مجلـــة جـــامـعـة الســـعيـد للعلـــوم التطـبيقيـة
Al – Saeed University Journal of Applied Sciences
journal@alsaeeduni.edu.ye
Vol (7), No(1), Jun., 2024 المجلد(7)، العـدد(1)، 2024م
ISSN: 2616 – 6305 (Print) ISSN: 2790-7554 (Online)

Hybrid architecture for a scalable Data Center Network Based on Blockchain and SDN

Dr. Abdulmalek Alqobaty
Dept of Computer Science
Faculty of Applied Science
Taiz University – Yemen

Noor Addeen M. Ahmed
Dept of Computer Science
National University - Yemen
nooralramsi51@gmail.com

Received: 14/10/2023 **Accepted: 4/11/2023**

Journal Website: https://journal.alsaeeduni.edu.ye

بنيه هجينة لشبكة مركز بيانات قابلة للتوسع مبنية على البلوكتشين والشبكات المُعرفة بالبرمجيات

د/ عبد الملك القباطي
قسم علوم الحاسوب كلية العلوم التطبيقية
جامعة تعز – اليمن

الباحث/ نورالدين محمد احمد قاسم
تخصص علوم الحاسوب
الجامعة الوطنية – اليمن

الملخص

أصبحت خدمات الإنترنت في الوقت الحاضر جزءا رئيسًا من حياتنا اليومية وكل تلك الخدمات يتم تشغيلها داخل مراكز بيانات في مراكز البيانات ويتم توصيل عدد كبير من الخوادم من خلال شبكات مراكز البيانات الداخلية ((Data center networks (DCNs). لذلك تصبح غالبية حركة البيانات على الإنترنت تمر ايضا داخل مراكز البيانات. تتضاعف حركة المرور في مراكز البيانات الضخمة كل عام إلى عامين تعد تلك الزيادة في حركة البيانات تحديًا لقابلية التوسع لعمليات التوصيل والشبكات البينية داخل مراكز البيانات مع الارتفاع السريع في تطبيقات الويب والحوسبة السحابية خلال العقد الماضي تزايد الاهتمام بتقنيات مراكز البيانات في الواقع تعد قابلية التوسع في شبكات مراكز البيانات الداخلية من أكبر التحديات أحد الحلول للتخفيف من حجم المشكلة يكمن في تغيير طبولوجية شبكة مركز البيانات في هذه الورقة البحثية اقترحنا بنية هجينة لشبكة مركز البيانات تجمع بين SDN ونسيج hyperledger. عملنا على بناء نموذج أولي للحل المقترح حددنا أيضًا بعض مقاييس الأداء من أجل التقييم تتضمن المقاييس زمن تأخير الوصول في الشبكة الإنتاجية، والوقت اللازم للتنفيذ في هذا العمل استطعنا أن نثبت فقط أن الحل المقترح قابل للتطبيق وفي أعمال لاحقة سيتم تناول مناقشة النتائج والأداء والمقارنة مع نتائج الحلول الأخرى.

الكلمات المفتاحية: بنيه هجينة، بيانات قابلة للتوسع، البلوكتشين، الشبكات المُعرفة، البرمجيات.

Hybrid architecture for a scalable Data Center Network Based on Blockchain and SDN

Dr. Abdulmalek Alqobaty
Dept of Computer Science
Faculty of Applied Science
Taiz University – Yemen

Noor Addeen M. Ahmed
Dept of Computer Science
National University – Yemen

Abstract

Nowadays, the Internet services become a main part of our daily life. Almost all of the internet services are run in data centers, where a massive number of servers are connected through intra-data center networks (DCNs). However, the majority of internet traffic is inside data centers. The traffic in hyperscale data centers doubles every one to two years, which presents a scalability challenge for intra-data center interconnects and networking. The rapid rise of web based applications and cloud computing over the last decade has led to a growing interest in data center technologies. Scalability, in DCNs, is one of the foremost challenges. With the advent of cloud paradigm, data centers are required to scale up to hundreds of thousands of nodes. Therefore, Scalability, in DCNs, is one of the foremost challenges. One solution is to change the data center topology to mitigate the problem. In this paper, we have proposed an architecture that combines SDN and hyperledger fabric. We have built a prototype for the proposed solution. We also define some performance metrics for evaluation. The metrics include network latency ,throughput, and execution times. In this paper, we have just proved that the proposed solution is applicable. Discussion of the results, performance and comparison with the results of other solutions will be handled in a subsequent paper.

Keywords: Software-Defined Networks, Data Center, Data Center Networks, Edge Network Nodes, Core Network Nodes, Data Center Network scalability.

1. Introduction

Data center (DC) are stores of shared applications and data equipped with many computers, servers, storage systems, applications, and other related components such as routers, switches, and firewalls. Applications were built in the data centers to provide an access to data records [1-7]. By advent of client/server computing, organizations implemented client/server based applications on LAN's servers that were accessed by local or branch offices users. However, the volumes of data and transactions were growing rapidly, and the requests on the network became more complex. Thus, needs for scalable switching and remote access are came out.

Moreover, a three-tier, model-view-controller (MVC), model is activated and applications were built with web front end, business logic, and backend data stores. All tiers run on racks of servers in data centers. In MVC applications, the most traffic is transferred primarily from users to the application (north-south traffic). However, another traffics move from server to another (East-west traffic) increase because applications distribute their components over many servers or VMs. These situations create one of the most significant challenges for networks [8]. Anyway, it is already supported in modern data centers. Businesses continue deploying new applications creating a huge amount of inter-application communication. Day by day, applications are distributed and modular over racks of servers to support single business process. They require hundreds or even thousands of network connections, and present various challenges to the network. On the other side, industries adopt new technologies such as Software Defined Networks (SDN) [9], Virtualization[10], and Blockchain [11]. These technologies can help to design a scalable architecture for the data center network.

SDN provides greater flexibility and scalability and allows network administrators to manage network services through software. Hence, changes in the network topology can be handled easily and quickly manor based on the network state. Initially, SDN architecture has been designed with a centralized architecture, in which a single controller manages the entire network [12]. The controller is responsible for the connection and routing. It simplifies the management of the network, but it may not scalable

and efficient to handle the burden of environments of large-scale and data center networks.

The main problem of data center network (DCN) comes from the high traffics and inefficient of network administration, and hence DCN may not efficient for super data center. Even SDN architecture with a centralized architecture may not properly solve this problem. We, therefore, believe that distributed controllers design is a better scalable solution for solving supper data center. This paper proposes a new scalable architecture for DCN based on Hyperledger Fabric and SDN.

The main motivation to combine SDN and Hyperledger Fabric is to design a scalable, secure, flexible, and cost-effective DCN architecture. SDN makes manage and scale the network easier and enables flexible to network topology. It provides a greater control over network access and traffic flow. Hyperledger Fabric enables the network to handle increased transaction number. It also provides flexible application development and deployment, and enables the creation of custom applications to meet specific business requirements. In addition, it provides a secure and transparent system for managing transactions.

The combination of the two technologies helps to ensure security of data center against attacks and unauthorized access. SDN enables the consolidation of network hardware, reducing the need for physical equipment configurations, while Hyperledger Fabric allows for efficient and cost-effective transaction processing. Finally, the architecture that based on the SDN and Hyperledger Fabric can provide mechanisms for ensuring high availability and fault tolerance, providing reliable services.

