Design and Implementation of Distributed Controller Clustering for Solving the Issue of Single Failure in SDN Networks

Wed Kadhim Oleiwi
University of Babylon, Iraq.
E-mail: wsci.wed.kadhum@uobabylon.edu.iq

Alharith A. Abdullah
University of Babylon, Iraq.
E-mail: alharith@itnet.uobabylon.edu.iq

Received June 17, 2021; Accepted October 15, 2021
ISSN: 1735-188X
DOI: 10.14704/WEB/V18I2/WEB18395

Abstract

Abstract Software-Defined Networks (SDN) It is a centralized control structure in the network that opens up new possibilities that did not exist before. The significant characteristic of this innovative approach is the focus on the capability of proposing networks of high dynamicity and programmability to transform the intelligence of underlying systems to the networks via controllers. The main issue of the SDN approach is found in its security, mainly due to its central-controlling architecture since the entire network is controlled from a central point. This makes it very vulnerable to single-point failure. In this paper, a fully Distributed SDN controller is proposed for solving the one point failure which exists within the single SDN controller. In general, the concept involves forming cluster of distributed controllers whereby each controller controls its domain and can thereby share the load within the network. The experimental results of the proposed system show an increase and enhancement in the performance of the network. The single-point failure issues have been overcome. The throughput of the proposed system increased with 20% while the packet loss rate was minimize with 33%.

Keywords

SDN, Central Controller, Distributed Controller, OpenDayLight.

Introduction

Software-Defined Networking (SDN) is a developed approach that aims towards simplifying network managements and enabling networks to be built in order to meet particular criteria. The traffic flows can be dynamically directed and controlled in order to
achieve maximum performance benefits (Midha, S, et al., 2021). SDN has gained great attraction over the past years and has been regarded as the key of future 5G networks (Rojas, E, et al., 2021).

SDNs aim to tackle the issues related to the fixed architecture found in traditional networks, characterized by their decentralization and complexity. A modern network demands more flexible and easy form of trouble-shooting. In SDNs, the network intelligence is centralized within a single network element, as the network packets are forwarded from data to control planes (Midha, S, et al., 2021) (Rojas, E, et al., 2021).

By storing the configuration information at one place rather than dividing them over several network devices, a more global and centralized perspective is provided to the network controllers. The latter enables the calculation of optimal routes throughout the network, as well as the construction of flow tables and inserting them into the switches (Midha, S, et al., 2021) (Keerthana, B., et al., 2022).

The central controller provides the abstractions of the network's infrastructure to the application layer in order to make a forwarding decision and install the rules that represent the forwarding table on routers/switches. According to these rules, a switch packet is forwarded among ports (Barakabitze, A, et al., 2019). In this context, the construction of the control plane using one SDN controller can lead to a number of issues. First, the central SDN controller could form a single point of failure. Second, the range of networks which are controlled by a one SDN controller is restricted. So, more than one SDN controller needs to be operated to control the network area as one entity, whereby all data and services created by each controller should be shared with the other controller (Chuck Black, P, et al., 2014).

The distributed SDN controller aims towards resolving the one point of failure issue which exists in single SDN controllers. The main issue of the SDN approach is found in its security, mainly due to its central-controlling architecture since the entire network is controlled from a central point. This implies that by forming multiple controllers to share the load within networks, these controllers could replace others whenever they crash (Yustus Eko Oktian, S, et al., 2017).

A single forwarding path request could deliver the packets to the destination in a direct way, as if all switches are connected to one SDN controller. This SDN controller is distributed on a physical level, and centralized on a logical level. The East/Westbound API is a communication channel made for distributed SDN controllers only, providing the
connection among several SDN controllers for coordinating purposes crash (D. Espinel Sarmiento, A., et al., 2021). The proposed system enables the treatment of security issues in the SDN controllers by distributed architecture. The mechanism aims to decentralize the control plane for overcoming the single point failure, security issues, scalability issues, and the controller's reliability, as well as maintaining a network-wide view. To implement the proposed system, three controllers are deployed instead of the single (central), logical controller for managing the network devices, using more than one controller connected to each network device. In this case, the least number of devices will be connected to one controller at the same time, so that they can share its management.

The main contributions of the present study can be stated in the following way:

- Providing a study of distributed architecture for SDN.
- Proposing a flat distributed system with a master controller for each network domain.
- Providing higher availability: in case one controller crashes, the other instances are working and remain available.

