Quality of services in software defined networking: challenges and controller placement problems

Siham Aouad[1], Issam El Meghrouni[2], Yassine Sabri[3], Adil Hilmani[2], Abderrahim Maizate[2]
[1]Smart Systems Laboratory, National School of Computer Science and Systems Analysis, Mohamed V University, Rabat, Morocco
[2]RITM-ESTC/CED-ENSEM, Hassan II University, Casablanca, Morocco
[3]Laboratory of Innovation in Management and Engineering for Enterprise (LIMIE),
ISGA Institut Supérieur d'Ingénierie and des Affaires, Rabat, Morocco

Article Info

Article history:

Received Jul 16, 2023
Revised Nov 9, 2023
Accepted Dec 9, 2023

Keywords:

Controller
Inconsistency
Load balancing
Quality of service
Scalability
Software defined networking

ABSTRACT

Quality of service (QoS) is pivotal for ensuring effective and reliable network performance, yet achieving end-to-end QoS within current network architectures remains a persistent challenge. The emergence of software defined networking (SDN) addresses limitations in traditional networking by offering a centralized control plane. This allows dynamic resource management and efficient enforcement of QoS policies by network administrators. However, the controller placement problem (CPP) within SDN poses a significant challenge, as identifying the optimal placement of controllers is a non-deterministic polynomial-time hardness (NP-hard) problem. Researchers are actively working on solutions to address this challenge, especially in large-scale networks where deploying controllers becomes complex. Additionally, maintaining QoS in terms of controller management presents another hurdle. This paper explores these challenges, delving into the literature and providing a comprehensive analysis of controller performance metrics related to QoS parameters such as load balancing, reliability, consistency, and scalability. By addressing these challenges, the research aims to enhance QoS within the SDN framework.

Corresponding Author:

Siham Aouad
Smart Systems Laboratory, National School of Computer Science and Systems Analysis
Mohamed V University
Madinat Al Irfane, Rabat, Marocco
Email: siham.aouad@ensias.um5.ac.ma

1. INTRODUCTION

Quality of service (QoS) is crucial for ensuring efficient and reliable network performance by providing differentiated treatment to various types of network traffic. Traditional networks have long relied on QoS mechanisms to prioritize critical applications, manage bandwidth allocation, and control network congestion. However, despite their widespread usage, these traditional networks often encounter significant limitations when it comes to achieving optimal QoS levels. A significant drawback of QoS in conventional networks stems from the absence of detailed control and adaptability. These networks typically implement QoS mechanisms at the router or switch level, relying on basic prioritization schemes such as integrated services (IntServ) [1] and differentiated services (DiffServ) [2] or however, these mechanisms often lack the ability to provide fine-grained control over individual flows or applications, resulting in suboptimal allocation of network resources [3]–[7]. Multi-protocol label switching (MPLS) [8] is utilized to simplify routing table lookups through the implementation of labeling techniques. Nevertheless, it is statically configured without the capability for real-time reconfiguration and adaptability. Another significant limitation is scalability. Traditional networks struggle to maintain consistent QoS levels as network traffic increases or changes dynamically. The scalability issue becomes more pronounced when accommodating high-bandwidth applications, multimedia streaming, or the proliferation of internet of things (IoT) devices, all of which demand efficient QoS management. With the emergence of software defined networking (SDN) [9], a potential solution arises to address these issues and open up avenues for the development of novel QoS frameworks.

SDN has arisen as a transformative technology, providing unparalleled flexibility, scalability, and programmability in the administration of networks. Through the separation of the data plane from the control plane, SDN facilitates centralized network control, dynamic resource allocation, and swift service provisioning. However, despite its numerous advantages, SDN faces critical challenges when it comes to ensuring optimal QoS for diverse services and applications. The primary objective of QoS in SDN is to guarantee reliable and predictable network performance, encompassing factors such as latency, bandwidth allocation, packet loss, and overall service availability.

Achieving and maintaining optimal QoS levels in SDN networks is a complex task that requires addressing various challenges, with one of the key challenges being the controller placement problem (CPP). The CPP entails strategically positioning controllers within the network to effectively manage and control network traffic [10]. As an non-deterministic polynomial-time hardness (NP-hard) problem, finding an optimal solution for the CPP becomes computationally demanding, especially in large-scale SDN deployments. The placement of controllers directly impacts QoS-related parameters, including reliability, scalability, consistency, and load balancing. Consequently, thorough investigation and analysis of the CPP and its implications on QoS performance are essential to design efficient SDN architectures. This paper presents an in-depth and evaluative analysis of the obstacles within SDN and the CPP with the aim of improving QoS in SDN networks. The controllers in SDN networks play a crucial role in maintaining QoS by possessing a comprehensive network perspective, necessitating their proper placement. Challenges arising from controllers that can influence QoS encompass aspects like network security, network management, resource utilization, network programmability, and network service management. These challenges can be mitigated by addressing reliability, scalability, consistency, and load-balancing concerns within the SDN framework [11]–[14].

