Received: 7 March 2022 | Revised: 27 May 2022 | Accepted: 15 June 2022
DOI: 10.1002/dac.5296

RESEARCH ARTICLE

WILEY

A comprehensive survey on software-defined networking for smart communities

Rajat Chaudhary[1] | Gagangeet Singh Aujla[2] | Neeraj Kumar[3] | Pushpinder Kaur Chouhan[4]

[1]Department of Mathematics and Computer Science, Karlstad University, Karlstad, Sweden
[2]Department of Computer Science, Durham University, Durham, UK
[3]Thapar Institute of Engineering and Technology, Punjab, India
[4]British Telecom, London, UK

Correspondence
Gagangeet Singh Aujla, Department of Computer Science, Durham University, Durham, DH1 3LE, UK.
Email: gagangeet.s.aujla@durham.ac.uk

Summary
The need to provide services closer to the end-user proximity leads to the exchange of a large volume of data generated from the smart devices deployed at different geo-distributed sites. The massive amount of data generated from the smart devices need to be transmitted, analyzed, and processed. This requires seamless data exchanges among geo-separated nodes, which results in a considerable burden on the underlying network infrastructure and can degrade the performance of any implemented solution. Therefore, a dynamic, agile, and programmable network management paradigm is required. To handle the challenges mentioned above, software-defined networking (SDN) gained much attention from academia, researchers, and industrial sectors. Shifting the computational load from forwarding devices to a logically centralized controller is a dream of every network operator who wants to have complete control and global visibility of the network. Also, the concept of network functions virtualization (NFV) in SDN controller is required to increase resource utilization efficiency. Thus, in this paper, a comprehensive survey on SDN for various smart applications is presented. This survey covers the infrastructural details of SDN hardware and OpenFlow switches, controllers, simulation tools, programming languages, open issues, and challenges in SDN implementation with advanced technologies such as 5G and microservices. In addition, the challenges on the control plane and data plane are highlighted in detail, such as fault tolerance, routing, scheduling of flows, and energy consumption on OpenFlow switches. Finally, various open issues and challenges future scope of SDN are discussed and analyzed in the proposal.

KEYWORDS
data center networks, edge computing, fog computing, healthcare ecosystem, internet of things, network functions virtualization, software-defined networking

1 | INTRODUCTION

Smart cities have become the vision of various organizations across the globe for providing uninterrupted services and experiences to end-users. From smartphones to bursty traffic, from social media to smart utilities, and from e-healthcare to smart infrastructure, smart technologies, including wireless sensor networks, smart grid, smart transportation, and e-healthcare, have completely engulfed the geographic ecosystem. Multimodal sensors, multidimensional data sources, powerful gadgets, and smart devices act like standalone machines to provision complicated services to a wide range of smart applications to provide Quality of Experience (QoE) and ambient living to the citizens of a smart ecosystem. Moreover, the evolution of the Internet of Things (IoT) and tactile internet have revolutionized industrial organizations through machine-to-machine and device-to-device communication. A modern set of standards and dynamic technologies have envisioned new horizons with data locality using edge/fog computing, a cloud of things, mist computing, and osmotic computing. Smart communities (e.g., intelligent transportation, smart healthcare, and smart utilities) are expanding exponentially in vertical and horizontal directions. The enterprise model of Everything-as-a-Service (EaaS) built over the underlying service-oriented architecture of smart communities rely on the five attributes listed by the National Institute of Standards and Technology (NIST). These attributes consist of (i) measured services, (ii) on-demand self-service, (iii) broad network access, (iv) resource pooling, and (v) rapid elasticity.[1] Figure 1 illustrates various service-oriented verticals in a typical smart city environment.

This entire service-oriented ecosystem can be viewed as a cloud of smart things, in which the data collection, analysis, and processing are handled closer to the location of the user. However, the user base of entire smart communities is huge enough to generate millions of zettabytes of data daily.[2] For example, a typical smart city having a population of one million generates data of 200 million GB on a daily basis.[3] According to a survey, report,[4] 204 million emails are exchanged, 5 million searches are made on Google, 1.8 million Facebook users make likes, 350,000 tweets are exchanged, and an amount of $272,000 is spent on Amazon to purchase various amenities, and more than 15,000 tracks are downloaded over *itunes* daily. With billions of devices embedded inside the smart city infrastructure, the entire framework critically depends on efficient data management. The

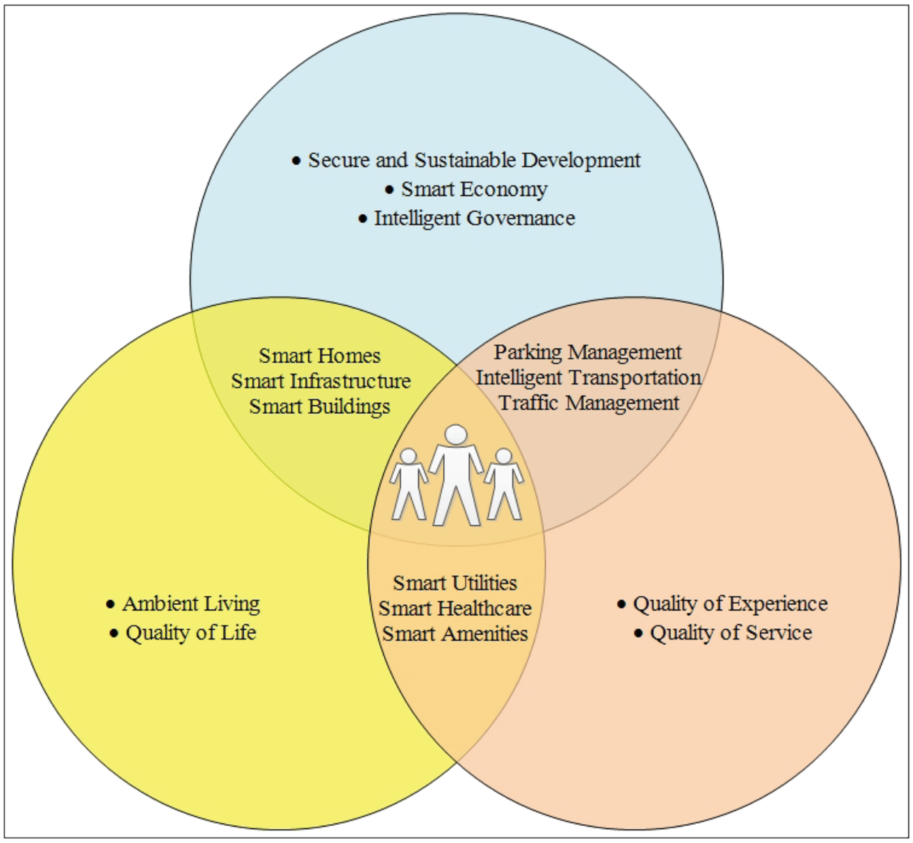


FIGURE 1 Various verticals of smart communities

realization of smart cities relies on the efficient management of the large volume of data generated by various devices and sensors. For this purpose, one of the essential requirements is the seamless transmission of data across various geo-separated nodes.

However, an exponential increase in the ever-growing data traffic progressively erodes the margins of the network operators in terms of bandwidth, availability, and data rates. The tenable vision of having complete control of the network is a long dream of every network operator, irrespective of the network infrastructure vendor. Network devices were considered configurable black boxes until modern technological development enabled them to be programmed and configured according to users' requirements. This holistic visibility and control over the entire network through a logically centralized remote server breaks the vertical integration of traditional networks by software-defined networking (SDN). The provisioning of services to the end-user domain in a smart city scenario embraces various dynamic changes in the topological setup over a potential deployment of heterogeneous devices and sensors. SDN is a suitable candidate for providing agile, flexible, and scalable network management in smart application ecosystems. In the context of a global scenario, a lot of large-scale service providers such as Facebook and Google are working on the trends of softwarized data centers for sustainable development of smart applications.[5]

1.1 | Scope of this survey

Keeping in view the above-discussed points, this survey is focused on the relevance of SDN in various smart applications. Specifically, this survey considers the deployment of SDN in four vertical domains, namely, (1) data center networks (DCN), (2) edge/fog computing, (3) IoT-based applications, and (4) wireless body area networks (WBAN). SDN is a promising technology which addresses the challenges of strong control and data plane coupling in traditional networks. By deploying SDN controllers in various smart applications, it is possible to change the network configurations dynamically. In this direction, various existing proposals have discussed different aspects related to SDN architecture. The potential research surrounding SDN and its prospective deployments were at its zenith during the last 5 years (20014 to 2018). Most of the existing surveys on SDN considered diverse parameters such as software and hardware OpenFlow (OF) switches, controller types, programming languages, emulation and simulation tools, flow table management, security, energy management, fault tolerance, network virtualization, and hypervisors' software. For example, authors in an earlier study[6] discussed various aspects related to the evolution of programmable networks. Similarly, in a previous work,[7] the authors investigated various aspects related to SDN from implementation points of view. Lara et al[8] discussed different aspects related to the OF paradigm in networking. Akyildiz et al[9] investigated the role of SDN and OF in traffic engineering. In earlier studies,[10,11] the authors discussed various aspects related to SDN in a comprehensive manner. In another research,[12] the authors investigated various network virtualization hypervisors in SDN environments. In Mijumbi et al,[13] the focus remained on the concepts of network function virtualization. Moving toward a different perspective, Trois et al[14] surveyed various programming languages and their perspective impact on SDN. In another related work, Nguyen et al[15] investigated various rule placement problems in OF networks. In an Guck et al,[16] different unicast routing algorithms were discussed with respect to quality and performance parameters. In addition, Alvizu et al[17] discussed the role of SDN in transport networks. In Huang et al,[18] various approaches and challenges concerning different testbeds for large-scale SDN were analyzed. Baktir et al[19] discussed the potential benefits of SDN in edge computing. A major discussion in previous studies[20,21] focused on the security challenges and possible solutions in SDN. In an earlier work,[22] the authors presented various SDN architectures along with a discussion on security and energy efficiency. Khan et al[23] highlighted various threats related to the topology discovery in SDN. After analysis of all the above-discussed proposals, a comparative analysis is provided in Table 1. It is pertinent to mention that none of the above-discussed proposals highlighted any aspect related to the potential deployment of SDN and OF concepts for smart applications. Specifically, neither of the existing proposals has discussed the deployment of SDN and its related concepts in DCN, edge and fog computing, IoT, and healthcare ecosystems. To the best of our knowledge, this is the first survey of its kind, which starts from the historical evolution of programmable networks then move on to the architectural details and OF infrastructural components and finally discusses different deployment models of SDN in smart applications (DCN, edge/fog computing, IoT, and WBAN).

TABLE 1 Comparative analysis of existing surveys of SDN with the proposed survey

Authors	Year	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Nunes et al.[6]	2014	✓	✓	✓	×	✓	×	×	×	×	×	✓	×	×	×	✓
Hu et al.[7]	2014	×	✓	✓	✓	×	✓	×	✓	✓	✓	✓	×	✓	×	✓
Lara et al.[8]	2014	✓	✓	✓	×	✓	✓	×	✓	✓	×	✓	×	×	×	✓
Akyildiz et al.[9]	2014	×	✓	✓	×	×	×	×	✓	✓	×	✓	×	×	×	✓
Kreutz et al.[10]	2015	✓	✓	✓	✓	✓	✓	×	✓	✓	✓	✓	×	×	×	✓
Xia et al.[11]	2015	✓	✓	✓	✓	×	×	×	×	✓	×	✓	×	×	×	✓
Blenk et al.[12]	2015	×	✓	✓	✓	×	✓	×	✓	✓	✓	×	×	×	×	×
Mijumbi et al.[13]	2016	×	×	×	×	×	✓	✓	×	✓	×	✓	×	×	×	✓
Trois et al.[14]	2016	×	✓	✓	✓	×	✓	×	×	✓	×	×	×	×	×	✓
Nguyen et al.[15]	2016	✓	×	✓	×	×	✓	✓	✓	✓	×	✓	×	×	×	✓
Guck et al.[16]	2018	×	×	×	×	×	×	×	×	×	×	✓	×	×	×	✓
Alvizu et al.[17]	2017	×	×	✓	✓	×	✓	×	×	✓	✓	✓	×	×	×	✓
Huang et al.[18]	2017	✓	×	✓	×	✓	✓	×	×	✓	✓	✓	×	×	×	✓
Baktir et al.[19]	2017	✓	×	✓	×	×	✓	✓	×	✓	×	✓	✓	✓	✓	✓
Yan et al.[20]	2016	×	×	✓	×	×	✓	×	✓	✓	×	✓	×	×	×	✓
Hayward et al.[21]	2016	×	×	✓	×	×	✓	×	✓	✓	✓	✓	×	×	×	✓
Rawat et al.[22]	2017	×	×	✓	✓	×	✓	✓	✓	✓	×	✓	×	✓	×	✓
Khan et al.[23]	2017	×	×	✓	×	×	✓	×	×	✓	✓	✓	×	×	×	✓
Molina et al.[24]	2018	✓	✓	✓	×	×	✓	×	✓	×	×	✓	×	✓	×	✓
Zhao et al.[25]	2019	×	✓	✓	×	×	✓	×	✓	✓	×	✓	✓	✓	×	✓
Rafique et al.[26]	2020	×	✓	✓	×	×	✓	×	✓	✓	✓	✓	✓	✓	×	✓
Haji et al.[27]	2021	✓	✓	✓	✓	✓	✓	×	✓	✓	✓	✓	✓	✓	×	✓
Pusuluri et al.[28]	2021	×	✓	✓	×	×	✓	✓	✓	✓	×	×	✓	✓	✓	✓
Bhardwaj et al.[29]	2022	✓	✓	✓	×	×	×	×	✓	✓	×	×	✓	✓	×	✓
Ma et al.[30]	2022	×	×	✓	×	×	✓	×	×	✓	×	✓	✓	✓	×	✓
Proposed Survey	-	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

Note: 1: SDN Hardware & Software Switches; 2: SDN Controllers; 3: OF-Switch Flow Entries Management; 4: Programming Languages; 5: Simulation Tools; 6: Security; 7: Energy-efficiency; 8: Fault Tolerance; 9: NFV; 10: Hypervisor Softwares; 11: DCN; 12: Fog Computing; 13: IoT/WSN; 14: Healthcare Network/WBAN; 15: Open Issues & Challenges; ✓: considered; ×: not-considered.
Abbreviation: SDN, software-defined networking.

1.2 | Contributions of this survey

This article presents a comprehensive survey on SDN deployment in smart applications. In the first part of the survey, the background and history of the evolution of SDN are discussed in Section II. Then in Section III, SDN architecture and OF infrastructure components are discussed, wherein various OF components and switches are classified concerning different parameters. In Section IV, various deployment models for SDN are discussed with respect to four application domains, namely; 1) DCN, 2) fog computing, 3) IoT-based applications, and 4) healthcare network. The open issues and research challenges are discussed in Section V. Finally, the article is concluded in Section VI. Figure 2 depicts the different sections spanning the article.

2 | BACKGROUND AND HISTORY OF SDN

This section presents the background and history of evolution of SDN. Moreover, the comparison between SDN technology with traditional networks is also presented.

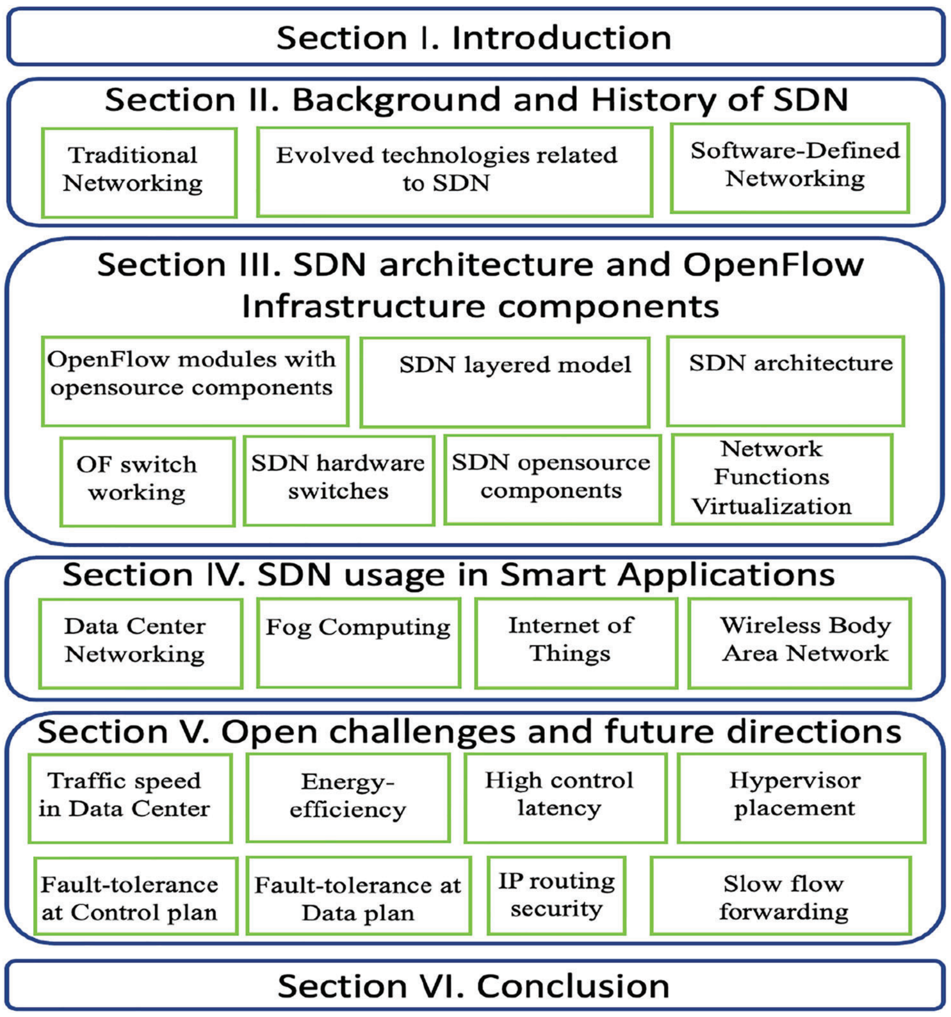

Section I. Introduction
Section II. Background and History of SDN
Traditional Networking
Evolved technologies related to SDN
Software-Defined Networking
Section III. SDN architecture and OpenFlow Infrastructure components
OpenFlow modules with opensource components
SDN layered model
SDN architecture
OF switch working
SDN hardware switches
SDN opensource components
Network Functions Virtualization
Section IV. SDN usage in Smart Applications
Data Center Networking
Fog Computing
Internet of Things
Wireless Body Area Network
Section V. Open challenges and future directions
Traffic speed in Data Center
Energy-efficiency
High control latency
Hypervisor placement
Fault-tolerance at Control plan
Fault-tolerance at Data plan
IP routing security
Slow flow forwarding
Section VI. Conclusion

FIGURE 2 Overview of the proposed survey

2.1 | Traditional IP networks and evolution of SDN

The traditional networking based on TCP/IP model has limitations such as–tightly coupled planes, distributed architecture, manual configuration, inconsistent network policies, error-prone, security, and inability to scale.[31] The traditional networks are based on the client-server model, where the client sends the request for services on web applications and the server replies as a response for the services, which is a distributed mode of communication. In traditional networking, the control and the packet forwarding parts are integrated and embedded in the hardware devices. The strongly coupled nature makes it difficult to modify the network policies. Moreover, the distributed architecture is vulnerable to various types of security threats. So, it becomes extremely complicated to troubleshoot the network. Apart from this, one of the most important aspects is the difficulty of maintaining consistency while modifying the network policies. With a substantial increase in the workloads in the IT infrastructure, the demand for increased bandwidth is also expected. The traditional networking domain is statically arranged in such a manner that the growing bandwidth demands of the IT infrastructure end up in the redesigning of the complete topology. To overcome these issues of traditional networks, a prominent technology named SDN, which works on the ideas of OF architecture, is being widely deployed in different network domains. SDN was first introduced in 2006 at Stanford University in the clean slate research project.[6,10] SDN is a centralized model that receives all the requests from the clients located at the data plane connected with OF switches. The controller is the decision-maker completely isolated from the forwarding plane that monitors all the network activities globally and deploys the optimal solution on the OF switches. The SDN architecture helps to improve network scalability, flexibility and cost-cutting infrastructure by deploying services like dockers, containers, microservices and serverless computing instead of using a physical machine (PM) and virtual machines

(VMs).[32–34] Figure 3 shows the architecture of the traditional networking (TCP/IP) protocol suite based on the client-server model and modern SDN-based programmable networks.

SDN is a logically centralized software capable of controlling the entire network. It advocates the concept of software-based networking and can be termed as a "softwarization" solution that provides network programmability by removing the network intelligence from the forwarding devices. It solves the issues of the traditional network architecture to accommodate the increased workloads of modern networking. SDN architecture provides better services to manage enterprise, wide area, and DCNs. The control or network brain and the data forwarding plane are decoupled in SDN architecture. So the forwarding nodes become control free and perform only packet forwarding functionality. Rather than the manual configuration of all the network devices by an administrator, SDN allows program-based software control to automate the tasks and increases the flexibility in modifying the network policies dynamically. Another feature of SDN architecture is its centralized architecture which helps to assign, modify, and audit policies with a global network view. These centralized policies are used for routing, security, maintaining consistency among physical and virtual workloads, load balancing, and global monitoring of the entire network.[6,35,36]

2.2 | History of the evolution of SDN

In this section, the history of the evolution of SDN is summarized from its inception till today. Moreover, the yearly growth trends of various programmable network domains are also highlighted in Figure 4. The various evolved technologies related to SDN are described below.

