Distributed Software-Defined Networking Management: An Overview and Open Challenges

Rawan S. Alsheikh[1]†, Etimad A. Fadel[1] and Nadine T. Akkari[2]

[1]Department of Computer Science, Faculty of Computing and Information Technology, King Abdulaziz University, Abdullah Sulayman, Jeddah 21589, Saudi Arabia.

[2]Department of Computer Science and Information Technology, Jeddah International College, Ibn Rasheed Elfehri, Jeddah 23831, Saudi Arabia.

***Abstract*—Distributed software-defined networking (SDN) architecture satisfies the minimum requirements for wide-area networks. The distributed controllers are connected in various topologies, including hierarchical and flat, which include logically centralized physically distributed, and fully distributed controllers. The distributed SDN architectures are qualitatively explored as a more suitable solution for managing fluctuating networks in large-scale deployments, with the goal of optimizing overall network performance, particularly for applications that can tolerate some level of inconsistency, such as load balancing or routing. The logically centralized physically distributed SDN controller architecture allows SDN controllers, in conjunction with the deployed SDN applications, to centrally coordinate the network due to the conciliated global network view. That is created through the synchronization process between controllers. However, inter-controller synchronization creates an overhead that affects the system's performance. In addition, the amount of inter-controller synchronization is vulnerable to the chosen consistency approach the application can tolerate. Although static eventual consistency is frequently employed in modern SDN systems to provide effective scalability, it is argued that it does not place limits on the state inconsistencies that SDN applications will tolerate. Hence, the adaptive consistency models need to be investigated. The study showed that a flat, logically centralized physically distributed architecture with an adaptive consistency approach would be more suitable for solving large-scale fluctuating network management considering scalability, reliability, and maximizing performance.**

***Index Terms* –Adaptive consistency, Distributed SDN architecture, Large-scale networks, Logically centralized physically distributed controllers, Network performance optimization**

I. Introduction

Software-defined networking (SDN) is an evolving strategy that separates the network control plane from the data plane. SDN operates across three layers: Application Layer (Management Plane), Control Layer (Control Plane), and Infrastructure Layer (Data Plane). Fig. 1 shows these interactions (Akyildiz, 2014), (Keshari, Kansal and Kumar, 2021).

SDN control plane architecture can be centralized (Fig. 1) or distributed (Fig. 2). This paper focuses on large-scale wide-area networks (WANs) with key scalability, reliability, and performance requirements.

The single SDN controller, as in Fig. 1, has a single point of failure (SPOF) problem. Moreover, it is challenging for one controller to handle the whole network due to the diversity of network application needs and the growth of network size. Getting the network view was difficult after that. These encourage network designers to consider including multi-controller architectures (Yu, Qi and Li, 2020), (Almadani, Beg and Mahmoud, 2021), (Aslan and Matrawy, 2016), (Blial, Ben Mamoun and Benaini, 2016), (Ahmad and Mir, 2021), (Oktian, et al., 2017), (Tadros, Mokhtar and Rizk, 2019).

In distributed SDN architectures, the data layer is divided into multiple domains, each managed by a controller. These controllers communicate through East/West-bound interfaces to exchange inter-domain information, allowing the deployment and configuration of SDN applications on large-scale networks (Ahmad and Mir, 2021), (Chen, et al., 2017), (Hoang, et al., 2022).

Distributed controllers are connected in various topologies, including hierarchical and flat, which include logically centralized physically distributed, and fully distributed controllers (Blial, Ben Mamoun and Benaini, 2016) (Espinel Sarmiento, et al., 2021). Each architecture has strengths and weaknesses. This paper qualitatively compares distributed SDN architectures and seeks to identify the most efficient SDN architecture for managing large-scale networks, balancing consistency, scalability, and performance.

Hierarchical architectures (e.g., Kandoo [Hassas Yeganeh and Ganjali, 2012] and B4 [Jain, et al., 2013]) feature a root controller with a global network view, whereas local controllers manage only their domains. Although easier to manage, this approach introduces latency due to cross-domain communication through the root controller, leading to performance degradation (Blial, Ben Mamoun and

ARO-The Scientific Journal of Koya University
Vol. XII, No. 2 (2024), Article ID: ARO.11468. 166 pages
Doi: 10.14500/aro.11468
Received: 09 November 2023; Accepted: 19 September 2024
Regular review paper; Published: 30 September 2024
†Corresponding author's e-mail: rsalshaikh@kau.edu.sa

Benaini, 2016), (Ahmad and Mir, 2021), (Oktian, et al., 2017), (Espinel Sarmiento, et al., 2021).

Flat architectures with logically centralized but physically distributed controllers (e.g., HyperFlow [Tootoonchian and Ganjali, 2010], Onix [Koponen, et al., 2010], ONOS [Open Network Operating System (ONOS) SDN Controller for SDN/NFV Solutions, no date], ElstiCon [Dixi, et al., 2014], and Orion [Ferguson, et al., 2021]). Allow each controller to manage intra- and inter-site operations, forming a global network view through synchronization (Blial, Ben Mamoun and Benaini, 2016), (Espinel Sarmiento, et al., 2021). However, synchronization can create overhead, affecting performance (Ahmad and Mir, 2021). Balancing consistency and performance remains a challenge.

Flat architectures with fully distributed controllers (such as DISCO [Phemius, Bouet and Leguay, 2014] or ODL [Home - OpenDaylight, no date]) operate with reduced inter-controller communication, prioritizing local optimization over global interests (Blial, Ben Mamoun and Benaini, 2016), (Espinel Sarmiento, et al., 2021), (Bannour, Souihi and Mellouk, 2018b), (Informatique and Informatique, 2021).

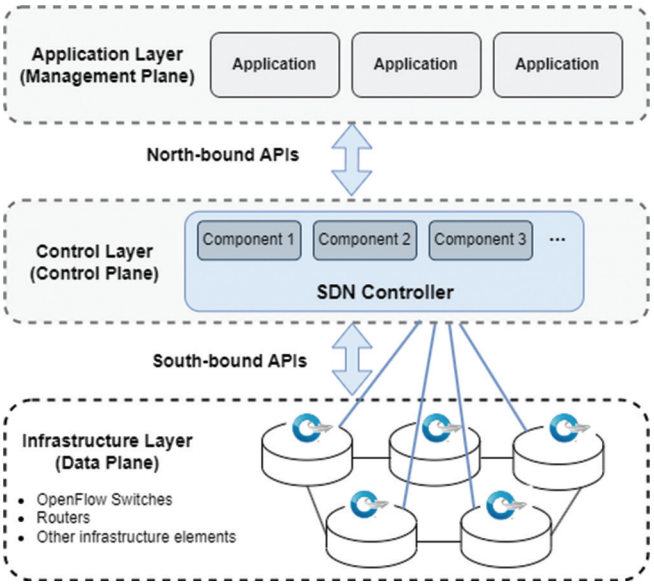

Fig. 1. Overview of software-defined networking architecture.

