Centralized & Distributed SDN Controller Architecture

Joshua Okosun
Department of Electrical and Computer Engineering
University of Ottawa
Jokos008@uottawa.ca

Isaac Ampratwum
Department of Electrical and Computer Engineering
University of Ottawa
Iampr061@uottawa.ca

***Abstract* - Software-defined networks (SDN) is a networking paradigm which, by separating the control plane from the data forwarding plane, can eliminate limitations on existing network infrastructures. The resulting implications include; centralization of network control and decision-making, programming on the control plane and simplifying the operation of the forwarding plane. SDN has several advantages over traditional networks however centralizing all control and decision-making introduces scalability, reliability and availability concerns. Multiple controllers are therefore important to the SDN architecture. This paper, hence, gives a review of the centralized and various multicontroller architectures present today. It also gives examples of these architectures, both implemented and under research.**

Index Terms – SDN, API, Centralized and Distributed Architectures

I. INTRODUCTION

The internet, due to its adequate and universal abstraction levels on the IP layer (network layer), has scaled largely since inception, supporting a wide range of applications and any kind of underlying physical network [1]. The Internet has however gotten to a point where new architectures that better suit emerging applications are now extremely difficult to explore. Software-Defined Networking (SDN) has emerged as a solution to this problem [2].

SDN separates the control plane from the data plane in networking devices (e.g. switches, routers) unlike traditional networks where they are combined on each network device. SDN seeks to centralize the decision making such that:

1. It simplifies the operation of network devices preventing them from having to deal with complicated decision-making tasks
2. The controller can understand the needs of applications being programmed above its layer and also resources that are available to it in the underlying network [3]
3. The controller is not vendor specific and can be implemented in software by any vendor as long as it can communicate with devices through its channel for communication.

This new concept seems to suggest using a single centralized controller to control the forwarding components in the network.

A single centralized network, which early SDN architectures suggested [4], in a SDN architecture raises three major concerns: first is efficiency, which will be constrained as just one controller exists in the network, secondly, scalability is a very pressing issue as using one controller for a fixed network might be efficient but what then happens when expansion plans set in and more switches or routers are to be added to the network, thirdly, high availability, which is comprised of two major factors, security and redundancy. Security is key to any network design. Using a single controller means that if the controller is compromised then an attacker gains access to every single network device in the network. Redundancy is also a major factor to be considered in any design. A single controller may fail at any time and throw the entire network into disarray with switches left clueless without the control plane.

These factors have led researchers to look into integrating multicontroller architectures in their different design plans and several works have been brought forward using multicontrollers. This has therefore motivated us to put together a report on the different architectures either being implemented or researched on. This paper is organized as follows: Section II provides a review of SDN architecture and its various components. Section III will contain the different centralized and distributed architecture designs with examples. In sections IV and V, we discuss the distributed architecture design requirements and choice. In section VI we discuss the various challenges being faced with the distributed architecture designs and the open research work in the field respectively. Finally, in section VII we give our conclusion.

II. SDN ARCHITECTURE

The SDN architecture has six major components [4]. These are:

Management plane: This is the topmost layer of the SDN architecture. It contains network applications managing the control logic of SDN. Examples of applications and services at the management plane are load balancing, routing or even custom applications written by service providers. This layer, via APIs allows orchestration and automation of the SDN architecture [5].

Control plane: This is perhaps the most important layer of the architecture. It houses the actual controller in charge of forwarding rules and policies, via the southbound interface, to the infrastructure layer [5].

Infrastructure plane: This is also known as data plane. This layer houses the network devices (switches, routers, etc.). Using the southbound APIs, the infrastructure layer interacts with the control plane to collect forwarding rules and policies to apply them when they receive data packets [5].

Northbound interfaces: This allows for communication between the control and management planes via application programming interfaces (APIs) [5].

East-west interfaces: This permits communication between multiple controllers in the SDN architecture. They generally utilize distributed protocols (e.g. BGF and OSPF) or notification and messaging systems.