- **Active networking**: Active networking[10] was an initiative to promote software-based programmable networks. US Defense Advanced Research Projects Agency (DARPA) introduced the active networking project in the early 1990s. The routing in active networking is based on two models: (a) programmable switches/routers and (b) capsules.[6] In the former model (programmable switches/router), the instruction sets are simply downloaded and sent from the control logic. The flow rules for packet forwarding are processed without any modification in the packet format. The programming model uses an out-of-band signaling concept (use different bands or dedicated bands) to execute the

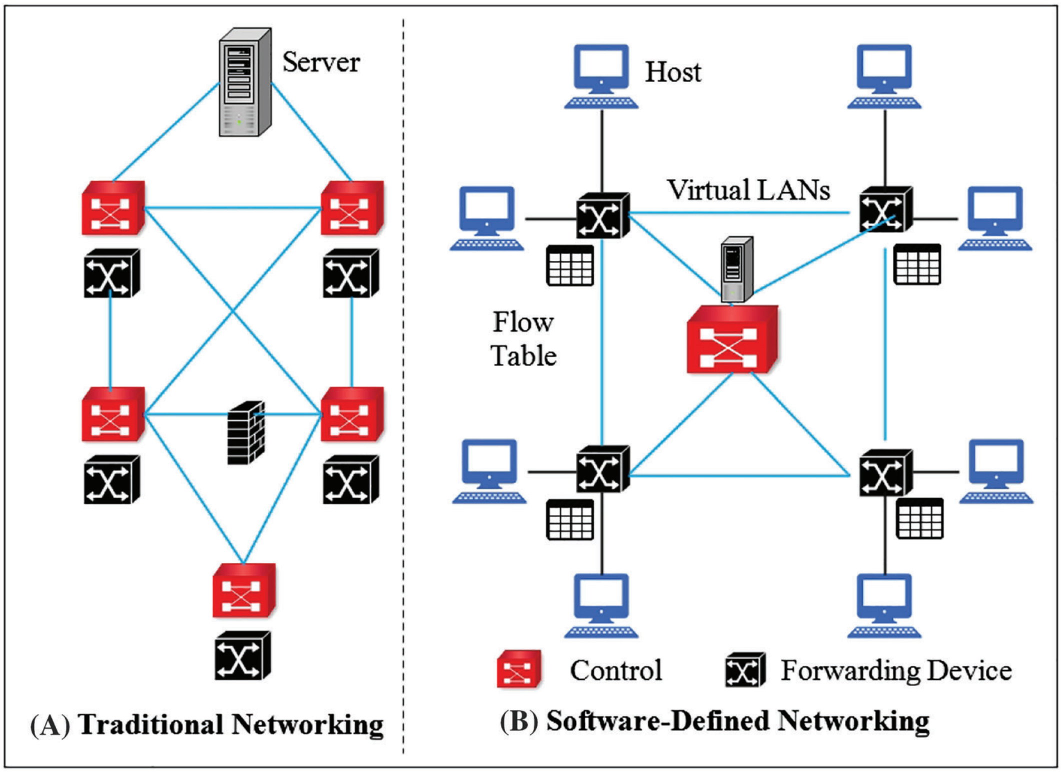


FIGURE 3 Architecture of traditional networking and software-defined networking (SDN) programmable networks

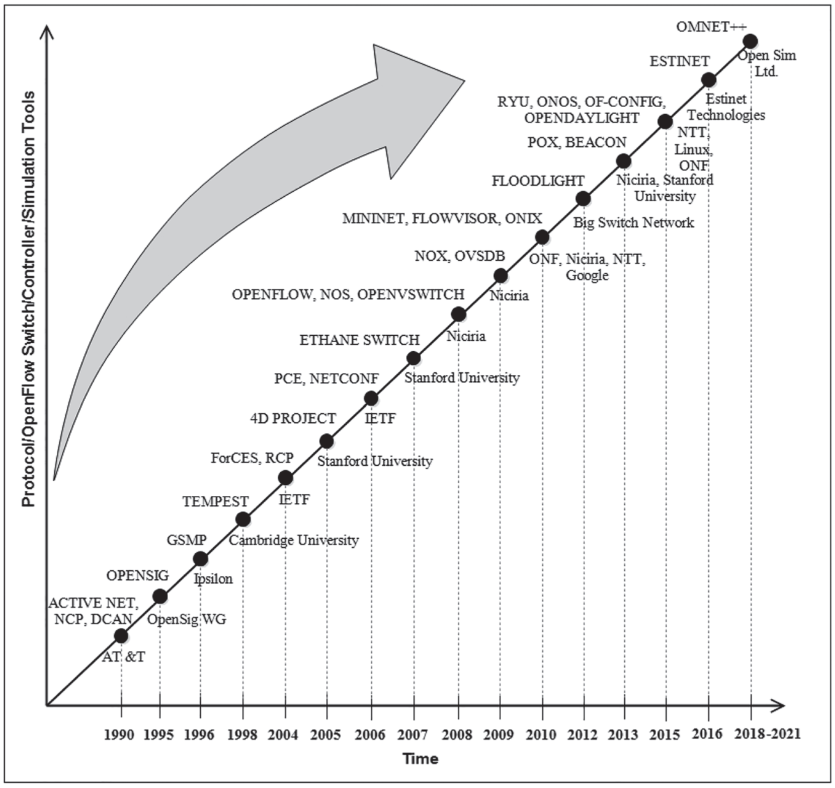


FIGURE 4 Evolution of software-defined networking (SDN)

control information at the network devices for managing channels. In the later model (capsule), the control information that needs to be executed is encapsulated in program fragments. The tiny programs are enveloped in the messages and are executed through the forwarding devices. Among these models, the latter became popular as it significantly impacts the management of network paths by installing a new functionality in the data plane. An example of active networking to create the software routers are SwitchWare,[37,38] Click,[39] XORP,[40] Quagga,[41] and Bird.[42]

- **Network control point (NCP)**: In the mid-1990s, when the Internet was becoming popular, an initiative was taken by AT&T telecommunication company with the inception of a project named NCP.[10] This project targeted the decoupling of the control signaling of the telephone lines from the data plane. The control logic is completely isolated from the forwarding nodes to yield maximum scalability and flexibility in device management. In this way, NCP improved the overall channel efficiency of telephone signaling.
- **Devolved control of ATM networks (DCAN)**: A project DCAN[6] started in the mid-1990s focused on the scalable operations of the ATM networks. This project aimed the design of a protocol that split up the control function of the physical ATM switches from the application feature. An external dedicated workstation entity handled this split-up control function. The external entity manages the resource allocation to the ATM switches and provides the improved QoS.
- **OpenSig**: In 1995, an open signaling movement[6,10] was introduced by OpenSig WG. The movement started workshops on open signaling to analyze the challenges faced by combining the control and the forwarding nodes. An open signaling project promoted the wider adoption of open, extensible, and programmable interfaces in the ATM networks and the Internet. The idea of this proposal was to develop a programmable intellectual network, which can be possible only by the separation of the switch controller and the switch fabric. The signaling between both planes was performed using the switch independent interfaces. The open signaling allows both levels to work independently. The open interfaces support interoperability and innovation in the design of the hardware equipment.
- **General switch management protocol (GSMP)**: The Ipsilon networks designed an asymmetric protocol in 1996 for WAN named GSMPv1.[6] Later on, GSMPv2 and GSMPv3 were released in 1998 and 2003, respectively. These protocols were designed to control and manage ATM switches, frame relay, or multiprotocol label switching using an external controller. Various tasks performed by the GSMP protocols include monitoring and reporting statistics, supervising network switch ports, record configuration information, inserting or updating switches on multipoint

connections, and creating and exempting switch connections via a controller. The GSMP protocol enabled high performance in ATM networks and was responsible for granting permission to regulate both unicast and multicast switch connections.

- **Tempest**: Based on the motivation of programmable networks in ATM switches, virtualization was a new concept which laid its foundation in the early 1990s. Cambridge University started the earliest initiatives on network virtualization in switches. In context with ATM networks, the tempest[43] framework was the first project started in 1998, which supports switch virtualization. The virtualization in the tempest framework helped to share single physical network resources by multiple independent switches. The multiple switches are created virtually, residing on a single physical ATM switch. The multiple switches and the virtual network are substitutes for the control architecture of an operational ATM network.
- **Forwarding and control element separation (ForCES)**: The Internet engineering task force (IETF) designed ForCES protocol[6,10] in 2004. This protocol consolidated two elements, that is, the forwarding element (FE) and the control element (CE). The message sharing between the CEs and the FEs was based on the master–slave architecture via the ForCES protocol. The CEs are the masters, and the FEs are the slaves. The function of the FEs is the flow forwarding of every packet. The CEs are the logic function blocks (LFBs) responsible for creating the control logic and installing the control information by signaling. They also monitor how the FEs process each packet.
- **Routing control platform (RCP)**: In Feamster et al,[44] an RCP protocol was designed in 2004 to reduce the complexity of the border gateway protocol (BGP). The IP routers in BGP perform interdomain routing in an autonomous system. The fully distributed path selection computation is done in each domain. RCP overcomes this limitation by separating the functionality of interdomain routing from IP routers. Thus, the computation of the best path selection for each router in every domain was performed at the control plane. Finally, the result is intimated at the data plane for the best path in every domain, and the routing information of RCP is exchanged with other domains also.
- **4D project**: In 2005, Stanford University proposed a 4D approach[45] for network control, which was based on a bottom-up layered architecture. The lowest layer was the data plane, the middle layer comprised discovery, and dissemination planes, and the topmost layer was the decision plane.[46] The decision plane was the management layer which works to create logic for routing, load balancing, security, interface configuration, and access control. It can monitor the global topological view, and accordingly, the logic is elevated to the forwarding nodes. The discovery layer performed functions such as physical device discovery, building their logical entities, the association between the physical and logical entities, the capacity of the interface which is suitable for a particular switch/router, and discovery of the capacity of the switches to hold the flow table entries. The dissemination layer establishes the secure connection between the decision and data planes and delivers timely and reliable control information. The data layer performs functions such as packet filtering, address translation mapping, next-hop forwarding, queue management, and tunneling.
- **PCE and NETCONF**: In 2006, IETF standardized two communication protocols: (1) path computation element (PCE)[47] and (2) network configuration (NETCONF) to determine the network path based on a network graph. PCP is the request/response communication protocol based on a client–server model. The server computes an optimal route based on on-demand requests from clients. It is suitable for ATM, Ethernet, MPLS, and BGP computations, wherein high bandwidth services are essential. On the other hand, NETCONF deals with the network configuration and sends a notification whenever any change in the configuration occurs. NETCONF protocol[10] uses XML-based data encoding, which is designed to install, update, and delete the network configurations as per the dynamic network nodes.
- **ETHANE network**: A network architecture called Ethane network was designed by Stanford University[48] in 2007. The communication in this network between hosts *X* and *Y* is based on five activities. These activities are switches and port pairs registration, switches bootstrapping for secure channel connectivity, host authentication, flow setup, and forwarding. The Ethane network architecture was designed in such a way that the communication between end hosts cannot exchange messages directly, but they have to follow all the activities mentioned above.
- **OF switches**: There are massive improvements performed in OF switches of multiple vendors like CISCO, Juniper, IBM, HP, and NEC that support different operating systems like NOS, JUNOS, but the best part is that the OF switches are programmable that are open source and are not vendor-specific.
- **SDN controllers**: The list of SDN controllers (POX, NOX, opendaylight, floodlight, ONOS, Ryu, Hyperflow) with features specifications are shown in Table 2.
- **Simulation tools**: The network simulators are useful for performing the network simulations. There are a list of simulation tools such as mininet, NS-2, Estinet, OMNET++, and time analysis which are helpful for the SDN

TABLE 2 List of SDN hardware switches and open source components

Type	Name	Developer	Language	OS	Model
OF-Hardware Switches	2960x, 3000x	Cisco	-	Cisco IOS, NOS	-
	EX9200	Juniper	-	JUNOS, NOS	-
	G8264	IBM	-	z/OS, NOS	-
	5400 zl, 8200 zl	HP	-	OS independent	-
	PF5820	NEC	-	OS independent	-
	MLX Series	Brocade	-	OS independent	-
	3920	Pica8	-	OS independent	-
	7150 series	Arista Networks	-	OS independent	-
	X8	Extreme Networks	-	OS independent	-
OF-Software Switches	9000x, 3000x	Cisco	Python	Cisco NX-OS, Linux	Both
	Contrail-vrouter	Juniper	Java, Python	XtreemOS, Linux	Both
	Switchlight	Big Switch	Java	switch light OS, Linux	Centralized
	OpenWRT	Stanford University	Java, Python	Linux	Both
	OFsoftswitch	Ericson	C, C++	Linux	Both
	XorPlus	Pica8	C, C++	Linux	Both
	OpenvSwitch	ONF	C, Python	Linux	Both
Logical, Physical Interfaces (switch ports)	L2 (physical), virtual LANs (logical)	ONF (Open Networking Foundation)	C++, Java, Python	Linux, Mac, Windows	Both
Controllers	Opendaylight (ODL)	Linux Foundation	Java	Linux	Distributed
	Floodlight	Big Switch	Java	Linux, Mac, Windows	Centralized (multi-threaded)
	POX	Niciria	Python	Linux	Centralized
	NOX	Niciria	C++, Python	Linux	Centralized
	Beacon	Stanford University	Java	Linux, Mac, Windows	Centralized (multi-threaded)
	Ryu	NTT	Python	Linux	Centralized (multi-threaded)
	Hyperflow	(virtual)	Java	Linux	Distributed
	Kandoo	(virtual)	C++, Python	Linux	Distributed
	Disco	(virtual)	Java	Linux, Mac, OS	Distributed
	Smartlight	(virtual)	Java	Linux	Distributed
Simulation Tools	Mininet	Stanford University	Python	Linux	Centralized
	Estinet	Estinet Technologies	C, C++	Linux, Windows	Centralized
	NS-2	LBNL	C++, Python	Linux	Distributed
	CloudSimSDN	University of Melbourne	Java	Linux	Distributed
	OMNET++	Open Sim Ltd.	C++	Linux, Mac, Windows	Distributed

TABLE 2 (Continued)

Interfaces	Features and config commands
OF, OVSDB, BGP	Max. Capability- $16K^*$ MAC addresses, 16 static IPv4 and IPv6 unicast routes, $1K^*$ active VLANs
OF, VxLAN, OVSDB, LLDP	Max. Capacity: 64 GB DRAM, 64 GB SSD, 512000 forwarding entries, 32000 VLANs, 256000 IPv4 and Ipv6 multicast routes, bandwidth 480 Gbps/slot
BGP, OF	Max. Capacity: 48 Gb ethernet ports for bandwidth, 128000 MAC addresses, 960 million packets per second (Mpps) with latency 880 ns, 126 interfaces/switch, 4094 VLANs
OF, Rest APIs	Max. Capacity: 10000 flow entries, MAC address of 64000 entries, throughput 564 pps
Rest and Nova API	Max.- 160000 flow entries, 48 ports, 4094 VLANs
OF, OVSDB	Max.- 128000 flow entries, 9.5 billion pps routing, 4094 VLANs
OF, OSPFv3	Max.- 4094 VLANs, forwarding speed 960 Mpps, latency 900 ns
OF, OVSDB	Latency 350 ns, routing 960 Mpps, 4096 VLANs
OF, BGP	Latency 2.3 microsecond, 11.4 billion pps, 4094 VLANs
OF, REST	Enable openflow agent, configure- physical devices and interfaces
REST API	Enables routing and switching, load balancing, security and API services
OF, Rest APIs	Thin switching software for physical and virtual switches
OF	Configure wireless router to an OF-enabled switch
OF, Rest APIs	Provides secure channel to connect switch to controller
OF	It is licensed free OF-software switch to support switching capabilities at layer 2 and 3
OVSDB, OF	Virtual software switch and ported to multiple hardware platforms
OF, NetConf, OVSDB, PCEP	Msg. (C $\rightarrow$ S): read, modify, packet-out, handshake, switch configuration. Sync Msg.: hello, echo request/reply. Async. Msg.: packet-in, port status
Rest, OSGi	Performs topology discovery, shortest path forwarding
Rest APIs	Performs behavior modification, routing, forked from the Beacon controller
Adhoc API, OpenFlow, OVSDB	Performs path selection, load balancing, queryable topology graph and support for virtualization.
Adhoc APIs	It is the first OF controller. Tasks- topology management, routing, switching
Adhoc APIs, IBeaconProvider	Performs routing, supports both event based and threaded operation
Rest APIs, RPC	Commands: rest_router.py, rest.firewall, rest_topology.py, rest_conf_switch.py
Rest APIs	Perform event synchronization of multiple controllers, guarantees loop-free forwarding and is resilient to network partitioning.
OF, OVSDB	Creates local controllers for local switches and a centralized root controller for elephant flow detection
Rest APIs	Perform intra-domain and inter-domain functionalities with other controllers
Rest APIs	Enables fault tolerant hierarchal master slaves controller
OF	Free license (CLI) Commands: sudo mn, arp, mac, switch, ovsk, sudo, mn, –test, pingall
OF	Paid license (GUI), Uses tunnel interfaces and sync simulation clock to control the order execution of events
OF	It integrates with OFSwitch, supports built-in to emulate an OF for real-time simulations.
OF	(GUI Support) It simulates DCs, pSwitches and virtual topologies. It performs discrete event-based simulation, VM placement and migration policies.
OF	It uses .ned files to assign addresses to nodes then reads configuration file omnetpp.ini

Abbreviations: OF, OpenFlow; SDN, software-defined networking.

controller, OF switches, and hosts to deploy the network topology and run the simulation results related to bandwidth consumption, memory or CPU utilization, latency computation, jitter and throughput, and so on.

In addition to the above-discussed variants, various other SDN projects investigate different OF modules and components proposed in the global market from 2008 to 2021, which are explained in the subsequent sections.

3 | SDN ARCHITECTURE AND OF INFRASTRUCTURE COMPONENTS

The architectural and controller deployment details related to OF hardware infrastructure is elaborated in this section.

3.1 | OF modules with open source components

OF[7] refers to the open-source communication protocol developed by Stanford University and Niciria in 2008. OF protocol is a flow-oriented protocol that provides switch and port level abstraction to allow message sharing between the network controller and forwarding devices. OF-Switch comprises three modules: (a) multiple flow entries tables and a group table, (b) OF communication channel, and (c) OF protocol. The OF-switches connect to each other by an OF-port so that each OF-switch establishes secure communication with the network controller with OF protocol. The firmware of OF-Switch and OF-vSwitch[7] are Pica8 and Indigo. The kernel of both switches is written in Linux 3.3.

The conventional operating systems (Cisco IOS, Juniper JUNOS, ExtremeXOS, and SR OS) are device-specific and are only compatible with their products. In 2008, the existence of OF protocol[7] in the market envisioned the idea of designing an operating system on the servers which is centralized and can control all the heterogeneous network devices by an OF protocol. Therefore, in 2008, an OF-based network operating system (NOS) (NOX, ONOS, ONIX)[49] was designed to provide a high-level abstraction for network resources. The SDN controller software gets installed on the PC server's NOS to configure the network policies built by an administrator. NOS provides an abstraction of essential services and interfaces to connect with forwarding devices. The southbound interface is the Open Virtual Switch Database (OVSDB) for OF-switches. The OVSDB interface must create multiple virtual OF-switches (OF-switches) as an instance of the physical OF-switch. The virtual switches are used to efficiently utilize network resources (memory, storage, and communication links). In 2015, the open network foundation (ONF) designed another southbound interface named IF-CONFIG.

Table 2 shows a comprehensive summary of all OF the modules comprising OF-hardware switch fabrics, OF-software switches, logical and physical interfaces, SDN controllers, simulation, and emulation tools.

3.2 | SDN bottom-up layered model

The architecture of SDN is based on various layers. The bottom-up layered architecture of SDN is shown in Figure 5. The bottom-up layered SDN model is a three-layered architecture also popularly called networking planes, that is, data, control, and the application or management planes. The lower layer is the data plane, which consists of network forwarding nodes like physical OF-switches, OF-routers, and gateways.[50] Additionally, OF-switches and ports are also defined for multiple hardware platforms. The forwarding nodes perform various operations such as packet header lookup, forwarding, encapsulation, pipeline processing, and data statistics management. The SDN controller communicates with the forwarding nodes via the southbound interface. The southbound interfaces such as OF protocol, OVSDB, protocol-oblivious forwarding (POF), ForCES, NetConf, and OF-Conf are used as communication protocols.[14,51,52]

The middle plane is called the control logic or network brain on which the SDN controller software resides.[53] The SDN controller is installed on the NOS. With the help of hypervisor software, master–slave controller architecture is designed. The network hypervisor creates multiple virtual switches on the data plane controlled by an SDN controller. The virtualization concept helps in failure recovery, wherein complete back-up of the master controller is preserved on the slave controllers. The efficient resource utilization is achieved by multithreaded controllers in the form of slave or virtual controllers created over a master controller. The communication between the master and slave controllers occurs through eastbound and westbound interfaces (such as hyperflow and application layer traffic optimization).