The rest of the paper is organized as follows: Section 2 presents preliminaries overview of Data Center, SDN, Hyperledger Fabric, the Hyperledger Fabric ordering service. Section 3 summarizes the related works that concern SDN with data center, Hyperledger fabric, and integration of Hyperledger fabric with SDN. The proposed architecture is described in Section 4. Section 5 presents Performance analysis and discussion of the proposed architecture. Future works are presented in Section 6. Section 7 concludes the paper.

2. Preliminary

2.1 Data Centers

The infrastructure of a data center (DC) typically includes servers, storage devices, networking equipment, power and cooling systems, and security measures such as fire suppression and physical access controls [3, 4, 6]. The servers and storage devices are housed in racks, which are organized in rows in large data halls. The networking equipment is used to connect the servers and storage devices to each other and to the outside world, typically through high-speed internet connections. The DCs can vary in size from small server rooms to large facilities multiple buildings. In addition, they can be owned, used and operated by single company and can be rented to other businesses.

Applications hosted within a data center can range from simple web applications to complex enterprise applications and machine learning models [1, 2, 7]. The data center provides the infrastructure and services necessary to support these applications (e. g servers, storage systems, databases, load balancers, firewalls, content delivery networks, etc.) It, also, provides the necessary network connectivity to enable users to access the applications over the internet or private networks.

2.2 SDN

Software-Defined Networking (SDN) is an architectural approach to networking that separates the control plane from the data plane, making network management more flexible and dynamic [8, 9, 13, 14]. SDN typically consists of three key layers as shown in Figure 1

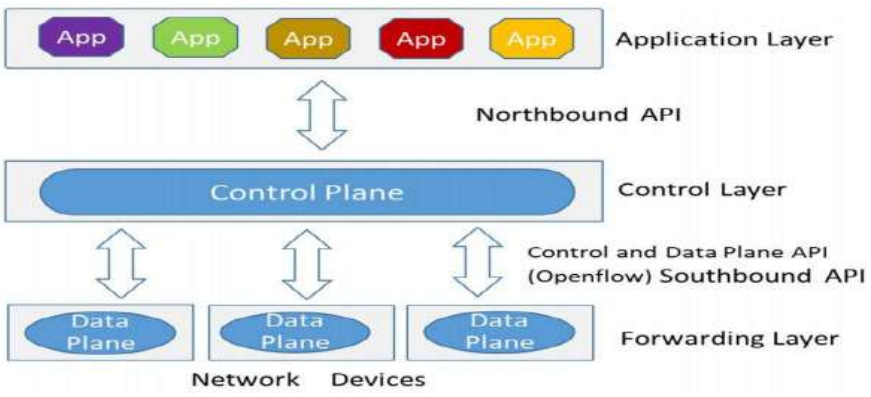

Figure 1: SDN Architecture

The Application Layer:

This layer contains SDN applications, which are responsible for defining network policies and services [15]. These applications communicate with the SDN controller to request network changes based on specific requirements.

The Control Layer:

The control layer is where the SDN controller resides. It is responsible for making global decisions about how traffic should be forwarded in the network [15]. The controller communicates with both the application layer and the Data layer to implement network policies and configurations.

The Data Layer:

This layer comprises the physical and virtual network devices, such as switches and routers. The SDN controller communicates with these devices to configure them according to the policies defined at the application layer. These devices then forward network traffic based on the instructions received from the controller [15].

Indeed, Software-Defined Networking (SDN) supports various types of APIs (Application Programming Interfaces) that enable communication and interaction with the SDN controller and network devices. Some of the key SDN API types include:

- **Northbound APIs:**

These APIs are used by SDN applications to communicate with the SDN controller [14, 17]. The Northbound API can be implemented using REST, Java, or other protocols.

- **Southbound APIs:**

Southbound APIs are used by the SDN controller to communicate with the network infrastructure devices in the data plane [8, 15]. Examples of southbound APIs include OpenFlow and NETCONF [16].

Although SDN provides many benefits, such as centralized control, programmability, and network abstraction, scalability is one of the main challenges facing SDN. However, there are some of the SDN performance and scalability issues that need to be addressed when deploying an SDN network [15]. To address this challenge, SDN controllers

can be distributed across multiple servers or data centers, and load-balancing algorithms can be used to distribute traffic and policies [18]. In addition, when the traffic volume increases, the data plane may become overloaded leading to network congestion. Moreover, SDN networks can be highly dynamic, with new devices and virtual networks being added and removed frequently. This can make the network more complex and too difficult to manage.

2.3 Hyperledger fabric

Hyperledger Fabric is an open-source blockchain framework under the Hyperledger project, designed for building enterprise-grade blockchain applications [3, 6, 7]. Hyperledger Fabric is one of the blockchain frameworks developed under Hyperledger [32]. It is a distributed ledger technology (DLT) that is designed to provide a secure, flexible, and scalable platform for building enterprise-grade blockchain applications. Here's an overview of Hyperledger Fabric, its components, workflow, and ordering algorithms:

Hyperledger Fabric Components:

Fabric is written in Go and uses the gRPC framework [6] for communication between clients, peers, and orders. Most important components of Hyperledger [11]**. Peer Nodes:** Execute smart contracts, maintain a copy of the ledger, and participate in consensus. **Orderer Nodes:** Responsible for managing the ordering of transactions into blocks. **Ledger:** Stores the immutable record of all transactions. The ledger can be implemented using different database technologies like LevelDB or CouchDB [44]. **Membership Service Provider (MSP):** Manages the identity of participants [45]. **Channel:** A private communication channel for transaction confidentiality [44]. **Chaincode:** Smart contracts that define business logic [44]. **Certificate Authorities (CAs):** Issue certificates for participants [45]. **Endorsement Policy:** Specifies the criteria for endorsing a transaction.

Hyperledger Fabric Workflow

The hyperledger fabric's modular architecture and flexibility make it suitable for wide range of enterprise blockchain use case. hyperledger fabric workflow typically involves three distinct phases that occur in sequence:

Transaction Proposal: A client proposes a transaction to peers, which may execute the associated chaincode and return results. **Endorsement:** The results from peers are collected and must meet the endorsement policy's criteria. **Ordering:** Endorsed transactions are bundled into blocks by orderer nodes. **Consensus:** Orderer nodes agree on the order of transactions in the blocks. **Commitment:** The blocks are distributed to peer nodes, which validate and commit the transactions to the ledger. The transaction flow is shown in Figure 2.