SDN has gained significant attention as a promising research area in modern computer network communication. However, one of the primary challenges within this field is achieving QoS in terms of controller management. Deploying controllers effectively becomes increasingly difficult in large-scale networks, posing challenges related to scalability and reliability, which are crucial for ensuring QoS inprogrammable networks.

To address these challenges, researchers are actively investigating various aspects. They aim to determine the optimal number of controllers required for an SDN infrastructure, identify suitable deployment locations for these controllers, and establish efficient communication between the controllers and attached devices [15]. This research aims to overcome these obstacles and provide answers to fundamental questions surrounding controller management in SDN.

This paper focuses on discussing the main objectives within this research domain, utilizing a taxonomy approach. By examining these aspects, the paper aims to contribute to the understanding and improvement of QoS in SDN networks, ultimately enhancing the overall performance and efficiency of programmable networks. The structure of the remaining paper is as follows: section 2, presents the architectural foundations of SDN and a summary of existing open-source controller platforms and architectural issues related to them. Section 3 explores the research challenges associated with enhancing QoS in SDN networks that utilize multiple controllers. Finally, section 4 concludes the paper, summarizing the key findings and implications.

2. SDN ARCHITECTURE AND CONTROLLER PLATFORMS: AN OVERVIEW

2.1. Architecture of software-defined-networking

SDN is a network architecture that introduces a novel approach to network management and control [16], [17]. Unlike traditional network architectures, SDN separates the control plane from the data plane, thereby enabling centralized control and programmability of the entire network [18]. The architecture of a SDN network is composed of three layers: the application layer, the control layer, and the infrastructure layer. Each layer plays a distinct role in the functioning and management of an SDN network as shown in Figure 1.

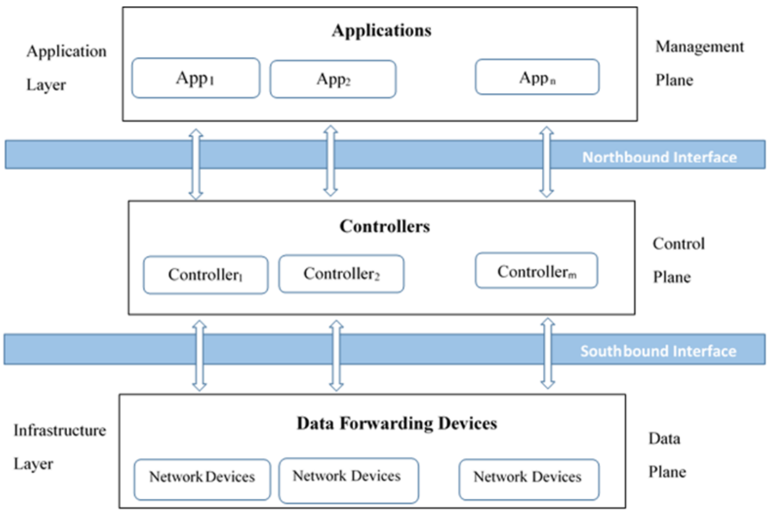

Figure 1. Standard SDN architecture

2.1.1. Application layer

The application layer is the topmost layer of the SDN architecture and encompasses the applications, services, and network management tools that leverage the capabilities provided by SDN. This layer includes a wide range of network applications such as load balancing, traffic engineering, security, and QoS management. These applications interact with the SDN controller through well-defined interfaces to request network services and configure network behavior.

2.1.2. The control layers

The control layer is the central intelligence of the SDN architecture. It comprises the SDN controller, which acts as the brain of the network. The controller communicates with the application layer and the infrastructure layer to orchestrate network behavior. Its primary function is to receive high-level instructions and policies from the applications in the application layer and translate them into low-level network configurations. The control layer is responsible for tasks such as topology discovery, network monitoring, and path computation. It maintains a global view of the network, making it possible to dynamically configure and manage network resources. The controller enforces network-wide policies, handles network events, and distributes the appropriate instructions to the infrastructure layer switches.

2.1.3. Infrastructure layer

The infrastructure layer forms the foundation of the SDN architecture and comprises the physical and virtual network devices, such as switches, routers, and gateways. These devices make up the data plane, responsible for forwarding data packets within the network. In an SDN network, the infrastructure layer switches are typically simpler in nature compared to traditional network devices.

2.1.4. The northbound interface

This interface serves as a mediator for communication between the upper and middle planes, as well as management and control entities. It facilitates the transmission of instructions from the application plane to the controllers. At present, OpenFlow is widely recognized as the prevailing southbound application programming interface (API), while a standardized northbound API is yet to be established. As the development of use cases is still ongoing, it may be premature to define a definitive standard for the northbound API. However, it is anticipated that a common northbound API will emerge as SDN continues to evolve. To fully harness the capabilities of SDN, it is crucial to have an abstraction layer that allows applications applications to operate independently of particular implementations.

