International Multidisciplinary Research Journal Reviews (IMRJR)

**International Multidisciplinary Research Journal Reviews (IMRJR)**
A Peer-reviewed journal
Volume 1, Issue 3, November 2024
DOI 10.17148/IMRJR.2024.010304

# SDN as a Key Enabler of a Software-Defined Environment

**Dr. Santosh Kumar Singh[1], Dr. V.R. Vadi[2], Dr. Asjad Usmani[3], Dr. P. K. Nayak[4]**

Associate Professor, Department of CS & IT, GGSIPU Affiliated College, New Delhi, India[1]

Professor, Director, DBIT, GGSIPU, New Delhi, India[2]

Associate Professor (HoD), Department of Management Studies, DBIT, GGSIPU, New Delhi, India[3]

Associate Professor (HoD), Department of Commerce, DBIT, GGSIPU, New Delhi, India[4]

**Abstract**: A digital society, where (nearly) everything is connected and accessible from anywhere, has been made possible by the Internet. Nevertheless, traditional IP networks are complicated and challenging to administer despite their broad use. It is challenging to set up the network with preset policies and to adjust it in response to changes, loads, and faults. The fact that the control and data planes of modern networks are bundled together adds to the difficulty of the situation. By dismantling vertical integration, separating the network's control logic from the underlying routers and switches, encouraging (logical) centralization of network control, and introducing the ability to program the network, **Software-Defined Networking (SDN)** is an emerging paradigm that promises to change this state of affairs. The key to the desired flexibility is the separation of concerns created between the definition of network policies, how switching hardware implements those policies, and how traffic is forwarded. SDN facilitates the creation and introduction of new abstractions in networking by breaking the problem of network control into manageable chunks, thereby streamlining network management and promoting network evolution. We provide an extensive survey on SDN in this chapter. Our comprehensive examination covers the network virtualization layers, hardware infrastructure, southbound and northbound APIs, network operating systems (SDN controllers), network programming languages, and network applications. We also examine cross-layer issues like troubleshooting and debugging. To predict how this new paradigm will develop in the future, we go over the primary SDN research topics and obstacles. We specifically cover new opportunities for carrier transport networks and cloud providers, as well as the design of switches and control platforms, with an emphasis on features like resiliency, scalability, performance, security, and dependability. Finally, we examine SDN's role as a fundamental facilitator of a software-defined environment.

**Keywords:** OpenFlow, network virtualization, network operating systems, programmable networks, network hypervisor, programming languages.

## I. INTRODUCTION

SDN eases the complexity of managing highly distributed physical network infrastructures with programmability and centralized control. Software-defined networking (SDN) is an innovative approach to network management that separates the control plane (which makes decisions about where traffic is sent) from the data plane (which moves packets based on the decisions of the control plane). This separation provides more flexibility, easier management, and greater scalability for network operations. Here are the key components and concepts of SDN:

1. Control Plane: This is the centralized part of the network that makes decisions about the routing of data packets. In SDN, the control plane is decoupled from the data plane and is typically implemented in a centralized controller. 2. Data Plane: Also known as the forwarding plane, this part of the network is responsible for forwarding packets based on the control plane's decisions. The data plane resides on network devices like switches and routers. 3. SDN Controller: The controller acts as the brain of the SDN network. It communicates with the network devices (data plane) using standard protocols like OpenFlow. The controller provides a centralized view and control of the entire network, enabling administrators to program the network via software. 4. Northbound APIs: These are interfaces between the SDN controller and the applications or services running over the network. They allow the applications to request services and define network policies without needing to understand the underlying network details. 5. Southbound APIs: These interfaces are between the SDN controller and the network devices. They facilitate communication and command execution, enabling the controller to instruct the devices on how to handle traffic. 6. Network Virtualization: SDN often involves the virtualization of network functions, allowing multiple virtual networks to run on a shared physical infrastructure. This virtualization improves resource utilization and provides greater flexibility in managing network resources.

**International Multidisciplinary Research Journal Reviews (IMRJR)**
A Peer-reviewed journal
Volume 1, Issue 3, November 2024
DOI 10.17148/IMRJR.2024.010304

Benefits of SDN

- Flexibility: Network administrators can dynamically adjust network configurations to meet changing demands without physical interventions.
- Scalability: Centralized control allows for easier scaling of network resources to handle increased traffic loads.
- Cost Efficiency: By decoupling the control and data planes, SDN can reduce the reliance on expensive proprietary hardware.
- Enhanced Security: SDN provides granular control over traffic flows, enabling better implementation of security policies.
- Improved Network Management: Centralized management simplifies the configuration and maintenance of the network, leading to reduced operational costs and complexities.

Applications of SDN

- Data Centers: Optimizing traffic flow and resource allocation within and between data centers.
- Cloud Computing: Enhancing the management of virtualized resources and services in cloud environments.
- Wide Area Networks (WANs): Improving the performance and management of large-scale WANs.
- Network Function Virtualization (NFV): Virtualizing network services such as firewalls, load balancers, and intrusion detection systems.

In essence, SDN transforms traditional static networks into flexible, programmable, and adaptable environments, meeting the dynamic needs of modern applications and services. Software-defined networking (SDN) plays a significant role in enhancing parallel and distributed computing by providing greater control, flexibility, and efficiency in network management. Here are some key ways SDN helps in this context:

1. Dynamic Resource Allocation: SDN allows for real-time reconfiguration of network resources, optimizing bandwidth and minimizing latency. This dynamic allocation is crucial for parallel and distributed computing tasks that require high data throughput and low-latency communication.

2. Network Abstraction: SDN abstracts the underlying network infrastructure, presenting a unified view of applications. This abstraction simplifies the development and deployment of parallel and distributed applications by hiding the complexity of the physical network.

