future internet

MDPI

Review

A Review of the Control Plane Scalability Approaches in Software Defined Networking

Abdelrahman Abuarqoub *

Department of Computer Science, Middle East University, 383 Amman 11831, Jordan
* Correspondence: aabuarqoub@meu.edu.jo

Received: 17 February 2020; Accepted: 7 March 2020; Published: 11 March 2020

Abstract: Recent advances in information and communications cloud-based services hold the potential to overcome the scalability and complex maintenance limitations of traditional networks. Software Defined Networking (SDN) surfaced as a promising paradigm to mitigate such limitations while offering flexible networks management. Particularly, SDN separates the control plane from the data plane to achieve abstraction of lower-level functionality, hence, allowing more efficient network management and utilization. However, SDN suffers from various performance and scalability problems leading to significant research efforts on maximizing the scalability of the control plane. This paper aims at reviewing different SDN controller scalability, topology-based and mechanism-based approaches, as well as discussing and analyzing how they attempt to solve the scalability challenge. Furthermore, this paper elaborates on the promising research trends and challenges. Our insights are also discussed to stimulate further research efforts addressing the control plane scalability in SDN.

Keywords: SDN; control plane; scalability; topological-based approaches; mechanisms-based approaches; parallelism optimization; routing-scheme optimization; machine learning optimization

1. Introduction

Emerging network services such as, high bandwidth, dynamic management, cloud computing and virtualization have gained enormous attentions in recent years. Software Defined Networking (SDN), is one of the predominant paradigms that aims to tackle limitations in traditional computer networks architectures in order to satisfy today's complex networking needs [1–4]. Thus, enabling a simplified networks management. One of the main factors affecting this evolution is introducing the concept of separation between network control logic (control plane) from the underlying hardware (data plane) to achieve abstraction of lower-level functionality [5–8].

As shown in Figure 1, SDN architecture encompasses data, control and application planes. Network devices in the underlying network infrastructure may include forwarding devices such as switches, routers, access points and so forth, which enable data transfer between end-users are placed on the data plane. This plane is programmed and managed by the control plane.

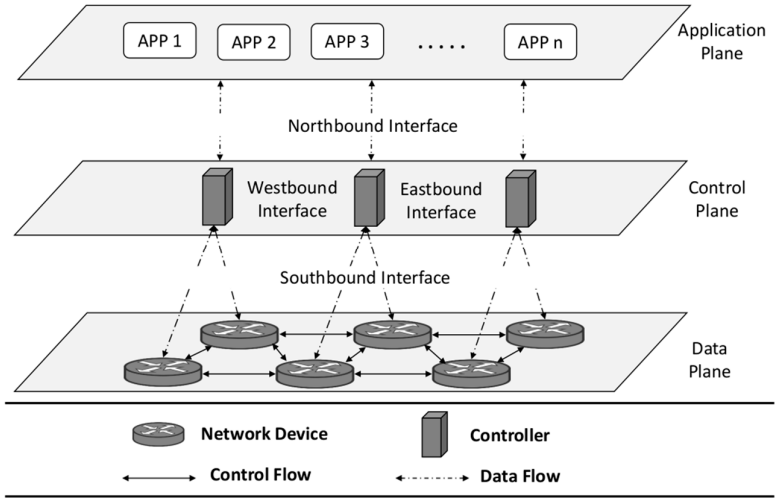

Figure 1. Software Defined Networking (SDN) General Architecture.

Control plane manages flow control and uses control logic to make decisions. The control plane is responsible for setting rules to manage forwarding devices located in the data plane. This communication occurs over the Southbound Application Programming Interface (API).

On the other side, the Northbound API is used for the communication between the control plane and application plane. The task of the controller here is to abstract the infrastructure complexity and to enable simplified applications. If multiple controllers are used in an SDN architecture, the communication across these controllers takes place over an east or west API, depending on the direction.

Although the separation of control functions addresses traditional networks issues and provides network flexibility, it also imposes several inevitable technical challenges to be addressed by researchers. These challenges are related to the control plane scalability, resilience and data consistency among others issues.

Designing a scalable control plane for SDN is one of the most significant challenges [9]. Initially, SDN has been designed with a centralized architecture [10,11], meaning that a single controller is responsible for managing the entire network. The design lacks scalability and does not fulfil the recent large-scale networks requirements. The main drawbacks of this approach is the high latency and causing bottlenecks around the controller. Additionally, a centralized architecture may form a single point of failure, in this manner affecting the network resilience.

Distributed controller's architecture [12–22] is one way to overcome computational load in the centralized design. It distributes the network flows through several controllers according to the topological model. Moreover, this design avoids the single point of failure risk. However, several issues are posed including—latency because of controllers' synchronization; controllers Placement; security issues.

Besides the above topological approaches, there are several existing work on improving the control plane scalability in SDN. Some works adopts mechanisms related approaches, one of these mechanisms is the use of the Parallelism-based Optimization [23–25], in which the performance of the controller is improved through using multi-core systems that uses parallelism multithreading in order to reduce data latency. Another mechanism is the use Routing Scheme-based Optimization [26,27], where the processed event resulted from routing decisions is reduced. These approaches aim at optimizing the routing scheme in terms of their flow tables. Recently, Machine Learning based Optimization [28–31] has emerged to enable prediction of the traffic using machine learning in order to take an intelligent decision and optimize routing.

Consequently, control plane scalability approaches can be classified into two main categories. Firstly, *topology* based scalability where the relation between the network topology (e.g., controllers'

number, locations, etc.) and the scalability issues is investigated. Secondly, *mechanisms* based scalability where the relation between the controllers' optimization mechanisms and scalability issues is investigated. Each of these main categories is further classified as shown in Figure 2. The benefits and shortcomings of different scalability approaches are presented as motivation for future work into optimization of scalability approaches.