2.1.5. The southbound interface

The southbound interface (SBI) plays a vital role as a communication mediator between the control layer and the infrastructure layer. It effectively facilitates the decoupling of these two layers through the utilization of the OpenFlow protocol. Additionally, the southbound API supports various functionalities and protocol plugins like simple network management protocol (SNMP), border gateway protocol (BGP), and network configuration protocol (NetConf). These plugins enable the controller to manage both recently added and pre-existing physical or virtual devices using southbound APIs like OpenFlow, OpFlex, OpenState, and protocol-oblivious forwarding (POF). Currently, OpenFlow is commonly acknowledged as the standard for the SBI in SDN systems.

2.1.6. OpenFlow protocol

The OpenFlow protocol is a key enabler of SDN, providing a standardized interface for communication between the SDN controller and network devices. It defines how the controller can program forwarding rules on network switches or routers, enabling centralized control over network flows [19]. OpenFlow operates on a simple premise: the controller maintains a flow table on each network device, which contains flow entries specifying how to handle specific types of traffic. When a packet arrives at a network device, it is matched against the flow table, and the corresponding action is taken as per the flow entry instructions. This dynamic forwarding behavior allows the controller to control traffic flows, implement QoS policies, and respond to network events in real-time [20].

2.2. Comparison of SDN controllers and architectural issues

2.2.1. Open source controllers platforms: enhancing QoS

A wide range of SDN controllers, both open source and commercial, are readily available. These controllers offer distinct features tailored to specific applications. Broadly speaking, controllers can be categorized as either distributed or centralized. Centralized controllers consolidate the logic of the control plane in a single location; however, they frequently encounter scalability issues owing to restricted capacity. On the other hand, distributed controllers do not encounter scalability issues and provide superior performance, particularly under high traffic loads. Table 1 provides a concise comparison of various controllers based on their distinct features. Beacon [21] is an open-source controller known for its focus on scalability and high-performance networking. Built using Java and employing the OpenFlow communication protocol. It prioritizes QoS parameters such as latency management and bandwidth allocation, ensuring efficient resource utilization and network responsiveness.Beehive [22] is a controller that emphasizes fault tolerance and scalability. It employs distributed architectures to handle large-scale networks, enabling seamless scalability and ensuring uninterrupted network operation. DCFabric [23] is a controller designed for data center networks. It excels in QoS parameters such as bandwidth allocation and latency management, facilitating optimized data transmission and efficient resource utilization within data center environments. Faucet [24] is an open-source SDN controller that emphasizes network security and QoS enforcement. It allows for fine-grained control over network flows, enabling administrators to prioritize and allocate resources based on QoS requirements. FloodLight [25] is a widely used and extensible SDN controller. It offers comprehensive support for QoS parameters, including bandwidth allocation, latency management, and load balancing. FloodLight provides a flexible platform for deploying QoS-aware network.

FlowVisor [26] is a controller that focuses on network slicing and resource isolation. It enables the creation of virtual networks with dedicated QoS parameters, allowing for customized resource allocation and isolation for different network slices. Kandoo [27] is a controller designed for multi-tenant environments. It ensures QoS by providing isolation and resource allocation mechanisms for different tenants, enabling efficient management and control of network resources. Loom [28] is programmed in Java, Loom emphasizes QoS parameters such as scalability, fault tolerance, and network management. Maestro [29] is a controller that prioritizes scalability and fault tolerance. It employs distributed control plane architectures and advanced load balancing techniques to ensure high performance and QoS in large-scale networks. NOX [30] is an early SDN controller that offers flexibility and programmability. While it may lack some advanced QoS capabilities, it provides a foundation for developers and researchers to investigate and incorporate features related to QoS. Onix [31] is a controller known for its focus on network programmability and flexibility. It enables the deployment of customized QoS policies and allows for fine-grained control over network behavior.

Open network operating system (OnoS) [32] is an open-source controller that supports QoS parameters such as latency management, bandwidth allocation, and network reliability. It offers a modular architecture, allowing for easy integration and customization of QoS-related features. OpenContrail [33] is a controller specifically designed for SDN in cloud environments. It emphasizes network virtualization and QoS-aware resource allocation to ensure optimal performance and isolation for different cloud tenants. OpenDaylight [34] is a widely adopted open-source SDN controller that provides extensive support for QoS parameters. It offers a modular and customizable framework, enabling the implementation of various QoS-related features and applications. OpenMUL [35] is an open-source controller designed for multi-layer networks. It focuses on QoS parameters such as bandwidth allocation, latency management, and network reliability, ensuring efficient resource utilization across multiple network layers. POX [36] is a lightweight and extensible SDN controller. While it may lack some advanced QoS capabilities out of the box, it provides a foundation for developers to implement and customize QoS-related features based. RYU [37] written in Python and employing OpenFlow, RYU addresses QoS parameters such as flexibility, programmability, efficient network management, and the ability to develop custom applications. These controllers offer various features, programming language choices, and communication protocols, providing flexibility and enabling efficient management of QoS parameters in SDN environments.