3. Centralized Control: SDN's centralized control plane enables efficient management of network resources, ensuring that data flows are optimally routed. This central control is particularly beneficial for distributed computing environments where data must be transferred between multiple nodes efficiently.

4. Scalability: SDN facilitates scalable network architectures, making it easier to manage large-scale distributed computing systems. The ability to quickly adjust network configurations helps in accommodating the growing number of nodes and varying workload demands.

5. Fault Tolerance and Reliability: By monitoring network health and dynamically rerouting traffic in case of failures, SDN enhances the reliability and fault tolerance of distributed computing systems. This capability ensures continuous operation even in the face of network disruptions.

6. Load Balancing: SDN enables intelligent load balancing across network paths and computing resources. This ensures that workloads are evenly distributed, preventing bottlenecks and improving overall system performance.

7. Quality of Service (QoS): SDN can enforce QoS policies to prioritize critical data flows in parallel and distributed computing environments. This ensures that high-priority tasks receive the necessary bandwidth and low-latency connections they require.

8. Enhanced Security: SDN provides granular control over network traffic, allowing for the implementation of fine-grained security policies. This is essential in distributed computing to protect sensitive data and prevent unauthorized access.

9. Integration with Cloud and Edge Computing: SDN seamlessly integrates with cloud and edge computing environments, which are often used in distributed computing. This integration helps in optimizing data flows between centralized data centers and distributed edge nodes.

International Multidisciplinary Research Journal Reviews (IMRJR)
A Peer-reviewed journal
Volume 1, Issue 3, November 2024
DOI 10.17148/IMRJR.2024.010304

By leveraging these capabilities, SDN significantly improves the efficiency, performance, and manageability of parallel and distributed computing systems.

One new networking paradigm that offers hope for addressing the shortcomings of existing network infrastructures is software-defined networking or SDN. By dividing the network's control logic (the control plane) from the underlying routers and switches that forward traffic (the data plane), it first breaks the vertical integration. Second, policy enforcement, network (re)configuration, and network evolution are made simpler by the separation of the control and data planes, which reduces network switches to simple forwarding devices and implements control logic in a logically centralized controller (or network operating system) [1].

A clear programming interface between the switches and the SDN controller can be used to achieve the separation of the control plane and the data plane. Through this clearly defined application programming interface (API), the controller has direct control over the state of the data-plane elements. OpenFlow is the most well-known illustration of such an API [2], [3]. One or more packet-handling rule tables, or flow tables, are present in an OpenFlow switch. Every rule matches a portion of the flow and applies specific operations (forwarding, modifying, dropping, etc.) to the flow. An OpenFlow switch can act as a router, switch, firewall, or take on other functions based on the rules that have been installed by a controller application.

The division of responsibilities between the formulation of network policies, their implementation in switching hardware, and traffic forwarding is a significant outcome of software-defined networking principles. The desired flexibility is largely due to this separation, which divides the network control problem into manageable chunks and facilitates the creation and introduction of new networking abstractions. This also simplifies network management and encourages network evolution and innovation.

Despite beginning as academic experiments, SDN and OpenFlow have gained a great deal of traction in the industry in the last few years [2]. Nowadays, the majority of commercial switch manufacturers integrate OpenFlow API support into their devices. With the primary objective of promoting and advancing SDN adoption through open standards development, the Open Networking Foundation (ONF) was funded by Google, Facebook, Yahoo, Microsoft, Verizon, and Deutsche Telekom due to the strong momentum surrounding SDN [3]. Since the early issues regarding SDN scalability were resolved [4] specifically, the misconception that logical centralization meant a physically centralized controller, a topic we will revisit later SDN concepts have developed and progressed from an academic endeavor to a profitable venture. Google, for instance, has connected its data centers worldwide by implementing a software-defined network. With this production network in place for three years, the business has been able to cut expenses and increase operational efficiency [5]. Another example is the network virtualization platform called NSX from VMware [6].

A commercial product called NSX uses SDN principles to provide a fully functional network in software that is provisioned independently of the underlying networking devices. Finally, to illustrate the significance of SDN from an industrial standpoint, the biggest IT companies in the world, ranging from carriers and equipment manufacturers to cloud providers and financial services firms, have recently joined SDN consortia like the OpenDaylight initiative and the ONF [7].

Certain architectural aspects of SDN have been surveyed in a few recent papers [8], [9], and [10]. OpenFlow overview and a brief review of related literature can be found in [8] and [9]. The three-layer stack that these OpenFlow-oriented surveys show is made up of controllers, the controller/switch interface, and high-level network services. The authors of [10] take things a step further and suggest a taxonomy for SDN.

To the best of our knowledge, the most thorough literature review on SDN to date is presented in this paper. We structure this survey as follows: in the first two sections, we provide background information, outline the rationale behind SDN, and describe the key ideas of this novel paradigm as well as its distinctions from conventional networking. This paradigm shift in networking is the result of combining tried-and-true ideas with two recent developments in networking: the availability of merchant switch silicon and the intense interest in workable forms of network virtualization. Many standardization efforts around SDN are ongoing and covered in Section III due to the high industry interest and potential to change the status quo of networking from multiple perspectives. The main body of this survey, Section IV, provides an in-depth analysis of the fundamental components of an SDN infrastructure through a layered, bottom-up methodology and research issue of SDN. This paper concludes with a discussion of current research efforts, challenges, future work, and opportunities in Section V.

## II. COMPUTER NETWORKING AND MOTIVATION FOR SDN

The data, control, and management planes are the three functional levels that make up computer networks. The networking devices, which are in charge of (effectively) forwarding data, are represented by the data plane. The protocols that are used to fill the data plane elements' forwarding tables are represented by the control plane. The software services, like SNMP-based tools [11], that are used to remotely monitor and configure the control functionality are part of the management plane. The management plane defines the network policy, the control plane applies it, and the data plane carries it out by forwarding data by the policy.

