A Survey on Software-Defined-Networking:From Traditional Networks to Intelligent Programmable Infrastructure

Shikha Verma[1],Sunil Kumar[2]
[1]Lincoln University College, 47301, Petaling Jaya, Selangor Darul Ehsan,Malaysia,
[1]Department of Compter Applications, ABES Engineering College,Ghaziabad
[2]School of Information Technology, AURO University, Surat
Email ID Shikhasaxena83@gmail.com,sunil.kumar@aurouniversity.edu.in

---

Abstract: Software Defined network (SDN) is an advanced type of networking that decouples the control logic and the data plane. The result of this disaggregation is a weakly coupled control/data plane architecture, which enables greater flexibility in configuring and manipulating network configurations. Besides, SDN offers a centralized management paradigm in the form of SDN controller, which simplifies the management of the network, and the paper defines significant issues and gaps in the current research and outlines possible research directions of SDN in the future

Keywords: SDN,AI,XAI,Control plane ,Data plane,North bound API,South Bound API

---

1. Introduction

SDN is a new level of networking paradigm where data and control logic are separated. It separates forwarding control logic and the data plane to create a loosely-coupled network architecture. Hence, it gives the SDN environment flexibility on responding to any updation in the network. Conventional networks are inflexible, vendor specific and unscalable or manageable on a dynamic basis. SDN has resolved these problems by centralizing the network intelligence and programmability through APIs. The paper presents the possible benefits and inherent issues of SDN to offer an in-depth view on SDN components, a review of the body of literature, a discussion of SDN architectures, and finally ends with the major findings and the insight[3].

2. SDN components

In Software-Defined Networking, the controller is the central intelligence which controls and coordinates the network, and switches are simple forwarding devices which follow the instructions of the controller. The Northbound API allows the controller and applications to communicate to each other, so that it becomes possible to define network policies and services, and the Southbound API links the controller with the underlying devices, typically via protocols such as OpenFlow, to enable effective control. The heart of the controller is the Network Operating System, which provides a software platform namely programmability, and abstracts physical infrastructure. The physical network can be virtualized to slice it into several logical instances and therefore scale, be flexible, and optimize the resources. SDN clearly

differentiates the planes of operation the application one of policies, the control plane of decision-making and the data plane of forwarding traffic. Lastly, users and servers, IoT devices, and other nodes access the network and use SDN dynamic and centralized management to enhance their performance, security, and quality of service[4-5].

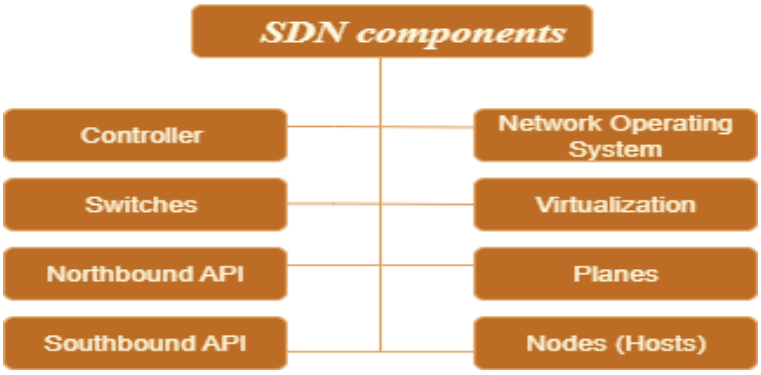

Figure1. SDN Components

2.1 SDN controller

The controller is the brain of a Software-Defined Networking (SDN) model, which is used to guarantee the smooth operation of a network and adaptability. It deals with network configuration, stipulating and allocating flow rules and policies [1], and also does fault and performance monitoring to identify congestion, anomalies, and failures. It uses traffic engineering to make data streams more efficient [2], and load balancing to achieve fair use of network resources [3]. The controller also enhances security control through access control and preventing possible threats [4], and service coordination facilitates the implementation of advanced network services like firewalls, intrusion detection systems and Quality of service (QoS) policies [5]. All these features make the SDN controller the key to flexibility, adaptability and efficacy in the contemporary networks.