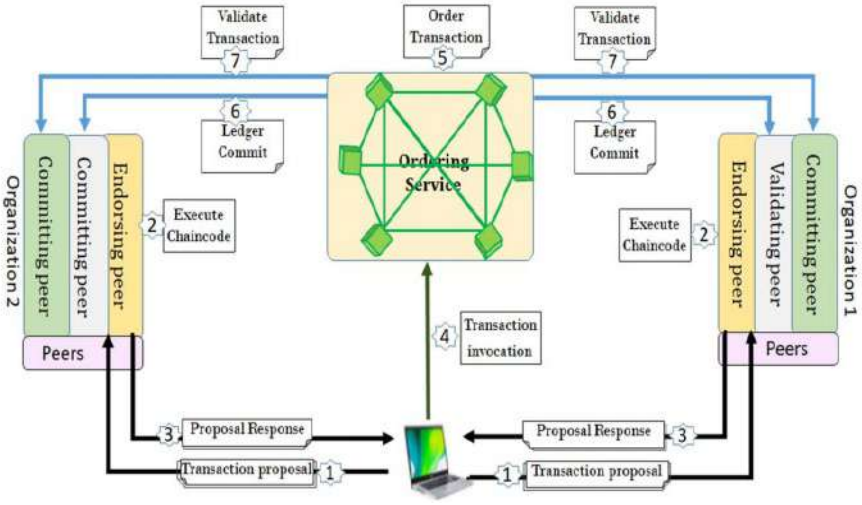

Figure 2: Hyperledger Fabric transaction flow

Ordering Algorithms

Distributed systems use ordering algorithms or consensus mechanisms to establish agreement among nodes regarding the ordering of transactions or events. Hyperledger Fabric uses a consensus mechanism known as Hyperledger Fabric's Consensus Algorithm. The most commonly used ordering algorithms or consensus mechanisms are Practical Byzantine Fault Tolerance (PBFT) [19], Raft (however, Raft is not currently used in Hyperledger Fabric), and Kafka-based ordering (is the recommended ordering service in Hyperledger Fabric). Kafka-based ordering uses PBFT to ensure that all ordering nodes agree on the order of transactions and the state of the ledger [44, 45].

3. Related Work

The integration of Software-Defined Networking (SDN) with data centers has been a topic of interest for researchers and industry experts. Several research works have been done, Abuarqoub, et al. [17] published a survey for scalability of SDN controllers. They have discussed the scalability in SDN architecture including control and data plane separation, request to a single centralized controller and switch-controller communication delay.

Distributed controllers bring advantages for scalable DCN such as load distribution and avoiding single controller failure, many works have tried to create a multi-controller SDN architecture. For examples ONIX [20], Hyper Flow [21], ONOS [22], DISCO [23], ELASTICON [24], KANDOO [25], and ORION [26].

The integration of Hyperledger Fabric with data centers for scalability is a rapidly evolving field, and organizations are continually exploring new approaches to address the challenges of scaling enterprise blockchain networks. Baliga et al. [33] conducted a set of experiments to measure the transaction throughput and latency of Hyperledger Fabric in different network configurations and under different workloads. The results showed that the transaction throughput of Hyperledger Fabric is highly dependent on the number of endorsing peers and the number of channels. Increasing the peers and channels can increase the throughput up to a certain point, after which the performance starts to degrade. Also, the increase of the transaction submission rate can cause higher transaction latency and a higher of transaction failures.

Some studies in the literature have discussed the potential of combining SDN and Blockchain for designed the access control mechanisms. Sharma et al. [37] proposed the use of blockchain technology to provide a distributed and decentralized approach to security. By using a blockchain, the network can maintain a secure ledger of transactions and data transfers, which can be verified by all nodes in the network. The proposed architecture also makes use of SDN. This allows for greater flexibility and scalability in the network, as well as improved security. They, also, proposed a distributed SDN controller, which is responsible for managing the network and enforcing security policies.

Liu et al. [40] argue that traditional networking solutions for IoT lack the necessary scalability, security, and flexibility to support the growing number of devices and applications. They presented a proposed framework, which consists of multiple IoT networks interconnected through a centralized SDN controller, which utilizes blockchain technology to manage network resources and transactions. The integration of blockchain technology with SDN controllers and switches that enabling secure and trusted communication among IoT devices and SDN infrastructure is presented by Tselios et al. [41] The authors discussed the exploration of blockchain technology as a security enhancement for IoT-related SDN deployments.

Alemany, et al. [42] proposed a solution to leverage the programmability and control of SDN to manage network resources and the transparency and immutability of blockchain to enable secure and trustworthy communication between multiple SDN domains.

4. The Proposed Model

We need to achieve scalability of DCN in trust management for the DC. We need to adopt the new technologies features of SDN and Hyperledger to achieve this goal. We build our architecture on top of the Hyperledger Fabric described in the section 2.3 including different business applications and activities as consortium participants.

4.1 The Proposed Architecture

The proposed architecture is shown Figure 3.It is divided into three main parts: Edge Network Nodes (ENNs) layer, Core Network Nodes (CNNs) layer, and the application layer.

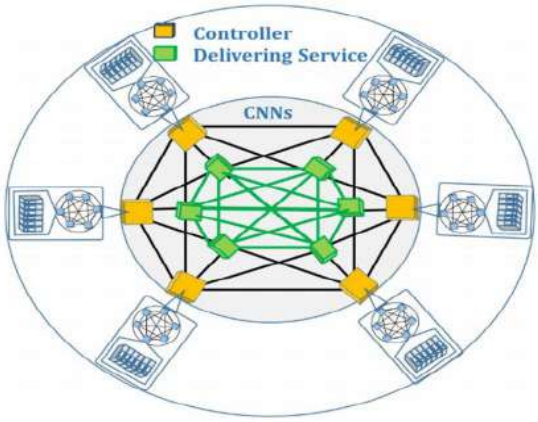

Figure 3: The Proposed architecure

ENNs Layer:

ENNs are a cluster instances that connect the user's machine to the cluster's machines. They allow users to perform their jobs on the edge node rather than the master nodes, which are crucial to the general operation of the system that helps in avoiding capacity losses on master nodes. In the proposed architecture, we group racks into clusters, each cluster corresponds to one application and act as organization. The use of racks cluster is to separate responsibilities in different slice to reduce network overhead. Each organization is controlled by ENN that acts as a service agent for the slice. Each ENN include peers, channels, and certificate authority (CA). It is responsible for maintaining authenticating and authorizing nodes, securing nodes within a cluster communication, and updating the local blockchain ledger. Each ENN includes several components which, can be customized and extended to meet the requirements of different applications. The main components of ENNs are client, membership service provider (MSP) and channels. Client can be any application on a node uses hyperledger fabric SDK or RESTful web services to interact with the hyperledger fabric network by proposing a transaction.

Each application has its own certificate authority (CA). We use Fabric CA for issuing public and private keys and digital certificates. The application has a list of peers to maintain a copy of the ledger, and simulate/endorse transactions by executing chaincode and providing a digital signature for validation.

The Membership Service Provider (MSP) is used to manage the identity and access control of network participants within the hyperledger fabric network. A channel allows specific participants' applications to transact privately and securely without involving all peers in the network. Channels enable the separating transactions and data, ensuring privacy and scalability. Only the members of the channel are involved in consensus, while other members of the network do not see the transactions on the channel. The proposed architecture can be configured with multiple channels, and multiple Applications can join a single Channel or join different channels for data sharing.

CNNs layer:

All ENNs are connected to form a P2P CNNs networks. As shown in Figure 3, the CNNs layer contains multiple controllers, and ordering nodes. The ordering nodes may directly run on a controller or be an independent host that can be accessed by multiple controllers. Each controller only belongs to one application (Organization). It is responsible for collecting data, make decisions, and managing, controlling the underlying sub-network (Application). All controllers issue consensus requests to the ordering nodes.