Southbound interfaces: This allows for interaction between the control plane and the data plane. The most commonly utilized and widely accepted southbound API is the OpenFlow protocol [4]. It is normalized by the Open Networking Foundation (ONF) and backed by various IT industry leaders.

Figure 1 below gives a graphical top-level view of the discussed SDN architecture components.

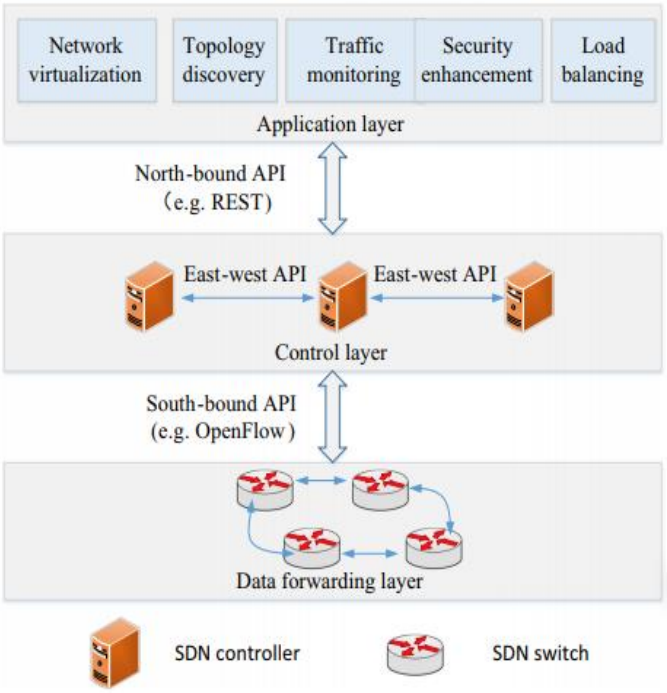

Figure 1: The SDN architecture [6]

III. SDN MULTICONTROLLER ARCHITECTURES

Multicontroller architectures are architectures where a set of controllers work collaboratively to achieve a certain degree of scalability and performance. In this section we will be discussing the various multicontroller architectures. Figure 2 below shows a breakdown of all the architectures we will be discussing, and how they are linked.

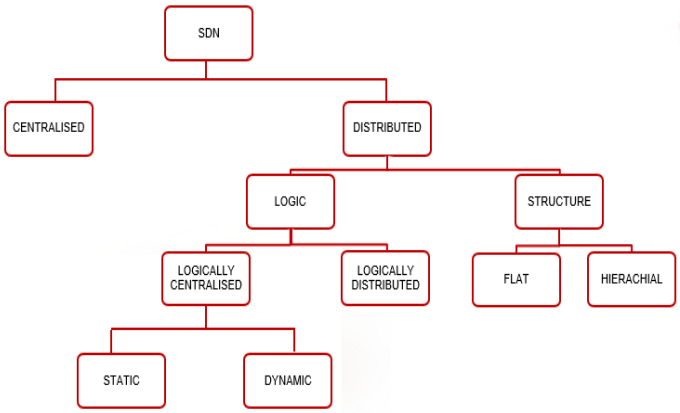

Figure 2: SDN Multicontroller architectures

1. Flat vs Hierarchical Architecture

A flat, also known as horizontal architecture, is an architecture where all the SDN controllers are placed on a single layer as shown in figure 3. The control plane here has just a single layer. Every controller has a partial view *(as illustrated by the uniquely colored switches and links)* of the networks and has the same responsibilities as all other controllers. The ONIX [7] is an example of a flat architecture

