An Design of Software Defined Networks and Possibilities of Network Attacks

[1]Anandakumar Haldorai and [2]Shrinand Anandakumar
[1]Department of Computer Science and Engineering, Sri Eshwar College of Engineering, India.
[2]The PSBB Millennium School, Coimbatore, India.
[1]anandakumar.psgtech@gmail.com, [2]shrinand.psbb@gmail.com

Article Info
Journal of Computing and Natural Science (http://anapub.co.ke/journals/jcns/jcns.html)
Doi: https://doi.org/10.53759/181X/JCNS202202012
Received 30 December 2021; Revised form 26 March 2022; Accepted 28 April 2022.
Available online 05 July 2022.
©2022 Published by AnaPub Publications.

Abstract – This article focusses on a rapidly evolving networking architecture known as Software Defined Networking (SDN) and the possibilities of hazards in the network. This architecture introduces decoupled infrastructure, which establishes customization in the networking system hence making it easy to manage, troubleshoot and configure. This paper focusses on the different aspects of the architecture leaving it an intermediate working in between scholarly application, adding on the elements such as security lapses, security behaviors, general security, programmability and design. In this paper, different points of weakness of the architecture have been evaluates, including the attack vector in every plane. This paper ends with a presentation for futuristic studies on the implications of attacks and potential solutions.

Keywords – Software Defined Networking (SDN), Application Programming Interface (API), Network Virtualization (NV).

I. INTRODUCTION

Software Defined Networking (SDN) is a quickly developing infrastructure beheading the tradition networking infrastructure concentrating its demerits in a limited aspect [1]. Some few decades ago, networking and programming were visualized as various domains with the introduction of SDN integrated themselves. This is to alleviate the current problems in the networking area and to advocate those networked devices and information be handled collectively. In order to make troubleshooting, management, and configuration simpler, SD N delivers a decoupled architecture and enables flexibility inside networks. SDN is a revolutionary paradigm in computer networking that promises to fundamentally alter how networks are configured and traffic is managed in real time.

Since its inception, SDN may be linked back to a variety of traffic control and network administration solutions. Since the beginning, one of the primary goals of developing centralised network management primitives has been to boost overall system performance and bring management capabilities to select parts of the network. It is widely accepted that SDN represents the result of these efforts. SDN is defined by the Open Networking Foundation (ONF) as "the detachment of the networking control and data planes from the forwarding devices or where a central controller manages many devices. Due to its tendency to decouple and totally programme the data plane from control logic, SDN frameworks reduce manual fine tuning of specific hardware elements.

All hardware is dynamically configured and managed according to end-user application needs, thanks to the paradigm's new centralized management structure. The control plane provides a high degree of abstraction that may be utilised by software developers to specify active network resource utilisation models and to optimise the network architecture fabric in line with changing service needs. Administrators and executives profit greatly from managing a diverse range of network equipment based on real-time traffic circumstances in order to efficiently provide services and provide business and technology updates. Academics and industry experts alike are paying close attention to SDN. It was formed in conjunction with OpenFlow Network Research Center (ONRC), which is an ONF industry conglomerate [2]. For a period of time, linked devices suffocate in their ability to perform their intended functions due to rising demands. Several factors have been cited as contributing to this problem, including the sheer volume of data, the rapid expansion of interconnected devices, and the requirement for high-speed data processing. As a result, security has been one of the most important operating considerations throughout this scenario, voiding any previous mitigating proposals. Suffocation can be alleviated by looking at both the present architecture of smart objects and the recent changes that have occurred over the last decade.

This paper discusses software-defined networking, one of several fascinating new ideas that have emerged in recent decades, and presents an analysis of the possibilities of attacks and vulnerabilities. Software-Defined Networking (SDN) is a novel movement in the segment of networks that aims to address the current issues experienced by traditional networked devices. Various faults were discovered in the course of the architecture's development and were made available to researchers in order to come up with mitigation strategies. Multiple network model actions were the subject of these bugs.

After a description of traditional network design and SDN architecture, this work channels down into the security components of the effort to achieve both quantitative and qualitative development.

This study focuses solely on the security concerns of SDN design, as there has already been sufficient research into the security of conventional network architecture. On the application layer of SDN framework, this effort aims to detect and address security issues. A fundamental security risk in SDN design has been highlighted in this paper, paving the way for additional research into how to mitigate this problem. Because of the SDN architecture's disconnected approach to interacting with subsystems and its capability to provide configurability features, this is a unique security challenge. This paper has been organized as follows: Section II focusses on related works about SDN. Section III focusses on a critical analysis of network programmability, network virtualization, and software defined networking. Section IV presents a review of the possibilities of attacks, which Section V draws conclusions to the research, and proposes further research directions.

II. LITERATURE REVIEW

Switches and routers are no longer tied together by a common control logic, thanks to Software Defined Networking (SDN). It fosters the logical abstraction of network management and provides the capability of configuring the network... Network functions could be made more flexible, responsive, and programmable as a result There are advantages to using a microcontroller to manage all forwarding devices, such as packet processor speed and throughput. Stanford and UC Berkeley collaborated on a project that resulted in SDN. It is possible to detach the control plane of an SDN network from the data plane, which is responsible for sending traffic to its destination. To accommodate numerous applications and services, the network's infrastructure can be abstracted to allow for direct programming. Experts and companies argue that this considerably simplifies the connectivity process. **Fig. 1** depicts a typical SDN structure based on the OpenFlow protocols, which has three layers instead of the standard two.