Table 1. SDN open source controller's comparaison

Controller name	Architecture	Multithreading	Programming language	Modularity	Consistency	Fault	Scalability
Beacon [21]	Centralized	Yes	Java	Fair	No	No	Yes
Beehive [22]	Distributed Hierarchical	Yes	Go	Good	Yes	No	No
DCFabric [23]	Centralized	Yes	C, Javascript	Good	Yes	Yes	Yes
Faucet [24]	Centralized	Yes	Python	-	Yes	Yes	Yes
FloodLight [25]	Centralized	Yes	Java	Fair	Yes	No	Yes
FlowVisor [26]	Centralized		C	-	No	No	No
Kandoo [27]	Distributed Hierarchical	Yes	C, C++, Python	High	No	No	No
Loom [28]	Distributed	Yes	Erlang	Good	No		
Maestro [29]	Centralized	Yes	Java	Fair	No	No	Yes
NOX [30]	Distributed	Yes	C++	Good	No	No	Limited
Onix [31]	Distributed	Yes	C++	Good	No	Yes	Yes
ONoS [32]	Distributed	Yes	Java	High	Yes	Yes	Yes
OpenContrail [33]	Centralized	Yes	C, C++, Python	High	Yes	No	Yes
OpenDaylight [34]	Distributed	Yes	Java	High	Yes	No	Yes
OpenMUL [35]	Centralized	Yes	C	High	No	No	Yes
POX [36]	Centralized	No	Python	Low	No	No	Yes
RYU [37]	Centralized	Yes	Python	Fair	Yes	No	Yes

2.2.2. Architectural issues in controller-based SDN networks

Architetural issues play a essential role in the design and implementation of controller-based SDN networks. With the advent of SDN, network architectures have undergone significant transformations, offering enhanced programmability, flexibility, and control. However, the architectural decisions in controller-based SDN networks can impact the QoS delivered to network applications and users. Enhancing network performance remains a subject of ongoing research, focusing on both the design and placement of controllers. Additional concerns such as adaptability, scalability, latency, security, and consistency also hold significant importance [14].

2.2.3. Scalability and resource management

Scalability is a key architectural challenge in controller-based SDN networks. As the network expands in complexity and size, the controller's capacity to handle an increasing number of network devices and flows becomes critical. Effective resource management techniques, such as load balancing, distribution of control plane functions, and intelligent allocation of computational resources, are necessary to maintain scalability and ensure optimal QoS [38], [39].

2.2.4. Centralized administration

Within the realm of SDN, the advent of a centralized control plane necessitates a heightened focus on effective administration and management. This shift brings forth architectural challenges, requiring the development of a robust and scalable framework for the centralized management of the network. This framework must address multifaceted aspects, encompassing configuration management to ensure network elements are appropriately set up, policy enforcement for maintaining compliance with defined rules, security measures to safeguard against potential threats, and comprehensive system monitoring to track and optimize overall performance. Successfully navigating these challenges is imperative to harness the full potential of SDN, ensuring not only the efficient execution of network tasks but also the resilience and security of the network infrastructure as a whole [40].

2.2.5. Controller placement problem

Determining the optimal placement of controllers in an SDN network is a critical architectural issue. Placing controllers strategically across the network can impact the overall efficiency, scalability, and fault tolerance of the system. Finding the right balance between the location and number of controllers is crucial for efficient network management [41], [15].

2.2.6. Controller inconsistency

In expansive SDN deployments featuring multiple controllers, the imperative of ensuring consistency and synchronization between these controllers emerges as a pronounced architectural challenge. The coordination of state and policies across distributed controllers demands meticulous attention to maintain a cohesive and unified perspective of the network. This challenge becomes particularly formidable in the face of dynamic network changes or policy updates, where the need for real-time coordination intensifies. Addressing this issue necessitates the development of robust mechanisms for communication and synchronization, enabling controllers to seamlessly exchange information and collectively adapt to evolving network conditions. In essence, the effectiveness of SDN in large-scale deployments hinges on the ability to overcome these challenges, ensuring not only the consistency of network-wide policies but also the responsiveness of the entire SDN ecosystem to dynamic shifts in network behavior [42].

2.2.7. Communication protocol

In the realm of SDN architectures, the establishment of efficient and reliable communication protocols among controllers stands as a cornerstone for ensuring seamless operation and effective coordination [40]. This imperative extends to the challenge of designing a robust protocol capable of facilitating the efficient exchange of control information, timely event notification, and seamless synchronization between diverse controllers. The choice and implementation of such a protocol significantly influence the overall performance and responsiveness of the SDN environment. Successfully addressing this architectural challenge is pivotal in cultivating a network infrastructure that can adapt dynamically to changing conditions, fostering an agile and responsive communication framework among controllers [41], [42].