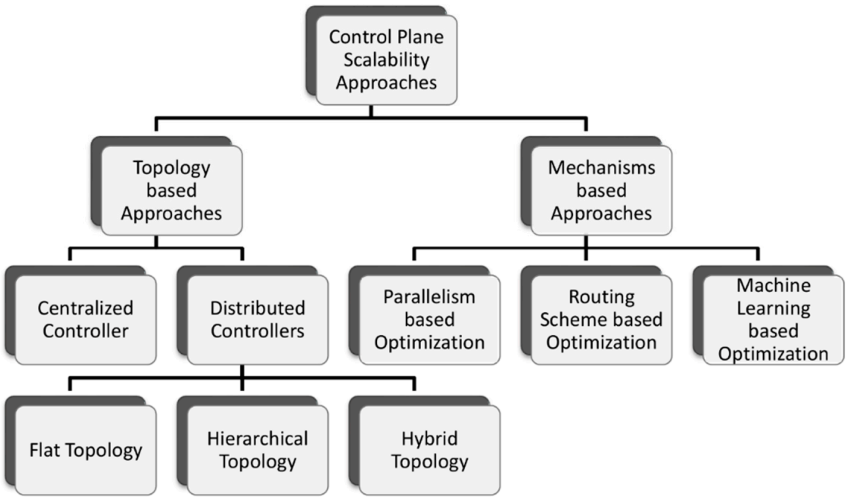

Figure 2. Classification of Control Plane Scalability Approaches.

The aim of this survey paper is to—(1) provide a classification taxonomy of the existing approaches devoted to achieve higher control plane scalability; (2) survey the topology-based and the mechanism-based approaches and evaluate them by their strengths and weaknesses; (3) provide a discussion and analysis on how these approaches attempt to solve the problem; (4) identify gaps and opportunities and elaborate on the promising research trends and challenges.

The rest of this paper is organized as follows. Section 2 provides a review of the recent work of different topology-based scalability approaches. Section 3 reviews and discusses the mechanisms-based optimization approaches. Section 4 provides a thorough discussion over control plane scalability approaches and discusses to inspire further research efforts addressing the scalability in SDN. Section 5 concludes the paper and identify some future work avenues.

2. Topology based Scalability

This section provides reviews of different topology-based scalability approaches. These approaches are classified into two architectures—centralized controller design and distributed controller design. The distributed controller design is further classified into three categories—flat topology, hierarchical topology and hybrid topology. The following subsections provide details on these designs and their respective strengths and weaknesses regarding control plane scalability.

2.1. Centralized Controller Designs

The centralized design creates networks that are easy to manage which can fulfil the requirements of small to medium-sized networks, see Figure 3. However, the design fails in managing more complex network environments like data centers and large-scale networks. This is due to the large amount of network traffic that would overload the controller. Thus, this design is less scalable compared to the distributed designs. Some approaches devoted to address its limitations, the most prominent solutions to this date are Ethane [10] and NOX [11].

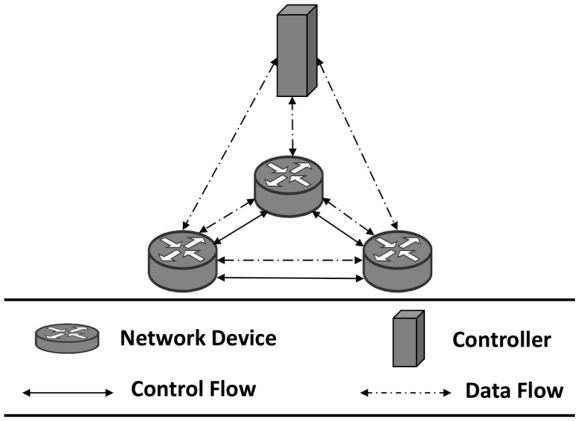

Figure 3. Centralized Controller Design.

Ethane architecture [10] aims at enhancing the enterprise networks by allowing network administrators to define policies where each request that does not match a flow entry has to pass through the controller. Three concerns are addressed by the Ethane architecture. First, high-level names policies should govern the network. Second, in order to improve the network control and view, these policies handles packets follow paths. Third, the network enforces consistent binding between packets and their origin (i.e., user, machine, address, etc.).

NOX operating system [11] was proposed due to the need for a centralized and uniform program interface that would make networks easier to manage. It handles the packets with basis on a flow's first packet traversing through the controller. This approach enables comprehensive control and observation of the network traffic. Researchers have investigated whether NOX and other generalized solutions are able to handle characteristic requirements of specialized networks such as data centers.

Both approaches NOX and Ethane are proposed to deal with the limitations of the centralized design. NOX enabled more manageable network on a single controller, however, the network size that it can handle is relatively small compared to the distributed designs. On the other hand, Ethane enables easy use of centralized controllers in enterprise networks, however, it fails in handling large enterprise networks. Despite the efforts of the researchers that constructed these proposals, it seems that centralized controller designs cannot keep up with the distributed designs in terms of scalability and throughput. Moreover, both approaches do not take into consideration the single point of failure risk. Apparently, depending on a single central controller is a huge risk as the whole network goes offline in case of controller failure.

2.2. Distributed Controller Designs

This section presents the approaches that distribute the workload among several controllers. This design is classified onto three types—flat topology, hierarchical topology and hybrid topology. These designs enable workload distribution and elimination of the single point of failure risk. Although distributed approaches have many benefits, they pose some challenges such as latency due to state synchronization, overhead from controller communication and policy consistency among controllers.

2.2.1. Flat Topology

In the flat network architecture, the network is divided into multiple segments, each of these segments is managed by a controller, see Figure 4. There are two strategies to implement the controller's network view for distributed controller architectures, the view could be global and local. The former allows the controller to view the entire network and the latter only allows local view of the topology where neighboring local networks are abstracted as logical nodes. The communication

between the controllers occurs over channels that relay information about their domains to each other. There are several approaches aim to address the areas of concern of these networks.