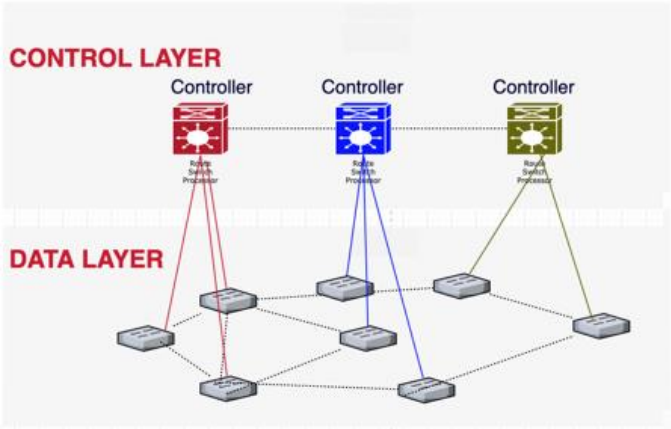

Figure 3: The Flat architecture

The hierarchical or vertical architecture is one where the SDN controllers are positioned vertically. In this architecture, the control plane has multiple levels as shown in figure 4. There are usually two or three levels. The controllers at different levels have unique responsibilities and take decisions based on partial network views. The KANDOO [8] SDN solution is a very good example of a hierarchical architecture.

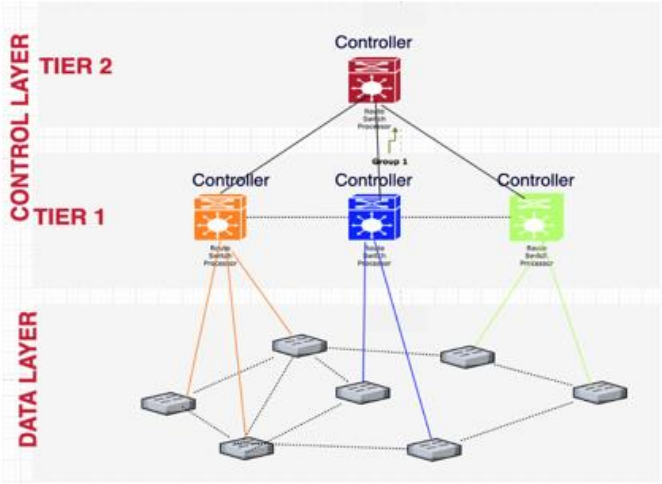

Figure 4: The Hierarchical architecture

Flat and hierarchical architectures both have advantages and disadvantages. These architectures both ameliorate the latency issue between switch and controller in comparison to single controller architectures. Whilst the flat architecture gives a higher resilience to failures, its use of multiple controllers on the same layer makes controlling them much more difficult. The hierarchical design on the other hand provides a simpler way to manage multiple controllers in the network because responsibilities are separated but it fails to eradicate the single point of failure issue.

2. Logically Distributed vs Logically Centralized Architectures

SDN distributed architectures can also be differentiated based on the logic of the control panel. This refers to how knowledge of the network is shared among controllers. The architecture can either be Logically centralized of Logically distributed.

Logically centralized Architecture

Blial et al [4] defined this architecture as one that takes advantage of the multicontroller design and at the same time considers a single controller as seen in figure 5 below. This implies that though each controller in the network has a part of the network it is responsible for; the data plane only sees one controller for the topology. All the controllers in this setup control equal part of the whole network and shares information regularly with one another giving each controller current network information and overview. Logically centralized architectures are mostly used in Intra-domain networks [9].

In a logically centralized architecture, the links and positions between controllers and the switches can either be static or dynamic. In a static architecture, the switches assigned to each controller is fixed. This can be problematic in real networks where network traffic is unpredictable [10]. Dynamic architecture gives flexibility where flow requests traffic from a particular switch can be forwarded to different controllers in order not to overwhelm a particular controller whiles the static architecture offers more stability and less overhead. Some examples of logically centralized architectures include ONOS, HyperFlow, Opendaylight, Ravana and ONIX.

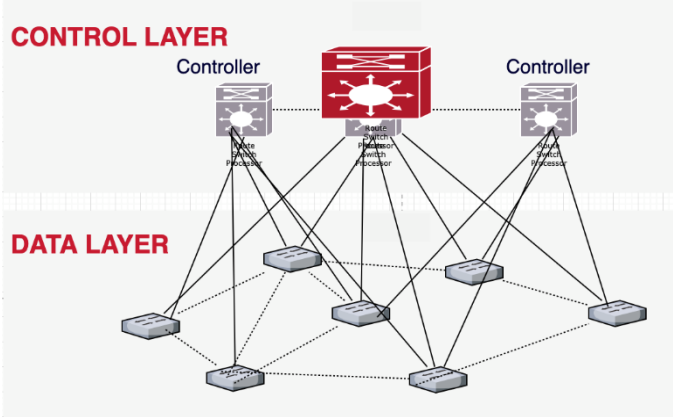

Figure 5: The Logically centralized architecture

Logically Distributed Architecture.