2.2.8. Multiple controller scheduling

In scenarios where multiple controllers are involved in the orchestration of SDNs, the effective coordination of tasks and the distribution of workloads among these controllers become critical architectural considerations. Efficient load balancing, which involves the distribution of network tasks evenly among controllers, is essential to prevent bottlenecks and optimize overall system performance. Moreover, effective task allocation ensures that each controller is assigned responsibilities in a manner that leverages its specific capabilities, contributing to optimal resource utilization. This intricate balance in workload distribution and resource allocation is indispensable for achieving high-performance, fault-tolerant, and scalable SDN environments, where the seamless operation of distributed controllers is paramount to the network's overall reliability and efficiency [42].

3. RESEARCH CHALLENGES: IMPROVING QOS IN MULTIPLE CONTROLLER SDN NETWORKS

Although research in SDN has contributed to the improvement of QoS, there is still a need for further investigation in this area. The two major challenges that require more research efforts are efficient controller placement and network reliability. According to surveys, the following are the research challenges aimed at enhancing QoS in multiple controller SDN networks:

– QoS-aware controller placement: designing algorithms and frameworks for optimal placement of controllers in multiple controller SDN architectures to enhance QoS provisioning. This involves considering factors such as network topology, traffic patterns, and QoS requirements to ensure efficient control and management [43].
– Distributed QoS management: developing mechanisms for distributed QoS management across multiple controllers to maintain consistent QoS policies and ensure seamless communication and coordination. Addressing challenges related to synchronization, consistency, and policy enforcement across distributed controllers is crucial for effective QoS provisioning.
– Load balancing and task allocation: designing intelligent load balancing and task allocation techniques to distribute the workload among multiple controllers [44]. This involves considering factors such as controller capabilities, network dynamics, and QoS requirements to optimize resource utilization and prevent overload situations [45].
– QoS monitoring and measurement: developing efficient methods for QoS monitoring and measurement in multiple controller SDN environments. This includes identifying appropriate QoS metrics, designing scalable monitoring frameworks, and analyzing QoS data to detect anomalies and proactively address performance degradation issues.

– Dynamic QoS adaptation: investigating adaptive QoS strategies that can dynamically adjust QoS parameters based on network conditions, traffic demands, and application requirements. This involves exploring techniques such as traffic engineering, bandwidth allocation, and congestion control to optimize QoS performance in real-time.
– QoS-aware traffic engineering: researching QoS-aware traffic engineering approaches to efficiently route traffic across the network, considering QoS requirements and resource availability. This includes developing algorithms that can dynamically adjust traffic paths to meet QoS objectives and mitigate congestion or performance bottlenecks.
– Controller security: the security of an entire network can be compromised due to security vulnerabilities in an SDN controller [46]. To mitigate potential attacks such as spoofing, tampering, denial of service (DoS) [47]–[49], and privilege elevation, it is essential to implement measures within the controller itself. These measures include processing an application permission framework, containment, and monitoring of resource utilization. By implementing these safeguards, the risks associated with security vulnerabilities can be minimized [50].

4. CONCLUSION AND PERSPECTIVES

In conclusion, this paper has delved into the importance of QoS within the realm of controller management in SDN. QoS plays a crucial role in ensuring optimal network performance, reliability, and user satisfaction. The results of this investigation underscore the significance of effective controller management for improving QoS provisioning in SDN environments. Throughout the paper, various research challenges and considerations regarding QoS in controller management have been identified and discussed. The issues of controller placement, synchronization, load balancing, and dynamic adaptation have emerged as critical areas that demand further investigation. Looking ahead, several promising research perspectives emerge in the domain of QoS and controller management in SDN. Researchers can explore and develop advanced QoS algorithms that consider not only traditional QoS metrics but also factors such as network dynamics, application requirements, and user preferences. This can enable more dynamic and personalized QoS provisioning in SDN environments. Moreover, incorporating artificial intelligence (AI) and machine learning (ML) techniques into SDN can reveal the new opportunities for optimizing QoS. AI-based approaches can enhance decision-making processes, traffic engineering, and proactive QoS management in real-time, leading to improved network performance. Furthermore, investigating autonomous management techniques, where SDN controllers have self-configuring, self-optimizing, and self-healing capabilities, can revolutionize QoS provisioning. Autonomous decision-making can adapt to network changes, resolve issues promptly, and maintain desired QoS levels.