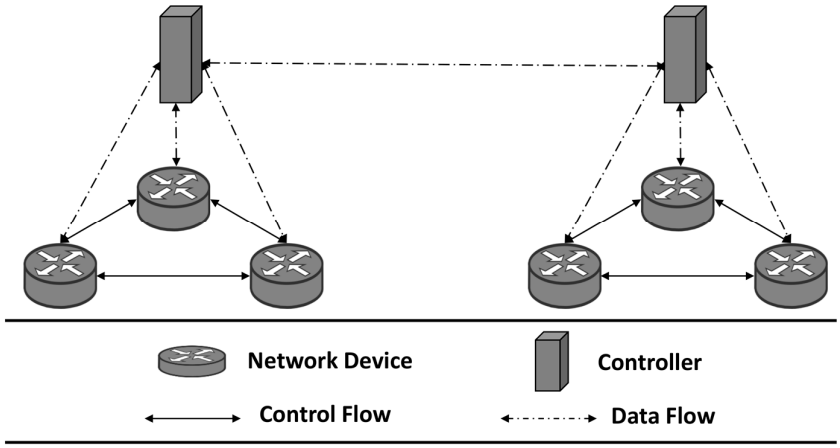

Figure 4. Distributed Flat Controllers Design.

Hyperflow [21] is a distributed event-based control plane for OpenFlow. It was implemented as a modification over NOX and is responsible for synchronizing the network-wide view between controllers, by propagating events. Hyperflow aims to reduce the control traffic needed to synchronize controllers, avoid the chance of conflicts in the state of the application and minimize the burden of modifications in the application.

Onix [22] is a distributed network platform for large-scale production networks devoted to tackle challenges such as operating on a global view of the network, failure recovery mechanisms and, providing consistent network state distribution. Onix focuses on three areas of concern to enhance the scalability. First, ensure that the network states for applications are consistent and durable. Second, Network Information Base (NIB) partitioning by single controller to reduce burden on each controller. Third, cluster aggregation for a hierarchical structure.

ElastiCon [12] is a distributed controller architecture that uses a controller pool to distribute the workload evenly between all the controllers according to traffic conditions. The controller pool is being dynamically managed at all times, which means that the workload can be easily relocated through adding or removing controllers to/from the controller pool.

DISCO [13] (DIstributed Sdn COntrol plane) is an architecture where controllers controlling separate domains exchange their global view information among each other to ensure consistency between the network views of all controllers. There are two main sections of the DISCO framework. The inter-domain section handles the flows between the networks by sharing all the necessary information about the state of the network. The intra-domain part is in charge of the domain functionalities of the controller.

ONOS [14] is a distributed SDN control platform aiming at enhancing scalability, performance and coping with heterogeneity of networks. In order to eliminate bottlenecks and reduce the risk of single point of failure, it scale a network operating system horizontally. It enables large-scale networks to be divided into different segments that can be controlled by different instances of ONOS.

Hyperflow, Onix and DISCO adopt the global view strategy of distributed controller designs. The main benefit of global view strategy is that all controllers work similarly, facilitating setup and management. These approaches address the same main challenge, albeit in different ways, while solving some other problems. The main challenge is global view consistency among all controllers. This is an important concern since some of the controllers may not have up to date information, which could lead to connections latency or even connection failure.

ElastiCon and ONOS adopt the local view strategy of distributed controller designs. The main benefit of local view strategy is that controllers are responsible for their own segments, leading to

even distribution of the workload within controllers. This helps in reducing the connection latency and avoiding controller failure.

2.2.2. Hierarchical Topology

In hierarchical architecture, local controllers deal with local applications requirements which do not require global view of the network, while other applications which require global view of the network are handled by the top controller, also known as Root, see Figure 5. This design differs from the flat design in the network view of the controllers. The controllers in the hierarchical architecture are at the lower tiers which do not have network-wide view, while in the flat designs controllers have network-wide view of the network. The following approaches have been proposed to tackle challenges with this design.

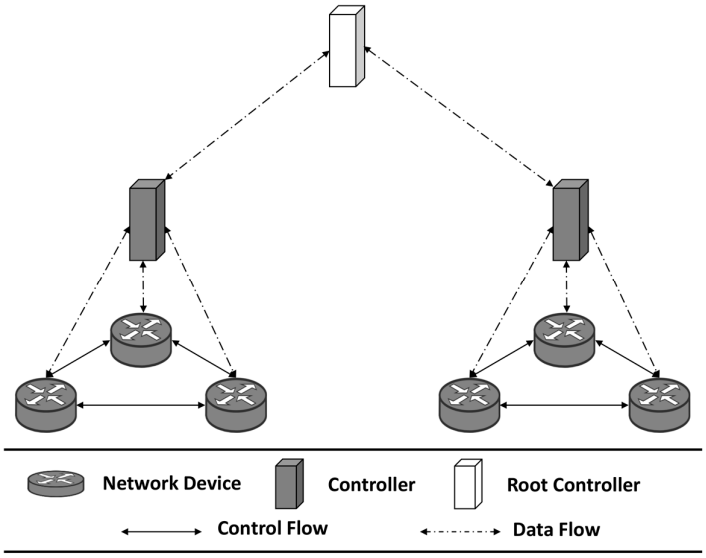

Figure 5. Distributed Hierarchical Controllers Design.

Kandoo architecture [15] aims to scale controllers by reducing the amount of frequent events on the control plane, as these events cause heavy overhead. This architecture consists of two tiers, the bottom tier contains controllers which only communicate with the root controller as they do not have communication links between them. The top tier comprises a root controller that can communicate with the other controllers in the network. In order to reduce the number of events handled by the root controller, local controllers handle all the frequent and resource heavy events such as flow arrivals. The main benefit of this architecture is that it scales better than normal OpenFlow implementation, which is the most used implementation in SDN networks. However, the main drawback is that it is not fully compatible with OpenFlow. FlowVisor has been proposed to enable Kandoo to coexist with OpenFlow. However, Kandoo controllers will not propagate OpenFlow events unless the root controller subscribes to that event.

Logical xBar [16] is a recursive building block used to turn the control plane into a centralized abstract hierarchical structure. It works by aggregating small units and forwarding them onto larger units. It introduces the following two building blocks—the Logical xBar which is the programmable interface responsible for switching packets among ports and the Logical Server which is responsible for control plane computations and managing the forwarding table. This design does not require the physical network to be hierarchical, because by using the principles of Logical xBar the network obtains an abstracted hierarchy. Hierarchical designs suffer heavily from path stretch problems, which have prompted researchers to introduce the following proposals to solve these problems.

