	East/West Protocol	Data Storage	Disadvantages
Hyperflow [26]	C++	NOX	Flat	Broker based(P2P)	Publishing/subscription	WheelFS file system	Resource wasting, increase controller load[30]
SOX/DSOX[27]	C++	NOX	Hybrid	Cluster	service bus or extended BGP protocol	-	Resource wasting, increase controller load[30]
Kandoo[28]	C++	Kandoo	Hierarchical	Leader-based	RTPS-DDS	In memory	increase controller number[30]
ONOS[31]	Java	ONOS	Flat	Leader-based	Raft	Distributed data Structure	Resource wasting, increase controller load[30]
Opendaylight[32]	Java	Opendaylight	Flat	Leader-based	Akka(Gossip), Raft	MD SAL Database	Resource wasting, increase leader controller load[30]
Tran and Ha Manh in [33]	Java	ONOS	Hybrid Controller Network	Leader-based	Gnutella protocol	Distributed data Structure	Resource wasting, increase leader controller load[30]

Waixi [34]	C++	Floodlight	Hierarchical	Broker based(P2P)	Generic Routing Encapsulation (GRE)	BigDB (NoSQL, Cassandra-based database)	increase controller number[30]
The Propose DSDN System	Python	Opendaylight	Flat	Leader based (Topology shard),p2p(Inventory shards)	Raft,Akka(Gossip)	MD SAL Database	-

There are some works that work on evaluate the performance of central SDN's network as in [35], It analyzes and compares the throughput and latency performance of numerous well-known open source controllers, including ONOS, Ryu, Floodlight, and OpenDayLight, using the OpenFlow benchmarking tool Cbench. in[36]proposed Best practices for quantitative controller evaluation are outlined, went into the possibilities of benchmarking tools for SDN controllers, the most important result of this study was The literature is full with suggested controllers that never see the light of day because their authors fail to disclose enough details for a third party to implement them, This means that any evaluation of them must be based exclusively on theoretical concerns. In [37], OpendayLight and open networking operating system (ONOS), two of the most potent and well-known SDN controllers, are experimentally compared , Mininet is used as an experimenting platform, Wireshark is used to record and examine live packet traffic. In [38] they take a close look at three different OpenFlow-based SDN controllers ,The latency, throughput, and scalability of the controllers FloodLight, OpenDayLight (ODL), and Ryu are measured. To do this, Mininet is utilized as a platform to run the Cbench tool in a

virtual setting. A comparison study between two(Onos , ODL) SDN's controller performance proposed ,the well-known ping and Iperf tools employs in his research[39] to determine the network's latency and bandwidth.

In [41]and [42] the researcher work on topology discovery performance evaluation , They conduct hands-on experiments to compare the topology discovery and topology updating capabilities of the ONOS and OpenDaylight controllers.

All the work above works with central SDN's network except [40], which works with SDN's network with two SDN 's controllers, but this does not represent a distributed SDN network where the minimum number of SDN 's controllers in a DSDN network is three, as it will explain in chapter two. In [43]the researcher work on one experiment, a centralized controller was utilized to compare the proactive mode to the reactive method, while in another experiment, a separate controller was employed to compare the two modes and their respective circumstances.

In this dissertation the Onos controller used at the early of the work as DSDN platform that support the DSDN ,but it prove instability through the work , So the another SDN controller was used which is a OpenDayLight controller ,that show it performance and stability in work.to evaluate it performance the wireshark was used to capture the real packet with proposed system

1.3.2 Blockchain_Based DSDN

Z. Abou El Houda, A. S. Hafid[44] the authors in this work, proposed ablockchain_based DDoS mitigation method in the context of software defined networks across multiple domains, intra domain and

inter domain, allows a mitigation along the path of an ongoing attack and a mitigation near the origin of the attack, which allow abridged the huge cost of forwarding packets, across multiple domains, which consist mostly of useless amplified attack traffic the experiment result shows high accuracy in detecting illegitimate flows, making it a promising approach to mitigate DDoS attacks .

H. Yang, Y. Liang [45], the approach employs blockchain with distributed SDN controllers. All the controllers are connected via blockchain in a distributed manner within different control domains, master-slave role. At the software level, each controller in the control plane is loaded with the identical distributed ledger maintained by consensus plane, and smart contracts utilize the consistent data in the distributed ledger to provide the customized network function. The consensus plane performs multi-controller consensus for the pending-process services and inserts the results into a block data structure on a distributed ledger, while the contract plane contains smart contracts to perform automatic network functions. The blockchain-based solve a number of security issues, including fault tolerance enabled by blockchain consensus, data consistency based on distributed ledger.

Jiasi, Weng & Jian [46], propose design a monolithic security mechanism for SDN based on Blockchain. The mechanism decentralizes the control plane to overcome the single-point failure, while maintaining a network-wide view depends on the blockchain layer that recorded. The mechanism also guarantees the authenticity, traceability, and accountability of application flows, and hence secures the programmable configuration, using the authentication algorithm to access control among controller-App. Moreover, a secure controller-switch channel to

further protected resources and communication in SDN by used security protocol.

M. Singh, G. S. Aujla[47] in this work, the authors prevent a projecting malicious or uncontrolled traffic flows, provided a secure software-defined network through a deep-learning-based blockchain framework, where various features extracted for attacks detection through Botlzaman machine. Additionally they work on validate any packet request by voting-based consensus, while the zero proof of knowledge used for validate and switch register.

Shashidhara R, Ahuja N[48] to ensure the network reliability and safety, the authors proposed a Modified-Delegated Proof of Stack as the consensus protocol that implemented using smart contract. The experiment results show the secure of the proposed work and energy efficient in addition to the minimize the signaling overhead, bandwidth requirements provides a higher throughput.

Xiong, Ao, et al.[49], This study proposes a hierarchical distributed control approach and network attack detection algorithm based on blockchain consensus, as well as the Jaccard similarity coefficient for detecting network vulnerabilities. this work compares the security performance of the ClusterBlock model with the existing model based on the OpenFlow protocol using simulated tests. The ClusterBlock model provides more steady bandwidth and greater security performance, according to the evaluation findings.

Weichen Lian, et al. [50]uses blockchain to secure flow rules in SDN and to detect compromised nodes in the network when the proportion of malicious nodes is less than one-third. The scheme places the flow strategies into blockchain in form of transactions. Once an unmatched flow rule is detected, the system will issue the problem by

initiating a vote and possible attacks will be deduced based on the results.

Table 1.2: The Comparison of past Blockchain_Based Distributed Software Defined Networks (SDNs)approaches to the proposed System

No	Approaches	Objective of The Paper	Key Findings
1	Z. Abou El Houda, A. S. Hafid[44]	DDoS attack mitigation in the context of software defined networks across multiple domains, intra domain and inter domain	ablockchain_based DDoS mitigation method, show high accuracy in detecting illegitimate flows, making it a promising approach
2	H. Yang, Y. Liang [45],	multi-controller consensus for the pending-process services	blockchain-based solved a number of security issues, including fault tolerance enabled by blockchain consensus, data consistency based on distributed ledger
3	Jiasi, Weng & Jian [46],	Proposed a mechanism to decentralizes the control plane to overcome the single-point failure, in addition to maintaining a network-wide view depends on the blockchain layer that recorded	guarantees the authenticity, traceability, and accountability of application flows
4	M. Singh, G. S. Aujla[47]	prevent a projecting malicious or uncontrolled traffic flows	framework for a secure software-defined industrial network through a deep-learning-based blockchain
5	Shashidhara R, Ahuja N[48]	SDNnetwork reliability and safety	framework for energy efficient in addition to the minimize the signaling overhead
6	Xiong, Ao, et al[49],	SDN network attack detection	The proposed model provides more steady bandwidth and greater security performance

7	Weichen Lian, et al [50]	secure flow rules in SDN and to detect compromised nodes in the network	Once an unmatched flow rule is detected, the system will issue the problem in SDN network .
8	The Propose DSDN Testbed	Consensus mechanism depends on blockchain technology, a consensus mechanism ; among distributed SDN controller ,that was supported by a private blockchain network .	Consistency of flow rule; inventory shards, among distributed SDN controller One of the most crucial aspects of guaranteeing SDN's effective functioning .

1.4 Problem Statement

- The security issue in the central SDN networks that represented in the single point of failure ; when the central SDN controller crash that led to lose the mind of the network .
- Each SDN control is restricted to a certain number of devises ,data plane that it can manage efficiently.
- Overhead of Distributed controllers communicate with each other and exchange the data that necessary to manage the network.
- maintains a Data consistency between DSDN controllers to retains a global view of the network and synchronization.

1.5 Dissertation Aim & Objectives

- To design architecture of a distributed SDN network that which is fully distributed to overcome master-slave and leader-follower umbrella .
- To allow the system manger the ability to manage the DSDN controller remotely by blockchain technology

- To improve DSDN performance by reduce time overhead of DSDN controller communication through proposing new consensus mechanism .
- Design architecture of a distributed SDN network to overcome the Security issue of the central SDN networks ; single point of failure .
- Provide an ability to scalability and manage devise more than that central SDN controller can .
- Configure a coordinate strategy to maintain a network-wide view among the controllers in the DSDN system to have to Visualize the entire network, and help in case one of the controller fails .
- Propose a new consensus mechanism depending on blockchain technology to maintain the consistency of flow rules among DSDN controllers.

1.6 Dissertation Contributions

The main contributions of the present dissertation can be stated in the following points:

- Propose a testbed ; virtual environment , for a fully distributed Software Define Network system overcomes master-slave and leader-follower umbrella.
- Prove the ability and the efficiency of Distributed SDN in scalability ,a networks of large size.
- Introduce a blockchain as addition consensus mechanism with Opendaylight controller ; blockchain-enabled flow rules

installation that enforced new rules and maintains consistency within the distributed SDN controllers . Developing a smart contract ;which only can mine the new block of flow-rule, using python language .

- The performance distributed SDN networks was evaluated using several criteria (Single Point of Fail , Topology Discover ,Throughput ,Latency ,Recourse consumer) , and proved its efficient and secure .

1.7 Dissertation Layout

The remains of the Dissertation chapters are described in the following:

Chapter Two contains a background and summary of the SDN and DSDN networks and Blockchain technology .

Chapter Three explains the construction of central SDN network , Distributed SDN network ,private blockchain and suggested the unique consistency mechanism of Distributed SDN, and performance Comparison between central and distributed SDN network.

Chapter Four explains the results and investigations that have been acquired based on the proposed technique.

Chapter Five shows the work's conclusions, Also, it offers different future work suggestions .

Chapter Two

The Theoretical Background

2.1 Overview

Integrating blockchain technology with the SDN network and taking use of its capabilities may provide a solution to SDN problems. Given that the SDN's centralized controller design makes it more vulnerable to assaults, one possible solution is to have the affiliated parties construct a consortium chain network to ensure the network's safety. By using blockchain technology, it can guarantee SDN security and consistency and pave the way for a more scalable and efficient SDN architecture.

Blockchain-based solutions show potential, but it is vital to utilize them efficiently without adding unnecessary computational overhead. Furthermore, existing solutions depend primarily on manual operations and ignore SDN behavior-monitoring applications.

This chapter, introduce the specific details of the software-defined network background, distributed SDN background , and blockchain network technology background that will be developed in this dissertation and will be discussed in depth below.

2.2 Software Define Network (SDN)

Software Defined Network Fast advancements in networking and other telecommunications technologies have led to the development of software defined networking. The usage of this tool will significantly accelerate the coordination of new user services and bring about other useful effects. An SDN's "brain," or controller, manages the network's hardware and switches and communicates with applications[52].

The origins of SDN concepts may be traced back to the public switched telephone network, where the separation of the control and data planes was implemented to simplify provisioning and maintenance [52]. In the computer science department at Stanford University, the Ethane project pioneered the use of open source software in split control/data plane architectures. Ethane's easy-to-implement switch architecture inspired the creation of the OpenFlow protocol[52].

Work on OpenFlow continued at Stanford, with test beds being built to evaluate the protocol's potential use both inside a single campus network and over the WAN as a backbone for interconnecting many campuses. OpenFlow switches from NEC and HP, as well as white boxes from Quanta Computer, were used in a small number of academic research and production networks beginning in around 2009[52]. SDN and OpenFlow were supported by the Open Networking Foundation (ONF), which was founded in 2011[52].

2.2.1 Architecture of Software Define Network(SDN)

It seems that every network has both a control plane and a data plane. On most people's minds, a router's or switch's decision-making happens in the control plane. Because of its software foundation, this relies on the central processing unit (CPU) rather than dedicated

hardware. The data plane refers to the super-fast channel that runs via a router or switch[50]. In order to communicate with the device, data packets utilize the data plane. The overall architecture is very decentralized in traditional IP networks due to the close integration of the control and data planes into a single piece of networking hardware. Both sides of the switch or router may communicate with each other. When it comes to conventional IP networks, the control and data planes are intertwined and built into the same networking hardware, but the network architecture as a whole is highly distributed[50]. The control plane and data plane of a typical design are intertwined. Once a route between nodes has been programmed and configured, the control plane will communicate this information to the data plane. As an example of a device that may do this function, consider an Ethernet switch. The data link layer of the Open System Interconnection (OSI) model is where an Ethernet switch operates, and it is fed by both of the model's planes. As shown in Figure (2.1), the forward logic of a switch forms part of the control logic that coordinates and manages all of these elements [50][52].

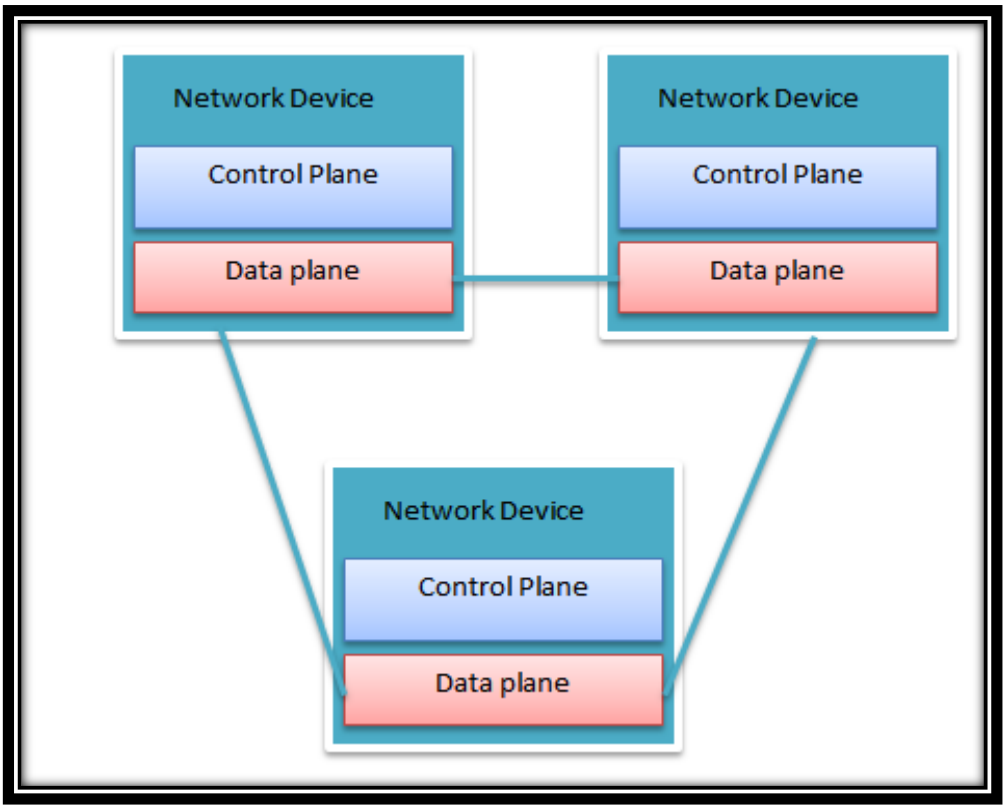


Figure 2.1: The Traditional Network Technology Architected

With Software Defined Networks (SDN), the control and data planes are decoupled and the network is managed from a central location. Figure (2.2)depicts the three layers that make up the SDN architecture [53]. Applications are able to interact with one another through an interfaces between the programs. More detail on these layers will be follow:

1) **Application layer:** By programming in the service-aware behavior of network resources, SDN applications may represent network services or business applications at the application layer. Application-control interfaces, also known as northbound interfaces, are used by these programs to talk to the SDN control layer so that network resource behavior and characteristics may be dynamically modified by the SDN control layer. Figure (2.2) depicts the SDN architecture, which describes how the northbound interface ad well as associated information and data

models are used to take advantage of the SDN control layer's abstracted view of network resources[54].

2) **SDN control layer:** This layer provides the application layer with the ability to control the behavior of network resources, data transfer, and processing in a dynamic and deterministic manner. Through the northbound interface, SDN applications communicate with the SDN control layer to specify how network resources should be managed. The resource-control interfaces (also known as southbound interfaces) are responsible for relaying control signals from the SDN control layer to the underlying network resources. SDN applications' settings and/or functionalities are encapsulated in information and data models. How much abstraction is used depends on the kind of services and applications that will be delivered[54].
3) **Data layer:** Data packets are transported and processed by network nodes at the resource layer in accordance with directives from the SDN control layer sent over a southbound interface[54].

So The SDN architecture, as defined by the ONF[54], is:

- Programmable: Network control is directly programmable because it is decoupled from forwarding functions.
- Agile: By abstracting control from forwarding, administrators may dynamically modify network-wide traffic flow to satisfy shifting demands.
- Programmatically configured: Because SDN programs do not rely on proprietary software, network administrators may easily create dynamic, automated SDN programs that swiftly configure, manage, protect, and optimize network resources.

- Northbound API: is the API that permits communication between the control layer and the business application layer. There is not yet a northbound API that adheres to standards.
- Southbound API: is the API that permits communication between the infrastructure layer and the control layer. OpenFlow, XMPP, and the network configuration protocol are three protocols that can facilitate these conversations.
- Business Applications: These are programs that end users may utilize right away. Supply chain management, customer relationship management, and video conferencing are examples of possibilities.

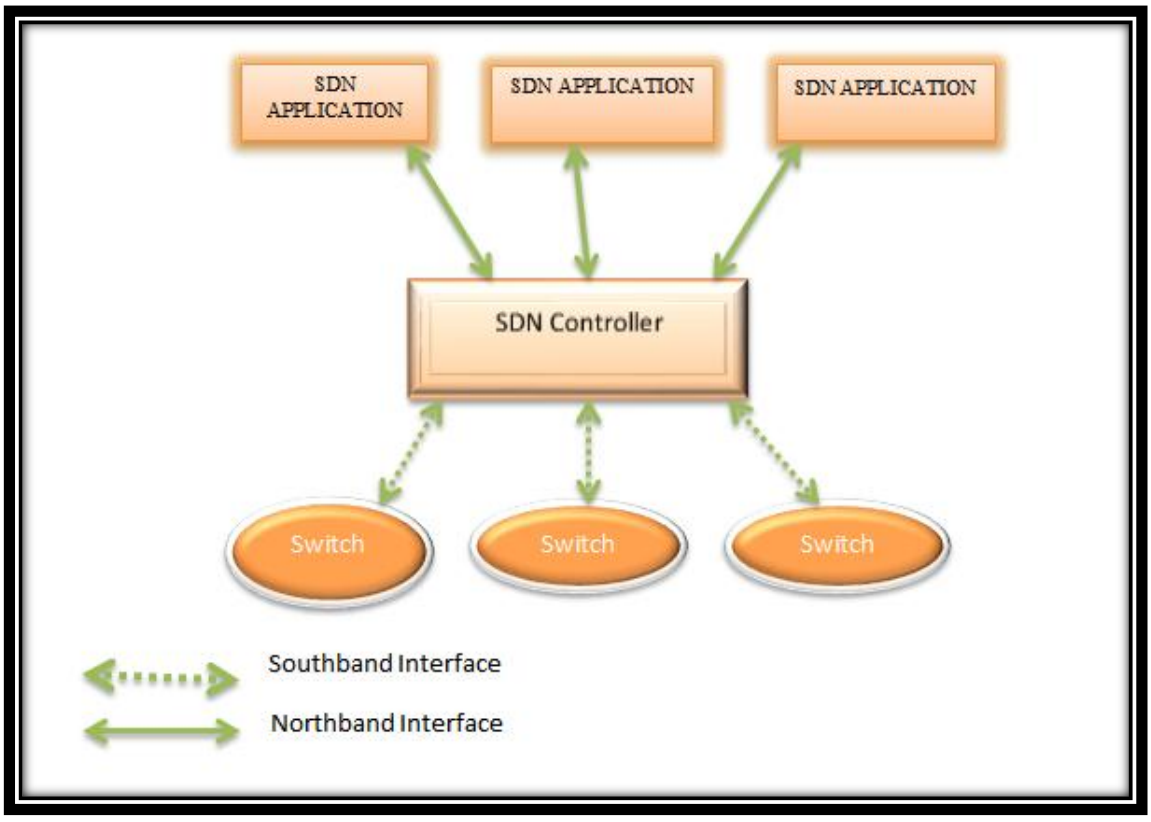


Figure 2.2: The SDN Network Technology Architected

2.2.2 The Relationship between SDN and Virtualization

The capacity to simulate hardware elements like storage, computer devices, and network resources via software is known as virtualization.

A virtualized software object is created by virtualization that precisely replicates the capabilities of the underlying hardware. By running them as separate instances of software over the operating system (OS) of these devices, virtualization decouples these entities from the underlying hardware in the meantime. Then, depending on the requirements of the system, it is possible to dynamically construct or destroy these entities [55][56].

SDN should be coupled with virtualization to enable its flexibility and expand its softwarisation capabilities. Although virtualization and SDN are distinct ideas, they are connected[56]. This connection results from the SDN's assistance in the development and standardization of the Network Operating System (NOS), which serves as the foundation for virtualization. Additionally, SDN feeds centralized control, which greatly simplifies and improves the administration of virtualized equipment. In contrast, virtualization promotes SDN programmability and offers potent tools for testing and simulating SDN network innovation[56][57].

Network virtualization (NV) makes use of the existing network infrastructure to offer traffic and address separation as well as multi-tenancy, which is often seen in datacenter networks. In order to implement stretched/extended networks a mobile VMs and a dynamic resource reallocation were employed [57]. NV is already the most popular use case for SDN controllers [58]. Many of which may be found in datacenters, which mainly employ two methods for NV placement [58]. To enable a hop-by-hop virtualization, the fabric can first be directly programmed [57], [58]. The next step is to create a network overlay.

The goal of Network Functions Virtualization (NFV) is to use virtualization technology to simulate operations that are now performed by network devices on high-volume, commodity servers, switches, and storage devices. Virtual network functions, which can be moved to different network locations and instantiated as necessary without the need to install new equipment, can be used by network operators to replace proprietary middle boxes and network devices .Network operators may provide data plane services like NAT, WAN acceleration, and web caching while NFV offers control plane services like content-centric networking, on-demand virtual networks, and binding of cloud networks by combining NFV with SDN[59], [60].the relation among the three concepts that was discussed above is explained in the Figure(2.3).

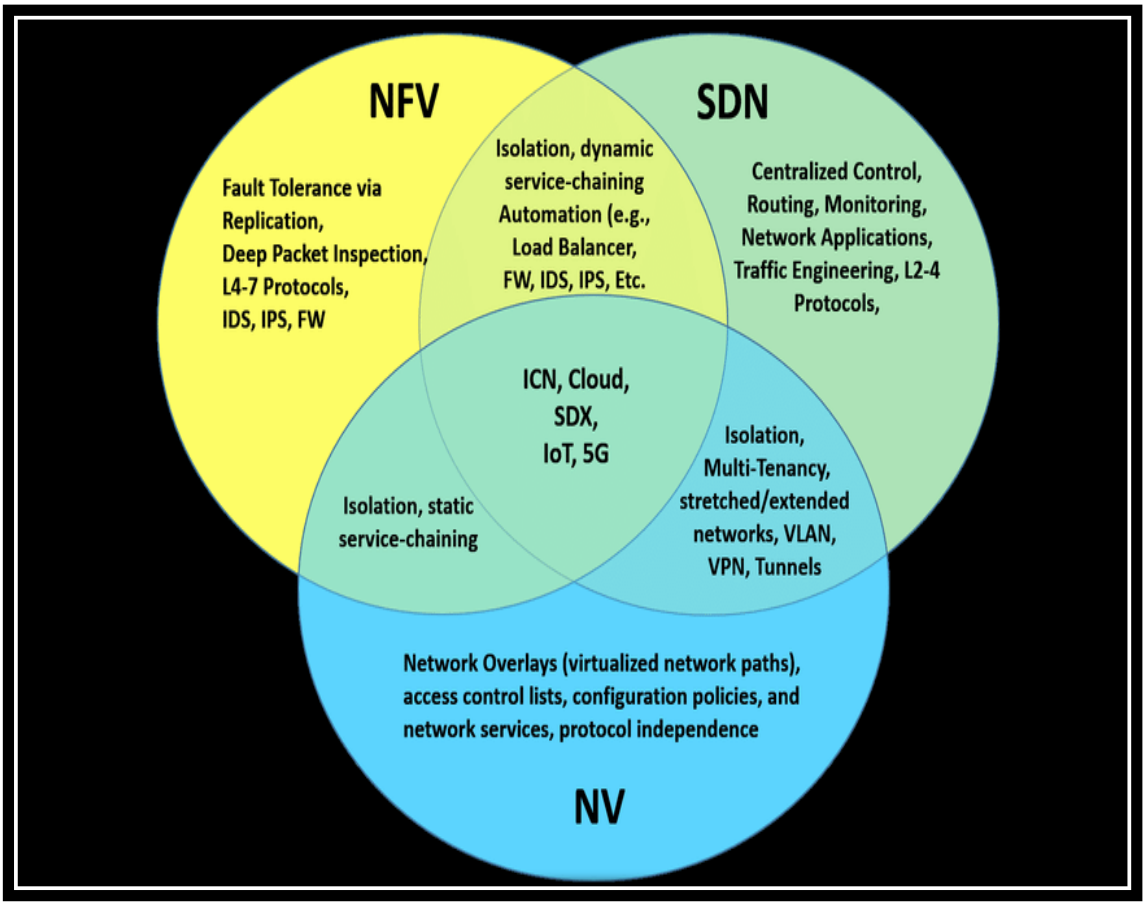


Figure 2.3: The Relationship Between SDN, NV and NFV[61].

2.2.3 OpenFlow Protocol

OpenFlow, as depicted in Figure(2.4), is the first proven standard communication interface in SDN's architecture between the control and forwarding layers . Developed by the OpenFlow Network Foundation (ONF), this protocol allows the controller to communicate with the forwarding layer of networking devices such as physical and virtual switches and routers. Furthermore, it enables management of the forwarding plane. Finally, OpenFlow, which is part of the controller's Service Abstraction Layer (SAL) or southern layer, is a standardized communication interface that will let network devices (switches, routers, etc.) to interact with one another [61][62].

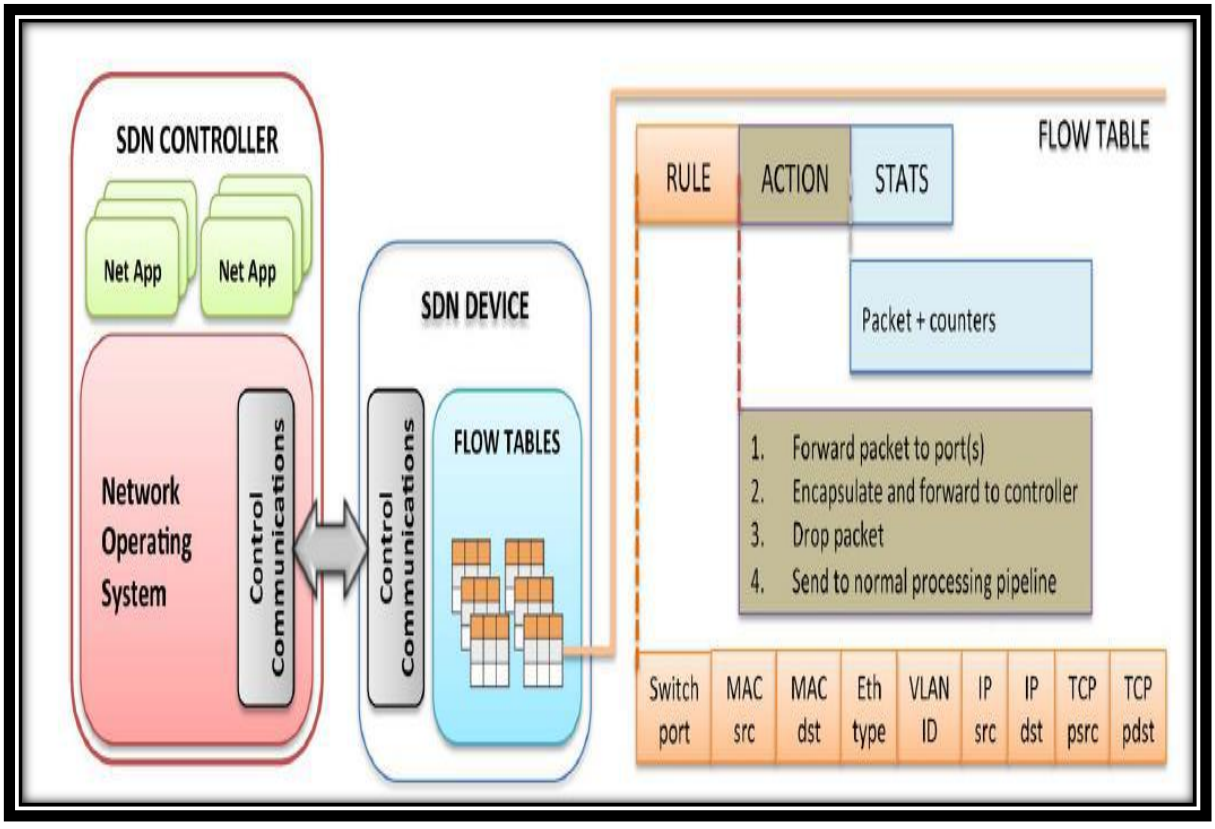


Figure 2.4:The OpenFlow-enabled SDN Devices*[62]*

2.2.3.1 OpenFlow Protocol Messages

A conversation between an OpenFlow switch and a controller is known as the OpenFlow protocol. To provide a safe OpenFlow connection, the protocol is often operated on top of SSL, or Transport Layer Security (TLS). The controller now has the means to edit the flow tables by adding, modifying, and removing entries. The OpenFlow protocol is capable of handling the following three varieties of message[63][64]:

- **Controller-to-Switch:** These exchanges of information are prompted by the controller and they need a reply from the switch on occasion. The controller may manage the switch's logical state, such as configuration and information regarding flow and group table entries, with the use of this collection of messages. Additionally, the packet-out message is covered here as well. When a switch transmits a packet to a controller and the controller chooses to forward the packet to an output port on the switch instead of discarding it, the switch must send this message.
- **Asynchronous:** These messages are sent out automatically, without any involvement from the controller. In these messages, the controller may find information about a wide range of statuses. When no matching flow table entry is found, the switch may send a packet to the controller using the packet-in message, which is also provided.
- **Symmetric**: These transmissions are not being solicited by either the controller or the switch. Simply, they get the job done well. When first connecting a controller to a switch, a lot of "hello" messages are sent back and forth. The controller and the switch employ echo request and reply messages to test the latency or

bandwidth of the link between them, or to simply verify the device is functioning properly. As part of the development process for future OpenFlow versions, the experimenter message is being used to try out new functionality.

2.2.3.2 Installation of Flow Table Entries

Switch flow table entries for flows are often introduced in one of two ways: reactively or proactively. When anything like a connection failure or a change in ACL rules occurs, it may be necessary to update the flow table as well. Once the new flow or ACL rules are in place, the old entries for the old flow must be removed and the new entry for the new flow must be added[65].

- **Reactive Flow Table Allocation**

  During network startup, the switch flow table is initially empty . As soon as a switch connects to the controller, the two communicate to agree on the features that each will provide. This allows the controller to individually identify each connected switch. There are two possible scenarios in which the flow table will be completed. When the switch gets its first packet of flows, the first occurrence of this phenomenon happens. There will be a table-miss because of the missing entry[65]. The switch employs a reactive method to deal with the table-miss by means of a packet-in event, which is then sent to the controller. The controller determines the appropriate entry and updates the flow table of the relevant switch. The second case, shown in Figure (2.5) , entails reinstalling flow entries due to flow expiration or flow change[65].

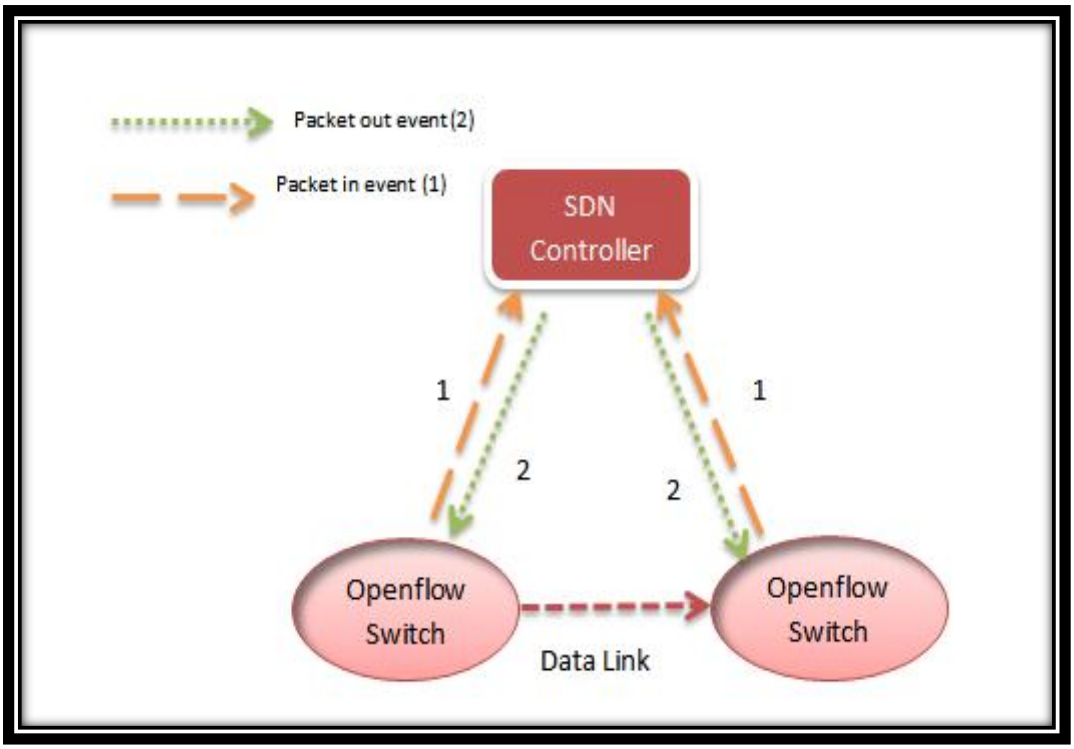


Figure 2.5: The Reactive Entry Installation Concept

- **Proactive Flow Table Allocation**

  Since the controller may establish forwarding rules before the flow actually arrives, the packet is processed much more rapidly in the proactive mode. The switch may process an incoming flow whose packet header information matches an existing item in the flow table without contacting the controller, as it is shown in Figure (2.6)[65]

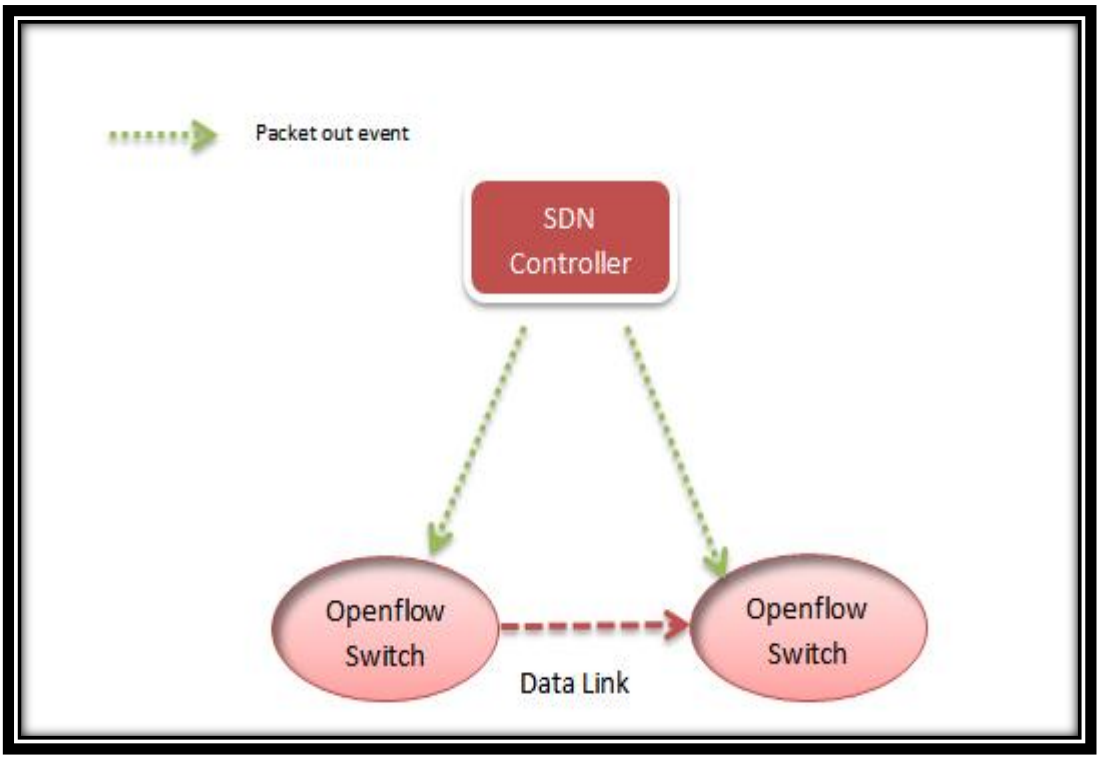


Figure 2.6: The Proactive Entry Installation Concept

2.3 Distributed Software Define Network(DSDN)

The control plane should generally be physically dispersed while providing logical centralization for scalability, availability, and resilience [66]. By taking these steps, be confident that in the event of a controller failure, another controller will be ready to take over network control. Distributed control plane design, however, is undoubtedly one of the most difficult issues in software defined networking[67][68] .Running more than one controller concurrently and a number of disjoint pathways between them for fail-over in the event of link and node failures results in control plane redundancy. Calculating maximum of controller failures before the network going down is involved (Equation 2.1).

$$C = 2F + 1 \quad \text{……(2.1)[69].}$$

Where C is a deployment controller in network ,F is a maximum of controller failures before going down.