The ordering nodes run the ordering service. It is responsible for ordering transactions and creating blocks. It receives endorsed transactions from the controllers and orders them into a total order. The ordering service generates blocks containing the ordered transactions and distributes them to the committing peers. This ensures that all peers in the network have the same consistent order of transactions.

The application layer:

The application layer focuses on providing a user-friendly interface, enabling smart contract execution, managing client identities and access, facilitating communication, integrating with SDN for network management, and storing and retrieving data within the block chain network.

4.2 Architecture Workflow

The architecture flow involves several steps and components as shown in Figure 4. We illustrate the steps of the proposed architecture in the following steps.

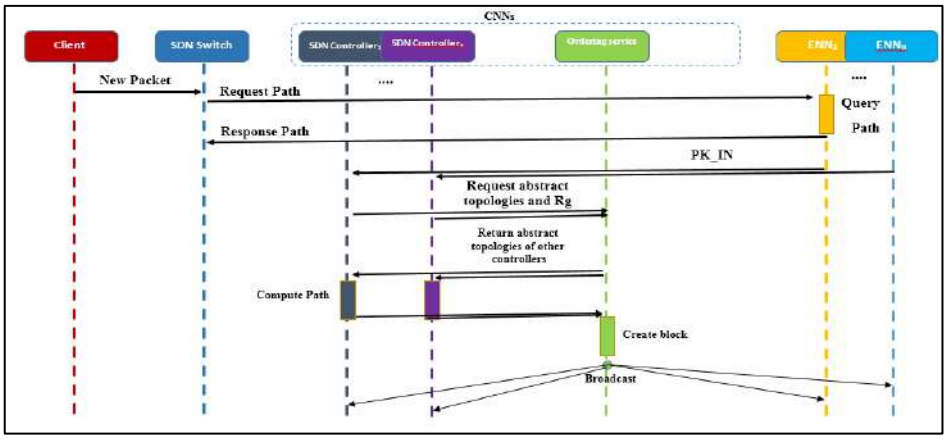

Figure 4: sequence diagram that illustrates the process of the proposed architecture

The request proposal:

When one client want to forward traffic to another client, a request proposal is created and signed by the client then submitted to the network. When the proposal request for a new flow arrives to the network device, the network device will check its flow table for flow rules corresponding to that particular request. If a matching entry for the flow exists, instructions for that specific flow will be executed; otherwise, the request will send to the endorsing peers in the ENN layer as a transaction.

Endorsement:

The endorsing peers in ENN layer receive the transaction and simulate its execution by executing the corresponding smart contract code against the current state. They validate the transaction's correctness, ensuring that it meets the predefined endorsement policies and doesn't violate any rules defined by the smart contract.

Response proposal:

If the transaction is valid, the endorsing peers generate an endorsement signature. Then, the endorsing peer in the ENN layer fetch the route of the proposal request from the blockchain, and delivers the table entries to the corresponding network device automatically, and the following traffic can be transferred according to the new table entries.

SDN Controller Interaction:

If the transaction is invalid, the ENNs communicates with the SDN controller to pass the generated data or instructions. The smart contract code within Hyperledger Fabric processes the transaction request and generates the necessary data or instructions for SDN operations. It can encapsulate the required network configurations, traffic flow rules, or any other relevant instructions.

Ordering Service:

deploy an ordering service that can receive and order updates to the network topology and device status, register all SDN controllers with the ordering service and configure an ordering service to collect and distribute network state information from the controllers. The ordering service utilizes the collected local views to generate a consolidated global view of the network. This consolidated view represents the combined knowledge of the

network's state from all controllers. The consolidated global view is then communicated back to each controller. Each controller now has access to the complete and synchronized view of the network, enabling them to analyze and make decisions based on the network's state.

Path computation:

Once the SDN controller has received the network state information from the Ordering service. It computes the best path for traffic, it can use various algorithms such as Dijkstra's algorithm or the Bellman-Ford algorithm. These algorithms take into account factors such as link bandwidth, link latency, and network policies to determine the most efficient path for traffic and send it as transactions to the ordering service.

Ordering:

The transactions are collected and grouped into blocks. These blocks are sent to the ordering service, which establishes a total order by assigning a unique sequence number to each transaction. After that they deliver all ordered transactions within blocks to all peers on the channel according to the endorsement policy. The ordering service ensures consistency and agreement on the order of transactions across the network.

Validation:

The ordered blocks are distributed to all validating peers in the Hyperledger Fabric network. Each validating peer independently verifies the endorsements, validates the transactions within the block, and ensures that the state changes comply with the smart contract rules and policies.

Consensus:

The validating peers run a consensus algorithm raft to agree on the validity of the block. Consensus ensures that a sufficient number of validating peers reach an agreement on the block's validity before it can be committed.

Commitment:

Once the block of transactions is considered valid and committed, the transaction state changes are applied to the shared ledger, which represents the current state of the network. These state changes can include updates to network configurations, routing tables, or any other network-related data.

4.3 Consensus algorithm

In the proposed design, we used the Raft ordering service to verify the transaction because it has the features of a faster and less complicated consensus. Raft is a crash fault-tolerant (CFT) ordering service built on the etcd Raft protocol implementation [44, 45], We select Raft because the raft-based ordering service is closer to decentralized network, and eventually enables us to have a fully decentralized Fabric network when the BFT ordering service does become available [44]. It is, also, relatively simple to understand and implement. Moreover, Raft provides strong consistency guarantees, which ensures that all nodes have a consistent view of the replicated state. Finally, the ordering service of Raft algorithm can also be used to enforce policies and access control across the network. This ensures the network security and operation within the boundaries of the intended design.

5. Performance analysis and discussion

There are many different types of applications that may need to connect to each other within a data center, depending on the specific requirements and use cases of the data center. Some additional types of applications include Enterprise Resource Planning (ERP) applications, Internet of Things (IoT) applications, Business Intelligence (BI) applications, e-commerce applications, and video and audio streaming applications. The ability to connect these applications effectively is critical to the success of the data center and the applications it supports. For example, E-commerce applications: E-commerce applications are software programs that are used to sell goods or services online. These applications often need to connect to other applications in the data center, such as inventory management systems or shipping and logistics platforms, in order to manage orders and fulfill customer requests.

5.1 Performance measurement parameters

The scalability evaluation of a data center architecture based on Hyperledger Fabric and SDN would involve analyzing various performance metrics to determine how well the architecture can handle increased loads and demands. Some of the metrics that would be important to consider include:

Throughput:

This refers to a mount of transactions that can be processed per unit of time. To evaluate scalability, it would be important to measure how much the throughput increases or decreases as the load on the system increases.

Latency:

This refers to the amount of time it takes for a transaction to be completed. Lower latency is generally better, as it allows for faster processing and response times. To evaluate scalability, it would be important to measure how latency changes as the load on the system increases.

Overall, evaluating the scalability of a data center architecture based on Hyperledger Fabric and SDN requires careful analysis and testing of various performance metrics to determine how well the system can handle increased loads and demands.

5.2 Implementation of the proposed Architecture

We have simulated our proposed model to assess the feasibility of our architecture. All the experiments are on Intel Core i7 CPU 3.40 GHz with 16 GB memory running on Ubuntu Linux v18.04.2 as operating system. We will simulate our proposed model on top of a private Hyperledger fabric network and use Mininet at each edge and orderer nodes to build ONOS SDN-enabled controller nodes. In our simulation, the client connecting to the peer nodes was used to invoke the chain code. So, we used SDK to deploy a blockchain client. We implemented chain codes in Golang and are deployed on all peers. The test codes were written in Golang. We also use the Hyperledger Caliper tool to teste the model.