Orion architecture [17] enables one administrator to manage the network to alleviate the above challenges. It comprises three tiers—the bottom tier consists of the network devices; the middle tier consists of the controllers and the top tier contains sub-domain controllers. These controllers have

network-wide views of their own network segments. The inter-controller's communication is achieved using a distributed protocol. In order to tackle the path stretch problem, the abstracted hierarchical routing method requires the segment controller to pre-compute the inner hops from every inner switch to all edge switches and send the result to the domain controller. This involves calculations of high computation complexity that may overload the controllers.

FlowBroker architecture [18] aims to enhance the efficiency of communication between different domains in terms of load balancing and network performance. The hierarchy contains one or more super-controllers (Brokers) at the top of the hierarchy and domain controllers at lower levels. Each domain controller can connect to one or more Brokers according to the network requirements. This enables collaboration between Brokers, hence make them aware of the abstracted network states which is received from domain controllers residing at lower levels. However, this architecture struggles with large-scale networks.

The above hierarchical architectures are beneficial as they distribute the workload between controllers. However, most of existing approaches do not take into consideration the risk of having a single root controller, since controllers at the lower level can temporarily take over its tasks. Additionally, many hierarchical architectures do not tackle an important problem which is path stretch problems, leading to relatively high latency as the traffic take longer to pass through the network.

2.2.3. Hybrid Topology

This design differs from the flat and hierarchical topologies in a way that the involvement of control plane as well as data plane devices on the decision processing and controlling the network and that is why this design is called hybrid, see Figure 6.

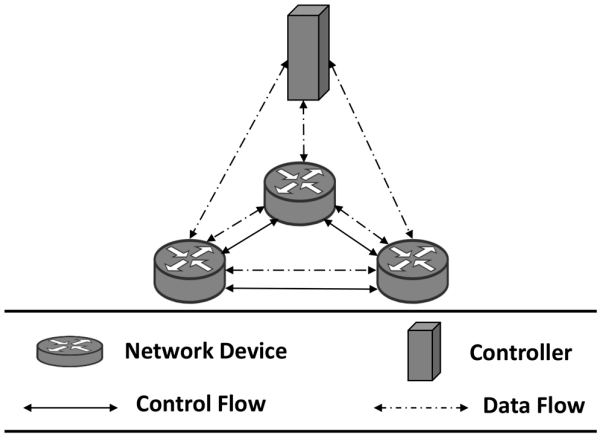

Figure 6. Hybrid Design.

Aiming at improving scalability, this design assigns certain control functionalities to data plane leading to reduction in the workload of the controllers. For instance, some rules can be installed proactively or reactively in the switches. Furthermore, allowing the data plane to handle the flows where possible leads to enhancing the performance in terms of latency and bandwidth utilization.

The DIFANE architecture [32] aims to concentrate traffic in the data plane through assigning the packet management to the "Authority Switches". It contains one controller that maintains an algorithm that divides the switches rules to mitigate rule fragmentation among authority switches. This enables the network to react efficiently to policy changes, topology changes and host mobility.

DevoFlow [19] (Devolved Flow) aims to obtain global view and full control of the network by enabling frequent communication between the data and control planes. In order to reduce the overhead resulted from such communication, DevoFlow assigns some functionalities of the control plane to the data plane. Thus, the controller only has to control significant and long-lived flows. Additionally, wild-cards are used to combine different rules leading to reduction in the

communications required between the controller and the switch. However, DevoFlow has some drawbacks regarding security and traffic engineering.

The Fibbing architecture [20] tackles the traditional distributed link-state protocols and aims to apply a central control over these protocols. The centralized controller is responsible for path computation which is based on the needs of operators. However, the traditional protocols that are implemented on the distributed controller of the network is responsible for the computation of Forwarding Information Base (FIB) entries and installing them on data plane devices. Therefore, Fibbing improves scalability and gets both the advantages of distributed traditional protocols and centralized control.

The hybrid design aims to reduce the overload on the controller. Overloading controllers leads to complications in the processing and causes data latency. By involving data plane devices in the network control, the overload of the controllers can be reduced. However, several existing approaches still suffer from inconsistency between data plane devices.

3. Mechanisms-Related Approaches

In the previous section, different topology-bases scalability approaches have been discussed.

There is other type of scalability approaches exist within the recent literature devoted to exploit different optimization methods in order to mitigate the foregoing scalability issues. The rest of this section present a review of Parallelism-based optimization, Control Plane Routing Scheme-based optimization and Machine Learning-based optimization methods.

3.1. Parallelism-based Optimization

This category is based on the concept of parallelism, where multiple controller solutions exploit parallelism in form of multithreading. This optimization method aims at improving the throughput in terms of traffic by increasing the number of used threads within the controller. Examples for these kind of controllers are NOX-MT [23], Beacon [24] and Maestro [25].

As mentioned previously, the NOX operating system was initially implemented to work in a single-threaded environment. Recently, NOX-MT [23] has been introduced as a successor to work in a multithreaded environment. Aiming at improving the performance of the controller, NOX-MT exploits some optimization methods like multithreading and I/O batching. Leading to increasing the performance of NOX-MT by the factor 33 compared to NOX. NOX-MT operates with tables, for example, for the allocation of MAC addresses to the switches port numbers. Therefore, it is guarantees that packets will be sent on the right path across the network. MAC addresses of new sources will be updated in the table data structure.

Beacon [24] is an open-source controller based on the Java programming language. Beacon has high potential for developers since Java has the advantage of sufficient memory management as well as a clear handling for memory leaks. IBeaconProvider is a widely used Java interface in Beacon to interconnect with the OpenFlow protocol. There are various libraries that enables for instant creating applications to design routing mechanisms on the application plane. It supports several platforms and the high performance complement this mentioned aspect. It improves the network performance through different event handling. Compared to other methods, Beacon uses a shared queue of all packets from the switches. Due to exploiting parallelism in the controller, multiple threads have the ability to perform the buffered requests in the queue. This helps in increasing the amount of handled requests by the controller.