A logically centralized, physically distributed approach is preferable for administering large-scale solutions. This enables any application within the cluster to access and manage the global network view (Tadros, Mokhtar and Rizk, 2019), (Espinel Sarmiento, et al., 2021. However, inter-controller communication costs remain a significant challenge, which does not receive as much concern in the research community (Blial, Ben Mamoun and Benaini, 2016), (Alowa and Fevens, 2020), (Espinel Sarmiento, et al., 2021).

Tradeoffs between strong and eventual consistency models impact network performance: strong consistency ensures accurate data but introduces synchronization overhead, whereas eventual consistency offers reduced overhead with temporary data inconsistencies. This paper explores adaptive consistency, which dynamically balances these tradeoffs to optimize performance in large-scale SDN networks (Aslan and Matrawy, 2016), (Ahmad and Mir, 2021), (Levin, et al., 2012), (Foerster, Schmid and Vissicchio, 2019), (Bannour, Souihi and Mellouk, 2018a).

This work's contributions are as follows:

- We analyze the key challenges distributed SDN architectures face, particularly focusing on scalability, reliability, consistency, and synchronization overhead in large-scale networks to determine the most effective architecture for network administration
- We provide a qualitative comparison of hierarchical, flat, and logically centralized physically distributed SDN architectures, highlighting their strengths and weaknesses in managing large-scale fluctuating networks. Based on this analysis, we suggest that a logically centralized, physically distributed architecture is most suitable for network administration

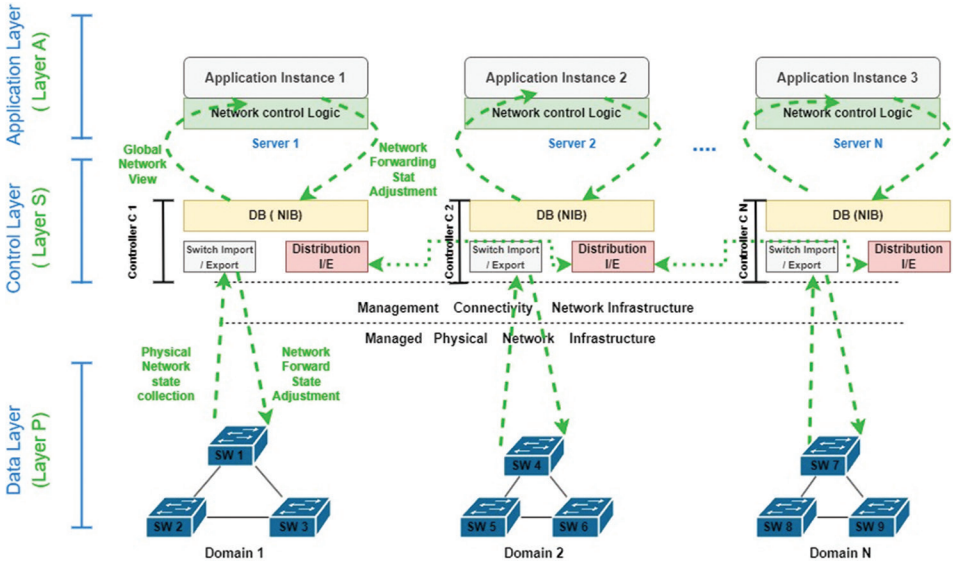

Fig. 2. Distributed software-defined networking architecture with network state distribution.

- We evaluate strong, eventual, and adaptive consistency models in the context of SDN, emphasizing their impact on network performance, scalability, and application requirements
- We propose using an adaptive consistency model for logically centralized, physically distributed SDN architectures. This approach aims to solve the synchronization overhead problem, which introduces latency and degrades performance. By employing dynamic consistency, the synchronization process can adapt to network conditions, reducing overhead whereas maintaining system performance, particularly for applications that can tolerate some inconsistency.

The rest of this paper is structured as follows: Section 2 discusses the Architecture and Components of Distributed SDN. Distributed SDN architecture challenges are discussed in section 3. Section 4 presents Distributed SDN Topologies and Qualitative Evaluation. Section 5 focuses on Network State Consistency. Section 6 discusses Distributed SDN: Future Directions. The study is concluded in section 7.

II. The Architecture and Components of Distributed SDN

Understanding how distributed SDN architecture is designed and operates is crucial. Fig. 2 presents a distributed SDN system consisting of three main layers: the Application Layer (Layer A), the Control Layer (Layer S), and the Data Layer (Layer P). This layered structure resembles the Onix SDN control systems (Koponen, et al., 2010) and the work on (Levin, et al., 2012). The architecture consists of five essential components: Data layer, connectivity infrastructure, instances of the distributed control plane, control logic, and database. The green dashed and dotted arrows in Fig. 2 illustrate the points where network states are exchanged across the system.

A. Data Layer

Network switches and routers are illustrated as blue boxes at the lowest layer in Fig. 2. These forwarding devices retain the network data plane state in the Forwarding Information Base and associated metadata such as flow ports and packet counters. These devices run the software necessary to support interfaces, such as OpenFlow, enabling controllers to read and modify the network state by updating forwarding table entries. A controller's domain includes all switches and hosts directly connected to it (Oktian, et al., 2017), (Koponen, et al., 2010), (Levin, et al., 2012).

B. Connectivity

The connectivity framework supports bidirectional communication, depicted by the green dotted arrows in Fig. 2 for interactions between control plane instances, and by the green dashed arrows for communication between switches and control plane instances. It facilitates convergence during link failures and employs standard protocols, such as OSPF or IS-IS, to preserve the forwarding state (Koponen, et al., 2010).

The East-bound interface (red box in Fig. 2) allows SDN controllers to exchange information, whereas a West-bound interface can connect SDN controllers with legacy systems (Ahmad and Mir, 2021), (Hoang, et al., 2022).

Controllers can interconnect vertically (hierarchical) or horizontally (flat). However, intercontroller communication costs remain a significant challenge in distributed SDN environments (Blial, Ben Mamoun and Benaini, 2016), (Espinel Sarmiento, et al., 2021), (Alowa and Fevens, 2020).