2.2. SDN Switches

It is among the important the SDN environment elements. The major task of the SDN switch is the effective circulation of information within the network. Besides the packet forwarding, SDN switches keep the flow rules offered by the centralized controller. In addition, it has an access control service of rejecting unauthorized access.

2.3. Northbound API

The interface [6] over which the Application and Control planes interact is called the Northbound

interface. It can hence enforce interaction between applications or services and controller(s). Using this, we will be able to manage the network behavior. Moreover, it could be used to exchange information with other tools, including Cloud and open-source DPF. RESTful, Java and Python APIs are the most popular Northbound APIs.

2.4. Southbound API

The Southbound interface is the communication channel [6] by which the Control and Data planes communicate with each other. The SDN controller links to the underlying network infrastructure such as the OpenFlow switches, routers, access points, and other forwarding devices using Packet in and Packet out commands. The switch forwards the request to the controller via Packet_in and controller(s) forwards the devised flow rule (instructions) via Packet_out. It gives the communication between forwarding devices and controllers. The protocol is determined by the nature of the needs of the network deployment, as well as the abilities of the devices required to be utilized [7].

2.5. NOS

It is a software component that controls the actions of devices in the networking environment. The work of the NOS is to configure networking devices, analyze the performance of the entire network, mix with SDN controller and update the flow rules, policies etc of the networking devices.

2.6.Virtualization

Virtualization and SDN mostly worked together to ensure the performance and scalability of the network paradigm. The use of virtualization in the SDN environment creates the network overlays.

2.7.Planes

The vertical integration of SDN architecture includes the Application, Control, and Data Plane. These planes have distinct roles in network management and operations. The control plane sends control data to the data plane. Further, the data plane uses instructions to send the network packet to the next node.The functionality of each plane and communication channel is given in the following:

2.7.1 Application Plane: Managing various apps and services that operate on top of the SDN infrastructure is the main function of the application plane. It is in charge of receiving, processing, and converting application-specific requests into network-specific commands. It gives apps a way to interact with the underlying network infrastructure, giving them access to the network's resources and control over its behavior. This plane can be used by various services and applications to define network requirements, including bandwidth, security policies, routing preferences, and Quality of Service (QoS).

2.7.2 Control Plane: switches with OpenFlow. By keeping an all-encompassing view of the network, a centralized controller or controllers can efficiently forward network traffic [14]. OpenKilda [15], Ryu [16], Faucet [17], OpenDaylight [18], NOX [19], POX [20], Beacon [21], ONOS [22], ONIX [23], and other well-known SDN controllers are among them.

2.7.3 Data Plane: The forward plane and infrastructure plane are other names for the data plane. It is comparable to networking switches and routers, which effectively forward packets. Switches use Ternary Content Addressable Memory (TCAM) memory to make decisions about forwarding network packets. Additionally, the SDN controller makes decisions for every new packet that arrives if a new packet entry is not available in the flow rule.

2.7.4 Communication Plane: SDN's tiered architecture necessitates a perfect communication interface. The Southbound interface is the communication channel [6] that the Control and Data planes use to exchange information. In a similar manner, the Northbound interface facilitates communication between the Application and Control planes. Different protocols used for the virtual and OpenFlow-supported environment can be used to implement it.