REFERENCES

[1] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss, *An Architecture for differentiated services*, RFC 2475, 1998.
[2] R. Braden, D. Clark, and S. Shenker, *Integrated services in the internet architecture: an overview*, RFC 2475, 1994.
[3] L. Subramanian, I. Stoica, H. Balakrishnan, and R. H. Katz, "OverQoS: offering internet QoS using overlays," *Computer Communication Review*, vol. 33, no. 1, pp. 11–16, Jan. 2003, doi: 10.1145/774763.774764.
[4] D. Kostić, A. Rodriguez, J. Albrecht, and A. Vahdat, "Bullet: high bandwidth data dissemination using an overlay mesh," in *Proceedings of the nineteenth ACM symposium on Operating systems principles*, Oct. 2003, pp. 282–297, doi: 10.1145/945445.945473.
[5] S. Floyd, M. Handley, J. Padhye, and J. Widmer, "Equation-based congestion control for unicast applications," *ACM SIGCOMM Computer Communication Review*, vol. 30, no. 4, pp. 43–56, Aug. 2000, doi: 10.1145/347057.347397.
[6] J. Yan, W. Muhlbauer, and B. Plattner, "Analytical framework for improving the quality of streaming over TCP," *IEEE Transactions on Multimedia*, vol. 14, no. 6, pp. 1579–1590, Dec. 2012, doi: 10.1109/TMM.2012.2187182.
[7] J. Yan, K. Katrinis, M. May, and B. Plattner, "Media-and TCP-friendly congestion control for scalable video streams," *IEEE Transactions on Multimedia*, vol. 8, no. 2, pp. 196–206, Apr. 2006, doi: 10.1109/TMM.2005.864265.
[8] E. Rosen, A. Viswanathan, and R. Callon, "Multiprotocol label switching architecture," IETF RFC3031, p. 61, 2001.
[9] T. D. Nadeau and K. Gray, *SDN: software defined networks*. USA: O'Reilly Media, Inc., 2013.
[10] A. Shirmarz and A. Ghaffari, "Taxonomy of controller placement problem (CPP) optimization in software defined network (SDN): a survey," *Journal of Ambient Intelligence and Humanized Computing*, vol. 12, no. 12, pp. 10473–10498, Jan. 2021, doi: 10.1007/s12652-020-02754-w.
[11] M. Karakus and A. Durresi, "Quality of service (QoS) in software defined networking (SDN): a survey," *Journal of Network and Computer Applications*, vol. 80, pp. 200–218, Feb. 2017, doi: 10.1016/j.jnca.2016.12.019.
[12] W. Wang, Y. Tian, X. Gong, Q. Qi, and Y. Hu, "Software defined autonomic QoS model for future Internet," *Journal of Systems and Software*, vol. 110, pp. 122–135, Dec. 2015, doi: 10.1016/j.jss.2015.08.016.
[13] D. Kreutz, F. M. V. Ramos, P. E. Veríssimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-defined networking: a comprehensive survey," *Proceedings of the IEEE*, vol. 103, no. 1, pp. 14–76, 2015, doi: 10.1109/JPROC.2014.2374752.
[14] F. X. A. Wibowo, M. A. Gregory, K. Ahmed, and K. M. Gomez, "Multi-domain software defined networking: research status and challenges," *Journal of Network and Computer Applications, vol. 87, pp. 32–45, Jun. 2017, doi: 10.1016/j.jnca.2017.03.004.*
[15] A. K. Singh and S. Srivastava, "A survey and classification of controller placement problem in SDN," *International Journal of Network Management, vol. 28, no. 3, Mar. 2018, doi: 10.1002/nem.2018.*

[16] N. Feamster, J. Rexford, and E. Zegura, “The road to SDN: an intellectual history of programmable networks,” *Computer Communication Review*, vol. 44, no. 2, pp. 87–98, Apr. 2014, doi: 10.1145/2602204.2602219.

[17] D. Kreutz, F. M. V. Ramos, and P. Verissimo, “Towards secure and dependable software-defined networks,” in *HotSDN 2013 - Proceedings of the 2013 ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking*, Aug. 2013, pp. 55–60, doi: 10.1145/2491185.2491199.

[18] T. Dargahi, A. Caponi, M. Ambrosin, G. Bianchi, and M. Conti, “A survey on the security of stateful SDN data planes,” *IEEE Communications Surveys and Tutorials*, vol. 19, no. 3, pp. 1701–1725, 2017, doi: 10.1109/COMST.2017.2689819.

[19] N. McKeown *et al.*, “OpenFlow: enabling innovation in campus networks,” *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 2, pp. 69–74, Mar. 2008, doi: 10.1145/1355734.1355746.

[20] ONF, “OpenFlow switch specification version 1.4.1,” *Open Network Foundation*, p. 227, 2015.

[21] D. Erickson, “The Beacon OpenFlow controller,” in *HotSDN 2013 - Proceedings of the 2013 ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking*, Aug. 2013, pp. 13–18, doi: 10.1145/2491185.2491189.