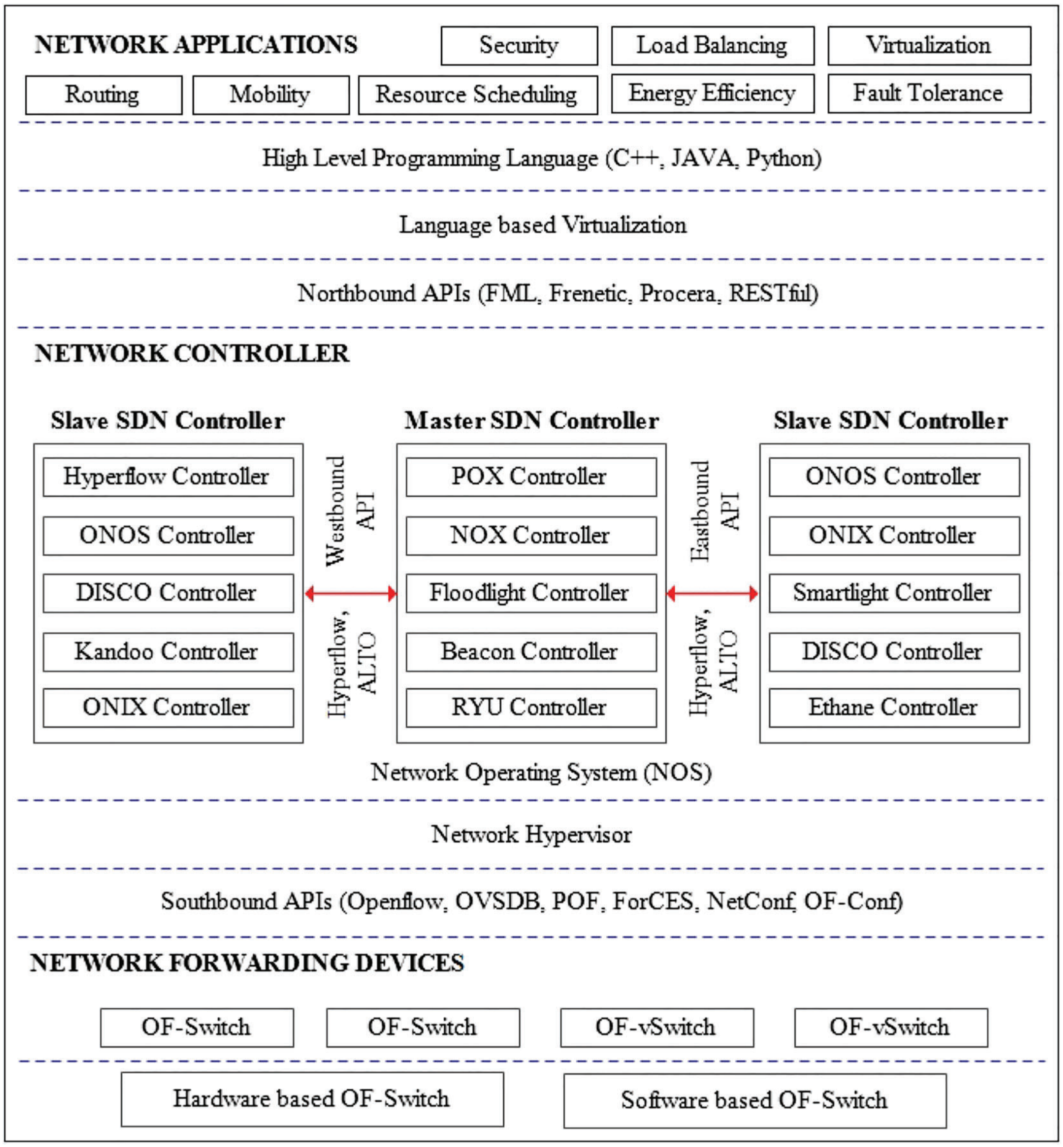


FIGURE 5 Layered architecture of software-defined networking (SDN)

The topmost layer is the application or management plane, wherein various types of network applications run. The network applications include routing, security, wireless and mobility, load balancing, virtualization, resource scheduling, energy efficiency, and fault tolerance. The programmers handle the network applications remotely by different high-level languages (C++, Java, Python),[14] and the instructions are executed on the SDN controller to control the forwarding nodes. SDN applications can exchange messages with the network controller via northbound interfaces. The common northbound interfaces are field manipulation language (FML), Frenetic, Procera, and representational state transfer (REST) APIs.[11]

3.3 | SDN architecture and OF switch working

OF-switch has two components in an SDN architecture, i.e., hardware and software. The hardware component of an OF-switch is a hardware switch fabric (application-specific integrated circuit) of multiple vendors having distinct ternary content addressable memory (TCAM) capacity, which can store multiple flow tables entries.[11,54] The software component of an OF-switch is a programmable switch which performs necessary actions on the flow rule instructions. The software/programmable OF-switch consists of multiple flow tables (flow tables 0 to N) connected via a pipeline.

Each flow table has four fields: match rule, priority, action, and a counter. The match rule checks the attributes such as switch port number, MAC address, Ethernet type, VLAN address, IP address, and TCP address.[8,55] The priority field represents the data priority as high, low, or medium on the basis of various data categories. For example, the data priority of voice and text data is set as high, images are set as a medium, and video has the lowest priority based on bandwidth consumption. The action field is responsible for taking the necessary actions on the data packets based on the assigned priority. The actions can be "to forward the packet to egress ports," "forward normal in a pipeline," "forward to the controller," or "drop the packet." The counter field is responsible for counting the number of packets processed in the queue and their lifetime. The flow tables also have timeout mechanisms consisting of hard and idle timeouts. The hard timeouts are activated after a fixed amount of time, and idle timeouts occur after a period of inactivity.[15] The flow activity of the overall system and how the flow rules are matched and forwarded from a sender node to a receiver node via switches and controllers are explained with the help of an example. In this activity, we have the following components talker (sender), the listener (receiver), centralized user configuration (CUC), centralized network configuration (CNC), and OF switches. The role of the CUC is to collect the information related to talker/listener nodes based on advertisement message and then transmit the request to the CNC for network configuration and finally, collect the statistics of the data transmitted in case of matched flow rules, otherwise trigger the notification back to the CNC for mismatch flow rules and perform the same process again. In the first step, the talker creates a source code of the message (like suppose, talker_adverstise.cc) which is transmitted to the edge switch. In the next step, the edge switch forwards the message to the CUC, but CUC does not perform any computation related to network configuration or path computation for these tasks. It is dependent on the network controller (CNC). So the message (talker advertise) is forwarded by the CUC to the CNC. The CNC consists of three components (reservation handler, path computation engine, and configuration engine). In the second step, the reservation handler application receives the request from the CUC of the talker and advertises a message, which is saved in the database. Then it triggers the notification to the path computation engine application for path computation. The optimization module works at the path computation engine where suppose a Python code is executed and returns the output (like suppose, OPCE.py) as the computed paths by the optimizer to the configuration engine application. The computation engine application computes the forwarding rules installed on the OF switches generated based on idle timeouts and hard timeout policies and returns the output (like suppose, configuration_engine.cc) to the reservation handler. In the third step, the reservation handler informs the CUC that the CNC computes path computation and forwarding rules, so inform the listener ready message to the talker so that the talker finally starts the transmission of the message. In the last step, the talker performs the transmission of the message, which follows the policies of the code deployed by the optimization application, path computation application, and forwarding rules application to forward the message to the correct listener node. In case of a mismatch in the forwarding rules, the CUC triggers the notification to the CNC for new forwarding rules and repeats this process iteratively.

Figure 6 shows the network packet flow from host A having a particular source address (e.g., MAC source: 00-00-00-00-00-00-00-0A, IP source: 20.0.0.1) to host B address (say, MAC destination: 00-00-00-00-00-00-00-0B, IP destination: 20.0.0.2) through OF-switches. The steps used in this process are listed below.

Step 1: Initially, the data packets of host A are sent to the edge switch through access points (Wi-Fi, cellular base station).

Step 2: The packet is received at ingress port 1 of an OF-switch. The programmable switch initiates the lookup process. The flow table stored in an OF-switch performs rule matching for each packet header. The matching field returns the output as no existing matching rule in an OF-switch.

Step 3: The OF-switch first encapsulates the packet and transmits a notification to the SDN controller in case the matching rule does not exist.

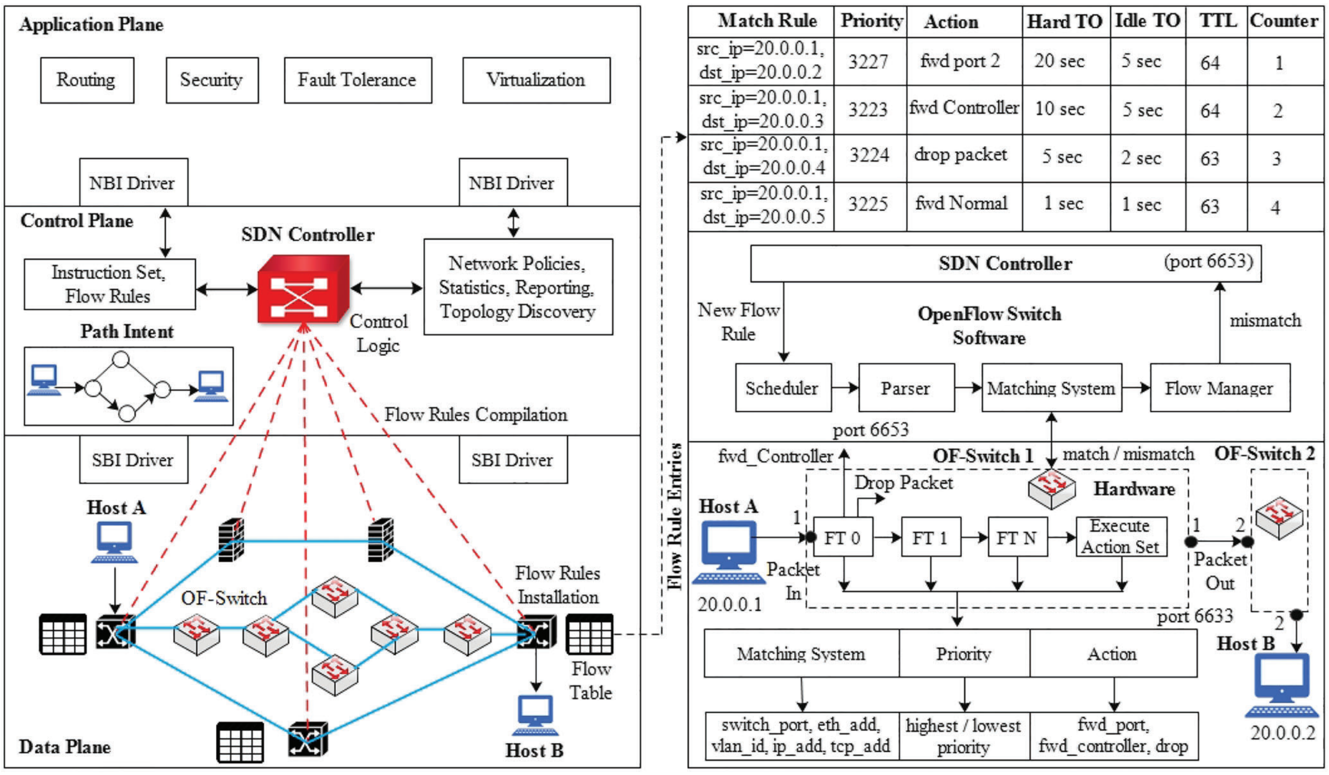
Step 4: The controller creates a new flow and matching rule for routing the packet to the destination address.

Step 5: The controller installs the matching fields on port 1 of an OF-switch and installs the action field on port 2 of the switch.

Step 6: The flow table is updated on the OF-switch. The lookup process is repeated and returns the output as a flow rule for a successful match.

Step 7: Based on priority order, the necessary action to forward the packet on port two is performed. Therefore, the packet is forwarded to the next switch, where the flow table lookup process is repeated. After successful matching, the packet is forwarded to the destination address (host B).

In case the rule is not matched, the action performed is to drop the packet and inform the controller to build a new flow rule. The controller generally listens on port 6653 for every request.



Match Rule	Priority	Action	Hard TO	Idle TO	TTL	Counter
src_ip=20.0.0.1, dst_ip=20.0.0.2	3227	fwd port 2	20 sec	5 sec	64	1
src_ip=20.0.0.1, dst_ip=20.0.0.3	3223	fwd Controller	10 sec	5 sec	64	2
src_ip=20.0.0.1, dst_ip=20.0.0.4	3224	drop packet	5 sec	2 sec	63	3
src_ip=20.0.0.1, dst_ip=20.0.0.5	3225	fwd Normal	1 sec	1 sec	63	4

FIGURE 6 Software-defined networking (SDN) architecture and working of an OpenFlow switch

3.4 | Network functions virtualization (NFV)

The advancement in technology led to the expansion of the size of the physical infrastructure, thereby making it more capable of offering innovative services to users/tenants. However, the scarcity of network resources always exists as the tenants' demands rise exponentially. The only solution is to efficiently utilize the physical resources among tenants. Network virtualization is an imminent solution wherein the physical infrastructure sharing capabilities are capitalized to meet the high demands of tenants. The concept of NFV intertwined with SDN in 2012 at Stanford University with the development of OF architecture.[11,12] The concept of virtualization in an SDN architecture is the slicing of physical infrastructure into multiple logical isolated virtual SDN networks. The network slicing technique improves the utilization of hardware resources. Each independent network slice has its resources such as link bandwidth, switch CPU resources, switch forwarding tables, and network topology. On this basis, network slicing is performed at each layer. For example, at the physical layer, the wavelength division multiplexing (WDM)[17] creates an instance of the physical resources. Similarly, the network slices are created by provisioning VLANs at the data link layer. MPLS creates the network slices of the forwarding tables in an OF switch, and virtual routing and forwarding (VRFs along with the specific VLAN mapping) create virtualization at the network layer (layer 3).[13] The network hypervisor (VM monitor) is a software installed at the NOS to create network virtualization in the SDN architecture. Both hypervisors' centralized and distributed architecture are used to monitor the virtual SDN networks. Figure 7 shows an architecture of a network hypervisor in an SDN architecture. The physical controller (pSDN) monitors the global topological view to make intelligent flow forwarding decisions. The pSDN is connected via a network orchestrator and virtual manager to handle interdomain routing. The network orchestrator acts as a centralized hypervisor and resides on the pSDN. The distributed hypervisors (northbound and southbound) are located at the top of the network orchestrator and virtualize the multidomains. The northbound and southbound hypervisors are installed at the control plane using the OF protocol. The hypervisor creates a logically virtual SDN network (vSDN) on the top of the pSDN. Figure 7 depicts three virtual SDN network slices, where a virtual/proxy controller (vSDN) is located in each virtual slice. Tenants run multiple applications simultaneously on every vSDN controller in all virtual networks at the application plane. NFV technology is useful in SDN to create the network, storage, and server virtualization. Network virtualization runs virtual network applications such as routing, firewall, access control, and load balancing. It also creates an instance of the network

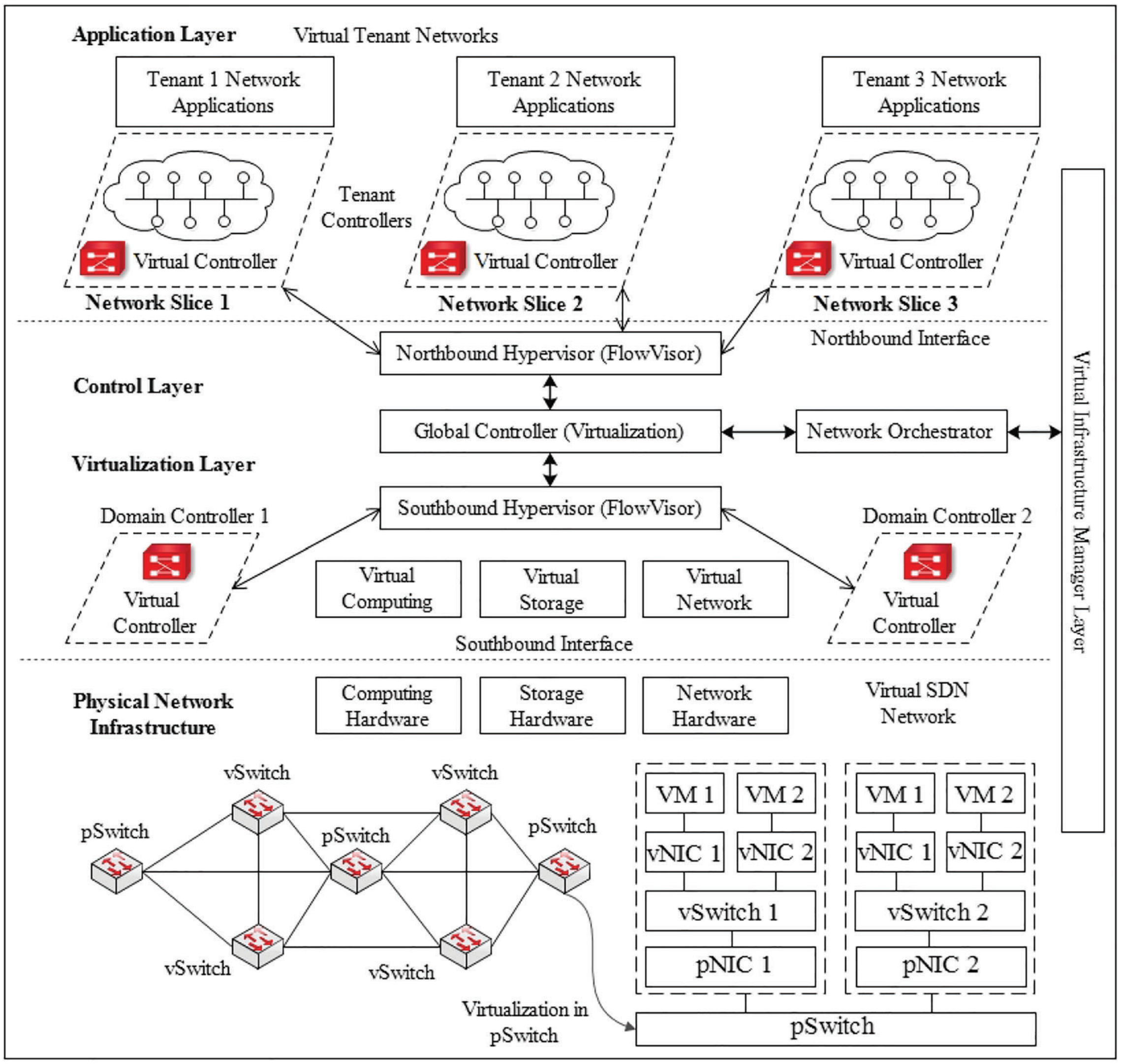


FIGURE 7 Network hypervisor in an software-defined networking (SDN) architecture

resources such as virtual switch (vSwitch), virtual router (vrouter), and vSDN.[13] At the data plane, multiple vSDNs create an instance of the physical switch (pSwitch) to create logically isolated vSwitches.

The pSwitch consists of multiple physical network interface cards (pNICs), and each pNIC creates a vSwitch by creating virtual NICs via a hypervisor. The hypervisor software creates virtual NICs to run several VMs built on the top of an individual PM because each VM requires its vNIC. Therefore, the flow table entries are executed both at the pSwitch, and the vSwitches.[56] The virtualization layer is a proxy or middleware between the forwarding nodes and controller and between the controller and the network applications. FlowVisor is the most popular hypervisor software used to efficiently utilize physical hardware infrastructure in the SDN planes. Table 3 shows the list of various network hypervisor software along with their description.

4 | SDN DEPLOYMENT IN SMART APPLICATIONS

Figure 8 shows a scenario where an SDN controller is deployed in various smart applications for smart communities. These applications include DCN, edge/fog computing, IoT, and WBAN. In all these applications, SDN controllers are deployed to improve communication to minimize the latency, increase bandwidth utilization, and reduce the overall operational and capital expenditures.[57] The taxonomy for the potential SDN controller deployment models in various smart applications is as shown in Figure 9. A detailed description is provided below.

TABLE 3 List of network hypervisor softwares integrated with SDN

Hypervisor softwares	Protocols and APIs	Architecture	OFS	OS	CPI	DPI	Description of the hypervisor features
FlowVisor[12]	JavaScript Object Notation, OpenFlow	Centralized	✓	✓	×	✓	Middleware between the tenants SDN controllers and the OF-pSwitches. Isolates: bandwidth, topology, flow space, switch CPU, forwarding tables, control channel.
VeRTIGO[12]	REST, NetConf, OpenFlow	Centralized	✓	×	×	×	Provides additional capacity in FlowVisor by creating virtual links and ports.
OpenVirteX[12]	JavaScript Object Notation RPC, OpenFlow	Centralized	✓	✓	×	✓	Each network slice has its complete flow space without overlapping.
FlowN[12]	OpenFlow, OVSDB	Centralized	✓	×	✓	×	Provides a MySQL database and performs container-based application virtualization.
BYOC-Visor[56]	OpenFlow, Generic Routing Encapsulation tunnels	Centralized	✓	✓	✓	×	Provides customized and secure services to tenants through security applications.
Ad-Visor[12]	JavaScript Object Notation, OpenFlow, GRE	Centralized	✓	✓	×	✓	Improves topology abstraction and sharing of the flowspace by vSDNs. The tenant SDN controller inspects only end-points of the virtual path for packet forwarding.
Slice Isolator[12]	OpenFlow, NetConf	Centralized	✓	✓	✓	✓	Allow different vSDNs controllers to exchange flow tables for similar functions. Allow interface isolation between the vSDNs and physical port to map the incoming data.
MobileVisor[12]	OpenFlow with GPRS Tunneling Protocol (GTP)	Centralized	✓	✓	✓	✓	Proxy software and acts as a mobile controller. Slices the mobile packet network according to services offered by the mobile operators (3G, 4G).
Optical FlowVisor[12]	OpenFlow, GRE	Centralized	✓	✓	×	✓	Allow mobile operators to create virtual optical networks (VONs) on the physical infrastructure. Each isolated VON is a collection of virtual optical switches connected via virtual optical links. The services offered by virtual links depend on the wavelength channels in a single optical fiber.
Hyperflex[12]	OpenFlow, REST APIs	Distributed	✓	✓	✓	✓	Prevents network to degrade its performance by over-utilizing the hypervisor layer. Allocate each tenant with a particular amount of CPU resources in advance.
AutoSlice[12]	OpenFlow, REST APIs	Distributed	✓	✓	✓	✓	Share the workloads by slicing the hardware infrastructure into multiple physical domains. Assign multiple proxies controllers and set one proxy on each physical domain.
AutoVFlow[12]	NetConf, OpenFlow, REST APIs	Distributed	✓	✓	✓	✓	Perform slicing of the infrastructure into multiple domains and acts as a container for each domain. The distributed administrator in multiple domains is configured by a centralized hypervisor. A virtual identifiers/MAC address is used as an identifier of the flow packets and is unique for every domain.