International Multidisciplinary Research Journal Reviews (IMRJR)
A Peer-reviewed journal
Volume 1, Issue 3, November 2024
DOI 10.17148/IMRJR.2024.010304

Conventional IP networks feature a highly decentralized architecture with closely coupled control and data planes that are embedded in the same networking devices. In the early days of the Internet, this was thought to be significant because it appeared to be the most effective method of ensuring network resilience, a critical design objective. The performance of the network has increased quickly with this approach.

Nonetheless, as has been frequently documented in the networking literature, the result is a highly intricate and largely static architecture (e.g., [12], [13], [14], [15]). It is also the main cause of traditional networks' rigidity and complexity in terms of management and control. Innovation is challenging in a vertically integrated industry, which is primarily caused by these two traits.

In today's networks, network misconfigurations and related errors are incredibly common. For example, BGP router configuration errors have been reported to exceed 1000 [16]. Very undesirable network behavior (such as packet losses, forwarding loops, the creation of unwanted paths, or service contract violations) can arise from a single misconfigured device. Undoubtedly, a solitary incorrectly configured router can jeopardize the uninterrupted functioning of the entire Internet for several hours [17], [18].

A limited number of vendors provide proprietary solutions of specialized hardware, operating systems, and control programs (network applications) to support network management. Network operators must purchase and maintain various management solutions along with the specialized teams that go along with them. Building and maintaining a networking infrastructure has high capital and operating costs, as well as lengthy return on investment cycles. These factors impede innovation and the introduction of new features and services, such as load balancing, traffic engineering, energy efficiency, and access control.

In modern networks, a plethora of specialized parts and middleboxes, including intrusion detection systems, firewalls, and deep packet inspection engines, are widely used to mitigate the lack of in-path functionalities within the network. According to a recent survey conducted on a few enterprise networks, the number of middleboxes in use today is comparable to the number of routers [19]. Middleboxes have improved in-path functionalities, but overall, they have made network design and operation more complex.

## III. WHAT IS SOFTWARE-DEFINED NETWORKING?

The concepts and efforts surrounding OpenFlow at Stanford University gave rise to the term SDN (Software-Defined Networking) in the beginning [20]. When SDN was first introduced, it described a network architecture in which a remote-control plane that is disconnected from the data plane is in charge of managing the forwarding state in the latter. The networking industry has frequently moved away from this initial definition of SDN, labeling anything involving software as SDN. For this reason, we try to give a much clearer definition of software-defined networking in this section. A network architecture consisting of four pillars is what we define as an SDN.

(1) There is a separation of the control and data planes. Network devices that will become basic (packet) forwarding elements lose their control functionality. (2) Rather than destination-based, forwarding decisions are made based on flow. A set of actions (instructions) and a set of packet field values serving as a match (filter) criterion define a flow in general terms. A flow is a series of packets that travel from a source to a destination in the context of SDN/OpenFlow. At the forwarding devices, every packet in a flow is given the same service policies [21], [22]. The behavior of various network device types, such as routers, switches, firewalls, and middleboxes, can be unified thanks to the flow abstraction [23]. Unprecedented flexibility is made possible by flow programming; it is only restricted by the capabilities of the flow tables that are in place [2]. (3) The Network Operating System (NOS) or so-called SDN controller is an external entity that receives control logic. Operating on commodity server technology, the NOS is a software platform that offers the necessary abstractions and resources to make it easier to program forwarding devices using an abstract, logically centralized network view. As such, its function is comparable to that of a conventional operating system. (4) Software programs that communicate with the underlying data plane devices and operate on top of the NOS can be used to program the network. This is regarded as SDN's primary value proposition and a fundamental feature.

Keep in mind that there are various other advantages associated with the logical centralization of the control logic in particular. First, compared to low-level device-specific configurations, changing network policies via high-level programming languages and software components is easier and less prone to error. Secondly, a control program can respond to erroneous modifications in the network state automatically, preserving the high-level policies. Third, the creation of increasingly complex networking features, services, and applications is made easier by the centralization of the control logic in a controller that has global knowledge of the network state.

An SDN can be defined by three basic abstractions, which follow the SDN concept presented in [5]: (i) forwarding, (ii) distribution, and (iii) specification. Since abstractions are already widely present in many computer architectures and systems, they are vital research instruments in computer science and information technology [24]. The forwarding abstraction should ideally conceal the specifics of the underlying hardware and permit any forwarding behavior that the network application (the control program) requests. One implementation of this abstraction, which is comparable to an operating system's "device driver," is OpenFlow.

International Multidisciplinary Research Journal Reviews (IMRJR)
A Peer-reviewed journal
Volume 1, Issue 3, November 2024
DOI 10.17148/IMRJR.2024.010304

SDN applications should be protected from the whims of the distributed state by the distribution abstraction, which logically centralizes the distributed control problem. A common distribution layer, which in SDN is housed in the NOS, is necessary for its realization. This layer serves two primary purposes. Initially, it is in charge of setting up the forwarding devices with the control commands. To provide network applications with a comprehensive network view, it also gathers status data regarding network devices and links at the forwarding layer.

The final abstraction is specification, which ought to enable a network application to specify the intended behavior of the network without having to take on the task of carrying it out. Network programming languages and virtualization solutions can help achieve this. These methods translate the abstract configurations expressed by the applications based on a reduced, abstract network model into a physical configuration for the global network view that the SDN controller exposes. The principles, building blocks, and architecture of SDN are shown in Figure 1.