2.3.1 Distributed SDN Controllers' Characteristic

The implementation of distributed SDN is not governed by any standards, hence many SDN adopters create various distributed SDN controller versions to suit their needs. But despite that, there are a set of basic characteristics that must be available in the system in order to be called **Distributed SDN**[66][70][71][72][73], and these characteristics are:

1- East/Westbound API

The communication route reserved exclusively for DSDN controllers is the East/Westbound API. For the purpose of coordination, this API offers a link between several SDN controllers. For instance, SDN controllers can create a global network state by sharing their local network state using this API. Another instance of how this API is put to use which is in the SDN controller failover system[71][72]. Other controllers must be prepared to step in and take over the job of the failing controller if one of the SDN controllers in the cluster fails. Finally, the cluster leader can employ a load-balancing system among the SDN controllers using this API. Network performance and efficiency are at their highest when the leader can evenly spread the network's weight across the SDN controllers[71].

2-Network Domains

A single SDN controller manages a domain-named section of the wide area network in a distributed SDN controller. A domain of an SDN controller is defined as the collection of all switches that are directly linked to the same controller and all hosts that are connected to that

switch[72].Depending on the organization in charge of that particular network, network administrators may divide the wide area network into a number of domains. The area where the same company manages the network is known as the administrative domain[72].

3-Local network state:

Local network state, which is kept in the SDN controller locally, is a representation of the status of the network in a particular domain at any given time. The host joining or leaving the network, or a connection going up or down, is just a few examples of events that the SDN controller can be aware of this state[70][72].

4- Global network-wide state

The global network state, which is a representation of the status of the network across several domains, is also required by the SDN controllers. Each controller in many domains must share portions of their local network state with other controllers in order to build a global network state. The global network state is then created by SDN controllers by combining their local network state with that of the others. There is no set standard that dictates what data should be shared across SDN controllers[70][72].

2.3.2 The Basic Distributed SDN Architecture

How to construct the Distributed controller architecture is crucial when installing many controllers. The fundamental Distributed controller architecture may be split into **flat design** and **hierarchical design**[70], In **flat design**, a network is divided into different domains, each of which is managed by a controller located within of its own local network view. In order to acquire a comprehensive perspective of the

network, controllers interact with other parties via their east-westbound interfaces. The flat design of a Distributed controller is shown in Figure(2.7). HyperFlow , Onix ,Opendaylight and Onos[70];are common examples. The flat architecture increases the control plane's capabilities, but it also necessitates complex controller administration and additional control overheads. To provide a consistent network view, the controllers must often communicate with one another.

Hierarchical designs employ two-layer controllers: a domain controller that operates switches in its local domains and executes local control programs, and a root controller that oversees domain controllers and keeps track of the entire network. The hierarchical controller structure found in Kandoo [70] is typical. The root controller interacts with domain controllers to obtain domain information, but the domain controllers do not interact with one another. The fundamental hierarchical organization is shown in Figure (2.8).

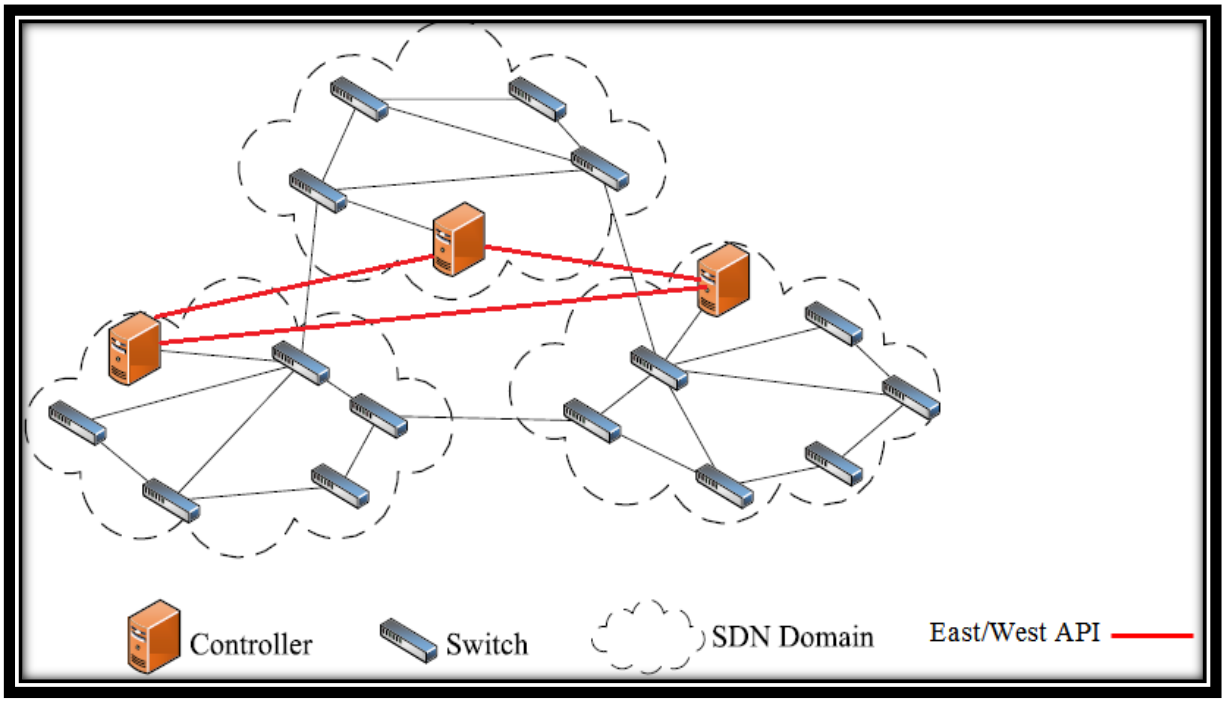


Figure 2.7 : The Flat Architecture Distributed SDN[70][71]

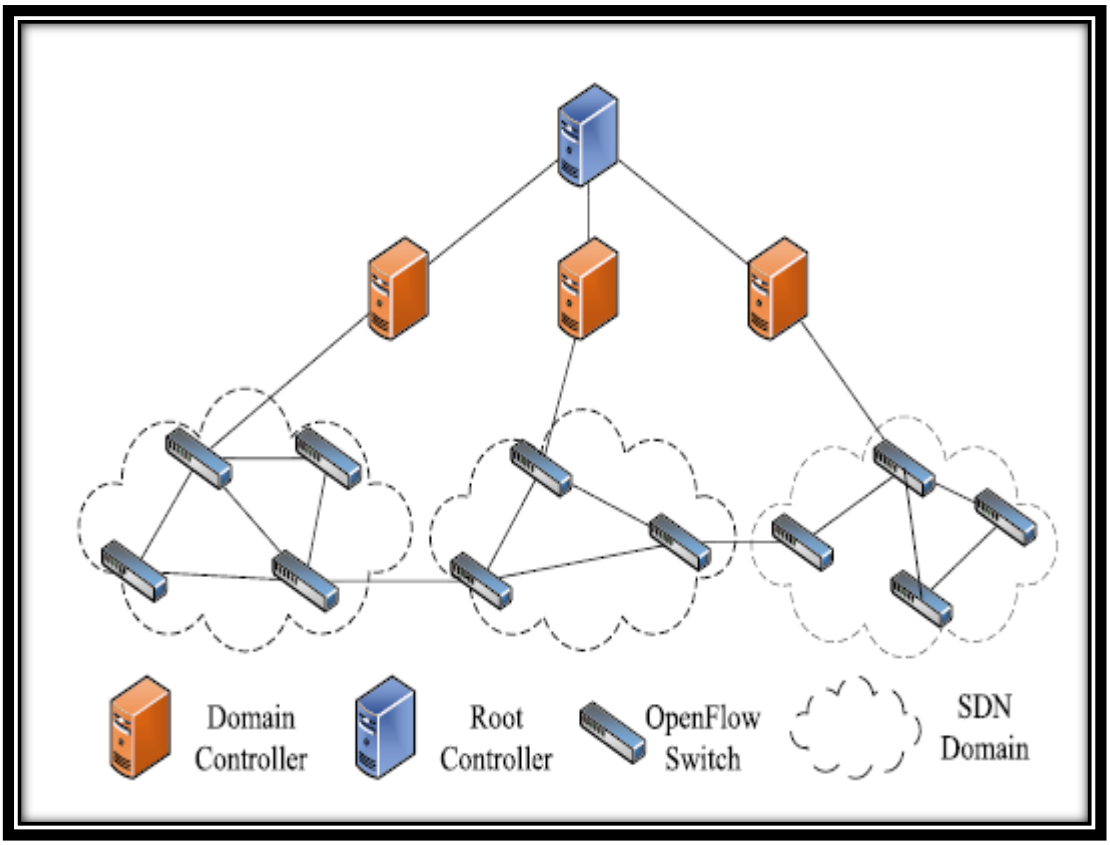


Figure 2.8 : The Hierarchical Architecture Distributed SDN[70][71]

2.3.3 Consistency Concept in Distributed SDN Architecture

The distributed design of SDN has many apparent benefits, but there are also certain restrictions and disadvantages to this approach. Consistent data among the many controllers is the main problem. Any momentary network split can lead to anomalies in the system, compromising a portion of the SDN since updates to the rest of the system would not reach the affected controller, making its data out-of-date[74][75].

2.3.3.1 The Consistency Models

Consequently, based on their consistency guarantees, distributed systems can use two models, as follows[74][75]:

1-Eventually Consistent Model:

An ultimately consistent model would "eventually" update

each of its member controllers, as suggested by the name. This kind of paradigm is used to share data that is less time-sensitive or non-critical[74][75][76]. As a result, there may be a period when obsolete or inaccurate data is present on a hacked controller, but gradually the data becomes consistent across the whole network. Even if this information is out-of-date, the network's performance is not significantly impacted. Additionally, a finally consistent model is employed when system availability is valued more highly than accuracy, allowing switches to access data even if it has been out of date. Gossip protocols guarantee eventual consistency utilizing the epidemic style of propagation, whereby random pairs of neighbors check their version of the data, known as anti-entropy, and agree on an acceptable final state if concurrent updates have happened (this is known as reconciliation) [75][76].

2- Strong Consistent Model:

On the other hand, a powerful consistent model constantly updates all of its controllers. Switches cannot access the data until the controller has been properly updated[75]. When the information being transferred is crucial and out-of-date information might cause irregularities in network performance, this sort of model is utilized. Strong consistency models place a limit on system availability in favor of consistency, preventing access to controllers with out-of-date data until they are changed. Strong consistency is provided by consensus-based protocols like Raft [75], which demand that the majority of the replicas accept

the change before the leader commits it and uses it to build the client response.

At the tradeoff of higher synchronization and transmission costs, the strong consistency model makes sure that all the distributed controllers have access to the most recent network information. Strong consistency during recurrent state update might obstruct state progression and shutdown the network, which would prolong switch to controller latencies. Strong consistency models like this provide considerable scalability challenges. However, eventual or weak consistency models allow for concurrent read operations, and these read operations could momentarily produce values that are different from the updated values[75][76]. Such divergent values supplied by the SDN controllers may result in a global image of the network that is inconsistent, which may cause inappropriate application behavior. Unpredictable actions from the control plane A control plane's consistency might vary greatly. The performance of a network can be greatly impacted by inconsistency in the control plane[76]. Designing for a global view that is consistent across all controllers while balancing policy enforcement and performance is a challenge[76].

2.3.4 Multiple SDN's Controllers

As demonstrated in Figure(2.9), the switch may communicate with either a single controller or multiple controllers [77][78]. Moreover, the switch can continue to function in OpenFlow mode if any one controller or controller link breaks. OpenFlow(1.3) defines three controller operating modes: master,

equal, and slave [78]. For a controller, equality is the default role. The controller in this role has complete authority over the switch. Slave controllers can only read from the switch. Another option is for the controller to request a switch role change to master, in which case it would have full access to the switch . but the switch would ensure that it was the only controller in the master role and the others would be in the slave role[78].

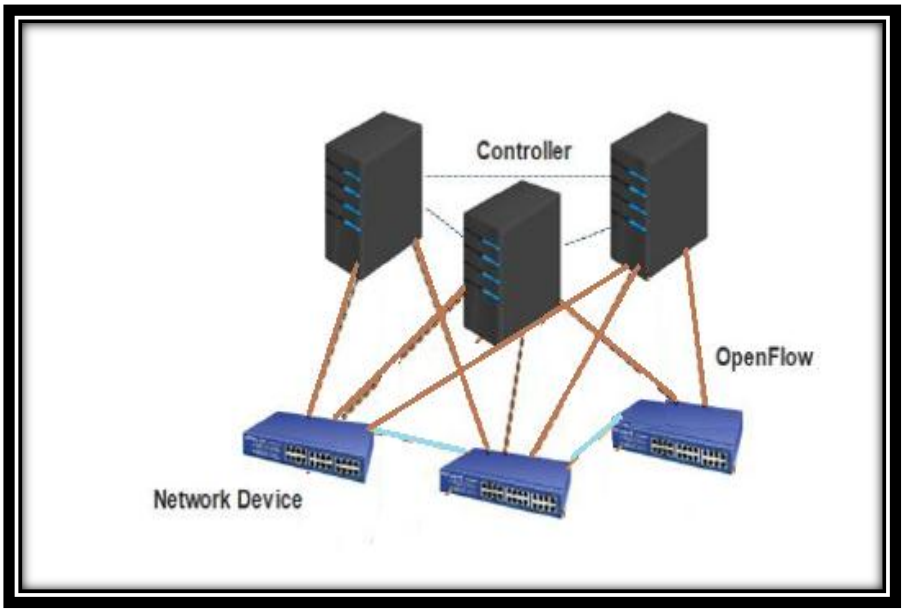


Figure 2.9 Multiple Controllers Managing Multiple Network Devices

2.4 Opendaylight SDN Controller

The Opendaylight controller is one of available DSDN Controllers that was used in the proposed system of this dissertation . By cooperating with the leading networking firms, Opendaylight (ODL) was founded with the goals of accelerating SDN adoption and building a strong foundation for the virtualization functions . ODL is Java virtual machine (JVM) software that can operate on any operating system and hardware that enables Java[79].

As the control plane is centralized, a network with a single controller has various flaws, including security and scalability. To solve this issue, the opendaylight [79] community has devised a clustering technique that allows numerous controllers to function simultaneously, resulting in a high availability and single point failure.

Since the goal is to operate as a single controller with centralized logic, running numerous controllers bring additional challenges. When many controllers collaborate, the global state of the cluster is spread among all the controllers in cluster. Controller's state of each member, SDN's controller, in cluster should be consistent with the rest. As a result, a technique for synchronizing the functioning of controllers' cluster is required. Opendaylight controller employs a variety of strategies and algorithms to handle these issues, which create additional control traffic[79].

2.4.1 The Structural Design of Opendaylight

ODL is a multi-protocol, scalable, extendable, and highly available controller architecture designed to software defines network utilizations. ODL offers a Model-Driven Service Abstraction Layer (MD-SAL) that enables creating applications[79]. The Figure(2.10) shows the framework of ODL;4th Beryllium release that used in this work .

Effortlessly operate on a range of southbound protocols and hardware . It also has internal plugins which expand the network's capabilities and introduce new services. ODL has dynamic plugins, for instance, that make it possible to get both network topology and statistics[79].

Basic ideas serve as the foundation for applications, and MD-SAL leverages them to establish communication patterns and deliver services and behavior depending on developed-supplied YANG models [79]:

• Data trees: the ability of modeled and displayed the data with tree or sub tree.

- Operational data tree is sued by service providers who use the MD-SAL, describe the system's current condition and exemplify a feedback loop for apps to monitor network/system situation[79].

- Configuration data tree indicates the system or network's intended state. Customers fill it with their intention-expressing words[79].

• Instance identifier is an unambiguous identification that is specific to a node or sub tree in a YANG data tree. A node or sub tree from conceptual data tree is retrieved using this method[79].

• Notification: From the provider's point of view, it is an asynchronous, temporary event those consumers that can consume and perhaps react to[79].

• RPC: request-reply message pair in asynchronous fashion. Consumer initiates a request, which provider then receives and responds to with a message[79].

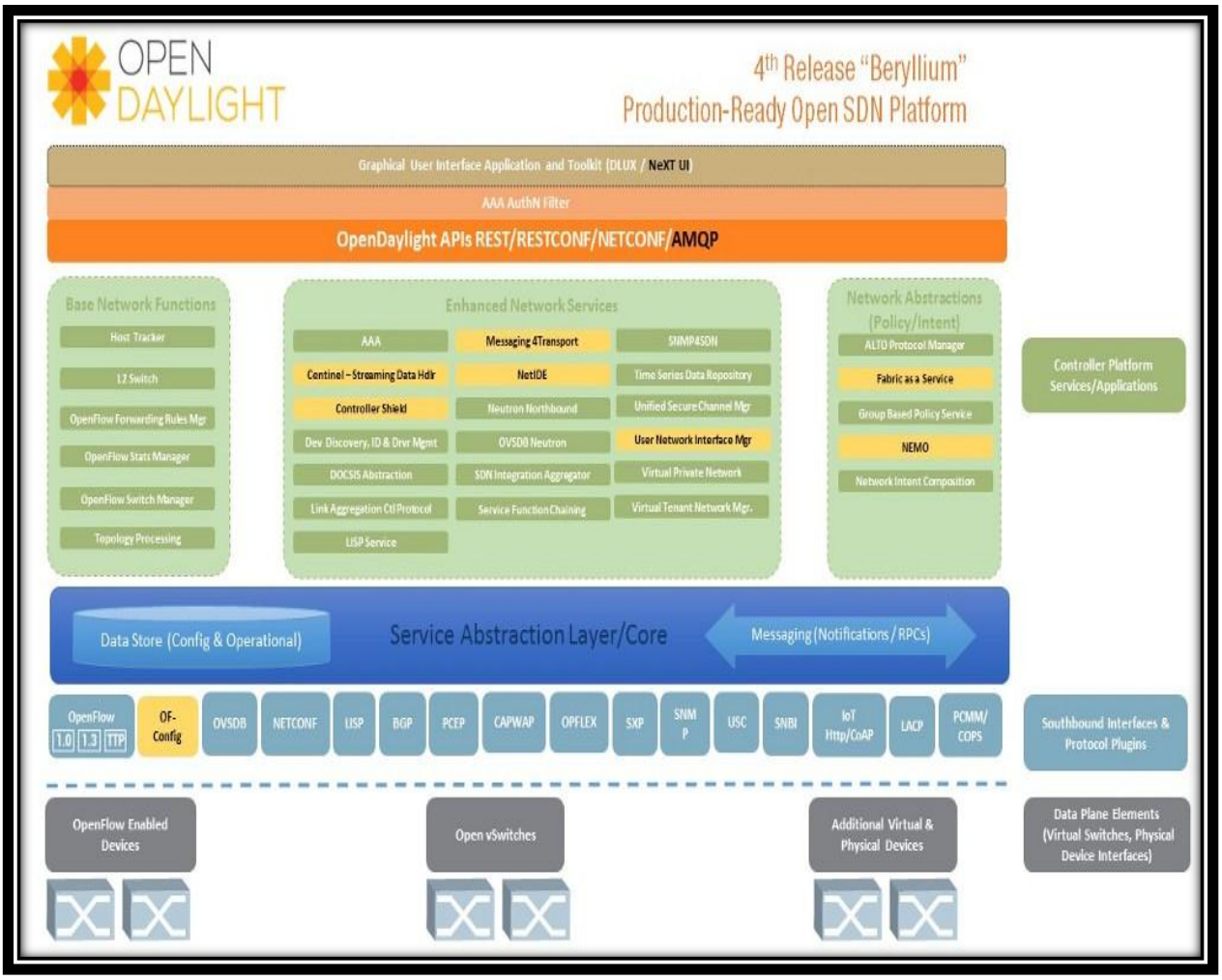


Figure 2.10 The Architecture Framework OpenDaylight Release,4th Beryllium [79].

2.4.2 Service Abstraction Layer (SAL)

The controller is used by applications to acquire network intelligence, execute its algorithm for analytics, and then orchestrate the new rules throughout the network. On the southbound, many protocols, such as OpenFlow 1.0, OpenFlow 1.3, Border Gateway Protocol (BGP-LS), and others, are available as plugins. The OpenDaylight controller begins with a southbound OpenFlow 1.0 plugin. Other OpenDaylight authors start contributing code to the controller. These modules are dynamically connected into a Service Abstraction Layer (SAL)[79].When writing applications for OpenDaylight, developers

have the option of using either the API-Driven SAL (AD-SAL) or the Model-Driven SAL (MD-SAL).

- **API-Driven Service Abstraction Layer (AD-SAL):** The manual coding of the adaptation functionality between southbound and northbound plugins was a requirement for the initial SAL developers. They also had to identify which APIs were utilized by each individual plugin. It was soon clear that scaling an effort of the size of OpenDaylight would not be possible by manually developing the SAL APIs and adjustments to enable additional plugin features[79].
- **Model-Driven Service Abstraction Layer (MD-SAL):** On the basis of the interface models and data specified by developers of applications, the MD-SAL is an expandable middleware component with message-bus inspiration that offers messaging and data storage features [79].
    - It offers infrastructure and a framework for application and inter-application communication by defining a communal layer, ideas, the components of a data model, and patterns of message [79].
    - It offers standardized assistance payload serialization and adaptation, as well as user-defined transport formats as XML / JSON [79].
    - MD-SAL employs the YANG such a modeling language to the data specifications and interface [79].

The primary dissimilarities concern the respective API usages of provider and consumer plugins. Providers in MD-SAL build a model of the information and services, they make available. The models are written out as YANG schemas. The plugin's APIs are then standardized

for its users with the help of a YANG compiler. Since these APIs are automatically produced by a tool, they share a great deal of semantic and functional parity.

2.4.3 RESTCONF Interface

The RESTCONF refers to any system-to-system interface that only employs HTTP to any format of communication . Within the framework of ODL[79]:

- NETCONF protocol, supported operation is specified , created data stores.
- Modeling of data structures and their iterations is done using the YANG modeling language.
- The SAL's RESTCONF protocol explains way of translate a YANG specification to the a REST interface, that subsequently for a purpose of data's access/modify .

Access to the controller's data storage is made possible through RESTCONF. XML and JSON are both supported for translating YANG models for data's request/response according to the YANGTools project [79]. The Content-Type field in the HTTP header must be used to specify the media type of request data (input).The Accept field in the HTTP header must be set to reflect the media type of the response data (output).Instance identifiers are commonly used in RESTCONF endpoint routes (URIs ' URI link: /restconf/config/ <identifier>') to direct users to particular nodes for actions.

2.4.4 Distributed Datastore

The Akka framework forms the basis of ODL's distributed datastore, which is split into three shards: inventory, topology, and

toaster. The cluster's flow rules for the managed switches are stored in the inventory shard, nonetheless, information about the topology of the network is stored in the topology shard[80][81] .As more flows or switches are added, a larger inventory or topological shard is the result. Distributing and replicating shards among the cluster nodes, with the first shard being copied from Member 1 to Members 2 and Members 3. At the beginning, member 1 updates the Leader with its information. There is no direct replication between members, thus when new information is discovered, it is communicated to the other members by the Leader. They are tacked on separately[82][83][84].

With the toaster shard, you may specify which Remote Procedure Calls (RPC) are allowed in each shard and what should happen when an RPC is received. Initialization begins with the assignment of each shard to a particular cluster node, which may or may not be the same for each shards[83]. In order to avoid data loss in the case of a node failure, it is then replicated to all other nodes in the cluster .Each copy of a chunk in a distributed system is constantly synced with every other replica to ensure data consistency .Because of the importance of reaching agreement in a distributed system, ODL employs the Raft algorithm [84][85].

2.4.4.1 Raft Algorithm

Each cluster member is assigned one of three states leader, candidate, or follower by the Raft algorithm for each shard . Before doing anything further, Raft calls for an election to be held among the cluster members. One of the members in the cluster will be elected as the shard leader if it receives the most RequestVote messages. As demonstrated in [83], ODL designates one member as the leader of all

shards by default. Typically, this is the member that initiates the process. The leader must identify inconsistencies across the shard's associated cluster members and then replicate the correct data by transporting an AppendEntries signal. Raft also regularly sends out "heartbeat" signals in the form of empty AppendEntries messages[82][83].If the leader does not send a HeartbeatRsp message within the allotted period, the followers will switch to the candidate state and start the election process over again. A follower notifies the leader through generate transaction message whenever it acquires new information, such as a newly connected switch[82][83][84]. The leader then updates the other members of the cluster with the transaction details. As all shard changes are sent to the leader, and it is the leader's job to send those updates to the rest of the followers, the leader's function is vital.[85] [86][87] Figure(2.11)explain process of the leader election with Raft.

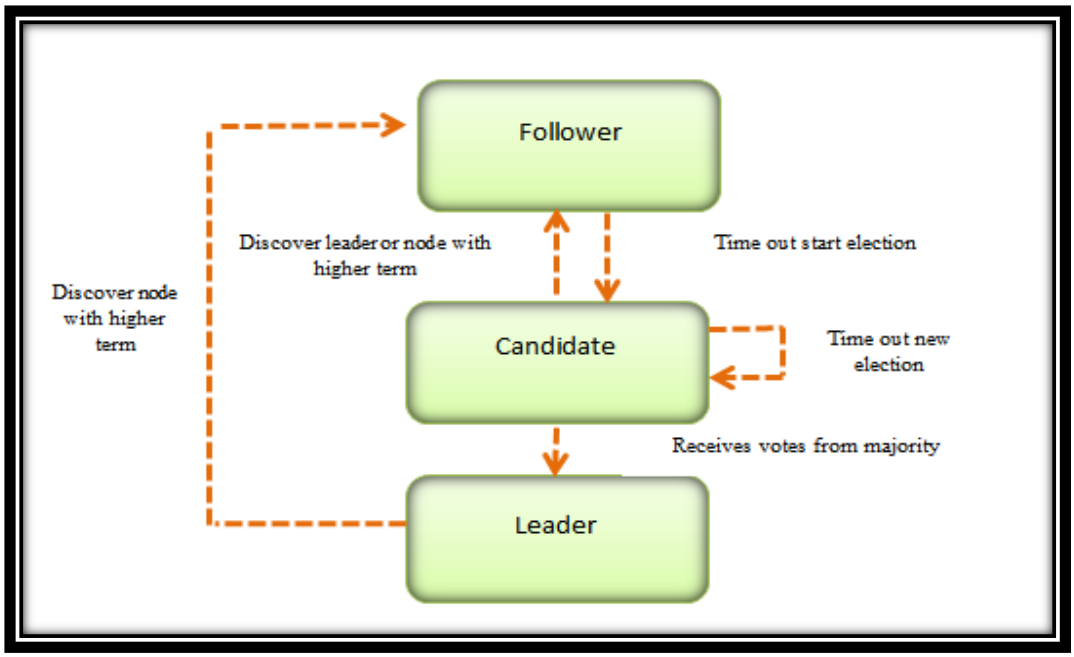


Figure 2.11:The Leader Election with Raft[82][86]

2.5 Blockchain Technology

Due to the revolutionary changes brought about by the Internet in the past several decades, it is now quite simple to successfully communicate information over great distances with people you do not know and do not trust[88].There are still numerous obstacles to overcome, despite the fact that many paper-based procedures have been replaced by electronic ones and that businesses spend vast sums of money on cyber threat prevention, detection, mitigation, and recovery. With blockchain, it is now able to use technology to guarantee both the data and the process of data exchange[88].

2.5.1 The Blockchain Concept

There are generally three distinct kind of network: distributed, centralized, and decentralized Picture 2.3 Web 2.0 is the centralized system that is now in use, and it stores all communication between two parties on a server that is privately run and maintained. The distributed system, which is a subset of the decentralized system and enables nodes to store and transfer material that has been published from a specific entity without having an influence on the stored data, acts as public storage for the network[88]. In the same way, decentralized data sharing makes use of a distributed ledger system[88]. Each node may, however, make its own choices, which may affect the data it holds, as shown in the Figure(2.12).

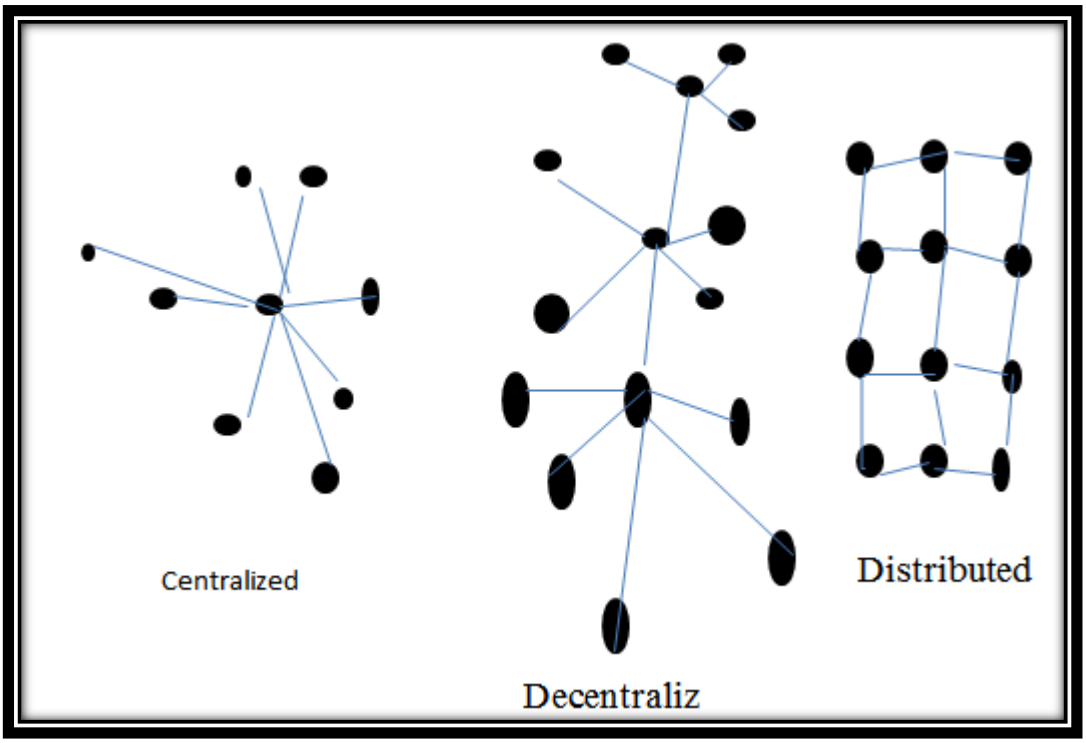


Figure 2.12: The Centralized, Decentralized, and Distributed Networks[88]

People assert that the Internet will become decentralized using the technologies of data distribution. In contrast, decentralization requires that each participant gets a copy of the distributed ledger (network's database) in order to be an active node capable of regularly updating the ledger. Due to the fact that any member of a decentralized network may access a full copy of the ledgers, members do not need to trust one another in order to verify the accuracy of the received ledger[88][89].

With the help of blockchain new technology, centralized trust-based systems would be replaced with decentralized trust-free ones. That is to say, it offers a system of trust-free exchange between any parties involved without the requirement for a third party to be engaged in the transactions. Because everyone participating in the transaction can see and confirm it immediately, Blockchain technology may be utilized to do away with the need for centralized organizations and databases, resulting in truly transparent and trust-free systems[88][89].

However, a lot of people mistakenly believe that Bitcoin's (a digital currency) supporting technology is Blockchain [89]. Although this was the initial intent, Blockchain is capable of much more. Despite how the name "blockchain" sounds, it refers to a group of distributed ledger technologies that may be used to store and monitor any valuable information[90].

In a blockchain system, the ledger is mirrored in several, identical databases that are each hosted and kept up to date by a different interested party. All of the other copies are concurrently updated when modifications are made to one copy. As transactions take place, records of the money exchanged and the goods traded are permanently recorded in each ledger. Ownership cannot be verified or transferred through third-party intermediaries[89][90].

In essence, blockchain is a data management platform that manages blocks of data rather than a single entity, storing a comprehensive record of committed transactions and digital events in a sequence. It is a clever fusion of several technologies, including distributed databases, peer-to-peer networks, and encryption techniques. It offers a decentralized means to create a confidence (p2p) system and holds great promise way of computation and collaboration [89][90] . Properties like security, immutability, robustness, transparency, auditability, and softly are provided through using the block chain[90],so several well-known software applications use this architecture. By using a time stamp to confirm each data transmission in a peer-to-peer network, it is possible to include a safe chain of previously transferred data. Without a centralized authority, the data interchange can be managed, In more detail, a blockchain stores a record of every data exchange, in which the record is referred to as a "ledger" and every data

exchange is referred to as a "transaction". Each confirmed transaction can be included as a "block" to the ledger[90] ,as it explain in Figures(2.13),(2.14).

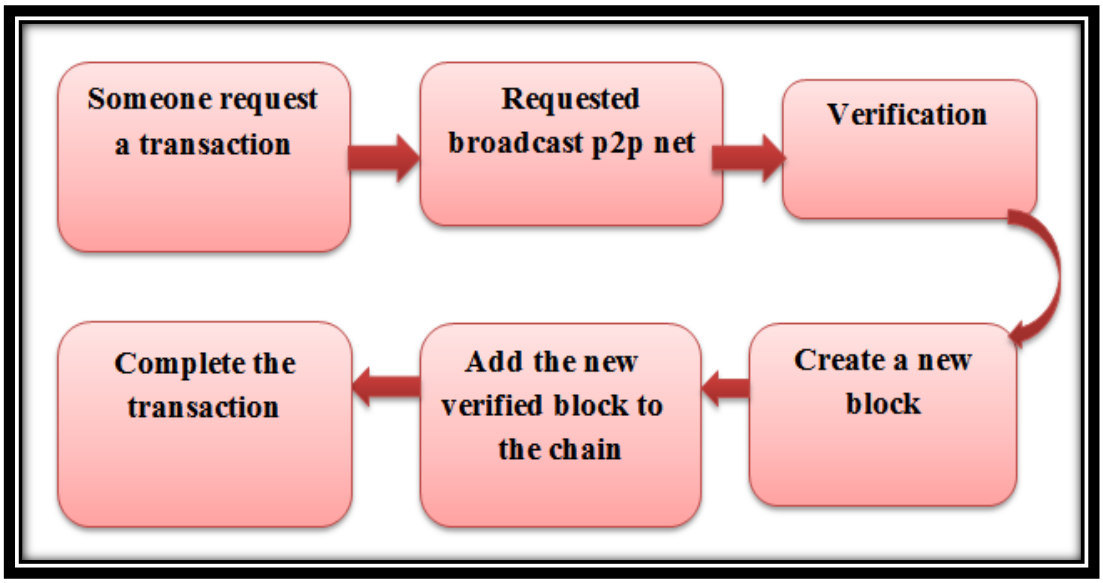


Figure 2.13 The Overview of Blockchain Concept

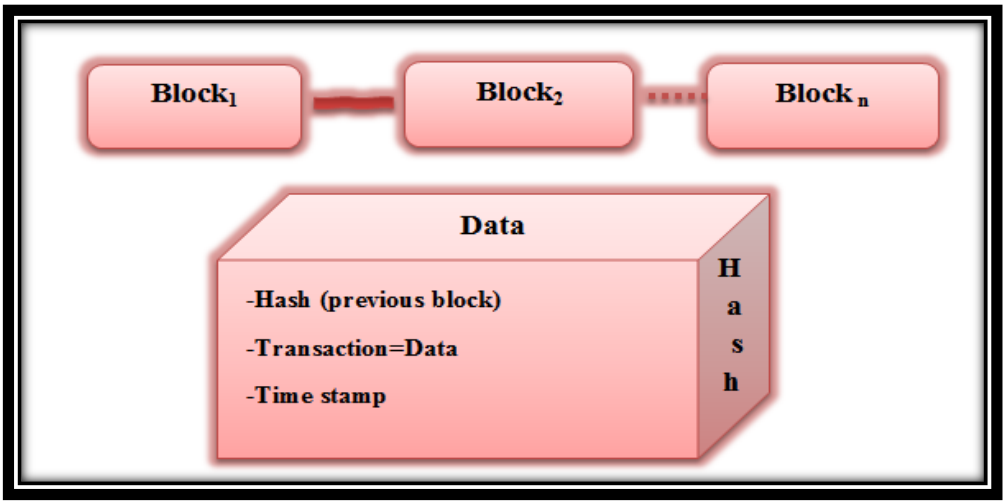


Figure 2.14: The Data Block structure" A blockchain is an electronic ledger that links data in a chain of blocks."[90]

The main components of Blockchain[90][91]

- **Transaction**: A blockchain's most fundamental building block is the transaction. A transaction may be an actual exchange of money or goods, or it can be as basic as a hash file reflecting a single word or as sophisticated as an entire computer program. The entity being sent is digitally signed by the sending party, and the moment it was received by the network is recorded ;time stamped.
- **Block:** new transactions are grouped together in a "block" or can each New transaction in a "block" . Several variables, including the specific blockchain protocol, determine the size of the block and the rate at which it is filled. The miners have to go through a difficult mathematical challenge before they can create a new block.
- **Chain:** When a block is complete, it is added to the chain and linked to the previous block using a sophisticated hashing algorithm that makes blocks increasingly secure the longer they have been in the chain. No previous data may be altered once a new block has been added to the chain; immutable. As the blockchain expands, it creates a chronological record of events since transactions and blocks are built in the order they are received. It is akin to an accounting ledger in that it keeps track of transactions.
- **Consensus algorithm:** Network members do a series of checks to ensure the validity of the updated blockchain that contains the new block before it is published permanently to the blockchain. There are a variety of ways to do this, but ultimately, all of the network's members must agree that the block was built and

inserted properly according to the protocol. The blockchain idea relies on this consensus reaching because it establishes a single record of "truth" that can be accepted by all parties. After consensus is obtained, the blockchain is distributed to each member of the network. Every node in the network has a full and identical duplicate of the database, so any changes made to one copy would be instantly apparent to everyone else.

2.5.2 Blockchain Type

Using a secure network to conduct transactions or exchange information is the most fundamental use case for a blockchain. However, the use of blockchain and distributed ledger technology varies depending on the situation[92][93].

- **Public blochchain**, a distributed ledger system without constraints and permissions is known as a public blockchain. Anyone with an internet connection may sign up on a blockchain platform to join the network as an authorized node and become a part of the blockchain. It is permitted for a node or user who is a member of the public blockchain to view recent and old data, confirm transactions or complete proof-of-work for an incoming block, and engage in mining. The mining and trading of cryptocurrencies is the most fundamental usage of public blockchains, examples include Bitcoin[94][95] and Ethereum[96].
- **Permissioned blockchain,** an exclusive or permission-based blockchain that only functions in a closed network is referred to as a private blockchain. In businesses or organizations where only a few people are part of a blockchain network, private blockchains are typically employed. The governing organization is in charge of determining the degree of security; authorizations, permits, and

accessibility.Table2 shows the main difference among these Blockchain types[92].