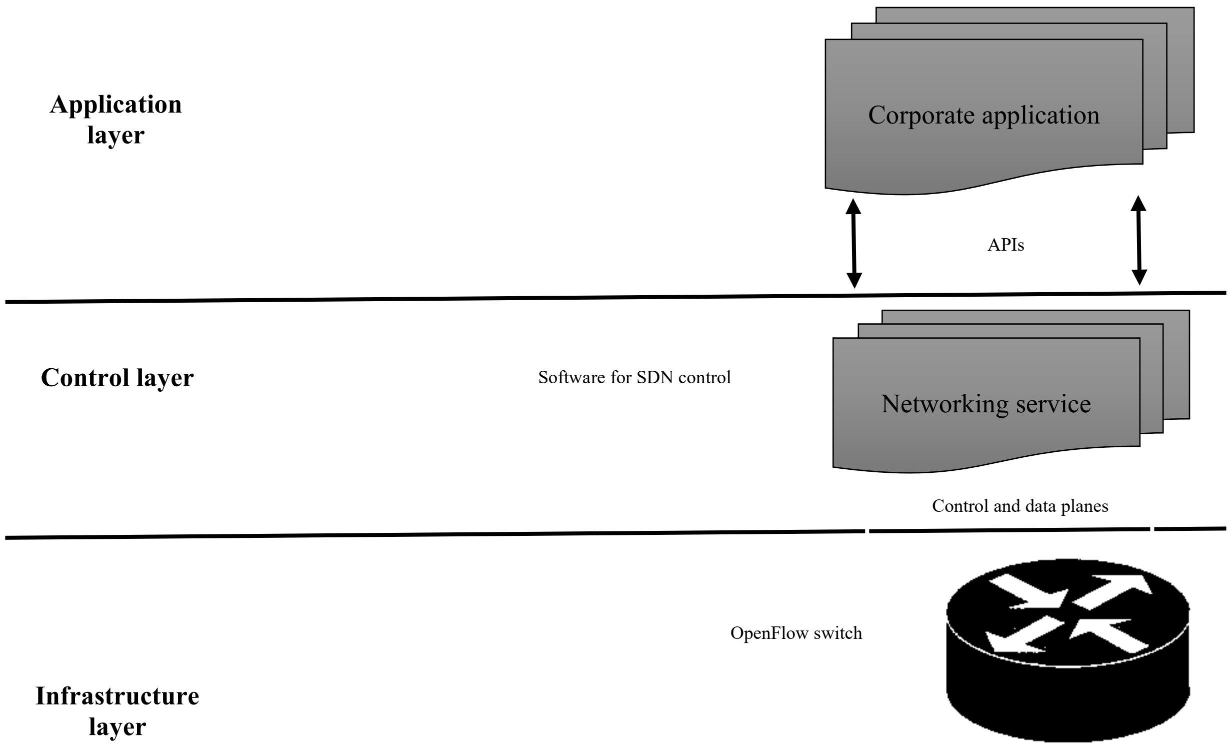

Fig 1. SDN architecture's layers

Well-defined programmatic interfaces between the controller and switches may be used to separate the control and data planes. The controller has complete command and control over the data plane elements. This is accomplished through an Application Programming Interface (API) [3]. A good example of this kind of API is OpenFlow, which is widely used today. Depending on the OpenFlow switch, packet handling rules are stored in one or more flow tables. **Table 1** shows the six fields that make up a flow table entry. Dropping, rerouting, and otherwise altering the traffic according to the rule is carried out on the subset of data that matches it. It is up to the flow tables to decide what to do with the packets that arrive.

Table 1. Entry fields of the flow table

Fields	Definition
Cookie	Opaque dataset sent by the OpenFlow controllers
Timeout	Maximum efficiency timeline of free timeline before an overdue entry
Instruction	Pipeline or action processing
Counter	Statistics for packet matching
Priority	Matching precedences of entries
Match	Port, metadata forwarded from the last flow table

Due to the dissociation of the planes, the communication system between the controller and switches is critical. **Fig. 2** depicts how the OpenFlow protocol specifies the API for communications between SDN devices.

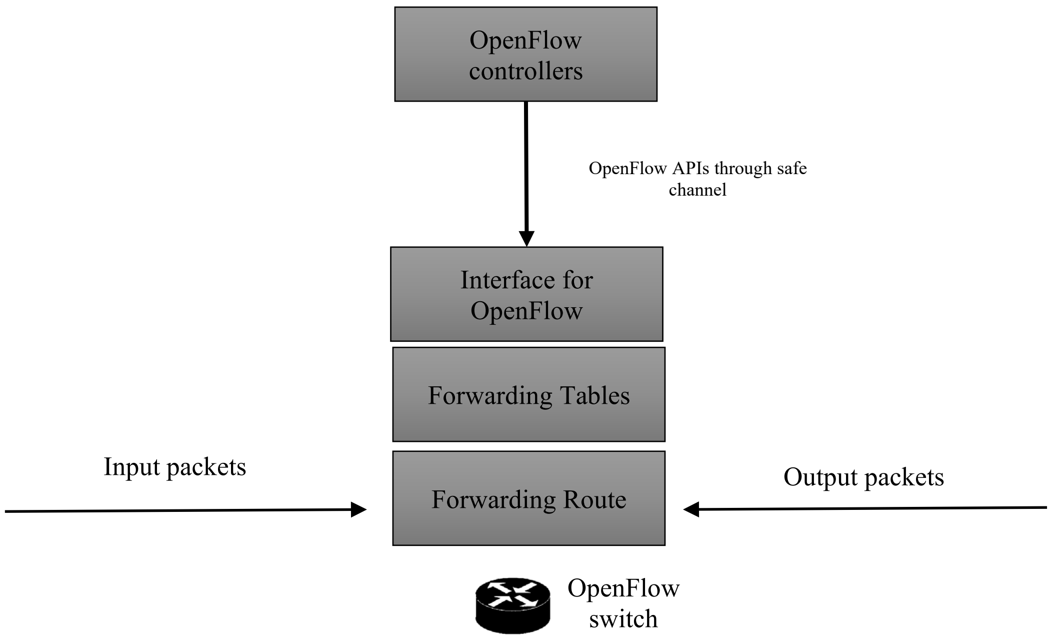

Fig 2. OpenFlow protocol structure

It is possible to change and remove flow entries in a switch's table of flows using SDN controllers. An OpenFlow controller receives a packet of data from a switch and conducts the operation either reactively or proactively. Using a secure communication channel, the controller and switch may communicate. To support flow-based forwarding, the OpenFlow switch maintains a flow table or tables.