The remaining part of the work can be outlined in the following way. The second section shows the related work. The third is a review about Distributed Software Define Network. The fourth section shows the proposed system. The fifth section involves the implementation and evaluation results. Finally, section sixth states the conclusions of the study.

**Related Work**

This part presents a review of the related research studies that focus on the distributed Software Defined Networks (SDNs).

Hyperflow, in this work, the authors (A. Tootnchian, Y, et al., 2010) proposed 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.

SOX/DSOX, the authors in (M. Luo, Y, et al., 2012) proposed a *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.

While in the work of (S.H. Yeganeh, Y, et al., 2012) 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 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.

Moreover, (Phemius, K., et al., 2014), proposed DISCO, which is an approach which can be adopted in larger networks. The network consists of smaller networks, whereby every controller is in charge of its domain while maintaining a global view over the network. The Advanced Message Queuing Protocol (AMQP) is used; whose defining feature is its routing type for communication among controllers, mainly through the Floodlight controller with inter-domain modules.

As for Berde and others in (Berde, M., et al., 2014), 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 (J. Medved, R., et al., 2014) 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 (Srirama, S., et al., 2021). 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, H., et al., 2018) 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, E., et al., 2019) 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, W., et al., 2020) presents a model for controller 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.

**Distributed Software Define Network Review**

The distributed SDN controller architecture is not only concerned about the operation multiple SDN controllers at the same time to manage networks. It also deals with the way through which these controllers communicate with each other and exchange the data necessary to manage the network. They have the same global view and operate as one entity (central controller) (R.K. Das, F., et al., 2020).

There are two major classes to categorize the distributed SDN control architecture, according to the physical organization of SDN controllers. These architectures can be either hierarchical or flat (R.K. Das, F., et al., 2020).

**The Hierarchical SDN Control Architecture**

The hierarchical model (vertical architecture) runs from top to bottom through the SDN controllers. The top controller (root) operates two or more controllers, and each of the
bottom controllers (leafs) operate one domain while maintaining its local view, as show in Figure (1). The global network state is only available in the top (root) controller. The local controller must first request network information from the top controller, which is similar to the client-server model (R.K. Das, F. et al., 2020).

**Figure 1 The hierarchical SDN control architecture**

The Flat SDN Control Architecture

The flat model (horizontal architecture) involves a p2p model whereby every controller can communicate directly to other controllers during the distribution of network information. Each of them operates their own domain, as shown in Figure (2). It replicates all local network states from SDN controllers to each another (R.K. Das, F. et al., 2020) (Tian, X, et al., 2020).

This architecture can categorize to:

**a. Fully Distributed**

In this category, each SDN controller only has its local network state, as there is no global view for the network. Any node failure impacts a subpart of the infrastructure (its domain).

**b. Logically Centralized but Physically Distributed**

It is architecture with only one controller in domain. Each time a controller creates shard of its local data and replicates it with other controllers to maintain a global view of the network. All controllers present themselves as one single controller.
A summary of comparison between central and distributed controllers is shown in Table (1).

![Figure 2 The flat SDN control architecture](image)

<table>
<thead>
<tr>
<th>Comparison elements</th>
<th>Central Controller</th>
<th>Distributed Controller</th>
</tr>
</thead>
<tbody>
<tr>
<td>No. of operating system</td>
<td>One controller</td>
<td>Multiple controller</td>
</tr>
<tr>
<td>Network view</td>
<td>Global view</td>
<td>• All controllers maintain the Global view (logically central physically distributed)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Each controller maintain its local view (fully distributed)</td>
</tr>
<tr>
<td>Control architecture</td>
<td>Central</td>
<td>1- Hierarchy</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2- Flat</td>
</tr>
<tr>
<td>Issues</td>
<td>• Security (single point of failure)</td>
<td>• Coordinate of the controllers</td>
</tr>
<tr>
<td></td>
<td>• Performance</td>
<td>• Maintain a global view of the network</td>
</tr>
</tbody>
</table>