Logically centralized architecture, though very productive, is quite ineffective at the intra-domain level with multiple ASs **[9].** In such applications, network managers leverage the logically distributed architecture. In this architecture, the controllers are both physically and logically distributed. Each controller has just a view of the network part it controls. Controllers make decisions based on a partial view of the network unlike the case of logically centralized architecture. Examples of Logically distributed architectures are DISCO and SDX based controllers

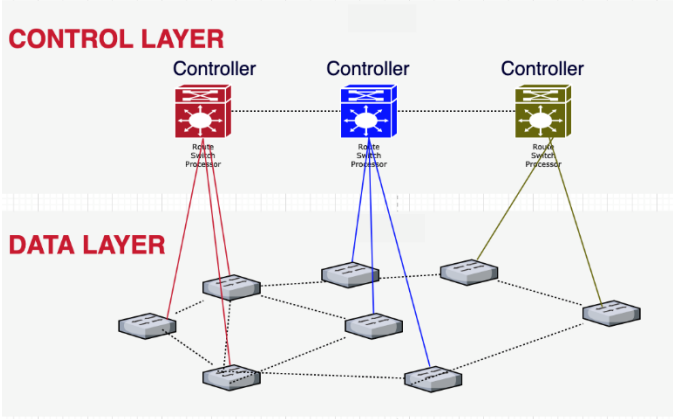

Figure 6: The Logically distributed architecture

IV. DISTRIBUTED ARCHITECTURE DESIGN REQUIREMENTS

Distributed architecture deployment has gained a lot of attention in both industry and research communities. In this section, we present a summarized literature on the requirements taken into consideration for various architectures available as well as design trends.

Examples of design requirements satisfied by various distributed architecture designs are

- fault tolerance

availability

- consistency

It is important to note that, existing design platforms may meet one, two or all three of the outline requirements based on the use environment.

Fault tolerance

Fault tolerance is the ability of the network to maintain a certain level of functionality when a network fault occurs. A fault can be when a controller fails. Achieving fault tolerance is a vital requirement in distributed system architecture design. In fault scenarios like the failure of a controller, having more than one controller in the set up can reduce the impact of a controller failure on the network. [10] outlines a list of controller platforms and their fault tolerant mechanisms. Fault tolerance can be achieved through passive and active replication mechanisms [11]

In passive replication, there is a primary controller that oversees all the switches in the architecture. When the primary controller fails, a back-up controller takes over as the primary controller of the architecture. Active replication on the other hand allows multiple controllers to have communication with the switches in the network. In this case when a controller fails, other controllers can still control the switches since the already have an established connection.

Another possible fault in the network is a node or link failure. In such situations, the switch basically sends a message to the controller informing it of the occurred fault and the controller computes new routes based on its current view of the network

Availability

Availability can be defined as a proportion of time an SDN system is in functioning mode [12]**.** Availability entails two major things; redundancy and security [10]**.** Rules backup, controller load and switch request are factors that can be considered to improve availability [10]**.**

By having multiple backups to active rules, the backups can move up as active rules when a fault renders the present active rule useless. Also, by controlling the weight of requests on a controller, you can improve the availability of the controller. You can achieve this with a distributed controller setup and balance the loads among the controllers. Security features must also be put in place to prevent attackers from compromising controllers. Having multiple controllers allow for continued function even if one is compromised for security reasons.

Consistency

In a distributed controller environment, consistency is key to maintain network stability. Several solutions based on diverse techniques such as clustering have been proposed by the research community. Maintain consistency in communication and decision making among multi-controller set up is very crucial.