Efficiency of SDN controllers in the context of packet-in message hybrids Poisson channels has been estimated by Oliveira, Xavier-de-Souza, and Silveira [4]. Using packet sojourn time, they depicted OpenFlow packet forwarding. Analysis by Cho, Lee, Kim, and Ryoo [5] of an SDN switch's flow-table size, packet delivery rate, rule quantity, and rule positioning are included. An end-to-end approach to delay estimation and management was taken into consideration in this research. To approach the Ibrahim and Whitt's [6] concept, an open Jackson networks with a controller can be used. An algorithm for traffic control and a hybrid route forwarding network have been suggested by Giocomazzi, Musumeci and Verticale [7] for traffic planning and load distribution in software-defined cellular networks. In the context of bursty and linked arrivals, Miao et al. tested SDN performance using the Markov Modulated Poisson Process (MMPP).

These interconnected devices are always linked, and the smothering impact they have on people are examined here, as well as significant advances made over the last decade in this area. An increasing number of interesting and innovative concepts have developed in recent years, and this paper focuses on one of them: Software-Defined Networking (SDN). One of the emerging initiatives in the battle against the inadequacies of conventionally connected devices is SDN.

III. CRITICAL ANALYSIS

Network Programmability

It's no secret that network managers wish for more programmability in network devices since the current way of setup (mostly through CLI) is effective but sluggish and labor-intensive when it comes to altering settings as networks develop. The US Défense and Advanced Research Projects Agency (DARPA) [8] projected the issue of incorporating novel frameworks into the present system architecture as early as the early 1990s and its reconfigurations needed. Similar to the phrase active networks, which advocates unique calculations on packets in order to drastically minimize individual device programmability, this word was coined about the same time. Firewalls or application services, such as a router trace programme, might have been employed, as could active nodes receiving new network directives. It failed to take off due to the absence of low-cost network connectivity and the lack of an obvious application like today's cloud services. During the mid-1990s, another network programming project was the ATM Network Devolved Control. Development of infrastructure and services for ATM network scalability was DCAN's primary goal. DCN technology is based on the idea that choices about ATM switch management should be detached from the devices and instead be assigned to a third party, the DCAN manager.

Programming instructions are used by the DCAN manager to control the network parts, much as SDN today. AT&T's GeoPlex was another initiative that aimed to incorporate automation into the network's infrastructure. To ensure that the project's networking infrastructure functioned, Java was used. GeoPlex was created to work on any Internet-connected device as a platform for controlling infrastructure and applications. Interoperability with user computer systems meant that the soft switch extrapolation cannot be used to reconfigure real devices. An important contribution to network programability was made by RFC 6121's Extensible Message And Presence Protocol (XMPP), which allowed for near real-time communication systems and the evaluation of availability in conjunction with messaging. Server-side contact data is maintained by XMPP clients that connect to the network's domain controller, which may inform other clients when an interaction is available. In data centre connectivity and the upcoming Internet of Things (IoT) framework, it is currently used to control system components since messages are deflected (in real time) instead of being represented as in SMTP/POP. It is possible to send CLI management requests via the Internet using the clients of the XMPP protocol on network clients. XMPP has gained a lot of momentum in the SDN (southbound API) space because of its widespread usage in legacy computing systems.

ALTO, an IETF working group's effort to improve P2P traffic via the identification of close peers, has been expanded to include discovering resources in network infrastructure. This information is then sent on to the ALTO server, which uses it to create a thorough orchestration for the currently running apps based on its understanding of the resources that are now accessible. An IETF initiative called Interface to the Routing System (I2RS) [9] proposes a divide between the centralized administration of the network and the applications that use it. Instead of a rigid centralised SDN, traditional routing techniques are implemented on networking hardware in I2RS. The system enables for the use of distributed routing while enabling specific applications to affect routing choices as needed. Standardization and working groups aren't the only places where network programmability has advanced. Some technology companies, such as Cisco, have made an effort to make it easier for developers to create applications that can be seamlessly integrated into a network. OnePK (Cisco one) is a programmable framework that allows users to tailor traffic flows and view network data to make deployments simpler as business demands change. An application-centric infrastructure, or ACI, is being developed by Cisco to better integrate software and hardware, based on operational needs.

Network Virtualization (NV)

The term "Network Virtualization" [10] refers to the ability to display many network topologies on a single piece of hardware or equipment. Virtualization has progressed from basic VLANs to a variety of intermediate technologies and testbeds. Among the noteworthy endeavors are those of Tempest, VINI, and Cabo. Tempest first presented the notion of decoupling control systems from switches at Cambridge in 1998. In ATM systems, Tempest was an early effort to segregate traffic relaying and control duties, and it worked. The Tempest project has several elements of today's Software-Defined Networking (SDN) technology. McCoy and Rawat [11] may use VINI to concurrently install and test multiple ideas on a variety of network topologies leveraging simulated routing technology, user data, and network signals.