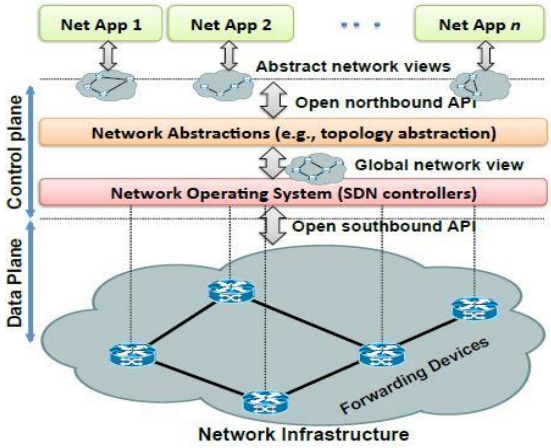


Fig. 1. SDN architecture and its fundamental abstractions.

SDN, on the other hand, separates the control plane from the network devices and turns the network operating system, or SDN controller, into an external entity. This strategy offers a number of benefits:

- All applications can utilize the same network information (the global network view), which may lead (arguably) to more consistent and effective policy decisions while reusing control plane software modules.
- These applications can take actions (i.e., reconfigure forwarding devices) from any part of the network.
- These applications become easier to program since the abstractions provided by the control platform and/or the network programming languages can be shared. Therefore, there's no need to plan out exactly where the new functionality will be placed.
- The process of integrating various applications gets easier [25]. Applications for load balancing and routing, for example, can be integrated successively, with load-balancing choices taking priority over routing guidelines.

Terminology We now list the key terms used throughout this work to identify the various components of an SDN.
Forwarding Devices (FD) are data plane devices, either hardware- or software-based, that carry out several basic operations. The incoming packets are subjected to actions by the forwarding devices, such as forwarding to designated ports, dropping, forwarding to the controller, and rewriting headers, using well-defined instruction sets, such as flow rules. These instructions are installed in the forwarding devices by the SDN controllers that implement the southbound protocols. They are defined by southbound interfaces, such as OpenFlow [2], For CES [26], and Protocol-Oblivious Forwarding (POF) [27]. Data Plane (DP): Wired cables or wireless radio channels are used to connect forwarding devices. The data plane is made up of the network infrastructure's networked forwarding devices. Interface Southbound (SI): The southbound API, which is a component of the southbound interface, defines the instruction set of the forwarding devices. Moreover, the SI specifies the protocol for communication between control plane elements and forwarding devices. The communication between the control and data plane components is codified in this protocol.

Control Plane (CP): Using precisely defined SI embodiments, control plane elements program forwarding devices. Therefore, the control plane can be thought of as the "network brain."
The controllers and applications that make up the control plane contain all of the control logic. Northbound Interface (NI): Application developers may be provided with an API by the network operating system. An application development common interface, or northbound interface, is represented by this API. A northbound interface typically abstracts to program forwarding devices the low-level instruction sets utilized by southbound interfaces.

International Multidisciplinary Research Journal Reviews (IMRJR)
A Peer-reviewed journal
Volume 1, Issue 3, November 2024
DOI 10.17148/IMRJR.2024.010304

Management Plane (MP): The management plane consists of a collection of programs that use the NI's functions to implement operation logic and network control. Applications like load balancers, firewalls, routing, monitoring, and so on are included in this. In essence, the policies are defined by a management application, and these are then converted into instructions specific to southbound traffic and used to program the forwarding devices' behavior.

There have been more recent attempts to define SDN using a layered approach [28], [10]. One initiative at IRTF SDNRG [28] proposes a management plane at the same level as the control plane, i.e., it classifies solutions into two categories: control logic (with control plane southbound interfaces) and management logic (with management plane southbound interfaces). This is done from a practical standpoint while still attempting to maintain backward compatibility with existing network management approaches. Stated differently, the management plane can be understood as a control platform that supports the protocols and services associated with traditional network management, including SNMP [11], BGP [29], PCEP [30], and NETCONF [31].

Apart from the broader definitions mentioned above, the term SDN is frequently employed to describe a variety of systems, including open-source data planes (Pica8 Xorplus [32], Quagga [33], specialized programmable hardware devices (NetFPGA [34]), virtualized software-based appliances (Open Platform for Network Functions Virtualization - OPNFV [35]), extensible network management planes (OpenStack), white-box / bare-metal switches with open operating systems ( Cumulus Linux [32], open-source data planes (Pica8 Xorplus [32], Quagga [33]), and open-source switches with open operating systems (OpenStack).

The SDN (and SDN-related issues) standardization landscape is already broad and is predicted to continue to grow. While some of the work is being done in Standard Development Organizations (SDOs), related efforts are also being done at community or industrial consortia (like Open Daylight, OpenStack, and OPNFV) and have produced results that are frequently thought to be contenders for de facto standards. These outcomes frequently take the shape of open-source implementations, which have evolved into a standard approach for advancing SDN and associated networking and cloud technologies [36]. SDN concepts span several IT and networking domains, which contributes to this fragmentation from access to core network segmentation standpoint as well as from a technology standpoint (from optical to wireless).

Network Function Virtualization (NFV) is being worked on at the European Telecommunication Standards Institute (ETSI) with the help of a recently established Industry Specification Group (ISG). The goals of NFV and SDN are seen as complementary; they both aim to increase network innovation by enabling programmability and completely altering the network operating model through automation and a move toward software-based platforms. Last but not least, the 3GPP consortium for the mobile networking industry is researching virtualized network management. This project is in line with the ETSI NFV architecture and will therefore probably benefit from SDN.

## IV. SOFTWARE-DEFINED NETWORKS: BOTTOM-UP

As seen in Figure 2(b), an SDN architecture can be represented as a combination of various layers. Every layer serves a specific purpose. Certain components, like the northbound API, southbound API, network operating systems, and network applications, are always present in an SDN deployment; however, other components, like hypervisor- or language-based virtualization, might only be present in specific deployments.