5.3 Proposed Network Topology

The sample network topology for the Data Center based on Hyperledger fabric and SDN consists of two applications. Each application is act as an organization, Org1, Org2. Org1, Org2 have equal rights over the network configuration. they will set up and initialize a blockchain network. Org1 and Org2 call for a private communication within the channel. Channel 1 (AppC1) is governed according to the policy rules specified in channel policy 1 (AppCP1). It is managed by peer (AP1) and peer (AP2), where smart contract 1 (AppS1) and ledger 1(L1) are hosted. Each application can hold multiple peers in our use case, for example, two peers per application established by Org1 and Org2 for the sake of extra redundancy of data.

5.4 Results and Evaluation

In this section, the performance of our architecture will be analyzed based on the evaluating the scalability of the architecture and testing of various performance metrics to determine execution time, average latency, throughput, and scalability.

5.4.1 Performance Evaluation for single Application (Organization) and Single Peer

A. Throughput

The performance of our proposed model using various metrics such as throughput, as depicted in Table 1 and Figure 5 show the throughput of executing the query () and invoke () functions using different #transactions 10,100,1000 and 10000. The architecture based on fabric has higher throughput for query () function than invoke () function in all number of transactions. We note that the average throughputs between the two functions of implemented platforms increases as the number of transactions increases.

Table 1: Throughput for invoke () and query ()

No. of transactions	Invoke ()	Query ()
10	44	273
100	174	444
1000	155	461
10000	169	490

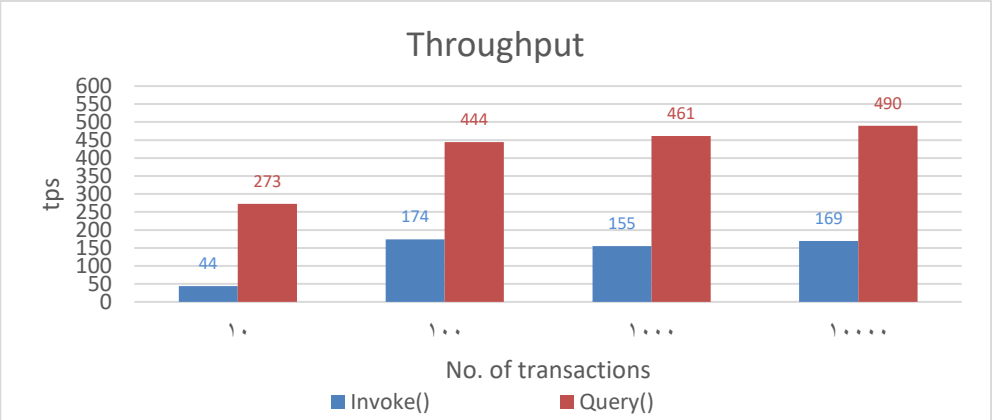

Figure 5: Throughput for invoke () and query () functions

B. Latency

As shown in Table 2 and Figure 6 The average latency of executing the query () and invoke () functions, respectively, using different #Transactions 10,100,1000, and 10000. In the query () function the average latency for our architecture is similar of the average latency for the invoke () function, it can be noted that the average latency for the two functions grow larger when the number of transactions increases.

Table 2: Latency for invoke () and query () functions

No. of transactions	Latency-Invoke ()	Latency-Query ()
10	**0.022**	**0.022**
100	**0.205**	**0.205**
1000	**1.37**	**1.37**
10000	**10.97**	**10.97**

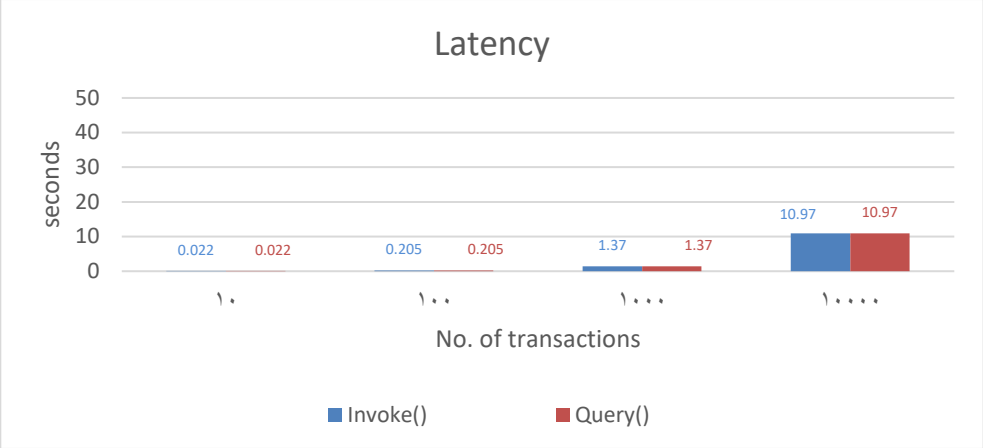

Figure 6: Latency for invoke () and query () functions

C. Execution Time

We evaluated the execution time of our architecture by changing the number of the transactions and analyzing the execution time for the different functions. In general, when the number of transactions grows, the execution times increase. As shown in Table 3 and Figure 7 The execution times of our architecture in query () function increases as the #transactions grow larger. similarly, for the invoke () function, also the executing times increases as the #transactions grow larger. The difference between the

execution times of the two Functions in invoke () function is large compared to the query () function.

Table 3: Execution Time for invoke () and query ()

No. of transactions	Execution Time-Invoke ()	Execution Time -Query ()
10	0.027	0.037
100	0.575	0.225
1000	5.41	2.17
10000	59.17	20.41

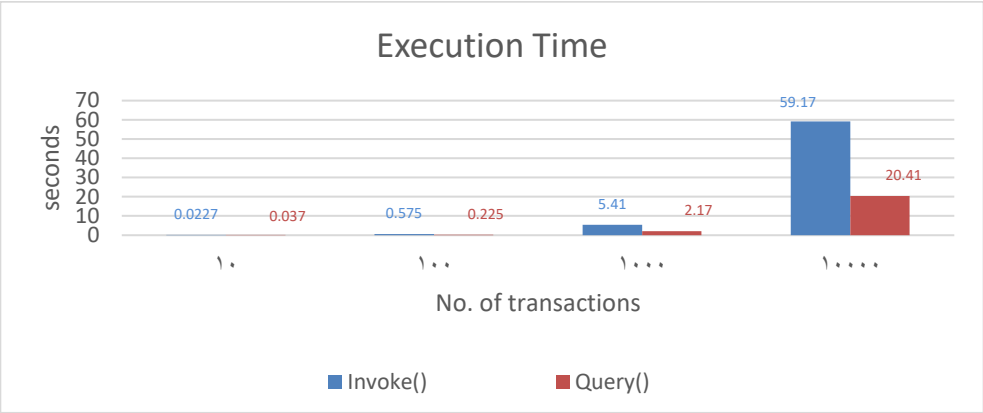

Figure 7: Execution Time for invoke () and query () functions

5.4.2 Performance Evaluation for single Application (Organization) and Multiple peers

In this section, the scalability of the architecture will be analyzed by varying the number of nodes up to 20 nodes, while still measuring the same metrics latency, throughput, and execution time in transactions of 1000-10000. The main goal was to study the effect of increased peer nodes number on the total throughput and average latency. The results showed that that the architecture can handle 1000 concurrent transaction when the number of nodes in the network exceeds 6. The results also showed that the architecture cannot handle 10000 concurrent transactions when the number of nodes in the network exceeds 6.