Beginning in 2006, the VINI project was designed to evaluate new protocols and services. Using VINI-enabled networks, operators could run several protocol stacks on the same physical network using separate virtual switches that controlled traffic forwarding separately. It was proposed in 2007 that infrastructure and services be separated [12]. For this reason, Cabo advocates for the decoupling of infrastructures from service providers so that network operators may dynamically adjust their offerings to customers. Services may be deployed on network gear from multiple infrastructure providers utilizing Cabo's virtualization and bespoke traffic routing. Virtualization may be used to share resources like specialised memory assignment or traffic relaying program services that operate on the same system when numerous logical routers are operating on the same general-purpose computer.

Overlay technologies {e.g., VXLAN (Virtual Extensible Local Area Network)} have been created to circumvent the constraints of conventional networking technology. In VXLANs, a MAC-in-IP tunnel is used to connect the two endpoint switches. VXLAN-like GRE-based NV may be used instead. NVGRE has a slightly modified header structure for MAC-to-IP tunneling. To aid with load balance, VXLAN and NVGRE employ UDP-over-IP packet formats, whereas NVGRE makes use of the GRE standard protocol.

Stateless Transport Tunnels (STT) are used by MAC-in-IP tunneling, a contemporary virtualization technique. The concept of a virtual network is present in STT, but it is hidden under a context ID. More virtual systems and delivery models can be handled by STT context IDs since they are 64 bits long. STT utilizes the TCP Segmentation Offload (TSO) provided on many servers' network interface cards in order to surpass NVGRE and VXLAN (NICs). The quantity of data that may be sent to the NIC with a single TSO transmit request can be substantial, resulting in lower system overheads. STT employs TCP to unicast packets between tunnel ends in the stateless manner associated with TSO, as its name implies. Virtual machines may be operated on specialised hardware and storage for high-volume software platforms, such as DNS, remote access, firewall, and caching. Businesses are taking use of virtualization of network functions to further reduce operating and capital costs [16-20].

Evolving Architecture

Perspective of Design and Programmability

Requirement for SDN

A thorough rethink of the present networking architecture was necessary by the fast development of the Internet's infrastructure (public and private), as well as the widening range of applications. Conventional systems, which require a wide range of operations for traffic switching, forwarding, offering authentication, and maintaining quality of services, still require a great deal of coordination to employ distributed technologies. Conventional networking gear is limited to a small number of low-level configuration instructions, which makes monitoring many network devices and updating rules difficult. Due to the network's dependence on manual reconfiguration in response to shifting traffic patterns, errors are common. Tools now accessible might not be capable of providing the precision and mechanization required for the best potential configurations.

As a result, a new paradigm is needed in order to fulfill a wide range of operational requirements while allowing for innovative applications like dynamic deployment and provisioning and simple programming, all inside the same framework. Many technical and operational challenges were eventually resolved with the development of the SDN traffic management framework, including **Table 2** below.

Table 2. Technical and operational challenges solved with the SDN traffic management framework

Technical and Operational Challenge	Details
Automation	Automation and technology, which lowers overall operating costs and facilitates better troubleshooting, reduced unexpected downtime, ease of standardization, deployment of networking resources and appropriate application payloads as necessary, is the result.
Dynamic resource management	Assisting virtualization technology by dynamically modifying the network's topology and allotted resources, which could be further assisted
Orchestration	Coordinating the control of the majority number of network equipment, such as those found in data centers and big college campuses,
Multitenancy support	Renters are increasingly demanding total control over their address, topologies, route, and encryption in order to separate the tenanted architecture from managed services as the cloud platform market continues to develop.
Open APIs	Modular modules that give abstraction, define jobs through APIs, and are not concerned with implementation specifics. No particular protocol for communications between two nodes will be specified in this communication.
Greater Programmability	The capacity to adjust device behavior and configurations in actual time in response to changing traffic circumstances is a critical need for current network deployment.
Integrated security	Network fabric integration allows for more accurate detection of security problems and a more streamlined management process.
Integrated resource management	Aside from firewalls and other security measures, networks should have a flexible infrastructure that can accommodate dynamic additions such as load workloads and resource supervision.
Better results	A structure for controlling traffic technological solutions that incorporates capacity calculations and load-balancing as well as a greater degree of consumption in order to lessen the carbon impact.
NV	Flexibility in providing network resources, like switches and routers, without having to worry about their physical location
Visibility and real-time monitoring	Enhancing device connection and real-time surveillance

The SDN management plane gives a clear overview of the dispersed network, allowing for more effective orchestration and automating of network operations. In contrast to conventional protocols, SDN is able to anticipate extra service needs and assign resources before they are needed. User-defined regulations for individual implementation traffic flows are also provided by SDN-based network programs. An in-depth look of SDN platform architecture is provided here. If we are going to talk about and comprehend the changing network architecture, we need to be more specific about two things. Here, we'll go over two areas of network architecture design that are always evolving: scalability and security. Current problems, such as an ever-increasing number and volume connected devices and data created and the capacity of devices to keep up with the massive amounts might be solved by thinking outside the box of traditional network architecture.

In order to address the aforementioned challenges, the current network design incorporates a number of international agreements, proprietary mechanisms, and algorithms. The challenges are eventually overcome and they perform or operate as expected. They complicate the network's design from a different vantage point, though. A few decades earlier, connectivity was in a comparable predicament, but for a distinct issue. Due to a lack of available IPv4 addresses, the IPv6 protocol was developed. But even if IPv6's addressing method is the most secure and expandable, there are still difficulties in entirely switching over to IPv6 and abandoning IPv4. Over the next decade, the recommended remedy to a problem was not implemented entirely, but at least to a significant extent. When defining new scopes for existing difficulties or concerns in the network design, this experience is taken into account.