C. Distributed Control Plane Instances

The control plane operates as a distributed system, with each instance functioning as a network operating system (NOS) (Koponen, et al., 2010), (Levin, et al., 2012). Controllers oversee the management of the network state (Layer S) and provide programmatic interfaces for accessing and modifying control logic. Control plane instances collaborate within the cluster through distribution I/E, enabling SDN applications (Layer A) to interact with a simplified, abstract representation of the physical network (Koponen, et al., 2010), (Levin, et al., 2012). Each controller maintains a Network Information Base (NIB) housed in a database (yellow box in Fig. 2) (Levin, et al., 2012).

D. Control Logic

The control logic, represented by the green box in Fig. 2, operates on the controller's API. It uses network state information to determine the intended network behavior (Koponen, et al., 2010), (Levin, et al., 2012).

E. Database

The database management system (yellow box in Fig. 2) is crucial for the distributed control plane, storing intra- and inter-domain information for each controller. It could be utilized as a method of information sharing between controllers, obviating the necessity for a specific communication protocol. SDN solutions use either SQL or NoSQL databases (Espinel Sarmiento, et al., 2021).

A distributed database architecture facilitates the scalability of the control plane and enhances its ability to manage system failures efficiently (Koponen, et al., 2010).

III. Distributed SDN Architecture challenges

While the SDN project has the potential to revolutionize and improve networks, it is still in its early phases of tackling a wide range of difficulties (Fig. 3), including scalability, reliability, consistency and synchronization overhead, interoperability, East–West interface implementations, and security (Informatique and Informatique, 2021), (Hussein, et al., 2018).

Although reliability and scalability are thought to be the two key drawbacks of centralized SDN control architectures, they are equally significant considerations when creating a physically distributed SDN architecture (Informatique and Informatique, 2021), (Hussein, et al., 2018), and (TS, 2019). This paper will address key issues such as interoperability

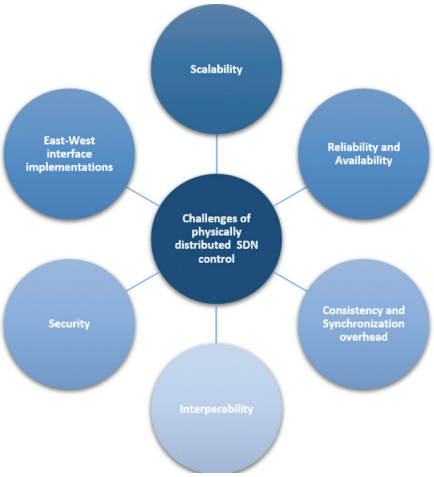

Fig. 3. The main challenges of physically distributed software-defined networking control.

and security; however, the primary focus will be on the challenges related to the consistency and synchronization overhead.

A. Consistency and Synchronization Overhead

In a physically distributed architecture, consistency necessitates preserving an up-to-date network-wide perspective (Ahmad and Mir, 2021). Consistency management is achieved through an inter-controller synchronization process, which ensures that the network view is aligned across all controllers (Oktian, et al., 2017).

Addressing consistency challenges whereas considering trade-offs in SDN controller platforms is vital. Achieving consistency necessitates a synchronization process, which can impose significant overhead on the system. While lower consistency levels reduce synchronization overhead, they increase the risk of state conflicts. On the other hand, strict consistency enforces more frequent synchronization, leading to higher overhead in the control plane (Sakic, et al., 2017). This overhead increases latency and impacts the system's scalability. Latency, often referred to as responsiveness, is the time the system requires to respond to flow requests (Tadros, Mokhtar and Rizk, 2019), (Hoang, et al., 2022), (Espinel Sarmiento, et al., 2021), (Informatique and Informatique, 2021), (Levin, et al., 2012), (Akyildiz, 2014).

B. Interoperability

Interoperability among SDN controllers is essential for the effective operation of distributed controller systems. Another significant operational problem related to SDN's maturity, growth, and commercial usage is the compatibility across various SDN controller systems from various suppliers (Yu, Qi and Li, 2020), (Informatique and Informatique, 2021).

The fact that each SDN controller has its own communication method makes it extremely challenging for SDN networks to communicate information between various domains (Yu, Qi and Li, 2020).

However, contrary to the wide acceptance of the standardization of OpenFlow's southbound interface, the research community has not given the East–Westbound interface the required attention to provide interoperability and synchronization among SDN controllers. This is driven by the fact that there is no need for SDN network compatibility because the transition from the traditional network to SDN is still just a minor one. However, as SDN continues to demonstrate its benefits, The need for standard East-West protocol is essential to the internet's existence in the near future (Hoang, et al., 2022).

C. Security

Another significant issue that needs to be researched is SDN security. The integrity of data flows between SDN controllers and switches is still not guaranteed (Informatique and Informatique, 2021). In addition, authentication procedures are urgently needed in distributed control architecture to validate and verify controller instances (Ahmad and Mir, 2021). This paper does not consider security issues, as the focus is primarily on applications that can tolerate a certain level of inconsistency, such as load-balancing or routing applications.

IV. Distributed SDN Topologies and Qualitative Evaluation

Several research studies, such as Yu, Qi and Li, 2020, Hoang, et al., 2022, Espinel Sarmiento, et al., 2021), and Informatique and Informatique, 2021, have introduced a thorough evaluation of cutting-edge distributed SDN controller platforms.

Several topologies, such as hierarchical or flat, are used for distributed controller interconnection. The flat architecture can be logically centralized physically distributed, or fully distributed (Ahmad and Mir, 2021), (Espinel Sarmiento, et al., 2021). Logical classification in our research was adopted by (Espinel Sarmiento, et al., 2021), (Bannour, Souihi and Mellouk, 2018b), (Informatique and Informatique, 2021).

Fig. 4 illustrates some differences between these SDN distributed architectures and how each architecture preserves the network state. It distinguishes between local (domain-specific) and global (network-wide) states. The local state reflects the current network status within a controller's domain, tracking events such as host connections and link changes. In contrast, the global state represents the overall network status across all domains (Oktian, et al., 2017).

Each topology affects key issues in distributed SDN controllers, such as scalability, reliability, latency, consistency, and synchronization overhead (Keshari, Kansal and Kumar, 2021), (Ahmad and Mir, 2021), (Oktian, et al., 2017), (Bannour, Souihi and Mellouk, 2018b), (Hu, Li and Huang, 2014). Table I summarizes the characteristics of hierarchical, flat logically centralized, and flat fully distributed topologies based on these factors.