Another parallelism-based optimization approach was Maestro [25]. In order to improve optimization while using multithreading structure, Maestro utilizes batching of packets to individual destinations. Different requests are grouped together in a batch. Multiple threads may then be used to execute the pending requests. As a result, Maestro prioritizes the different requests. In terms of performance, Maestro can handle up to 600.000 requests/sec.

The above three mentioned approaches can be simply be implemented as a single controller in SDN. These approaches aims to scale the network through performance improvements rather than manipulating the network topology. The absence of inter-controllers synchronization mechanisms

also decreases the amount of the traffic passes through the network. However, there are some drawbacks, the single controller design may result in having bottleneck around the controller. Moreover, the single controller is susceptible to the single point of failure problem.

3.2. Routing Scheme based Optimization

Routing scheme-based optimization is another way to improve control plane scalability. It worth explaining how the OpenFlow protocol deals with flows in SDN. Supposing that a host sends a certain request to the switch, the switch will check its local flow table if the packet matches to an existing flow entry in order to send the packet to its right destination. If the switch does not find the entry in its flow table, it forwards the packet to the control plane. The controller then sends a flow modification message to the switch. Upon receiving the message, the switch stores the received entry in its flow table. This enables the switch to handle the same request in the future without contacting the control plane [33]. The following approaches try to optimize the routing scheme in terms of their flow tables.

The RuleTailor optimization scheme [26] enables a controller to handle more requests in a certain amount of time through focusing on high speed flow table updates to achieve a higher performance within SDN. RuleTailor applies various methods to alleviate bottlenecks around controllers like as instruction type transformation, pseudo deletion and match field distance. In order to reduce latency in updating the flow table, RuleTailor uses the above mentioned methods together with priority sorting. Furthermore, it reduces the match field distance enabling faster modifications of the flow table. This enables RuleTailor as a TCAM memory based approach to speed up the performance by the factor of 10 in comparison to previous solutions like Tango.

Another solution is Agg-ExTable [27], which can be classified under the category of routing scheme-based approaches. Since OpenFlow has evolved to the usage of multiple flow tables (MFT), different solutions take advantage of this development. In order to improve the control plane scalability, Agg-ExTable administrates the MFT in a very efficient way. With particular algorithms MFT entries are being regularly aggregated in order to conserve resources, like memory space. Another algorithm attempts to find out frequently used flow entries, so that they can be stored in a frontend ExTable and accessed faster. Therefore, AggExTable is able to save approximately 45% of space in terms of flow tables and represents a good way of resolving the above mentioned issues with MFT.

When thinking about the strengths of the above approaches, it seems that they are more efficient than the parallelism-based approaches. Certainly, it is an advantage to exploit the power of parallelism in order to handle requests within the control plane faster but it does not reduce the network load in terms of operations. However, to modify the routing-scheme allows you to reduce the network load so that less operations has to be performed in order to achieve the same result. This can be used to increase the scalability of the controller.

Despite the advantages of Routing Scheme Based Optimization, it faces some issues which limit the usage of the above approaches. All of them are TCAM-based, a certain kind of memory that has high speed flow table lookup and editing process in the switch. TCAM is considered an expensive solution and many research works are devoted to reduce the usage of TCAM in OpenFlow-based network devices. Additionally, Routing Scheme Based Optimization aims to conserve resources in terms of memory space used to manage flow tables efficiently. Algorithms are used in certain periods to reorder lookup tables and hence enabling fast packet matches. However, the limiting factor to increase the potential of the routing scheme based optimization is the uncertainty about the upcoming traffic.

3.3. Machine Learning based Optimization

Machin Learning is an intelligent methodology that has emerged as promising technique in the domains of classification and prediction [34,35]. In SDN, the flow rule setup causes delays in controllers' response time. Referring to the reactive mode in OpenFlow-based SDN, this delay introduced by the controller is important for its scalability. Some research proposals suggest to

increase the controllers hardware and the memory. Yet, this solution just applies to hardware components. Recently, besides the previously mentioned approach, several proposals suggest bringing machine learning into SDN. In fact, there are many issues in SDN that can be tackles using this technique. However, focusing on the control plane scalability, machine learning can be exploited to deal with this issue using the following ways.

Recently, the authors of Reference [28] proposed Bayesian Machine Learning algorithm for Flow Prediction in SDN Switches. The proposed algorithm enables switches to predict the traffic using machine learning, leading to reduction of the amount of unnecessary communications with the controller. Indeed, the switch infers the type of flow concerning the not matching packets within the flow table. Appropriate rules for these packets can then be applied directly without causing any delay.

Another recent work that tackles control plane scalability using machine learning is DROM [29]. It exploits machine learning to optimize routing and table lookup processes. It applies Deep Deterministic Policy Gradient (DDPG) together with the DDPG Routing Optimization Mechanism (DROM) to achieve higher scalability. DROM uses neural networks to make flow table lookups more efficient in terms of storage usage and hence reduce delays.

To outline the strengths of machine learning based approaches, the powerfulness of the technique has to be mentioned as the first aspect. Analyzing data in real-time condition enables new possibilities to the network management. Smart decisions reducing the network load, lead to a higher performance without causing any overhead while applying the proposed solutions within the network. The different characteristics and components of SDN represent a very good base for applying machine learning techniques to improve routing mechanisms in SDN. The simplified view of the whole network from a centralized control plane enables applications to intervene in the entire network traffic. Although applying machine learning to SDN can enable novel possibilities, it also has some challenges with an increasing complexity of the network. Occurring errors will be difficult to detect inside the process itself and can cause a longer time of unavailability.

4. Discussion and Possible Future Directions

This section provides a discussion about the above reviewed approaches and compares them against each other in terms of throughput and Flow Setup Latency. Additionally, this section provides suggestion on addressing the challenges identified earlier in the paper as well as identifies gaps that could potentially lead to new research. Table 1 lists the reviewed approaches. Table 2 shows a comparison summary of the reviewed control plane scalability approaches.

It can be noted that the last three approaches, the ones concerning parallelism techniques are the ones that offer the best throughput and the least flow setup latency. However, this does not mean that they are the best approaches to implement in all scenarios, because even though the other approaches offer less throughput they have their own advantages.