The SDN Architecture

Using modularity-based ideas, the fundamental design of SDN is quite comparable to formal computer programming methodologies . There are three main levels in an SDN-based network design: the Application plane (the layer that handles applications), the control plane (the layer that handles the network's underlying hardware), and the data plane (the layer that handles network traffic) (see **Fig. 4**). There are well defined borders between the planes, as well as particular roles and APIs for each aircraft to interact with the other planes. **Table 3** presents the planes of the SDN architecture in **Fig. 4**. **Fig. 3** presents a contrast of the current scattered/decentralized traffic management of individual devices with the centralized SDN architecture. The following are the most important parts of the structure

Table 3. Planes of the SDN architecture

Plane	Details
Data (forwarding) plane	When referring to a network's data plane, we mean everything from switches to routers to virtual networking gear to firewalls and so on. Using a set of forwarding regulations dictated by the management plane, the data plane's main goal is to expeditiously transmit internet traffic. As a result of SDN's architecture, forwarding knowledge and settings for each network device are moved to the control plane, rendering the systems and networks highly inoculative. Using APIs, data and management planes communicate (southbound). Currently, the OpenFlow protocol is the preferred southbound communication system supported by a number of companies and the Open Networking Foundation.
Control plane	End-user application constraints dictate how traffic should be routed across the networks, and it is up to the control plane to translate those needs into actual network rules that can be sent to the data plane. Control planes are built on SDN controllers, which have at their core a central control element. Data plane elements receive forwarding principles from an SDN controller and use them to meet their respective goals, like traffic prioritization, access management, bandwidth control, and quality of service. On larger networks, several SDN controllers are possible to provide extra redundancy. Introducing a network fully programmable via the control plane enables real-time manipulation of flow tables in specific components depending on network performances and service needs. Using the controller, network administrators have access to detailed information on the network's inner workings, making it easier to optimize the system's performance.
Application plane	The applications plane consists of both network-specific and business-related software. Using northbound APIs, applications are given with an abstract picture of the network's structure. Apps may get a more complete view of the networking latency, reliability and bandwidth data at a high abstraction level. In response to these requests, the SDN controller preconfigure certain network resources in the control plane.

Centralized administration of network components offers administrators more power, allowing them to change service quality and tailor network architecture as required, based on current network circumstances (see **Fig. 3**). Load-balancing operations like streaming movies and big file transfers across dedicated channels may be used during times of heavy network traffic. Service providers such as VoIP may take over the network during emergencies (such as fire alarms or building evacuations), allowing telephony to take priority over all other network activities

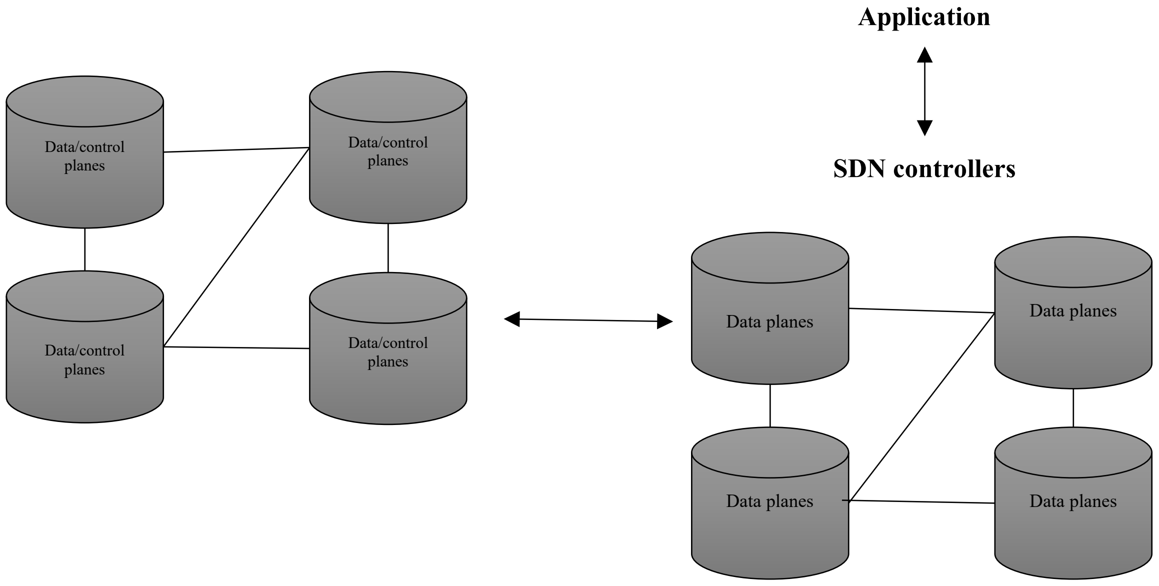

Fig 3. Comparison between decentralized and centralized network control

Network programmability is presented as a solution to the current network architecture's difficulties. NV enables resource management and security concerns to be addressed via the use of software that can be applied to the network as a whole. Before this, network architectures permitted a limited amount of programmability and virtualization. There is no distinction if they are presently on the market since they are already accessible, but they were not customisable. This comment dispels a frequent misconception that computing and virtualization are novel ideas. The structures, interfaces, support for many programming languages and scripting, online coding tools, and implementation program interfaces all play a role in internet programmability. Assisting at differing stages of the network, they make it easier for linked devices to communicate with one other via a more straightforward network architecture.

In network infrastructures, technologies like OpenFlow act as a go-between between the algorithms and the advancing devices. There are a number of different languages and scripts that may be used when programming for the network in the same way that the protocols do. JAVA API, a variety of library features, and the REST applications program interface round out the network programming resources. These entities provide a robust foundation for network programming, allowing it to subjugate the conventional network design.

One of the most intriguing aspects of software-defined connectivity is its potential to provide an entirely new face to a network domain. Many efforts have been made over the last 30 years to overcome the difficulties and complexities of conventional network architecture designs via the use of software-defined networking. There are several initiatives like GeoPlex, an AT&T venture and Supranet Transactions Server from Ericson in the early 2000s that show the traces and trails. Using software-defined connectivity, a network can be made to run more effectively with software applications. This is an improvement over traditional network programming. Here, we purposefully use the word "effectively" because of how software-defined connectivity works. The control and data planes are separated in traditional network architecture in favor of a more flexible approach based on software [21-23].