- **Hybrid blockchain** includes components from both public and private networks. Alongside a public system, businesses have the option to put up private, permission-based networks. This manner, just some pieces of information included in the blockchain is made available to authorized users, while the rest of the information remains accessible to the public.

Table 2.1: The Main differences between public and private blockchain approaches[92][97][98]

Property	Public blockchain	Private blockchain
Consensus determination	Each node could take part in the consensus process.	Is fully controlled by one organization and the organization could determine the final consensus.
Read permission	Public	Could be public or restricted
Efficiency	Low "As there are many nodes in the public blockchain network, it takes a long time for blocks and transactions to spread. As a result, there is a low transaction throughput and a significant latency"	High
Centralized	No	Yes "controlled by single group"
Access	Permission less	Permissioned
Consensus process	Need to prove of work(PoW)	Not need(PoW)

2.5.3 The Blockchain Hash Function

To ensure the integrity of the blockchain, each new block includes the hash of the previous block's header. Because of this, it is difficult to alter a blockchain's blocks undetected. Since modifying one brick requires producing new copies of all following blocks, this operation grows more difficult as more bricks need to be altered.

The Secure Hash Algorithms are a family of cryptographic hash functions published by the National Institute of Standards and Technology (NIST) as a U.S. Federal Information Processing Standard (FIPS), a hash is a mathematical operation that takes a file as input and outputs a reasonably brief code that may be used to identify that file [99]. A hash contains the following important characteristics: It is unique; two separate files can never create the same hash since only one of them is capable of doing so. It cannot be reversed, by examining a file's hash; you cannot determine what the file was [99].

SHA-256 is one of the strongest hash functions available. SHA-256 is not much more complex to code than SHA-1, and has not yet been compromised in any way. The 256-bit key is defined in the NIST (National Institute of Standards and Technology) [99]. (Equation 2.2) shows how to get the hash (h) from a message (M) using compression function (H).

$$h = H(M) \quad \text{(2.2)[99]}$$

Where M is the input message and h is the digest generated using the hash algorithm H.

2.5.4 Blockchain applications/platforms Described

1. **Bitcoin** is a program with a specific use (cryptocurrency), a type of digital money. Without a single administration or central bank, it is a decentralized digital currency that anyone can use and join. It may be transmitted directly between users on the peer-to-peer bitcoin network without the use of a middleman [100].

1- Ethereum is an open-source software platform based on blockchain technology that lets programmers create decentralized apps and distribute them.

2- Monax's eris-db supplied a free and open-source platform called eris-db, the first blockchain client on the market with a permissionable blockchain design, to let developers create, ship, and run blockchain and smart contract-based applications for business systems and processes.

3- Multichain is a tool for developing and implementing private blockchains. Through the integrated administration of user permissions, it resolves the linked issues of mining, privacy, and openness.

2.5.5 Smart Contract

An application that can be programmed and executed on a blockchain network is known as a smart contract. Smart contracts are computer programs that may carry out actions automatically in accordance with the conditions of a contract or agreement. They are made to cut back on intermediaries' participation and lessen the expense of enforcement. The code itself is intended to serve as the last adjudicator of "the transaction" it denotes [101] [102].Figure (2.15) shows the smart contract with blockchain technology.

Since the launch of the first smart contract platform, Ethereum [101], in 2015, smart contracts have grown to be one of the blockchain industry's most cutting-edge subjects. Smart contracts are unchangeable and immune to outside attacks according to this kind of self-executing agreement that relies on the code. Without the need for middlemen or other third parties to act as mediators, smart contracts enable the execution of reliable transactions [102] [103]. This feature is particularly beneficial since it greatly lowers conflict concerns and cuts down on operational expenses and operating time. Smart contracts provide a wide range of new applications to address issues in the real world, including insurance /financial services, supply chain transparency, mortgage transactions, digital identification, and records management [102][103][104].

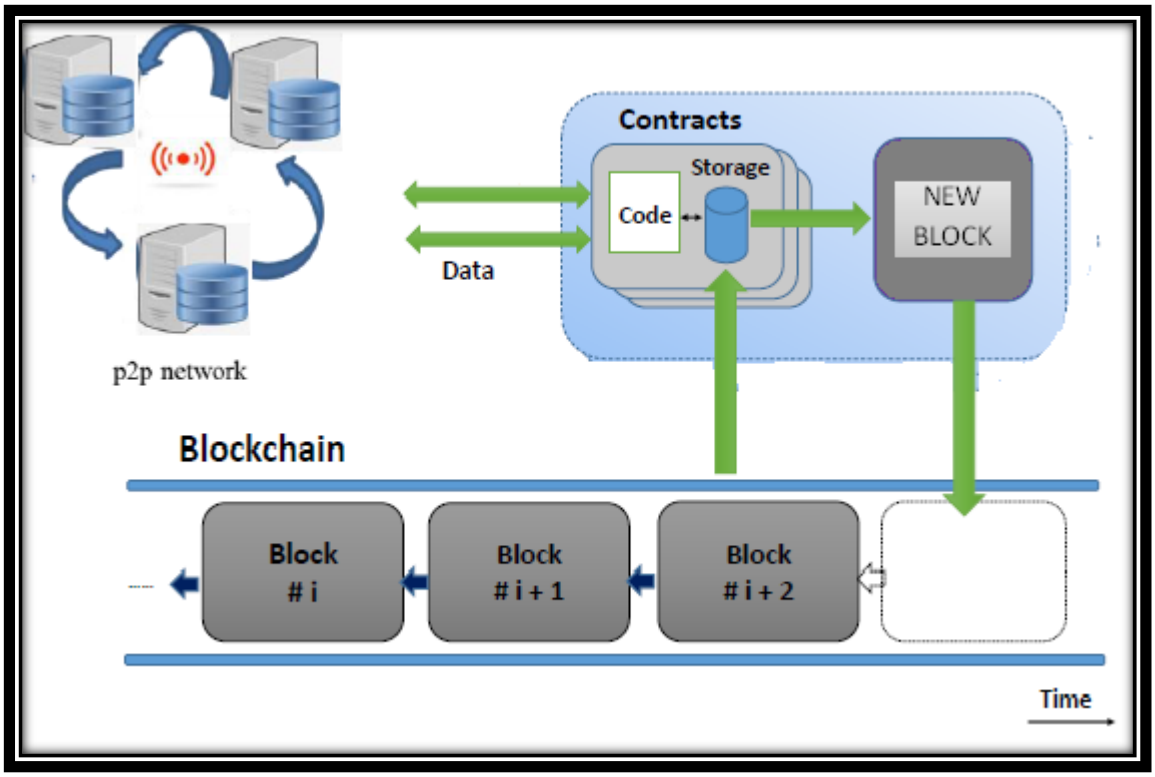


Figure 2.15: The SmartContract Concept with Blochchain Networks

2.6 Experimental Environment

It is difficult to work with such a system directly because SDN is a new technology. Using the emulator might lead to a different answer. In order to simulate the SDN in a virtual environment, the emulator combines hardware and software resources. Utilizing the following, the experimental analysis is carried out:

2.6.1Graphical network simulator (GNS3)

For network simulation, configuration, testing, and debugging, GNS3 is used by tens of thousands of network engineers every day around the globe. GNS3 is used to manage topologies of any size, from a few devices on a laptop to many devices across several servers or in the cloud [105]. On the other hand, GNS3 may be used for a wide range of purposes, including business presentations and proofs of concept. Using new software, like management or SDN software, may be difficult and expensive, but GNS3 makes it easy and cheap. It allows you to test for compatibility with different vendors without having to invest in expensive, specialist hardware. One may divide GNS3 into two distinct pieces of software [105]:

- **The GUI for the GNS3-all-in-one software:** The graphical user interface(GUI) client of GNS3 ,the unified software suite may be used by anybody, regardless of operating system preference, to plan out their network architectures. This is often the case with screenshots [105].
- **The GNS3 virtual machine (VM):** To make use of GNS3, it is suggested to use a virtual machine (VM) implementation, such as VMware Workstation, Virtual box, or Hyper-V, to run it on own computer, or may run it remotely on a server using VMware ESXi, or even in the cloud [105].

For GNS3's all-in-one GUI client software to work, a server process must host and operate the devices in the network. Several alternatives exist for the server software itself: In contrast to the remote GNS3 VM and local GNS3 VM, which are both managed from the computer where GNS3 is installed, the local GNS3 server is managed locally. If you are working on a window machine, for instance, the GNS3 graphical user interface and the local GNS3 server are both processes running in the background [105].

2.6.2 VMware Workstation

VMware Workstation is robust virtualization software for X86 and AMD64/intel64.VMware Workstation enables the bridging of preexisting host network adapters and the sharing of local disks and USB devices with a guest operating system. Simulated disk drives are possible; ISO image files may be used to emulate optical drives, and .vmdk files can be used to simulate hard drives. VMware Workstation Pro provides the option to take a "snapshot" at any time, which preserves the current state of a virtual machine. When a snapshot is restored, the virtual machine is effectively returned to the moment when the snapshot was taken [106] and is unaffected by any changes that occurred after the snapshot was taken [106] [107].

2.6.3 Mininet

Mininet is a network infrastructure simulator used in this study. It is feasible to run several hosts, switches, routers, and connections on a single Linux kernel. Even though it just uses lightweight virtualization, it can make a single computer seem to be a whole network, with all of the same kernel, system, and user code. Mininet's virtual hosts, switches, connections, and controllers all act in ways that are extremely close to

what one would expect from physical hardware. However, they are generated by computer programs. The importance of role-playing a future-use scenario cannot be overstated. Our feature will be invaluable in this study when analyze the controller's behavior as it transports different packet kinds. The Mininet emulator achieves its primary objectives by using the Linux Ubuntu system. This first objective is tied to the platform's model execution capabilities, while the second is tied to the kernel model's ability to perform system analysis, configuration, creation, and interfacing [108].

2.6.4 Open virtual Switch (OVS)

An open-source software switch designed for use in virtualized server settings, with the goal of implementing a switch platform, allows for the expansion and control of the forwarding operations to be programmed, and supports standard administration interfaces can be ported into ASIC switches [108].

2.6.5 Wireshark

Wireshark, a free and open source packet analyzer, is used to probe the functioning of the network. Wireshark, the most well-known and widely-used network protocol analyzer in the world, is an indispensable resource for network engineers. Since it provides granular insight on network activity, it has become the de facto standard for a wide range of industries, including businesses, nonprofits, governments, and educational institutions. Wireshark is first conceived of in 1998 by Gerald Combs, and its continued development is made possible by the unpaid labor of networking specialists from throughout the globe [109].

2.7 SDN Network Metric

Measuring network metrics to characterize its condition is an important part of network management. There are three broad classes of SDN network metrics: topology discovery, traffic, and performance.

2.7.1 SDN Network topology Discover

An SDN controller needs up-to-date knowledge of the network state, in particular the network topology, in order to be able to govern the network and deliver services like routing. Therefore, any SDN system must have a trustworthy and effective topology discovery process. Most SDN controllers utilize the OpenFlow Discovery Protocol (OFDP) [14] to discover network topology, which is based on the Link Layer Discovery Protocol (LLDP) [110][111].

Using Feature Request and Feature reply [110] [111] OpenFlow messages, the controller learns the physical ports and other characteristics of a linked switch. While by producing regular LLDP messages, the controller establishes the switch-to-switch adjacency.

With a network made up of two switches, s1 and s2, Figure (2.16) depicts the OFDP protocol. There are two active ports on each switch, p1 and p2, which are linked to the controller and the other switch, respectively. A LLDP packet is enclosed in a Packet OUT message and sent to s1 by the controller in order to identify the unidirectional link (s1 and s2). Instructions are provided in the Packet OUT message for s1 to use port p2 to deliver the LLDP packet to s2. The LLDP packet is received through port p2 by s2, which then wraps it in a Packet IN message and transmits it back to the controller. After receiving the LLDP packet, the controller determines that s1 and s2 are connected in a

single direction .The opposite link (S2, S1) is also found using the same procedure, as all other switches in the network.

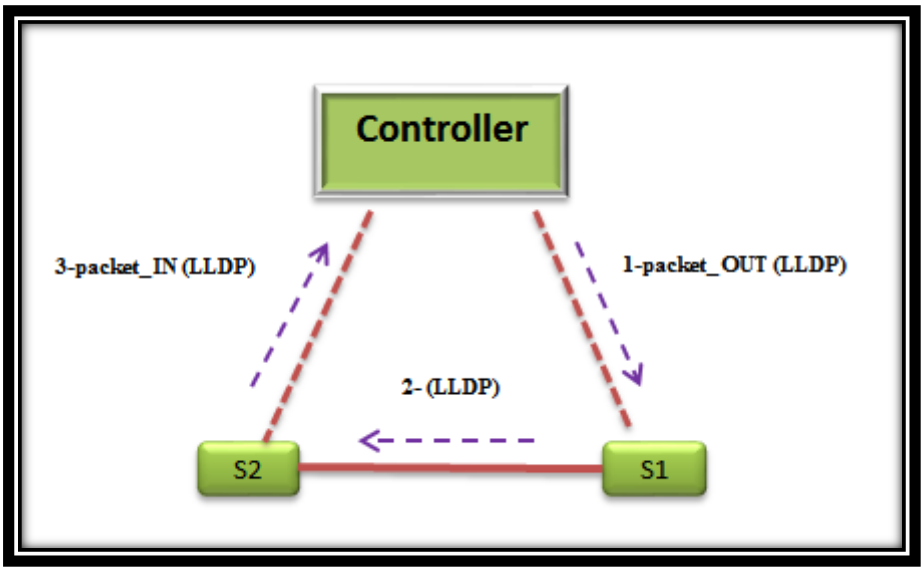


Figure 2.16 The Topology Discovery in SDN

The amount of time required for a controller to recognize a network's whole topology is known as a topology discovery cycle. In order to identify all unidirectional links in the network, the time interval between the controller's first Packet OUT message and its last Packet IN message is determined. The topology discovery procedure is periodically run by default every 5 seconds for ODL in order for the controller to keep an updated topology of the network[110][111][112].

$$T_{\text{topology discovery cycle}} = T_{\text{last Packet IN message}}(\text{LLDP}) - T_{\text{first Packet OUT message}}(\text{LLDP}) \ldots\ldots (2.3)$$

Where $T_{\text{topology discovery cycle}}$ is an amount of time required for a controller to recognize a network's whole topology, $T_{\text{last Packet IN message}}(\text{LLDP})$ is time of last Packet IN message in topology discovery cycle , $T_{\text{first Packet OUT message}}(\text{LLDP})$ the time of first Packet OUT message in topology discovery cycle.

2.7.2 Traffic Metrics

Traffic metrics may look at how users behave on a network by measuring the number of packets sent or the time it takes for packets to travel between nodes. Data flow from OpenFlow switches and control flow from the network to the SDN both contribute to network traffic in the context of SDN.

2.7.3 Performance Metrics

Performance metrics include wide range of metrics to evaluate the performance of an SDN-based platform networks[112][113].

- **Latency(sec):**Elapsed Time (MSG_{Packet_in} , $MSG_{FlowMod}$)……(2.4)
  Where MSG_{Packet_in} is a packet in message, while $MSG_{FlowMod}$ is a Flow mod message.
- **Throughput(packets/sec):** $\sum MSG_{FlowMod}/T$ ……...(2.5)
  Where $MSG_{FlowMod}$ is a Flow mod message ,while T is monitoring time.
- **Dropped Packets** $=N_{PS} - N_{PR}$……(2.6)
  Where $\mathbf{N_{PS}}$ is the number of the packets sent ,while $\mathbf{N_{PR}}$ is the number of the packets received.
- **Bandwidth**= Received $_{Bytes}$+ Transmit $_{Bytes}$ ……..….(2.7)
  Where Received $_{Bytes}$ is the transmitted byte, while Transmit $_{Bytes}$ is the received bytes during a predefined timing interval.
- **Write time overhead=** $T_{TR-}T_{TS}$ ……..…(2.8)

  Where the Write time overhead metric measures how long it takes for an application to get a response when it requests a change to the SDN controller's resources. where T_{TS} is an application's request being sent to an SDN controller , while T_{TR} controller's subsequent is a response to the application.

Chapter Three
The Proposed Blockchain based DSDN

3.1 Overview

Traditional SDN's Network architecture is based on a single controller. As a result, network performance becomes heavily dependent on the effectiveness of the one controller in the Control Plane, which is unacceptable for any application that requires security, scalability and reliability. Despite the fact that SDN has numerous benefits, its practical use is limited due to the system's unsatisfactory dependability and fault-tolerance capabilities.

Distributed SDN architecture has been suggested as a solution to these SDN challenges. A straightforward distributed Control Plane with several controllers makes up the suggested design. In addition, Blockchain technology was proposed as a consensus mechanism with DSDN, where consistency is an essential factor in DSDN to realize scalability, reliability, and high availability .All of these topics and more will be covered in this chapter.

3.2 The Proposed System

This Chapter introduces the specific details of the proposed system, which works firstly on a fully distributed SDN network to overcome the single point of failure, enhance scalability, and provide CPU consumers that are related to the leader controller, break free of the leader-follower structure common to consistency algorithms like the raft algorithm, in which the leader controller is responsible for any update to the state of the network's data in order to maintain strong consistency between the system's controllers, resulting in the leader control using more resources than the rest and adding additional time to the estimation process. Where an Opendaylight-based distributed system is used, each controller is responsible for its own domain, distributing the load evenly across the controllers in the network. while secondly providing a consensus mechanism that mainly includes Blockchain-enabled flow rules. This mechanism allows the system manager to enforce new configuration flow rules on all DSDN controllers at the same time, ensuring the consistency of the data state of the network among the DSDN controllers in the proposed system. Figure 3.1 shows the flowchart of the proposed system.

The Proposed System Phases:

- **Planning & Requirements Gathers phase** In this phase, gathering all the materials we'll need to roll out the network upgrades. The criteria should be used as a guide to ensure that the planning is in line with the parameters of scope and resource.
- **DSDN Network Design Phase** The design specification is the starting point for the development of proposed system, A complete and well-detailed design that satisfies technical criteria is what the network design specification provides. It includes

manageability, scalability, consistency, availability, and manageability criteria.

- **Implementation of The Proposed System** in this phase DSDN-Based Platform was built with a custom Topology using a GNS3 ,mininet ,Opendaylight as SDN controller, Perform Several Tests to Evaluate the Performance of DSDN network(SPOF, Scalability, Topology discover , resource consumer)where a wireshark software used to capture a real packet between the control plane and data plane to analysis the data ,and build/ integration a blockchain network with the DSDN network to propose a blockchain based DSDN controller flow rules as a consensus mechanism.
- **Security Discussion of The Proposed System** the confidentiality -integrity-availability (CIA) triangle was the basis for our security evaluation of the proposed system, where utilizes it to gauge how well our system secure.

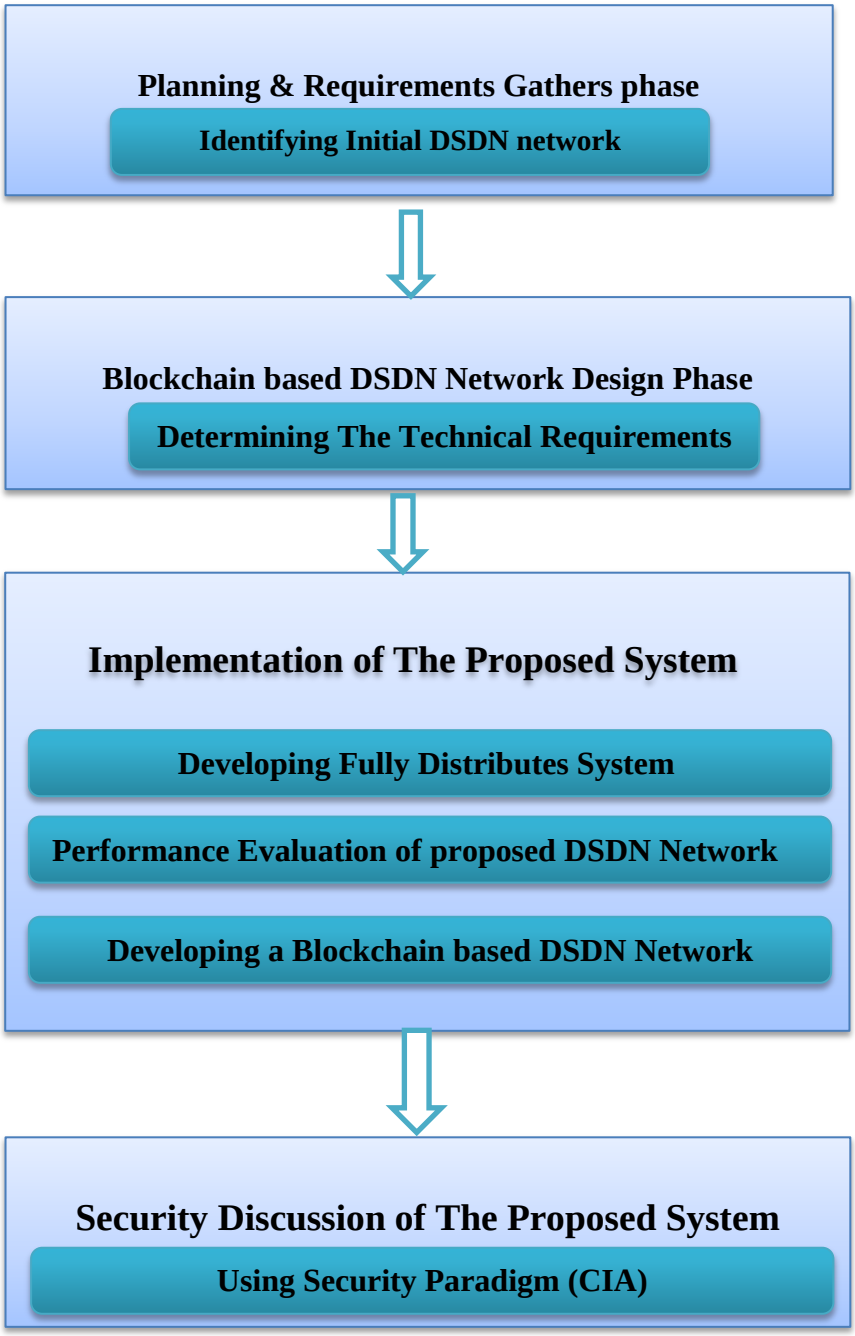


Figure 3.1: The Flowchart of Proposed System

3.2.1 Planning & Requirements Gathers phase

The first and most difficult step in constructing a DSDN network is deciding on an appropriate environment, as this necessitates working with a controller that supports the DSDN network and is a real control rather than a simulation, as well as the environment being conducive to building a blockchain network and offering the opportunity to integrate it with the DSDN network. Where in the proposed system numerous controllers that can share the burden on the network, and one controller can take over another controller when it breaks .The chosen environment must all on simulated a data plane that controlled remotely by SDN controllers .After search on the platform that provide all the requirement that mention above ,the GNS3 was the best chosen for implement the proposed DSDN system where it provide all the requirement that needed. The GNS3 enable installation and configuration of VMware workstation as a virtual machine that require for each one the SDN controller in proposed system, and for mininet which represent a data plane. For blockchain network built and integration with DSDN the high level programming language was needed as python that supported by GNS3 with Flask platform that necessary to build the nodes of blockchain as web application and configuration the rest API between the blockchain and the DSDN controllers. To capture a packet that send among SDN controllers plane and the data plane that it necessary to evaluate the performance of the proposed system , the GNS3 support the wireshark that provide this ability .

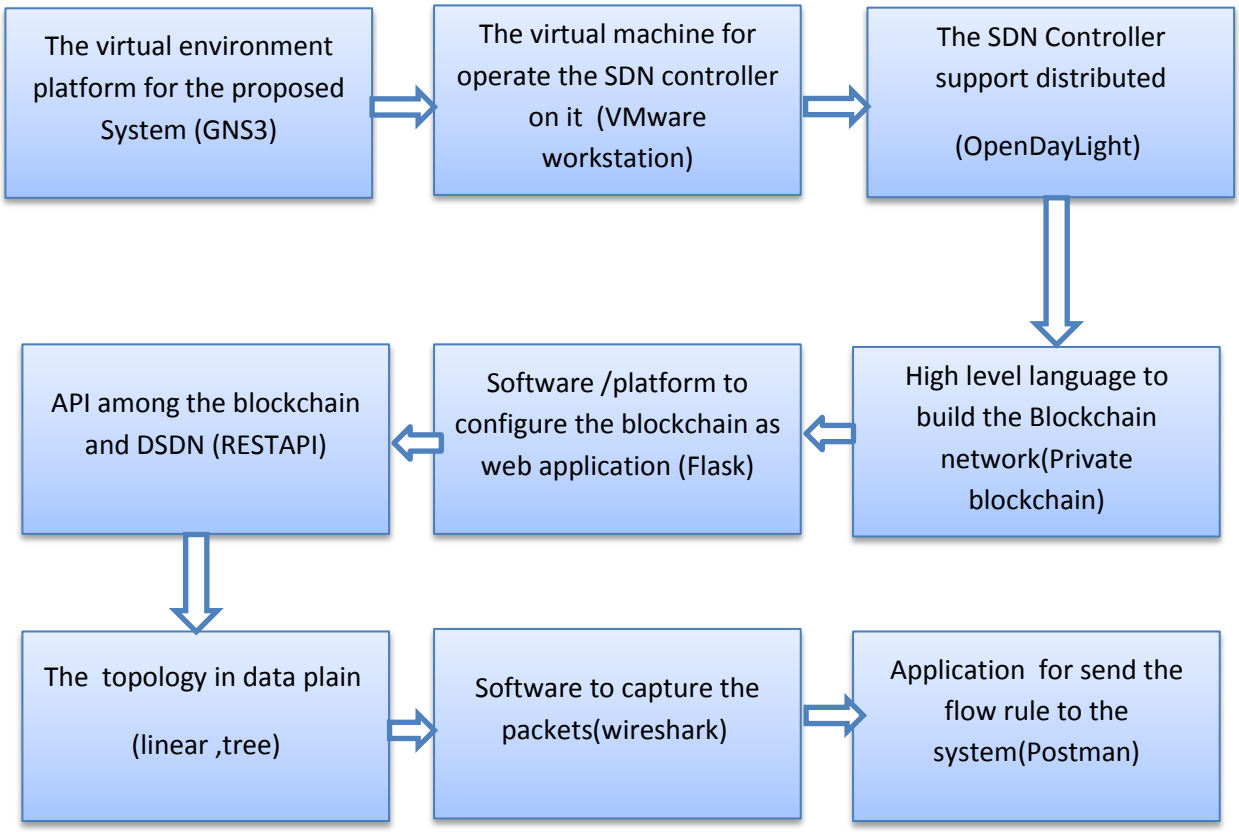


Figure 3.2: The Flowchart of Environment Requirements of the proposed System

3.2.2 Blockchain based DSDN Network Design

In this phase the design of the proposed system shows in Figure (3.3) was employed to meets a technical requirements. It includes specifications to maintenance availability, reliability, security, scalability, consistency ,and manageability.

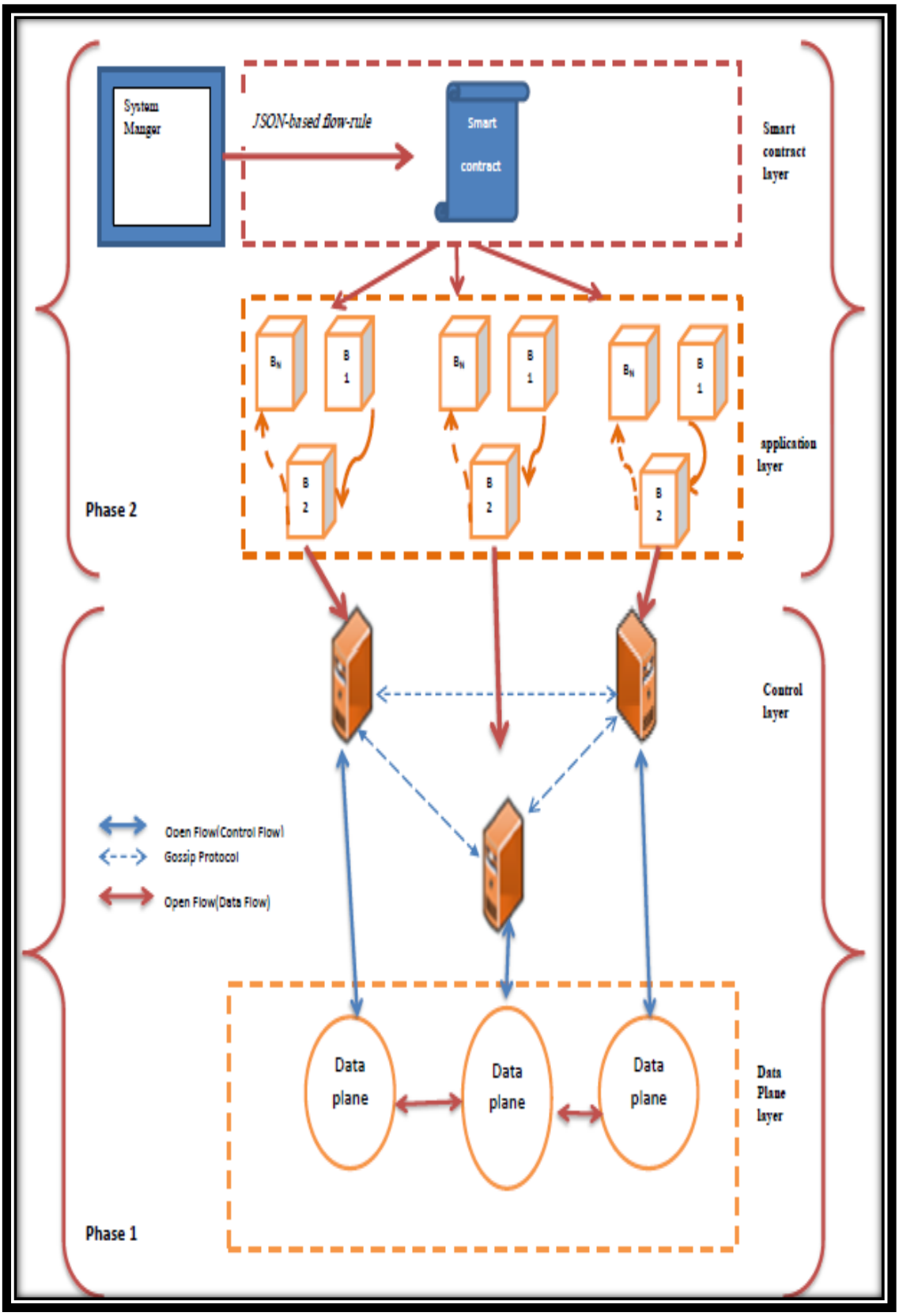


Figure 3.3: The Design of Proposed System

3.2.2.1 A Fully Distributed SDN Network

One of most issues of SDN controller that support distributed system that explain in chapter two ,is the uneven CPU consumer of the leader controller in the DSDN network .To avoid this issue the fully distributed SDN Network was proposed where each controller control its domain ; number of OVS switches and hosts ,witch connect to its controller remotely through the control path ,the southbound API between the controller and its domain was a OpenFlow 1.3 .The custom topology was built in mininet that divided the infrastructure of the network to domains each one control and connect remotely to one controller ,when one of the controller in the system was fail to manage its domain for any reason software/hardware the nearest controller work on mange its domain. The design of the fully DSDN proposed in Figure (3.4).

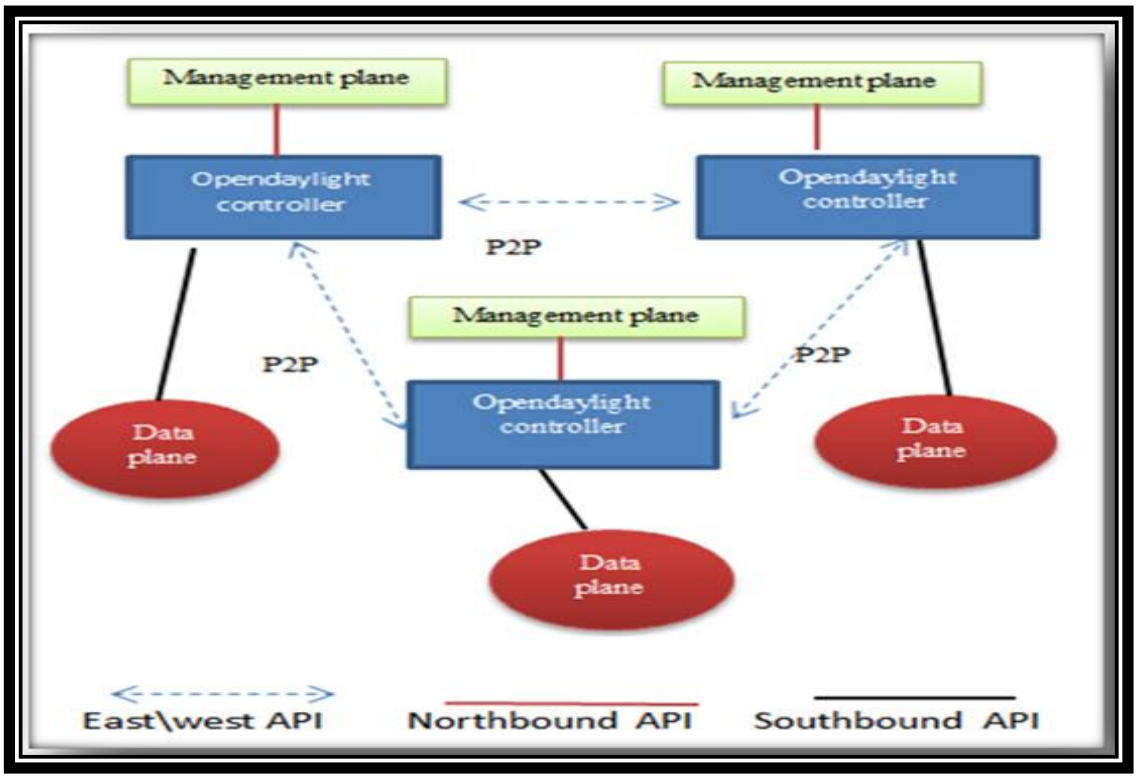


Figure 3.4: The Design of Distributed SDN Networks

3.2.2.2 The Developing SDN Controller Specification

In order to design the DSDN network that maintains the global view of the network and take part of the consistency of the data state among the SDN controller some developing on SDN Controller specification was propose as shows in Figure(3.5). The developed done on the machine replication technique in SDN controller.

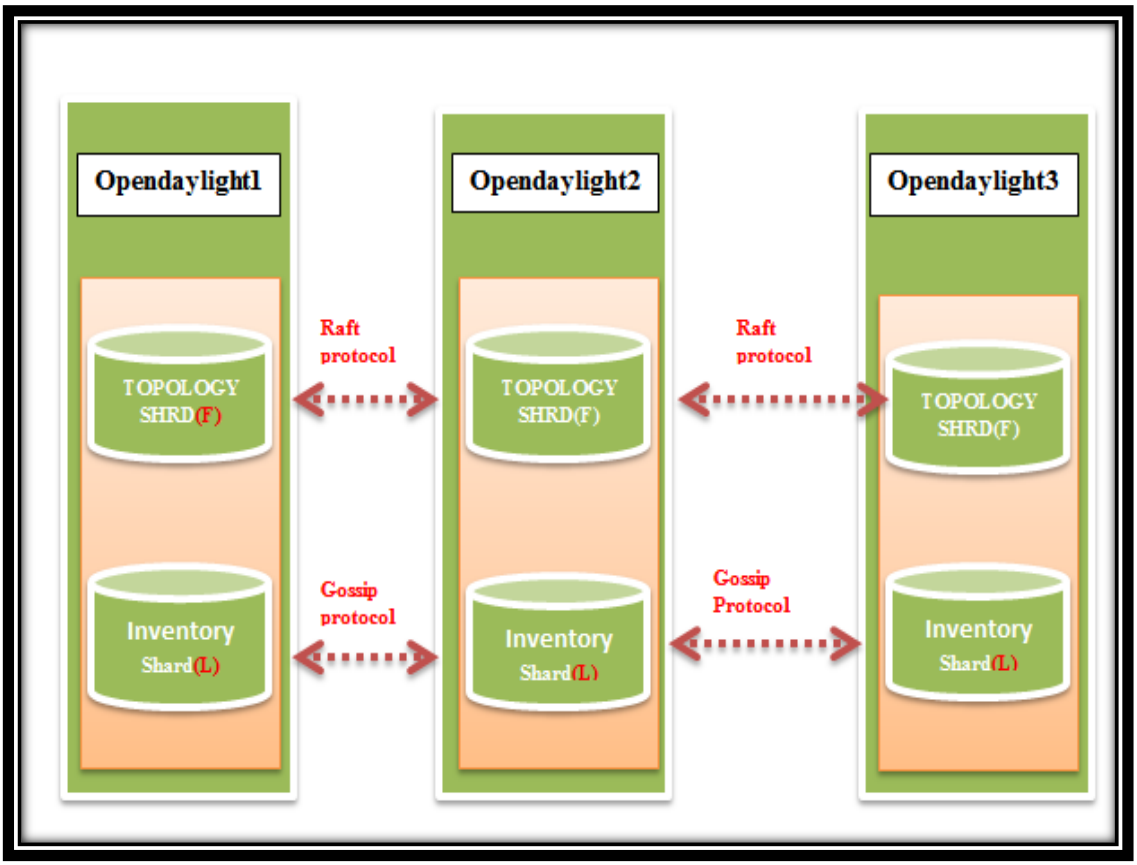


Figure 3.5 :Configure the Replicated of Network Data Base

As explained in Chapter two the opendaylight controller depends on machine replication algorithm to achieve consistency and a synchronous in DSDN, in order to do this some changes must be done in configuration file of opendaylight controller , where a portion or the entirety of an OpenDaylight MD-SAL data store is stored as data shards. For instance, one shard may have all the inventory data, while another

would hold all the topological data. The network configured to replicated only the topology shards as explain in the Figure (3.4) and the steps below :

Step1: Open the opendaylight distribution file in machine.

Step 2: Open the following .conf file:

configuration/initial/module-shards.conf .

Step 3: assign the shard name to topology :

(name = "topology")

Step4: update the replicas so that each shard is replicated to all three nodes.

```
replicas = [
                    "member-1",
                     "member-2",
                      "member-3"
          ]
```

Where ,for topology shard the machine replication stay work in concept leader – follower for RAFT algorithm where this shard was update each 5 second. As it explain in Figure (3.5).