By investigating the reviewed control plane scalability approaches, it can be noted that the single controller topologies NOX and Ethane offer less throughput than the single controller parallelism approaches, however NOX and Ethane are cheaper solutions. The distributed approaches Onix, ElastiCon and ONOS may offer less throughput but they offer the reduced risk of single point of failure by employing multiple controllers. The hierarchical proposal Kandoo offers high throughput due to the powerful root controller which resides at the top tier of the hierarchy. However, despite being a hierarchical proposal, Orion offers significantly less throughput than Kandoo. This is due to it sacrifices throughput to solve the path stretching problem that afflicts hierarchical architectures. Finally, the hybrid proposal DIFANE offers very high throughput due to the inclusion of rules in the switches that make them more independent. Therefore, they do not need to communicate with the controller every time traffic is traversed.

Approaches such as DIFANE, despite not being a parallelism based approach achieved the highest throughput outside this category, it uses a single controller which would mean that just like the parallelism based approaches and it is susceptible to the single point of failure problem. However, DIFANE addresses that problem by making it so that the switches can compute paths and learn about

topology changes, which makes the switches able to keep the network online even in the case of controller failure. In this regard, no other similar proposals have been found that address the single point of failure while offering such high throughput. This revealed that there are still needs further attention of researchers.

Table 1. Summary of the Control Plane Scalability Approaches.

Category	Controller Scalability Approach	Reference
Centralized Topology	Ethane	[10]
	NOX	[11]
Distributed (Flat Topology)	Hyperflow	[21]
	Onix	[22]
	ElastiCon	[12]
	DISCO	[13]
	ONOS	[14]
Distributed (Hierarchical Topology)	Kandoo	[15]
	Logical xBar	[16]
	Orion	[17]
	FlowBroker	[18]
Distributed (Hybrid Topology)	DIFANE	[32]
	DevoFlow	[19]
	Fibbing	[20]
Parallelism based Optimization	NOX-MT	[23]
	Beacon	[24]
	Maestro	[25]
Routing Scheme based Optimization	RuleTailor	[26]
	Agg-ExTable	[27]
Machine Learning based Optimization	BML Flow Prediction	[28]
	DROM	[29]

Table 2. Performance Comparison of the Control Plane Scalability Approaches.

Controller Scalability Approach	Flow Setup Latency	Throughput	Category
Ethane	Avg 2.7 ms	Up to 11 K	Centralized topology
NOX	Avg 49 ms	Up to 30 K	
ElastiCon	Avg 2.3 ms	Up to 30 K	Distributed (Flat topology)
Onix	Avg 3.5 ms	Up to 200 K	
ONOS	Avg 34 ms	Up to 19 K	
Kandoo	-	Up to 1.3 M	Distributed (Hierarchical topology)
Orion	Avg 15.3 ms	Up to 50 K	
DIFANE	Avg 0.9 ms	Up to 3 M	Distributed (Hybrid topology)
Beacon	Avg 24.7 μs	Up to 12.8 M	Parallelism
NOX-MT	Avg 2 ms	Up to 1.8 M	
Maestro	Avg 55 ms	Up to 3.5 M	

When it comes to control scheme routing based mechanisms we noted that there are two kinds of mechanisms that aim to improve the scalability of the control plane, these approaches use different underlying techniques to solve the problem. Approaching the scalability issue by applying machine learning displayed a very high potential regarding traffic prediction, using real-time data, efficiently organizing flow tables and reducing overhead within the network.

For the future research, there are several SDN control plane research challenges along with existing proposals including:

1. Machine learning will be gathering more attention due to its potential to be used to improve upon already existing proposals. However, existing proposals such as those in References [30,31,36] can still be improved by combining some techniques that have been researched up to this date and incorporating them into a new solution.
2. As controller placement [1,37,38] has a significant impact on the overall performance (i.e., delays, fault tolerance, resiliency, etc.), it is considered one of the challenge that worth more investigations.
3. Flow Rule Setup Latency [39,40] in the context of control plane scalability is a challenge that still needs further attention of researchers.

5. Conclusions

SDN is a predominant paradigm for many networking environments such as data centers, enterprise, cloud networks. The major advantages of SDN are enabling efficient network administration, allowing its programmability and agility and achieving abstraction of lower-level functionality. This is due to decoupling the data plane from the control plane. Although SDN has many benefits regarding network and flow management, it still has many limitations on both performance and scalability. Control plane scalability in SDN is one of the most significant challenges. This paper sheds the light on SDN control plane scalability problem. It begins by providing a classification taxonomy of the existing approaches devoted to achieve higher performance in the control plane. An overview of these approaches is also provided assuming that the scalability challenges of the control plane can be tackled by using methods such as—reducing amount of requests to improve scalability of a single controller or distributing the traffic load between multiple controllers; other methods that look at the problem from different perspective employ various optimization mechanisms including—parallelism-based optimization, routing scheme optimization and more recently, machine learning based optimization. An assessment of the reviewed approaches performance in terms of flow setup latency and throughput is provided. Additionally, open research issues related to control plane scalability were also identified.

Security was common concern among all the reviewed SDN approaches; hence, in future work, we plan to investigate the resilience of various control plane systems against a variety of security attacks [41,42]. In particular, investigating how to leverage SND features to secure the Internet of Things [43,44] presents an opportunity that is of great interest to both research communities. Finally, communication and synchronization between controllers in large-scale systems remains a central area of research in the SDN community.

Funding: This research received no external funding.

Acknowledgments: The author is grateful to the Middle East University, Amman, Jordan for the financial support granted to cover the publication fee of this research article.

Conflicts of Interest: The author declares no conflict of interest.