The Proposed System

The system proposed in this work is a fully distributed system using an OpenDayLight controller with equal role. This controller supports the logically centralized and physically distributed SDN architecture, in light of the master-slave approach. Each member in cluster has a variety authority to read/write operation on a shard role. Only the controller with leader role has the ability to replicated the shards and write operation to the infrastructure, so the leader node has a load in update/forwarding to other members (T. Hu, Z. et al., 2018). The md-sal architecture of the ODL cluster depends on its basic modules: the sal-distributed-datastore and the Remote RPC (remote procedure call) Connector (Koponen, T., et al., 2016). In the proposed system, there is only one leader.
role for cluster members, whereby each controller manages its domain. For all members of the cluster there is a Remote RPC Connector module to communicate among controllers using two components: Akka Remoting and Akka clustering. This is done without the sal-distributed-data store module, so that each controller plays a leader in its domain and there no shard to replicate among controllers. Additionally, in order to ensure the consistency of execution rule in data plane, the priority of flow entry is assigned depending on the connection among controllers and domains. The priority takes the value=1 if the rule is sent by the controller that manages the domain to which the switch belongs. Otherwise, the priority takes the value=2, as explained in Algorithm (1).

AlGORITHM 1 Setting up the priority SDN controller domain

PARAMETERS:
C: the a ODL controller
D: the domain
S: openflow switch

1-BEGIN
C1 → D1(S1,S2,S3)
C2 → D2(S1,S2,S3)
C3 → D3(S1,S2,S3)

2- FLOW_ENTRY SETUP
1. IF C 1 Send a flow entry to any switch in domain D1 the priority =1.

Otherwise the priority =2.

2. IF C 2 Send a flow entry to any switch in domain D2 the priority =1.

Otherwise the priority =2.

3. IF C 3 Send a flow entry to any switch in domain D3 the priority =1.

Otherwise the priority =2.

END

The proposed system provides eventual consistency through the Gossip protocol, membership managing protocol and failure detecting mechanism to detect link failures. A fully distributed SDN network architecture with an OpenDayLight controller is explained in Figure (3). Algorithm (2) explains the steps of the proposed system.
Implementation and Evaluation Result

The software used in this experiment is presented in Table (2). These software are run on a HP DESKTOP-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. Three Ubuntu-servers type 18.04.3LTS are used, upon each of which an OpenDayLight controller is installed through the configuration of a custom network topology using mininet. Figure (3) 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. This topology remotely connects three ODL controllers. All data networks make use of OpenFlow switches (Open Vswitch 1.3). The first experiment for evaluating the system is a failover experiment, which is used to evaluate the security of the system against single-point failure issues. The results show that by turning off one controller, for any reason, a packet loss of 33% occurs, this loss for new coming traffic, as shown in Figure (5). Meanwhile in the proposed system which operates three controllers, the sudden failure of a single controller leads to a packet loss rate of 0.009%. In this case, the loss is due to technical problems in the network, such as poor connection, heavy load on the switches but not due to the loss of one of the controllers.
Table 2 The software used in the experiments of this research

<table>
<thead>
<tr>
<th>Software</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>GNS3</td>
<td>Graphical network simulator</td>
</tr>
<tr>
<td>Mininet</td>
<td>Custom Network topologies</td>
</tr>
<tr>
<td>Open vSwitch</td>
<td>Virtual SDN Switch</td>
</tr>
<tr>
<td>OpenDayLight</td>
<td>SDN controller Platform</td>
</tr>
<tr>
<td>VMWare Workstation</td>
<td>Virtualization Software</td>
</tr>
<tr>
<td>Ubuntu-server 18.04.3LTS</td>
<td>Host Operation System</td>
</tr>
<tr>
<td>Python</td>
<td>Programming language</td>
</tr>
</tbody>
</table>

Figure 3 The Proposed System

Figure 4 The implementation of proposed system in GNS3
While the second experiment is to evaluate the performance of the proposed system, the throughput criteria is used, which mean successfully delivered packet over the network from source to destination. In this work we evaluate the throughput of flow rule installation in central controller and the proposed system with different No. of device in data plane, the result show that the proposed system enhancement the throughput with 20%, the result is show in the figure (6), the result show that the throughput of central system is decrease with 15% against no. of device in data plane increase because of overload on the central controller, while in the proposed system it is only 4%. in figure (7) sample of OpenFlow messages exchange in Proposed System.
Conclusions

Distributed SDN controllers are found to be promising when it comes to resolving the single-point failure found in SDN controllers, and sharing the load in the network as one controller can take over another controller when it crashes. The proposed mechanism decentralizes the control plane to overcome the single-point failure. The SDN system was built in a virtualization environment with some advanced feature and application installed to simplify management the network's data plane. The proposed system, multiple-controller with equal role functionally, provides a promising result in security, reliability and redundancy. For future works, it is suggested to work on integrating the smart controller and blockchain with the system to provide more security in distributed SDN systems.

References