Consistency in distributed architectures include state consistency, version update consistency and rules update consistency [10]

- State consistency requires that all controllers have an identical view of the network globally.
- Version update consistency ensures that multiple controllers have newest state rather than hold the old state of the network [10]
- Rules update consistency requires that controllers should have the same policies for flow rules to prevent conflicting forwarding actions on switches.

V. DISTRIBUTED ARCHITECTURE DESIGN CHOICE

To satisfy the various design requirements outline in the previous section, various SDN designers employ different trends. In this section, we will outline common choices in distributed architecture design.

The choices can be based on switch to controller connection strategy, network information distribution strategy, controller coordination strategy and management traffic connection strategy.

Management traffic link strategy

This choice outlines how the traffic for network management is connected. There are two options, in-band or out-of-band **[13].** Out-of-band requires a dedicated link for network control traffic whereas in-band uses the existing data links to transfer the management traffic. In-band may be slower, but it is cheap and out-of-band is expensive since extra bandwidth must be dedicated solely for management traffic.

Switch to controller connection strategy

In a distributed controller architecture, the strategy for the communication between switches and controller plays a major role. In a distributed architecture, switches should be to communicate to multiples controllers which was not the case in earlier SDN architectures. When a controller fails, the switches should be able to establish communication with other controllers without a network personnel being present to reconfigure the set up. This dynamic communication can be achieved by IP aliasing or OpenFlow master/slave connection [13].

IP aliasing is associating more than one IP address to a network interface [14]. Each controller is assigned a specific IP at any given time by the cluster leader to establish communication between the controller and switches. If that controller fails, the cluster leader automatically re-assigns that IP address to any controller in the cluster to establish a switch controller communication.

The second approach is the OpenFlow master/slave connection. This approach utilizes the ability of the switch to connect to at least two controllers based on their function. OpenFlow version 1.2 allows the switch to connect master and slave controllers simultaneously. If the master controller fails, the cluster head can switch the roles of the controllers making the slave controller now a master and can now control the switch.

Network Information Distribution strategy
This strategy outlines how information is shared among controllers to ensure consistency. Based on the model, flat or hierarchical, some or all the controllers may need to know the global state of the network. The 2 models achieve information sharing in distinct ways. In hierarchical model, it is only the root controller that has a global view of the network. All local controllers must query the root controller before it executes any inter-domain communication.
In a flat architecture, the controllers use east/west APIs to update each other of their local network state. Every controller in this architecture has a global view of the network through this distribution strategy.

Controller coordination strategy
This choice is about who can orchestrate the network. In a scenario where two controllers perform contrasting functions on the same switch, the switch will have a challenge deciding which action to take. This can occur if proper coordination method is not implemented. Two common approaches are leader-based and leaderless coordination.
In leaderless coordination, each controller has the same network control rights. To install actions on switches belonging to other controllers, a controller simply contacts that controller in charge of the switch. HyperFlow uses this approach [13].
In the leader-based approach, there exists a controller with more control rights than the other controllers. All controllers get permission from the cluster head before installing flow rules and network actions. Controller platforms combine different design choices to achieve functional requirements as outlined in previous section.

VI. CHALLENGES OF DISTRIBUTED ARCHITECTURE AND FUTURE RESEARCH WORK

Distributed Architecture has several challenges that the research community must give special attention to. These include issues with scalability, reliability, consistency, interoperability, monitoring and security. [9]

Scalability
As the network expands, communication between the controller and switches delay and latency increases. Several works have been done to solve the scalability problem. Some researchers have suggested giving the switches some capabilities so the total load of the controller in the set up will be reduced. This approach increases the complexity of the OpenFlow switches. The more effective solution is the increase the number of switches as the network size increases. In depth concerns with this approach include how many controllers are to be used in a particular network size and where should these controllers be placed. These issues are research worthy.

Reliability
The problem of single point of failure when dealing with a centralized SDN architecture can be solved by implementing a physically distributed architecture. Having multiple controllers has its own issues like coordination and information sharing. As discussed in previous chapters, there are ways to mitigate these issues. We however believe more work can be done to improve on these issues regarding which messages should be exchange between the controllers.

Consistency
Achieving consistency while keeping good network performance in SDN partitions is a tough task **[9].** This is an avenue for further research.