Figure 2 shows SDNs from a tri-fold perspective. As previously mentioned, the SDN layers are depicted in the figure's center (b). A plane-oriented view and a system design perspective are shown in Figures 2(a) and 2(c), respectively.

Each layer is introduced in the sections that follow, working from the bottom up. The fundamental characteristics and ideas of each layer are described in terms of the various technologies and solutions. Furthermore, methods and resources for debugging and troubleshooting are covered.

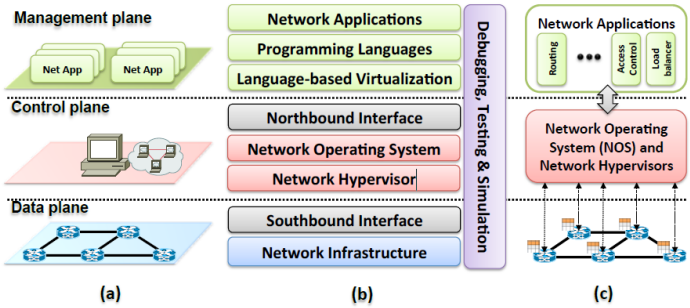


Fig. 2. Software-defined networks in (a) planes, (b) layers, and (c) system design architecture

A. Layer I: Like a conventional network, an SDN infrastructure is made up of a variety of networking devices, such as switches, routers, and middlebox appliances. The primary distinction is that those once-dominant physical devices are now merely forwarding components without embedded control or software capable of making decisions on their own. As seen in Figure 2(c), the network operating system and applications serve as a logically centralized control system that

International Multidisciplinary Research Journal Reviews (IMRJR)
A Peer-reviewed journal
Volume 1, Issue 3, November 2024
DOI 10.17148/IMRJR.2024.010304

receives the network intelligence from the data plane devices. What's more, these new networks are (conceptually) constructed on top of open and standard interfaces (like OpenFlow), which is an essential strategy for guaranteeing interoperability and compatibility between various data and control plane devices in terms of configuration and communication. Stated differently, because of the wide range of proprietary and closed interfaces and the distributed nature of the control plane in traditional networks, these open interfaces make it possible for controller entities to dynamically program heterogeneous forwarding devices.

B. Layer II: Interfaces that go south Southbound interfaces, also known as southbound APIs, serve as the vital link between the control and forwarding elements, making them the essential tool for distinct control and data plane functionality. These APIs are still, however, closely linked to the forwarding components of the underlying virtual or physical infrastructure.

C. Layer III: Network Hypervisors

In contemporary computers, virtualization is already a standardized technology. The virtualization of computing platforms has become commonplace due to the rapid advancements of the last ten years. Recent reports indicate that there are already more virtual servers than physical servers [37], [38].

D. Layer IV: Controllers and Operating Systems for Networks Conventional operating systems control the concurrent access to the underlying resources (such as the hard drive, network adapter, CPU, and memory), offer abstractions (such as high-level programming APIs) for accessing lower-level devices, and offer security protection mechanisms. System and application developers' lives are made easier by these features and resources, which are important facilitators of higher productivity. The development of numerous applications and the evolution of different ecosystems (like programming languages) have both been greatly aided by their widespread use.

E. Layer V: Interfaces heading north Two important abstractions in the SDN ecosystem are the North- and Southbound interfaces. A common northbound interface is still up for debate, but a widely accepted proposal for the southbound interface – OpenFlow - has already been made. Since use cases are still being developed, it might still be too early to define a standard northbound interface at this time [39]. In any case, as SDN develops, a common (or de facto) northbound interface should emerge. To fully utilize SDN, network applications must have an abstraction that frees them from particular implementations.

F. Layer VI: Language-oriented Virtualization The ability to express modularity and to support various levels of abstraction while maintaining desired features like protection are two crucial aspects of virtualization solutions. For example, various views of a single physical infrastructure can be enabled by virtualization techniques. One virtual "big switch," for instance, might be a conglomeration of multiple underlying forwarding devices. Because they can now view the network as a straightforward "big switch" rather than having to consider the order in which forwarding rules must be installed, application developers' work is inherently made simpler. The development and implementation of sophisticated network applications, such as those about advanced security, are made much simpler by this type of abstraction.

G. Layer VII: Languages used in programming for many years, programming languages have been abundant. High-level and potent programming languages like Java and Python have replaced low-level, hardware-specific machine languages like assembly for x86 architectures in academia and industry. The computer industry has undergone a dramatic change due to developments toward more reusable and portable code [40], [41].

H. Layer VIII: Applications for Networks The "network brains" are represented by network applications. They put into practice the control logic that will be converted into data plane commands that will control how the forwarding devices behave. Take a simple application as routing as an example. This application's logic is to specify the route that packets will take to get from point A to point B. To accomplish this, a routing application must select the path to take from A to B based on the topology input and give the controller instructions to install the necessary forwarding rules in each forwarding device along the path.

A. Research Issues of SDN

SDN has various research issues like Security, Performance, Scalability and many more. The division of the control and data planes in SDN opens up new possibilities for creativity and adaptability. But it also brings with its special security problems and other weaknesses that hackers could take advantage of. There are drawbacks to SDN's centralized architecture, in which the controller, or control plane, controls and routes network traffic. Even while it makes management and orchestration easier, a single controller-level failure could cause extensive network interruption or compromise [42 – 47].