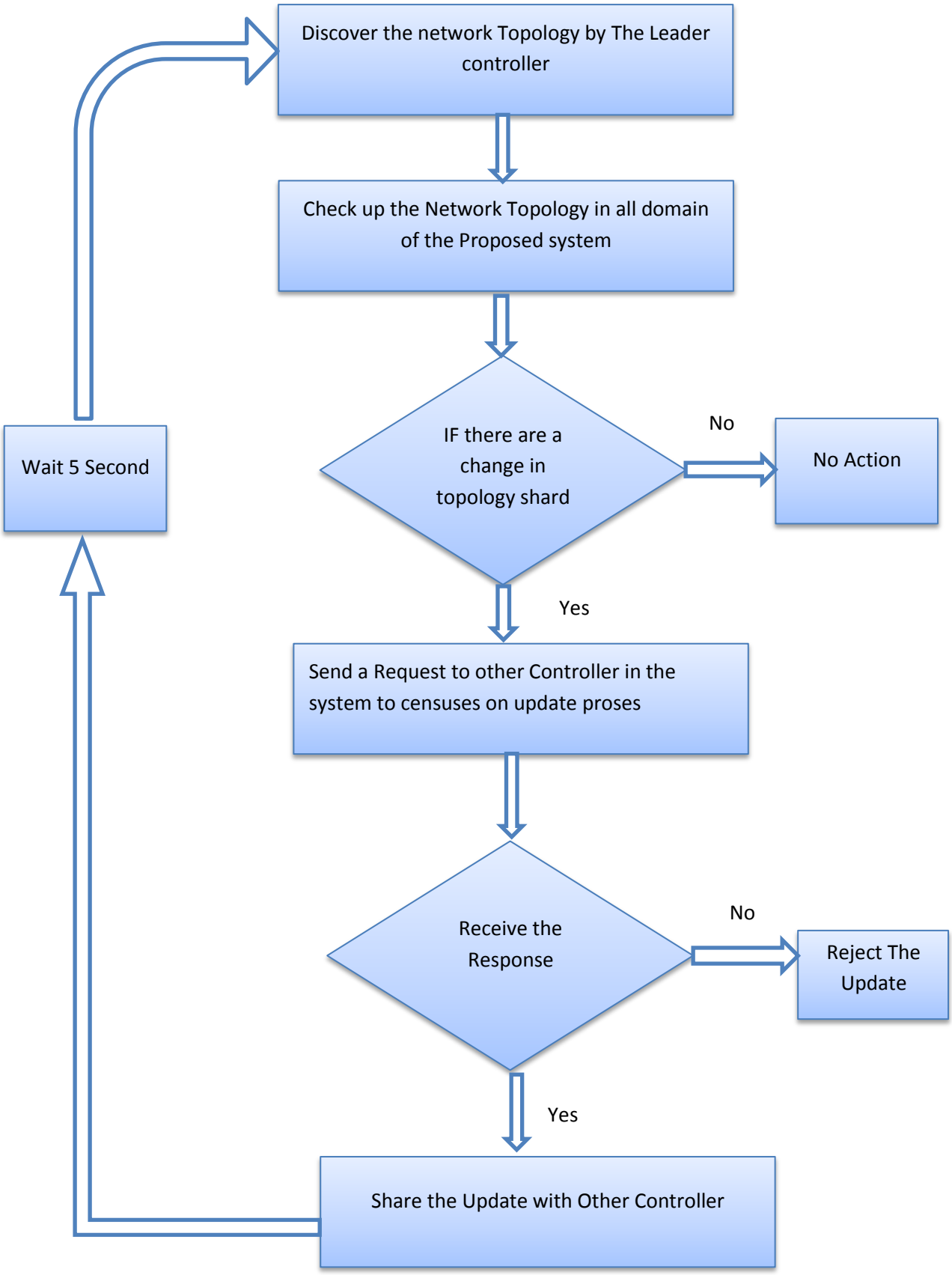


Figure 3.6 :Configure the Topology shards Replicated of DSDN

While ,for Inventory shard the machine replication In a network work according to a gossip protocol, all of the nodes regularly share information with a smaller group of other nodes. In most cases, this will be the collection of nodes that surround each node. Every node in the network can only see what's happening in its domain. After a certain number of periodic updates, all nodes will have the same global data state regarding the Inventory shard; flow rule.as it explain in Figure (3.7).

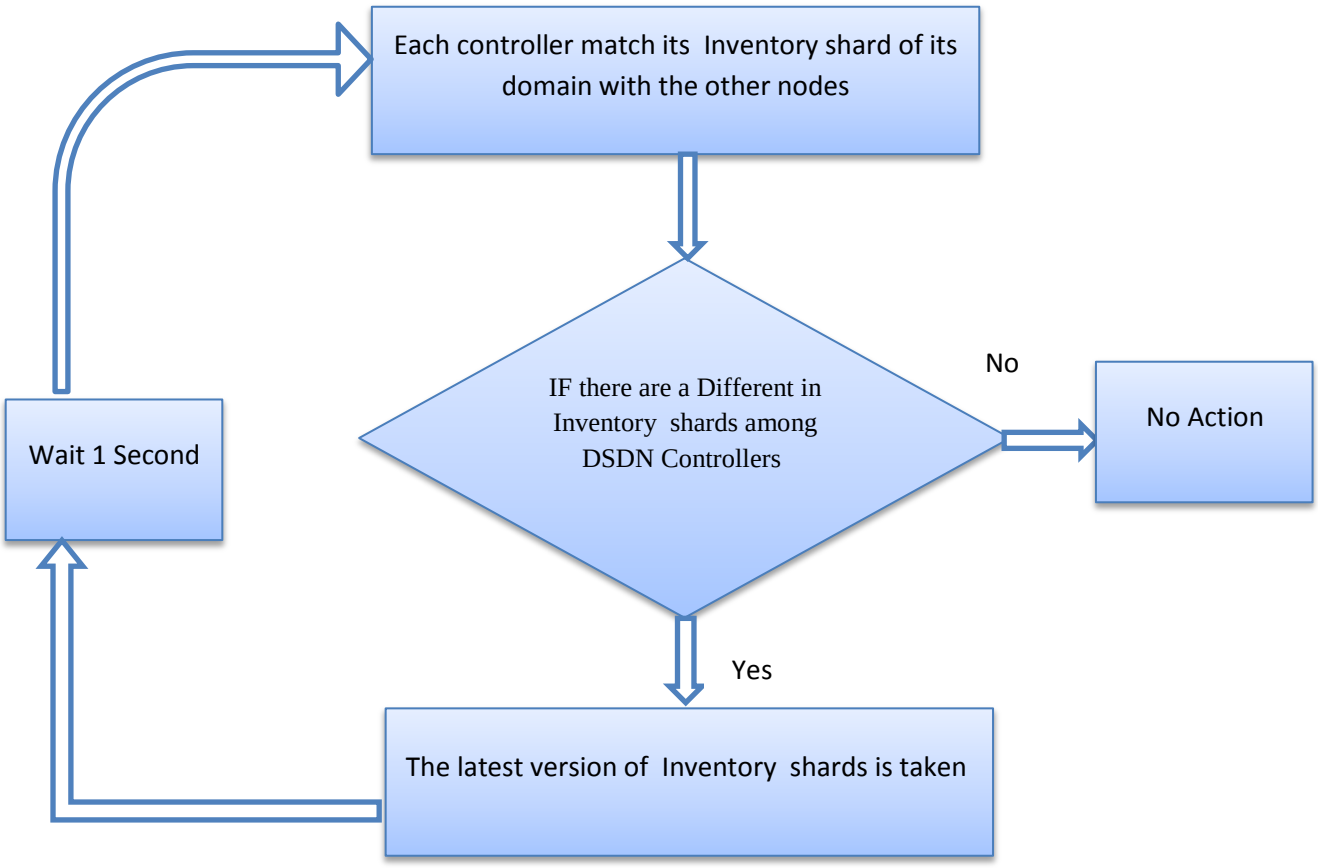


Figure 3.7 :Configure the Inventory shards Replicated of DSDN

3.2.2.3 A Consensus Mechanism with DSDN controllers

In figure (3.8) the Consensus Mechanism with DSDN controllers was shown, where Propose a new consensus mechanism depending on blockchain technology enabled new Configuration flow rules installation in DSDN controller at the same time ,which help in remotely mange of the all DSDN controllers ,a save on the consistency in the proposed system.

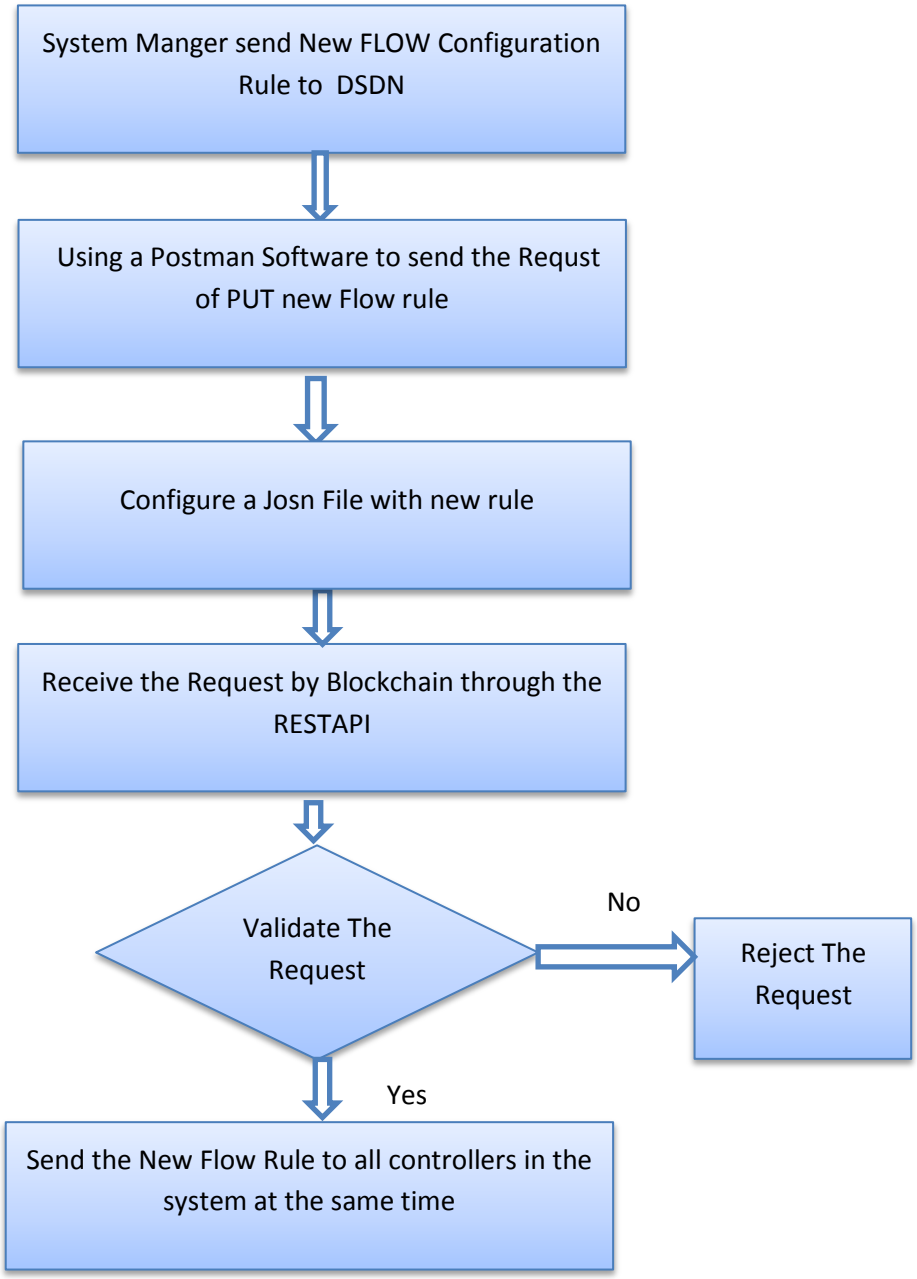


Figure 3.8 :The Design of a Consensus Mechanism based blockchain with DSDN controllers

3.3 Implementation of The Proposed System

In this phase, a DSDN-based platform was built with a custom topology using GNS3, mininet, and Opendaylight as the SDN controller; tests were run to assess the network's performance in areas like scalability, topology discovery, and resource consumption; and a blockchain was integrated into the system to propose a DSDN controller that operates on the blockchain.

3.3.1 Distributed Software Define Network (DSDN)

In this section, the method of structuring and building a DSDN will be delved into according to the proposed work, The Distributed Software Define Network (DSDN) is building, with remote SDN controllers managing and monitoring the entire network. The SDN's network is divided into three domains, each of which is managed by one of the controllers, as explained in Figure (3.4). The Algorithm(3.1) and flowchart in Figure(3.9) are shown the steps of building of (DSDN). A test for DSDN is proposed ,the SDN's network be fully distributed where each controller is able to managing its domain and in the same time when one of these controller fail the other active controllers mange its domain .

Algorithm 3.1: Building a Distributed Software Define Network:

Input:	- Remote SDN controller -Emulated network topology -Four Server -VMware (server1,server2, server3, server4) -Topo= custom topology
Output :	Network will manage and monitor by Distributed SDN Controller
1-	**Begin**
2-	**While** ((i=1) <= no.of Controllers in DSDN)
3-	Initialize (Server **(i)** ,1'st VMware)
4-	Indicate (Server(i),IP Address)
5-	Initialize (Server(i),SDN controller)
6-	Initialize (Server(i),SDN controller)
7-	Start (Server(i), Remote SDN controller)
8-	Enable distributed(Remote SDN controller)
9-	Setup required feature (Remote SDN controller)
10-	**End Loop**
11-	**While** ((i=1) <= no.of Controllers in DSDN)
12-	**if** (Remote SDN controllers _ IP Address $\in$ Distributed Software Define Network)**Then** Configuration (Server(i), Remote SDN controller)
13-	**End If**
14-	**End Loop**
15-	Initialize (server4, Emulated network topology)
16-	Loop ((i=1) <= no.of Controllers in DSDN)
17-	Start (server(i), Remote SDN controller)
18-	End
19-	Start (server 4, Emulated network topology))
20-	Start (Emulated network topology ,Topo (connect (Remote SDN controllers _ IP Address , Ports)))
21-	**END.**

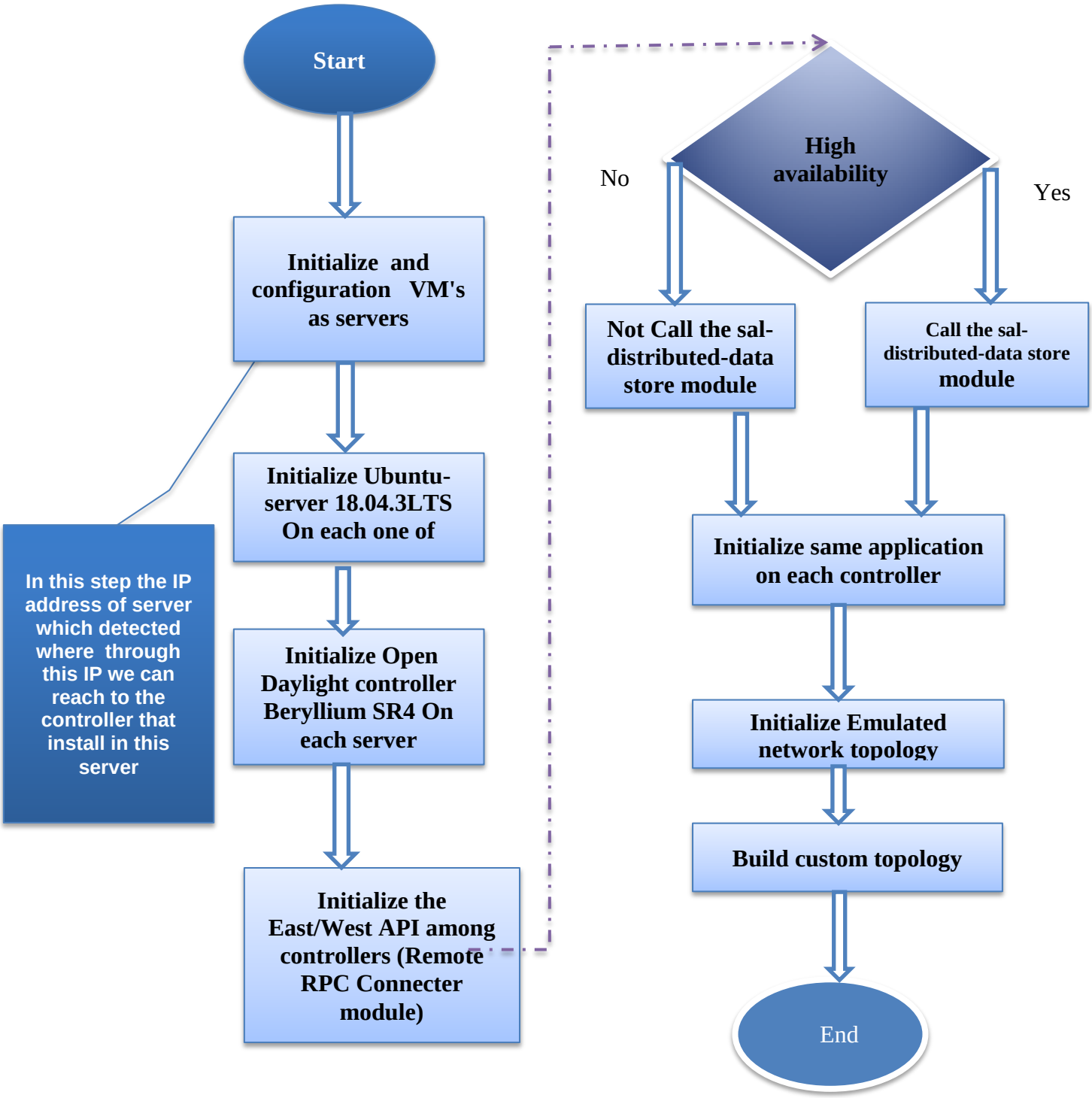


Figure 3.9: The Flowchart of Distributed SDN Networks Configuration

Four virtual machines is used to implements the Distributed SDN's network ,three of them used to remote SDN's controllers(control plane) that work on controlled the custom topology of the network (data plane)which works on the fourth virtual machine. The virtual machines must have connectivity to each other and to the system hosting them, there is also a need for Internet connectivity. To achieve this, all virtual machines will have configured as "host-only network" and link to as explained in Figure (3.10).

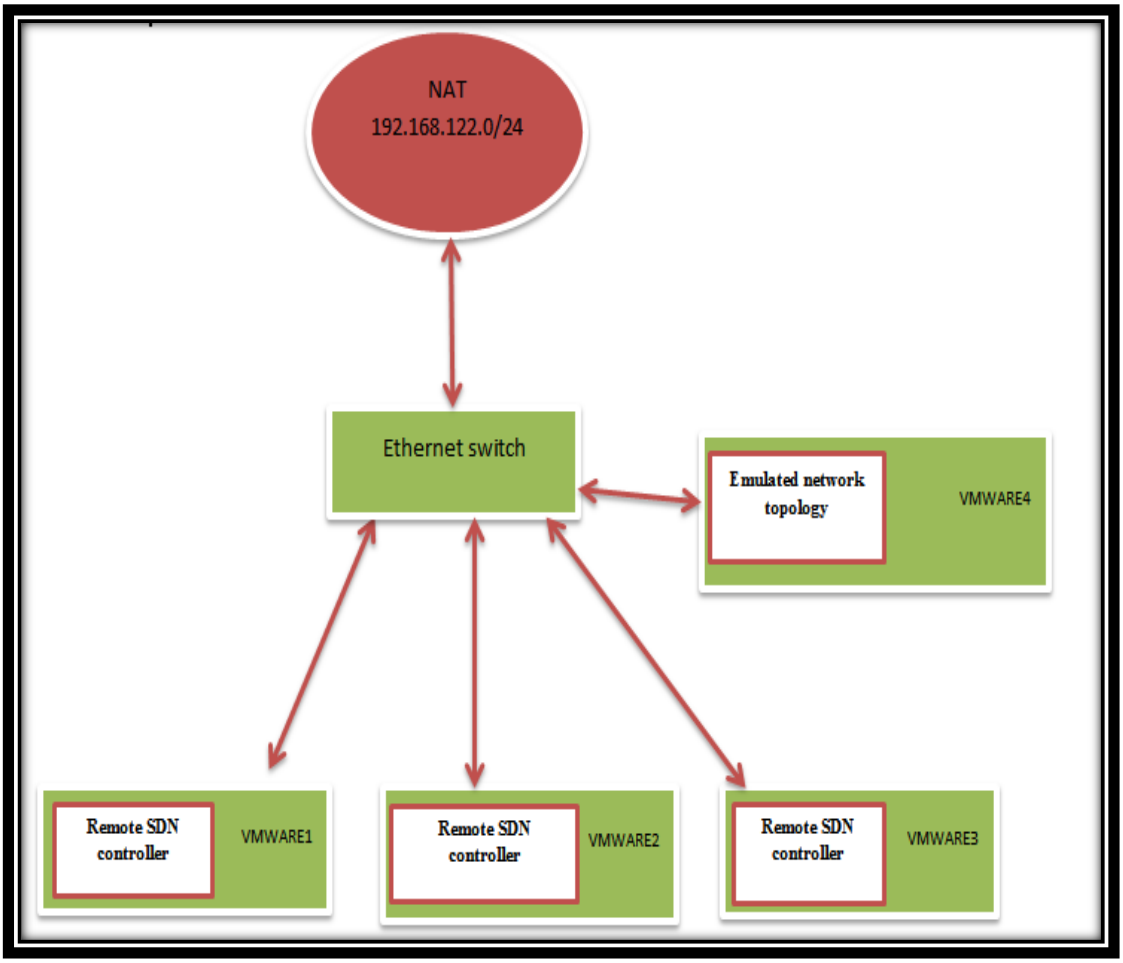


Figure 3.10: Virtual Machines Connectivity of Distributed SDN Networks

To start the East-West API among distributed SDN controllers in the proposed system where the Opendaylight is used as a remote SDN controller, so must change some of the configuration files in each controller as explained in the following steps:

Step1: Open the Opendaylight distribution file that resides in virtual machine.

Step 2: Open the configuration file the Opendaylight distribution

Step 3: Assign the hostname to the IP address of the machine on which this file resides and OpenDaylight will run.

Step4: Assign the seed-nodes to the IP address of any of the machines that will be part of the distributed system.

Step 5: Identify the identity of each controller in DSDN as member-i. the 1st controller allocated with the member-1 identity , the 2nd controller with the member-2 role, and the 3rd controller with the member-3 identity .

3.3.2 Performance Evaluation of proposed DSDN Network

In this section Several Tests to Evaluate the Performance of DSDN network(SPOF, Scalability, Topology discover , resource consumer) was Perform ,where a wireshark software used to capture a real packet between the control plane and data plane to analysis the data**.**

3.3.2.1 Security Issues Evaluation DSDN Network compared to Traditional SDN

The security issues of Distributed SDN ; Single Point of Failure (SPOF) and Fault Tolerance, are examining assuming that the one of controller .It is attacked, making it unable to connect to the domain controlled by, lost the control path, as it shows in Figure (3.11).

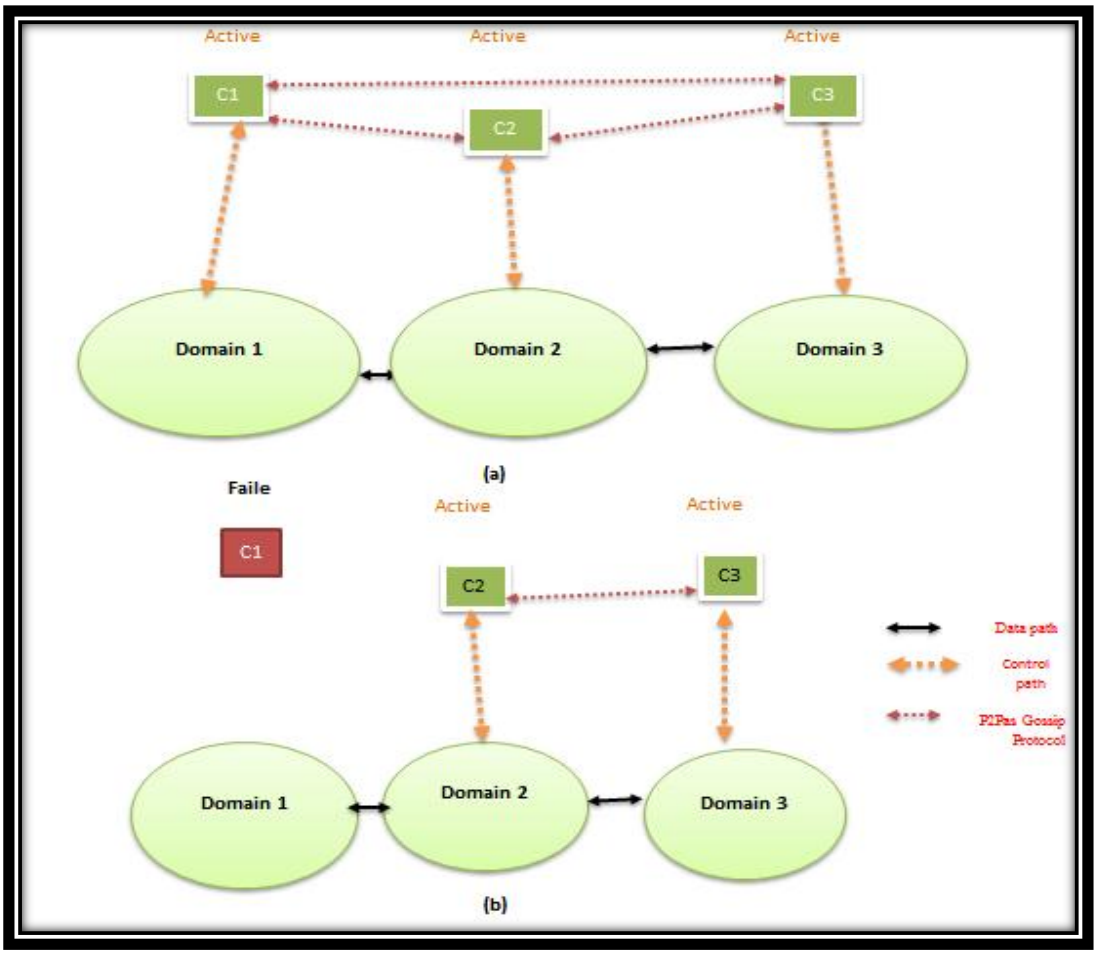


Figure 3.11: Comparison Security Issues of SDN Against DSDN

In the Figure (3.11) (a) where all the controller is active and each of the controllers mange its domain ,in our configuration for DSDN the topology shard is update(Read/Write) by the leader controller on this shard only, while the inventory shard is update locally on each domain , and synchronous with other domain every second with other domain through Gossip Protocol.

In Figure (3.11)(b) assuming that controller 1 is attacked, making it unable to connect to the network domain controlled by, lost the control path. The packet loss used as metric to evaluation the controller fail case with steps that explain in Algorithm(3.2) a and Algorithm(3.2)b. where in Algorithm(3.2)b assuming that two of controllers are attacked.

Algorithm (3.2)a: Packet Lose Rate to Evaluation the security issues of Distributed Software Define Network:

Input:	-Remote SDN Controller -Emulated network topology) -VMware (server1,server2, server3, server4) -Topo= custom topology
Output :	Evaluation Result of Packet Lose Rate(**PLR**)
1-	**Begin**
2-	Start (server1, Remote SDN controller)
3-	Start (server2, Remote SDN controller)
4-	Start (server3, Remote SDN controller)
5--	Start (server 2,Emulated network topology)
6-	Start (Emulated network topology ,Topo)
7-	Connect (Topo, Remote SDN controllers (IPs, PORTs))
8-	Check connect (Topo(nodes), Ping All Toole)
9-	Compute (PLR, Remote SDN controller (ACTIVE))
10-	Logout(Remote SDN controller (Fail)) //Controller 1
11-	Delete (Topo(nodes),flow table) // Domain 1
12-	Check connect (Topo(nodes),Ping All Toole)
13-	Compute (PLR, Remote SDN controller (Fail))
14-	**END.**

Algorithm (3.2)b Packet Lose Rate to Evaluation the security issues of Distributed Software Define Network

Input:	-Remote SDN Controller -Emulated network topology) -VMware (server1,server2, server3, server4) -Topo= custom topology
Output :	Evaluation Result of Packet Lose Rate(**PLR**)
1-	**Begin**
2-	Start (server1, Remote SDN Controller)
3-	Start (server2, Remote SDN Controller)
4-	Start (server3, Remote SDN Controller)
5--	Start (server 4, Emulated network topology)
6-	Start (Emulated network topology ,Topo)
7-	Connect (Topo, Remote SDN controllers (IPs, PORTs))
8-	Check connect (Topo(nodes),Ping All Toole)
9-	Compute (PLR, Remote SDN Controller (ACTIVE))

10-	Logout(Remote SDN Controller (Fail)) //Controller 1 & Controller 2
11-	Delete (Topo(nodes),flow table) // Domain 1 & Domain 2
12-	Check connect (Topo(nodes),ping all Toole)
13-	Compute (PLR, Remote SDN Controller (Fail))
14-	**END.**

3.3.2.2 : Scalability Issues Evaluation DSDN Network compared to Traditional SDN

The Scalability of Distributed SDN , the throughput and latency are used as metrics to evaluate the Distributed SDN controller scalability case. The rate of flow response (flow rule) is determined by the throughput of the controller system, and in a similar way, the latency of the controller is governed by the amount of time taken by the controller to reply to a flow request. When there are too many flow requests coming in from switches for the single controller to handle all at once, the performance of scalability with a single controller is significantly hampered. The throughput and latency are used as metrics to evaluate the remote SDN's controller scalability case with steps that are explained in algorithms (3.3) and (3.4) .

The *ping all Tool* is starting connection each host in data plane with others hosts to check the connection between the data plane and remote control this from side(control path), while in the other side is check the connection among hosts in data plane(data path). So at the start of ping all tools when (h_i)**ping** (h_j),the host (h_i) are not know the MAC address of host(h_j) so it sends the ARP request to OVS switch that link with to find the IP address of the host that tries ping it, here the switch is capsulation the request and is sent to the remote SDN's controller as packet_in message. The Address resolution Protocol (ARP) handler module in remote SDN's controller process the request packet_out message send to all port of the switch that send the packet_in

message.at the end of this process the host get the IP address and MAC address but there is no flow entry in OVS switch so packet_in with ICMP protocol send to the remote SDN's controller to the rule that learn the switch how (h_i)**ping** (h_j) the response be as FLOW_mode message. all this work is done by module that activated previously in remote SDN's controller , odl-l2switch-switch and as explain in algorithm(3.5).

Algorithm 3.3: Throughput Evaluation to the Scalability of Distributed Software Define Network

Input:
-Remote SDN controller
-Emulated network topology
-VMware (server1,server2, server3, server4)
-Topo= Custom Topology
-SW={S_1,S_2,…..S_M} //SW is list of OVS switches in Topo
-Time of monitoring is T
- Set M // M is number of OVS switchs in Topo
- Set N // N is Maximum number of OVS switchs can be Exam
-Set //K is The amount of increase in the OVS switchs

Output : Evaluation Result of Throughput

```
1-  Begin
2-  Setup(Remote SDN controller, reactive mode)
3-  Start (server1, Remote SDN controller)
4-  Start (server2, Remote SDN controller)
5-  Start (server3, Remote SDN controller)
6-  Start (server4,Emulated network topology )
7-
8-  While (M<N)do
9-      Repeat
10-       Start (Emulated network topology ,Topo(M )
11-       Connect (Topo, Remote SDN controller (IP, PORT ))
12-       Check connect (Topo ,ping all Toole )
13-       Start packet capture (wireshark tool, control path)
14-       Compute the Throughput (NO.OF Packet_Out(Flow-Mode),T)
15-   Until (experiment number)
16-   M=M+K
17-
18- End Loop
```

19-	**End.**

Algorithm 3.4: Latency Evaluation to the Scalability issues of Distributed Software Define Network:

Input:	-Remote SDN Controller -Emulated network topology -VMware (server1,server2, server3, server4) -Topo= Custom Topology -SW={S_1,S_2,…..S_M} //SW is list of OVS switches in Topo -Time of monitoring is T - Set M // M is number of OVS switchs in Topo - Set N // N is Maximum number of OVS switchs can be Exam -Set //K is The amount of increase in the OVS switchs
Output :	Evaluation Result of Latency
1-	**Begin**
2-	Setup(Remote SDN controller, reactive mode)
3-	Start (server1, Remote SDN controller)
4-	Start (server2, Remote SDN controller)
5-	Start (server3, Remote SDN controller)
6-	Start (server4,Emulated network topology)
7-	
8-	**While (M<N)do**
9-	**Repeat**
10-	Start (Emulated network topology ,Topo(M)
11-	Connect (Topo, Remote SDN Controllers (IP, PORT))
12-	Check connect (Topo ,Ping All Toole)
13-	Start packet capture (wireshark tool, control path)
14-	Check connect(Topo, Ping(h_1,h_M))
15-	Compute the Latency (Diff(Packet _In (ICMP),Packet _Out(Flow-Mode))
16-	**Until (experiment number)**
17-	M=M+K
18-	Average. Latency[(SUM(Latency[**M])/ experiment number**]
19-	**End While**
20-	**End.**

Algorithm 3.5: Starting Of Ping All tool of DSDN

Input: Topo= custom topology
SW={S_1,S_2,….,S_N} //SW is list of OVS switches in Topo
H={h_1,h_2,h_3,….,h_M} //H is list of hosts in Topo
L={l(h_1),l(h_2),……,l(h_N)} //L is list of function that return a S_j that h_i link with.
C={c(S1),c(S2),……,c(SN)} //C is list of function that return a controller that Si link with.

Output : Flow Table for each S_i in SW.

```
1-   Begin
2-   Setup(Remote SDN controller, reactive mode)
3-   Start (server1, Remote SDN controller )
4-   Start (server2, Remote SDN controller )
5-   Start (server3, Remote SDN controller )
6-   Start (server 2, Emulated network topology)
7-   Start (Emulated network topology ,Topo )
8-   Connect (Topo, Remote SDN controller (IP, PORT ))
9-   Check  Connect (Topo(H),ping all Toole )
10-    For  (i=1,M)
11-     For (j=1,M)
12-      Ping(hi,hj)      // i≠j
13-         Send (ARP request,  l(hi))
14-         Packet_in=  Encapsulation (ARP request)          // IN Switch
15-         Send to(packet_in , c(l(hi))                     // IN Switch
16-            Packet_out=  Process(packet_in )              // IN Controller
17-            Send (Packet_out,  l(hi))                     //IN Controller
18-         Send(Packet_out,l(hi)(Port))                     // IN Switch
19-         Send (ICMP request,  l(hi))
20-         Packet_in=  Encapsulation (ICMP request)         // IN Switch
21-         Send to(packet_in, c(l(hi)))
22-            Packet_out(Flow_mode)=Process(packet_in )     //IN Controller
23-            Send (Packet_out(Flow_mode),  l(hi))          //IN Controller
24-     End For
25-   End For
26-  END.
```

Other metrics are examined such as topology discover, delay and the bandwidth of DSDN and they are compared with central SDN:

1 -Topology Discover

Starting the custom topology with custom no.of switch and using the wireshark software to capture real time traffic between the data plane and control plane (DSDN) , the average of LLDP cycle computing through recording the sending time of the first LLDP, the Link Layer Discovery Protocol where each OVSswitch had send a PacketIN encapsulated LLDP message back to the controller. These PacketINs are answering the LLDP messages the controller has send to discover the network, for each cycle, the time of the LLDP cycle calculated which is the difference between the sending time of the first LLDP message sent by the controller and the reception time of the last LLDP message where this cycle happened every 5sec in opendaylight controller, as it explain in algorithm (3.6) .

Algorithm 3.6: Topology Discover Time Evaluation of Distributed Software Define Network

Input:	-Remote SDN controller -Emulated network topology) -VMware (server1,server2, server3, server4) -Topo= Custom Topology -SW={S_1,S_2,…..S_M} //SW is list of OVS switches in Topo -Time of monitoring is T - Set M // M is number of OVS switchs in Topo - Set N // N is Maximum number of OVS switchs can be Exam -Set //K is The amount of increase in the OVS switchs
Output :	Evaluation Result of Topology Discover Time
1-	**Begin**
2-	Setup(Remote SDN controller, reactive mode)
3-	Start (server1, Remote SDN controller)
4-	Start (server2, Remote SDN controller)
5-	Start (server3, Remote SDN controller)
6-	Start (server4, Emulated network topology)
7-	**While (M<N)do**
8-	**Repeat**

9-	Start (Emulated network topology ,Topo(M)
10-	Connect (Topo, Remote SDN Controller (IP, PORT))
11-	Check connect (Topo ,ping all Toole)
12-	Start packet capture (wireshark tool, control path)
13-	Compute the $LLDP_T$ ($Packet\_IN(LLDP)_L$ – F_ $Packet\_IN(LLDP)_F$)
14-	**Until (5 experiment)**
15-	M=M+K
16-	Average . $LLDP_T$ [(SUM($LLDP_T$ [**experiment**])/**experiment number**]
17-	End
18-	**End.**

3.3.3 Developing a Blockchain based DSDN Network

The system manger can manage the controllers in SDN network remotely , the blockchain technology as consensus mechanism is proposed with DSDN networks to increase and guarantee the security and consistency of the control layer and the flow rules in the data layer flow table. Building the private blockchain between the system manger and the controllers in DSDN , the Blockchain-enabled flow rules with control layer work on distributed ledger to keep tracks of all the switches working correctly and maintaining the same flow rules.

This section explains the techniques/method that used to build the blockchain network and smart contract and deploy it in the Distributed SDN, which includes the below steps Figure (3.12):

Step 1: Build a consensus plane of DSDN's Controllers.

Step 2: Represent the new contribution made in this work, build a Smart Contracts that receive the flow rule that required installing in system as transaction from the system manger. It is the only can create or mine the new block

Step 3: Build the REST API's to communication among the node ,application.