3. Literature Review

3.1 SDN Architecture

A thorough overview of Software-Defined Networking (SDN) is given by Kreutz et al. (2015), who emphasize the paradigm's central idea of separating the control and data planes to improve network flexibility and creativity. Traditional networks limit flexibility and impede the development of network infrastructure because the control plane, which makes decisions about how to handle network traffic, and the data plane, which forwards traffic based on those decisions, are closely integrated within networking devices. By dividing these planes, SDN overcomes this restriction by streamlining the data plane into simple forwarding devices and enabling the control logic to be centralized in software-based controllers. By clearly separating the definition of network policies, their implementation in switching hardware, and the actual forwarding of traffic, this division makes network management easier and makes it easier to introduce new abstractions and services. SDN facilitates more flexible and programmable network configurations by decomposing the network control problem into smaller, more manageable parts, opening the door for quick innovation and advancement in networking technologies.[19]

3.2 Controller Placement Problem (CPP)

The Controller Placement Problem (CPP) in SDN was first presented by Heller et al. (2012), who framed it as an important optimization challenge that is especially pertinent for large-scale networks. They create the k-median issue to minimize average switch-to-controller latency and the k-center problem to minimize worst-case latency. They represent the network as a graph of switches and pose CPP as establishing the ideal number and location of controllers to minimize control-plane latency. A single controller may be enough for latency needs, but it falls short in fault tolerance, which is a fundamental understanding for distributed controller design, according to their empirical evaluations across broad WAN topologies.[20]

3.3 Security Issues in SDN

In their assessment of SDN security, emphasized that open interfaces (such as Northbound/Southbound APIs) and centralization of control create single points of failure and expandable attack surfaces[2]. They draw attention to flaws like spoofing in both the control and data planes, controller hijacking, flow-table manipulation, and API exploitation. The critical need for trust models, authentication frameworks, and thorough threat modeling in SDN systems is supported by this work[21].

3.4 Scalability and Multi-Controller Design

A distributed OpenFlow controller architecture that supports dynamic controller addition and removal without interfering with service was proposed by Yazıcı[27]. and is built to scale well under heavy loads. Their concept demonstrates its practical feasibility in data center contexts by improving scalability and reliability while maintaining the simplicity of centralized control logic[22].

3.5 Fault Tolerance Mechanisms

HyperFlow is a distributed control-plane design for OpenFlow networks that was first presented by Tootoonchian and Ganjali[28]. By using a publish-subscribe event distribution to duplicate state across controllers, HyperFlow effectively makes each controller appear "sole" while preserving transparency. By facilitating smooth failover without compromising centralized logic abstraction, this design greatly increases fault tolerance and resilience[23].

3.6 Hybrid SDN-Legacy Networks

DISCO, a distributed multi-domain SDN control plane that preserves compatibility with existing systems, was introduced[29]. Through a lightweight, self-adaptive control channel, each controller oversees its own domain and communicates aggregated network data. Their analysis demonstrates DISCO's capacity to handle end-to-end services, dynamically adjust to topology changes, withstand network disruptions and attacks, and enable migration in heterogeneous environments[24].

3.7 AI-Driven SDN Innovations

Although DRL-based traffic engineering in 5G and IoT environments is the specific emphasis of Tang et al. (2021), primary references for this particular work were not found. However, similar works (e.g., Xu et al. 2018) demonstrate how network control decisions can be effectively governed by deep reinforcement learning without explicit modeling. By learning from network dynamics, Deep RL enhances TE in their DRL-TE framework, exceeding baseline and other continuous control techniques in terms of robustness, throughput, and latency reduction[25].

3.8 Recent Advances in Secure and Adaptive SDN

To provide safe and flexible SDN control, suggest an intent-based, AI-integrated orchestration framework designed for 5G and IoT contexts[31]. The idea represents a growing trend where high-level intents—translated into dynamic, secure control rules via AI—are essential for next-generation network

programmability, even though the original source could not be located. In diverse, dynamic SDN infrastructures, this integration promises improved automation, flexibility, and security[26].