A. Hierarchical

The hierarchical architecture, such as Kandoo (Hassas Yeganeh and Ganjali, 2012) and B4 (Jain, et al., 2013),

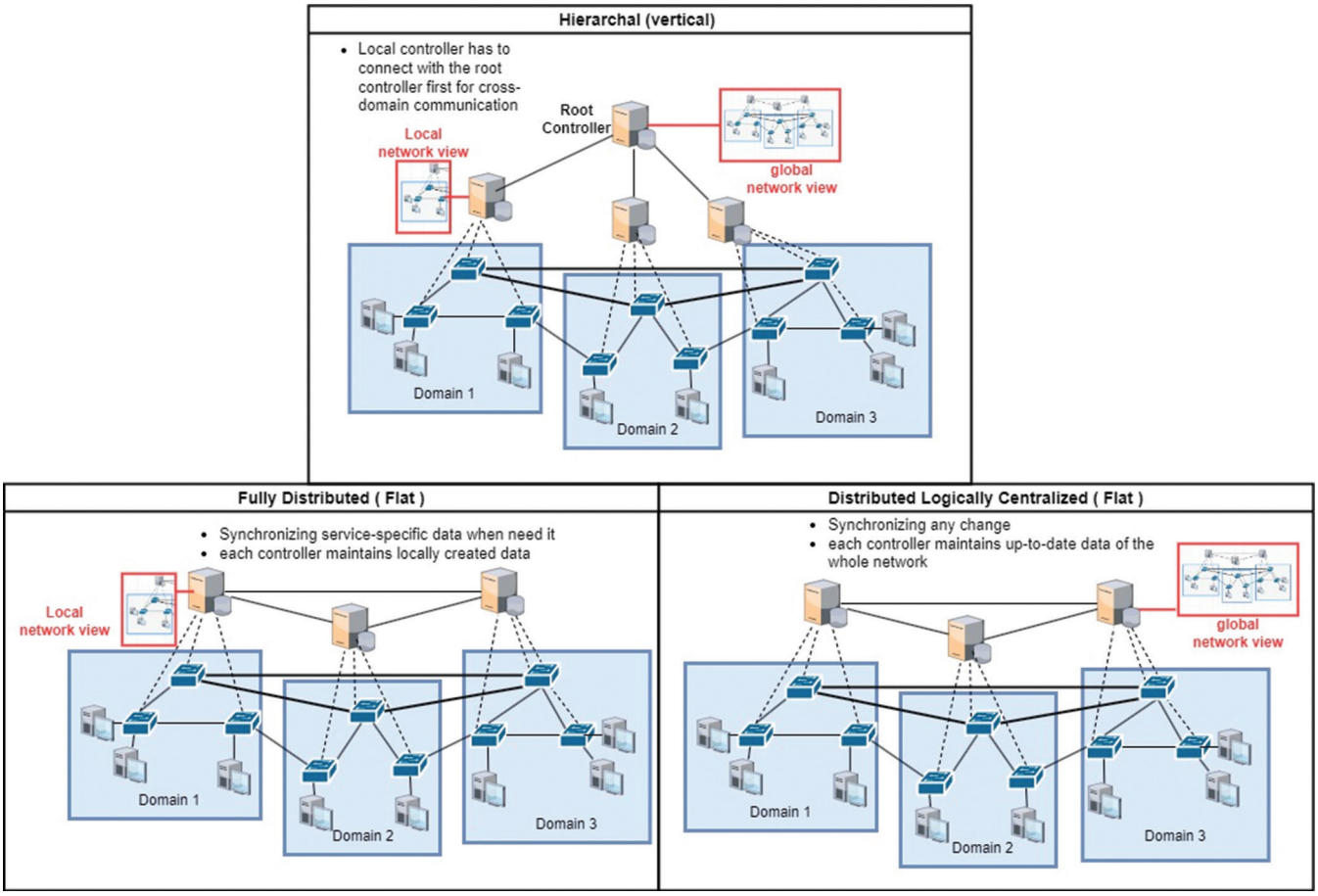

Fig. 4. Distributed software-defined networking topologies.

TABLE I
DISTRIBUTED SDN TOPOLOGY CHARACTERISTICS

Distributed Controller Topology	Controller-to-Controller connectivity model	Scalability	Robustness (reliability)	Latency	Managing consistency	Synchronization Overhead	Real case scenarios
Hierarchical	Vertical	High	Low (root controller SPOF)	High	Centralized in the root controller	Medium	Kandoo (Hassas Yeganeh and Ganjali, 2012) and B4 (Jain, *et al*., 2013)
Logically Centralized Physically Distributed	Horizontal (Flat)	Medium	High	Medium (depends on the technical execution of the solution)	By synchronization and creating a global view	High/Medium (depending on the chosen consistency model)	Elasticon (Dixi, *et al*., 2014), HyperFlow (Tootoonchian and Ganjali, 2010), Orion (Ferguson, *et al*., 2021), DragonFlow (OpenStack Docs: Distributed Dragonflow, no date), Onix (Koponen, *et al*., 2010), and ONOS (Open Network Operating System (ONOS) SDN Controller for SDN/NFV Solutions, no date)
Fully Distributed	Horizontal (Flat)	Very High	Very High	Low	Manage consistency at the application level	Low (shares with other instances when necessary).	DISCO (Phemius, Bouet and Leguay, 2014) and ODL (Home - OpenDaylight, no date)

organizes controllers in a vertical arrangement, with the root controller positioned at the top. This root controller manages the global network state, whereas local controllers are responsible for handling their specific domains. The root controller ensures network-wide connectivity by overseeing subordinate controllers (Blial, Ben Mamoun and Benaini, 2016), (Ahmad and Mir, 2021), (Oktian, et al., 2017), (Espinel Sarmiento, et al., 2021).

While this method improves scalability over centralized systems, robustness remains limited due to the root controller

being a SPOF (Blial, Ben Mamoun and Benaini, 2016), (Ahmad and Mir, 2021), (Espinel Sarmiento, et al., 2021), (Informatique and Informatique, 2021).

Replicating the root controller across multiple controllers can reduce SPOF risk, but having too many root controllers can complicate coordination and reduce network efficiency (Oktian, et al., 2017), (Espinel Sarmiento, et al., 2021).

There are no East/West-bound API connections between the local SDN controllers in the cluster. Instead, they communicate solely with the root controller. This dependency for cross-domain communication introduces latency and performance degradation (Ahmad and Mir, 2021), (Oktian, et al., 2017), (Espinel Sarmiento, et al., 2021).

B. Flat

In flat architectures, controllers are arranged horizontally, each managing part of the network, whereas simultaneously carrying out the same responsibilities (Blial, Ben Mamoun and Benaini, 2016), (Ahmad and Mir, 2021).