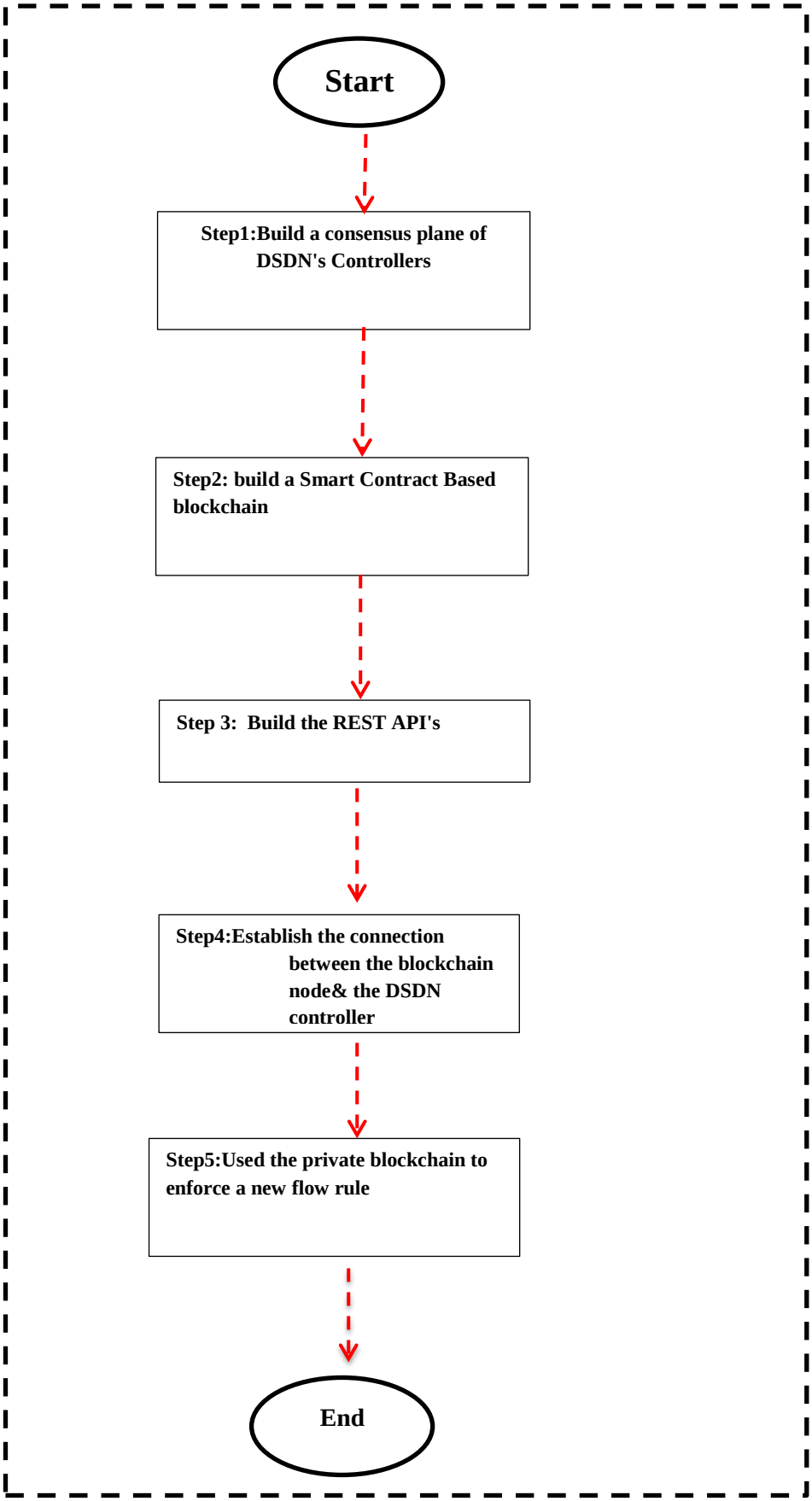


Figure 3.12: Second Phase of Proposed System

Step4: Establish the connection between the blockchain node(application),and the DSDN controller through northbound API.

Step 5: As an instance of using blockchain technology; used a private blockchain to enforce a new flow rule on the DSDN controller.

3.3.3.1 The Consensus Plane of DSDN's Controllers

In this step building of the consensus plane of DSDN's Controllers is proposed using an applications as a node of blockchain and in the same time as an application of SDN controller. Each one reside in the server that operates the opendaylight controller of DSDN and has an ability to communicate with through the northbound API.

Following is the state of creating a private blockchain for a particular party. The main provision in transferring data to happen in the environment for transmitters (system manger) and receptors (SDN controllers) that private blockchain are created via three nodes.as Figure (3.1) shows the ingredients of the blockchain architecture can be explained in the following point:.

a) Nodes: application that communicate with SDN controller

b) Transactions: creating the blocks of a blockchain, update/enforce flow rules in DSDN controllers .

c) Block: the data construction appropriated for preserving transaction(flow rules) spread to all nodes within the created networks.

d)BlockChain: a series of blocks in a particular arrangement .

f)Smart Contract: Unique smartly established contract to communicate with data.

In the proposed system the consensus plane is works on guarantee the consistency inventory shards in DSDN's controllers and gets the same update at time as it is explained in Algorithm (3.7). when the consensus plane receive the updating blockchain ;updating by the smart contract adding new block , it works on Extract the last block and sends it to all controller in DSDN system at the same time.

Algorithm 3.7: Inventory Shard Consensus of DSDN's Controllers

Input:	Update Blockchain ledger //Update by Smart Contract
Output :	DSDN's Inventory Shard Updating
1-	**Begin**
2-	**IF** (new (blockchain. Index) > previous(blockchain. index) **THEN**
3-	Lastblock = blckchain. Last block
4-	**IF** (Lastblock.prevHash = blockchain. prevblock.hash **THEN**
5-	**Send** the Lastblock to DSDN's Controllers in system
6-	**Put** (Controller. Enventory.URL ,Northbound, Last block)
7-	**Return** "Transaction Accepted"
8-	**End If**
9-	**Else If**
10	**Return** "transaction rejected"
11-	**End.**

Blockchain API is a Rest API which enables the broadcasting of blockchain update to DSDN's controllers in proposed system a REST API of blockchain node in Flask application which is p2p API , through it the SDN's application can be reached and update the smart chain . Where PUT /POST is a write permission ;method, that allow to send HTTP requests to the application to modifying the smart chain with

3.3.3.2 The Smart Contract Based Proposed Blockchain

Building a smart contract shows in this step that receives the flow rule that required to installing in system as transaction from the system manger. It is the only can create or mine the new block. There is no need to generate many versions of a single block, like the Bitcoin and Ethereum networks do, to allow miners to select a valid and accurate block. Currently, utilizing the Flask framework in order to interface with the " blockchain" via HTTP protocol queries. Where the system manager connects to the smart contract API on the network through Postman by submitting a POST transaction request, flow rule, to the server address. A simple JSON-based flow-rule is encoded . Algorithm (3.8) explains all the operation done inside the smart contract.

Algorithm 3.8: Smart Contract Function

Input:	-Flask frameworks
Output :	New Block
1-	**Begin**
2-	**Registeration Function:** **REG_Node(IP_add , Port)** /* The system manager send request though the RESTAPI to smart contract to register a new node */ **List[].add(IP_add , Port)** /* After receiving the response the new node added to the private network*/

3-	**Block generator Function:**
	IF (details of block exists and Nodes is exists) **THEN**
	IF(Nodes is registered in the nodes list) **THEN**
	New_Transaction(JSON-based,flow-rule)
	New_block(index,prev_hash,timestamp, New_Transaction).
	/* After receiving the response of send a new JSON-based flow-rule a new block was generate with ***/**

4-	**Blockchain generator Function :** **blockchain.append(New_block)** /* add the block that produce in previouse phase to thse blockchain*/

5-	**Distributed Function :** **Blockchain_dis(List[IP_add , Port],blockchain)** /* If the blockchain is update the distributed it among all node in List[]:*/
6-	END.

In *Blockchain generator* the blockchain creating by adding a new block of transaction; new flow_rule that's the system manger wants to install in DSDN controllers, to the blockchain containing (timestamp ,Index , transaction, and previous hash) as shown in Algorithm (3.9).

Algorithm 3.9: Smart Contract based Blockchain algorithm

Input:	-New Transaction: new flow rule needs to added to DSDN's controllers
Output :	-Updated Blockchain //Adding new block
1-	**Begin**
2-	**IF** (details of New Transaction **and** Nodes **are exists**)**Then**
3-	**Call** method for create NewBlock
4-	**Add** a transaction.(Flow_Rule) **to** the NewBlock
5-	**Create** dictionary of NewBlock
6-	**Set** NewBlock .index=++
7-	**call** NewBlock .timestamp
8-	**Set** NewBlock **.**prevHash= prevblock.Hash()
9-	**Append** (blockchain, NewBlock)
10-	**SEND (**blockchain ,List.nodes) //nodes are SDN's controllers
11-	**If** response==accepted then
12-	**Return** Transaction is Done
13-	**Else If**
14-	**Return** Transaction is Not Done
15-	**End If**
16-	**END.**

The Algorithm (3.10) shows the constructing of the REST API of the Smart Contract , which through it the system manger can reach to smart contract and send new transaction (flow rule),when the smart contract receives the transition ,check the required filed in posted data if it exits , it works on to generate new block and append it with blockchain and send it to the all nodes in its list. Each block in smart must content with following field(index , timestamp , transaction, previous_hash) .

Algorithm 3.10: Smart Contract REST API

Input:	-Flask frameworks
Output :	REST API
1-	**Begin**
2-	**Define** a route that enable the System Manger reach the Smart Contract @app.rout('/transaction/new', methods =['post'])
3-	**IF (List . node >0) THEN**
4-	Index++ Call Blockchain.new_transaction(required field) Response with new block(index,transaction,previous_hash, timestamp)
5-	**END.**

3.3.3.3 Enforce a New Flow Rule Blockchain based DSDN Controller

Instance of sending the new rule as transaction to DSDN's controller to update the inventory shard and enforce a flow rule ;sending by system manger. Using the postman application by submitting a POST transaction request, flow rule, to the server address use its API with a simple JSON-based flow-rule .

3.3.4 Security Discussion of The Proposed System

Our assessment of the suggested system's security was based on the well-known information security paradigm known as the confidentiality-integrity-availability (CIA) triangle. The suggested system is safe from dangers that might compromise the security goals. These goals include confidentiality, which ensures that only authorized parties have access to data, integrity, which ensures that data is accurate and cannot be manipulated, and availability, which ensures that data parties can actually be reached.

Chapter Four

The Results Discussion and Analysis

4.1 Overview

This Chapter deals with the testbed of Distributed SDN network that was simulation in GNS3.where the testbed built based on Opendaylight controller ,mininet emulator and OpenFlow protocol. It evaluates the performance the central SDN network against the Distributed SDN network spatially in **security issues**(single point of failure) and **scalability**, where this comparison between the performance of central SDN network and distributed SDN ,is essential to determine whether the proposed Distributed SDN network is practical before implementing them in the real world. Blockchain network as consensus mechanism with Distributed SDN network is also evaluated in this Chapter.

The Chapter consists of two phase: the first one contains the simulation process of the testbed ,the utilized parameters, and the result of performance evaluation between traditional SDN network and distributed SDN. The second presents the process of built the Blockchain network which contains the nodes of the network built ,Rest API built ,smart contract built ,c consensus mechanism , connect the Blockchain network with distributed SDN and the result of evaluation blockchain _based Distributed SDN network as a consensus mechanism. To confirm this collaboration between the new technologies SDN network and blochchain network resulted in an improvement in terms of the required consistency in distributed SDN networks.

4.2 Distributed Controls Strategy (phase one)

This phase aims to implement and comparison between traditional SDN network and distributed SDN network and performance evaluation , especially with essential issues that encounter the traditional SDN network(central).

4.2.1 DSDN Network Simulation In GNS3

In this section we show the configuration of distributed SDN in GNS3as a test bed table (4.1) lists the software that is used for this section. These software are run on a HP DESK-TOP-F5259A5, with a Processor Intel (R) Core (TM) i7-10750H CPU @2.60GHz, RAM 16.0GB. The GNS3 is used as the environment on which all other software is set up. The Ubuntu-servers type 18.04.3LTS are used, an OpenDayLight controller is installed through ,the configuration of a custom network topology using mininet. The Python script used to build a custom topology in mininet and the applications of the opendaylight controller. This topology remotely controlled by three ODL controller. All data networks make use of Open Flow switches (OVS switch 1.3) .

Figure(4.1) shows the design of the central SDN in GNS3, using Beryllium SR4 OpenDayLight controllers and the Python script to build a custom topology in mininet and the applications of the opendaylight controller. This topology remotely controlled by the ODL controller. All data networks make use of Open Flow switches (OVS switch 1.3).

While for configuration a distributed SDN in GNS3as a test bed . can see in the appendix A Figure (A.1) ,that the Beryllium SR4 Opendaylight controllers operate on the Ubuntu Server with IP address

(192.168.122.209) , (192.168.122.208) and(192.168.122.120) while the mininet on IP address (192.168.122.128) .

To test the connection between the remote controllers and mininet ,a custom topology is operated on the mininet as we can see in the appendix A Figure(A.2). the reachability testing through the of use the Pingall Toole, the result of test shows that the reachability is 100% ,and there is 0% dropped packet as seen in the A appendix Figure(A.3).

Table (4.1): The Software Used in the Experiments of this Work

Software	**Function**
GNS3	Graphical network simulator
Mininet	Custom Network topologies
Open Flow Switch	Virtual SDN Switch
OpenDayLight	SDN controller Platform
VMware Workstation	Virtualization Software
Ubuntu-server 18.04.3LTS	Host Operation System
Python	Programming language
Wireshark	Packet Capture

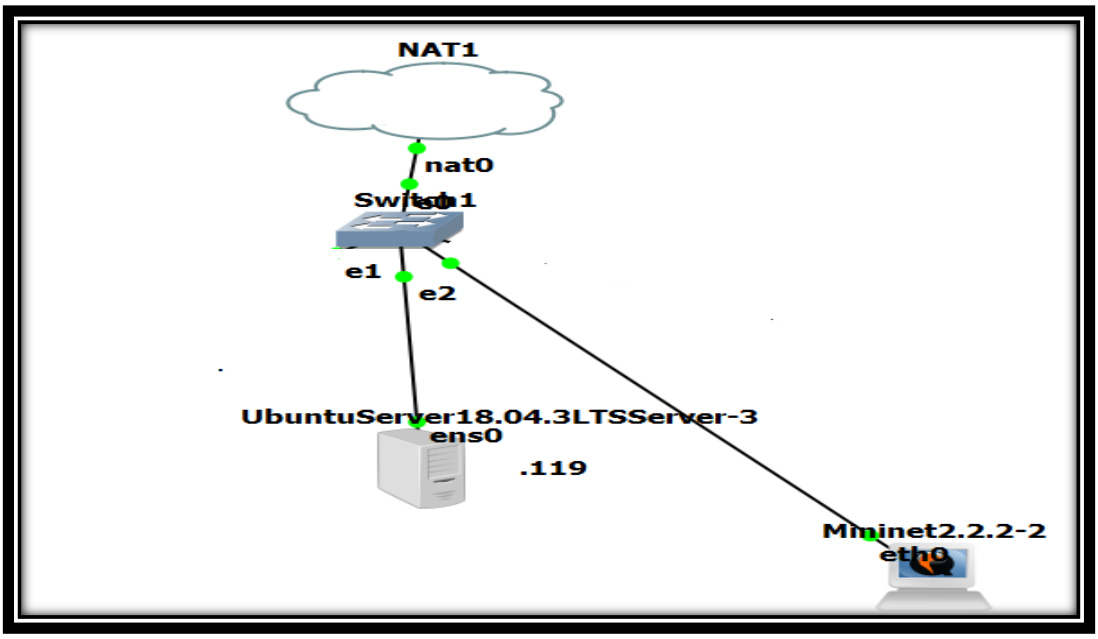


Figure 4.1:The Design of the Central SDN Network in GNS3

As we can in the Figure (4.1),that the Beryllium SR4 Opendaylight controller operates on the Ubuntu Server with IP address (192.168.122.119) while the mininet on IP address (192.168.122.128) .

To test the connection between the remote controller and mininet ,a custom topology is operated on the mininet. With opendaylight Dlux , the topology that control remotely by the opendaylight controller can be seen ,as the Figure(4.2) shown .

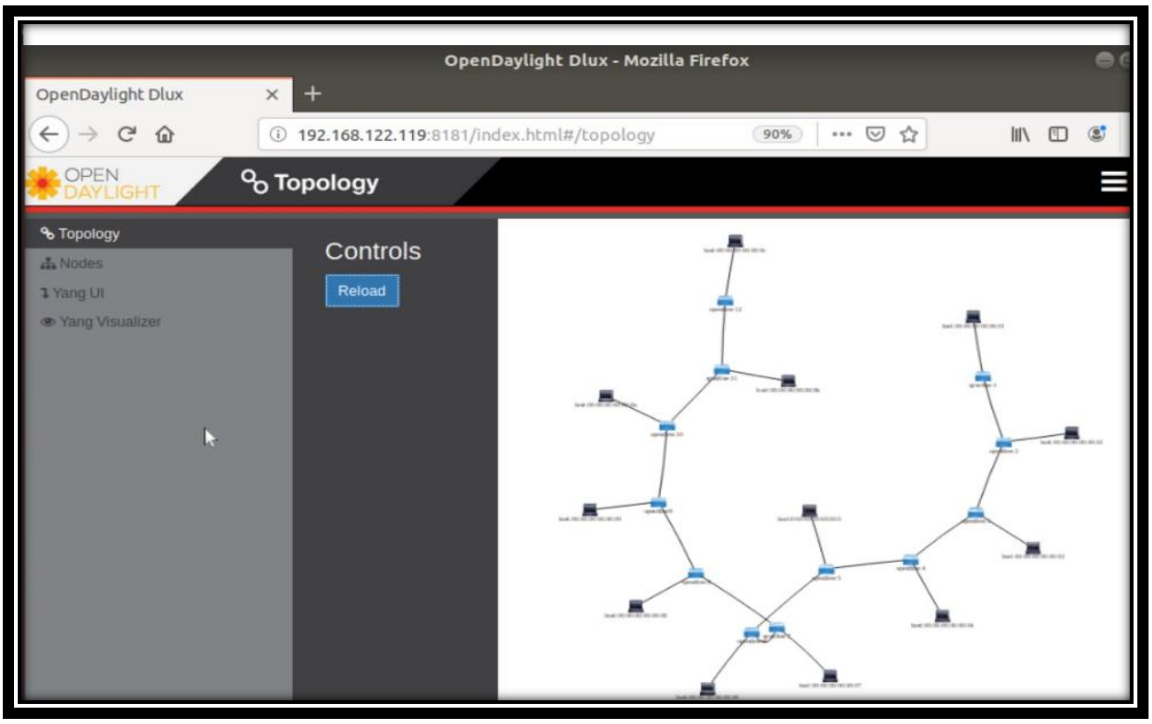


Figure 4.2:The Connection Between the Remote Controller and Custom Topology in Mininet in Dlux

Figure(4.3) shows the reachability testing through the use the Pingall Toole , the result of test shows that the reachability is 100% ,and there is 0% dropped packet.

```
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12
h2 -> h1 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12
h3 -> h1 h2 h4 h5 h6 h7 h8 h9 h10 h11 h12
h4 -> h1 h2 h3 h5 h6 h7 h8 h9 h10 h11 h12
h5 -> h1 h2 h3 h4 h6 h7 h8 h9 h10 h11 h12
h6 -> h1 h2 h3 h4 h5 h7 h8 h9 h10 h11 h12
h7 -> h1 h2 h3 h4 h5 h6 h8 h9 h10 h11 h12
h8 -> h1 h2 h3 h4 h5 h6 h7 h9 h10 h11 h12
h9 -> h1 h2 h3 h4 h5 h6 h7 h8 h10 h11 h12
h10 -> h1 h2 h3 h4 h5 h6 h7 h8 h9 h11 h12
h11 -> h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h12
h12 -> h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h11
*** Results: 0% dropped (132/132 received)
mininet> iperf
*** Iperf: testing TCP bandwidth between h1 and h12
*** Results: ['15.1 Gbits/sec', '15.1 Gbits/sec']
mininet> _
```

Figure 4.3:The Reachability Testing

4.2.2 The Performance Evaluation Comparison Between Central SDN network and DSDN

This section shows the result of evaluation's result between performance of central SDN's network and DSDN's network. The Wireshark software used to analysis the real time capture network packets to comparison the performance with metrics as Topology Discover, scalability (Throughput , latency) and server source consumer (CPU and Memory).

4.2.2.1 The Security Issues of Traditional SDN

This section is a study to the security issue that faces the traditional SDN network ,which represents in single point of failure. Where the controller can be fail to reach and mange/control its infrastructure as result for software/hardware issues . To do this, the remote controller (opendaylight) is logout/shout down and then deleted the flow entry in OVS switches.

The test of reachability in this case shows that ,the dropped packet is 100% and the received packet is 0% as shown in the Figure (4.4).

Figure 4.4: The Reachability Testing

This result comes because that the controller cannot reach to switches and install the required flow entry to deal with coming packet, so all the packet is dropped.

4.2.2.2 Study of the Scalability Issues in a Traditional SDN

This section is a case study to the Scalability issue that faces the traditional SDN network ,in this test, the performance of the controller is evaluated with two metrics **Throughput** and **Latency** the experiment done with two topology linear topology and tree topology with diffident number of OVSswitch start 20 switch and in each experiment increase the number with 20 switch. The Figure(4.5) shows The results of case study to evaluate the Throughput with tree topology and linear topology, when the number of switches (nodes) in data plane are 20 the average throughput number that can be dealt with by the opendaylight controller near to 45 (Flows/MS),while this number decreases to reach 9 (flows/MS) when the number of OVSswitches on data plane reach to 180 . With linear topology when the number of switches (nodes) in data

plane are 20 the average throughput that can be deal with by the opendaylight controller near to 39 (Flows/MS),while this number decreases to reach 5 (Flows/MS) when the number of switches on data plane reach to 180. The Figure (4.5) shows that the throughput of SDN's controller decreased when the number of OVSswitches on the data plane increased This is due to an increase in the load on the controller by increasing the number of nodes in the data plane , which causes a decline in its performance. In addition, the topology of the data plane has an effect on the throughput; the same Figures show that the throughput of SDN's controller with tree topology is higher than SDN's controller with linear topology with the same number of OVS switches on the data plane This is due to the nature of the connection of nodes with each other in the topology of infrastructure and the propagation mechanism between

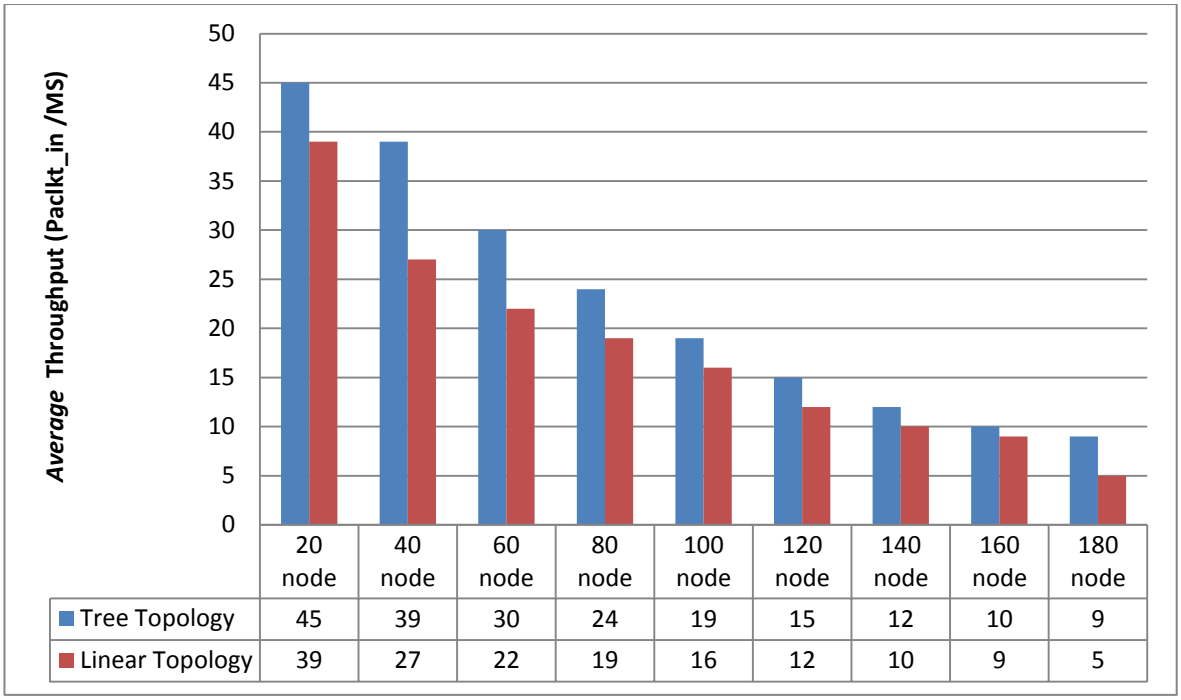
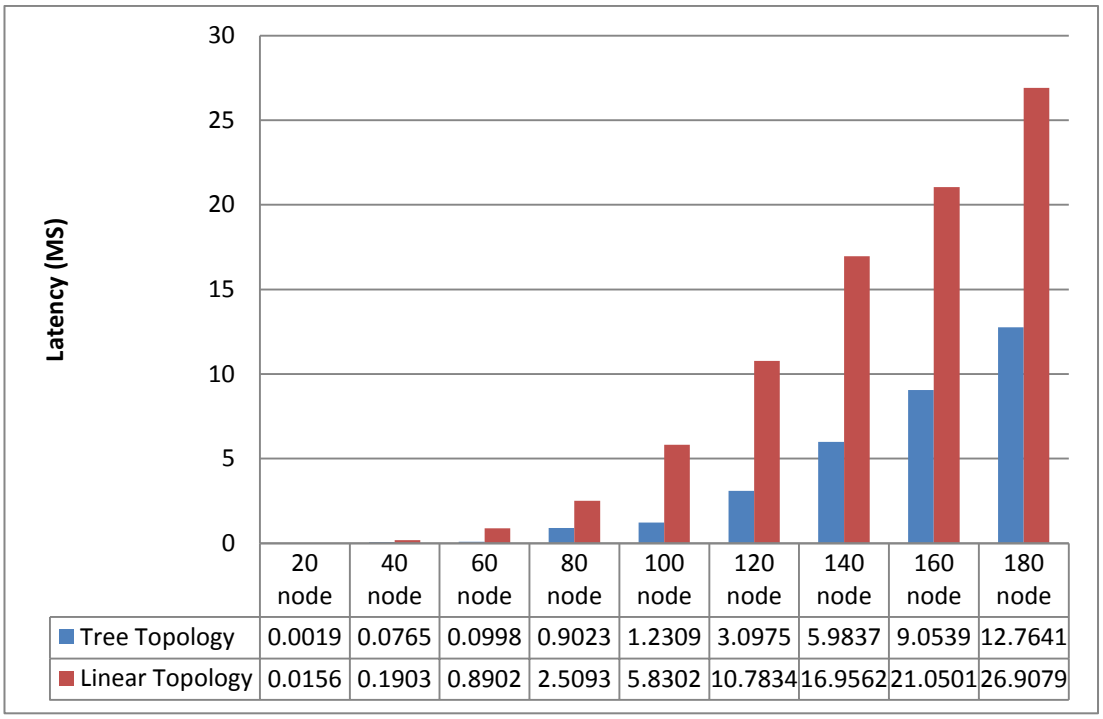


Figure 4.5:The Average Throughput of Central SDN Controller

The Figure(4.6) shows the result of latency metrics of opendaylight remote controller . The result of case study with tree topology is shown in Figure (4.6) , when the number of switches (nodes) in data plane are 20 the response time the opendaylight controller near to 0.0019(MS),while this number increases to reach 12.7641 (MS) when the number of switches on data plane reaches to 180 . With linear topology when the number of switches (nodes) in data plane are 20 the response time of the opendaylight controller near to 0.0169(MS),while this number increase to reach 26.9079 (MS) when the number of switches on data plane reach to 180. The Figures (4.6) shows that the latency metrics of SDN's controller increases when the number of OVSswitches on the data plane increase due to increase the requests that must process it and provide the replay for it this cause in increase in latency. In addition, the topology of the data plane has an effect on the latency metrics; the same Figures show that the latency of SDN's controller with tree topology is lower than SDN's controller with linear

topology with the same number of OVSswitches on the data plane This occurs because of the architecture of the infrastructure and the process of transmission between the nodes.



	20 node	40 node	60 node	80 node	100 node	120 node	140 node	160 node	180 node
Tree Topology	0.0019	0.0765	0.0998	0.9023	1.2309	3.0975	5.9837	9.0539	12.7641
Linear Topology	0.0156	0.1903	0.8902	2.5093	5.8302	10.7834	16.9562	21.0501	26.9079

Figure 4.6:The Average Latency of Central SDN Controller

4.2.2.3 Single point of Failure (SPoF)

This experiment examines (SPoF) issue **First**, in central SDN's network, where OVS switches, fail in forwarded the incoming new packet without connect with remote controller and installed new flow entry the dropped packet was 100% ,while in DSDN when all controllers were operated the dropped packet was 0% ,while when one of this three controllers were logout the percentage of dropped packet raise to 12% ,and this as result to the system not fulfillment the required of DSDN with the minimum number of controller is three. And this percentage raise to reach 22% dropped packet when we logout additional controller and the system operates with one controller.as it is shown in Figure(4.7).

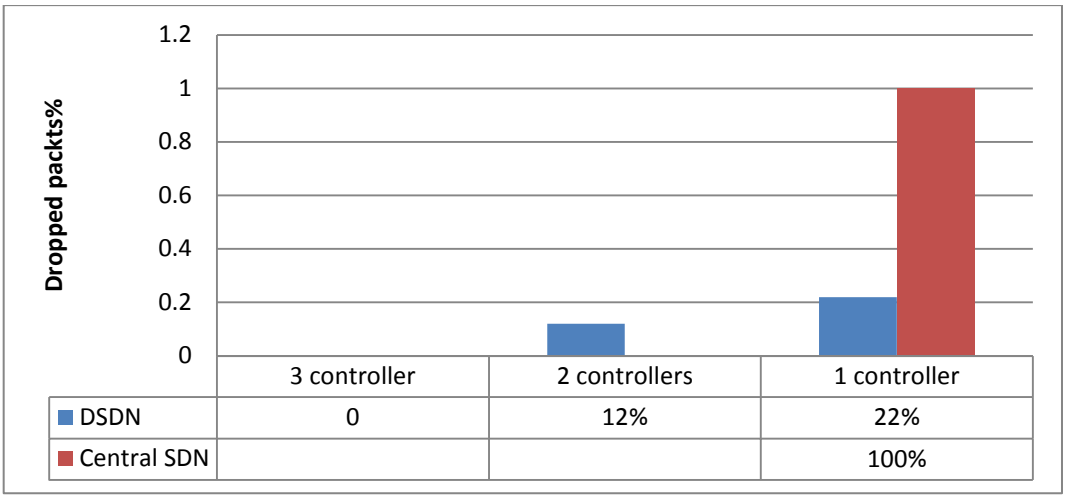


Figure 4.7:The Single Point of Failure in Central SDN Controller VS. Distributed SDN Controller

4.2.2.4 Topology Discover

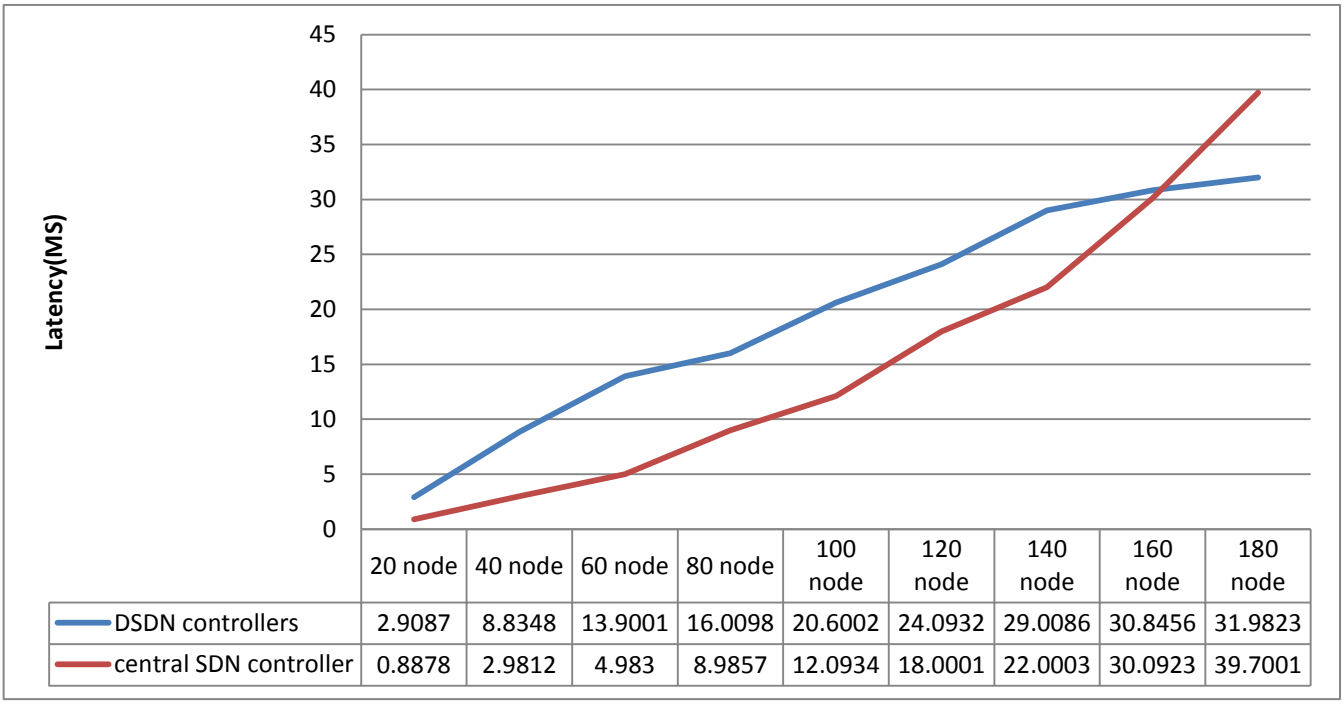
On VM of the mininet ,start the tree topology with custom no.of switch and using the Wireshark to capture real time traffic between mininet and ODL controller ,computing the average of LLDP cycle recording the sending time of the first LLDP, the Link Layer Discovery Protocol where each switch had send a PacketIN encapsulated LLDP message back to the controller. These PacketINs are answering the LLDP messages the controller has send to discover the network. For each cycle, calculated the time of the LLDP cycle which is the difference between the sending time of the first LLDP message sent by the controller and the reception time of the last LLDP message where this cycle happened every 5sec in opendaylight controller.

Figure (4.8)shows that the performance of the central controller is better than the distributed until the no.of node in data plane near to 160 this can be because of consistency among the controller in the distributed ,but when the infrastructure increases the performance of distributed ODL becomes better than central controller ,where its

performance becomes more stable, this due that the load of the network distributed among the DSDN controller instead of the traditional SDN where all the load of the infrastructure lay on central controller .

	20 node	40 node	60 node	80 node	100 node	120 node	140 node	160 node	180 node
DSDN controllers	8.9087	12.8348	20.9001	27.0098	27.6412	28.9812	29.98731	30.0015	30.9823
central SDN controller	2.8878	4.9812	8.983	13.9857	15.0934	18.0923	22.9003	28.0923	48.0923

Figure4.8:The Tree Topology Discover



	20 node	40 node	60 node	80 node	100 node	120 node	140 node	160 node	180 node
DSDN controllers	2.9087	8.8348	13.9001	16.0098	20.6002	24.0932	29.0086	30.8456	31.9823
central SDN controller	0.8878	2.9812	4.983	8.9857	12.0934	18.0001	22.0003	30.0923	39.7001

Figure4.9:The linear Topology Discover

Figure (4.9)shows that there are a Progressive relationship between the time of Topology Discover and no .of nodes in data plane . The performance of central controller is better than distributed until the no.of node near to 160 node, while the Topology Discover of distributed ODL after this number be more stable. In general the time required for discover the linear topology is less than time of tree topology with 5% , This is due to the mechanism of linking between nodes in infrastructure and the nature of packet propagation among them where in linear topology when h1 send packet to h5 as example the packet must goes through all the nodes between the sender and the receiver .

4.2.2.5 Scalability

To analyze the network performance when the number of OVS Switches is increased where can be define the scalability as the ability of the controller to control and manage large number of the nodes without an obvious change in performance, for this experiment we use each of the throughput and latency as metrics to scalability of Opendaylight controllers.

At the beginning , the ODL controllers work in Proactive mode as default, so must be set ODL to **Reactive** mode modifying some configuration files as Figure (4.10) shows.