References

1. Muthanna, A.; Ateya, A.A.; Khakimov, A.; Gudkova, I.; Abuarqoub, A.; Samouylov, K.; Koucheryavy, A. Secure and Reliable IoT Networks Using Fog Computing with Software-Defined Networking and Blockchain. *J. Sens. Actuator Netw.* **2019**, *8*, 15.
2. Ikpehai, A.; Adebisi, B.; Rabie, K.M.; Anoh, K.; Ande, R.E.; Hammoudeh, M.; Gacanin, H.; Mbanaso, U.M. Low-Power Wide Area Network Technologies for Internet-of-Things: A Comparative Review. *IEEE Internet Things J.* **2019**, *6*, 2225–2240.
3. Walshe, M.; Epiphaniou, G.; Al-Khateeb, H.; Hammoudeh, M.; Katos, V.; Dehghantanha, A. Non-interactive zero knowledge proofs for the authentication of IoT devices in reduced connectivity environments. Ad Hoc *Networks* **2019**, *95*, 101988.
4. Hammoudeh, M.; Arioua, M. Sensors and Actuators in Smart Cities. *J. Sens. Actuator Netw.* **2018**, *7*, 8.
5. Bakhshi, T. State of the Art and Recent Research Advances in Software Defined Networking. *Wirel. Commun. Mob. Comput.* **2017**, *2017*, 7191647.

6. Ateya, A.A.; Muthanna, A.; Vybornova, A.; Algarni, A.D.; Abuarqoub, A.; Koucheryavy, Y.; Koucheryavy, A. Chaotic salp swarm algorithm for SDN multi-controller networks. *Eng. Sci. Technol. Int. J.* **2019**, *22*, 1001–1012.
7. Muthanna, A.; Shamilova, R.; Ateya, A.; Paramonov, A.; Hammoudeh, M. A MEC/SDN-enabled architecture for vehicular networks. *Internet Technol. Lett.* **2019**, e109, doi:10.1002/itl2.109.
8. Ateya, A.; Muthanna, A.; Kirichek, R.; Hammoudeh, M.; Koucheryavy, A. Energy-and Latency-Aware Hybrid Offloading Algorithm for UAVs. *IEEE Access* **2019**, *7*, 37587–37600.
9. Yang, H.; Ivey, J.; Riley, G.F. Scalability Comparison of SDN Control Plane Architectures Based on Simulations. In Proceedings of the 2017 IEEE 36th International Performance Computing and Communications Conference (IPCCC), San Diego, CA, USA, 10–12 December 2017; pp. 1–8.
10. Casado, M.; Freedman, M.; Pettit, J.; Luo, J.; McKeown, N.; Shenker, S. ETHANE: Taking Control of the Enterprise. *ACM SIGCOMM Comput. Commun. Rev.* **2007**, *37*, 1–12.
11. Gude, N.; Koponen, T.; Pettit, J.; Pfaff, B.; Casado, M.; McKeown, N.; Shenker, S. NOX: Towards an operating system for networks. *ACM SIGCOMM Comput. Commun. Rev.* **2008**, *38*, 105–110.
12. Dixit, A.A.; Hao, F.; Mukherjee, S.; Lakshman, T.V.; Kompella, R. ElastiCon: An Elastic Distributed SDN Controller. In Proceedings of the 2014 ACM/IEEE Symposium on Architectures for Networking and Communications Systems, Los Angeles, CA, USA, 20–21 October 2014.
13. Phemius, K.; Bouet, M.; Leguay, J. DISCO: Distributed Multi-Domain SDN Controllers. In Proceedings of the 2014 IEEE Network Operations and Management Symposium (NOMS), Krakow, Poland, 5–9 May 2014; pp. 1–4.
14. Berde, A.P.; Gerola, M.; Hart, J.; Higuchi, Y.; Kobayashi, M.; Koide, T.; Lantz, B.; O'Connor, B.; Radoslavov, P.; Snow, W.; et al. ONOS: Towards an Open, Distributed SDN OS. In Proceedings of the Third Workshop on Hot Topics in Software Defined Networking, Chicago, IL, USA, 22 August 2014.
15. Yeganeh, S.H.; Ganjali, Y. Kandoo: A Framework for Efficient and Scalable Offloading of Control Applications. In Proceedings of the First Workshop on Hot Topics in Software Defined Networks, Helsinki, Finland, 13 August 2012.
16. McCauley, J.; Panda, A.; Casado, M.; Koponen, T.; Shenker, S. Extending SDN to large-scale networks. *ONS* **2013**. Available online: http://www.cs.columbia.edu/~lierranli/coms6998-10SDNFall2014/papers/Xbar-ONS2013.pdf (accessed on 10 March 2020).
17. Fu, Y.; Bi, J.; Gao, K.; Chen, Z.; Wu, J.; Hao, B. Orion: A Hybrid Hierarchical Control Plane of Software-Defined Networking for Large-Scale Networks. In Proceedings of the 2014 IEEE 22nd International Conference on Network Protocols, Raleigh, NC, USA, 21–24 October 2014.
18. Marconett, D.; Yoo, S. FlowBroker: A Software-Defined Network Controller Architecture for Multi-Domain Brokering and Reputation. *J. Netw. Syst. Manag.* **2015**, *23*, 328–359.
19. Curtis, A.; Mogul, J.; Tourrilhes, J.; Yalagandula, P.; Sharma, P.; Banerjee, S. DevoFlow: Scaling Flow Management for High-Performance Networks. In Proceedings of the ACM SIGCOMM 2011 Conference, Toronto, ON, Canada, 15–19 August 2011; Volume 41.
20. Vissicchio, S.; Tilmans, O.; Vanbever, L.; Rexford, J. Central Control Over Distributed Routing. *Sigcomm Comput. Commun. Rev.***2015**, *45*, 43–56.
21. Tootoonchian, A.; Ganjali, Y. HyperFlow: A Distributed Control Plane for OpenFlow. In Proceedings of the 2010 Internet Network Management Conference on Research on Enterprise Networking, San Jose, CA, USA, 27 April 2010.
22. Koponen, A.T.; Casado, M.; Gude, N.; Stribling, J.; Poutievski, L.; Zhu, M.; Ramanathan, R.; Iwata, Y.; Inoue, H.; Hama, T.; et al. Onix: A Distributed Control Platform for Large-Scale Production Networks. In Proceedings of the 9th USENIX Conference on Operating Systems Design and Implementation, Vancouver, BC, Canada, 4–6 October 2010.
23. Tootoonchian, A.; Gorbunov, S.; Ganjali, Y.; Casado, M.; Sherwood, R. On Controller Performance in Software-Defined Networks. In Proceedings of the 2nd USENIX Conference on Hot Topics in Management of Internet, Cloud and Enterprise Networks and Services, San Jose, CA, USA, 24 April 2012.
24. Erickson, D. The Beacon Openflow Controller. In Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking, Hong Kong, China, 16 August 2013.
25. Cai, Z.; Cox, A.L.; Ng, T.S. Maestro: A System for Scalable OpenFlow Control. Available online: https://scholarship.rice.edu/bitstream/handle/1911/96391/TR10-11.pdf?sequence=1&isAllowed=y (accessed on 10 March 2020).