Abbreviations: CPI, control plane isolation; DPI, data plane isolation; OFS, OpenFlow support; OS, open source; SDN, software-defined networking.

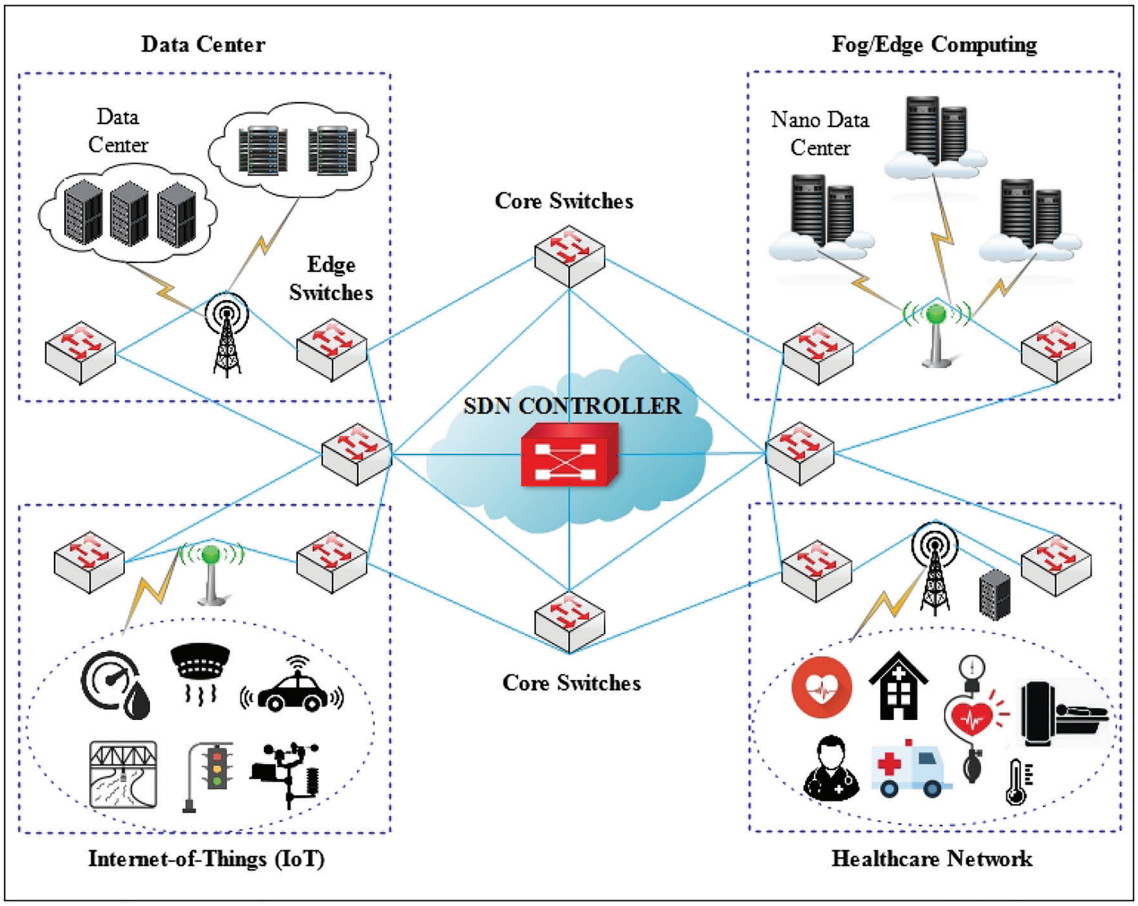


FIGURE 8 Use of software-defined networking (SDN) controller in real-world applications

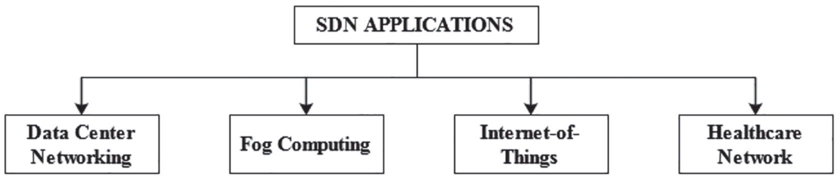
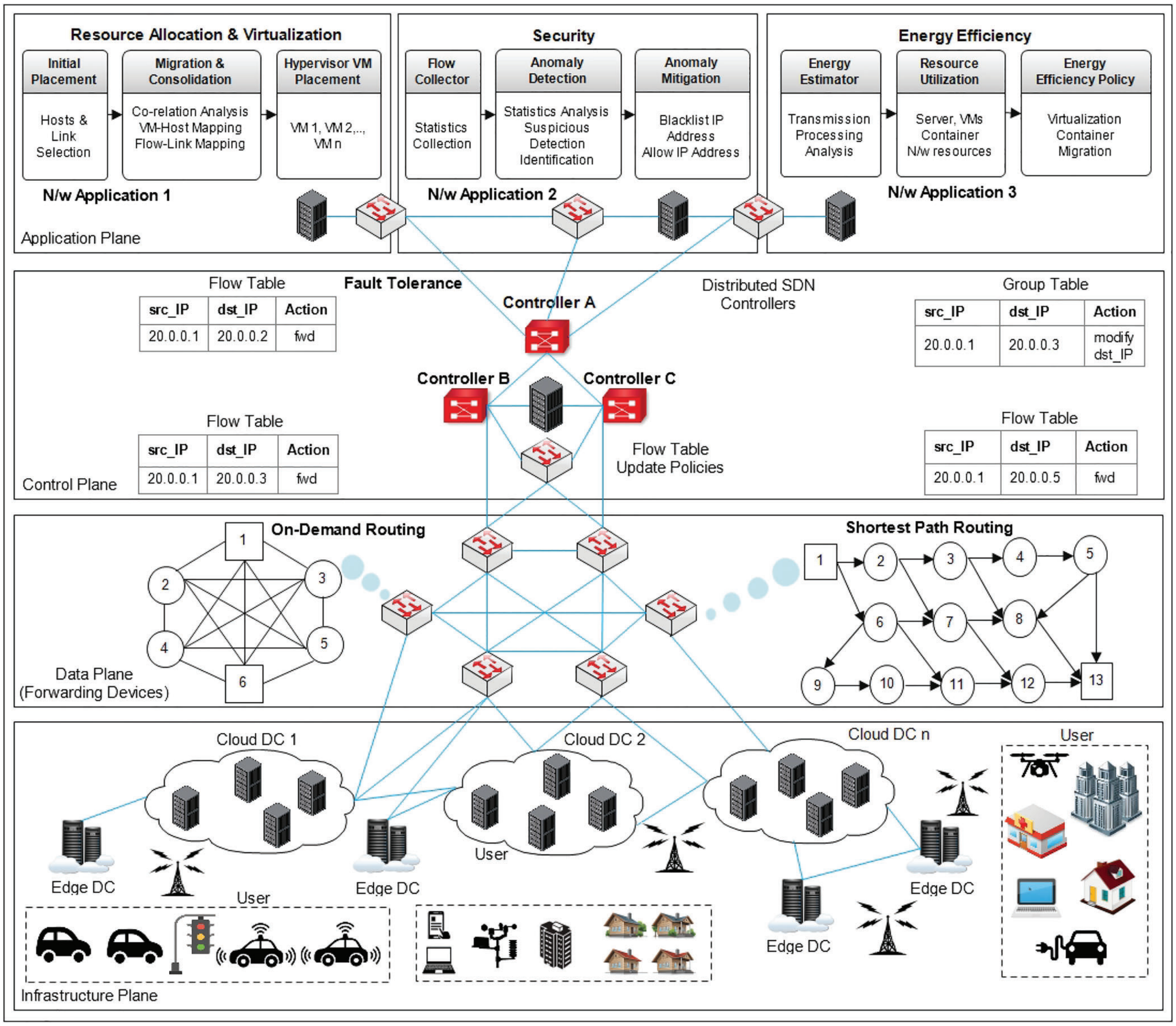


FIGURE 9 Deployment of software-defined networking (SDN) controllers in various applications

4.1 | DCN

SDN controller is deployed in DCN to resolve the issues related to energy management, failure recovery, resource allocation, routing, security, and virtualization. The communication architecture of software-defined data centers (SD-DC) is shown in Figure 10. Apart from the traditional SDN planes (application, control, and data), an additional plane just below the data plane named as infrastructure plane is used wherein various infrastructure is located. All these planes are discussed in context to DCN as below.

Some of the major applications considered at the management plane are resource allocation, virtualization, energy efficiency, and security. The first application is resource allocation and virtualization. Virtualization helps to achieve efficient allocation and utilization of resources. To achieve these goals, controller performs three steps: (a) *initial placement* based on hosts and links selection; (b) *VM migration and consolidation* based on correlation analysis among the network resources, mapping between the VM and hosts, and flow-link mapping; and (c) *hypervisor VM placement* is selected on the basis of the number of VMs required. Using these three steps, the remote programmer handles resource allocation and virtualization. Moving ahead, to handle the security threats on the control and data plane, the controller



Flow Table

src_IP	dst_IP	Action
20.0.0.1	20.0.0.2	fwd

Group Table

src_IP	dst_IP	Action
20.0.0.1	20.0.0.3	modify dst_IP

Flow Table

src_IP	dst_IP	Action
20.0.0.1	20.0.0.3	fwd

Flow Table

src_IP	dst_IP	Action
20.0.0.1	20.0.0.5	fwd

FIGURE 10 Communication architecture of software-defined data centers (SD-DC) and edge DC

follows three processes: (a) *flow collector* which maintains the record of the data flow statistics, (b) *anomaly detection* which monitors the statistics analysis and detects the suspicious identification of IP addresses, and (c) *anomaly mitigation* which blocks the flow of blacklisted IP addresses and allows only normal IP addresses.[58] Using these three processes, the security issue is mitigated by the programmer. The next challenge is to integrate energy efficiency with other applications. The controller handles this application using three activities: (a) *energy estimation* during transmission and processing; (b) *resource utilization* of servers, VMs, containers; and (c) *energy efficiency policies* implemented for virtualization and containers migration. These network applications are deployed over an SDN controller at the control plane. At the control plane, the distributed SDN controllers (say A, B, and C) are interconnected so that the issue of fault tolerance can be solved. The SDN controller broadcasts the updated flow policies, and the flow tables get modified and updated on the distributed controllers accordingly. The flow between the distributed SDN controllers and the forwarding devices is selected at the data plane through shortest path routing (SPR). Finally, multiple cloud DCs and edge DCs are responsible for providing services to the end-users at the infrastructure plane. Therefore, the dynamic traffic is handled on the basis of on-demand routing so that the flow table policies can be dynamically modified by using the programmable SDN controller. The taxonomy of SD-DC networks is as shown in Figure 11. A comprehensive discussion of the existing proposals related to the deployment of SDN controllers in DCN is given below.

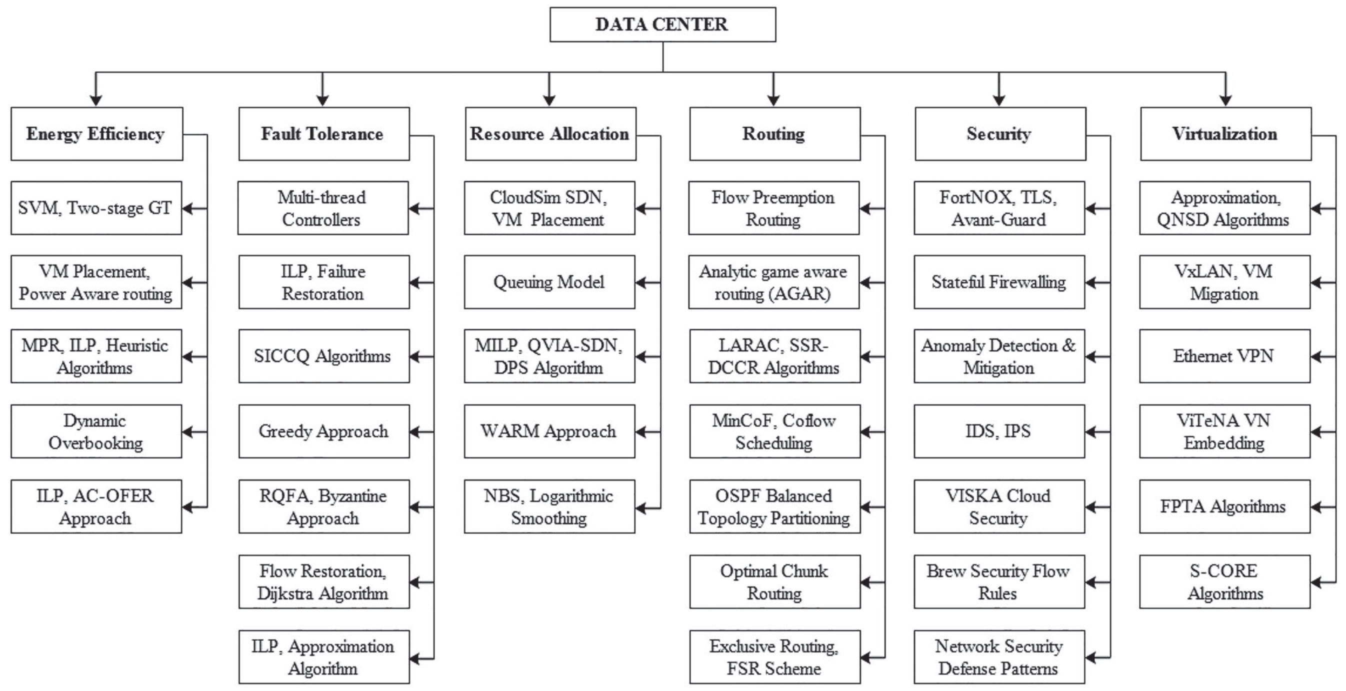

DATA CENTER
Energy Efficiency
SVM, Two-stage GT
VM Placement, Power Aware routing
MPR, ILP, Heuristic Algorithms
Dynamic Overbooking
ILP, AC-OFER Approach
Fault Tolerance
Multi-thread Controllers
ILP, Failure Restoration
SICCQ Algorithms
Greedy Approach
RQFA, Byzantine Approach
Flow Restoration, Dijkstra Algorithm
ILP, Approximation Algorithm
Resource Allocation
CloudSim SDN, VM Placement
Queuing Model
MILP, QVIA-SDN, DPS Algorithm
WARM Approach
NBS, Logarithmic Smoothing
Routing
Flow Preemption Routing
Analytic game aware routing (AGAR)
LARAC, SSR-DCCR Algorithms
MinCoF, Coflow Scheduling
OSPF Balanced Topology Partitioning
Optimal Chunk Routing
Exclusive Routing, FSR Scheme
Security
FortNOX, TLS, Avant-Guard
Stateful Firewalling
Anomaly Detection & Mitigation
IDS, IPS
VISKA Cloud Security
Brew Security Flow Rules
Network Security Defense Patterns
Virtualization
Approximation, QNSD Algorithms
VxLAN, VM Migration
Ethernet VPN
ViTeNA VN Embedding
FPTA Algorithms
S-CORE Algorithms

FIGURE 11 A Taxonomy of software-defined networking (SDN) in data center networking

4.1.1 | Energy management in DCN

Nowadays, most smart applications and services are hosted over massive DCs located at geographically distributed locations. DCs are responsible for providing various resources (storage, memory, network, and computational resources) on a pay-per-use basis to the end-user in order to run various applications. With an exponential increase in Internet services, the application base dependent on DCs has also witnessed huge growth. The demand for the end-users domain has also increased substantially due to this growth. Therefore, to handle massive workloads, the DCs infrastructure has been expanded horizontally and vertically. With such an expansion, a huge amount of energy is consumed by the geo-distributed DCs to serve millions of user requests daily. A major chunk of the energy consumed in DCs is related to underlying network infrastructure. Being programmable technology, SDN has a lot of potentials to handle the critical issue of energy consumption dynamically.[59] In this direction, various proposals have been presented, wherein SDN has played an important role in energy savings in DCN. Some of the proposals are discussed below. Aujla et al[60] proposed an SDN-based energy management scheme in cloud DCs. The authors highlighted the role of SDN in DCNs in terms of energy saving. Also, in Aujla and Kumar,[61] the authors designed an energy-aware flow scheduling mechanism which provides a large amount of energy savings and improves latency in DCNs. In another work, Yu et al[62] optimized the energy consumption cost by presenting a VM placement and power-saving routing algorithm. The VM placement problem involves placing several VMs on a single host PM to run various network applications concurrently to achieve maximum resource utilization. The authors focused on the resource allocation for VMs and bandwidth requirements to save the total power consumption of OF-switches and routers. Similarly, Aujla and Kumar[63] proposed two approaches for the integration of renewable energy sources with SD-DCs. The first approach was the support vector-based machine classifier for workload distribution of incoming jobs. The second approach was based on game theory to handle the scheduling of workload requests. A consolidation approach was also presented to compute the optimal number of OF-switches required to serve the flow traffic with minimum latency. The results show that the proposed approach can save operational costs and improve the network delay time.

Zeng et al[64] examined the issue of energy management in DCN and formulated it using integer linear programming (ILP). The authors proposed a multipath routing and computation-efficient heuristic algorithm to save the energy consumed by the activated OF-switches. The simulation results proved that the scheduling time of the activated switches was reduced from hours to seconds in contrast to the existing algorithms. In Son et al,[65] the authors designed a VM migration algorithm with a dynamic overbooking strategy instead of an existing fixed booking strategy to handle the dynamic workloads. The dynamic resource overbooking strategy reduced the energy consumption of hosts by placing reservation requests for the network resources. Therefore, prior knowledge of workloads was predicted based on the online historical analysis to compute the resource allocation ratio. Similarly, Amokrane et al[66] formulated online-flow routing as an ILP problem in order to save the overall energy consumption at DCs. The author solves the ILP problem using Ant Colony based on the Online Flow-Based Energy-efficient Routing (AC-OFER) approach. The AC-OFER approach uses four steps: (a) to compute K possible paths for each incoming flow from a user, (b) to compute a probabilistic solution by choosing the maximum probability and score among K paths, (c) to choose an optimal solution, and (d) to update the pheromone trails for each flow. The experimental results show that the AC-OFER approach reduced the overall energy costs up to 35%, 44%, and 49% compared with minimal residual capacity (MRC), SPR, and load balancing approaches. Chaudhary et al[67] proposed an energy optimization and flow forwarding scheme to resolve the issues of power consumption on OF switches at the data plane. The proposed solution applies a heuristic scheme for path computation using the datasets of NorthAmerica, where the simulation results are performed on OMNET++.

4.1.2 | Fault tolerance in DCN

Failure tolerance is one of the most important issues in context with the centralized controller to manage the network policies in geo-distributed DCs. Being a single point of failure, this aspect has witnessed huge interest from researchers, but the problem is still not fully investigated. In this direction, Akyildiz et al[9] presented a deep analysis of the traffic engineering (TE) approach in SDN. The authors focused on four parameters: (1) flow scheduling, (2) fault tolerance, (3) topology update for new or existing policies, and (4) network traffic analysis. They introduced deploying distributed controllers, multithreaded controllers, and generalized controllers for failure recovery. Moreover, some of the existing proposals have utilized the property of network virtualization to distribute the logical controllers among tenants. In this context, in a previous research,[68] the authors formulated an ILP problem for handling the virtual network resource allocation issue. For this purpose, an OF-based failure restoration algorithm is designed, which dynamically diverts the network traffic for link failure restoration. The proposed algorithm is compared with the existing pro-active path computation algorithm. Finally, the evaluations are performed on a Mininet simulator using a POX controller to link 20 end points with 44 bidirectional network links. The results obtained depict that the proposed approach can restore around 60% of damaged links, while the pro-active algorithm restores only 32.6% of damaged links.

In another work,[69] the author presented a congestion control framework and solved the congestion events in DCs using SDN-based incast congestion control via queue-based monitoring (SICCQ) algorithm. The SICCQ algorithm implements two major events, that is, packet arrivals to the switch ports and incast detection timer expiry. The incast congestion happens when a many-to-one traffic pattern is followed. It means multiple synchronized requests are sent to a single DCN. In addition, a SICCQ hypervisor algorithm is designed to handle the incoming packets of different types, that is, incast ON, incast OFF, and TCP acknowledgement. Liu et al[70] addressed an important issue related to controller placement problem (CPP) in terms of network reliability. The CPP issue is resolved using two algorithms named: (1) clustering-based global optimization algorithm and (2) controller placement algorithm based on greedy approach. The K-mean clustering algorithm splits the complete domain into multiple subdomains, and each subdomain consists of multiple OF-switches controlled by a single controller. The greedy approach finds an optimal solution for multipaths to compute the link's reliability between each cluster's node and the controller. Similarly, in earlier study,[71] the authors handled the issue of CPP as controller assignment in fault-tolerant SDN (CAFTS). The CAFTS problem is resolved by the Byzantine approach using a cost-efficient requirement first assignment (RQFA) algorithm. The RQFA algorithm maintains a set of controllers as candidates for the assignment. The candidates with minimum residual capacity are initially assigned to accommodate the switches, and then the performance is checked. If, in a case, the performance is not satisfactory, then a new set of candidates is assigned.

In another work, Astaneh et al[72] proposed a dynamic approach using the optimal Dijkstra-like path cost algorithm. This algorithm solves the issue of flow restoration, and the results obtained show that the proposed approach minimizes the number of operations up to 50% with a minimum link failure. Similarly, Xie et al[73] addressed the fault-tolerance issue in multiple controllers in DCN. The authors designed an optimal solution by choosing an approximation algorithm to minimize the overall network overhead on the controllers. The proposed technique supports failure recovery on distributed controllers and reduces the communication overhead generated from state synchronization. Likewise, in an earlier study,[74] the authors proposed an optimal placement, resilient, and controller availability scheme to address the issues of network scalability and resilience for software-defined data centers. The proposed scheme computes the stable network partitioning and optimal availability of primary and backup controllers for multiple controller placement by using cooperative game theory and testing the resilience level up to two in the case of controllerless nodes.