To tackle consistency, flat systems can use either leader-based or leaderless coordination. In leader-based systems (e.g., Onix [Koponen, et al., 2010], ONOS [Open Network Operating System (ONOS) SDN Controller for SDN/NFV Solutions, no date], and ElstiCon [Dixi, et al., 2014]), a cluster elects a leader to manage communication (Oktian, et al., 2017). Leaderless systems (e.g., HyperFlow [Tootoonchian and Ganjali, 2010] or DISCO [Phemius, Bouet and Leguay, 2014]), allow controllers to communicate directly, sharing equal roles (Oktian, et al., 2017).

Flat architectures enhance failure resilience and performance but complicate consistency management (Blial, Ben Mamoun and Benaini, 2016), (Informatique and Informatique, 2021).

When a horizontal interface is used, it means that in addition to the time needed for the system to reply to a user request locally (such as CPU, memory, and thread utilization), the time required to synchronize with remote sites and provide a final response must also be taken into account. The synchronization process will create a synchronization overhead affecting the system's performance. The technical execution of the solution may cause this latency and will affect the total time required to provide an inter-site service, which affects the system responsiveness (Ahmad and Mir, 2021), (Oktianm, et al., 2017), (Espinel Sarmiento, et al., 2021).

The Flat architecture can be logically centralized physically distributed, or fully distributed (Blial, Ben Mamoun and Benaini, 2016), (Espinel Sarmiento, et al., 2021), (as per Fig. 3). Both topologies will be explained in the following subtitles.

Logically centralized physically distributed

In this architecture, each site has one controller responsible for intra- and inter-site operations, as seen in systems such as, HyperFlow (Tootoonchian and Ganjali, 2010), Onix (Koponen, et al., 2010), ONOS (Open Network Operating System (ONOS) SDN Controller for SDN/NFV Solutions, no date), ElstiCon (Dixi, et al., 2014), and Orion (Ferguson, et al., 2021). Whenever a controller creates or modifies a network resource, it broadcasts any changes (e.g., device or link failure) to other controllers through synchronizations. In synchronization, each controller in many domains must share a portion of their local network state with other controllers to build a global network state. However, no standards dictate what information must be shared (Blial, Ben Mamoun and Benaini, 2016), (Oktian, et al., 2017), (Tadros, Mokhtar and Rizk, 2019), (Hoang, et al., 2022), (Alowa and Fevens, 2020), (Levin, et al., 2012).

Distributed hash tables, transactional databases, and partial quorum techniques (Saito and Shapiro, 2010) are a few examples of distributed, replicated storage formats that can be utilized to implement the NOS state distribution and management (Levin, et al., 2012), (Saito and Shapiro, 2010).

As shown in Fig. 3, a logically centralized physically distributed structure allows multiple controllers to share network information, functioning as a single controller. This design mirrors the original SDN proposal, enabling centralized network control logic whereas distributing responsibility across controllers. It allows management from a global network perspective, improving control in SDN environments (Blial, Ben Mamoun and Benaini, 2016), (Tadros, Mokhtar and Rizk, 2019), (Espinel Sarmiento, et al., 2021), (Bannour, Souihi and Mellouk, 2018b).

There are several ways to obtain network information from SDN controllers. First, is polling, where controllers periodically request updates, even if no changes have occurred. Second, the more efficient publish/subscribe method, where controllers only receive updates when changes happen(Oktian, et al., 2017). In addition, a shared distributed data store allows controllers to exchange states (Espinel Sarmiento, et al., 2021).

The shift toward logical centralization of control within the distributed SDN paradigm helps mitigate the complexity of distributed systems. In this context, incorporating a Knowledge Plane into the architecture can leverage various machine learning (ML) techniques, such as Deep Learning. Collecting network knowledge and then utilizing that knowledge to control and manage the network, exploiting the capabilities of SDN logically centralized control (Mestres, et al., 2017).

However, these systems' effectiveness depends on the capabilities provided by the database system. Even that some of them have specific database systems built for them such as ONOS (Open Network Operating System (ONOS) SDN Controller for SDN/NFV Solutions, no date) distributed controllers linked through Atomix system (Atomix, no date), they fall short of issues such as network partitioning (depending on the database system) or data locality awareness (Espinel Sarmiento, et al., 2021).

Fully distributed

In fully distributed architectures such as DISCO (Phemius, Bouet and Leguay, 2014), controllers are both physically and logically distributed. Each controller maintains a local network view and communicates with other controllers only when necessary to exchange service-specific data. This approach reduces communication overhead and alleviates

scalability limitations encountered in centralized designs (Blial, Ben Mamoun and Benaini, 2016), (Espinel Sarmiento, et al., 2021), (Informatique and Informatique, 2021), (Bannour, Souihi and Mellouk, 2018b).

Fully distributed architecture avoids the initial tendency of SDN by giving many controllers different roles throughout the network (Blial, Ben Mamoun and Benaini, 2016). This architecture suits multi-domain heterogeneous environments, especially WANs and overlay networks. It has the ability to function under multiple Autonomous Systems (ASes) operating under various administrative domains in large-scale networks such as the Internet (Bannour, Souihi and Mellouk, 2018b), (Informatique and Informatique, 2021).

In this architecture, intra-domain modules handle core functions such as network monitoring, whereas inter-domain modules manage communication between domain controllers using protocols such as Advanced Message Queuing Protocol or Representational State Transfer APIs for the East-West interface (Almadani, Beg and Mahmoud, 2021), (Ahmad and Mir, 2021), (Espinel Sarmiento, et al., 2021), (Bannour, Souihi and Mellouk, 2018b).

Fully distributed systems provide robustness against network disconnections, as failures affect only part of the infrastructure (Espinel Sarmiento, et al., 2021). Fully distributed solutions face drawbacks such as static division into independent entities, contrasting with David D. Clark's Knowledge Plane theory (Clark, et al., 2003), which advocates centralized management. In addition, network optimization is local, with entities following their own policies rather than serving the overall network's interests (Bannour, Souihi and Mellouk, 2018b), (Informatique and Informatique, 2021).

Fully distributed systems face challenges in dynamic environments, where conflict resolution and tasks such as dynamic IP assignment are more complex. Moreover, in fully distributed architectures, consistency is managed at the application level due to independent local databases. The East-West interface lacks built-in conflict resolution, requiring additional calls to handle conflicts. These solutions focus on read/write operations and database concurrency, which is beyond our scope (Espinel Sarmiento, et al., 2021).

The researchers (Hu, Li and Huang, 2014) evaluate the SDN controller's scalability in handling flow initiation requests. According to their findings, the best scalability points are found in the flat fully distributed SDN controller with a slight gap. The hierarchical model competes for second place. Interestingly, in their experiment, the logically centralized physically distributed SDN controller suffers significantly (Oktian, et al., 2017).