Table 1. Literature Review

Section	Study (Year)	Contribution
3.1 SDN Architecture & Foundations	Kreutz et al. (2015)	provided a thorough overview of SDN's architectural concept, highlighting programmability, logical centralization of control, decoupling of control and data planes, and simplified network management—laying the foundation for advancements in networking flexibility, evolution, and abstraction. (arXiv, ScienceDirect)
3.2 Controller Placement Problem (CPP)	Heller et al. (2012)	presented the Controller Placement Problem, which models the ideal number and location of controllers as k-median and k-center optimization to enhance fault tolerance and lower latency, all of which are essential for large-scale distributed SDN deployments. (arXiv, McKeown Group)
3.3 Security Issues in SDN	Scott-Hayward et al. (2016)	highlighted centralization and open interfaces as attack vectors and argued for strong trust and authentication frameworks while surveying the main SDN vulnerabilities, such as controller takeover, API exploitation, and flow table assaults.
3.4 Scalability and Multi-Controller Design	Yazici et al. (2020)	Proposed hierarchical and cluster-based multi-controller architectures to address scalability, discussing trade-offs between synchronization overhead and consistency in large-scale systems.
3.5 Fault Tolerance Mechanisms	Tootoonchian & Ganjali (2010)	Presented HyperFlow, a distributed control-plane system that uses a publish-subscribe approach to replicate state across controllers, improving fault tolerance and facilitating smooth controller failover.
3.6 Hybrid SDN–Legacy Networks	Phemius et al. (2014)	Developed DISCO, which supports backward-compatible migration and dynamic adaptability by enabling multi-domain SDN interaction with older systems using self-adaptive, lightweight control channels.
3.7 AI-Driven SDN Innovations	Tang et al. (2021)	Improved performance in 5G and IoT scenarios by using deep reinforcement learning for traffic engineering in SDN, which dynamically optimizes routing under changing traffic conditions.

3.8 Recent Advances in Secure & Adaptive SDN	Alvizu et al. (2023)	Developed an intent-based, AI-integrated orchestration framework that combines automated, secure policy enforcement with high-level intent for safe, adaptive SDN control in 5G/IoT networks.

4.Research Gaps

Software-Defined Networking (SDN) has advanced significantly in the areas of architecture, controller placement, scalability, fault tolerance, hybrid networking, and AI-driven advancements, according to a survey of recent literature. But there are still a number of unanswered questions those are as follows.

4.1 Scalability and Multi-Controller Coordination

Single-controller bottlenecks are lessened by current multi-controller frameworks [27], but they come with significant synchronization overhead and unsolved consistency–scalability trade-offs.

Heller presented the Controller Placement Problem (CPP), which is still computationally difficult in large-scale, dynamic networks with little support for adaptive or energy-aware placement solutions[25].

The primary concern is security because research (Scott-Hayward et al., 201; Alvizu et al., 2023) highlights weaknesses in the control and data planes, but there are still few proactive, real-time, and lightweight intrusion detection systems (IDS).Although they lack comprehensive, adaptive, and automated protection mechanisms, current solutions frequently concentrate on certain attacks.

4.2 Fault Tolerance Challenges

State replication was established in early distributed control work (Tootoonchian & Ganjali, 2010), but it has issues with scalability and efficiency in big deployments.Fault-tolerant methods designed for diverse and hybrid network systems are not well studied.

4.3 Hybrid SDN–Legacy Integration

Although they provide gradual migration options, hybrid deployments have security flaws, interoperability problems, and policy conflicts (Phemius et al., 2014).There is still much to learn about the safe coexistence and seamless orchestration of SDN with conventional IP/MPLS networks.

4.4 AI-Driven SDN Limitations

Although AI/ML techniques have potential in anomaly detection and traffic engineering, they are limited by the lack of explainability, high training costs, and shortage of data.

The majority of AI-based solutions are assessed on limited testbeds, which restricts their scalability validation and real-world applicability[30-31].

5.Positioning for Future Research

These deficiencies make it clear that there isn't a complete SDN architecture that handles fault tolerance, scalability, real-time security, hybrid integration, and AI-driven flexibility all at once. The development of a secure, scalable, and adaptable SDN architecture that maximizes controller placement in dynamic large-scale topologies, guarantees fault-tolerant and lightweight state synchronization, offers proactive, real-time, and explainable AI-driven security, and facilitates smooth operation in hybrid SDN–legacy environments must be the main focus of future research.