4.1.3 | Resource allocation in DCN

Resource management (bandwidth utilization and channel allocation) in SDN architecture is another major issue faced by the DCN. In this direction, Son et al[75] resolved the resource management issue using a *CloudSimSDN* simulator. The authors compared the performance of *Mininet* and *CloudSimSDN* simulators for resource allocation tasks in DCs. According to this analysis, the *Mininet* simulator is suitable for configuring and testing SDN controller resources. But, on the other hand, the cloud characteristics such as workload assignment, VM placement, and VM migration rules are difficult to simulate on *Mininet* emulator. For these specific tasks, the *CloudSimSDN* simulator is the most suitable candidate. In another work, Li et al[76] addressed the issue of flow-level scheduling in the SDN environment. The authors implemented a queue scheduling scheme to provide QoS guarantees for flow scheduling on SDN switches. The data are categorized into application-specific queues, and the applications with the highest priority are served first. The numerical results show that the scheme minimizes the overall latency by 28%, increasing the average throughput to 90.17%. In a similar work, Aujla et al[50] used decision tree and queue management schemes to provide energy-efficient resource allocation in DCN.

In an earlier study,[77] the authors formulated the virtual resource allocation problem in DCs using mixed integer programming (MIP). The virtual allocation specifies the necessary requirement to configure the virtual resources. For example, the configuration of vSwitches requires the number of flow table entries, memory, and processing power, while the configuration of VMs requires memory, storage, and virtual CPUs. The author presented a QoS-aware virtual infrastructure allocation scheme for SDN-based DCs (QVIA-SDN) to identify the number of physical resources available for hosting the virtual infrastructure. In addition, the authors implemented a deterministic path selection (DPS) algorithm to minimize the number of switches by mapping flow entries. For this purpose, the existence of a physical path to host virtual links is checked. In a previous study,[78] the authors proposed a workload-aware revenue maximization (WARM) scheme to increase the net profit of service providers. The simulation results proved that the WARM approach efficiently utilizes the VMs to compute the best routing path for all applications. The proposed scheme results in a decrease in the round-trip time (RTT) of jobs compared twith the round-robin algorithm and open shortest path first approach. In another research,[79] the authors presented a Nash bargaining solution (NBS) that improves the network bandwidth to service providers and saves the delay time. The authors considered the resource allocation issue as an optimization problem and proposed a request allocation scheme based on logarithmic smoothing to solve the optimization problem.

4.1.4 | Routing in DCN

The energy minimization problem in DCs can be resolved by applying an efficient routing scheme. In this direction, Zhang et al[80] designed a routing scheme named flow preemption routing with redundancy elimination (FPR-RE). This scheme manages flow scheduling and eliminates the data redundancy in OF-routers. It inspects the flow execution time and energy consumption of traffic flow. The author used a fat-tree topology and simulated the proposed scheme using Mininet and POX controllers. In order to choose the optimal routing path for a cloud gaming network, Amiri et al[81] designed a bi-objective optimization scheme. The author also designed an analytic game aware routing approach to delivering a high-quality experience for online players by reducing the delay and improving the bandwidth utilization. The experimental results show that the proposed approach minimized the end-to-end (E2E) latency by 19%, 17%, and 14% compared with Dijkstra, equal-cost multipath routing (ECMP), and the Hedera routing algorithms. The proposed scheme also improved the bandwidth utilization by 5% and 7% compared with ECMP and Hedera routing algorithms.

In another work, Guck et al[16] presented a survey on routing algorithms for QoS in SDN. The authors identified four dimensions (4D) suitable for the QoS routing approach. The first dimension represents the topology category, while the second and third dimensions represent the scaling of the topology into horizontal and vertical axis, that is, (X-axis, Y-axis), and the fourth dimension represents delay constraints. The author analyzed two best algorithms: (1) Lagrange relaxation-based aggregated cost (LARAC) and (2) search space reduction delay-cost-constrained routing (SSR-DCCR) to fit all the 4D space. Similarly, Chiu et al[82] reduced the communication time in DCN by designing a routing scheme, namely, coflow scheduling and minimal coflow scheduling and routing (MinCoF). The results obtained for the proposed scheme are compared with the OF-scheduling and routing, where the proposed scheme outpowers the average execution time by 12.94%. Caria et al[83] presented an open shortest path first balanced topology partitioning scheme to control prioritized data traffic. The designed scheme partitions the network topology into multiple subdomains and proves that each subdomains network control capabilities perform better compared with the single network topology. In another work, Wu et al[84] addressed the issue of massive data migration in inter-DCs in order to achieve maximum available bandwidth utilization. The authors proposed a bandwidth-reserving algorithm to reserve the channel links and bandwidth for sending blocks of newly arrived tasks. In addition, the authors presented two approaches, that is, a dynamically adjusting approach and a future-demand friendly approach. The proposed approach manages full utilization of the available bandwidth at each interval by transmitting a chunk along with the pending future transmission schedules on the network links. The performance of the algorithms improves the routing optimality and computational delay. In Li et al,[85] the problem of high power consumption by DCs is solved by adopting an energy-aware flow scheduling using an exclusive routing scheme in the time dimension for each flow. The proposed algorithm schedules each flow by defining flow priorities based on flow size and deadline. The results show that the average flow completion time decreases if the highest priority is selected for minor flows comparable with fair-sharing routing (FSR). In a previous study,[86] authors proposed a bloom filter based fast flow forwarding scheme at the control plane for big data migrations in SDN-based multicloud environments. In Aujla,[87] the authors solved the issue of traffic congestion in the traditional networking architecture. The proposed solution is a data offloading scheme using Stackelberg game theory to make intelligent decisions for flow forwarding of traffic based on priority manager and load balancer approach at the control plane.

4.1.5 | Security in DCN

Security is another major issue faced in the DCN environment. It becomes crucial to building consistent security policies to analyze the anomalous traffic patterns in SDN and NFV platforms. In an earlier study,[20] the authors performed a comprehensive survey on the security challenges across SDN-based cloud DC environments. The survey focused on the distributed denial of service (DDoS) vulnerabilities that occurred by an adversary on each SDN plane. Additionally, the safeguard mechanisms are also discussed to mitigate DDoS attacks. Similarly, Yan et al[88] stressed that the DDoS attacks launched by an adversary on the application, control, and infrastructure planes could have adverse effects on the performance. An adversary can launch the DDoS attack using the following: network applications, northbound API, attacking controller, westbound and eastbound API, attacks on switches, and southbound API. In a different work, Lorenz et al[89] designed fine-grained security policies to defend from security threats by implementing stateful firewalling within an SDN and NFV integrated architecture. The authors used three approaches for the software-based stateful firewall placement patterns. A first approach is a controller-centric approach based on placing a firewall in controller software. A second approach is a VNF-centric approach wherein a virtualized firewall is deployed on the cloud infrastructure, and the connection is set up at the data plane. The third approach is the hybrid SDN/NFV approach which implements controller-centric at the controller for intensive connections and VNF centric at the data plane for connections setup.

Giotis et al[90] identified malicious events (typically DDoS attacks) in hardware and software-based OF-devices. The authors designed an anomaly detection technique using an entropy-based method. The proposed scheme analyzed all the collected flow statistics using a time window of 30 s to monitor the abnormal activities in the traffic patterns. Moreover, once the malicious events and the entropy values of the IP address and port address (layers 1–4 attacks) are detected, the authors implemented an anomaly mitigation technique using OF to drop the corresponding malicious traffic. In an earlier study,[91] the authors introduced an intrusion detection and prevention system (IDS/IPS) in order to mitigate malicious attacks. The suspicious activity is identified by deploying IDS and IPS at all the gateway nodes so that the SDN controller may block such malicious traffic from the control flow. In earlier research,[92] the authors presented a cloud security service named VISKA to identify malicious switching nodes by partitioning the data plane into virtual networks using virtualization rather than detecting the malicious activity on the entire physical network. The dynamically isolated partition creates virtualized network views of the data plane and easily localizes the abnormal forwarding behaviors at the switch level.

In various existing proposals, the proof-of-concept prototype of a conventional layer-3 firewall in SDN is discussed. However, none of the proposals addressed the issue of flow rule conflicts at any layers. Pisharody et al[93] addressed the problem of flow rule conflicts at the control and data layer in distributed SDN controllers like ODL controllers. The authors designed a brew security policy mechanism and a novel visualization approach to resolve conflicts in flow rules. Initially, the proposed scheme applies a global flow rule prioritization method for the distributed SDN controllers. Then, the cross-layer checks the flow rule conflicts by extending the conventional firewall policies of conflict classification. Finally, the authors implemented a visualization method to monitor the conflicts graphically by a network administrator. In another study,[94] the authors discussed building an adequate security solution in service function chaining (SFC). This technique guides data traffic flows using network functions in SDN and NFV architecture. The authors presented a defense mechanism known as network security defense patterns (NSDP) to provide optimal placement of virtual security functions while deploying SFC in SDN. The security functions are transparently added to the traffic flows to perform security checks on different application types at each level.

4.1.6 | Virtualization in DCN

Feng et al[95] identified three important issues in network virtualization, that is, (i) optimal placement of virtual network functions, (ii) flow consideration in limited available resources, and (iii) the allocation of network resources. The authors resolved all these issues using the fast approximation algorithm named queue-length based service distribution (QNSD). In another work, Zhao et al[96] discussed the limitations of VLAN to build large multitenants DCN. The VLAN provides inefficient use of available network links, limited scalability (uses 12-bit segment ID that supports a maximum of 4094 tenants ID) to address layer two segments, and inefficient placement of multitenants segments. The authors adopted an SDN-based virtual extensible local area network (VxLAN) architecture.[18] VxLAN uses a 24-bit segment ID that supports more than 16 million tenants and has greater extensibility to address the issues of VLAN. In another study,[97] the authors enhanced the SDN functionalities for ethernet virtual private network (EVPN). The authors designed an SDN-based DCs framework to automate EVPN deployment and dynamic routing policies for EVPN.

In an earlier study,[98] the authors identified a performance degradation problem of isolated virtual networks in a DCN. The authors designed a scalable solution called the ViTeNA approach based on a virtual network embedding algorithm to provide a high bandwidth guarantee to tenants. The algorithm monitors the global network outlook to allocate virtual nodes. The proposed solution consolidates the virtual networks on limited physical resources and enforces incremental consolidation for every request instead of periodic consolidation. Similarly, Wang et al[99] formulated two problems, that is, (a) VM migration time when multiple VMs are migrated and (b) bandwidth requirement for each migration. The authors considered the problem a MIP problem and solved it by using a fully polynomial time approximation algorithm. The designed approach is compared with the existing primary programming problem and the state-of-the-art algorithms. The results show that the designed scheme decreases the service downtime by 20% and the VM migration time is decreased by 40%.

In an earlier study,[100] the authors proposed a framework for live VM management in order to increase the throughput and reduce the overall network-wide communication cost. The proposed S-CORE scheme combines three orchestration algorithms, namely, round-robin, best-fit, and look-ahead algorithm. The scheme uses two DC network topologies, that is, *canonical tree* and *fat tree* topology and is implemented in an NS-3 simulator. The results show that the aggregate throughput is increased to six times based on 12 live VM migrations, and by migrating 50% VMs, the VM-to-VM communication cost is reduced up to 70%.

The overview of various existing proposals for DCN using SDN architecture is shown in Table 4.

4.2 | Fog and edge computing

Edge and Fog Computing are the latest revolutions in the cloud era, which can be viewed as a complement to the cloud DCs.[101] These technologies have added value to the existing cloud infrastructure to reduce the end-to-end (E2E) latency by migrating the cloud services toward the edge nodes. The edge nodes are closer to the end-users and can deliver services on the move. However, such an ecosystem faces stiff challenges from the dynamic network requirements, which can be solved using the SDN paradigm.[102] For example, Baktir et al[19] presented a survey on software-defined edge computing (SD-EC). The authors highlighted the advantages of technologies such as fog computing, cloudlet, and mobile edge computing (MEC) which an SDN controller manages. The SDN enables network programming at the network's edge to reduce the overall service latency. The taxonomy of software-defined-fog computing (SD-FC) network is as

TABLE 4 Existing proposals on DCN using SDN architecture

Authors	Techniques	Controller	Topology	Setup description	Coding/ simulator	Issues considered 1	2	3	4	5	6	Parameters 7	8	9	10
61	SVM, game theory	Centralized	Tree	25 servers; EVs- 16 kWh and 20 kWh saves upto 40% energy	Matlab	✓	×	✓	×	×	×	✓	✓	×	×
62	Power-aware routing	Distributed	Fat-Tree, BCube	16 servers; 20 VMs; RP_PAR (2.9kW) saves 12% energy	-	✓	×	✓	×	×	✓	×	×	✓	×
64	Minimum switch activation routing	Centralized	Fat-Tree	10 pods; 12,500 flows takes 11s saves around 48% energy	-	✓	×	×	✓	×	×	×	✓	×	×
65	Overbooking dynamic resources	Centralized	Fat-Tree	16 hosts; 4 edge; 4 aggregate switches saves 30.29% energy	CloudSim	✓	×	✓	×	×	✓	✓	×	✓	×
66	AC-OFER approach, Ant Colony	Centralized	Tree	100 APs; 27 switches; 2 router; 50 requests/hr saves 7% energy	Java	✓	×	✓	✓	×	×	✓	✓	×	×
69	SICCQ approach, queue monitoring	Centralized	small Fat-Tree	1 core; 2 aggregate switches; 48 servers/racks takes 200 ms	ns-2	×	✓	×	×	×	✓	×	✓	✓	×
70	Greedy approach, K-means clustering	Distributed	Internet Zoo	Multi-Path: 6 controllers; 37 nodes gives 0.2 failure rate	-	×	✓	×	×	×	×	×	✓	×	×
71	Requirement first assignment	Distributed	Fat-Tree	100 servers; 20 switches and 2 controllers takes 100 ms	Mininet	×	✓	✓	×	✓	×	✓	✓	✓	×
73	Approximation algorithms	Distributed	Jellyfish	16000 servers; 1000 switches; each controller capacity: 10-50 reduced the time complexity from $O(C.N^2)$ to $O(N \times \gamma)$	-	×	✓	×	×	×	×	✓	✓	×	×
75	VM placement	Centralized	Tree	548,341 hosts; 19,999 switches saves around 23% energy	CloudSim	✓	×	✓	×	×	×	×	✓	✓	×
76	Queuing model	Centralized (Floodlight)	Tree	3 application flows: (a) voice, (b) video, (c) generic flow. Reduced delay by 99.9%, 90.6%, 27.84%. Throughput increased by 90.1%, 76.06%, 18.5%.	Mininet	×	×	✓	✓	×	×	✓	✓	✓	✓
78	WARM approach	Centralized	Fat-Tree	4 pod; 16 VMs; 20 switches; bandwidth between VM and each switch is 1 Gbps with execution time 2.03 $\times 10^{-5} s$ increases revenue by \$50.52	Mininet	×	×	✓	✓	×	✓	×	✓	✓	×
79	Logarithmic smoothing, G/G/1	Centralized	Tree	5 DCs; 100 application instances; 500 end-users reduced user delay around 300 ms	-	×	×	✓	×	×	×	✓	✓	✓	×

TABLE 4 (Continued)

Authors	Techniques	Controller	Topology	Setup description	Coding/ simulator	Issues considered						Parameters			
						1	2	3	4	5	6	7	8	9	10
80	RE-FPR algorithm	Centralized (POX)	Fat-Tree	$k(k+1)$ switches; k^2 hosts; 10 Mbps each link bandwidth saves energy upto 3-5%	-	✓	×	×	✓	×	×	✓	✓	✓	×
81	AGAR approach	Centralized (POX)	Fat-Tree	20 vswitches; 600 links; 1000 each vswitches table capacity; 20 Mbps each link bandwidth reduced delay by 9.5%	Mininet	×	×	×	✓	×	×	✓	✓	✓	×
82	MinCof, Coflow scheduling	Centralized	spine-leaf Fat-Tree	4 spine switches; 4 leaf switches; 16 hosts; 100 Mbps each link bandwidth reduces the completion time by 12.94%	-	×	×	✓	✓	×	×	✓	✓	✓	✓
83	OSPF	Centralized	Cost266, JANOS-US-CA topology	39 node; 61 links; 1432 flows- the average traffic loss is 0.82% and congestion links is 0.26% increases utilization upto 50%	-	×	✓	✓	✓	×	×	✓	✓	×	×
84	Optimal chunk routing	Centralized	Tree	10 DCs; 500 average chunks; 100 MB each chunk size reduced the delay around 10s	Java	×	×	×	✓	×	×	✓	✓	✓	×
85	Exclusive routing	Centralized	Fat-Tree, BCube	3456 servers; 512 MB flow size; 1 Gbps capacity; 1000/s average flow rate saves 35.02% and 26.56% energy	-	✓	×	✓	✓	×	×	✓	✓	✓	✓
90	Anomaly detection and mitigation	Centralized (NOX)	tree	Exp.1,2,3: (50, 100, 500) Mbps average traffic rate; (1/64, 1/64, 1/256) sampling rate; (50-200, 200-500, 1000-2500) attack rate pkts/s	Python	×	×	×	×	✓	×	×	✓	×	×
93	Brew security flow rules	Distributed (ODL)	Reingold-Tilford tree	10,000-1,00,000 flow rules; 1 GB flow size reduces latency by 5%	Mininet, Java	×	×	×	×	✓	×	✓	✓	×	×
94	Security defense patterns	Distributed (ODL)	Fat-Tree	k= 128: 540800 nodes; 1572864 links, execution time is 15.9 s and is 2.6 times faster	-	×	×	✓	✓	✓	✓	✓	×	✓	✓
99	FPTA algorithm	Centralized	Tree	12 DCs; 19 links; 100 VMs reduces latency by 50 %	Matlab	×	×	✓	✓	×	✓	✓	✓	✓	×
100	S-CORE algorithms	Centralized	Canonical, Fat-Tree	12 live VM migrations; 50% VM migration-throughput increases by 6 times and cost reduced upto 70%	Mininet	×	×	✓	✓	×	✓	✓	×	✓	✓

Note: 1: Energy Efficiency; 2: Fault Tolerance; 3: Resource Allocation; 4: Routing; 5: Security; 6: Virtualization; 7: Cost; 8: Latency; 9: Bandwidth; 10: Throughput; ✓: considered and ×: not-considered.
Abbreviations: DCN, data center networking; SDN, software-defined networking.

shown in Figure 12. Some of the major components in SD-FC and MEC architecture are energy management, resource scheduling, security, and virtualization are depicted in Figure 10. The forwarding devices at the data plane forward the traffic on the infrastructure plane. At the infrastructure layer, multiple DCs, including edge DCs, are located. The edge DCs reduce the network delay by providing the services to the end-users in a single hop. These detailed issues of SD-FC are discussed below.

4.2.1 | Energy-efficiency in fog computing

The resource-constrained nature of edge nodes makes it essential to design energy-efficient technologies and techniques in the SD-FC ecosystem. In this direction, Faruque et al[103] analyzed the problem of energy management in homes over SD-FC platform. The authors monitored the home power consumption and efficiently managed it by controlling the home appliances such as–EV chargers, heating, ventilation, and air conditioning (HVAC) management. The home energy control issue was resolved using three control-as-a-service (CaaS) schemes, namely, (i) smart HVAC CaaS, (ii) smart EV charger CaaS, and (iii) smart transformer CaaS scheme. These schemes do efficient energy management and control of home appliances, and the results show that power consumption decreased significantly. In another work, Aujla et al[104] discussed the issue of huge energy consumption involved due to inter-DC migration. The author considered the multi-edge cloud scenario and presented a workload slicing and scheduling algorithm to solve the data accelerated operations. In addition, the author proposed an energy-aware flow scheduling algorithm for taking intelligent decisions by the SDN controller for traffic engineering. Finally, a multileader, multifollower Stackelberg game is formulated to handle the issue of inter-DC migrations. Sarkar et al[105] worked on the problems related to power consumption, CO_2 emission, service latency, and cost in cloud DCN. The authors presented a fog computing model to serve the user demands using a mathematical approach. The results obtained prove that the overall service latency is reduced up to 50.09% with 50% appliances demanding requests spontaneously.