I. Security Issues with SDN

a. The vulnerability of the controller: Since the controller is the brain of the network, it has a lot of power, and if it were compromised, an attacker could take over or interfere with the entire system. Because of this, controllers are desirable targets for privilege escalation, unauthorized access, and Denial-of-Service (DoS) attacks.

b. Tampering and Modification of Flow Rules: Protocols like OpenFlow, which are typically used to handle communication between the control plane and data plane, might be weak if improperly secured. Unintentional

International Multidisciplinary Research Journal Reviews (IMRJR)
A Peer-reviewed journal
Volume 1, Issue 3, November 2024
DOI 10.17148/IMRJR.2024.010304

network activity, packet interception, or data leakage can result from malicious actors intercepting or changing flow rules.
c. Threats to the Data Plane: Data plane devices, such switches and routers, are in charge of carrying out the commands from the control plane. However, if these devices are not secure, attackers might circumvent network regulations and perhaps introduce malware into the system by inserting malicious packets or traffic flows.
d. Security of Northbound and Southbound APIs: SDN controllers connect to switches (southbound) and applications (northbound) via APIs. Unauthorized access or control over network components may result from malicious requests or commands being injected into these APIs. Although protecting these APIs is essential, it can be difficult because security must be balanced with flexibility and simplicity of integration.

II. SDN Security Research Directions

a. Systems for detecting and preventing intrusions (IDPS): It is crucial to create sophisticated IDPS that work with SDN architecture. By tracking network traffic patterns and identifying irregularities instantly, these systems offer a preemptive defense against possible dangers. Some methods improve anomaly detection using machine learning, enabling these systems to automatically adjust and pick up on new threats.
b. Security via Blockchain: Applications of blockchain technology in SDN security are promising, especially when it comes to protecting flow rule transactions between the data and control planes. A decentralized ledger makes it more difficult for attackers to alter flow rules covertly because every modification is recorded in an unchangeable chain.
c. Protocols for Secure Communication in SDN: To ensure secure control-data plane interactions, protocols such as OpenFlow require improvements. Although Transport Layer Security (TLS) is extensively used, research is being done on safe key management systems and advanced encryption algorithms to increase its resilience to complex attacks.
d. Mechanisms for Access Control: To stop unwanted access, the controller must have strict access control implemented. To reduce the likelihood of insider threats and illegal acts, two strategies are being investigated: Role-Based Access Control (RBAC) and Attribute-Based Access Control (ABAC). These strategies limit access according to user roles or attributes, respectively.

III. Model Research and Structures

The Princeton-developed FRESCO (architecture for Security in OpenFlow Networks) architecture offers an expandable platform for SDN security applications. It improves the controller's capacity to monitor, identify, and neutralize threats by enabling the quick deployment and prototyping of security modules.

Security Service for SDN Networks (SENSS): Designed to counter Distributed Denial-of-Service (DDoS) attacks, SENSS uses SDN's centralized control plane to identify and stop DDoS attacks instantly by blocking or rerouting malicious traffic according to flow statistics and traffic patterns.

IV. Prospects for the Future

Researchers continue to place a high premium on resolving security issues as SDN develops. There is a lot of interest in creating AI-powered security solutions that can proactively recognize, understand, and neutralize new threats. To strengthen encryption standards and make SDN communications resistant to even the most sophisticated adversaries, quantum cryptography integration is also being investigated.
To sum up, security in SDN includes a number of crucial components, such as strong access control systems, flow rule integrity, and controller protection. To provide comprehensive security solutions that can take advantage of SDN's centralized architecture without sacrificing its efficiency and flexibility, more research is needed.

## V. ONGOING RESEARCH EFFORTS, CHALLENGES & CONCLUSION

One of the causes is the vertical integration and vendor-specificity of the control and data planes. The fact that common networking devices are closely related to line products and versions is another supporting factor. To put it another way, every product line may have unique configurations and management interfaces, which suggests lengthy production cycles for upgrades or product updates (like new firmware or device versions). All of this has resulted in significant barriers to change and innovation and caused vendor lock-in issues for owners of network infrastructure.
The emergence of Software-Defined Networking (SDN) has allowed for the resolution of these enduring problems. Among the core ideas of SDN are the separation of the control and data planes, the development of a logical central "network brain," and the introduction of dynamic programmability in forwarding devices through open southbound interfaces. The data plane elements developed into simple, programmable, extremely effective packet forwarding devices, while the control plane elements are now represented by a single entity, the controller or network operating system.

International Multidisciplinary Research Journal Reviews (IMRJR)
A Peer-reviewed journal
Volume 1, Issue 3, November 2024
DOI 10.17148/IMRJR.2024.010304

Applications that implement the network logic run on top of the controller and are much easier to create and implement than those that use traditional networks.

Adopting a global perspective makes enforcing policy consistency easy. SDN is a major paradigm shift in the design and development of networks, and it has sped up the rate of innovation in networking infrastructure.

A single thorough and comprehensive overview of the fundamentals, concepts, and difficulties of SDNs has not yet been found in the literature, despite some recent and intriguing attempts to survey this new chapter in the history of networks [8], [9], and [10]. To close this gap, the current paper employed a tiered approach to thoroughly analyze the state of the art about software-defined networking concepts, ideas, and components, covering a wide range of current solutions as well as potential future paths. First, we made a comparison between this new paradigm and conventional networks, and we talked about how academia and industry influenced software-defined networking.

Using a bottom-up methodology, we presented a comprehensive synopsis of what we believe to be the eight core aspects of the SDN problem: The order of importance is as follows: 1) hardware infrastructure; 2) southbound interfaces; 3) network virtualization (a layer acting as a hypervisor between the network operating systems and the forwarding devices); 4) network operating systems (common programming abstractions made available to network applications); 5) northbound interfaces; 6) virtualization through the use of slicing techniques made possible by special purpose libraries, programming languages, and compilers; 7) network programming languages; and 8) network applications.

Through the creation of a cutting-edge research and development environment, the advancement of switch and controller platform design, the evolution of device and architecture scalability and performance, and the promotion of security and dependability, SDN has successfully paved the way for next-generation networking.

Shortly, there will be a lot more activity related to SDN. Some of the emerging topics that need more research are software-defined environments (SDE), networks-as-a-service cloud computing, the path to SDN migration, and extending SDN towards carrier transport networks.