Open network framework has assisted the phrase software-defined connectivity since 2011, despite previous attempts to coin a name for it. Founded by more than 200 companies from all over the world, the ONF is a global consortium dedicated to standardization and the improvement of the new approach. OpenDayLight, a similar organization to ONF, aims to ensure that various industry specifications are met. As far as protocols go, ONF's OpenFlow standard is a good one for establishing data-to-control plane communication. Another cloud computing software platform, OpenStack, is designed to provide development facilities as and when they are needed.

The design separates the forwarded gadgets from the controlling gadgets, allowing for an eagle-eye viewpoint and management over the system. The physiological gadgets are disconnected from the controllers in the software-defined internet architecture. Switches and other physical appliances in the network serve primarily to simplify the complexities of network resource use and management. It is the controller's job to govern the network, as its name implies, by delivering commands to all the forwarded devices based on an upgraded topological view and the instructions that are performed by the programs that are designed for the network sitting above the controller. To begin, the controllers would be responsible for configuring and managing the network. It would also keep tabs on and fix any issues that arise. When the controller and information plane need to communicate, OpenFlow's Southbound API comes in handy.

Customized software elements would be used by the programs that reside just above controllers in the applications plane to automate these controllers functions, such as customizing, managing, and analyzing. Application-controller communication is referred to as northbound API. In contrast, the Northbound API lags behind and is vendor-centric compared to the Southbound API. Software-defined networking and network programming's ability to distinguish themselves from conventional networking have been outlined in this study, which clarifies the two major divergences in today's network design. The security considerations will be covered in detail in the following sections.

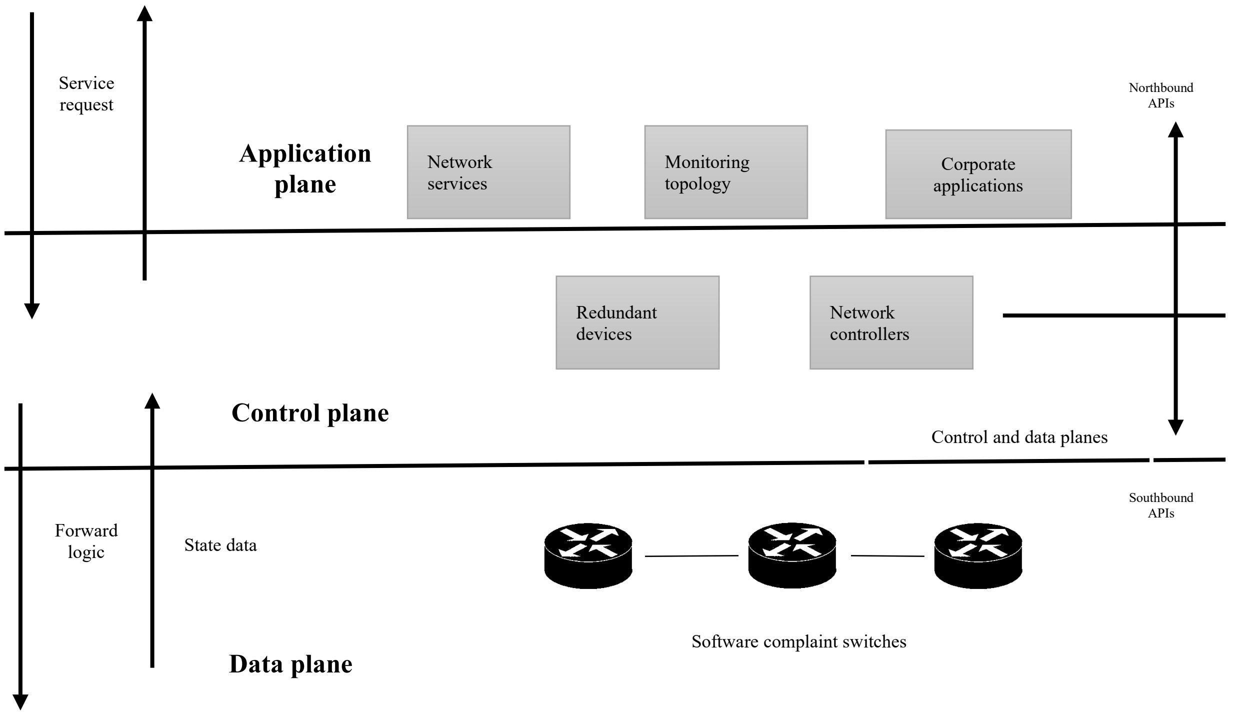

Fig 4. SDN architecture.

Perspective of Security, Behavior, and Lapses

The SDN architecture offers various benefits over the conventional networking infrastructure due to its flexibility in changing network configuration. In the prior section, it was indicated that a decoupled design provides security advantages. To monitor and manage the network's data, the controller enjoys a commanding position thanks to its decoupled architecture. There are various parts of operation of the network that go into managing data flow, such as inspecting packets accessing the system and rebalancing the load inside forwarding devices. It is easier to react to network security problems with SDN design since it has a single point of control. Regardless of the size of the network, the number of data it manages, or any other factor, the necessity of privacy cannot be emphasised. Scalability, accessibility, and resource use of a network may all be evaluated in the same way using a basic set of criteria. If a network is under attack, these indications have little value. As a result, examining the behaviour of the SDN architecture and any security holes becomes even more critical.