4.2.2 | Resource scheduling in fog computing

In an earlier study,[106] the author designed a modeling and simulation tool named as "*iFogSim*" to perform resource scheduling in fog environments. The *iFogSim* package consists of two modules, namely, (i) *cloud only placement*: in which deployment of the application modules runs in the DCs and (ii) *edge ward placement*: the deployment of the application modules runs at the edge of the network (like routers, access points). The *iFogSim* helps to measure the impact of resource scheduling in terms of latency, congestion, energy consumption, and overall cost. In another work,

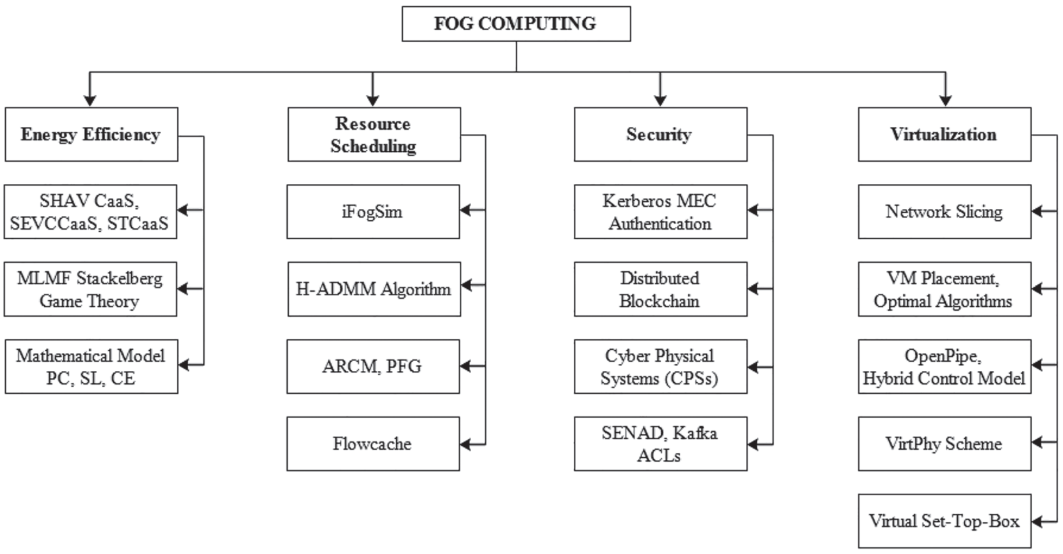

FOG COMPUTING
Energy Efficiency
SHAV CaaS, SEVCCaaS, STCaaS
MLMF Stackelberg Game Theory
Mathematical Model PC, SL, CE
Resource Scheduling
iFogSim
H-ADMM Algorithm
ARCM, PFG
Flowcache
Security
Kerberos MEC Authentication
Distributed Blockchain
Cyber Physical Systems (CPSs)
SENAD, Kafka ACLs
Virtualization
Network Slicing
VM Placement, Optimal Algorithms
OpenPipe, Hybrid Control Model
VirtPhy Scheme
Virtual Set-Top-Box

FIGURE 12 Software-defined networking (SDN)-based fog computing taxonomy

Yin et al[107] designed a distributed resource sharing algorithm named as hybrid alternating direction method of multipliers (HADMM). The designed scheme dynamically adjusts the massive application data by transferring the cloud data to the fog nodes for pre-processing and analysis. The authors formulated the problem to decide the amount of data processed by the fog nodes. The proposed scheme reduces the communication overhead by increasing the participation of fog nodes and updating the variables of fog nodes in parallel. Aliyu et al[108] presented an adaptive resource capacity management (ARCM) approach to optimize the resource capacity for real-time mobile edge computing applications. Moreover, a partition form games (PFG) approach in SDN-based intercloud architecture is adopted for capacity management and load balancing. Similarly, Ruia et al[109] addresses the problem of increased workload on the controller as the routers and switches generate frequent requests. The author presented a flow cache as a software-based cache for temporarily storing the contents of the recent flow table entries installed on the switches. The flow cache in SDN architecture acts as a proxy layer between the controller and switches, providing the controller with a large amount of flow table space. In another study,[63] the authors proposed an OF-switch load consolidation scheme built over a two-stage game for resource scheduling in an edge-cloud environment that achieves a substantial reduction in energy consumption and latency.

4.2.3 | Security in fog computing

Edge devices are more prone to security threats due to their distributed deployment. Hence, a security paradigm in a fog computing environment is required on the gateway nodes, controller, edge switches, and end hosts. In this direction, Rawat et al[22] presented a survey on the security attacks in SDN. This survey focused on spoofing, intrusion, anomaly, and DDoS attacks. An adversary can target the mobile cloudlet platform to breach the security in the SD-EC environment using any of the above-mentioned attacks. In another work, Chaudhary et al[110] focused on the DDoS attacks on the mobile cloudlet architecture. The cloudlet model or MEC has the capability to provision the cloud resources at the edge of the mobile network for mobile users. The authors presented a Kerberos authentication mechanism as a reliable authentication service to defend against DDoS attacks. In another research,[111] the author presented a secure distributed fog nodes model using blockchain techniques in SDN architecture. The fog nodes allow the deployment of services at the edge of the network, which improves the overall performance. The proposed model monitors the real-time attacks on the fog nodes, reduces the service latency and increases the throughput.

In an earlier study,[112] the authors discussed cyber-physical systems (CPS) with respect to the communication hijacking in mobile phones to laptop transmissions in fog computing. The authors focused on the issues of a man-in-the-middle attack launched by an adversary on fog devices (gateway). The authors also investigated the impact of CPU and memory consumption on fog devices due to the attack. Tseng et al[113] discussed serious security concerns about the deployment of malicious application injection on the controller's runtime. The malicious applications lead to the injection of app-to-control threats, which exhaust the network resources. The authors designed a secure network application deployment (SENAD) on the controller software. The proposed scheme partitioned the controller into the data plane controller for secure flow rules and the application plane controller to design secure applications for authentication, access control, authorization, and resource isolation.

4.2.4 | Virtualization in fog computing

The concept of overlay network decouples the network services from the underlying physical infrastructure. The overlay network is built on the top of the public Internet and encapsulates the packet to reach the endpoint network node, where the packet gets de-encapsulated. An example of overlay networks in SDN is content delivery networks, virtual private networks, voice over IP (VoIP) network, and peer-to-peer network. The protocols supporting overlay networks are VxLAN, Network Virtualization Overlays 3 (NVO3), stateless transport tunneling (STT), Network Virtualization using Generic routing Encapsulation (NVGRE), and GRE tunneling. Bruschi et al[114] worked on the issue of scalability in the fog computing environment. The author proposed a network slicing scheme to support multidomain fog services. The network slicing scheme provides the isolation of services into multiple slices. The proposed scheme offers high scalability by reducing the number of flow rules in the overlay networks. The proposed scheme is implemented in Matlab, and the unicast forwarding rules are installed in the overlay network and compared to the existing approaches (fully meshed and OpenStack approach). The performance of the overlay network shows that the number of unicast forwarding rules decreased by one order of magnitude or four times as compared to the existing approach.

Li et al[115] addressed the issue of VM placement in homogeneous and heterogeneous cloudlet mesh infrastructure. The authors considered the same maximum bandwidth capacity and VM slots for each cloudlet node in homogeneous cloudlet mesh. The authors designed an optimal algorithm to increase the amount of accepted VMs in the homogeneous cloudlet mesh. In the case of heterogeneous cloudlet mesh architecture, the authors considered each cloudlet's different number of VM slots and maximum bandwidth capacities. The authors presented another optimal algorithm to increase the amount of accepted VMs in the heterogeneous architecture. The results show that the limited bandwidth resources still cope with the growth of accepted VMs in the cloudlet mesh and an increase in VM slots. In a different work, Liang et al[116] designed a collaborative model of SDN and virtualized radio access networks (SDVRANs) in fog computing. The authors proposed a software-as-a-service scheme named "*OpenPipe*" (virtual resource chain) to perform network-level virtualization and network operations via network applications. Moreover, a hybrid control model integrates SDN and NFV in fog computing. The hybrid model consists of two levels, i.e., a higher level with an SDN controller located on the cloud layer and a lower level with local controllers located at the edge of the networks (fog sub-SDNs). Moreover, the scalability of the network is increased using a *FlowVisor*, which creates a virtual instance of the SDN and local controllers.

In,[117] the authors showed that the current DCs are not suitable for NFV orchestration services due to the issues such as–latency, throughput, and cost. The authors proposed a *VirtPhy* scheme, a fully programmable architecture for NFV orchestration in fog computing with the capability to deliver fast, innovative services in edge DCs. The proposed scheme uses a hypercube server-centric topology with testbeds implemented using the OpenStack cloud platform, resulting in improved service latency, cost, jitter, and throughput. In,[118] the author presented a virtual set-top box scheme by creating virtual entities to replace the physical set-up box placed at the user's home. The authors proposed a network softwarization approach to support personal cloud services in SDN and NFV architecture in fog computing. The evaluation proved that the designed approach achieves high QoS and low latency for static and dynamic users.

Table 5 shows the summarized existing proposals in SDN supported edge/fog computing environments.

4.3 | IoT

Software-defined Internet of things (SD-IoT) is a new computing paradigm which emerged to support interoperability in heterogeneous devices. The objective of IoTs is sharing of information globally based on the unique identity of each device. However, with the expansion of smart cities in modern real-life scenarios, billions of IoT devices are connected to the Internet. This leads to various issues such as energy management, failure recovery, resource allocation, routing, security, and virtualization. The taxonomy of the SD-IoT network based on these issues is shown in Figure 13.

Figure 14 shows the communication architecture of SD-IoT wherein all the application runs on an SDN controller. The control plane consists of a centralized SDN controller, which handles all multiple distributed controllers (depicted with the help of solid filled colors such as yellow, brown, blue, and violet). The distributed controllers are created using hypervisor software to design a fault-tolerant architecture. The cross symbol on the distributed controllers depicts a failure on the control plane. In such a case, the centralized controller smoothly accommodates the load of the failed controllers on the backup distributed controllers (brown and blue controllers) without any disturbance to the network operations. The hollow circles on the control plane are the core OF-switches, while on the data plane are the edge OF-switches. The edge OF-switches are directly connected with the infrastructure plane. The infrastructure layer comprises access points and multiple IoT domains. Each IoT domain consists of wireless sensors (smart home sensors, smartphones, medical sensors, etc.) for data collection, transmission, and computation. These IoT devices perform data sharing through SDN planes/infrastructure and face tough challenges concerning energy efficiency, failure recovery, resource allocation, routing, security, and virtualization.

After analyzing various factors, Table 6 shows the summarized overview of the existing proposals on SD-IoT. The detailed overview of the SD-IoT architecture and associated application is discussed as below.

4.3.1 | Energy-efficiency in IoT

The cloud DCs collect the data from IoT sensors using transmission networks to perform data analytics. But the transmission network handles the service latency for which a lot of energy is consumed by the sensors. In this direction, Hu

TABLE 5 Related proposals on fog computing using SDN architecture

Authors	Techniques	Dataset	Controller	Setup description	Coding/ simulator
103	SHAC CaaS, SEVC CaaS, ST CaaS	-	Centralized	Raspberry Pi (gateway router); Tiny OS-based TelosB mote sensors (connection & computation on fog nodes)	Java
104	Stackelberg game theory	Google workload traces	Distributed	1000 jobs; 3 Exp.: (1) cloud DCs, (2) edge devices, (3) edge cloud interplay. Saves energy consumption and reduces delay	Matlab
106	iFogSim	-	Centralized	Intelligent surveillance sensor: 4 fog devices; 1000 million instructions; 20,000 byte network length; the avg. inter-arrival time is 5 ms	CloudSim
107	H-ADMM algorithm	-	Centralized	Large Fog nodes: 100 fog nodes reduced communication overhead on controller	Matlab
108	Partition form Games	-	Centralized	VMs: 4096 MB; edgeIoT application: 75000; FogDCs: 4; 3,00,000 mips the service latency is less than 0.18s	CloudSim
109	Flow Cache	CAIDA traces	Centralized (Ryu)	200,000 packets; 22 Mbps throughput generate 24,500 individual flow. Latency: (controller & switch channel) is 4 ms; (switch channel)	Python
111	Distributed Blockchain	real-testbed	Distributed	6000 hosts; 2000 flow rate modules/s the execution time is 100 μs	TFN2K tool, Mininet
112	CPSs	-	Centralized	(Build video tunnel on gateway) Memory Consumption: increased from 15232 KB to 15324.8 KB; CPU Consumption: increased from 16.67% to 17.92 % (almost negligible)	-
113	SENAD, Apache Kafka ACLs	-	Centralized (floodlight)	10,000 packet_in; 1-10 sandbox (docker container) network application takes processing time 2.5946 ms-13.20 ms	Mininet
117	VirPhy	OpenStack Cloud	Centralized (Ryu)	Add 2 extra hops: 8 server nodes; 1 NFV orchestrator; 1 network node; 1 controller; 1 Gbps link capacity. Increases jitter from 0.06 ms to 0.08 ms; packet loss is at most 1.5% (negligible change)	Python, Mininet
118	Virtual setup-box	Real testbed (smart home)	Distributed	7 DCs nodes; 4 edge node; 6 users home network; 10 Mbps-30 Mbps avg. bit rates. Provide network latency of 20 ms (short range) and 150 ms (long range)	Java

(Continues)

TABLE 5 (Continued)

	Issues considered						Parm. improved		
Authors	1	2	3	4	5	6	7	8	9
103	✓	×	×	×	×	✓	×	×	×
104	✓	✓	✓	×	✓	✓	✓	✓	×
106	✓	✓	✓	×	✓	✓	✓	✓	✓
107	×	×	✓	×	×	✓	×	×	×
108	×	×	×	×	×	✓	✓	×	×
109	×	×	✓	✓	×	✓	✓	✓	✓
111	×	×	×	✓	×	✓	✓	×	✓
112	×	×	×	✓	×	✓	✓	✓	×
113	×	×	✓	✓	×	×	✓	×	✓
117	×	✓	✓	×	✓	✓	✓	✓	✓
118	✓	✓	✓	×	✓	✓	✓	✓	×

Note: 1: Energy Efficiency; 2: Service Migration; 3: Resource Scheduling; 4: Security; 5: Virtualization; 6: Cost; 7: Latency; 8: Bandwidth; 9: Throughput. Notations- ✓: considered, and ×: not-considered.
Abbreviation: SDN, software-defined networking.

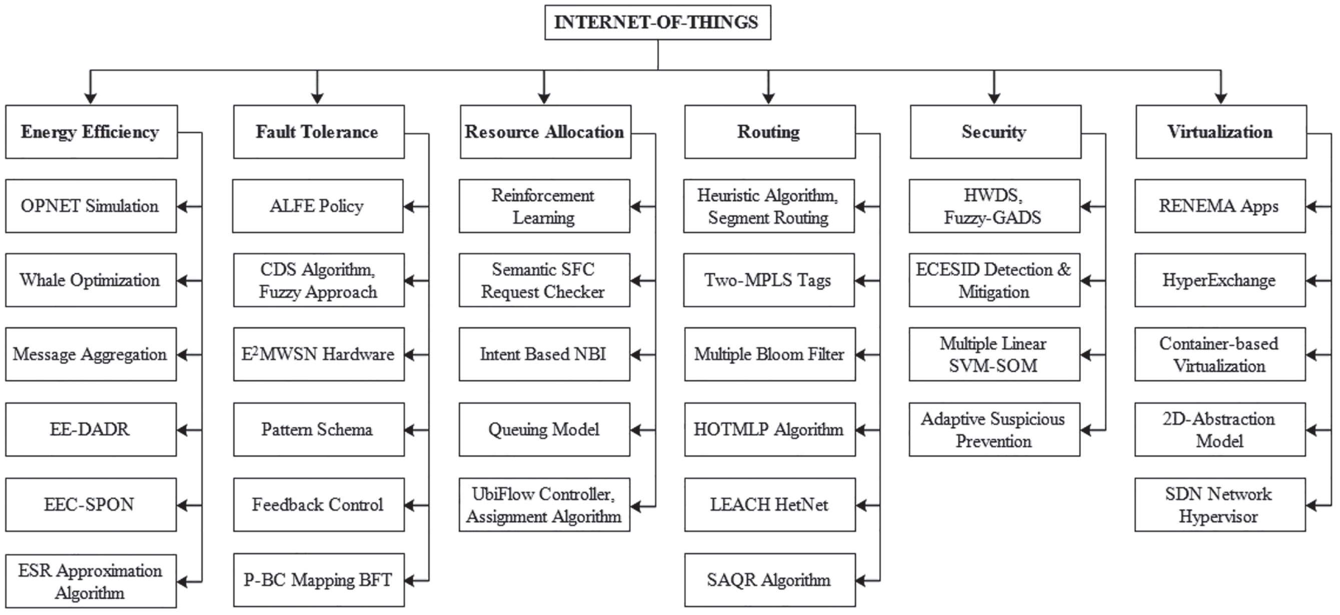


FIGURE 13 Software-defined networking (SDN) in internet of things taxonomy

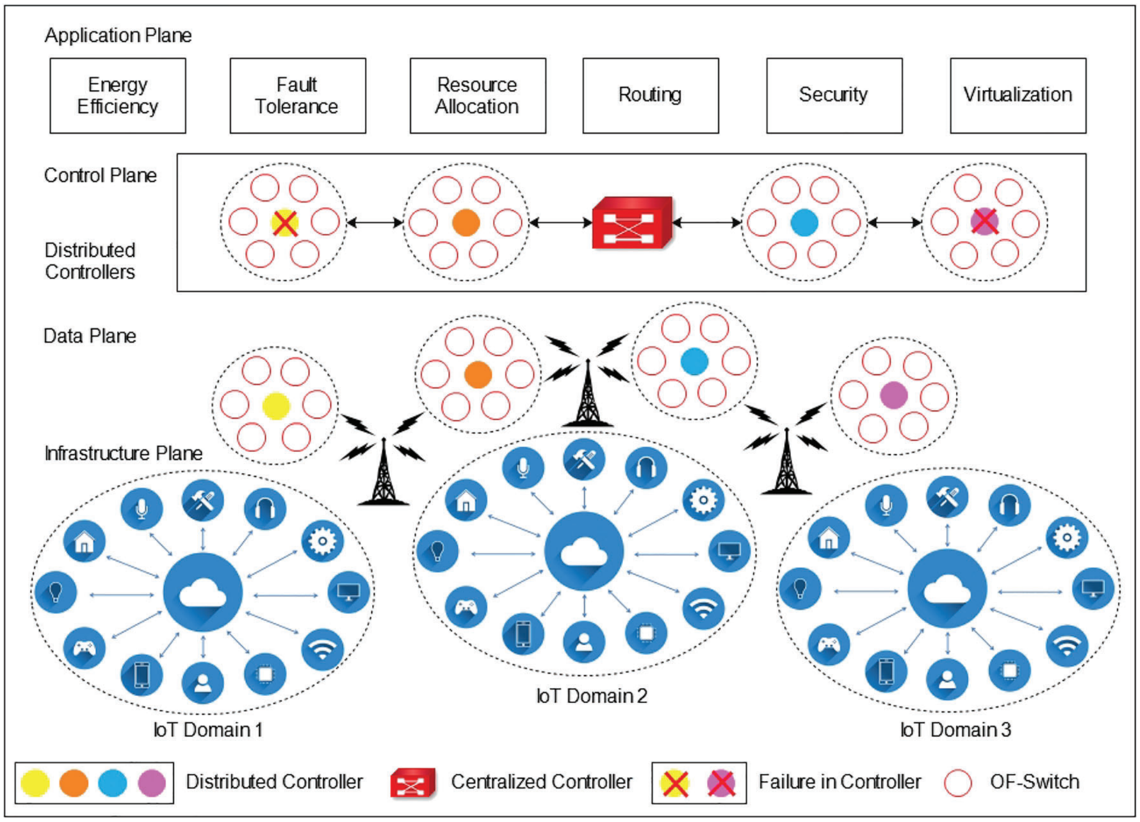


FIGURE 14 Communication architecture of software-defined internet of things (SD-IoT)

et al[119] addressed the issue of energy efficiency in IoTs and designed a hardware-in-the-loop emulation. The hardware-in-the-loop is a cost-efficient real-time emulation model based on OPNET simulation to design and analyze the errors in the complex testbeds. The module to implement the emulator is the system-in-the loop (SITL) interface. The designed simulator is compared with the SITL emulator. The results obtained show that the OPNET simulation is

TABLE 6 Relative comparison of proposals on SD-IoT

Authors	Techniques	Controller	Setup description	Coding/ simulator	Issues considered 1	2	3	4	5	6	Parameters 7	8	9	10	11
119	Hardware-in-the-loop emulator	Centralized (NOX)	Area: 375mx375m; 140 sensors; 2 Mbps data rate; 40-70m sensor range; 5W source node energy reduces energy consumption and reduces delay by upto 0.1s	OPNET	✓	×	×	✓	×	×	✓	✓	×	×	×
120	EE-DADR scheme	Centralized	Area:1000mx1000m; 2 Mbps data rate; 300 mW power of each node; 250 m radius; 50 nodes generates 25 packets/s of 25 kb each takes the execution time of 400s	OMNET++	✓	×	×	✓	×	×	✓	✓	✓	✓	✓
121	ESR scheme	Centralized	33 switches; 41 links; 1 Gbps data rate; 5 mW processing power; 200 mW transmission power; 1200-2000 sensors saves energy by upto 56%	-	✓	×	×	✓	×	×	✓	✓	×	×	×
122	CDS, fuzzy-logic	Centralized (multi-threaded)	Area:100mx100m; 49 nodes; 1 gateway node; 0.01 node density of each node increases the node reliability	Matlab	×	✓	✓	×	×	×	×	×	×	×	✓
123	Pattern schema	Distributed	6 switches; 2 hosts; 6 pattern topology; 9 links achieves flow completion time of 1.3s and with redundancy degree of 3	Mininet	×	✓	×	×	×	×	×	×	×	×	✓
124	P-BC mapping	Distributed	Controllers: f+1 primary; f backup reduces full-backup controllers upto 40% and partial controllers by 50% and reduces 50% controller load	-	×	✓	×	×	✓	×	✓	✓	×	×	✓
125	Reinforcement learning	Centralized (multi-threaded)	8 switches; 40 hosts achieves the overall delay as 0.1ms	Mininet	×	×	✓	×	✓	×	×	✓	×	×	✓
126	Semantic SFC checker	Centralized	20 SFC requests; each SFC has 5 VNFs takes the execution time of 0.85s to map the requests and for 512 nodes; 100 SFC requests; the mapping time is reduced by 50%	Java	×	×	✓	×	×	✓	✓	✓	✓	×	✓
127	Intent-based NBI	Distributed (ONOS)	standard deviation is 0.28ms achieves average latency of 31.7 ms and reduces the overall latency by 0.3ms	Mininet	×	×	✓	✓	×	✓	✓	✓	×	×	✓
128	Queuing model (M/M/m)	Centralized	20,000-45,000 million instruction/s; 100-800 requests; 5 virtual sensor reduces the response time by 20s	CloudSim	✓	×	✓	×	×	✓	✓	✓	✓	×	×
129	UbiFlow assignment algorithm	Centralized (multi-threaded)	3 servers; 3 switches; 1 Gbps link capacity of each switch; 20 access points; 100 Mbps Ethernet link to switches; 30 flows increases the avg. throughput by 42.24% and reduces latency by 62.07%	OMNET++	×	✓	✓	×	×	×	✓	✓	✓	✓	✓
130	Segment routing	Centralized	60 nodes; 201 links; 1 Gbps link capacity; 10-100 Mbps flow size; 100 requests improves throughput and rejection rate	Java	×	×	✓	✓	×	×	✓	×	✓	✓	✓