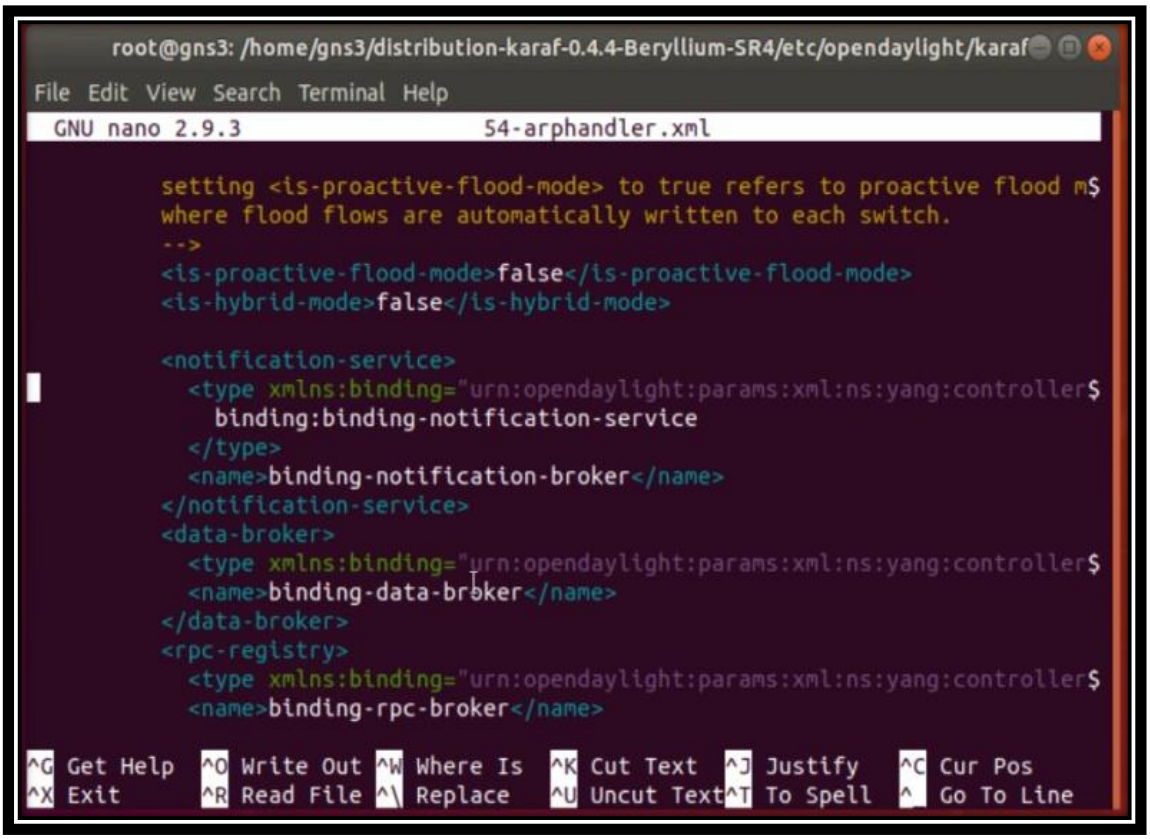


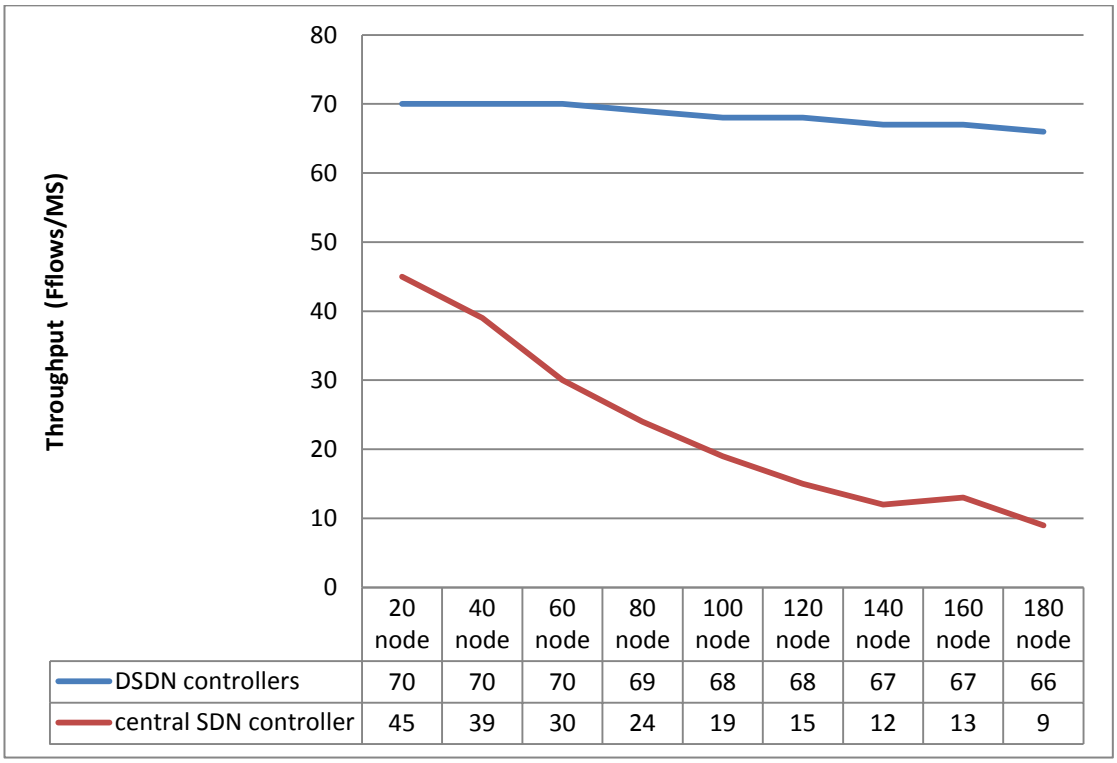
Figure4.10: The Setting ODL to Reactive Mode Modifying Some Configuration

4.2.2.5.1 Throughput

As it is shown in Figures (4.11) and (4.12) the rate at which the controller, the single controller, and the distributed controllers handle flow requests is proportional to the rate at which PACKET IN events with an ARP payload are generated .

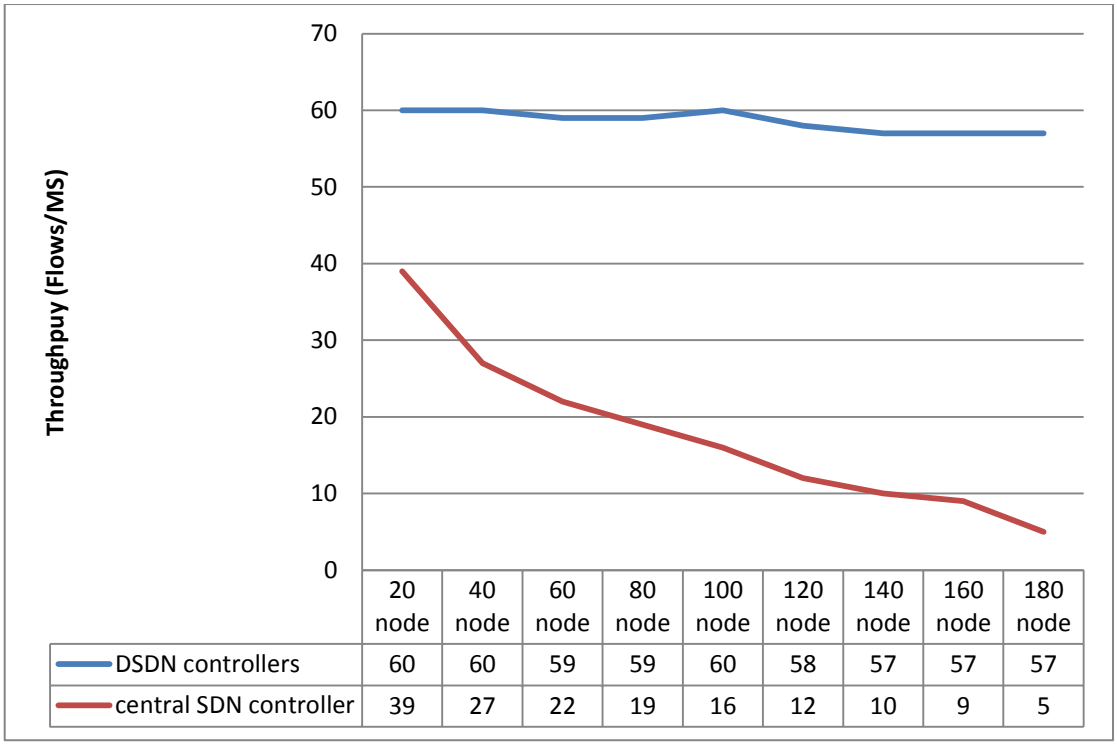
As it can be seen in (4.11) and (4.12), the throughput of decentralized opendaylight controllers is 45.4 percentage points which are higher than that of the centralized controller. In addition, when the number of devices in the data plane grows, performance remains more consistent with distributed SDN due to the even distributed of network load among DSDN controllers, while throughput drops down as the number of devices grows in the central controller ,because the all load of the network on one controller so with increase of request on it its performance was decrease and may led it to out of service . Finally,

observing that the throughput of SDN networks with a tree topology is more consistent than that of networks with a linear topology because there is a decrease in overall bandwidth utilization and an increase in the propagation time between nodes.



	20 node	40 node	60 node	80 node	100 node	120 node	140 node	160 node	180 node
DSDN controllers	70	70	70	69	68	68	67	67	66
central SDN controller	45	39	30	24	19	15	12	13	9

Figure 4.11:The Throughput of Opendaylight Controller for Tree Topology



	20 node	40 node	60 node	80 node	100 node	120 node	140 node	160 node	180 node
DSDN controllers	60	60	59	59	60	58	57	57	57
central SDN controller	39	27	22	19	16	12	10	9	5

Figure4 .12:The Throughput of Opendaylight Controller for Linear Topology

4.2.2.5.2 Latency

The time between the packets is sent to the controller and the response on it is a Latency. To compute the latency using the ping Toole between the first host in the network and the farthest one after the end of the first cycle of the topology discover.

According to the data, the centralized opendaylight controller has a 3.3 % greater latency than distributed SDN controllers as shown in Figures (4.13) and (4.14). Distributed SDN maintains a more consistent latency as the number of devices in the data plane grows, in contrast to the centralized controller, where latency grows as the number of devices grows. The latency with the linear topology is also clearly seen to be greater than the latency with the tree topology by a margin of 2.46% ; this is because the time it takes for intermediary nodes to propagate a packet to its destination grows proportionally with the number of hops that exist between the source and the destination.

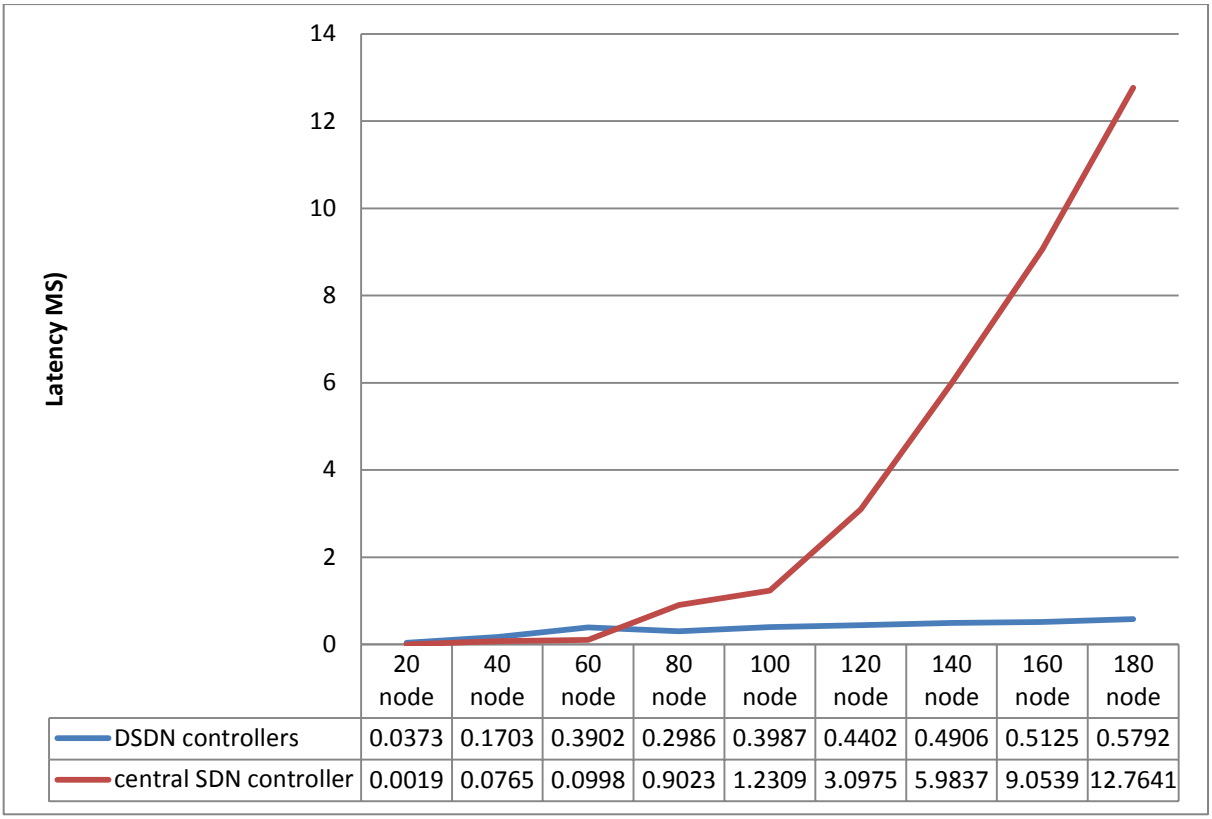


Figure 4.13: The Latency of Opendaylight Controller for Tree Topology

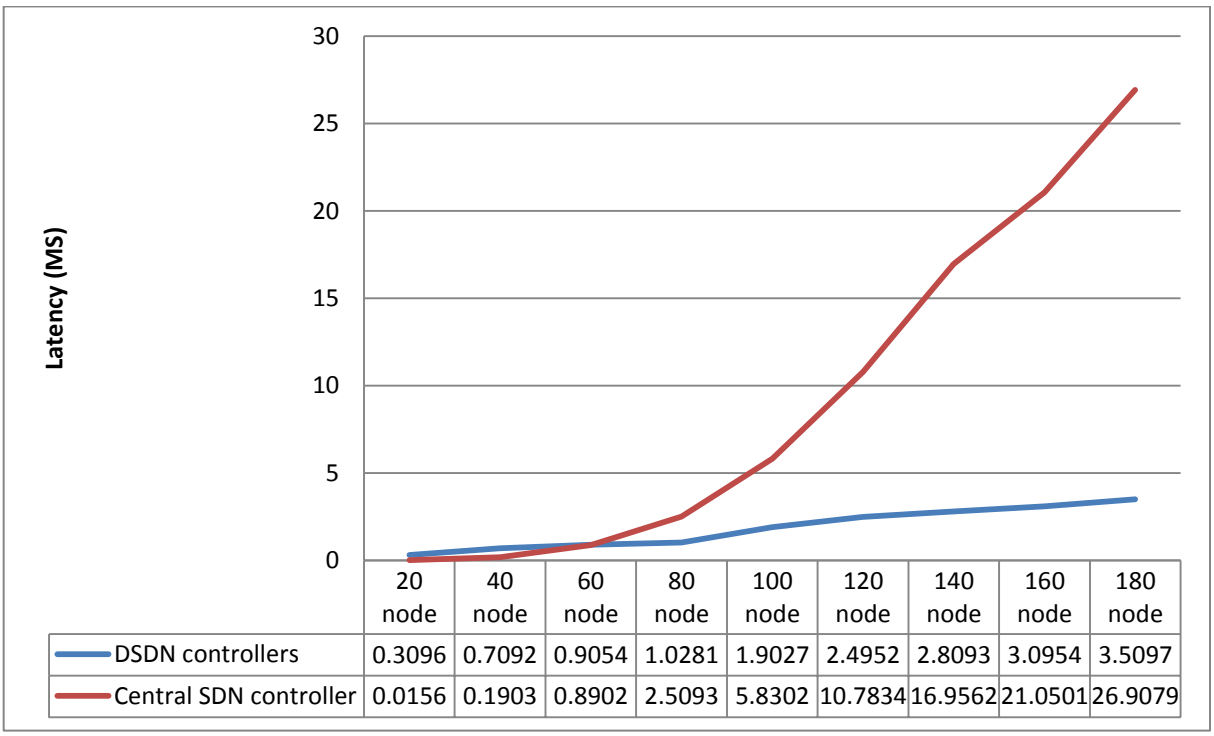


Figure 4.14: The Latency of Opendaylight Controller for Linear Topology

4.2.2.6 Server Source Consumer (CPU and Memory)

Figures (4.15) and(4.16) show that as the number of controllers in the network grows , so the number of users who could make use of the system's memory and its storage space. Experiment results show that after a few minutes of network activities, CPU consumption may drop by half, and the system becomes more stable this because at initially of the system the DSDN controller work on network topology discover ,and provide a OVS switches in data plane with appropriate flow rule that reside in flow table of OVS switches. but the main problems lie with memory as Figure(4.16) shows and this due to the machine replication among the DSDN controller to reach a consistency in data state of network , Taking into account the a computer used to carry out the work as it explain in section (4.2.1).

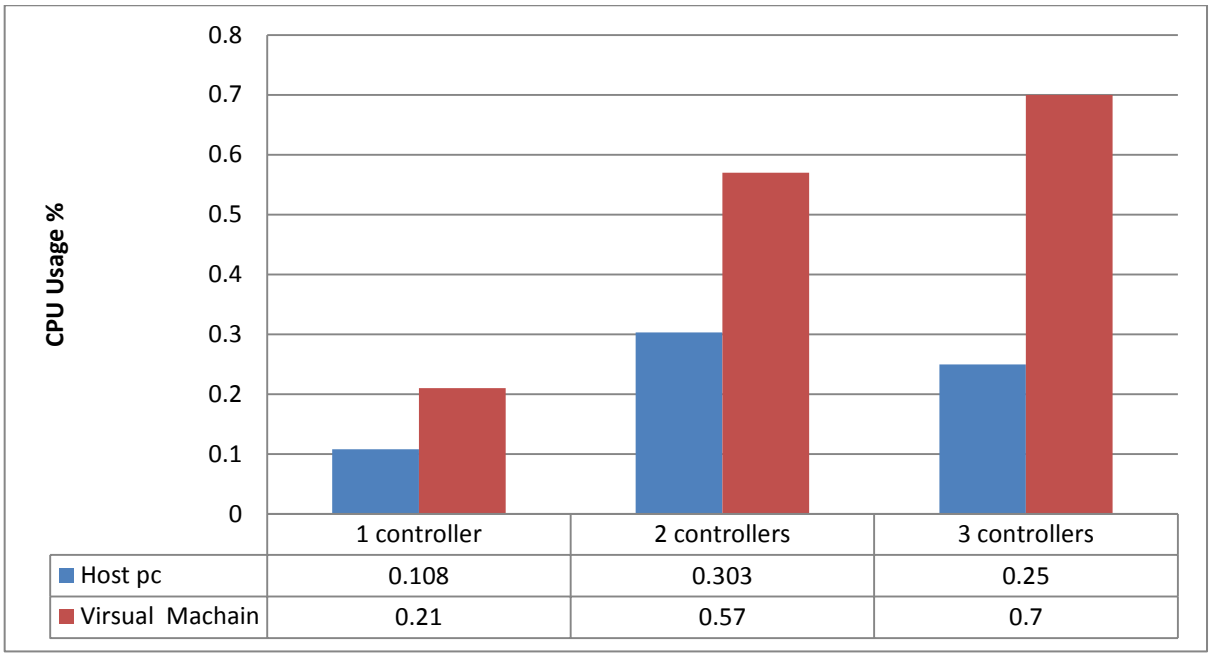


Figure 4.15 :The Servers CPU Consumer

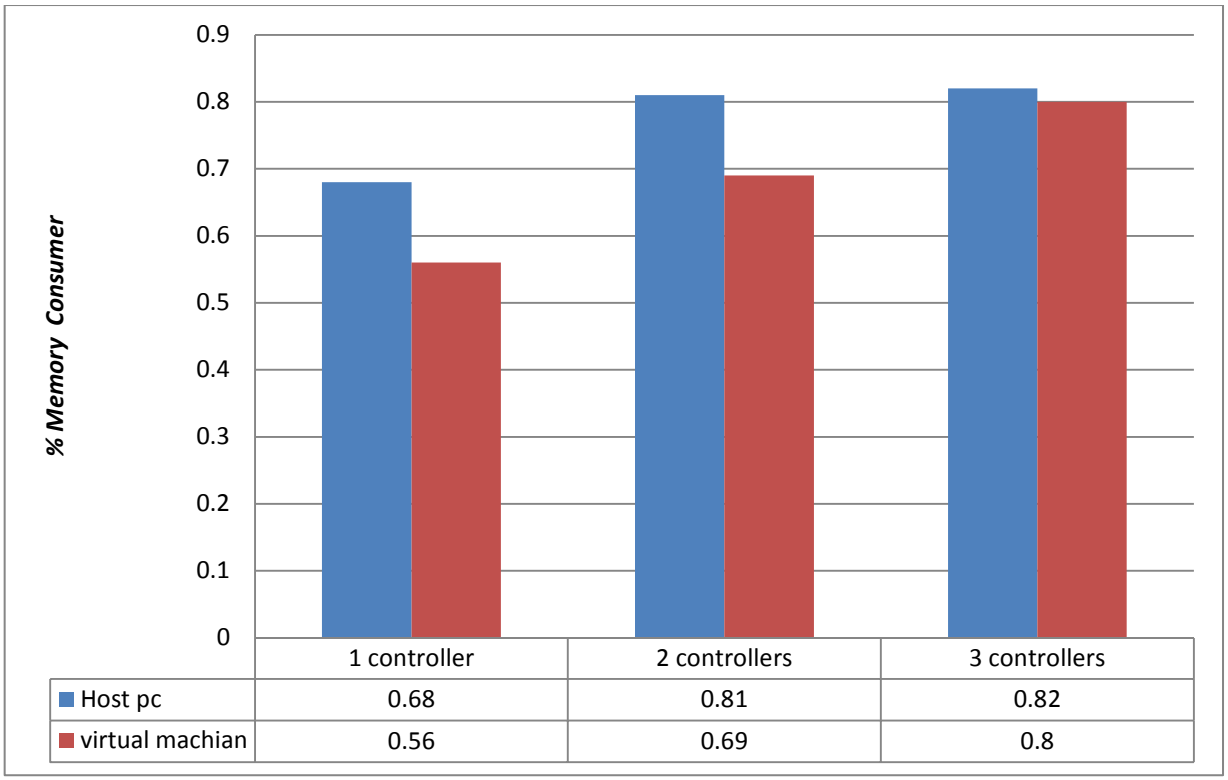


Figure4.16:The Servers Memory Consumer

4.3 The Blockchain-Enable Distributed SDN's Controller Flow Rule as Consensus Mechanism(Phase2)

This section shows the steps that taken to create a blockchain network, including the installation of nodes, the development of a Rest API and a smart contract, the establishment of a link between the blockchain and a distributed SDN, and the outcome of an evaluation of the blockchain-based Distributed SDN network's efficacy as a consistency mechanism. The goal of this part of the dissertation is to verify that when SDN networks and blockchain networks function together, the requisite consistency in distributed SDN networks is increased.

4.3.1 The Consensus plane of DSDN's Controllers

By using a python language ,three applications are built that each of them connects to one of the controller in the DSDN system ;work as consensus plane in proposed system. The first ones built on IP:192.168.122.209 Port:5001, the second one is with IP:192.168.122.208 Port:5000 while the last one is IP:192.168.122.120 Port:5000,where the virtual environment is configuration as an essentially step to build a Flask application as it shows in A appendix Figures (A.6),(A.7)and (A.8).

4.3.2 The Smart Contract based proposed Blockchain

This section is about building a smart contract that receives the flow rule that required installing in system as transaction from the system manger. It is the only one that can create or mine the new block

-The smart contract is built on server with IP:192.168.122.209 Port:5000.

- Building the RESTFULL API by utilizes the Flask framework and python programming language in order to interface with the " blockchain" via HTTP protocol queries as shown in Figure(4.17).

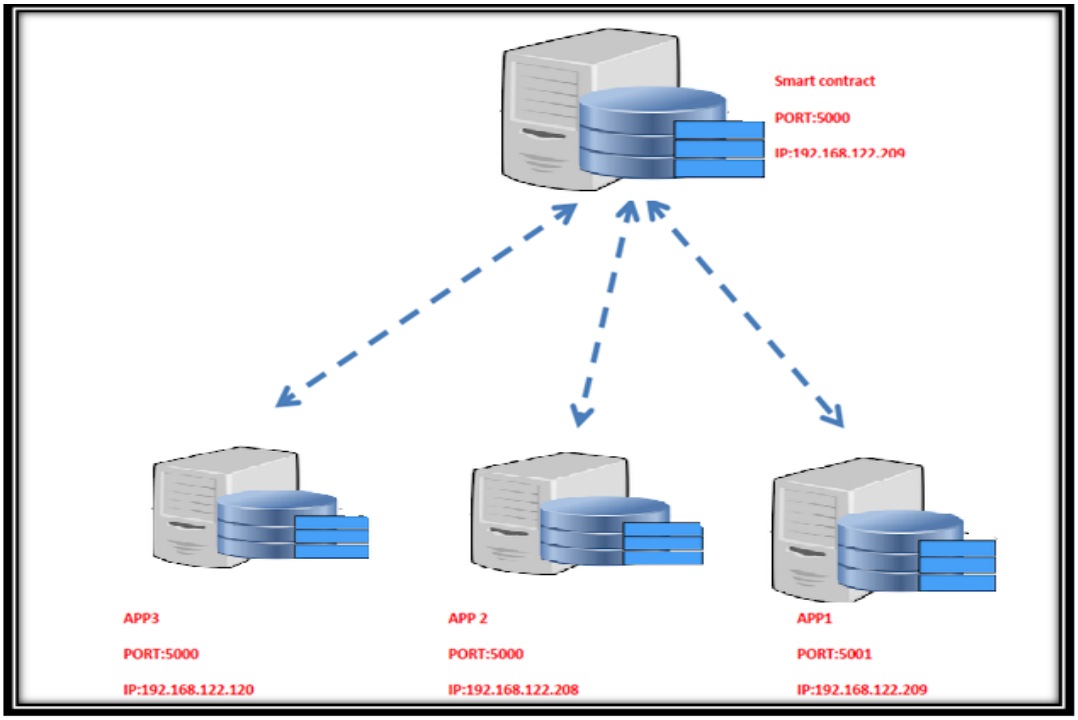


Figure 4.17:Design of Blockchain in Proposed System

Where the system manager connects to the smart contract API on the network through Postman by submitting a POST transaction request , flow rule, to the server address. A simple JSON-based flow-rule is encoded as shown in A appendix Figure(A.9) which explains all the operation done inside the smart contract.

The first operation that is done in smart contract shown in A appendix Figure (A.10) which represents in the registration of nodes of privet blockchain network in smart contract, as it is shown in the (A.11) adding new node with IP (192.168.122.120) to list in smart contract.

The initial state of the smart chain where there are no flow rule send throw the smart contract, where it represents the genesis block shown in appendix A Figure (A.12) .

As it is mentioned above in Figure (A.9) which shows the blockchain after post flow rule by the system manger to the three ODL controllers through the smart contract. Figure (A.13) shows that the

application that connects with the controller with IP 192.168.122.120 receives the update blockchain and update the inventory shard of the controller.

4.3.3 The Private Blockchain to Enforce a New Flow Rule on the DSDN Controller

Instance findings are presented here, using Postman a POST request was submitting as transaction , flow rule, to the server address, which in turn sends the new rule to the smart contract as a transaction. There is a straightforward JSON-based flow-rule that it employs through its API as it shows in Figure (A.14) in appendix A.

4.3.3.1 Performance Evaluation(phase 2)

By timing the duration of write transactions between system manger and the SDN controller, were be able to assess the performance impact incurred by the proposed architecture. How long it takes for the application to get a response after requesting a change to the ODL controller's resources which is what this metrics measures, by contrasting the time overheads with/ without blockchain based DSDN, the performance overhead caused by our solution is determined. The resource-updating process for proposed DSDN controllers, ODL, is shown in Figure (4.18). whereas that for legacy DSDN controllers, depicted in Figure (4.19); As shown in the Figure, the request must be sent to the leader controller, and then propagated from the leader controller to the other controllers in the system. A successful update is reported after the leader has received a response from the followers.

As case study to time overhead evaluation of write a new flow_rule; First, when an application called the ODL controller's REST API, used the blockchain based DSDN. Second, when an application called the ODL controller's REST API, no blockchain-based solution was used.

To quantify the time overheads caused with / without the suggested approach, using the POSTMAN tool as an application that makes a ODL controller's request., the HTTP POST request to the Smart Contract REST APIs was issued via the POSTMAN tool. Because a new rule is being deployed to all DSDN remote controllers, as a write transaction, this API demonstrates. the average overheads of separate write transactions was computed.

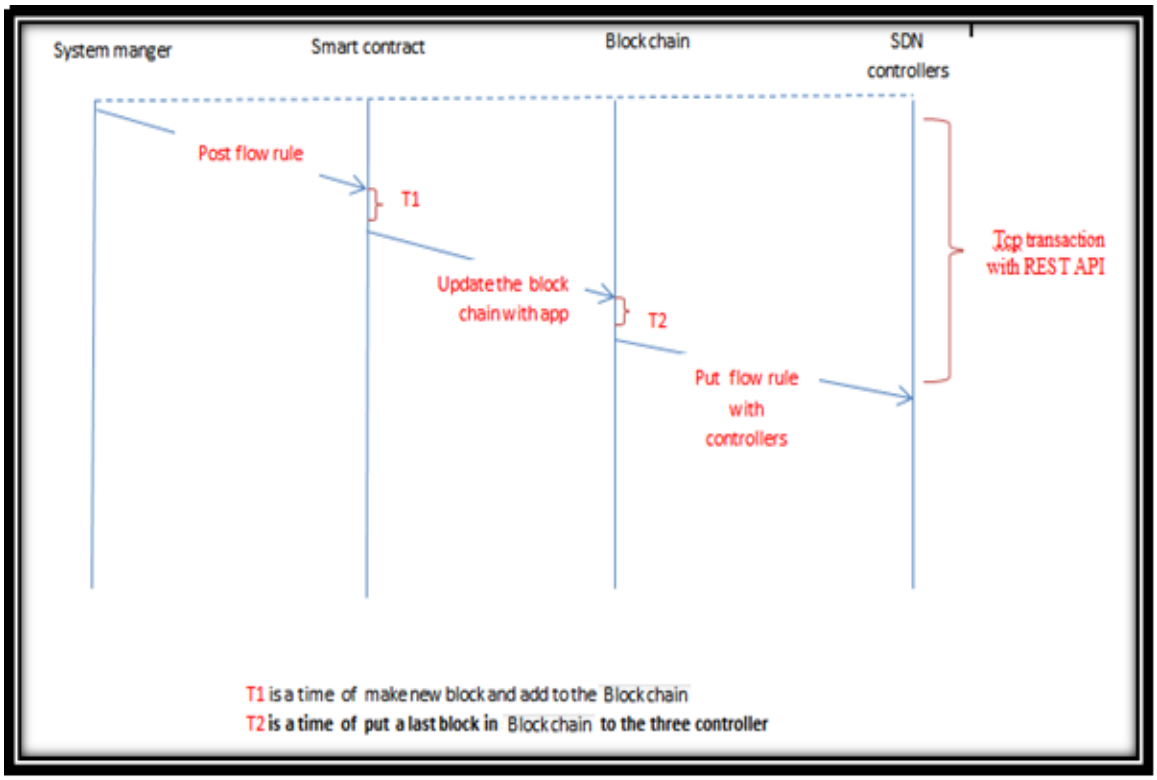


Figure 4.18: The Data Update with Proposed Model

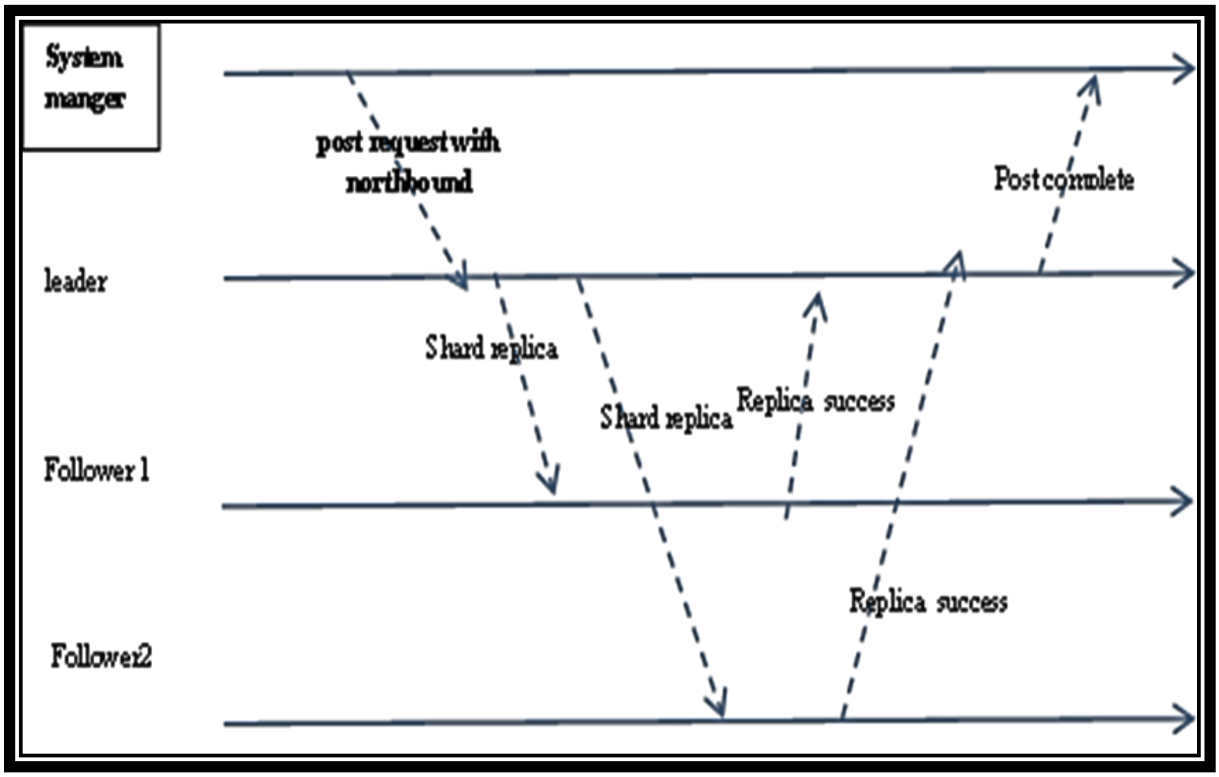


Figure 4.19: Data Update Legacy Opendaylight Controller

Figure (4.20)illustrates the results for the time consumption of the write transaction. It is obvious, from the figure, that applying blockchain technology decreases the packet processing time in our solution compared to the legacy method. Nevertheless, the block-based method decreases the overhead by only about 141ms.

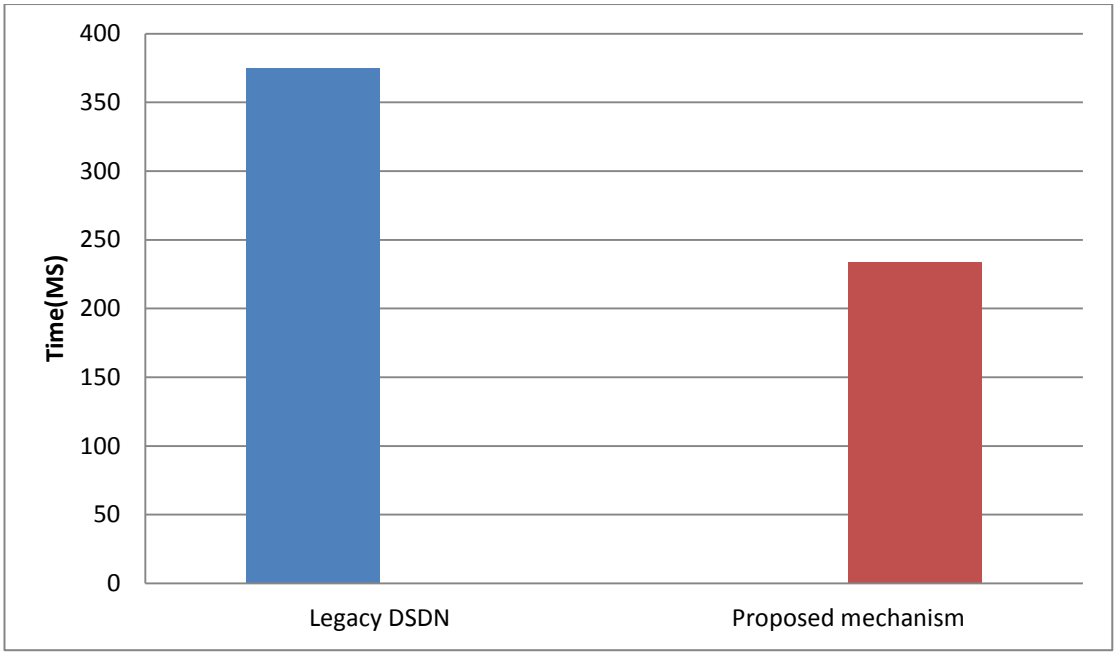


Figure 4.20: Evaluation of Write Time Overhead

4.4 Security Discussion of The Proposed System

The well-known information security paradigm known as the confidentiality-integrity-availability (CIA) triangle was the basis for our security evaluation of the proposed system as Table (4.2)shows , where utilizes it to gauge how well our system secure. The proposed system is protected from various threats to the security objectives. These objective are confidentiality ,which guarantees that data is shared and accessed only by approved parties, integrity ;which guarantees that data is correct and cannot be altered, and availability which guarantees that data\ parties are really accessible.

4.4.1 Confidentiality

The goal of maintaining information privacy are to prevent its disclosure to unauthorized parties. In order to maintain confidentiality, using a private blockchain in which only a registered DSDN's controllers may take part in the consensus process and the distribution of flow rules

is strictly regulated. Data at rest in blockchain ledger and in transit to DSDN controllers are both safeguarded in this manner. To ensure privacy, proposed system uses techniques as encryption and authentication.

- **Encryption technique**: This approach is meant to make data inaccessible to the an unauthorized SDN's controller during transmission between controllers layer and data layer ,and between blockchain network nodes. Utilizing a Hyper Text Transfer Protocol Secure (HTTPS) over Transport Layer Security (TLS) to encrypt data during transmission so the data in the transfer phase cannot be readable and allowed a secure channel to DSDN controllers.
- **Authentication technique:** In order to safeguard the DSDN controllers against intrusion, this method validates the IP address. Registering the IP addresses of DSDN controllers in smart contracts is the first line of security against illegal access to the SDN controller. Before a request may connect with the DSDN's controllers, its IP address is checked to make sure it matches the real preregister IP addresses recorded in our smart contract. Accordingly, this method reduces the risk of spoofing by verifying the identity prior to any interaction with the SDN's controllers.

Scenario 1: Assume that a malicious request to enforce fake flow rule, a session token for authentication, which is produced and confirmed by the smart contract, is something the attacker would need in order to prove its identity. If an attacker manages to steal the authentication session token, the situation is danger. Hence, this

method reduces the risk of spoofing by verifying the sender's identity before establishing contact with the SDN's controllers.

4.4.2 Integrity

Data integrity refers to the absence of unauthorized changes to the data. Both the data kept on blockchain ledger and the data transmitted to DSDN controllers must be guaranteed to be incorruptible. Using blockchain technology to protect the integrity of the information stored in the blockchain ledger and smart contracts. Using the blockchain feature, the suggested approach ensures the immutability of data stored in the blockchain nodes, since each block includes a hash value of its content and a hash value of the preceding block, forming secure linkages between the blocks. To ensure that no data is altered in transit, the proposed system makes use of REST APIs that are founded on the HTTPS protocol. This secures the connection between the blockchain network and the DSDN controllers, The information kept in the blockchain nodes and transmitted information are safe.

Scenario 2:

Consider the scenario where an attacker has successfully attacked one of the nodes in a blockchain network and is attempting to update the distributed ledger. The smart contract in the encrypted API can identify this kind of attack as soon as any change or update is made to the blockchain. Each node in the blockchain network calculates a unique hash for each block and establishes a connection to the hash of the prior block, allowing for the detection of discrepancies between the hash of a block that has been tampered with and the hash that was previously generated. Each node checks the last timestamp of the updated

blockchain and compares it to the latest timestamp of its own blockchain, and each node checks the number of blocks in the updated blockchain and compares it to the number of blocks in its own blockchain . So, in addition to checking the block's hash value and its connection to the hash of the preceding block, the validation checks the total number of blocks and the most recent timestamp in the blockchain. The information kept in the blockchain Peers is safe because to our secure architecture.