## REFERENCES

[1]. H. Kim and N. Feamster, “Improving network management with software-defined networking,” Communications Magazine, IEEE, vol. 51, no. 2, pp. 114–119, 2013.

[2]. N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, “OpenFlow: enabling innovation in campus networks,” SIGCOMM Comput. Commun. Rev., vol. 38, no. 2, pp. 69–74, Mar. 2008.

[3]. ONF, “Open networking foundation,” 2014. [Online]. Available: https://www.opennetworking.org/

[4]. S. Yeganeh, A. Tootoonchian, and Y. Ganjali, “On scalability of software-defined networking,” Communications Magazine, IEEE, vol. 51, no. 2, pp. 136–141, 2013.

[5]. S. Jain, A. Kumar, S. Mandal, J. Ong, L. Poutievski, A. Singh, S. Venkata, J. Wanderer, J. Zhou, M. Zhu, J. Zolla, U. H¨olzle, S. Stuart, and A. Vahdat, “B4: experience with a globally-deployed software defined wan,” in Proceedings of the ACM SIGCOMM 2013 conference on SIGCOMM, ser. SIGCOMM ’13. New York, NY, USA: ACM, 2013, pp. 3–14.

[6]. VMware, Inc., “VMware NSX Virtualization Platform,” 2013. [Online]. Available: https://www.vmware.com/products/nsx/

[7]. OpenDaylight, “OpenDaylight: A Linux Foundation Collaborative Project,” 2013. [Online]. Available: http://www.opendaylight.org

[8]. A. Lara, A. Kolasani, and B. Ramamurthy, “Network innovation using OpenFlow: A survey,” Communications Surveys Tutorials, IEEE, vol. 16, no. 1, pp. 493–512, First 2014.

[9]. B. Nunes, M. Mendonca, X.-N. Nguyen, K. Obraczka, and T. Turletti, “A survey of software-defined networking: Past, present, and future of programmable networks,” Communications Surveys Tutorials, IEEE, vol. 16, no. 3, pp. 1617–1634, Third 2014.

[10]. Y. Jarraya, T. Madi, and M. Debbabi, “A survey and a layered taxonomy of software-defined networking,” Communications Surveys Tutorials, IEEE, vol. PP, no. 99, pp. 1–1, 2014.

[11]. R. Presuhn, “Version 2 of the Protocol Operations for the Simple Network Management Protocol (SNMP),” RFC 3416 (INTERNET STANDARD), Internet Engineering Task Force, Dec. 2002. [Online]. Available: http://www.ietf.org/rfc/rfc3416.txt

[12]. T. Benson, A. Akella, and D. Maltz, “Unraveling the complexity of network management,” in Proceedings of the 6th USENIX Symposium on Networked Systems Design and Implementation, ser. NSDI’09, Berkeley, CA, USA, 2009, pp. 335–348.

[13]. A. Ghodsi, S. Shenker, T. Koponen, A. Singla, B. Raghavan, and J. Wilcox, “Intelligent design enables architectural evolution,” in Proceedings of the 10th ACM Workshop on Hot Topics in Networks, ser. HotNets-X. New York, NY, USA: ACM, 2011, pp. 3:1–3:6.

[14]. B. Raghavan, M. Casado, T. Koponen, S. Ratnasamy, A. Ghodsi, and S. Shenker, “Software-defined internet architecture: Decoupling architecture from infrastructure,” in Proceedings of the 11th ACM Workshop on Hot Topics in Networks, ser. HotNets-XI. New York, NY, USA: ACM, 2012, pp. 43–48.

**International Multidisciplinary Research Journal Reviews (IMRJR)**
A Peer-reviewed journal
Volume 1, Issue 3, November 2024
DOI 10.17148/IMRJR.2024.010304