[22] S. H. Yeganeh and Y. Ganjali, “Beehive: towards a simple abstraction for scalable software-defined networking,” in *Proceedings of the 13th ACM Workshop on Hot Topics in Networks, HotNets 2014*, Oct. 2014, pp. 1–7, doi: 10.1145/2670518.2673864.

[23] “An open source SDN controller for cloud computing data centers,” *GitHub, 2023. https://github.com/China863SDN/DCFabric.*

[24] J. Bailey and S. Stuart, “Faucet: deploying SDN in the enterprise,” *Communications of the ACM*, vol. 14, no. 1, pp. 45–49, Oct. 2016, doi: 10.1145/3012426.3015763.

[25] “Project Floodlight,*” danetsoft.com, 2023. http://data.danetsoft.com/projectfloodlight.org.*

[26] R. Sherwood *et al.*, “FlowVisor: a network virtualization layer,” *Network*, p. 15, 2009.

[27] S. Hassas Yeganeh and Y. Ganjali, “Kandoo: a framework for efficient and scalable offloading of control applications,” in *HotSDN’12 - Proceedings of the 1st ACM International Workshop on Hot Topics in Software Defined Networks*, Aug. 2012, pp. 19–24, doi: 10.1145/2342441.2342446.

[28] A. Kazarez, “LOOM Controller,” *GitHub, 2023. https://github.com/FlowForwarding/loom.*

[29] Z. Cai, A. Cox, and E. T. S. Ng, “Maestro: a system for scalable OpenFlow control,” *Cs.Rice.Edu*. p. 10, 2011.

[30] N. Gude, T. Koponen, J. Pettit, and B. Pfaff, “NOX : towards an operating system y for networks introduction,” *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 3, pp. 1–17, Jul. 2008, doi: 10.1145/1384609.1384625.

[31] T. Koponen *et al.*, “Onix: a distributed control platform for large-scale production networks,” in *9th USENIX Symposium on Operating Systems Design and Implementation (OSDI 10)*, 2010, p. 14.

[32] P. Berde *et al.*, “ONOS: towards an open, distributed SDN OS,” in *HotSDN 2014 - Proceedings of the ACM SIGCOMM 2014 Workshop on Hot Topics in Software Defined Networking*, Aug. 2014, pp. 1–6, doi: 10.1145/2620728.2620744.

[33] G. Ferro “Considering the future of Juniper’s contrail and OpenContrail/Tungsten,” packetpushers, 2018. *packetpushers.net https://packetpushers.net/considering-future-junipers-contrail-opencontrail-tungsten/.*

[34] “OpenDaylight: a linux foundation collaborative project,” opendaylight.org, *2023. https://www.opendaylight.org/.*

[35] D. Saikia, S. Kong, N. Malik, and D. Kim, “OpenMUL SDN platform,” *openmul.org, 2023. http://www.openmul.org/openmul-controller.html.*

[36] “POX controller manual current documentation,” nexrepo.github.io, *2023. https://noxrepo.github.io/pox-doc/html/.*

[37] A. A. Alashhab, M. S. M. Zahid, M. A. Azim, M. Y. Daha, B. Isyaku, and S. Ali, “A survey of low rate DDoS detection techniques based on machine learning in software-defined networks,” *Symmetry*, vol. 14, no. 8, p. 1563, Jul. 2022, doi: 10.3390/sym14081563.

[38] B. J. van Asten, N. L. M. van Adrichem, and F. A. Kuipers, “Scalability and resilience of software-defined networking: an overview,” *arXiv preprint*, Aug. 2014.

[39] S. H. Alnabelsi, H. A. B. Salameh, and Z. M. Albataineh, “Dynamic resource allocation for opportunistic software-defined IoT networks: Stochastic optimization framework,” *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 10, no. 4, pp. 3854–3861, Aug. 2020, doi: 10.11591/ijece.v10i4.pp3854-3861.

[40] S. J. Rashid, A. M. Alkababji, and A. S. M. Khidhir, “Performance evaluation of software-defined networking controllers in wired and wireless networks,” *Telkomnika (Telecommunication Computing Electronics and Control)*, vol. 21, no. 1, pp. 49–59, Feb. 2023, doi: 10.12928/TELKOMNIKA.v21i1.23468.

[41] M. H. H. Khairi, S. H. S. Ariffin, N. M. A. Latiff, and K. M. Yusof, “Generation and collection of data for normal and conflicting flows in software defined network flow table,” *Indonesian Journal of Electrical Engineering and Computer Science (IJEECS)*, vol. 22, no. 1, pp. 307–314, Apr. 2021, doi: 10.11591/ijeecs.v22.i1.pp307-314.

[42] Y. Zhang, L. Cui, W. Wang, and Y. Zhang, “A survey on software defined networking with multiple controllers,” Journal of *Network and Computer Applications*, vol. 103, pp. 101–118, Feb. 2018, doi: 10.1016/j.jnca.2017.11.015.