4.4.3 Availability

The goal of availability is to ensure that the system is available and responsible for a valid request. This security goal assures that the system is reliable and available and this is done by two techniques the first one is by distributed SDN network which is deals with the SPoF traditional SDN network that suffers from as it is explained in section (4.2.2.2) ;when one controller of the DSDN fails to manage/control its domain this led not to isolated this domain from the network where the manage/control this domain to the other control in the DSDN. The second one is to use blockchain technology, which operates on a distributed network and hence supports redundancy. Taking advantage of the distributed nature of blockchain networks, which consist of a number of nodes, each node in the network has its own copy of the blockchain ledger.

Scenario 3: Assume the scenario where the suggested system is vulnerable to a DoS/DDOS attack at the controller layer. Denial of Service describes an effort to prevent a resource from being used DoS/DDOS; attack involves sending an overwhelming number of bogus requests to a target in an effort to overload it and render it

unavailable to real users. Let's say an assault or overload has crippled one of the nodes controlling the SDN. Our system would keep running as usual, with additional SDN controllers taking care of incoming requests. Consequently, our architecture is resilient to a typical attack vector against a centralized SDN controller SPoF. Distributing a flow rule simultaneously with DSDN's controllers through blockchain, all of which share the same data storage and processing power which is crucial for optimizing network performance and reducing response times .

Table 4.2: Security Paradigm (CIA) Triangle with Proposed System

Security Objectives	Encryption Technique	Authentication Technique	Blockchain Technology	D SDN Network
Confidentiality	√	√		
Integrity			√	
Availability			√	√

Chapter Five

The Conclusions and Suggested Future Works

5.1 Overview

Regardless of the fact that SDN is a promising technology with a bright future. The SDN controller is a tempting target for attackers that wants to disable the network entirely ,as compared to conventional networks, SDNs are more vulnerable to intrusion because of their centralized nature, and the remainder of this statement may be written off as irrelevant. The Result Of Performance Evaluation Comparison Between Central SDN network and DSDN show that the dropped packet of (SPoF) in central SDN was 100% while the percentage of **Dropped Packet** decreases to 12% ,and 22% as one controller ,two controllers respectively fail in DSDN.. The proposed system showed its efficiency in overcoming the communication overhead and leader-follower /master-slave umbrella for enhancement its performance. The Throughput of distributed opendaylight controllers is 45.4 ratio greater than the centralized controller, while the centralized opendaylight controller has a 3.3% greater Latency than distributed SDN controllers ;where each of Throughput and Latency are SDN network Scalability metrics. A blockchain play an essential role in supporting compatibility and consistency between DSDN controls , help a system manger remote control to the proposed system, and provide a security.

5.2 Conclusions

- The proposed testbed ; virtual environment , for a SDN network is very appropriate and useful.

- The proposed system reduces the CPU use of leader controller , where the network burden was distributed to all controls equally.
- By allowing one controller to take over for another in the event of a loss, distributed SDN controllers show promise for addressing the single-point failure characteristic of SDN controllers and distributing network load more evenly.
- The consistency of the controller's rule set, ensuring trough using the blockchain enables flow rule as consensus mechanism to update the flow rules by the network management and simultaneously broadcast the updated rules to all the controllers .
- Evaluation of **Topology Discover** shows that there are a Progressive relationship between the time of Topology Discover and no .of nodes in data plane.
- The proposed blockchain-based distributed SDN decreases the **Write Time Overhead** by only about 141ms.
- The well-known information security paradigm is known as the **Confidentiality-Integrity-Availability (CIA)** triangle Prove that how well our system secure.

5.3 Limitation

- Memory consumption due to swapping copies network information among the controller in DSDN network.
- CPU consumption due to swapping Controller to Controller traffic in DSDN network.
- Using the ODL controller as SDN controller in the proposed System as it is very difficult to update and build new model in it.

5.4 Future Works

- Developing an application ledger with blockchain for each shards of network data shards, topology, inventory,..., .where every second the application works on checking its controller for updating in its shards .So, if there are update, they make a block for shards and distributed among other controller in the system in order to guarantee strong consistency in addition to minimize the load on leader controller
- An important feature is placement of controller in the network , this feature can be studied to detect the best placement of controller that works on enhancement the performance of the network.
- Employ the Quantum computing to enhancement the security of the proposed system.
- Study the possibility use only User Datagram Protocol (UDP) in communication between control plane and data plane ,and how can effect on latency and delay.
- Using a key encapsulation mechanism (KEM) ,Encryption the transaction between blockchain and SDN's controller and studying the Encryption effect on the performance of the SDN network.

References

[1] Z. A. El Houda, L. Khoukhi, and A. Hafid, "ChainSecure-a scalable and proactive solution for protecting blockchain applications using SDN,"in Proc. *IEEE Global Communication. Conf. (GLOBECOM)*, pp. 1–6, Dec. 2018.

[2] Panda, A., Zheng, W., Hu, X., Krishnamurthy, A., & Shenker, S. {SCL}: Simplifying Distributed {SDN} Control Planes. In *14th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 17)*, pp. 329-345, 2017.

[3] Ahmad, S., Mir, A.H, *Scalability, Consistency, Reliability and Security in SDN Controllers: A Survey of Diverse SDN Controllers*. J Netw Syst Manage, vol. ***29,*** *9*, 2021. https://doi.org/10.1007/s10922-020-09575-4

[4] Midha, S., & Tripathi, K. Extended Security in Heterogeneous Distributed SDN Architecture. *In Advances in Communication and Computational Technology*, pp. 991-1002, 2021.

[5] Zheng, Zibin, Et Al. "An Overview of Block-Chain Technology: Architecture, Consensus, And Future Trends." Big Data (Bigdata Congress), 2017 IEEE International Congress On. IEEE, 2017.

[6] Zhang, G., Li, T., Li, Y. et al. Blockchain- Based Data Sharing System for AI-Powered Network Operations. J. Commun. Inf. Netw. 3, 1–8, 2018. https://doi.org/10.1007/s41650-018- 0024-3

[7] Jiang, Xin & Liu, Mingzhe & Yang, Chen & Liu, Yanhua & Wang, Ruili. (2019). A Blockchain-Based Authentication Protocol for WLAN Mesh Security Access. Computers,904 Materials & Continua. 58. 45 59. 10.32604/cmc.2019.03863.

[8] Ren, Y., Leng, Y., Zhu, F., Wang, J., & Kim, H. J. Data Storage Mechanism Based on Blockchain with Privacy Protection in Wireless Body Area Network. Sensors (Basel, Switzerland), 19(10), pp. 2395, 2019. https://doi.org/10.3390/s19102395

[9] D. B. Rawat and A. Alshaikhi, "Leveraging Distributed Blockchain-based Scheme for Wireless Network Virtualization with Security and QoS Constraints," 2018 International Conference on Computing, Networking and Communications (ICNC), Maui, HI, pp. 332-336, 2018, doi: 10.1109/ICCNC.2018.8390344.

[10] Lin, Di, and Yu Tang. "Blockchain consensus based user access strategies in D2D networks for data-intensive applications." IEEE Access 6 : 72683-72690, 2018.

[11] Z. Liu, D. Wang, J. Wang, X. Wang and H. Li, "A Blockchain-Enabled Secure Power Trading Mechanism for Smart Grid Employing Wireless Networks," in IEEE Access, vol. 8, pp. 177745-177756, 2020, doi: 10.1109/ACCESS.2020.3027192.

[12] Moinet, Axel, Benoît Darties, and Jean-Luc Baril. "Blockchain based trust & authentication for decentralized sensor networks." arXiv preprint arXiv:1706.01730, 2017.

[13] Ren, Yongjun, et al. "Incentive mechanism of data storage based on blockchain for wireless sensor networks." *Mobile Information Systems*, 2018.
[14] She, Wei, et al. "Blockchain trust model for malicious node detection in wireless sensor networks." IEEE Access, vol. 7, pp. 38947-38956, 2019.
[15] Kim, Tai-Hoon, et al. "A novel trust evaluation process for secure localization using a decentralized blockchain in wireless sensor networks." IEEE Access, vol. 7, pp. 184133- 184144, 2019.
[16] Rathee, Geetanjali, et al. "A blockchain framework for securing connected and autonomous vehicles." *Sensors*, vol. 19(14), pp. 3165, 2019.
[17] X. Zhang and X. Chen, "Data Security Sharing and Storage Based on a Consortium Blockchain in a Vehicular Ad-hoc Network," *in IEEE Access*, vol. 7, pp. 58241-58254, 2019, doi: 10.1109/ACCESS.2018.2890736.
[18] C. Lin, D. He, X. Huang, N. Kumar, and K. R. Choo, "BCPPA: A Blockchain-Based Conditional Privacy-Preserving Authentication Protocol for Vehicular Ad Hoc Networks," *in IEEE Transactions on Intelligent Transportation Systems*, doi: 10.1109/TITS.2020.3002096.
[19] S. Boukria, M. Guerroumi and I. Romdhani, "BCFR: Blockchain-based Controller Against False Flow Rule Injection in SDN," *2019 IEEE Symposium on Computers and Communications (ISCC)*, Barcelona, Spain, pp. 1034-1039, 2019, doi: 10.1109/ISCC47284.2019.8969780.
[20] Jimenez, M.B.; Fernandez, D.; Rivadeneira, J.E.; Bellido, L.; Cardenas, A. A Survey of the Main Security Issues and Solutions for the SDN Architecture. IEEE Access, vol. 9, pp. 122016–122038, 2021. [CrossRef]
[21] Rojas, E., Amin, R., Guerrero, C., Savi, M., & Rastegarnia, A. Challenges and Solutions for hybrid SDN. *Computer Networks*, 2021.
[22] Keerthana, B., Balachandra, M., Hebbar, H., & Muniyal, B. Performance Comparison of Various Controllers in Different SDN Topologies. *In Expert Clouds and Applications*, pp. 297-309, 2022.
[23] Kreutz, D.; Ramos, F.M.; Verissimo, P.E.; Rothenberg, C.E.; Azodolmolky, S.; Uhlig, S. Software-defined networking: A comprehensive survey. Proc. IEEE 2014, 103, 14–76. [CrossRef]
[24] Suhail Ahmad and Ajaz Hussain Mir,"Scalability, Consistency, Reliability and Security in SDN Controllers: A Survey of Diverse SDN Controllers",J. Netw. Syst. Manage. Vol. 29(1), Jan 2021. https://doi.org/10.1007/s10922-020-09575-4
[25] M. N. Nxumalo, I. Mba and M. O. Adigun, "Comparative study for control plane scalability approaches in SDN productive networks," 2017 International Conference on Engineering and Technology (ICET), Antalya, Turkey, 2017, pp. 1-6, doi: 10.1109/ICEngTechnol.2017.8308175..
[26] B. Isong, R. R. S. Molose, A. M. Abu-Mahfouz and N. Dladlu, "Comprehensive Review of SDN Controller Placement Strategies," in IEEE Access, vol. 8, pp. 170070-170092, 2020, doi: 10.1109/ACCESS.2020.3023974..
[27] Zhihao Zeng, Xiaoning Zhang, Zixiang Xia," Intelligent Blockchain-Based Secure Routing for Multidomain SDN-Enabled IoT Networks", Wireless Communications and Mobile Computing, 2022.doi:10.1155/2022/5693962.

[28] Yuan Zhang, Lin Cui, Wei Wang, and Yuxiang Zhang. A survey on software defined networking with multiple controllers. J. Netw. Comput. Appl. 103, C (February 2018), 101–118. https://doi.org/10.1016/j.jnca.2017.11.015

[29] Raju, VR Sudarsana. "SDN controllers comparison." In Proceedings of science globe international conference, pp. 1-5. 2018..

[30] Zhu, Liehuang, Md M. Karim, Kashif Sharif, Chang Xu, Fan Li, Xiaojiang Du, and Mohsen Guizani. "SDN controllers: A comprehensive analysis and performance evaluation study." ACM Computing Surveys (CSUR) 53, no. 6 (2020): 1-40..

[31] Srirama, S.N., Dick, F.M.S., & Adhikari, M. Akka framework based on the Actor model for executing distributed Fog Computing applications. *Future Generation Computer Systems*, vol. *117,* pp. 439-452, 2021.

[32] Tran, H.M., Le, S.T., Le, H.D., & Tran, A.T. A Distributed Controller Approach Using P2P Protocol for Software Defined Networks. *In International Conference on Advanced Computing and Applications (ACOMP),* pp. 65-70, 2018.

[33] Amiri, E., Alizadeh, E., & Raeisi, K. An efficient hierarchical distributed SDN controller model. *In 5th Conference on Knowledge Based Engineering and Innovation (KBEI*), pp. 553-557, 2019.

[34] Liu, W., Wang, Y., Zhang, J. *et al.* AAMcon: an adaptively distributed SDN controller in data center networks. *Front. Comput. Sci.* vol. *14*, pp. 146–161, 2020. https://doi.org/10.1007/s11704-019-7266-6

[35] A. Dixit et al., "Towards An Elastic Distributed SDN Controller," in ACM SIGCOMM Computer Communication Review, vol. 43, no. 4. ACM, 2013, pp. 7–12.

[36] L. Mamushiane, A. Lysko and S. Dlamini, "A comparative evaluation of the performance of popular SDN controllers," 2018 Wireless Days (WD), pp. 54-59, 2018. doi: 10.1109/WD.2018.8361694.

[37] Zhu, Liehuang, Md Monjurul Karim, Kashif Sharif, Fan Li, Xiaojiang Du, and Mohsen Guizani. "SDN controllers: Benchmarking & performance evaluation." *arXiv preprint arXiv:1902.04491*, 2019.

[38] Badotra, Sumit, and Surya Narayan Panda. "Evaluation and comparison of OpenDayLight and open networking operating system in software-defined networking." *Cluster Computing*, vol. 23, no. 2, pp. 1281-1291, 2020.

[39] Mendoza, Daniel Haro, Luis Tello Oquendo, and Luis A. Marrone. "A comparative evaluation of the performance of open-source SDN controllers." Latin-American Journal of Computing, vol. 7, no. 2, pp. 64-77, 2020.

[40] MaheshiB.Dissanayake1,A.L.V.Kumari1 and U.K.A.Udunuwara PERFORMANCE COMPARISON OF ONOS AND ODL CONTROLLERS IN SOFTWARE DEFINED NETWORKS UNDER DIFFERENT NETWORK TYPOLOGIES, J. Res. Technol. Eng. vol. 2, no. 3, pp.94-105, 2021.

[41] T.BAH, M., Azzouni, A., Nguyen, M. T., & Pujolle, G. Topology Discovery Performance Evaluation of OpenDaylight and ONOS Controllers. 2019 22nd Conference on Innovation in Clouds, Internet and Networks and Workshops (ICIN), 2019. doi:10.1109/icin.2019.8685915.

[42] Wazirali, Raniyah, Rami Ahmad, and Suheib Alhiyari. "SDN-openflow topology discovery: an overview of performance issues." Applied Sciences, vol. 11, no. 15, 2021: 6999.

[43] Suad El-Geder, Performance Evaluation using Multiple Controllers with Different Flow Setup Modes in the Software Defined Network Architecture, Doctoral dissertation, Brunel University London, 2017.

[44] Z. Abou El Houda, A. S. Hafid and L. Khoukhi, "Cochain-SC: An Intra- and Inter-Domain Ddos Mitigation Scheme Based on Blockchain Using SDN and Smart Contract," in *IEEE Access*, vol. 7, pp. 98893-98907, 2019, doi: 10.1109/ACCESS.2019.2930715.

[45] H. Yang, Y. Liang, Q. Yao, S. Guo, A. Yu, and J. Zhang, "Blockchain based secure distributed control for software defined optical networking," China Communications, vol. 16, no. 6, pp. 42–54, 2019.

[46] Jiasi, Weng & Jian, Weng & Jia-Nan, Liu & Zhang, Yue. Secure Software-Defined Networking Based on Blockchain, 2019.

[47] M. Singh, G. S. Aujla, A. Singh, N. Kumar and S. Garg, "Deep-Learning-Based Blockchain Framework for Secure Software-Defined Industrial Networks," in *IEEE Transactions on Industrial Informatics*, vol. 17, no. 1, pp. 606-616, Jan. 2021, doi: 10.1109/TII.2020.2968946.

[48] Shashidhara R, Ahuja N, Lajuvanthi M, Akhila S, Das AK, Rodrigues JJPC. SDN-chain: Privacy-preserving protocol for software defined networks using blockchain. *Security and Privacy*. 4(6): e178, 2012. https://doi.org/10.1002/spy2.178

[49] Xiong, Ao, et al. "A distributed security SDN cluster architecture for smart grid based on blockchain technology." *Security and Communication Networks* 2021 (2021).

[50] Weichen Lian, Zhaobin Li, Chao Guo, Zhanzhen Wei, and Xingyuan Peng. 2021. FRChain: A Blockchain-based Flow-Rules-oriented Data Forwarding Security Scheme in SDN. *KSII Transactions on Internet and Information Systems*, vol. 15(1), pp. 264-284, 2021. DOI: 10.3837/tiis.2021.01.015.

[51] Zheng, Zibin, Et Al. "An Overview of Block-Chain Technology: Architecture, Consensus, And Future Trends." Big Data (Bigdata Congress), 2017 Ieee International Congress On. Ieee, 2017.

[52] C. Di Martino, U. Giordano, N. Mohanasamy, S. Russo, and M. Thottan,"In production performance testing of SDN control plane for telecomoperators," in *Proc. IEEE/IFIP Int. Conf. Depend. Syst. Netw.*, pp. 642–653, 2018.

[53] Henni, D.E., Hadjaj-Aoul, Y., Ghomari, A.: Probe-SDN: A Smart Monitoring Framework for SDN-Based Networks. In: The proceedings of the Global Information Infrastructure and Networking Symposium, France, pp. 1–6, 2017.

[54] Tank, G.P., Dixit, A., Vellanki, A., & Annapurna, D. "Software-Defined Networking-The New Norm for Networks",2012..

[55] L. Peterson et al., "Central Office Re-Architected as a Data Center," IEEE Communications Magazine, vol. 54, no. 10, pp. 96–101, 2016.

[56] M. Chiosi et al., "Network Functions Virtualisation: An Introduction, Benefits, Enablers, Challenges and Call for Action," in SDN and OpenFlow World Congress, pp. 22–24, 2012.

[57] J. G. Herrera and J. F. Botero, “Resource Allocation in NFV: A Comprehensive Survey,” IEEE Transactions on Network and Service Management, vol. 13, no. 3, pp. 518–532, 2016.
[58] A. Al-Shabibi and L. Peterson, “CORD: Central Office Re-architected as a Datacenter,” OpenStack Summit, 2015.
[59] P. Pate, “NFV and SDN: What â A˘ Z s the Difference?” SdxCental, ´ March 2013. [Online]. Available: https://www.sdxcentral.com/articles/ contributed/nfv-and-sdn-whats-the-difference/2013/03/
[60] Y. Li and M. Chen, “Software-Defined Network Function Virtualization: A Survey,” IEEE Access, vol. 3, pp. 2542–2553, 2015.
[61] J. H. Cox et al., "Advancing Software-Defined Networks: A Survey," in IEEE Access, vol. 5, pp. 25487-25526, 2017, doi: 10.1109/ACCESS.2017.2762291.
[62] Nehra, A.; Tripathi, M.; Gaur, M.S.; Battula, R.B.; Lal, C. SLDP: A secure and lightweight link discovery protocol for software defined networking. Comput. Netw., vol. 150, pp. 102–116, 2019.
[63] McKeown, N., Anderson, T., Balakrishnan, H., Parulkar, G., et al.: OpenFlow: Enabling Innovation in Campus Networks, In: SIGCOMM Computer Communication Review, vol. 38, no. 2, pp. 69–74, Mar. 2008.
[64] Limoncelli, T. A.: Openflow: a radical new idea in networking, In: ACM Queue, vol. 10, no. 6, 2012.
[65] Goransson, P., Black, C., & Culver, T. *Software defined networks: a comprehensive approach.* Morgan Kaufmann, 2016.
[66] T. Koponen, “Onix: A Distributed Control Platform for Large-scale Production Networks,” in OSDI, vol. 10, pp. 1–6, 2010.
[67] P. Berde et al., “ONOS: Towards an Open, Distributed SDN OS,” in Proceedings of the Third Workshop on Hot Topics in Software Defined Networking. ACM, pp. 1–6, 2014.
[68] L. Schiff and S. Schmid, “Study The Past If You Would Define The Future: Implementing Secure Multi-Party SDN Updates,” in Software Science, Technology and Engineering (SWSTE), 2016 IEEE International Conference on. IEEE, pp. 111–116, 2016.
[69] Murat Karakus, Arjan Durresi,"A survey: Control plane scalability issues and approaches in Software-Defined Networking (SDN)",Computer Networks, vol. 112, pp. 279-293, 2017 ISSN 1389-1286, https://doi.org/10.1016/j.comnet.2016.11.017
[70] T. Hu, Z. Guo, P. Yi, T. Baker and J. Lan, "Multi-controller Based Software-Defined Networking: A Survey," in IEEE Access, vol. 6, pp. 15980-15996, 2018, doi: 10.1109/ACCESS.2018.2814738.
[71] P. Isaia, L. Guan, Performance benchmarking of sdn experimental platforms, in: 2016 IEEE NetSoft Conference and Workshops (NetSoft), pp. 116–120, 2016. doi:10.1109/ NETSOFT.2016.7502456.
[72] B. J. van Asten, N. L. M. van Adrichem, F. A. Kuipers, Scalability and resilience of software-defined networking: An overview, CoRR abs/1408.6760. URL http://arxiv.org/abs/1408.6760
[73] R. Hanmer, L. Jagadeesan, V. B. Mendiratta, and H. Zhang, “Friend or foe: Strong consistency vs. overload in high-availability distributed systems and SDN,” in *Proc. IEEE Int. Symp. Softw. Rel. Eng.*, pp. 1–6, 2018.
[74] E. Sakic and W. Kellerer, “Response time and availability study of RAFT consensus in distributed SDN control plane,” *IEEE Trans. Netw. Service Manag.*, vol. 15, no. 1, pp. 304–318, Mar. 2018.

[75] E. Sakic and W. Kellerer, “Impact of adaptive consistency on distributed SDN applications: An empirical study,” *IEEE J. Sel. Areas Commun.*, vol. 36, no. 12, pp. 2702–2715, Dec. 2018.
[76] A. S. Muqaddas, P. Giaccone, A. Bianco, and G. Maier, “Inter-controller traffic to support consistency in ONOS clusters,” IEEE Trans. Netw. Service Manag., vol. 14, no. 4, pp. 1018–1031, Dec. 2017.
[77] A. Tootoonchian and Y. Ganjali, “Hyperflow: a distributed control plane for openflow,” *Proc. 2010 internet Netw.*, pp. 1–6, 2010.
[78] V. W. Protocol, “OpenFlow Switch Specification 1.5,” *Open Netw. Found. ONF*, vol. 0, pp. 1–205, 2013.
[79] Opendaylight, https://www.opendaylight.org/
[80] J. Bonér (Lightbend). *Akka*. Accessed: Apr. 5, 2019. [Online]. Available: https://akka.io/
[81] Akka Cluster. https://doc.akka.io/docs/akka/2.5/common/cluster.html
[82] Diego Ongaro and John Ousterhout. In Search of an Understandable Consensus Algorithm. In Proc. USENIX Annual Technical Conference, 2014.
[83] K. Choumas and T. Korakis, "When Raft Meets SDN: How to Elect a Leader over a Network," *2020 6th IEEE Conference on Network Softwarization (NetSoft)*, pp. 140-144, 2020, doi: 10.1109/NetSoft48620.2020.9165534.
[84] T. Kim, S. Choi, J. Myung, and C. Lim. Load balancing on distributed datastore in opendaylight SDN controller cluster. In Proc. IEEE NetSoft, 2017.
[85] Levin, D., Wundsam, A., Heller, B., Handigol, N., Feldmann, A.: Logically centralized? State Distribution Trade-offs in Software Defined Networks. In: Proceedings of the first workshop on Hot topics in Software Defined Networks, pp. 1–6, 2012.
[86] P. Vizarreta, K. Trivedi, V. Mendiratta, W. Kellerer and C. Mas-Machuca, "DASON: Dependability Assessment Framework for Imperfect Distributed SDN Implementations," in IEEE Transactions on Network and Service Management, vol. 17, no. 2, pp. 652-667, June 2020, doi: 10.1109/TNSM.2020.2973925
[87] Aslan, M., *& Matrawy, A. A Clustering-based Consistency Adaptation Strategy for Distributed SDN Controllers. 2018 4th IEEE Conference on Network Softwarization and Workshops (NetSoft), 441-448, 2018.*
[88] Ahram, Tareq, et al. "Blockchain technology innovations." Technology & Engineering Management Conference (TEMSCON), 2017 IEEE. IEEE, 2017.
[89] S., Sarmah. “Understanding Blockchain Technology.” Scientific & Academic Publishing, 2018. doi:10.5923/j.computer.20180802.0.
[90] Guru, D.; Perumal, S.; Varadarajan, V. Approaches towards Blockchain Innovation: A Survey and Future Directions. Electronics 2021, 10, 1219. [CrossRef]
[91] Dorri, A.; Kanhere, S.S.; Jurdak, R.; Gauravaram, P. LSB: A Lightweight Scalable Blockchain for IoT security and anonymity. J. Parallel Distrib. Comput. vol. 134, pp. 180–197, 2019. [CrossRef]
[92] T. Alharbi, "Deployment of Blockchain Technology in Software Defined Networks: A Survey," in IEEE Access, vol. 8, pp. 9146-9156, 2020, doi: 10.1109/ACCESS.2020.2964751.
[93] M. Swan. Blockchain: Blueprint for a new economy. O’Reilly Media, Inc., 2015.

[94] N. Bozic, G. Pujolle and S. Secci, "A tutorial on blockchain and applications to secure network control-planes," *2016 3rd Smart Cloud Networks & Systems (SCNS)*, pp. 1-8, 2016. doi: 10.1109/SCNS.2016.7870552

[95] Feng, Wei, Yafeng Li, Xuetao Yang, Zheng Yan and Liang Chen. "Blockchain Based Data Transmission Control for Tactical Data Link." *iSCI*, 2019.

[96] Nakamoto, Satoshi. "Bitcoin: A peer-to-peer electronic cash system." Decentralized Business Review, 2008: 21260.

[97] Nakamoto, Satoshi, and A. Bitcoin. "A peer-to-peer electronic cash system." Bitcoin.–URL: https://bitcoin. org/bitcoin. pdf 4, 2008: 2.

[98] Singh, H.J.; Hafid, A.S. Transaction confirmation time prediction in ethereum blockchain using machine learning. arXiv 2019, arXiv:1911.11592.

[99] F. PUB, "Secure hash standard (shs)," FIPS PUB 180, vol. 4, pp. 1–27, 2012.

[100] Casino, Fran, Thomas K. Dasaklis, and Constantinos Patsakis. "A systematic literature review of blockchain-based applications: Current status, classification and open issues." Telematics and informatics vol. 36, pp. 55-81, 2019.

[101] K. Christidis and M. DevetsikIoTis, "Blockchains and smart contracts for the internet of things," Ieee Access, vol. 4, pp. 2292–2303, 2016.

[102] Z. Zheng, S. Xie, H. Dai, X. Chen and H. Wang, "An Overview of Blockchain Technology: Architecture, Consensus, and Future Trends," 2017 IEEE International Congress on Big Data (BigData Congress), pp. 557-564, 2017. doi: 10.1109/BigDataCongress.2017.85.

[103] Li, Wenjuan, et al. "Towards blockchain-based software-defined networking: security challenges and solutions." *IEICE Transactions on Information and Systems* vol. 103, no. 2, pp. 196-203, 2020.

[104] J. Liu and Z. Liu, "A survey on security verification of blockchain smart contracts," IEEE Access, 2019.

[105] http://gns3.com

[106] "Supported host operating systems for Workstation Pro 12.x, 14.x and 15.x (2129859)". *VMware Knowledge Base. September 24, 2018. Retrieved January 26, 2019*

[107] "VMware Workstation 15.0.4 Pro Release Notes". *VMware.*Topo

[108] http://mininet.org/

[109] https://www.wireshark.org/

[110] Pakzad, F.; Portmann, M.; Tan,W.L.; Indulska, J. Efficient topology discovery in OpenFlow-based Software Defined Networks. Comput. Commun., vol. 77, pp. 52–61, 2016.

[110] Montero, R.; Agraz, F.; Pages, A.; Perello, J.; Spadaro, S. Dynamic topology discovery in SDN-enabled Transparent Optical Networks. In Proceedings of the 2017 International Conference on Optical Network Design and Modeling, Budapest, Hungary, vol. 15–18, pp. 1–6, May 2017.

[111] Zhao, X.; Yao, L.; Wu, G. ESLD: An efficient and secure link discovery scheme for software-defined networking. Int. J. Commun. Syst. 31, 2018, e3552.

[112] H. M.Noman ,An Improved Resource Allocation Approach for Software Defined Networks, College of Information Technology, University of Babylon, 2021.

[113] Algarni, Sultan, Fathy Eassa, Khalid Almarhabi, Abdullah Algarni, and Aiiad Albeshri. "BCNBI: A Blockchain-Based Security Framework for Northbound

Interface in Software-Defined Networking" *Electronics* vol. 11, no. 7, 996. 2022. https://doi.org/10.3390/electronics11070996.

Appendix A

Snapshot of Execution / Configuration of Proposed System

The figure(A.1) shows the design of the Distributed SDN in GNS3, using three Beryllium SR4 OpenDayLight controllers and the Python script to build a custom topology in mininet and the applications of the OpenDayLight controller. This topology remotely controlled by the ODL controller. All data networks make use of Open Flow switches (OVS switch 1.3) .

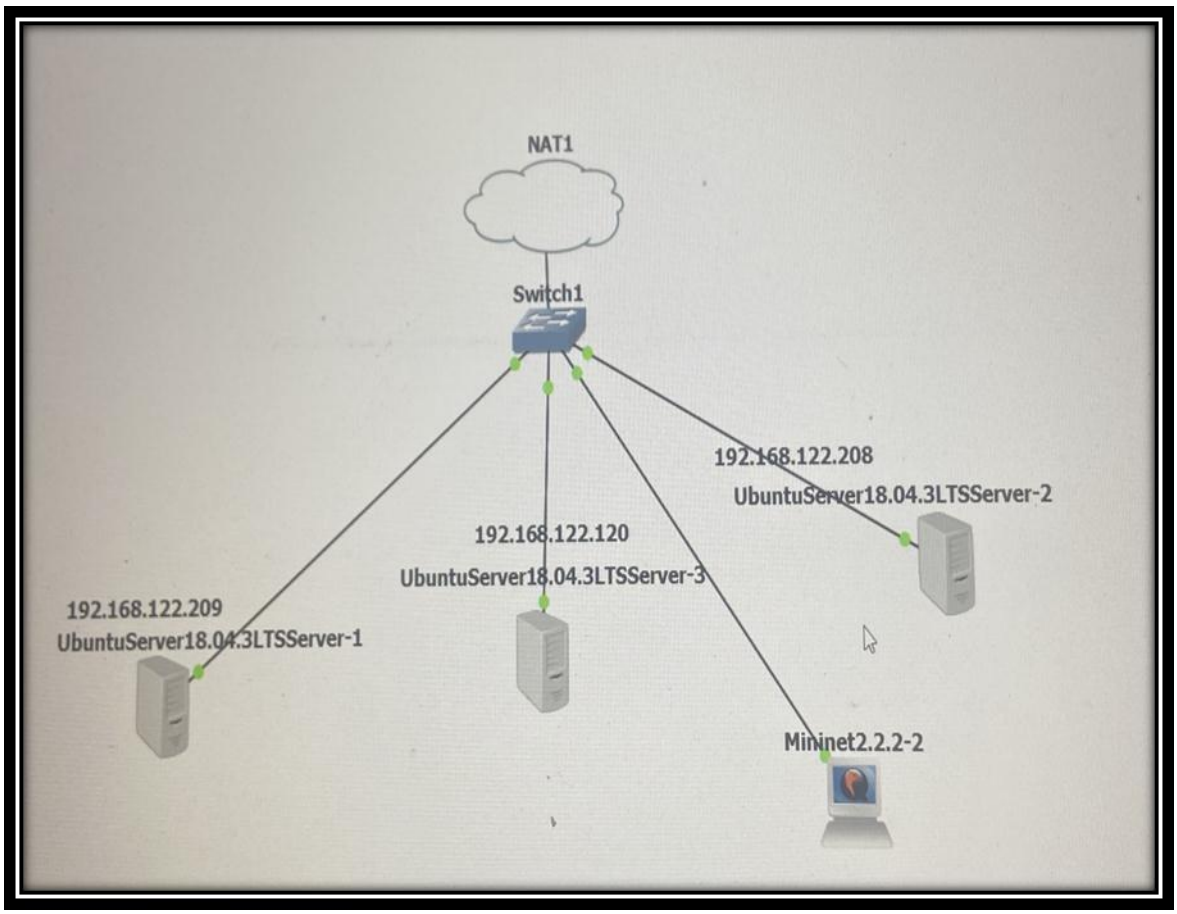


Figure A.1:The Design of the Distributed SDN Network in GNS3

The testing of the connection between the remote controllers and mininet by a custom topology that operated on the mininet as we can see in the Figure(A.2).

Figure A.2 The Custom Topology with Distributed SDN Network in GNS3

The figure (A.3) shows the reachability testing through the of use the Pingall Toole

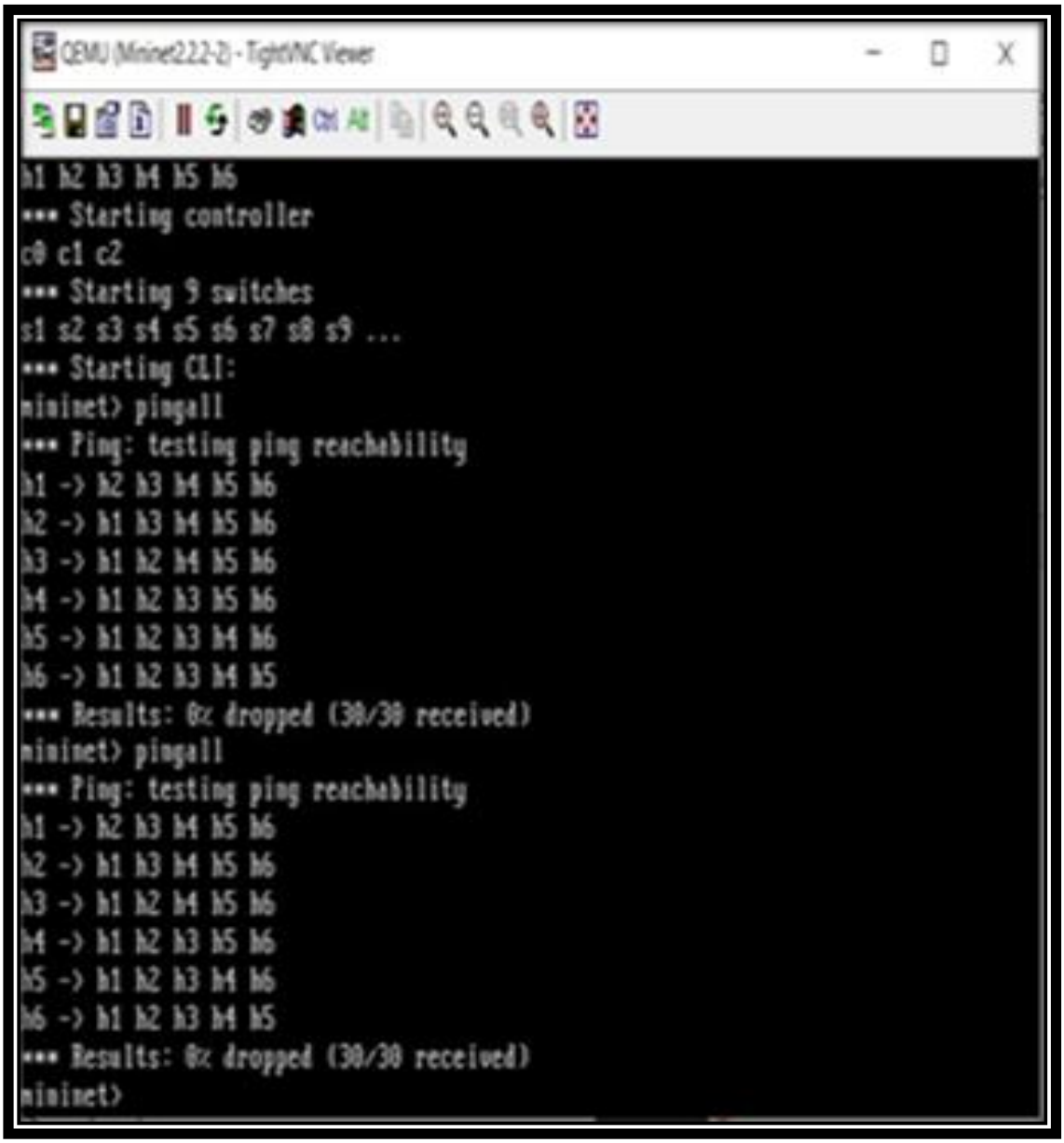


Figure A.3:The Reachability Testing

With opendaylight Dlux interface, the topology that controlled remotely by the opendaylight controller was shown , as can see in the Figure (A.4)

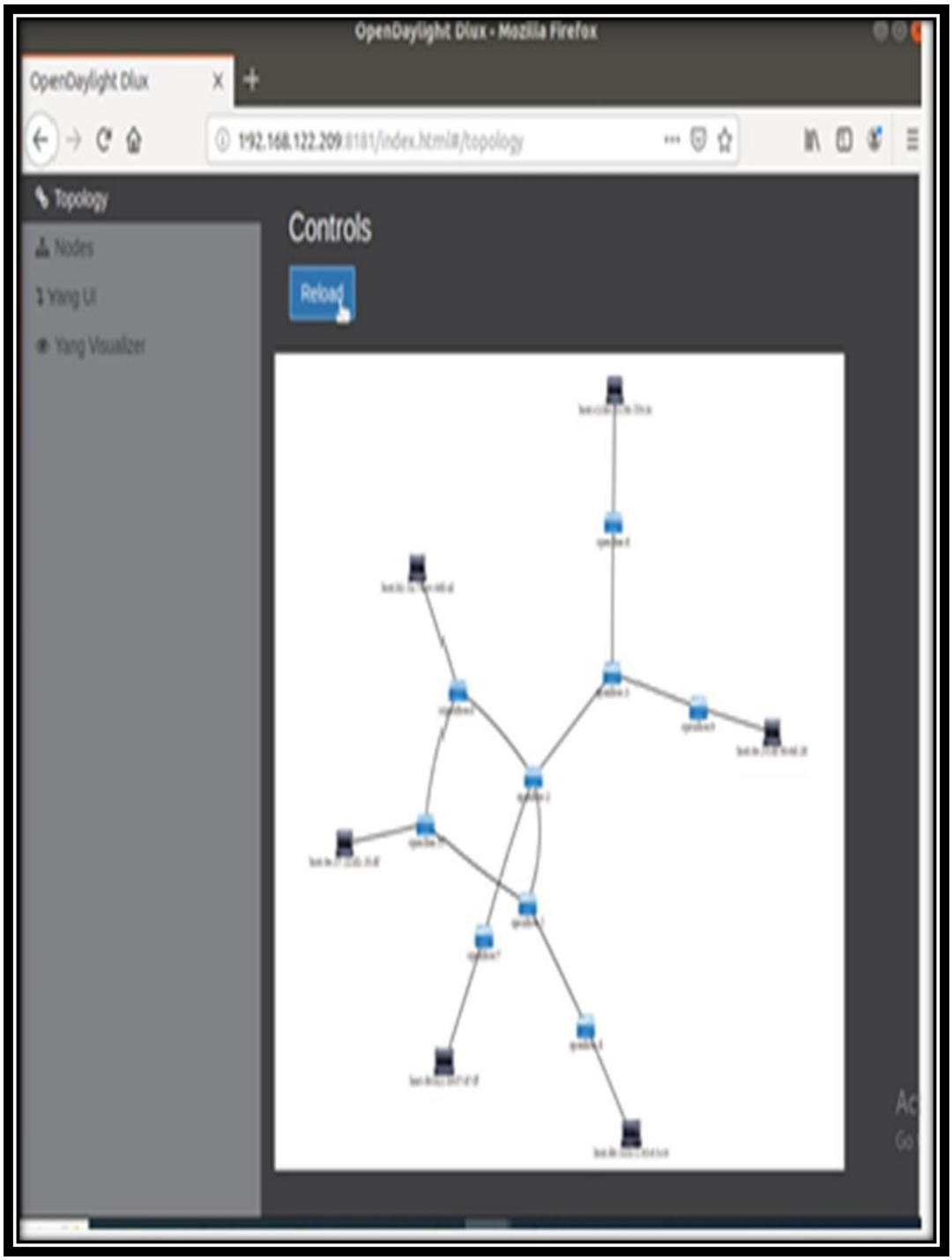


Figure A.4:The Connection Between the Remote Controllers and Custom Topology in Mininet in Dlux

the TCP communication that captures using wireshark tool, as Figure (A.5) shows , where it show the communication among the three SDN controllers in the proposed system to guarantee the inventory shard and topology shard consistency.