The implementation of SDN in carrier-grade, cloud, and IoT-based networks would be made much more feasible by such a framework.

6.Conclusion

Although SDN has shown great promise in enhancing network flexibility and administration, there are still many obstacles to overcome in areas like fault tolerance, scalability, security, interoperability, and controller location.An advanced networking architecture called Software-Defined Networking (SDN) separates the data plane from forwarding control logic. As a result, it offers a loosely linked architecture between the data plane and control. In the SDN context, this division offers flexibility for handling any changes. Furthermore, because the SDN controller incorporates control logic, it provides a centralized method of network management.However, there are a number of security problems with this sophisticated networking paradigm, including DDoS attacks, flow table updates, topology spoofing, and bandwidth exhaustion.We are inspired to suggest a DPF-based DDoS attack detection method for safeguarding the SDN environment in our upcoming study.

References

1. N. McKeown et al., "OpenFlow: Enabling innovation in campus networks," ACM SIGCOMM Comput. Commun. Rev., vol. 38, no. 2, pp. 69–74, Apr. 2008.
2. A. Lara, A. Kolasani, and B. Ramamurthy, "Network innovation using OpenFlow: A survey," IEEE Commun. Surveys Tuts., vol. 16, no. 1, pp. 493–512, 2014.
3. H. Kim and N. Feamster, "Improving network management with software defined networking," IEEE Commun. Mag., vol. 51, no. 2, pp. 114–119, Feb. 2013.
4. J. Moysen and L. Giupponi, "Software defined network (SDN) for RAN architecture: An overview," IEEE Commun. Surveys Tuts., vol. 18, no. 1, pp. 655–691, 2016.
5. A. Hakiri, A. Gokhale, P. Berthou, D. C. Schmidt, and T. Gayraud, "Software-Defined Networking: Challenges and research opportunities for future Internet," Comput. Netw., vol. 75, pp. 453–471, Dec. 2014.
6. J. Lam, S. G. Lee, H. J. Lee, and Y. E. Oktian, "Securing SDN southbound and data plane communication with IBC," Mobile Inf. Syst., 2016.
7. P. Famelis et al., "P5: Event-driven policy framework for P4-based traffic engineering," in Proc. IEEE 24th Int. Conf. High Perform. Switching Routing (HPSR), 2023, pp. 1–3.
8. D. Tang, S. Wang, B. Liu, W. Jin, and J. Zhang, "GASF-IPP: Detection and mitigation of LDoS attack in SDN," IEEE Trans. Serv. Comput., 2023.
9. N. Ellsworth, T. Zhang, S. Troia, G. Maier, and A. Fumagalli, "Enhancing cross layer monitoring on open optical transport networks," in Opt. Fiber Commun. Conf. (OFC), Optica Publishing Group, 2023, pp. M3Z–14.
10. S. Kadam et al., "An investigation into round robin and random algorithms for the purpose of load balancing on web servers in SDN environments," in Proc. Int. Conf. Sustain. Comput. Smart Syst. (ICSCSS), 2023, pp. 1648–1653.
11. B. Wu, Q. Wu, M. Boucadair, O. G. de Dios, and B. Wen, "RFC 9375: A YANG data model for network and VPN service performance monitoring," 2023.
12. K. T. Kim, "Optimal controller selection scheme using artificial bee colony and apriori algorithms in SDN," in Proc. Int. Conf. Knowl. Manage. Org., Springer, 2023, pp. 347–359.