TABLE 6 (Continued)

Authors	Techniques	Controller	Setup description	Coding/ simulator	Issues considered						Parameters				
					1	2	3	4	5	6	7	8	9	10	11
[131]	LEACH protocol	Centralized	Area: 920mx800m; 50 nodes; 10J initial energy; 512 packet size improves the E2E delay	ns-2	✓	×	×	✓	×	×	✓	✓	×	✓	✓
[132]	SAQR algorithm	Centralized (multi-threaded)	4 core switches; 4-16 edge switches; the execution time is 3.62ms. Reduces delay and packet loss by 88% and increases bandwidth by 86.5%	Mininet	×	×	✓	✓	×	×	✓	✓	✓	✓	✓
[133]	HWDS, fuzzy-GADS	Centralized	Exp.1, Exp.2, Exp.3, Exp.4: (512, 1024, 2560, 5120) hosts-mitigate the malicious hosts and achieves accuracy more than 98%	Scorpius	×	×	×	×	✓	×	✓	×	✓	×	✓
[134]	Mirai-botnet edge centric	Centralized	7 IoT devices; 30 wired hosts; 5s queue update interval; 10000 flow capacity; 250 ms timeout takes 6.02 s	Mininet	×	×	×	×	✓	×	✓	✓	×	✓	×
[135]	Multiple linear SVM-SOM	Centralized (POX)	CAIDA datasets; 4000 flows; 30s hard timeout computes detection rate; accuracy; and false alarm rate based upon TP, TN, FP, FN as 96.03%, 98.17%, 3.97%, 1.83%	-	×	×	✓	×	✓	×	×	×	✓	✓	✓
[136]	Adaptive suspicious prevention	Centralized (POX)	5,957 flows; 14.58 average packets/flow; increases bandwidth from 1234 Kbps to 1,263 Kbps in switches-controller channels. Reduced flow entries by upto 38.23% and reduces packet arrival per second by upto 36.17%	Mininet	×	×	✓	×	✓	×	✓	✓	✓	×	✓
[137]	Optimal controller placement	Distributed	34 nodes; 41 edge switches; 2 hypervisors; 17 multi-controller switches; Area: $5.10^3 - 2.9.10^3$ km reduced the overall latency by 42%	Python	×	×	✓	×	×	✓	✓	✓	×	×	×

Note: 1: Energy Efficiency; 2: Fault Tolerance; 3: Resource Allocation; 4: Routing; 5: Security; 6: Virtualization; 7: Cost; 8: Latency; 9: Bandwidth; 10: Throughput; 11: Packet Loss; ✓: considered, and ×: not-considered.
Abbreviation: SDN, software-defined networking.

energy-efficient, and the delay generated by OPNET is around 0.1 s, while the delay incurred in SITL emulation is around 0.4–0.5 s. In another work, Al-Janabi et al[138] focused on energy efficiency in software-defined wireless sensor networks (SDWSN). The author identified two major problems in sensor nodes, that is, resource limitations and arbitrary diversification in node density of regional location. The authors solved both issues and presented a new clustering scheme named a whale optimization algorithm. The scheme partitions the complete geographical area into multiple virtual zones (VZs) to maintain the equilibrium of cluster heads in each VZs. The experimental results proved that the proposed algorithm is energy-efficient and packet reception improved by 55% and 20% compared with the various existing clustering and routing protocols, respectively.

In a different work, Wang et al[139] mitigated the issue of trust management concerning two aspects, i.e., energy management and efficient routing. The proposed scheme detects malicious nodes in the SDWSNs. The author presented a centralized trusted approach on the SDN controller to monitor and decouple the malicious nodes. Moreover, a message aggregation approach and trusted routing mechanism are designed to monitor and report network attacks. The simulation results prove that the proposed scheme defends against malicious threats such as Greyhole and Blackhole attacks. Finally, the scheme improves the packet delivery ratio with low control overhead and reduces energy consumption compared to the existing approaches. Similarly, Wen et al[120] addressed the issues of energy management and E2E delay constraint in routing. The author presented a common algorithm, that is, energy-efficient and delay-aware distributed routing (EE-DADR), to solve both issues simultaneously. This scheme is compared with the existing approach, that is, greedy perimeter stateless routing (GPSR). The experimental results show that the EE-DADR consumes less energy and reduces the E2E delay than GPSR.

With the advancement in high bit-rate applications, software-defined passive optical network (SPON) technologies plays a crucial role in the broadband services in the 4G/5G network. The SPON architecture is designed by two network modules, that is, optical line terminal (OLT) and optical network units (ONUs). Zhao et al[140] analyzes that the SPON provides a high bit rate to the end-users but still the crisis of low bandwidth and the high-energy consumption is a critical issue. The author developed an energy-efficient control scheme by collaborating the optical access networks and metro network resources. The designed scheme improves the energy consumption in SPON and achieves high bandwidth utilization. In another study,[121] the author observed the problem of big data processing in SD-IoT. The author formulates an optimization problem called sensor data selection, data processing, and routing to reduce the overall energy consumption in SD-IoT. The proposed scheme performs energy-efficient sensor selection and routing (ESR) based on $\alpha log|K|$ approximation algorithms. The proposed scheme successfully finds an optimal sensor selection with reduced energy consumption in SD-IoT up to 56%. In another study,[141] the authors proposed power optimization and secrecy association algorithms to resolve the issues of high power consumption and secure data sharing in software-defined Internet of Vehicles environment. The proposed solution uses an energy harvesting scheme to improve energy efficiency and maximize the throughput compared with the existing schemes.

4.3.2 | Fault tolerance in IoT

SD-IoT is a distributed deployment of sensors controlled by central network management software. Therefore, it faces a very tough challenge from a single point of failure concerning centralized controllers. In a research,[122] the authors focused on the reliability of the SD-IoT network considering connectivity rate, average link quality, hop counts, and resource management. The author introduced a dynamic approach in SD-IoT to minimize the control overhead in the selection mechanism of network resources. The concept of distributed controllers enables the placement of primary, secondary, and local controllers to control the network nodes. The selection strategy of the number of local controllers is an important concern. The appropriate scheme in connected dominating sets (CDS) based on a fuzzy approach helps compute the relevant number of local controllers required to execute resources smoothly. Shi et al[142] primarily focused on the robustness and reliability in SDWSNs. The author designed an energy-efficient and fault-tolerant multicore architecture (EE-MWSN) to achieve low-energy consumption compared with the WSN platform (TinyOS and TelosB). In another study,[123] the author presented a pattern framework as a prototype model for SDN layered architecture. The framework consists of a pattern schema (specification and language) located at the application layer to modify the network topology. The pattern language develops the pattern rules to be executed on the controller. These pattern rules are installed on the forwarding devices to detect link failure and faults.

In a distributed controller environment, the issue of cascading failure generally occurs when a specific controller gets failed and then the network is overloaded and leading to packet loss. Hosny et al[143] worked on the issue of cascading failure problem. The fault tolerance in SDN is handled by using a feedback control based on the LB approach. Using a feedback control policy, the proposed scheme minimized the cascading failure problem on the distributed control plane. A feedback control policy broadcasts the report to all the distributed controllers and helps to automatically migrate the load to the neighbor controllers. The performance of the proposed approach helps to reduce the packet loss by up to 14% and minimize the packet delay by up to 17%. Although multiple controllers are deployed to achieve fault tolerance in the network, at the same time, security and the mapping from the switches to controllers and vice-versa is a major concerns. Mohan et al[124] identifies the issue of controllers overhead and byzantine threats on control messages when an individual switch is mapped to $3f+1$ controllers. The authors presented a primary and backup switch-controller (P-BC) mapping approach in which an individual switch is mapped to $f+1$ primary controllers and f backup controllers. The presented approach helps to mitigate byzantine attacks on f backup controllers. In addition, the proposed approach calculates the minimum number of controllers required with each controller capacity in order to avoid delay. The results obtained are compared with the traditional approach, which proves that the scheme minimizes the overall controller's requirement by up to 50% and improves the network load within controllers with a fairness index up to 0.92. Moreover, the SDN architecture faces the congestion problem due to limited memory capacity for storing the flow tables in the switches. However, the elephant flows (huge flows) over the network can be handled using an efficient cache mechanism to prevent the overflow of massive mice flows (tiny flows). Pan et al[144] computed a solution by designing an optimal flow cache policy named Adaptive Least Frequently Evicted (ALFE). The proposed approach is compared with the least recently used (LRU) cache placement policy and is tested over 1000 cache entries. The dataset traces are collected from CAIDA, and the simulation results show that the ALFE approach increases the cache hit by 15% compared with the LRU approach.

4.3.3 | Resource allocation in IoT

In highly dense smart cities, huge volumes of data are generated from smart sensors. Hence it becomes difficult to accomplish service requests to multiple service providers within a short period. The requirement of resource allocation in the SD-IoT network is mandatory. Munir et al[125] use the concept of reinforcement learning (RL) approach and presents an intelligent agent service fulfillment algorithm to resolve the issue of delay incurred through multiple service requests. The presented scheme is implemented in a fog-based controller unit and achieves high performance by reducing the processing and waiting for the delay. The proposed approach achieves the time complexity of $O(n^2)$, and for each sub-problems, the time complexity of $O(2^n.n)$. The results show that when the delay is 0.10 ms, it maximizes resource utilization. Bouten et al[126] focused on the issue of limited resource capacity constraints to map VNFs with the physical infrastructure and fast routing of data transmission in virtual paths. The author finds an optimal solution for mapping the VNFs or SFC requests in a reasonable time. They proposed a semantic SFC validation model to decrease the mapping time of service requests. The simulation results show that with the semantic matching on individual mapping of 100 service requests, the mapping time is reduced to 50%. Furthermore, considering the mapping of 20 SFC requests where each SFC has 5 VNFs, the average execution time is 0.85 s. The scheme significantly saves the mapping time of an algorithm.

Cerroni et al[127] analyzed the problem of low reliability and high latency generated due to an increase in the SFC requests. The author presented a reference multidomain SDN/NFV architecture by creating a service chain of network functions running on the hardware platforms and enhancing flexibility in service deployment. Additionally, an intent-based NBI architecture is presented in SDN for service orchestration. The average response time of the proposed model achieves high reliability with an average latency of 31.7 ms with 95% confidence intervals and minimizes the average latency up to 0.3 ms. In another work, Banaie et al[128] discussed data sharing by adopting virtualization to create virtual sensor resources on cloud platforms as an instance of physical sensors. The author used a queuing model to develop a scheduling approach to balance the load of virtual sensor resources. The proposed approach distributes the load among virtual resources and decreases the average completion time by around 20 s. However, in addition to resource allocation problems, ubiquitous flow control and device mobility is also important concern in the SD-IoT network. Wu et al[129] presented a *UbiFlow* control mechanism to address the issue of mobility. The designed approach partitioned different geographical locations and deployed distributed controllers. The *UbiFlow* controller examined the flow scheduling in each partition, and the results prove that the average flow transmission delay is less than 0.4 s for mobile devices. The performance of the new scheme is robust inflow scheduling and achieves stability in mobility management.

4.3.4 | Routing in IoT

The network performance can be improved by designing an efficient routing scheme in SD-IoT. Lee et al[130] design an efficient heuristic scheme for segment routing (SR) by an ordered MPLS approach to cope with the bandwidth requirements. The performance of the designed algorithm shows an improvement in the average rejection rate and average network throughput. Kitsuwan et al[145] addressed the issue of overflow in OF messages generated at the controller. The proposed scheme is two-MPLS tags to reduce the flow scheduling. The tags or labels are generated on each network packet to route the packets from source to destination. The first tag is labeled to escort the network packet from the source switch to the edge switch of a destination class, and the second tag is labeled to escort the packet from the edge switch to another switch of a local class. The experimental results show that the scheme reduces 81% of the flow messages and 74% reduction in the number of permanent flow entries compared to the existing schemes. Challa et al[146] proposed the issue of insufficient flow table memory and computed a solution for fast flow forwarding. The proposed scheme uses a probabilistic data structure named multiple bloom filter (MBF) as a space-efficient technique. The proposed approach is designed column-wise to store the flow entries in SRAM rather than TCAM and returns the error probability of less than 1%. The results are compared with the LRU approach and reduce the lookup times up to 37% with a flow table capacity of 2000 entries.

Zhijie et al[147] focused on the similar issue of memory shortage of flow tables and the routing flow forwarding speed. The author presented a scheme called a hash offset tree match on the longest prefix (HOTMLP). The scheme performs packet lookup and fast-forwarding in the SD-IoT network. The scheme applies to hash on the longest prefix rule matching to reduce the storage space, and the exact prefix matching is selected based on the tree approach. The results prove that the designed algorithm performs a faster lookup with space complexity of $(O(2+n/k))$. In addition, the presented routing method forwards around 10 million packets per second. In another study,[131] the author presented a low-energy adaptive clustering hierarchy (LEACH) heterogeneous routing protocol to achieve both energy efficiency and optimal routing. The routing scheme is simulated in NS-2 by considering a network of 50 nodes with an initial energy of 10 joules, and a trust value is 5. The designed scheme was compared with the existing routing methods, that is, ad-hoc on-demand distance vector (AODV) and dynamic source routing (DSR) and performs better performance in terms of energy and routing by reducing E2E delay. Lin et al[132] identify the issues of packet loss, delay, and bandwidth in existing routing protocols. The author proposed an efficient dynamic routing algorithm, that is, simulated annealing based QoS-aware routing (SAQR), to compute the optimal fit path using the cost function. The SAQR algorithm is compared with the layered controller approach's multinetwork information architecture (MINA). The results show that the proposed scheme achieves better fitness ratios with delay reduced to 88%, packet loss reduced to 90.8%, and bandwidth requirement of packet flows improved by 86.5%.

4.3.5 | Security in IoT

With the rapid deployment of wireless sensors in smart cities, device-to-device communication is more susceptible to security threats. As a single controller is handling the complete network infrastructure, the chances of malicious DDoS attacks may infect the heterogeneous IoT devices.[23] Hence, it becomes mandatory to provide defense mechanisms so that the controllers establish resilient and reliable connections with the switches. Gharaibeh et al[148] analyzed various data management and security techniques deployed in smart cities' SDN-IoT environment.

Assis et al[133] addressed the issue of DDoS attack as an enormous malicious request sent on the controller, which degrades the overall network performance. The designed approach mitigates the DDoS attack using game theory and Holt-Winters for digital signature (GT-HWDS). Moreover, the author uses fuzzy logic and genetic algorithms for digital signature (Fuzzy-GADS) to identify and mitigate suspicious behavior of the requesting nodes. The performance of the designed scheme shows that the alarms get triggered automatically on the controller by the GT-HWDS and Fuzzy-GADS algorithm whenever an anomaly is detected. In another study,[134] the author presented an architecture regarding the issue of Mirai-botnet DDoS attack (malware) infecting IoT devices. The defense mechanism is the edge-centric SDN-IoT framework. The presented framework performs two tasks, namely, (a) malware detection and mitigation of infected nodes and (b) flow handling algorithm on the controller. The simulation results show that the designed algorithm efficiently detects and mitigates the infected node with an average detection time of 6.02 s.

Phan et al[135] addressed a similar issue of DDoS attack in the SD-IoT network by using a hybrid model. The author presented a novel hybrid flow handler mechanism based on two classification modules, that is, support vector machine

(SVM) and self-organizing map (SOM). The various linear SVM initially classifies the flow entries in the OF-switches. The linear SVM representation checks whether the flow entries are present between two margin lines. If found, forward those entries to the SOM method for further judgment. The SOM module makes the final judgment of the suspected patterns based on the weights computation of neurons. The experimental results show that the proposed hybrid model attains better accuracy in terms of true positive (TP), true negative (TN), false positive (FP), and false-negative (FN) errors. Dao et al[136] presented an adaptive suspicious prevention (ASP) mechanism to prevent the SDN controller from DDoS attacks. The proposed technique is embedded with the OF protocol to protect the DDoS attacks on the switches. The performance analysis shows that the designed technique can defend from security attacks by up to 38%. In another study,[149] the authors worked on the issues of secure data distribution and transmission of information for the social Internet of Vehicles. To address this issue, the authors proposed a content-centric data dissemination scheme and secure data authentication mechanism using lattice-based cryptography for secure communication in the IoT ecosystem.

4.3.6 | Virtualization in IoT

With the advent of wireless sensors, the drive for virtualization is required in the cloud with the emergence of technologies like SDN and NFV. Virtualization can dynamically allocate the resources on the applications whenever required. Moyano et al[150] addressed the issue of manual configuration in residential networks. The author solved the issue of residential networks based on the NFV and SDN network-centric approach by configuring the residential gateways through network programmability. The author developed a residential network management application (RENEMA apps) on top of the SDN architecture as an application layer to control the network traffic and interact with the user devices. The presented framework improves the overall throughput of the home area network (HAN). In another study,[151] the author identifies the problem of scalability and extensibility in SDN-NFV architecture. The author designed a HyperExchange protocol on the control plane to offer an exchange of services in interdomain virtual networks. The protocol provides flexibility in peering and, in addition, supports authentication and authorization through remote APIs for interdomain tenants.

Drutskoy et al[152] also addressed the issue of scalability in SDN-NFV architecture. The author explored *FlowN* architecture having the ability to modify the mapping between the physical and virtual forwarding devices. Also, the author highlighted the benefits of container-based virtualization to minimize the virtual network mapping overhead of deploying multiple controller applications. The *FlowN* architecture provides the mapping between the virtual and physical resources as each tenant has an illusion of its own available address space, network topology, controller application, and database. Duan et al[153] presented a 2-D abstraction model of layer and plane architecture of SDN-NFV. The architecture discussed the layer dimension abstraction consisting of five layers: the physical, network interface, Internet, transport, and application layer. The plane dimension abstraction shows three SDN planes-data, control, and management. Blenk et al[137] examined the issue of high latency overhead incurred on the control plane due to the existence of multiple virtual controllers. They formulated the problem based on mixed-integer programming to optimize the placement of virtual controller or network hypervisor instances. The authors perform the trade-off strategy for optimal controller placement of virtual controllers so that maximum nodes are served. The overall network latency is reduced up to 42% by covering the distance of 5.10^3 km to $2.9.10^3$ km by virtual controllers.

4.4 | WBAN

WBAN has become a hotspot for remote medical facilities at home through smart sensors. Using wearable devices, clinical facilities, remote monitoring, and diagnosis services are provided to the ailing patients at their homes rather than visiting hospitals. The healthcare industry focuses on various parameters such as data collection through wireless sensors, data transmission via high-speed Internet, secure data storage in the cloud, and data analytics in data centers. Using the underlying network technologies like SDN, NFV and 5G, the healthcare industry is improving rapidly to ease the living standards of the users. The human-to-machine interaction between the users and doctors is possible using wearable devices and is expected to improve healthcare services through regular monitoring. In this direction, the software-defined wireless body area network (SD-WBAN) offers the best QoS to the end-users, but still, some issues affect the overall network performance. The major issues in SD-WBAN are routing, security, resource scheduling, and mobility.

Figure 15 shows the hierarchy of the communication architecture of SD-WBAN. The architecture consists of six layers divided into four planes. Layer 1 comprises rural clinical centers, doctors, patients, access points, and edge DCs. The wireless wearable sensors, connected to the patient, are monitored by the doctors sitting at a remote location. At layer 2, the OF-switches perform data forwarding. The sensitive medical data are shared securely between layers 1 and 3 through layer 2. The urban clinical centers are located on layer 3 and are interconnected with each other through multiple cloud DCs. The distributed controllers and cloud DCs reside on layers 3 and 4, which helps to interconnect urban clinical centers with the regional clinical centers. Next, the root node is at the top of the hierarchy, where the centralized controller resides. The root node is the headquarters of all the clinical centers. The centralized SDN controller monitors the global database and controls the national clinical center. Finally, layer 6 comprises network applications of SD-WBAN. Various issues related to SD-WBAN are resolved at the application plane by the programmer and deployed on the root node of the hierarchy. The root node modifies the dynamic traffic flow table policies per the network situations. The taxonomy of SD-WBAN is shown in Figure 16. The detailed discussion of the related work is discussed in the subsequent subsections.