Interoperability
There are no standards governing distributed architecture implementation. Every designer makes up a design to fit certain applications. This is a major concern for network interoperability among different architectures as well as coupling with legacy networks. Standardizing SDN implementation is key to achieving network Interoperability

Monitoring and Security
Efficient network monitoring is necessary to provide information for network function management. A more improved monitoring approaches that would help gather the appropriate network data with impacting the network performance and functionality.

Also, Network security is a very important challenge in SDN networks that must be thoroughly studied. SDN controllers are targets for crippling the whole network hence the need for more security solutions.

VII. CONCLUSION

SDN is based on the idea of separating the control plane from forwarding plane and centralizing the whole the whole control in one single controller to manage the network. However, over the years, the academia and industry realized that the future of SDN relies on distributed architectures, because centralized architectures have certain challenges including efficiency, scalability and availability.

We have provided a review on the on both centralized and distributed SDN architectures. Though exploring distributed architectures are vital for the growth of SDN systems, we do not yet see any ideal solution that caters for all challenges experienced in the centralized architectures. We found that each design exhibits strength when dealing with a particular challenge in centralized SDN architecture, like scalability for example, and then fall short in controlling the problem of point of failure.

In the coming years, further research must be done to deal with the many problems in distributed architectures like developing efficient communication process, adequate network design and controller placement.

REFERENCES

[1] Hoang, Doan B., and Minh Pham. "On software-defined networking and the design of SDN controllers." *2015 6th International Conference on the Network of the Future (NOF)*. IEEE, 2015.

[2] N. Mckeown, "How SDN will shape networking", https://www.youtube.com/watch?v=c9-K5_qYgA, accessed 30 Aug 2015.

[3] ONF, "Software-Defined Networking: The new norm for networks" , https://www.opennetworking.org/images/stories/downloads/sdnresources/white-papers/wp-sdn-newnorm.pdf, accessed 30 Aug 2015

[4] Blial, Othmane, Mouad Ben Mamoun, and Redouane Benaini. "An overview on SDN architectures with multiple controllers." *Journal of Computer Networks and Communications* 2016.

[5] D. Kreutz, F. M. V. Ramos, P. E. Verissimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-defined networking: a comprehensive survey," Proceedings of the IEEE, vol. 103, no. 1, pp. 14–76, 2015.

[6] Shu, Zhaogang, et al. "Security in software-defined networking: Threats and countermeasures." *Mobile Networks and Applications* 21.5 (2016): 764-776.

[7] Koponen, T., Casado, M., Gude, N., Stribling, J., Poutievski, L., Zhu, M., ... & Shenker, S. (2010, October). Onix: A distributed control platform for large-scale production networks. In *OSDI* (Vol. 10, pp. 1-6).

[8] Hassas Yeganeh, Soheil, and Yashar Ganjali. "Kandoo: a framework for efficient and scalable offloading of control applications." *Proceedings of the first workshop on Hot topics in software defined networks*. ACM, 2012.

[9] Bannour, Fetia, Sami Souihi, and Abdelhamid Mellouk. "Distributed SDN control: Survey, taxonomy, and challenges." IEEE Communications Surveys & Tutorials 20.1 (2018): 333-354.

[10] Zhang, Yuan, et al. "A survey on software defined networking with multiple controllers." Journal of Network and Computer Applications 103 (2018): 101-118.

[11] Fonseca, P., Bennesby, R., Mota, E., Passito, A., 2012. A replication component for resilient OpenFlow-based networking. In: Proceedings of Network Operations and Management Symposium, pp. 933–939.

[12] Nencioni, Gianfranco, et al. "Impact of SDN controllers deployment on network availability." *arXiv preprint arXiv:1703.05595* (2017).

[13] Oktian, Yustus Eko, et al. "Distributed SDN controller system: A survey on design choice." computer networks 121 (2017): 100-111.

[14] Wikipedia. "IP Aliasing" Internet:https://en.wikipedia.org/wiki/IP_aliasing, Apr 13,2019[Jan. 10,2019.]