In the event of any weaknesses, the centralised control provided by this dynamic architecture may be advantageous. ' In the blink of an eye, a whole network may be brought to its knees by a cyberattack on the system's central controller. SDN architecture's benefits over conventional networking design are rendered insecure in this scenario. If you want a better view into the network's data flow, you must first grasp the infrastructure in terms of security. It is the flow table that is checked for matches when a packet reaches an SDN-enabled forwarding device. As long as a positive match is detected in the flow table, packets will be sent to their final destination. packet-in" message is delivered to control plane in case there is no flow table entry for the packets. Data plane flow tables of the gateways will be updated by the controller depending on the customised programs and protocols it receives. SDN design outperforms traditional network infrastructure for its efficient forward plane data flow.

Data movement choices are made in various places in the SDN architecture shown below. They are divided up into several situations to help identify any security holes or flaws in the design. Entropy-based algorithms, evolutionary computation, and learning algorithms might all be incorporated into the architecture to address the design's security issues. A study by Madureira, Araújo and Sampaio [13] indicated that subweak spots may be identified by diving into the various planes. Gaps in the system might reveal not just the flaws, but also the attacks. The goal of this work is to examine the security elements of SDN architecture and pave the path for their solution, not to delve into depth about the weak spots in each layer. Each level of the hierarchical structure of an SDN architecture makes it subject to attack. Based on an investigation of SDN's growing design, the architecture may be less efficient than it should be. Next, we'll look at the many sorts of attacks that may occur and how to avoid them. This will help us better understand how to improve the network's design and prevent future assaults.

IV. ANALYSIS OF POSSIBILITIES OF ATTACKS

The present state of the growing network architecture is depicted in numerous extant works of literature. The design of software-defined networking has been carefully analysed in this paper, and weak areas have been discovered at all hierarchy levels of the infrastructure. SDN architecture is not inefficient because it is constantly addressing its weaknesses. Weaknesses and other issues in the network's security have received much of the attention, rather than the network's possible operations and activities. The emerging SDN architecture outperforms the conventional network architectural design in terms of resolving current issues. This network's productivity and security depend on fixing the weaknesses that currently exist in its security.

In the previous part, the detected weaknesses were further investigated to watch out for the possibility of attacks, their nature, and the influence that they may have on the network's efficiency. There is no doubt that the information presented here is crucial for understanding the many sorts of attacks that may be launched against this network and devising mitigation methods for each of them, as indicated at the outset of this report. Pandemic scenarios throughout the world, which are becoming more common in today's world, provide the attackers greater leeway to succeed in their assaults. Until a few years ago, operating from home and utilizing cloud services were not considered part of the infrastructure. Obviously, this necessitates an increase in security and network spending.

When existing network configurations move toward evolving network topology, the chances increase correspondingly regardless of their size, whether it is a corporate system or a data centre. In light of the present state of affairs, this project is making progress in classifying different sorts of assaults aimed at the rapidly changing network architecture. A survey of the available literature on attacks is used to categorise the assaults. As a result of these assessments, we've structured the following categories or branches based on the above-mentioned identified weak areas in our growing architecture. They are organized into six groups: Malicious Applications, Misconfigurations, Data Alterations, Denial of Service (DoS) attacks and Distributed Denial of Services (DDoS) attacks, Data Outflow and Access Problems [14].

Defending against the above-mentioned types of attacks leaves software-defined networking architectures open to a broader range of vulnerabilities than only the ones listed. This section includes a list of other assaults that might lead to procedures such as packet capture and analysis and other attacks, such as session hijacking and API compromises. These include compromising administrative credentials, network misdirection and man-in-the-middle threats. A well-thought-out design is essential to thwart both known and unknown threats. If and when the emerging network architecture does not take countermeasures focusing on the above-mentioned attacks, the systems are very vulnerable and easy to be subjected to these threat vectors. In order to better understand the results of these assaults, they are put over the previously stated weak areas in the design.

Table 4 provides an in-depth look at the overall assessment of the kinds of attacks, their implications, and their outcomes in the SDN architectural design. There are many ways to go about mitigating the various weak points in an SDN architecture now that the weak points have been identified and classified clearly. Different algorithms at varying tiers and for various reasons might be explored to enhance the system’s security. More algorithmic and advanced techniques like virtualization might be regarded to effectively mitigate the weak points given the intensity and context of these diverse attack patterns. The above-mentioned categories in this work will be useful to a wide range of technology researchers looking to demonstrate their expertise in mitigation strategies.

Table 4. Presentation of complementary advancements

Functionalities	Control elements	Complementary advancements
Virtualization	Networking device overlays and virtualization	NVGRE, VXLAN, Cabo, VINI, Tempest
Networking programmability	Configuration APIs, High-level networking abstractions, Low-level networking abstractions	GeoPlex, SNMP, NETCONF, Cisco onePK, I2RS, ALTO, DCAN, XMPP, Active Networks.
Centralized control	Delegated/Centralize control model	G/MPLS, Ethane, 4D, GSMP, OPENSIG, PCE, ForCES, BSD 4.4 Routing Socket, NCP

V. CONCLUSION AND FUTURE RESEARCH

This article concentrates on the architecture of Software Defined Network (SDN) and attack possibilities. Researchers found that SDN infrastructure is subject to a wide range of assaults that are comparable to attacks on traditional networking design. Conventional network architecture is similarly susceptible to attacks on SDN infrastructures. As a result of careful research and analysis, the phrase "exposed to comparable hazards" is correct up to the first point. The changing network architecture study does not have to go back to square one since, despite the fact that both these infrastructures are vulnerable to the same sorts of threat vectors, SDN has always had the upper hand when it comes to mitigating these attacks. SDN's interoperability and programmability cannot exist without its decoupled architecture. If the above-mentioned attacks are not adequately mitigated, the downside of this emerging network architecture will be apparent. Future research should also concentrate on defining how attacks operate in every attack vector and offering a viable mitigation strategy for each of the planes.