[15]. J. Pan, S. Paul, and R. Jain, "A survey of the research on future internet architectures," Communications Magazine, IEEE, vol. 49, no. 7, pp. 26–36, July 2011.
[16]. N. Feamster and H. Balakrishnan, "Detecting BGP configuration faults with static analysis," in Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation - Volume 2, ser. NSDI'05. Berkeley, CA, USA: USENIX Association, 2005, pp. 43–56.
[17]. R. Barrett, S. Haar, and R. Whitestone, "Routing snafu causes internet outage," Interactive Week, 1997.
[18]. K. Butler, T. Farley, P. McDaniel, and J. Rexford, "A survey of BGP security issues and solutions," Proceedings of the IEEE, vol. 98, no. 1, pp. 100–122, Jan 2010.
[19]. J. Sherry and S. Ratnasamy, "A survey of enterprise middlebox deployments," EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2012-24, Feb 2012.
[20]. K. Greene, "MIT Tech Review 10 Breakthrough Technologies: Software-defined Networking," http://www2.technologyreview.com/article/412194/tr10-software-defined-networking/, 2009.
[21]. P. Newman, G. Minshall, and T. L. Lyon, "Ip switching—atm under ip," IEEE/ACM Trans. Netw., vol. 6, no. 2, pp. 117–129, Apr.1998.
[22]. N. Gude, T. Koponen, J. Pettit, B. Pfaff, M. Casado, N. McKeown, and S. Shenker, "NOX: towards an operating system for networks," Comp. Comm. Rev., 2008.
[23]. H. Jamjoom, D. Williams, and U. Sharma, "Don't call them middleboxes, call them middlepipes," in Proceedings of the Third Workshop on Hot Topics in Software Defined Networking, ser. HotSDN '14. New York, NY, USA: ACM, 2014, pp. 19–24.
[24]. H. Alkhatib, P. Faraboschi, E. Frachtenberg, H. Kasahara, D. Lange, P. Laplante, A. Merchant, D. Milojicic, and K. Schwan, "IEEE CS 2022 report (draft)," IEEE Computer Society, Tech. Rep., February 2014.
[25]. M. Casado, N. Foster, and A. Guha, "Abstractions for software-defined networks," Commun. ACM, vol. 57, no. 10, pp. 86–95, Sep. 2014.
[26]. A. Doria, J. H. Salim, R. Haas, H. Khosravi, W. Wang, L. Dong, R. Gopal, and J. Halpern, "Forwarding and Control Element Separation (ForCES) Protocol Specification," Internet Engineering Task Force, Mar. 2010. [Online]. Available: http://www.ietf.org/rfc/rfc5810.txt
[27]. H. Song, "Protocol-oblivious Forwarding: Unleash the power of SDN through a future-proof forwarding plane," in Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking, ser. HotSDN '13. New York, NY, USA: ACM, 2013, pp. 127–132.
[28]. E. Haleplidis, S. Denazis, K. Pentikousis, S. Denazis, J. H. Salim, D. Meyer, and O. Koufopavlou, "SDN Layers and Architecture Terminology," Internet Draft, Internet Engineering Task Force, September 2014. [Online]. Available: http://www.ietf.org/id/draft-irtfsdnrg-layer-terminology-02.txt
[29]. Y. Rekhter, T. Li, and S. Hares, "A Border Gateway Protocol 4 (BGP-4)," RFC 4271 (Draft Standard), Internet Engineering Task Force, Jan. 2006. [Online]. Available: http://www.ietf.org/rfc/rfc4271.txt
[30]. J. Vasseur and J. L. Roux, "Path Computation Element (PCE) Communication Protocol (PCEP)," RFC 5440 (Proposed Standard), Internet Engineering Task Force, Mar. 2009. [Online]. Available: http://www.ietf.org/rfc/rfc5440.txt
[31]. R. Enns, M. Bjorklund, J. Schoenwaelder, and A. Bierman, "Network Configuration Protocol (NETCONF)," RFC 6241 (Proposed Standard), Internet Engineering Task Force, Jun. 2011. [Online]. Available: http://www.ietf.org/rfc/rfc6241.txt
[32]. A. Shang, J. Liao, and L. Du, "Pica8 Xorplus," 2014. [Online]. Available: http://sourceforge.net/projects/xorplus/
[33]. P. Jakma and D. Lamparter, "Introduction to the quagga routing suite," Network, IEEE, vol. 28, no. 2, pp. 42–48, March 2014.
[34]. "NetFPGA," 2014. [Online]. Available: http://netfpga.org/
[35]. Linux Foundation, "Open platform for NFV," https://www.opnfv.org, Sep 2014.
[36]. C. E. Rothenberg, R. Chua, J. Bailey, M. Winter, C. Correa, S. Lucena, and M. Salvador, "When open source meets network control planes," IEEE Computer Special Issue on Software-Defined Networking, November 2014.
[37]. T. Koponen, K. Amidon, P. Balland, M. Casado, A. Chanda, B. Fulton, I. Ganichev, J. Gross, P. Ingram, E. Jackson, A. Lambeth, R. Lenglet, S.-H. Li, A. Padmanabhan, J. Pettit, B. Pfaff, R. Ramanathan, S. Shenker, A. Shieh, J. Stribling, P. Thakkar, D. Wendlandt, A. Yip, and R. Zhang, "Network virtualization in multi-tenant datacenters," in 11th USENIX Symposium on Networked Systems Design and Implementation (NSDI 14), Seattle, WA, Apr. 2014, pp. 203–216.
[38]. T. J. Bittman, G. J. Weiss, M. A. Margevicius, and P. Dawson, "Magic Quadrant for x86 Server Virtualization Infrastructure," Gartner, Tech. Rep., June 2013.
[39]. J. Dix, "Clarifying the role of software-defined networking northbound APIs," May 2013. [Online]. Available: http://www.networkworld.com/news/2013/050213-sherwood-269366.html
[40]. M. Guzdial, "Education: Paving the way for computational thinking,"Commun. ACM, vol. 51, no. 8, pp. 25–27, Aug. 2008.
[41]. M. S. Farooq, S. A. Khan, F. Ahmad, S. Islam, and A. Abid, "An evaluation framework and comparative analysis of the widely used first programming languages," PLoS ONE, vol. 9, no. 2, 02 2014.

**International Multidisciplinary Research Journal Reviews (IMRJR)**
A Peer-reviewed journal
Volume 1, Issue 3, November 2024
DOI 10.17148/IMRJR.2024.010304

[42]. Kreutz, D., Ramos, F. M. V., Verissimo, P. E., Rothenberg, C. E., Azodolmolky, S., & Uhlig, S. (2015). "Software-defined networking: A comprehensive survey."Proceedings of the IEEE, 103(1), 14-76.

[43]. Scott-Hayward, S., Natarajan, S., & Sezer, S. (2016). "A survey of security in software-defined networks."IEEE Communications Surveys & Tutorials, 18(1), 623-654.

[44]. Nanda, P., & Chiueh, T. C. (2018). Anomaly-based intrusion detection in software-defined networking using machine learning."IEEE Access, 6, 56041-56051.

[45]. Sharma, P. K., & Park, J. H. (2018). "Blockchain based hybrid network architecture for the smart city."Future Generation Computer Systems, 86, 650-655.

[46]. Porras, P., Shin, S., Yegneswaran, V., Fong, M., Tyson, M., & Gu, G. (2012). "A security enforcement kernel for OpenFlow networks."Proceedings of the first workshop on Hot topics in software defined networks (HotSDN), 121-126.

[47]. Khan, S., Gani, A., Wahab, A. W. A., Shiraz, M., & Ko, K. (2018). "Network forensics: Review, taxonomy, and open challenges."Journal of Network and Computer Applications, 66, 214-235.