A. Average Latency

For the architecture the average latency of executing the invoke () function shown in Table 4 and, when the number of transactions is 1000.

The architecture succeeded in executing 1000 transactions when the number of peers in the network is large than 6 nodes. on other hand, when the number of transactions in the dataset is 10000. When the number of peers in the network is less than 6 nodes, the architecture outperforms. The architecture fails to execute 10000 transactions when the number of nodes in the network is greater than 6.

Table 4:Latency of 1000 and 10000 transactions

No. of Peers	Latency of 1000 transactions	Latency of 10000 transactions
1	5.135	59
2	4.229	52.3
4	5.025	53.42
8	4.435	-
12	4.86	-
16	5.9	-
20	3.42	-

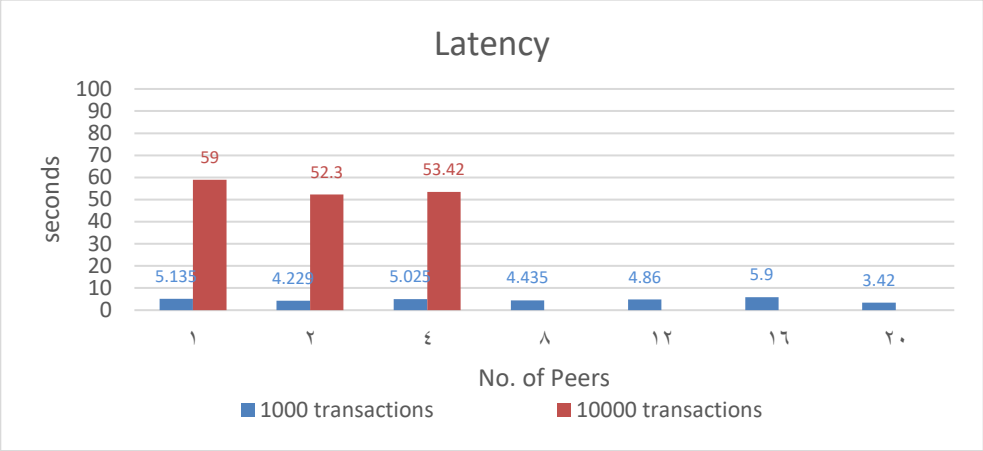

Figure 8 :Latency of 1000 and 10000 transactions

B. Execution Time

the execution time of the invoke () function shown in Figures 16 when the number of transactions in the dataset is 1000 and 10000, respectively. Similar to the average latency results, the architecture succeeded in executing 1000 transactions when the number of peers in the network is large than 6 nodes, but the execution time increases as the number of nodes in the network grows. When the number of transactions is 10000, the

architecture outperforms when the number of peers in the network is less than 6 nodes. As shown in Table 5 and Figure 9 the architecture fails to execute 10000 transactions when the number of nodes in the network is greater than 6.

Table 5: Execution Time of 1000 and 10000 transactions

No. of Peers	Execution Time of 1000 transactions	Execution Time of 10000 transactions
1	5.41	59.17
2	4.329	52.356
4	5.155	53.476
8	4.525	-
12	4.95	-
16	6.024	-
20	3.484	-

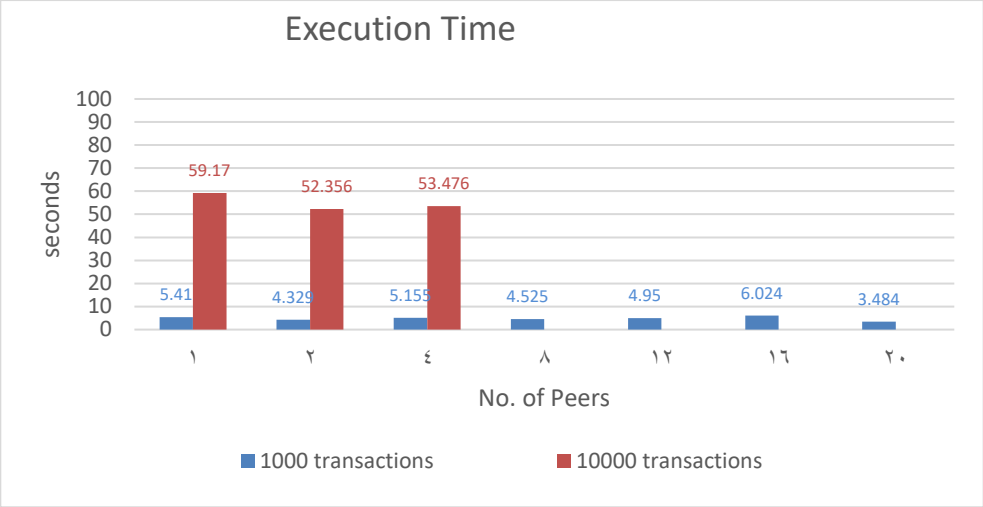

Figure 9: Execution Time of 1000 and 10000 transactions

C. Throughput

Similar to the average latency and execution time results, in terms of throughput in a scaling environment. Table 6 and shows the throughput for the invoke () function when the number of transactions in the dataset is 1000 and 10000, respectively.

Table 6: Throughput of 1000 and 10000 transactions

No. of Peers	Throughput of 1000 transactions	Throughput of 10000 transactions
1	185	169
2	231	191
4	194	187
8	221	-
12	202	-
16	166	-
20	287	-

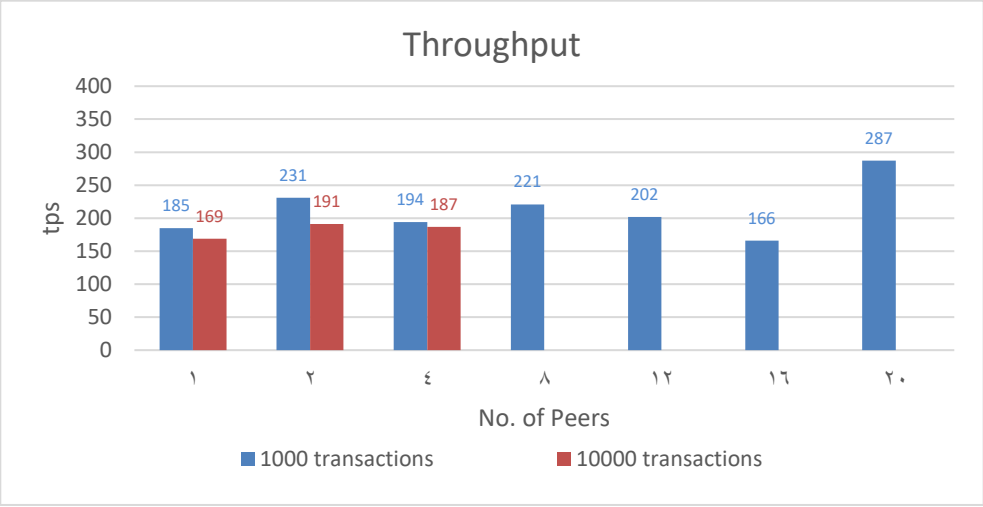

Figure 10: Throughput of 1000 and 10000 transactions

5.4.3Performance Evaluation for Multiple Applications (Organizations) and Multiple peers

To assess the scalability of our architecture, we have also evaluated the scenario when there are multiple applications (organizations) in the network. In this section, the scalability of our architecture will be applied to two Applications (organizational cases) and will be analyzed by measuring the same performance metrics, execution time, throughput, and latency and by comparing them to the results of our architecture with only one Application (organization).