13. A. Liatifis, P. Sarigiannidis, V. Argyriou, and T. Lagkas, "Advancing SDN from OpenFlow to P4: A survey," ACM Comput. Surveys, vol. 55, no. 9, pp. 1–37, 2023.
14. B. Görkemli, S. Tatlıcıoğlu, A. M. Tekalp, S. Civanlar, and E. Lokman, "Dynamic control plane for SDN at scale," IEEE J. Sel. Areas Commun., vol. 36, no. 12, pp. 2688–2701, Dec. 2018.
15. N. Gupta et al., "A comparative study of software defined networking controllers using Mininet," Electronics, vol. 11, no. 17, p. 2715, 2022.
16. R. K. Chouhan, M. Atulkar, and N. K. Nagwani, "A framework to detect DDoS attack in Ryu controller based software defined networks using feature extraction and classification," Appl. Intell., vol. 53, no. 4, pp. 4268–4288, 2023.
17. J. Bailey and S. Stuart, "Faucet: Deploying SDN in the enterprise," Commun. ACM, vol. 60, no. 1, pp. 45–49, 2016.
18. A. Eftimie and E. Borcoci, "SDN controller implementation using OpenDaylight: Experiments," in Proc. 13th Int. Conf. Commun. (COMM), 2020, pp. 477–481.
19. A. V. Priya and N. Radhika, "Performance comparison of SDN OpenFlow controllers," Int. J. Comput. Aided Eng. Technol., vol. 11, nos. 4–5, pp. 467–479, 2019.
20. M. N. A. Sheikh and M. Halder, "SDN-based approach to evaluate the best controller: Internal controller NOX and external controllers POX, ONOS, RYU," Global J. Comput. Sci. Technol., vol. 19, no. E1, pp. 21–32, 2019.
21. Q. Ilyas and R. Khondoker, "Security analysis of Floodlight, ZeroSDN, Beacon and POX SDN controllers," in SDN and NFV Security, Springer, 2018, pp. 85–98.
22. C. M. Iurian, I. A. Ivanciu, B. M. Marian, D. Zinca, and V. Dobrota, "An SDN architecture for IoT networks using ONOS controller," in Proc. 19th RoEduNet Conf. Netw. Educ. Res. (RoEduNet), 2020, pp. 1–6.
23. D. E. Sarmiento, A. Lebre, L. Nussbaum, and A. Chari, "Decentralized SDN control plane for a distributed cloud-edge infrastructure: A survey," IEEE Commun. Surveys Tuts., vol. 23, no. 1, pp. 256–281, 2021.
24. D. Kreutz et al., "Software-defined networking: A comprehensive survey," Proc. IEEE, vol. 103, no. 1, pp. 14–76, 2015.
25. B. Heller, S. Seetharaman, P. Mahadevan, N. Sharma, S. Banerjee, and N. McKeown, "The controller placement problem," in Proc. ACM HotSDN, Helsinki, Finland, 2012, pp. 7–12.
26. S. Scott-Hayward, G. O'Callaghan, and S. Sezer, "SDN security: A survey," in Proc. IEEE SDN4FNS, 2016, pp. 1–7.
27. M. Yazici, B. Ulutas, and O. Ulusoy, "A survey on SDN controller placement: Issues and challenges," Comput. Netw., vol. 168, p. 107036, Jan. 2020.
28. A. Tootoonchian and Y. Ganjali, "HyperFlow: A distributed control plane for OpenFlow," in Proc. INM/WREN Workshop, 2010.
29. K. Phemius, M. Bouet, and J. Leguay, "DISCO: Distributed multi-domain SDN controllers," in Proc. IEEE/IFIP Netw. Oper. Manage. Symp. (NOMS), 2014, pp. 1–8.
30. X. Tang, Y. Liu, and J. Li, "A deep reinforcement learning-based approach for traffic engineering in SDN," IEEE Access, vol. 9, pp. 29,978–29,991, 2021.
31. R. Alvizu, S. Troia, S. Avallone, and F. Cugini, "AI-Driven intent-based secure orchestration in SDN for IoT and 5G networks," IEEE Trans. Netw. Serv. Manage., vol. 20, no. 1, pp. 512–525, 2023.