From Table I and the discussion, logically centralized physically distributed architectures are better for management solutions due to the global network view. Applications can access and control the environment from any part of the cluster (Espinel Sarmiento, et al., 2021). However, the synchronization required for consistency introduces overhead, reducing scalability and increasing latency. The degree of synchronization depends on the consistency model used. In the next section, we explore different consistency approaches.

V. Network State Consistency

In distributed SDN architectures, controllers must maintain consistent network information across their data structures (Ahmad and Mir, 2021), (Informatique and Informatique, 2021), (Zhang, Wang and Huang, 2018).

According to the (CAP) theorem, (C) Consistency, (A) Availability, and (P) Partition Tolerance, only two of the three qualities can be met in case of failure (Brewer, 2000), (Seth Gilbert and Nancy Lynch, 2002), (Panda, et al., 2013). A network with several highly available partitions (A and P) specifically results in a lower level of consistency. As a result, this outdated state impacts how applications work correctly. A system with significant consistency (C and P), on the other hand, leads to reduced network availability (Hoang, et al., 2022).

In addition, on a regular working system without considering failure, the choice between consistency and latency has also impacted distributed SDN architecture. This tradeoff with CAP is combined in the novel formulation PACELC (Abadi, 2012)—the PACELC theorem, which stands for Partitioned, Availability, Consistency Else Latency, Consistency. In light of this new formulation, in the case of network partitioning (P), one must choose between availability (A) and consistency (C) in a distributed computer system. Else (E), if the system is functioning normally in the absence of partitions, one must choose between latency (L) and consistency (C) (Abadi, 2012).

These theorems are essential as the distributed SDN controller, specifically, the logically centralized physically distributed architecture, uses the datastore to store the state of the entire network, which needs to be consistent.

The distributed SDN controller will probably inherit the characteristic of the implemented datastore (i.e., whether it is CP or AP) (Oktian, et al., 2017).

The consistency issue arises from the hardness of resolving the update problems in the network. In other words, maintaining consistency depends on the order of operations computed sequence in the execution network devices (Foerster, Schmid and Vissicchio, 2019).

Besides the degradation of application performance, inconsistency can cause other severe problems in the networks, such as isolation and reachability violation, black holes, and forwarding loops (Aslan and Matrawy, 2016), (Poularakis, et al., 2019).

Three consistency approaches can be applied: strong, eventual, and adaptive. Table II compares the consistency models according to Consistency level, Scalability, Availability, State synchronization Overhead, and Latency.

A. Strong Consistency Model

Strong consistency ensures that controllers operate with a consistent global view (Levin, et al., 2012). Strong consistency is based on a blocking synchronization procedure that prevents switches from reading data till the controllers are fully updated. It reduces the scalability and availability of the network and limits the system's responsiveness (Levin, et al., 2012), (Informatique and Informatique, 2021).

Table II
Consistency Models for Logically Centralized Physically Distributed SDN Architecture Comparisons

Consistency model	Consistency Level	Scalability	Availability	State Synchronization overhead	Latency
Strong consistency	Very high	Low	Low	Very High	High
Eventual consistency	Low	High	High	Medium	Medium
Adaptive consistency	Medium	High	High	Low	Low

Applications requiring high reliability, such as security-sensitive systems (e.g., firewalls), rely on strong consistency to prevent unauthorized traffic and ensure data accuracy (Aslan and Matrawy, 2016), (Espinel Sarmiento, et al., 2021), (Foerster, Schmid and Vissicchio, 2019).

However, most studies on distributed SDN networks focus on strong consistency (Hoang, et al., 2022). Network application developers may decide not to design their programs to be strongly consistent for various reasons. Strong consistency increases latency and reduces availability, which is problematic in dynamic networks with frequent node failures. Applications that require low latency or high availability often favor more flexible consistency models to maintain performance and responsiveness (Aslan and Matrawy, 2016).

B. Eventual Consistency Model

Eventually, consistent designs incorporate information as it becomes available and eventually reconcile modifications as each domain has knowledge of them. In other words, all controller copies will "eventually" converge over time and achieve global network view consistency. This resolves the problem of blocking during the synchronization period, which is encountered in strong consistency (Levin, et al., 2012), (Informatique and Informatique, 2021), (Sakic, et al., 2017).

Accordingly, controllers can handle higher update rates and react quicker, but they can temporarily have inconsistent network views, which could lead to inappropriate application behavior (Levin, et al., 2012), (Informatique and Informatique, 2021), (Saito and Shapiro, 2010). Many emerging applications for SDN controller platforms with high availability and scalability on a large scale choose eventual consistency (Informatique and Informatique, 2021).

C. Adaptive Consistency Models

Adaptively consistent architecture for distributed SDN controllers transforms the inconsistency issue into an automatic control in which the adaptivity module will automatically adjust the value of the synchronization period according to the performance of the target application. The adaptivity module utilizes a feedback loop from measurement or/and prediction extracted data from the fluctuating network environment (Aslan and Matrawy, 2016). In other words, the state synchronization takes place in accordance with performance and consistency restrictions established by the application at runtime by employing triggers according to specified thresholds to enable dynamic change of a consistency level (Bannour, Souihi and Mellouk, 2018a), (Sakic, et al., 2017). A system can use adaptive consistency to deploy applications that tolerate some inconsistency.

From Table II, whereas a strongly consistent network state leads to increased overhead, a weakly consistent network state will produce good performance but less accurate network functioning. Adaptive consistency is an effort to create a solution between the two consistency extremes.

VI. Distributed SDN: Future Direction

Standard static eventual consistency is commonly used for logically centralized physically distributed SDN Control. It suggests a method of synchronization process at fixed intervals, such as the one used by Orion (Ferguson, et al., 2021), in modern SDN systems to achieve effective scalability. It is argued that it provides no bounds on the tolerated state inconsistencies by SDN applications (Bannour, Souihi and Mellouk, 2018a). The fixed synchronization periods may result in utilizing outdated data, disrupt the network, and cause the application to perform less well by sending unnecessary synchronization messages.

Adopting the idea of adaptive consistency in SDN controllers is necessary to leverage administering solutions with a logically centralized physically distributed approach. Adaptive consistency addresses the limitations of static eventual consistency and strong consistency approaches (Aslan and Matrawy, 2016), (Bannour, Souihi and Mellouk, 2018a). This model combines the idea of eventual consistency with a cost-based approach to adjust the consistency level based on observed state convergence and the inefficiencies caused using stale state as inputs (Sakic, et al., 2017).