References

[1]. A. Gouin, A. Dupas, L. Gifre Renom, A. Benabdallah, F. Boitier, and P. Layec, “Real-time optical transponder prototype with autonegotiation protocol for software defined networks,” J. Opt. Commun. Netw., vol. 13, no. 9, p. 224, 2021.

[2]. H. Chen et al., “Research on dynamic load balancing of data center network based on openflow technology,” in 2020 IEEE International Conference on Power, Intelligent Computing and Systems (ICPICS), 2020.

[3]. E. Kemer and R. Samli, “Performance comparison of scalable rest application programming interfaces in different platforms,” Comput. Stand. Interfaces, vol. 66, no. 103355, p. 103355, 2019.

[4]. T. F. Oliveira, S. Xavier-de-Souza, and L. F. Silveira, “Improving energy efficiency on SDN control-plane using multi-core controllers,” Energies, vol. 14, no. 11, p. 3161, 2021.

[5]. C. Cho, J. Lee, E.-D. Kim, and J.-D. Ryoo, “A sophisticated packet forwarding scheme with deep packet inspection in an OpenFlow switch,” in 2016 International Conference on Software Networking (ICSN), 2016.

[6]. R. Ibrahim and W. Whitt, "Real-Time Delay Estimation Based on Delay History in Many-Server Service Systems with Time-Varying Arrivals", *Production and Operations Management*, vol. 20, no. 5, pp. 654-667, 2010. Doi: 10.1111/j.1937-5956.2010.01196.x.

[7]. P. Giocomazzi, L. Musumeci and G. Verticale, "Transport of TCP/IP traffic over assured forwarding IP-differentiated services", *IEEE Network*, vol. 17, no. 5, pp. 18-28, 2003. Doi: 10.1109/mnet.2003.1233914.

[8]. J. Daly, "Social Science Research at the Defense Advanced Research Projects Agency", *PS*, vol. 13, no. 4, p. 416, 1980. Doi: 10.2307/419045.

[9]. "Automated Map Generation for an Intelligent Routing System", *International Journal of Science and Research (IJSR)*, vol. 4, no. 11, pp. 1118-1121, 2015. Doi: 10.21275/v4i11.nov151380.

[10]. H. Alshaer, "An overview of network virtualization and cloud network as a service", *International Journal of Network Management*, vol. 25, no. 1, pp. 1-30, 2014. Doi: 10.1002/nem.1882.

[11]. J. McCoy and D. Rawat, "Software-Defined Networking for Unmanned Aerial Vehicular Networking and Security: A Survey", *Electronics*, vol. 8, no. 12, p. 1468, 2019. Doi: 10.3390/electronics8121468.

[12]. L. Herrera and O. Maennel, "A comprehensive instrument for identifying critical information infrastructure services", *International Journal of Critical Infrastructure Protection*, vol. 25, pp. 50-61, 2019. Doi: 10.1016/j.ijcip.2019.02.001.

[13]. A. Madureira, F. Araújo and L. Sampaio, "On supporting IoT data aggregation through programmable data planes", *Computer Networks*, vol. 177, p. 107330, 2020. Doi: 10.1016/j.comnet.2020.107330.

[14]. B. Maati and D. Saidouni, "CIoTAS protocol: CloudIoT available services protocol through autonomic computing against distributed denial of services attacks", *Journal of Ambient Intelligence and Humanized Computing*, 2020. Doi: 10.1007/s12652-020-02556-0.

[15]. Haldorai, A. Ramu, and S. Murugan, “Signal Processing Architectures, Algorithms, and Human–Machine Interactions in Urban Applications,” Computing and Communication Systems in Urban Development, pp. 49–67, 2019. doi:10.1007/978-3-030-26013-2_3

[16]. Haldorai, A. Ramu, and S. Murugan, “Artificial Intelligence and Machine Learning for Future Urban Development,” Computing and Communication Systems in Urban Development, pp. 91–113, 2019. doi:10.1007/978-3-030-26013-2_5

[17]. Haldorai, A. Ramu, and S. Murugan, “Energy Efficient Network Selection for Urban Cognitive Spectrum Handovers,” Computing and Communication Systems in Urban Development, pp. 115–139, 2019. doi:10.1007/978-3-030-26013-2_6

[18]. Haldorai, A. Ramu, and S. Murugan, “Social Relationship Ranking on the Smart Internet,” Computing and Communication Systems in Urban Development, pp. 141–159, 2019. doi:10.1007/978-3-030-26013-2_7

[19]. Haldorai, A. Ramu, and S. Murugan, “Cognitive Radio Communication and Applications for Urban Spaces,” Computing and Communication Systems in Urban Development, pp. 161–183, 2019. doi:10.1007/978-3-030-26013-2_8

[20]. Haldorai, A., Ramu, A., & Murugan, S. (2019). Machine Learning and Big Data for Smart Generation. Computing and Communication Systems in Urban Development, 185–203. doi:10.1007/978-3-030-26013-2_9.

[21]. Haldorai, A. Ramu, and S. Murugan, “Smart Sensor Networking and Green Technologies in Urban Areas,” Computing and Communication Systems in Urban Development, pp. 205–224, 2019. doi:10.1007/978-3-030-26013-2_10

[22]. G. Gokilakrishnan, S. Ganeshkumar, H. Anandakumar and M. Vigneshkumar, "A Critical Review of Production Distribution Planning Models," 2021 7th International Conference on Advanced Computing and Communication Systems (ICACCS), 2021, pp. 2047-2051, doi: 10.1109/ICACCS51430.2021.9441879.

[23]. S. Murugan and A. Haldorai, “Role of Machine Intelligence and Big Data in Remote Sensing,” Advances in Data Mining and Database Management, pp. 118–130, 2019.