26. Zhao, B.; Zhao, J.; Wang, X.; Wolf, T. RuleTailor: Optimizing Flow Table Updates in OpenFlow Switches With Rule Transformations. *IEEE Trans. Netw. Serv. Manag.* **2019**, *16*, 1581–1594.
27. Wang, C.; Youn, H.Y. Entry Aggregation and Early Match Using Hidden Markov Model of Flow Table in SDN. *Sensors* **2019**, *19*, 2341.
28. Baz, A. Bayesian Machine Learning Algorithm for Flow Prediction in SDN Switches. In Proceedings of the 2018 1st International Conference on Computer Applications & Information Security (ICCAIS), Riyadh, Saudi Arabia, 4–6 April 2018; pp. 1–7.
29. Yu, C.; Lan, J.; Guo, Z.; Hu, Y. DROM: Optimizing the Routing in Software-Defined Networks with Deep Reinforcement Learning. *IEEE Access* **2018**, *6*, 64533–64539.
30. Amaral, P.; Dinis, J.; Pinto, P.; Bernardo, L.; Tavares, J.; Mamede, H. Machine Learning in Software Defined Networks: Data Collection and Traffic Classification. In 2016 IEEE 24th International Conference on Network Protocols (ICNP), Singapore, 8–11 November 2016.
31. Audah, M.Z.F.; Chin, T.S.; Zulfadzli, Y.; Lee, C.K.; Rizaluddin, K. Towards Efficient and Scalable Machine Learning-Based QoS Traffic Classification in Software-Defined Network. In *Mobile Web and Intelligent Information Systems*; Springer: Cham, Switzerland, 2019; pp. 217–229.
32. Yu, M.; Rexford, J.; Freedman, M.J.; Wang, J. Scalable Flow-Based Networking with DIFANE. In Proceedings of the ACM SIGCOMM 2010 Conference, New Delhi, India, 30 August–3 September 2010.
33. Silva, W.J.A. Avoiding Inconsistency in OpenFlow Stateful Applications Caused by Multiple Flow Requests. In Proceedings of the 2018 International Conference on Computing, Networking and Communications (ICNC), Maui, HI, USA, 5–8 March 2018; pp. 548–553.
34. Alzu'bi, A.; Amira, A.; Ramzan, N. Compact Root Bilinear CNNs for Content-Based Image Retrieval. In Proceedings of the 2016 International Conference on Image, Vision and Computing (ICIVC), Portsmouth, UK, 3–5 August 2016; pp. 41–45.
35. Alzu'bi, A.; Amira, A.; Ramzan, N.; Jaber, T. Improving content-based image retrieval with compact global and local multi-features. *Int. J. Multimed. Inf. Retr.* **2016**, *5*, 237–253.
36. Mohammed, A.R.; Mohammed, S.A.; Shirmohammadi, S. Machine Learning and Deep Learning Based Traffic Classification and Prediction in Software Defined Networking. In Proceedings of the 2019 IEEE International Symposium on Measurements & Networking (M&N), Catania, Italy, 8–10 July 2019; pp. 1–6.
37. Das, T.; Sridharan, V.; Gurusamy, M. A Survey on Controller Placement in SDN. *IEEE Commun. Surv. Tutor.* **2019**, doi: 10.1109/COMST.2019.2935453.
38. Mohanty, S.; Priyadarshini, P.; Sahoo, B.; Sethi, S. A Reliable Capacitated Controller Placement in Software Defined Networks. In Proceedings of the 2019 3rd International Conference on Computing Methodologies and Communication (ICCMC), Erode, India, 27–29 March 2019; pp. 822–827.
39. Phan, T.V.; Hajizadeh, M.; Khải, N.T.; Bauschert, T. Destination-aware Adaptive Traffic Flow Rule Aggregation in Software-Defined Networks. In Proceedings of the 2019 International Conference on Networked Systems (NetSys), Munich, Germany, 18–21 March 2019; pp. 1–6.
40. Bera, S.; Misra, S.; Jamalipour, A. FlowStat: Adaptive Flow-Rule Placement for Per-Flow Statistics in SDN. *IEEE J. Sel. Areas Commun.* **2019**, *37*, 530–539.
41. Belguith, S.; Kaaniche, N.; Hammoudeh, M.; Dargahi, T. PROUD: Verifiable Privacy-preserving Outsourced Attribute Based SignCryption supporting access policy Update for cloud assisted IoT applications. *Future Gener. Comput. Syst.* **2019**, doi:10.1016/j.future.2019.11.012.
42. Walker-Roberts, S.; Hammoudeh, M.; Aldabbas, O.; Aydin, M.; Dehghantanha, A. Threats on the horizon: Understanding security threats in the era of cyber-physical systems. *J. Supercomput.* **2019**, doi:10.1007/s11227-019-03028-9.
43. Belguith, S.; Kaaniche, N.; Hammoudeh, M. Analysis of attribute-based cryptographic techniques and their application to protect cloud services. *Transactions on Emerging Telecommunications Technologies* 2019, e3667, doi:10.1002/ett.3667.
44. Ande, R.; Adebisi, B.; Hammoudeh, M.; Saleem, J. Internet of Things: Evolution and technologies from a security perspective. *Sustain. Cities Soc.* **2019**, 101728, doi:10.1016/j.scs.2019.101728.