An adaptive controller can be defined as a controller that has the ability to dynamically and autonomously adjust its configuration to reach a predetermined level of performance based on its requirements and measured in developed metrics (Aslan and Matrawy, 2016), (Bannour, Souihi and Mellouk, 2018a). Several works in the literature recently introduced adaptive consistency, such as Aslan and Matrawy, 2016, Bannour, Souihi and Mellouk, 2018a, and Sakic, et al., 2017.

From Table II and the discussion, the key advantages and motivations behind adopting an adaptive consistency model for the logically centralized physically distributed SDN architecture are as follows:

Reduced Complexity: By dynamically adjusting the consistency level, the adaptive model reduces the complexity of application development. Developers do not need to explicitly handle synchronization or worry about inconsistencies in real time. The model abstracts away the complexity of consistency management, making it easier to build and maintain applications.

Minimized Controller State Distribution Overhead: In traditional synchronization approaches, controllers often

exchange unnecessary synchronization messages, leading to increased overhead. The adaptive consistency model removes the distribution of unused messages, reducing the synchronization overhead while still maintaining the required level of consistency. This optimization improves the scalability and performance of the system.

Responding to Changing Network Conditions: SDN environments are dynamic, and network conditions can change rapidly. The adaptive consistency model allows controllers to respond quickly to these changes. It ensures that the system adapts its consistency level based on observed convergence rates, enabling controllers to make decisions based on the up-to-date information available.

Reduced Controller-to-Controller Contacts: By adjusting the consistency level based on observed state convergence, the adaptive model can minimize the frequency of controller-to-controller contacts. This reduction in inter-controller communication leads to improved system response time and efficiency.

VII. Conclusion

Distributed SDN architectures offer significant advantages over centralized designs, particularly in managing large-scale networks and addressing challenges such as SPOF, scalability, reliability, and performance bottlenecks. This study has shown that distributed architectures—whether hierarchical, flat, or logically centralized but physically distributed—each have their strengths and tradeoffs based on network needs. Hierarchical architectures improve scalability but introduce higher latency due to reliance on a root controller. Flat architectures enhance reliability and performance but complicate consistency management.

The logically centralized physically distributed architecture offers a balanced approach by combining centralized control logic with a global network perspective. This architecture is more efficient in resolving conflicts at the East-West interface and is better suited for global optimization than fully distributed architectures, which focus on local optimizations in multi-domain environments. However, synchronization overhead between controllers remains a key challenge, impacting scalability and latency.

Our analysis of consistency models—strong, eventual, and adaptive—reveals that strong consistency ensures data accuracy but adds overhead, whereas eventual consistency improves scalability at the cost of temporary inconsistencies. The adaptive consistency model, which dynamically adjusts synchronization levels based on real-time conditions, strikes a balance by reducing overhead and maintaining adequate performance, particularly for applications such as load balancing and routing.

In conclusion, our findings suggest that the logically centralized physically distributed architecture, combined with adaptive consistency, offers the best solution for managing large-scale fluctuating networks by minimizing synchronization overhead and improving scalability and reliability. Future work should further explore dynamic consistency models to better optimize the balance between consistency, scalability, and performance in SDN systems.

Declarations

All authors certify that they have no affiliations with or involvement in any organization or entity with any financial interest or non-financial interest in the subject matter or materials discussed in this manuscript. This study was not funded by any external sources or financially supporting bodies.

Data availability

No datasets were generated or analyzed during the current study.

References

Abadi, D., 2012. Consistency tradeoffs in modern distributed database system design: CAP is Only part of the story. *Computer*, 45(2), pp.37-42.

Ahmad, S., and Mir, A.H., 2021. Scalability, consistency, reliability, and security in SDN controllers: A survey of diverse SDN controllers. *Journal of Network and Systems Management,* 29(1), pp.1-59.

Akyildiz, I.F., 2014. A roadmap for traffic engineering SDN-OpenFlow networks. *Computer Networks*, 71, pp.1-30.

Almadani, B., Beg, A., and Mahmoud, A., 2021. DSF: A distributed SDN control plane framework for the East/West interface. *IEEE Access*, 9, pp.26735-26754.

Alowa, A., and Fevens, T., 2020. Towards minimum inter-controller delay time in software defined networking. *Procedia Computer Science*, 175, pp.395-402.

Aslan, M., and Matrawy, A., 2016. Adaptive Consistency for Distributed SDN Controllers. In: *2016 17th International Telecommunications Network Strategy and Planning Symposium, Networks 2016-Conference Proceedings*. Vo. 1, pp.150-157.

Atomix. Available from: https://atomix.io [Last accessed on 2023 Jun 07].

Bannour, F., Souihi, S., and Mellouk, A., 2018a. Adaptive State Consistency for Distributed ONOS Controllers. In: *2018 IEEE Global Communications Conference, GLOBECOM 2018-Proceedings*.

Bannour, F., Souihi, S., and Mellouk, A., 2018b. Distributed SDN control: Survey, taxonomy and challenges. *IEEE Communications Surveys and Tutorials*, 20(1), pp.333-354.

Blial, O., Ben Mamoun, M., and Benaini, R., 2016. An overview on SDN architectures with multiple controllers. *Journal of Computer Networks and Communications*, 2016, p.9396525.

Brewer, E.A., 2000. Towards Robust Distributed Systems. In: *Proceedings of the Nineteenth Annual ACM Symposium on Principles of Distributed Computing*, p.7.

Chen, M., Ding, K., Hao, J., Hu, C., Xie, G., Xing, C., and Chen, B., 2017. LCMSC: A lightweight collaborative mechanism for SDN controllers. *Computer Networks*, 121, pp.65-75.

Clark, D.D., Partridge, C., Ramming, J.C., and Wroclawski, J.T., 2003. A knowledge plane for the internet. *Computer Communication Review*, 33(4), pp.3-10.

Dixi, A., Hao, F., Mukherjee, S., Lakshman, T.V., and Kompella, R.R., 2014. ElastiCon: An elastic distributed SDN controller. In: *ANCS 2014-10th 2014 ACM/IEEE Symposium on Architectures for Networking and Communications Systems*, pp.17-27.

Espinel Sarmiento, D., Lèbre, A., Nussbaum, L., and Chari, A., 2021. Decentralized SDN control plane for a distributed cloud-edge infrastructure: A survey. *IEEE Communications Surveys and Tutorials*, 23(1), pp.256-281.