[43] H. H. Saleh, I. A. Mishkal, and D. S. Ibrahim, “Controller placement problem in software defined networks,” *Indonesian Journal of Electrical Engineering and Computer Science (IJEECS)*, vol. 27, no. 3, pp. 1704–1711, Sep. 2022, doi: 10.11591/ijeecs.v27.i3.pp1704-1711.

[44] I. Choukri, M. Ouzzif, and K. Bouragba, “Fault tolerant and load balancing model for software defined networking controllers,” *Indonesian Journal of Electrical Engineering and Computer Science (IJEECS)*, vol. 31, no. 1, pp. 378–385, Jul. 2023, doi: 10.11591/ijeecs.v31.i1.pp378-385.

[45] C. Fancy and M. Pushpalatha, “Traffic-aware adaptive server load balancing for software defined networks,” *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 11, no. 3, pp. 2211–2218, Jun. 2021, doi: 10.11591/ijece.v11i3.pp2211-2218.

[46] S. Aouad, I. El Meghrouni, Y. Sabri, A. Hilmani, and A. Maizate, “Security of software defined networks: evolution and challenges,” *International Journal of Reconfigurable and Embedded Systems (IJRES)*, vol. 12, no. 3, pp. 384–391, Nov. 2023, doi: 10.11591/ijres.v12.i3.pp384-391.

[47] B. Mladenov and G. Iliev, “Optimal software-defined network topology for distributed denial of service attack mitigation,” *Bulletin of Electrical Engineering and Informatics (BEEI)*, vol. 9, no. 6, pp. 2588–2594, Dec. 2020, doi: 10.11591/eei.v9i6.2581.

[48] H. Kamel and M. Z. Abdullah, “Distributed denial of service attacks detection for software defined networks based on evolutionary decision tree model,” *Bulletin of Electrical Engineering and Informatics (BEEI)*, vol. 11, no. 4, pp. 2322–2330, Aug. 2022, doi: 10.11591/eei.v11i4.3835.

[49] M. I. Kareem and M. N. Jasim, “Entropy-based distributed denial of service attack detection in software-defined networking,” *Indonesian Journal of Electrical Engineering and Computer Science (IJEECS)*, vol. 27, no. 3, pp. 1542–1549, Sep. 2022, doi: 10.11591/ijeecs.v27.i3.pp1542-1549.

[50] S. Ahmad and A. H. Mir, “Scalability, consistency, reliability, and security in SDN controllers: a survey of diverse SDN controllers,” *Journal of Network and Systems Management*, vol. 29, no. 1, p. 59, Nov. 2021, doi: 10.1007/s10922-020-09575-4.

BIOGRAPHIES OF AUTHORS

Siham Aouad holds a Ph.D. in Computer Engineering from Mohammadia School of Engineers EMI in 2014. She obtained her network engineering degree from the National School of Applied Sciences ENSA Tangier in 2005. Currently, she works at the department of communication networks at the National School of Computer Science and Systems Analysis ENSIAS. Her research interests span across various areas including wireless communications, WSN, smart cities, SDN, AI, virtualization, cloud computing, and security. She can be contacted at email: siham.aouad@ensias.um5.ac.ma.

Issam El Meghrouni is currently a Ph.D. student in the RITM (Networks, IT, Telecommunications and Multimedia) Laboratory at Hassan II University. Recognition gesture, machine learning, and deep learning are among his research interests. He can be contacted at email: magrouni@gmail.com.

Yassine Sabri was born on October 28, 1984, in Rabat, Morocco. He pursued his Ph.D. in the field of WSN Technology at the Laboratory of Science and Technology. In 2013, he joined the Department of Science and Technology at ISGA Rabat, Morocco, as an Assistant Professor. Yassine’s research interests encompass a broad range of topics, including wireless sensor networks, evolutionary computation, internet of things (IoT), and mobile computing. He can be contacted at email: yassine.sabri@isga.ma.

Adil Hilmani after completing his diploma in Network and Telecommunication engineering from the University of Seville in Spain, he went on to obtain his doctorate in computer engineering from ENSEM in Casablanca-Morocco in 2021. Currently, he serves as a professor at OFPPT in Kénitra, Morocco. His research interests lie in the areas of mobile networks and computing, wireless sensor networks, and embedded systems software for IoT. He can be contacted at email: adilhilmani@gmail.com.

Abderrahim Maizate after completing his diploma in Network and Telecommunication engineering from the University of Seville in Spain, went on to obtain his doctorate in computer engineering from ENSEM in Casablanca-Morocco in 2021. Currently, he serves as a professor at OFPPT in Kénitra, Morocco. His research interests lie in the areas of mobile networks and computing, wireless sensor networks, and embedded systems software for IoT. He is a member of IEEE. He can be contacted at email: maizate@outlook.com.