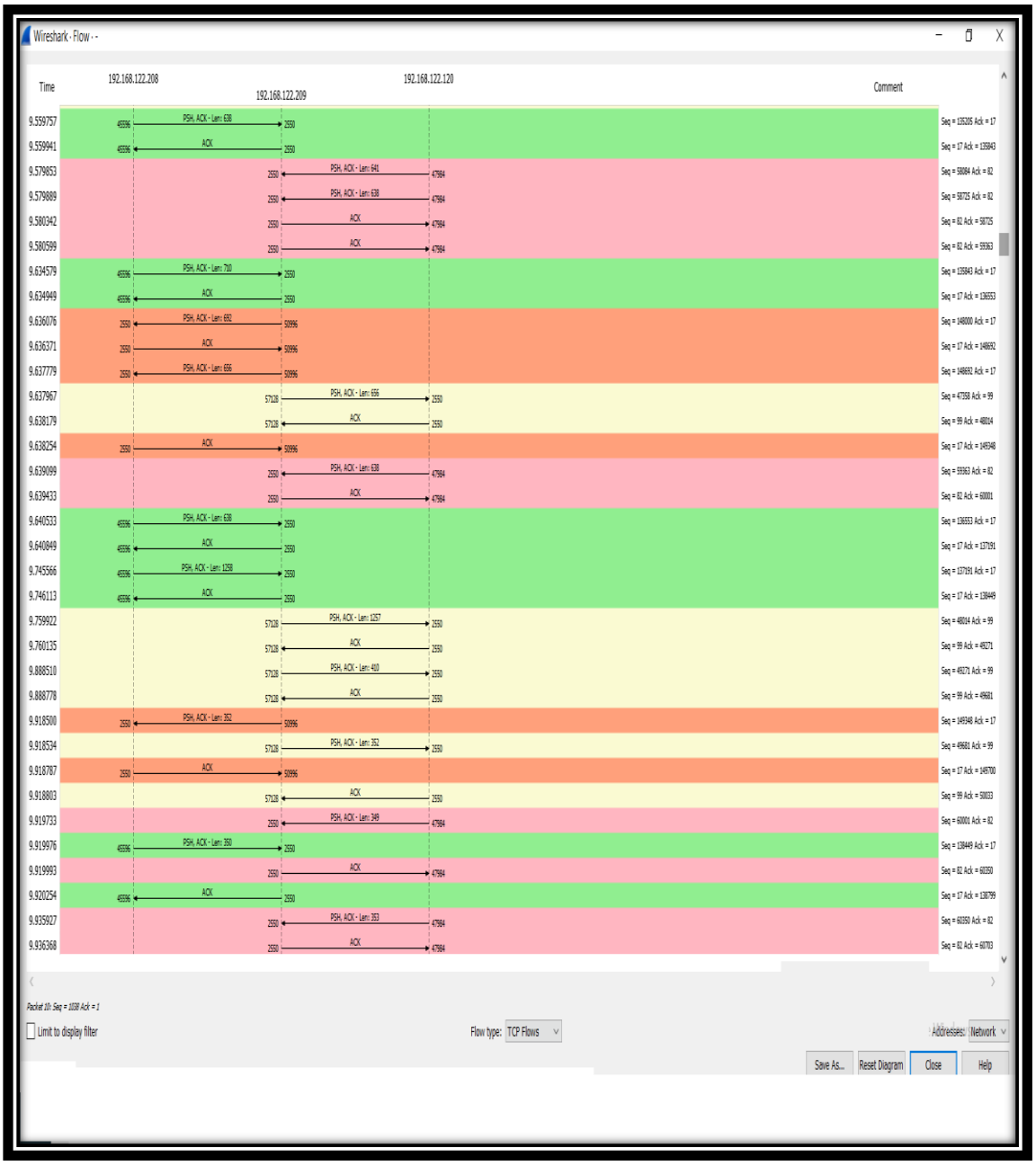


Figure A.5:TheTCP Communication among the Remote Controllers with Flow Capture by Wireshark

The virtual environment configuration of Flask application shown in Figures (A.6),(A.7)and (A.8).

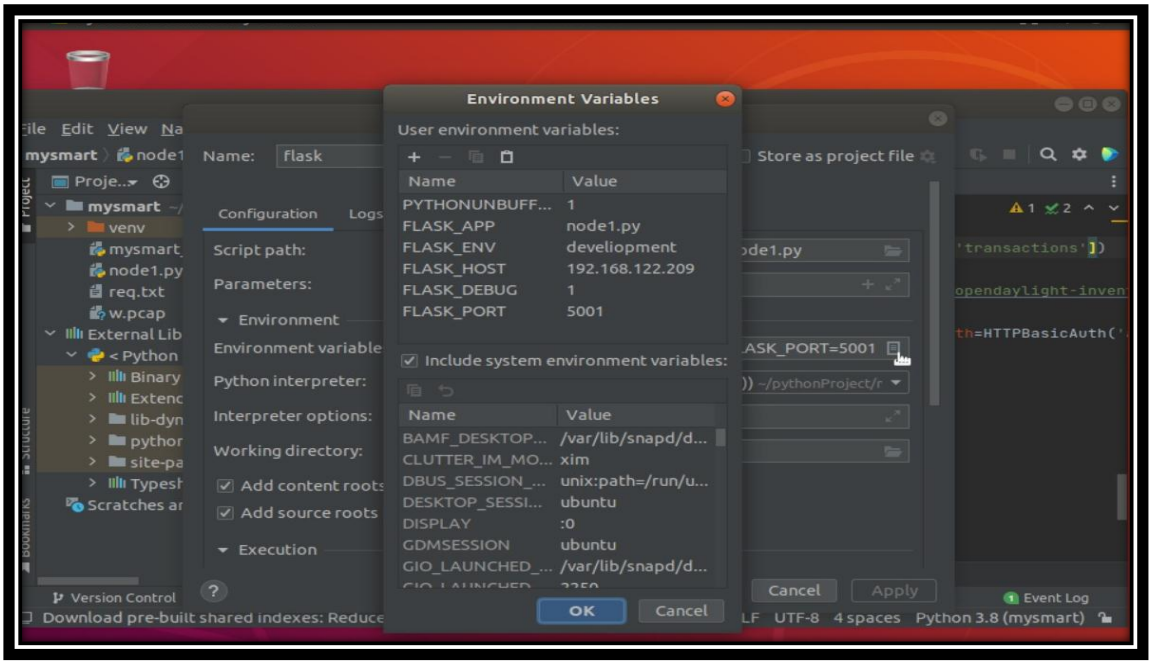


Figure A.6: The First Blockchain's Node of First Opendaylight Controller in DSDN

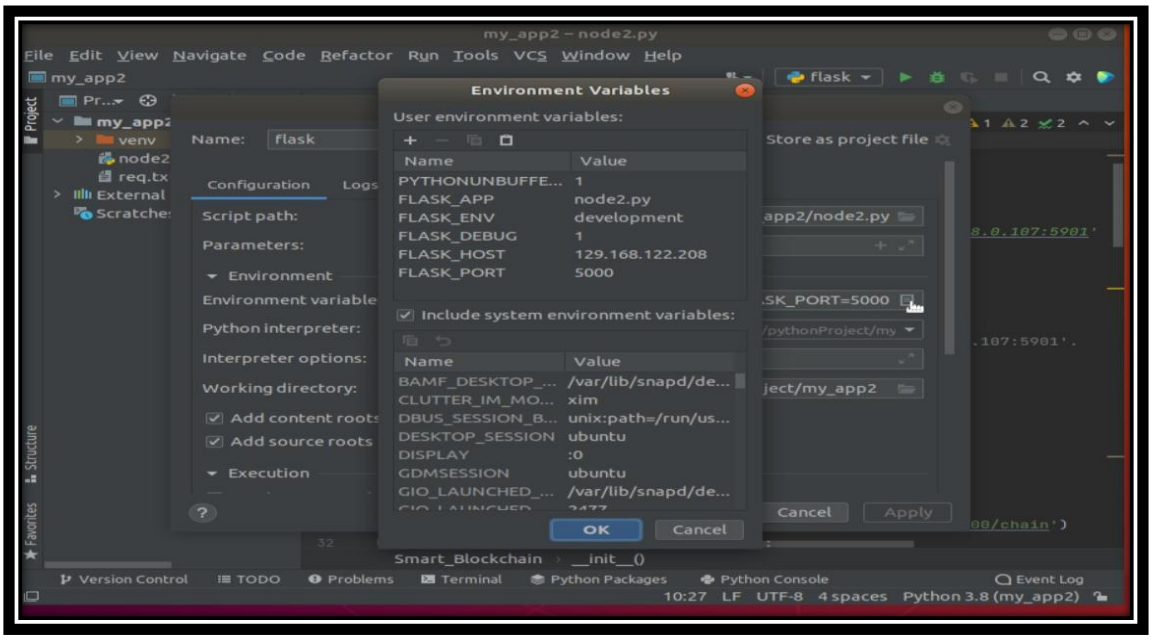


Figure A.7: The Second Blockchain's Node of Second Opendaylight Controller in DSDN

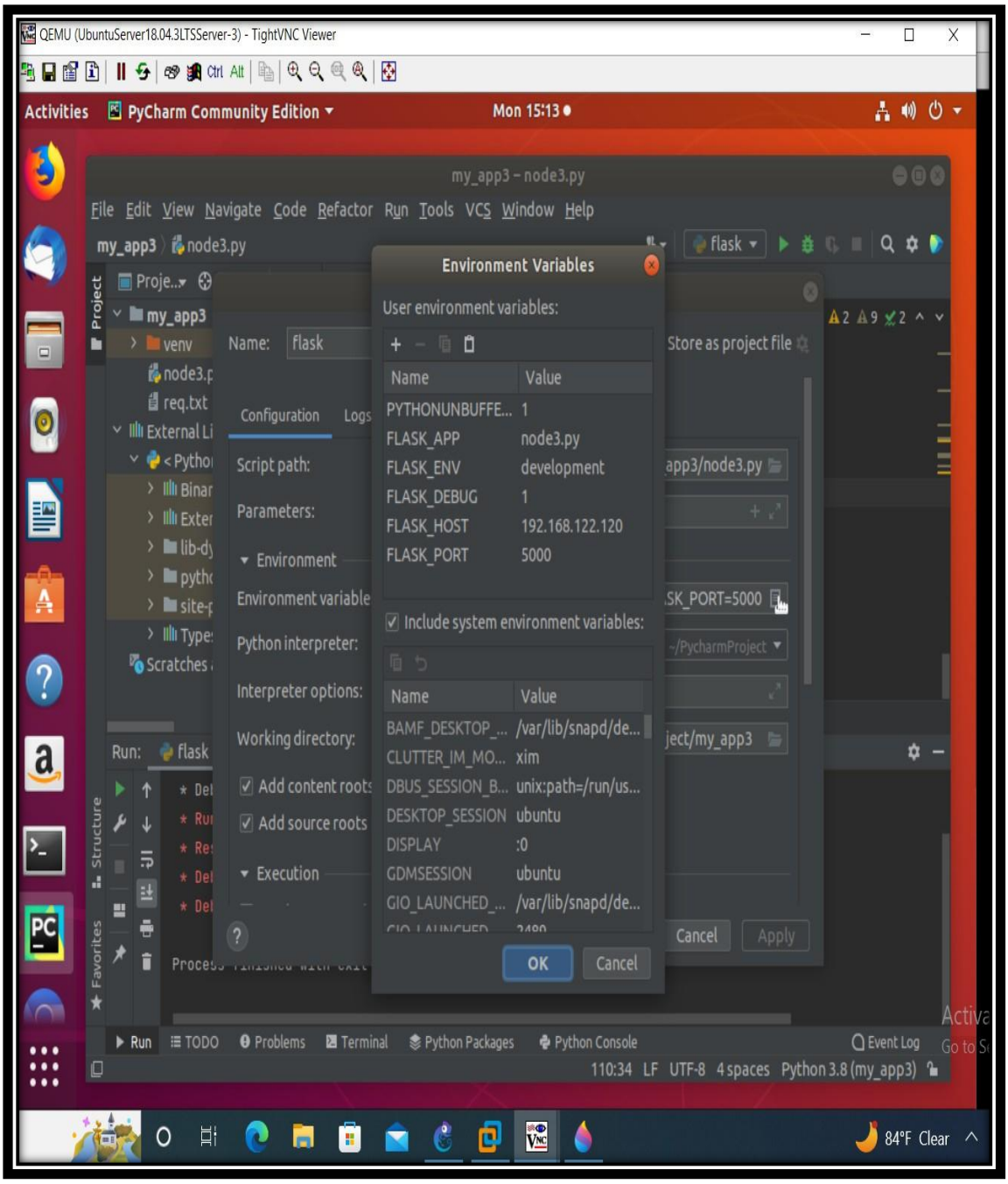


Figure A.8: The Third Blockchain's Node of Third Opendaylight Controller in DSDN

A simple JSON-based flow-rule is encoded as shown in Figure (A.9) which explain all the operation done inside the smart contact.

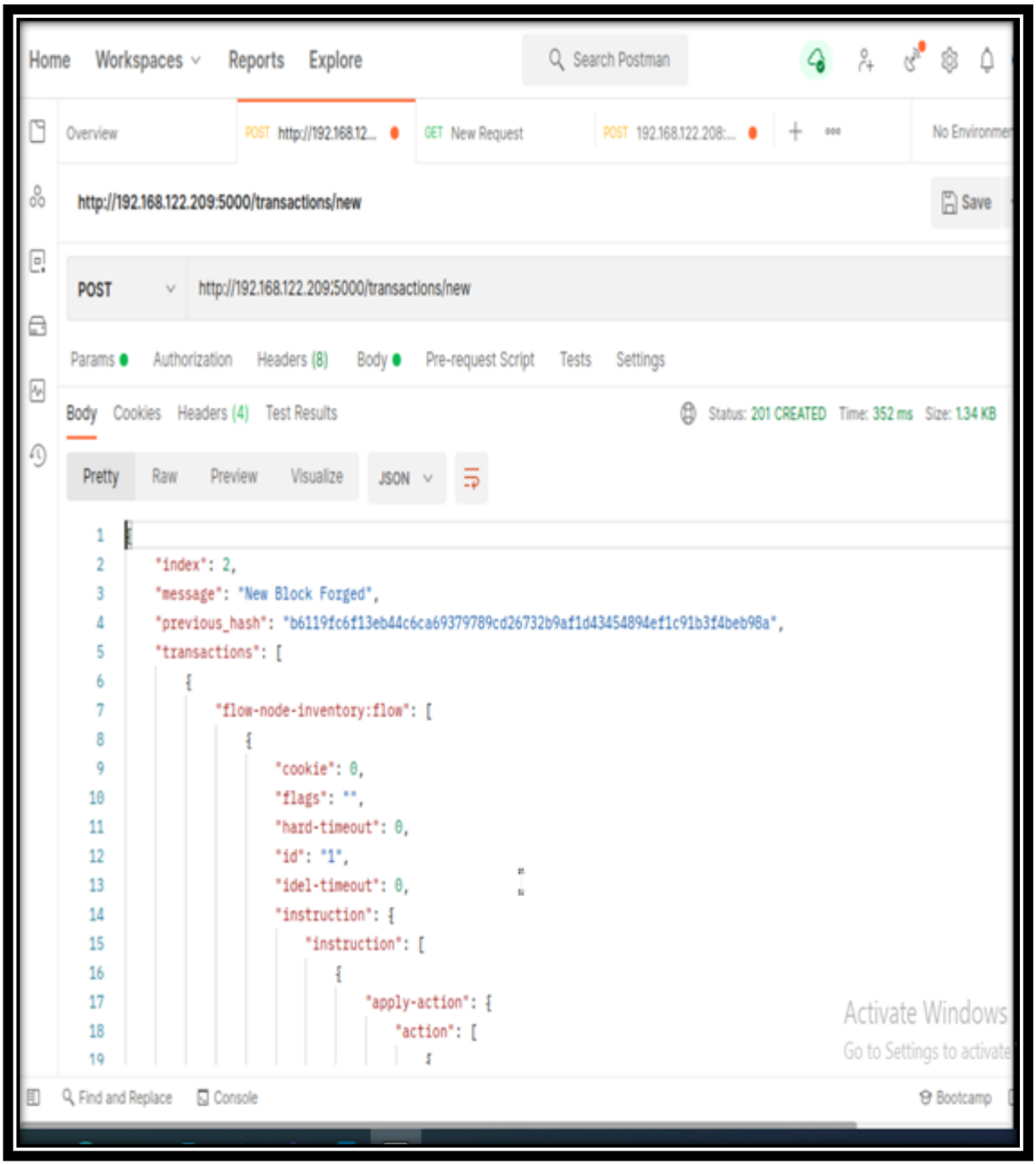


Figure A.9: The Smart Chain after Post Flow Rule by System Manger

The first operation that is done in smart contract shows in Figure (A.10)which represents in the registration of nodes of privet blockchain network in smart contract.

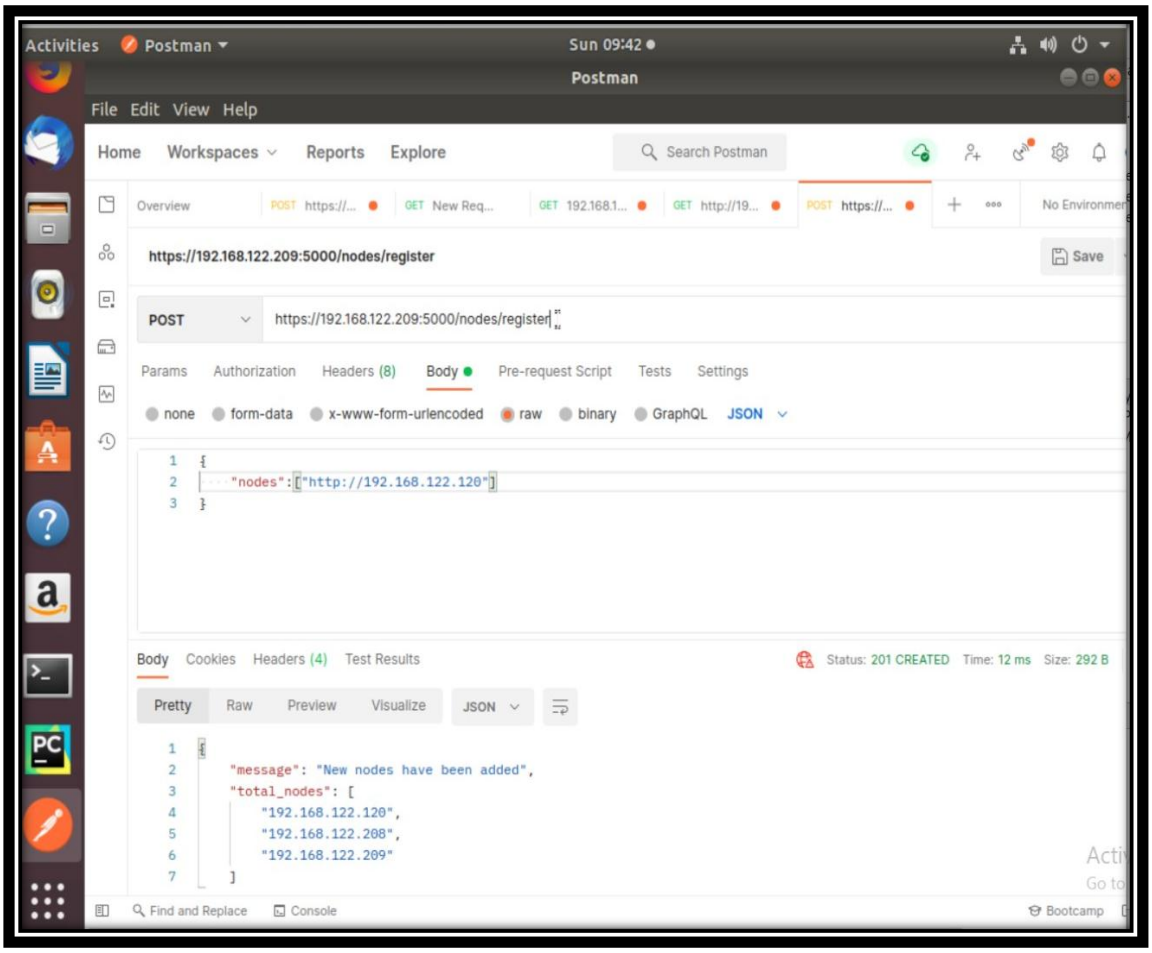


Figure A.10: The Registration Phase with the Smart Contract

The initial state of the smart chain where there are no flow rule send throw the smart contract shows in Figure(A.11) , where it represents the genesis block.

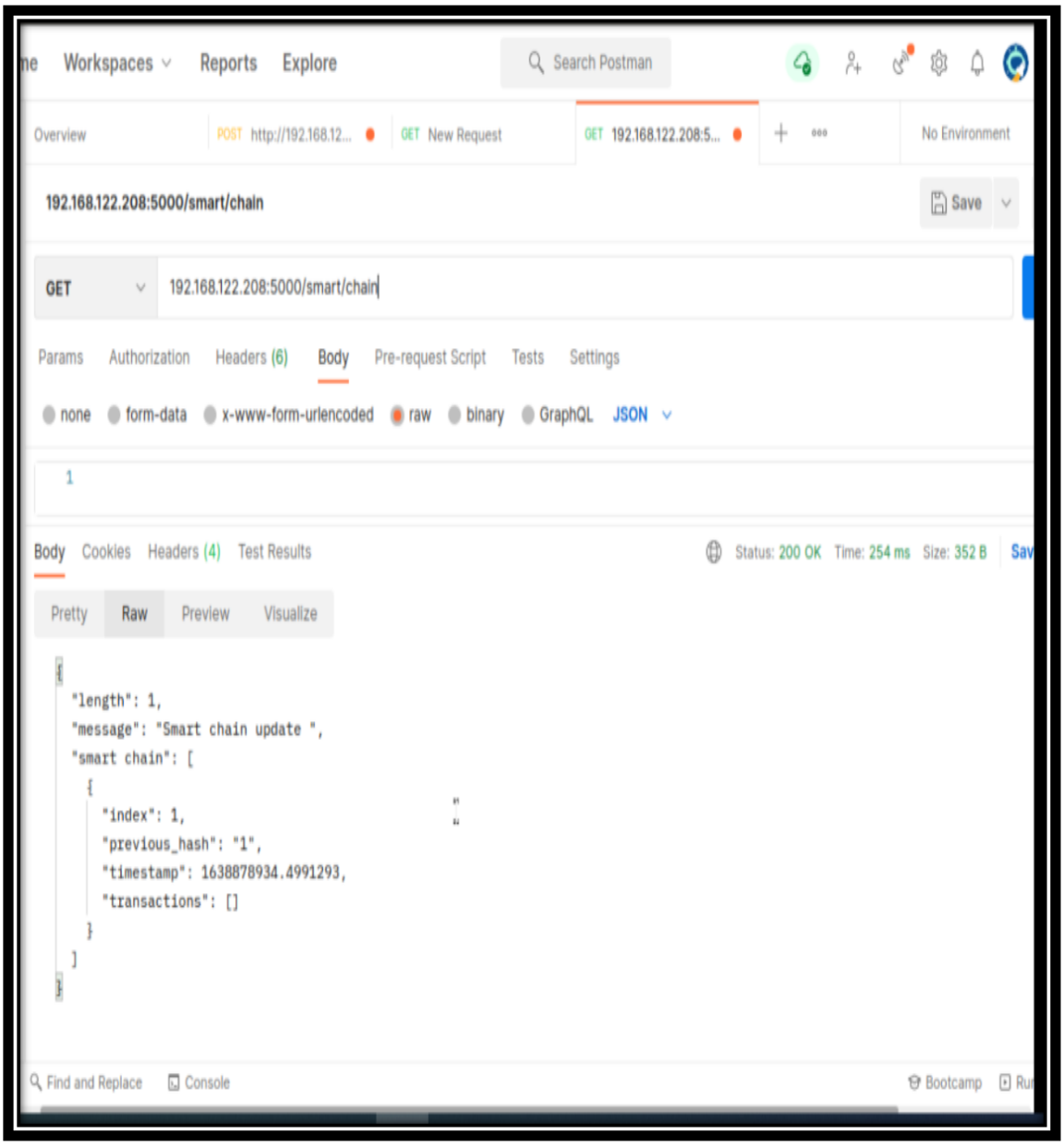


Figure A.11: The Initial State (genesis block)of the Blockchain

The application that connects with the controller with IP 192.168.122.120 receives the update blockchain and update the inventory shard of the controller as it shows in Figure(A.12).

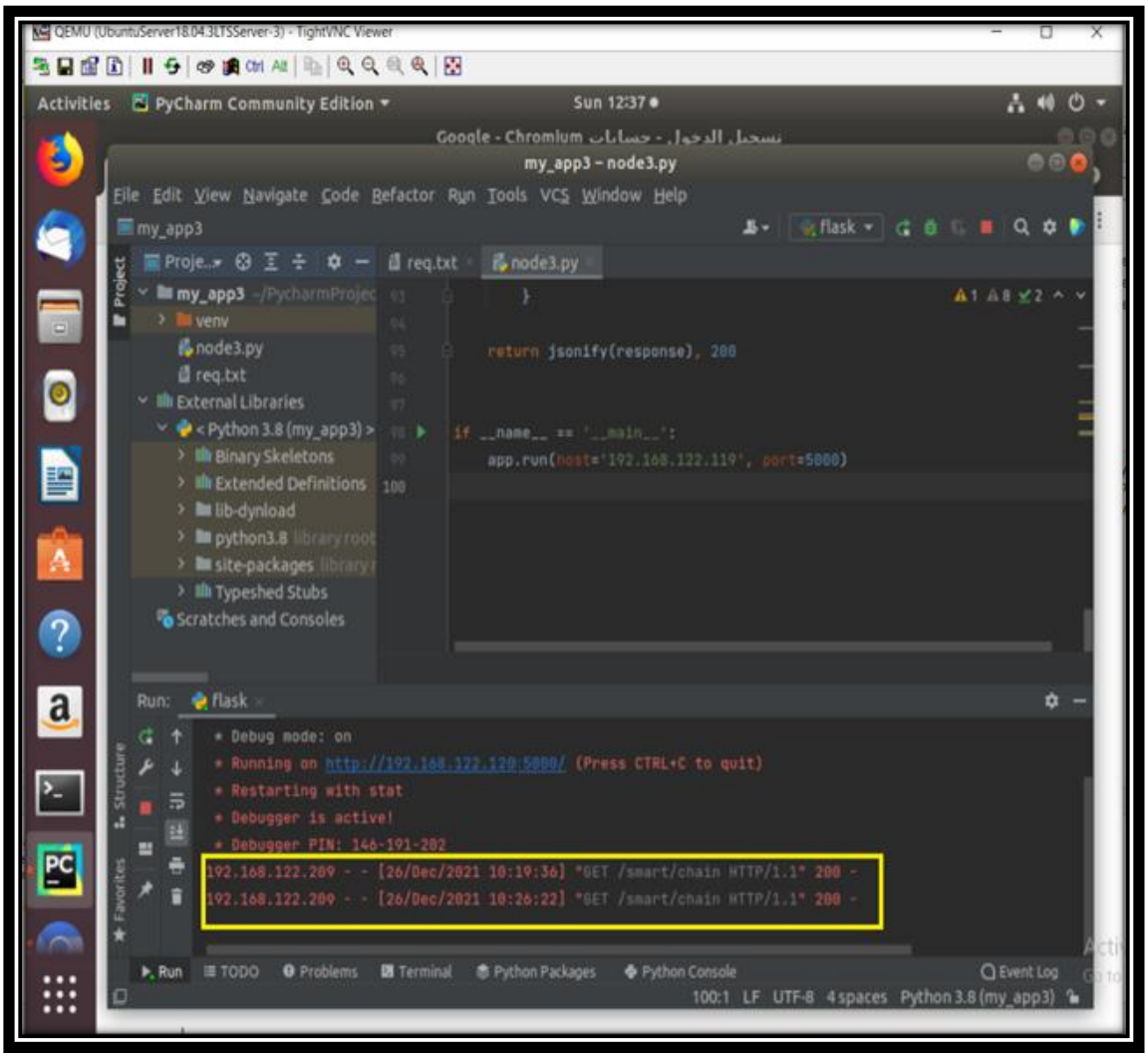


Figure A.12:The Application Node that Connect with The Controller

Using a Postman a POST request was submitting as transaction , flow rule, to the server address, which in turn sends the new rule to the smart contract as a transaction. There is a straightforward JSON-based flow-rule that it employs through its API as it shows in Figure (A.13).

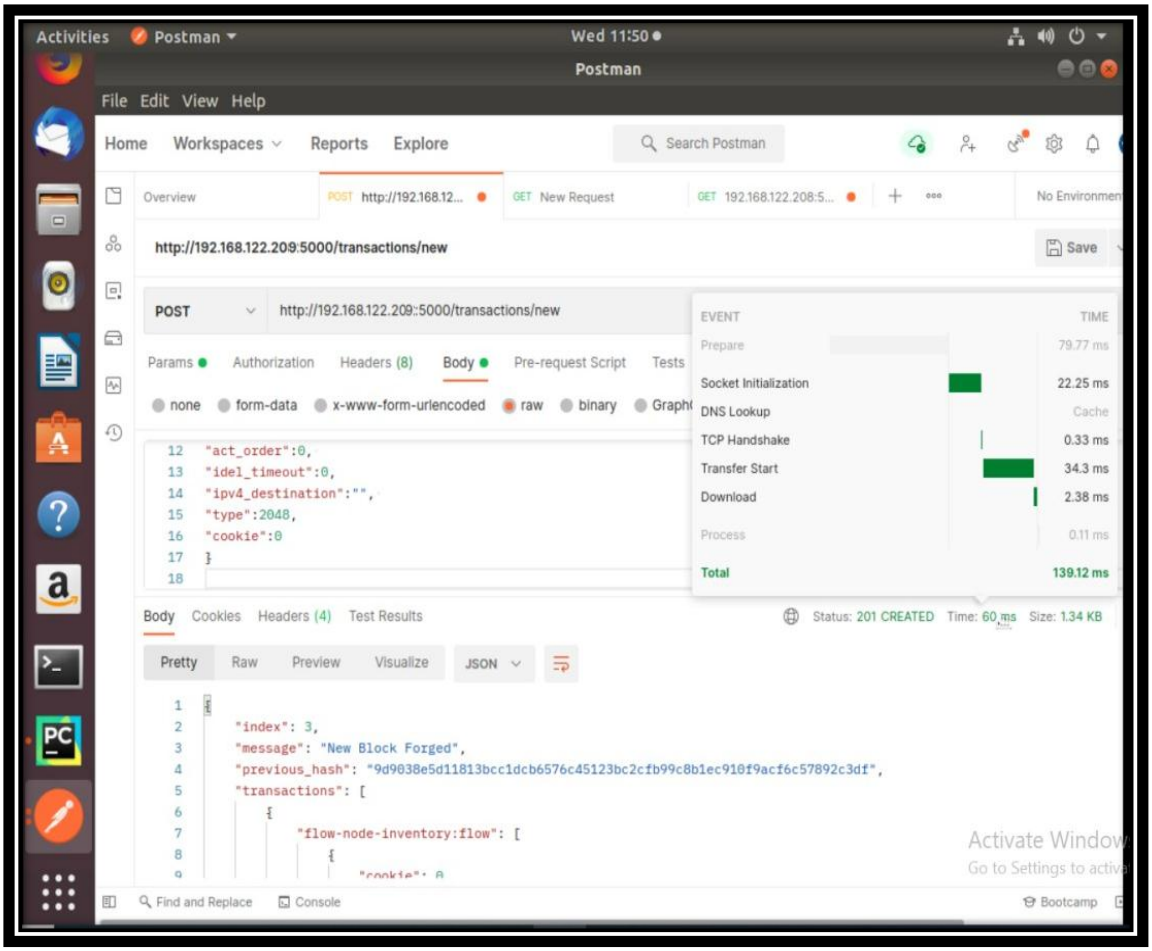


Figure A.13:the Enforcing a New Flow Rule on the DSDN Controller

الخلاصة

الشبكات المعرفة بالبرمجيات (SDN) هي تقنية متطورة لزيادة قابلية برمجة الشبكة. نظرًا لطبيعتها المركزية، فإن SDNعرضة بشكل خاص لنقطة واحدة من الفشل (SPoF) ومشكلات قابلية التوسع. تقدم SDN الموزعة (DSDN)للقضاء على نقطة الفشل الفردية ؛ مشكلة الأمان وقابلية التوسع الموجودة في وحدة تحكم SDN المركزية. يتمثل المفهوم الرئيسي في وجود العديد من وحدات التحكم التي يمكنها مشاركة العبء على الشبكة، ويمكن لوحدة تحكم واحدة تولي مهام وحدة تحكم أخرى عند تعطلها . بالإضافة إلى ذلك، يمكن أن يضمن استخدام تقنية blockchain أمان واتساق SDNويمهد الطريق لبنية SDN أكثر فاعلية وقابلية للتطوير.

في هذه الرسالة، تم اقتراح مزيج من نهج SDN وتقنية Blockchain للتغلب على نقاط الضعف في شبكة SDNحيث يكون النظام الموزع بالكامل المستند لوحدة التحكم Opendaylight كذلك معالجة مع مشكلة التفاوت في استهلاك وحدة المعالجة المركزية لوحدة التحكم. كل وحدة تحكم تتحكم في مجالها الخاص وبالتالي تشارك العبء عبر وحدات تحكم الشبكة. تم اقتراح عقد ذكي وآلية إجماع وبيانات موزعه للتطبيق لتحديث قواعد التدفق وبث قواعد جديدة لجميع وحدات التحكم في نفس الوقت، والمزامنة، مما يضمن اتساق مجموعة قواعد وحدة التحكم.

يتضمن النظام المقترح مرحلتان: المرحلة الأولى، اقتراح بيئة افتراضية كقاعدة اختبار تستخدم لبناء شبكة SDN الموزعة (DSDN) ، واقترحت المرحلة الثانية قواعد التدفق الممكّنة لـ Blockchain ؛ شبكة Blockchainالخاصة والعقد الذكي وآلية الإجماع وكيفية تكاملها مع شبكة SDN الموزعة.

تظهر نتيجة مقارنة تقييم الأداء بين شبكة SDN المركزية و DSDN أن dropped packet لـ (SPoF) في SDN المركزي كانت ١٠٠٪ بينما انخفضت النسبة المئوية dropped packet إلى ١٢٪ ، و ٢٢٪ كوحدة تحكم واحدة ، وفشل جهازي تحكم على التوالي في DSDN .يوضح تقييم أداء Topology Discover أن هناك علاقة تقدمية بين وقت اكتشاف Topology وعدد العقد في مستوى البيانات Throughput . وحدات تحكم opendaylight الموزعة أعلى بنسبة ٤٥.٤ نقطة مئوية من وحدة التحكم المركزية ، في حين أن وحدة التحكم opendaylight المركزية لديها Latency اعلى بنسبة ٣.٣٪ من وحدات تحكم SDN الموزعة ؛ حيث يكون كل من Throughput and Latency مقاييس قابلية توسيع شبكة SDN. يقلل SDN الموزع المقترح المستند إلى blockchain من Write Time Overhead بحوالي ١٤١ ملي ثانية فقط . كان نموذج أمن المعلومات المعروف باسم مثلث Availability- Integrity - Confidentiality (CIA) هو الأساس لتقييمنا الأمني للنظام المقترح ، حيث اثبت مدى جودة نظامنا آمنًا .

جمهورية العراق
وزارة التعليم العالي والبحث العلمي
جامعة بابل
كلية تكنولوجيا المعلومات
قسم شبكات المعلومات

منهجية فعالة لتعزيز الامنية والاتساق في وحدات تحكم الشبكات المعرفة برمجيا والموزعة بالاعتماد على السلسلة الكتلية

أطروحة مقدمة الى مجلس كلية تكنولوجيا المعلومات للدراسات العليا بجامعة بابل في استيفاء جزئي لمتطلبات درجة دكتوراه فلسفة في تكنولوجيا المعلومات /شبكات المعلومات

من قبل
ود كاظم عليوي حوشان

بإشراف

ا.م.د الحارث عبدالكريم عبد الله

هـ 1444 2023م