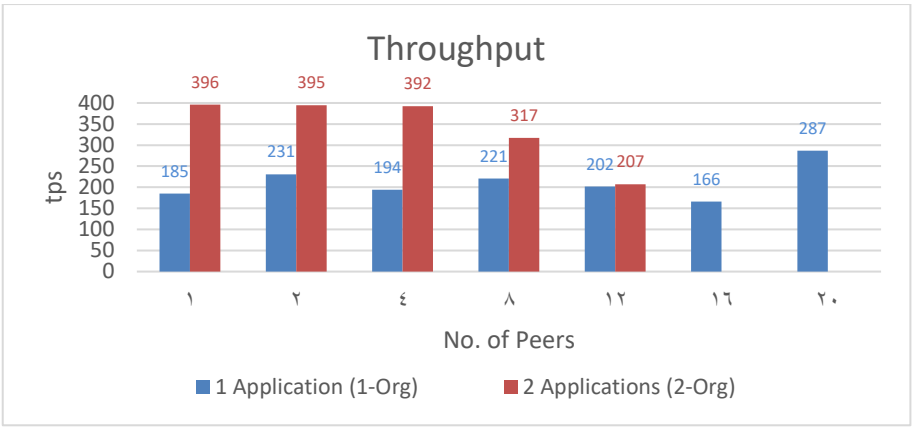

Figure 11: Latency for two applications

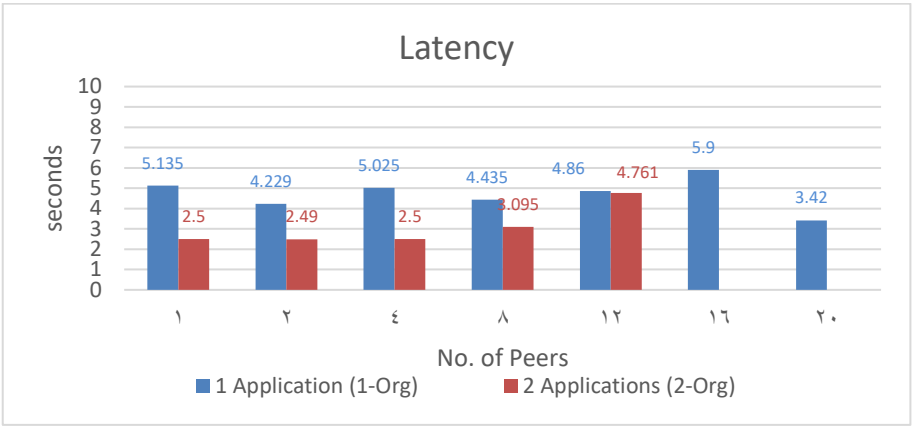

Figure 12: Throughput for two applications

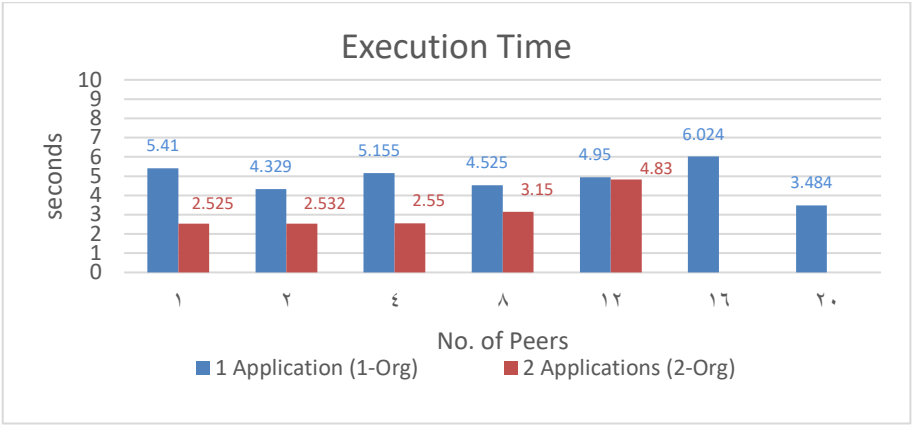

Figure 13: Execution Time for two applications

The architecture with two organizations (applications) has better performance in all evaluation metrics as shown in Figure 11, Figure 12 and Figure 13. The results also show that the architecture can scale up to 26 nodes 13 peers in every organization (application).

6. Future work

The future work of using Hyperledger and SDN for a scalable architecture in a data center involves several exciting areas of exploration and development. Here are a few potential directions for future work:

1. One area of focus is enhancing the performance and scalability of the Hyperledger blockchain network and SDN infrastructure. This can involve exploring techniques such as sharding, off-chain processing, and consensus protocol enhancements to achieve higher transaction rates and improved network performance.
2. Exploring the integration of Hyperledger and SDN with technologies like Internet of Things (IoT), artificial intelligence (AI), or machine learning (ML) can unlock new use cases and opportunities for data center architectures.

7. Conclusion

We believe the use of Hyperledger fabric and Software-Defined Networking (SDN) in the context of a scalable data center architecture brought several benefits and enable efficient management and control of resources. Hyperledger always offers many features such as permissioned networks, smart contracts, and privacy controls, making it suitable for enterprise-level applications. combining Hyperledger fabric and SDN a scalable architecture is achieved in the data center network. It provided some key advantages.

The first one is decentralization and security because Hyperledger's blockchain technology provides a distributed ledger that enhances security and trust by eliminating the need for a central authority. Transactions and data can be securely recorded and verified, reducing the risk of tampering or unauthorized access. The second advantage is efficient resource management. SDN enables centralized control and dynamic allocation of network resources in a data center. It allows administrators to programmatically configure and manage network devices, optimizing

resource utilization and improving overall efficiency. The third advantage is flexibility and scalability. SDN's programmability allows for easy scaling of network infrastructure to accommodate changing demands in a data center environment. It enables rapid provisioning of network services and resources, ensuring that the infrastructure can adapt to evolving requirements. Finally, enhanced network visibility. SDN provides greater visibility into network traffic, allowing administrators to monitor and analyze data flows. This visibility can help identify potential bottlenecks, optimize network performance, and enhance troubleshooting capabilities.

References

[1] Al-Fares, M., A. Loukissas, and A. Vahdat, *A scalable, commodity data center network architecture.* ACM SIGCOMM computer communication review, 2008. 38(4): p. 63-74.

[2] Couto, R.D.S., et al., *Reliability and survivability analysis of data center network topologies.* Journal of Network and Systems Management, 2016. 24: p. 346-392.