Ferguson, A.D., Gribble, S., Hong, C.Y., Killian, C., Mohsin, W., Muehe, H., Ong, J., Poutievski, L., Singh, A., Vicisano, L., Alimi, R., Chen, S.S., Conley, M., Mandal, S., Nagaraj, K., Bollineni, KN., Sabaa, A., Zhang, S., Zhu, M., and Vahdat, A., 2021. Orion : Google's Software-Defined Networking Control Plane Proceedings of the 18th USENIX Symposium on Orion : Google's Software-Defined Networking Control Plane. *Proceedings of NSDI 2021: 18th USENIX Symposium on Networked Systems Design and Implementation*, pp.83-98.

Foerster, K.T., Schmid, S., and Vissicchio, S., 2019. Survey of consistent software-defined network updates. *IEEE Communications Surveys and Tutorials*, 21(2), pp.1435-1461.

Hassas Yeganeh, S., and Ganjali, Y., 2012. Kandoo: A Framework for Efficient and Scalable Offloading of Control Applications. In: *Proceedings of the First Workshop on Hot Topics in Software Defined Networks, HotSDN '12*. p.19.

Hoang, N.T., Nguyen, H.N., Tran, H.N., and Souihi, S., 2022. A novel adaptive East-West interface for a heterogeneous and distributed SDN network. *Electronics (Switzerland)*, 11(7), pp.1-20.

Home-OpenDaylight. Available from: https://www.opendaylight.org [Last accessed on 2023 Mar 14].

Hu, J., Li, X., and Huang, J., 2014. Scalability of Control Planes for Software Defined Networks: Modeling and Evaluation. In: *IEEE International Workshop on Quality of Service, IWQoS*, pp.147-152.

Hussein, A., Chehab, A., Kayssi, A.I., and Elhajj, I.H., 2018. Machine Learning for Network Resilience: The Start of a Journey. *2018 5th International Conference on Software Defined Systems, SDS 2018*, pp. 59-66.

Informatique, S., and Informatique, G., 2021. Extending SDN Control to Large-scale Networks : Taxonomy, Challenges and Solutions To Cite this Version : HAL Id : Tel-03456621 Université Paris-Est Créteil THÈSE Docteur de l' Université Paris-Est Contributions Pour le Contrôle Distribué Dans les Résea.

Jain, S., Kumar, A., Mandal, S., Ong, J., Poutievski, L., Singh, A., Venkata, S., Wanderer, J., Zhou, J., Zhu, M., Zolla, J., Hölzle, U., Stuart, S., and Vahdat, A., 2013. B4: Experience with a Globally-deployed Software-defined WAN. *In: Proceedings of the ACM SIGCOMM 2013 Conference on SIGCOMM*, pp.3-14.

Keshari, S.K., Kansal, V., and Kumar, S., 2021. A systematic review of quality of services (QoS) in software defined networking (SDN). *Wireless Personal Communications*, 116(3), pp.2593-2614.

Koponen, T., Casado, M., Gude, N., Stribling, J., Poutievski, L,Zhu, M., Ramanathan, R., Iwata, Y., Inoue, H., Hama, T., and Shenker, S., 2010. Onix: A Distributed Control Platform for Large-scale Production Networks. *USENIX Conference on Operating Systems Design and Implement*, 10, pp.1-14.

Levin, D., Wundsam, A., Heller, B., Handigol, N., and Feldmann, A., 2012. Logically Centralized? State Distribution Trade-offs in Software Defined networks. *HotSDN'12-Proceedings of the 1st ACM International Workshop on Hot Topics in Software Defined Networks*, pp.1-6.

Mestres, S.A., Rodríguez Natal, A., Carner Marsal, J., Barlet R., Alarcón C., Sole, M., Muntés, M., Meyer, D., Barkai, S., Hibbett, M.J., Estrada, G., Coras, F.T., Ermagan, V., Latapie, H., Cassar, C., Evans, J., Walrand, J., and Cabellos Aparicio, A., 2017. Knowledge-defined networking. *Computer Communication Review*, 47(3), pp.1-10.

Oktian, Y.E., Lee S.G., Lee H.J., and Lam, J.H., 2017. Distributed SDN controller system: A survey on design choice. *Computer Networks*, 121, pp.100-111.

Open Network Operating System (ONOS) SDN Controller for SDN/NFV Solutions. Available from: https://opennetworking.org/onos [Last accessed on 2022 Jun 26].

OpenStack Docs: Distributed Dragonflow. Available from: https://docs.openstack.org/developer/dragonflow/distributed_dragonflow.html [Last accessed on 2021 Nov 21].

Panda, A., Scott, C., Ghodsi, A., Koponen, T., and Shenker, S., 2013. CAP for Networks. *HotSDN 2013-Proceedings of the 2013 ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking*, pp.91-96.

Phemius, K., Bouet, M., and Leguay, J., 2014. DISCO: Distributed Multi-domain SDN Controllers. *IEEE/IFIP NOMS 2014 - IEEE/IFIP Network Operations and Management Symposium: Management in a Software Defined World*.

Poularakis, K., Qin, Q., Ma, L., Kompella, S., Leung, K.K., and Tassiulas, L., 2019. Learning the Optimal Synchronization Rates in Distributed SDN Control Architectures. In: *Proceedings-IEEE INFOCOM*, 2019-April, pp.1099-1107.

Saito, Y., and Shapiro, M., 2010. Optimistic replication. *ACM Computing Surveys*, 37(1), pp.42-81.

Sakic, E., Sardis, F., Guck, J.W., and Kellerer, W., 2017. Towards Adaptive State Consistency in Distributed SDN Control Plane. In: *IEEE International Conference on Communications*.

Seth Gilbert and Nancy Lynch, 2002. Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services. *ACM SIGACT News*, 33(2), pp.51-59.

Tadros, C.N., Mokhtar, B., and Rizk, M.R.M., 2019. Logically Centralized-Physically Distributed Software Defined Network Controller Architecture. In: *2018 IEEE Global Conference on Internet of Things, GCIoT 2018*.

Tootoonchian, A., and Ganjali, Y., 2010. HyperFlow: A Distributed Control Plane for OpenFlow. In: *2010 Internet Network Management Workshop/Workshop on Research on Enterprise Networking, INM/WREN 2010*.

The Open Networking Foundation (ONF), 2019. *Reference Design SDN Enabled Broadband Access.* The Open Networking Foundation, United States.

Yu, H., Qi, H., and Li, K., 2020. WECAN: An efficient West-East control associated network for large-scale SDN systems. *Mobile Networks and Applications*, 25(1), pp.114-124.

Zhang, B., Wang, X., and Huang, M., 2018. Adaptive consistency strategy of multiple controllers in SDN. *IEEE Access*, 6, pp.78640-78649.