[3] Greenberg, A., et al. *VL2: A scalable and flexible data center network*. in *Proceedings of the ACM SIGCOMM 2009 conference on Data communication*. 2009.

[4] Kashif, B., et al. *A Comparative Study Of Data Center Network Architectures*. in *26th EUROPEAN conference on modelling and simulation, ECMS*. 2012.

[5] Niranjan Mysore, R., et al. *Portland: a scalable fault-tolerant layer 2 data center network fabric*. in *Proceedings of the ACM SIGCOMM 2009 conference on Data communication*. 2009.

[6] Singh, A., et al., *Jupiter rising: A decade of clos topologies and centralized control in google's datacenter network.* ACM SIGCOMM computer communication review, 2015. 45(4): p. 183-197.

[7] Yao, F., et al. *A comparative analysis of data center network architectures*. in *2014 IEEE International Conference on Communications (ICC)*. 2014. IEEE.

[8] Yang, H., J. Ivey, and G.F. Riley. *Scalability comparison of SDN control plane architectures based on simulations*. in *2017 IEEE 36th International Performance Computing and Communications Conference (IPCCC)*. 2017. IEEE.

[9] Xia, W., et al., *A survey on software-defined networking.* IEEE Communications Surveys & Tutorials, 2014. 17(1): p. 27-51.

[10] Bhardwaj, A. and C.R. Krishna, *Virtualization in cloud computing: Moving from hypervisor to containerization—a survey.* Arabian Journal for Science and Engineering, 2021. 46(9): p. 8585-8601.

[11] Yaga, D., et al., *Blockchain technology overview.* arXiv preprint arXiv:1906.11078, 2019.

[12] Ahmad, S. and A.H. Mir, *Scalability, consistency, reliability and security in SDN controllers: a survey of diverse SDN controllers.* Journal of Network and Systems Management, 2021. 29: p. 1-59.

[13] Bannour, F., S. Souihi, and A. Mellouk, *Distributed SDN control: Survey, taxonomy, and challenges.* IEEE Communications Surveys & Tutorials, 2017. 20(1): p. 333-354.

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

[15] Bhandarkar, S., G. Behera, and K.A. Khan, *Scalability issues in software defined network (SDN): A survey.* Advances in Computer Science and Information Technology (ACSIT), 2015. 2(1): p. 81-85.

[16] Miguel-Alonso, J., *A Research Review of OpenFlow for Datacenter Networking.* IEEE Access, 2022.

[17] Abuarqoub, A., *A review of the control plane scalability approaches in software defined networking.* Future Internet, 2020. 12(3): p. 49.

[18] Radojević, B. and M. Žagar. *Analysis of issues with load balancing algorithms in hosted (cloud) environments*. in *2011 Proceedings of the 34th international convention MIPRO*. 2011. IEEE.

[19] Zheng, X. and W. Feng. *Research on practical byzantine fault tolerant consensus algorithm based on blockchain*. in *Journal of Physics: Conference Series*. 2021. IOP Publishing.

[20] Koponen, T., et al. *Onix: A distributed control platform for large-scale production networks*. in *OSDI*. 2010.

[21] Tootoonchian, A. and Y. Ganjali. *Hyperflow: A distributed control plane for openflow*. in *Proceedings of the 2010 internet network management conference on Research on enterprise networking*. 2010.

[22] Berde, P., et al. *ONOS: towards an open, distributed SDN OS.* in *Proceedings of the third workshop on Hot topics in software defined networking*. 2014.

[23] Phemius, K., M. Bouet, and J. Leguay. *Disco: Distributed multi-domain sdn controllers*. in *2014 IEEE network operations and management symposium (NOMS)*. 2014. IEEE.

[24] Dixit, A.A., et al. *Elasticon: an elastic distributed sdn controller*. in *Proceedings of the tenth ACM/IEEE symposium on Architectures for networking and communications systems*. 2014.

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

[26] Fu, Y., et al. *Orion: A hybrid hierarchical control plane of software-defined networking for large-scale networks*. in *2014 IEEE 22nd International Conference on Network Protocols*. 2014. IEEE.

[27] Hu, T., et al., *Multi-controller based software-defined networking: A survey.* IEEE access, 2018. 6: p. 15980-15996.

[28] Gu, R., et al., *Software defined flexible and efficient passive optical networks for intra-datacenter communications.* Optical Switching and Networking, 2014. 14: p. 289-302.

[29] Xue, X., et al., *SDN enabled flexible optical data center network with dynamic bandwidth allocation based on photonic integrated wavelength selective switch.* Optics express, 2020. 28(6): p. 8949-8958.

[30] Pranata, A.A., T.S. Jun, and D.S. Kim, *Overhead reduction scheme for SDN-based data center networks.* Computer Standards & Interfaces, 2019. 63: p. 1-15.

[31] Almadani, B., A. Beg, and A. Mahmoud, *DSF: A distributed sdn control plane framework for the east/west interface.* IEEE Access, 2021. 9: p. 26735-26754.

[32] Chakravarthy, V.D. and B. Amutha, *A novel software-defined networking approach for load balancing in data center networks.* International Journal of Communication Systems, 2022. 35(2): p. e4213.

[33] Baliga, A., et al. *Performance characterization of hyperledger fabric*. in *2018 Crypto Valley conference on blockchain technology (CVCBT)*. 2018. IEEE.

[34] Li, W., et al. *Towards scalable and private industrial blockchains*. in *Proceedings of the ACM workshop on blockchain, cryptocurrencies and contracts*. 2017.

[35] Islam, M.A. and S. Madria. *A permissioned blockchain based access control system for IOT*. in *2019 IEEE international conference on Blockchain (Blockchain)*. 2019. IEEE.

[36] Thakkar, P., S. Nathan, and B. Viswanathan. *Performance benchmarking and optimizing hyperledger fabric blockchain platform*. in *2018 IEEE 26th international symposium on modeling, analysis, and simulation of computer and telecommunication systems (MASCOTS)*. 2018. IEEE.

[37] Sharma, P.K., et al., *Distblocknet: A distributed blockchains-based secure sdn architecture for iot networks.* IEEE Communications Magazine, 2017. 55(9): p. 78-85.

[38] Zeng, Z., X. Zhang, and Z. Xia, *Intelligent blockchain-based secure routing for multidomain SDN-enabled IoT networks.* Wireless Communications and Mobile Computing, 2022. 2022: p. 1-10.

[39] Chattaraj, D., et al. *On the design of blockchain-based access control scheme for software defined networks*. in *IEEE INFOCOM 2020-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. 2020. IEEE.

[40] Liu, L., et al. *Bs-iot: blockchain based software defined network framework for internet of things*. in *IEEE INFOCOM 2020-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. 2020. IEEE.

[41] Tselios, C., I. Politis, and S. Kotsopoulos. *Enhancing SDN security for IoT-related deployments through blockchain*. in *2017 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*. 2017. IEEE.

[42] Alemany, P., et al. *Blockchain-based connectivity provisioning in multiple transport sdn domains*. in *2021 International Conference on Optical Network Design and Modeling (ONDM)*. 2021. IEEE.

[43] Fernando, P. and J. Wei. *Blockchain-powered software defined network-enabled networking infrastructure for cloud management*. in *2020 IEEE 17th Annual Consumer Communications & Networking Conference (CCNC)*. 2020. IEEE.