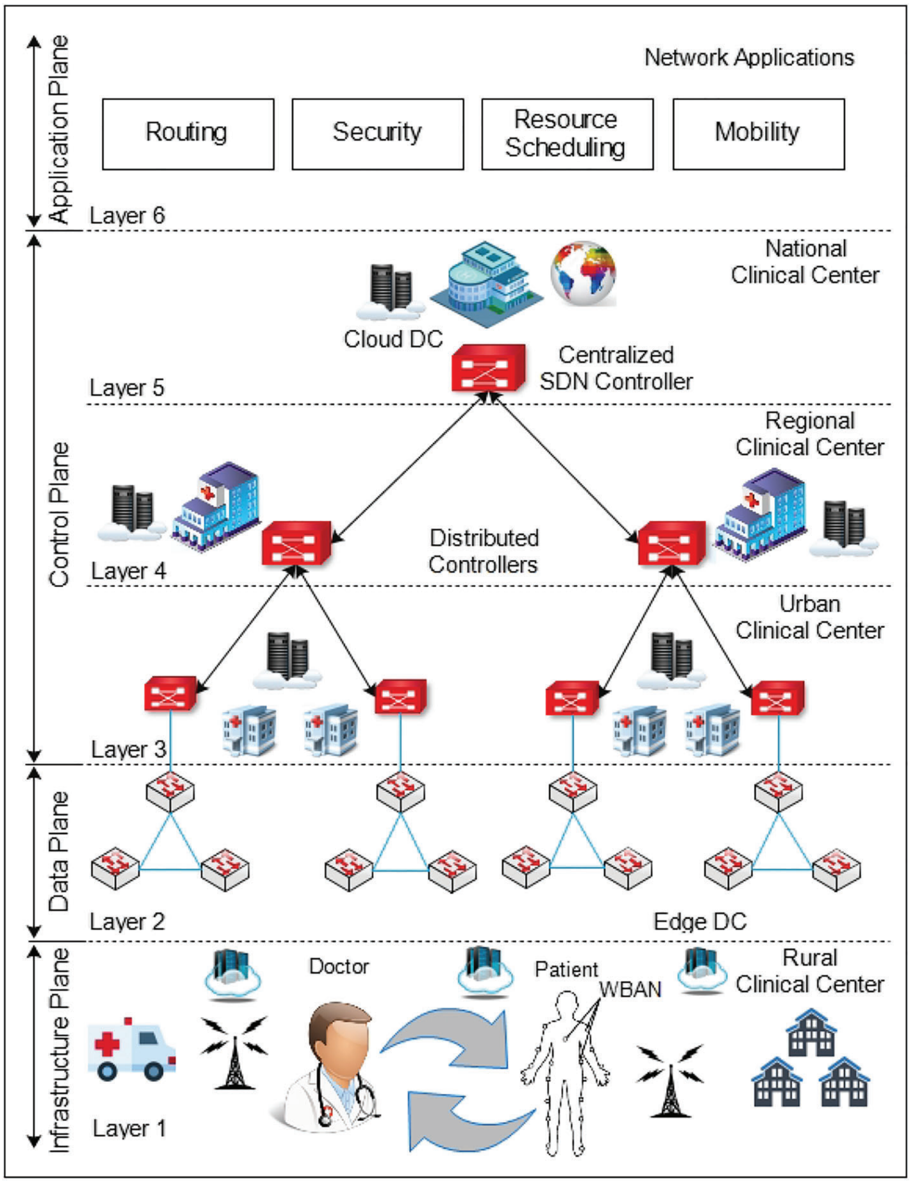


FIGURE 15 Communication architecture of software-defined wireless body area network (SD-WBAN)

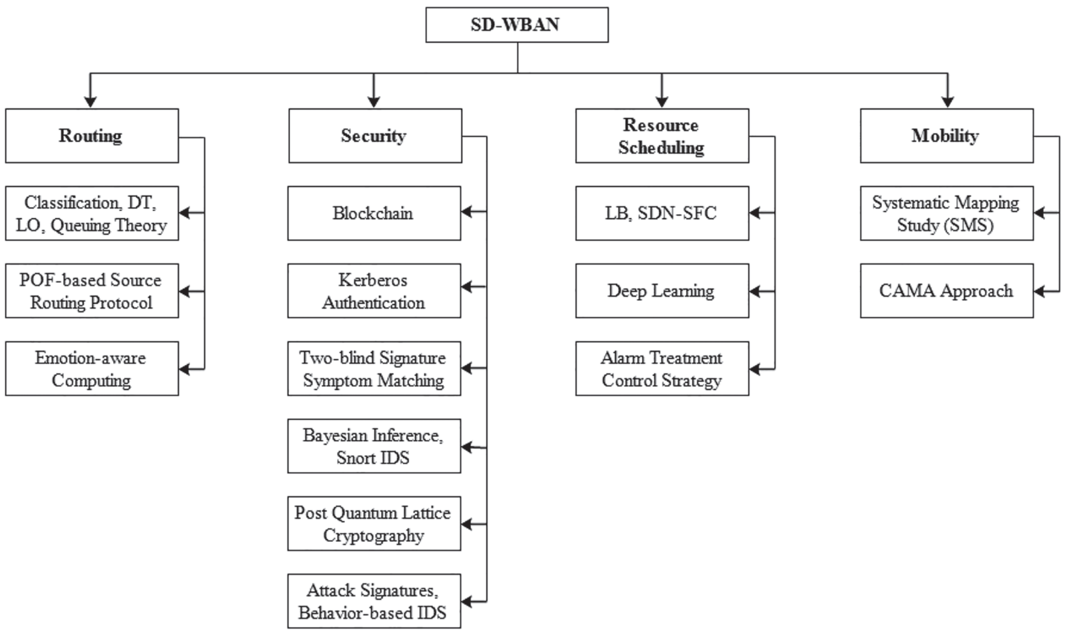

SD-WBAN
Routing
Classification, DT, LO, Queuing Theory
POF-based Source Routing Protocol
Emotion-aware Computing
Security
Blockchain
Kerberos Authentication
Two-blind Signature Symptom Matching
Bayesian Inference, Snort IDS
Post Quantum Lattice Cryptography
Attack Signatures, Behavior-based IDS
Resource Scheduling
LB, SDN-SFC
Deep Learning
Alarm Treatment Control Strategy
Mobility
Systematic Mapping Study (SMS)
CAMA Approach

FIGURE 16 Software-defined networking (SDN) in healthcare network taxonomy

4.4.1 | Routing in SD-WBAN

In smart communities, most people are health conscious, so healthcare monitoring (e.g., health application on iPhone) is performed by deploying intelligent sensors in smart homes for data collection and transmission. The collected data are transmitted to the cloud for data analytics and stored and processed through the cloud service providers. The major issue faced in the SD-WBAN is high latency as multiple requests are processed simultaneously, which may saturate the SDN controller. To handle this issue, Aujla et al[50] presented an integrated approach for dynamic traffic flow management. This approach is divided into three phases: (a) a dependency removal method is presented on the incoming requests at OF-switches, (b) an application-specific packet classification method is adopted on the controller by using a decision tree in order to classify the application features, and (c) the congestion and starvation problem is handled in classified traffic flow on the controller by using priority shifting (PS) scheme based on M/G/1 queuing model. The designed PS scheme is compared with the first-in-first-out (FIFO) scheme. The simulation results show that the average waiting time is reduced along with achieving high bandwidth provisioning.

In another work,[154] the authors utilized virtualized resources for handling the incoming data flow from healthcare devices. The authors achieved lower latency and higher throughput using edge DCs to serve patients' requests. Similarly, Li et al[155] addressed the issue of high latency and scalability during data transmission of e-healthcare in software-defined wide area networks (SD-WAN). The author presented a scheme named protocol-oblivious forwarding (POF) based source routing protocol. The proposed protocol processes each packet by replacing the conventional approach, that is, the table lookup process, with a new approach, that is, table pipeline processing, for efficient routing on SDN switches. The performance of the designed scheme is better than the conventional approach in large scale WAN because the controller configures only the ingress and egress edge switches. In contrast, the controller has to configure all the switches in the conventional OF-protocol. The simulation results show that the proposed scheme reduced the path setup latency to too much extent in SD-WAN for huge traffic flow. Lin et al[156] focused on improving the overall healthcare monitoring system. The author presented an architecture for big data applications in emotion-aware healthcare (BDAEAH). They designed a technique based on emotion computing by considering the user's emotions or illness as the most important factor in performing the better patient treatment. The presented BDAEAH architecture performs four functions mainly (a) collecting physical signals of user's emotions through wearable sensors, (b) data transmission on data centers through SDN and 5G services, (c) data analytics of emotion computing on data centers, and (d) display final decision to remote users or doctors. The designed architecture results in the improvement in resource utilization and the overall healthcare services to remote clients.

4.4.2 | Security in SD-WBAN

To highlight various security issues in SD-WBAN, Hayward et al[21] presented a detailed survey wherein various security threats occurring in the distributed controller framework were discussed. The list of security threats is as follows: unauthorized access (Controller Hijacking), data leakage (side-channel attacks), data modification (man-in-the-middle attacks), malicious or compromised SDN applications (illegal OF-rules), and DoS attacks (unauthorized flooding requests on controller-switches communication). Smart healthcare services mainly focus on data collection, aggregation, and sensor data analysis. However, security is the most important issue which arises as the sensitive patient record is shared on the insecure public channel through wireless technologies.

In another study,[157] the authors analyzed the importance of secure data storage and transmission in the e-healthcare system. The author presented an SD-IoT centralized architecture in cloud and fog computing. Using network virtualization, the multiple controllers are deployed on the control plane and implement a distributed model based on blockchain technology on each controller. The presented blockchain technology secures patients' records by validating every transaction and adding each authenticated data block to the blockchain. Shayokh et al[158] discussed the growth of virtual hospitals based on SDN technology to provide regular medical facilities for remote users. The major problem analyzed by the author in wireless body area networks (W-BAN) is mainly insecure content delivery of patient's sensitive records. The authors presented a secure authentication protocol, that is, Kerberos, as a defense mechanism for secure data delivery in WBAN for virtual hospitals. The experimental results show that the proposed protocol reduces latency and provides better security in packet inspection to verify whether it is normal or medical data.

The IoT devices play an important role in the smart healthcare system by monitoring end-users personal health information (PHI) and displaying the aggregated report on smartphones. The aggregated report is transmitted on remote DCs to check the authentication of each mobile patient. The authenticated mobile patients are allowed to share their reports on social networking to create a group based on symptom matching (SM). This process is called mobile healthcare social networks (MHSNs) and acts as a self-organizing approach to providing remote healthcare services to patients. Jiang et al[159] focused on the challenges faced by sharing PHI with unknown guests in MHSNs. In a case, if the unknown guest is an adversary, then the privacy of sensitive information no longer exists. To overcome the misuse of sensitive SM information, the author presented a technique known as two RSA-based blind signatures SM. The proposed two blind signature technique performs both coarse-grained and fine-grained SM approaches. Moreover, a probabilistic data structure called a bloom filter is also used to check the similarity degree of two patients' symptoms based on their weighted Euclidean norm. The performance of the designed scheme is proved to be practical in maintaining privacy-preserving for SM of patient records. The numerical result shows that the proposed scheme is tested for both the coarse-grained and fine-grained techniques by considering 50 and 100 symptoms. From the experimental results, the coarse-grained SM technique achieves an average CPU load of 16.4% for 50 symptoms and 17.3% for 100 symptoms, while the average power consumption for both the symptoms is 2205.2 and 2502.1 MW. Likewise, the fine-grained SM technique computes 50 and 100 symptoms to produce an average CPU load of 15.7% and 17.6%, while the average power consumption is 2126.3 and 2457.7 MW.

Meng et al[160] focused on the issues of malicious attacks launched by an intruder in SD-WBAN. The proposed scheme is a trusted Bayesian inference model deployed with Snort IDS to inspect normal or malicious devices inside healthcare organizations. The author surveyed 12 medical organizations and analyzed the performance of the proposed scheme. The experimental result proved that the proposed scheme decreases the trust value of malicious devices both on the controller and switch side. Chaudhary et al[161] analyzed two major security threats in SD-WBAN. An adversary launched an eavesdropping attack during data transmission and a man-in-the-middle attack during data storage on the public cloud. The author presented a lattice-based cryptography technique to mitigate both security attacks. The issue related to the data exchange between the patients and doctors is solved by using a key exchange authentication mechanism based on a lattice-based cryptosystem. The second issue of insecure data storage on the public cloud is solved by using encryption and decryption. Also, the author applied an access control mechanism to assign permissions of access rights for each patient and doctor whenever accessing any information. The performance of the designed scheme shows that the lattice cryptography technique is resilient to quantum attacks and reduces the communication cost and computation time.

In another study,[162] the authors examined the challenges related to suspicious activities occurring by traveling patients. The authors designed two approaches, that is, attack signatures and behavior based IDS, to monitor and trigger an alarm to the hospital staff if any suspicious activity is found from the roaming patients. The designed approach classifies the healthcare traffic from other traffic and imposes fine-grained security policies on the controller. The database of the security policies is maintained on the SDN controller to track the live location of each patient and monitor the communication among patients. Finally, the performance of the proposed scheme is observed to be capable enough to figure out malicious devices and dynamically modify the security policies to mitigate abnormal behavior in the hospital environment.

4.4.3 | Resource scheduling in SD-WBAN

The key consideration in smart healthcare networks (HCN) is resource scheduling because the problem of network congestion arises due to the lack of available resources. Li et al[163] solved the problem of resource scarcity and network congestion in SD-WBAN. The authors presented an optimal load balancing technique based on SDN and service function chain (SFC) to eliminate the requirement of a large number of resources. The SFC is built based on a metaheuristic approach, i.e., to simulate anneal (SA) algorithm. The main objective of the SA algorithm is to achieve minimal transmission time and cost for content delivery on the medical teams through load balancing. The performance of the proposed SA algorithm is compared with the greedy algorithm. The results show that the SA algorithm attains superior results than the greedy algorithm in terms of transmission time for fewer users (less than 50). In contrast, the greedy algorithm is more appropriate for a large number of users (150 or more).

In another study,[164] the authors discussed the comfort level achieved due to chemotherapy available at the user's home. The patient or users in daily routine activities can perform their medical treatments such as–heartbeat checkups, diabetes, voice disorder, blood pressure, etc. The authors presented a healthcare framework based on edge computing and applied a deep learning method to solve the problem of voice disorder assessment. The proposed framework achieves low latency as mostly the computation is performed at the edge nodes. The results show that the proposed approach takes 0.13 s to identify voice disorder and 0.23 s to classify voice disorder and, finally, attains 98.5% accuracy with more than 99.3% sensitivity. Rego et al[165] analyzed the challenges faced in smart cities during emergencies like accidents, terrorist activity, fire, and natural disasters. The key consideration is the major challenges of improper traffic flow management and high latency. The authors presented an SDN-IoT-based control strategy and proposed an alarm treatment algorithm to minimize the damage to humans and emergency resources. The control strategy performs efficient resource scheduling using traffic lights, cameras, and alarm triggered management systems. It dynamically diverted the traffic flow routes from normal behavior toward the emergency area by giving the highest priority to emergency traffic to reduce the delay time in providing emergency services. The simulation result shows that the proposed algorithm reduced the average latency from 26 to 17 ms and minimized the delay time by up to 33% in emergency traffic.

4.4.4 | Mobility in SD-WBAN

The SD-WBAN are adopting body sensor networks (BSN) in remote or rural areas for data collection and delivering the content to doctors via a wireless network for real-time diagnosis. The issue of mobility or signal fading generally arises in wireless networks during data transmission. The mobility in HCN may lead to packet loss and increase the delay time for data transmission. Hence, the dependability and reliability of the e-healthcare system are an important concern. Junior et al[166] performed a survey on mobility issues in context with the mobile cloud environment. The authors analyzed various researchers' systematic mapping study (SMS) strategies to handle mobility in a mobile cloud scenario. The mobility strategies considered by the authors are as follows: handover algorithm, mobility management, QoS support, multi-modal support, proxy support and so on. Silva et al[167] addressed the problem of mobility and poor QoS in SD-WBAN. The author presented a mobility control scheme called a context-aware mobile approach (CAMA) to resolve the issue of mobility in e-healthcare services. The authors considered the handover mechanism to enable the mobile nodes to be best connected. The proposed framework uses the Fuzzification system model for handover prediction based on the current eHealth session. The performance of the CAMA approach helps to achieve reliability in mobility issues by improving the point of attachment and handover decision. Furthermore, the proposed framework reduces the E2E delay and minimizes packet loss at content delivery.

The summarized overview of the existing proposals in SD-WBAN is shown in Table 7.

TABLE 7 Comparison of existing proposals on SD-WBAN

Authors	Techniques	Controller	Setup Description	Prog. Lang./ Simulator	Issues				Parameters				
					1	2	3	4	5	6	7	8	9
50	M/G/1 Queuing Model, Priority Scheme	Centralized	4 nodes; 10-100 Mbps data rate; 1000 B packet size; Exp.1, Exp.2, Exp.3: voice (high), video (medium), data (low). Reduces the avg. waiting time and cost	Matlab	✓	×	✓	×	✓	✓	✓	×	×
155	POF-based source routing	Centralized (POX)	2 hosts; 2 core switches; 2 edge switches; 1 controller. Minimizes overall path setup latency	Mininet	✓	×	×	×	×	✓	✓	×	×
158	Kerberos authentication	Centralized	15 switches; 20 routers; 10 Mbps link capacity; 300 MB message size. Improves latency and defend from security threats	Matlab, ns-3	×	✓	×	×	×	✓	✓	✓	✓
159	Two-blind signature SM, Bloom Filter	Centralized	1024-bit RSA; SHA-256; **Coarse Grained (50 and 100 symptoms)**: avg. CPU loads are 16.4% and 17.3%; avg. power consumption are 2205.2 mW and 2502.1 mW; **Fine Gained (50 and 100 symptoms)**: avg. CPU loads are 15.7% and 17.6%; avg. power consumption are 2126.3 mW and 2457.7 mW	Java	×	✓	✓	×	✓	×	✓	✓	✓
160	Bayesian inference	Distributed (ODL)	12 HC organization; 50-100 devices; 245 rules. **Controller**: Normal- avg. CPU workloads 15.3%; adversary- avg. CPU workloads 33.7%. **Switch**: Normal- avg. CPU workloads 6.8%; adversary- avg. CPU workloads 18.4%	Snort IDS	×	✓	✓	×	✓	×	×	✓	✓
162	Attack signature, behavior-based IDS	Distributed (ONOS)	Dataset: UNSW-NB; arduino uno; voltage regulator IC; Total Power: 6000 mAH, 5V battery; ESP8266 WLAN chip. Detect abnormal behavior and avoids congestion	OpenWrt	✓	✓	×	✓	✓	×	×	✓	✓
165	Alarm treatment control strategy	Centralized	**Emergency Traffic**: Reduced latency from 26 ms to 17 ms and minimize latency upto 33%. **Normal Traffic**: minimum latency is 0.012 ms and reduced latency by upto 50%	Mininet	✓	✓	✓	×	✓	✓	×	✓	×
167	CAMA, fuzzy logic	Centralized	Decreases E2E delay and minimize the overall packet loss	-	×	×	✓	✓	✓	✓	×	×	✓

Note: 1: Routing; 2: Security; 3: Resource Scheduling; 4: Mobility; 5: Cost; 6: Latency; 7: Bandwidth; 8: Resilient from Security Attacks; 9: Packet loss/Accuracy; Notations ✓: considered, and ×: not-considered.
Abbreviation: SD-WBAN, software-defined wireless body area network.

5 | OPEN CHALLENGES AND FUTURE DIRECTIONS

After the detailed analysis of the aforementioned existing proposals concerning various smart applications, the following research challenges have been identified.

- **Vision of traffic engineering at the DCs is moving at a slow pace**: With an increase in the number of clouds IP traffic and DCs IP traffic globally, the vision for enabling the network traffic to flow smoothly through the underlying network infrastructure is still not fully addressed. The traffic residing within the DCs, inter-DC traffic, and the end-users connected to the DCs through the Internet grow rapidly. However, the dynamic approach to handling such high traffic volumes is moving at a slow pace. Many studies and projects are going on worldwide, but still, a lot needs to be done.
- **Less focus on energy-efficient and fast flow forwarding**: Existing studies utilized flow forwarding as the primary source to perform data forwarding. None of the work tried to propose an energy-efficient and fast flow forwarding scheme.
- **Inefficient utilization of resources and servers in DCs**: Even after vast research in cloud computing worldwide, the over-provisioning of resources is a common practice, and there are various resources which are under-provisioned. This leads to inefficient utilization of resources which increases energy consumption, latency, congestion, and operational cost.
- **Less focus on multiple active controller assignment**: While deploying multiple SDN controllers, the decision on the location of the active SDN controllers is moving at a slow pace. The optimal decision helps each active SDN controller make intelligent decisions, thereby serving maximum connected switches and links. This issue of the controller assignment can lead to an increase in the average control latency.
- **Less focus on virtual hypervisor placement to maximize resource utilization**: With network virtualization, the problem of deciding where the hypervisor locations should be present and how many hypervisor instances are needed is still not addressed in multicontrollers networks.
- **No optimal trade-off scheme for fault-tolerance at the control and data plane**: The existing proposals fail to achieve fault tolerance in heterogeneous multi-cloud DCs. The problem of failure recovery at the data plane, control plane, and controller is a big issue as SDN is a centralized architecture1.
- **No focus on IP routing security**: The issues of security attacks may arise at the centralized controller and network forwarding devices which is still not resolved.
- **Network Reconfiguration in case of time-critical flows**: There are challenges related to reconfiguration in the network due to any frequent changes in the flows or device failure, which may lead to re-routing and rescheduling of existing flows. In this scenario, more attention needs to pay to how SDN can handle time-critical flows and best-efforts flow in case of a node failure, link failure, or changes in the network topology. What will be the effect of changing the gate control lists (GCL) of OF-switches in case of dynamic network configuration as static flow rules are already preconfigured on switches?
- **Less focus on SDN deployment using microservices**: There are challenges related to scalability, flexibility and highly expensive physical infrastructure maintenance. These issues can be resolved by migrating from the physical server to dockers and serverless computing to run small applications of SDN controllers and OF-switches. In this regard, how can the SDN use dockers and Kubernetes for deploying micro open network operating system (ONOS) services? The docker platform executes the GUI applications in milliseconds without installing any software and saves storage space needs to be considered.
- **Need to design high-level architecture of CNC**: The high-level design architecture of SDN that comprises CNC and CUC needs to understand. Which services and databases are required for topology, configuration, network reconfiguration, monitoring and metrics evaluation and how to do all these services communicate on the control plane as well as with the plugin adapter of the northbound and southbound APIs?

6 | CONCLUSION

Although traditional networking is widely adopted in WANs, network management has various challenges. The major issue occurring in traditional networking is the strong coupling of the data (forwarding devices) and the control (network brain) plane. The vertically integrated planes create the static and manual configuration of devices. Also, the conventional networking is vendor-specific which creates much complexity in the management of WANs. The key features of traditional networking lead to data inconsistency and huge network delay. SDN is a promising and innovative technology widely implemented in the digitalization world. The key features of SDN enabled decoupling the control part with the forwarding plane. The automation is achieved in network management through a programmable SDN network. In addition, SDN is a dynamic approach to modifying the network topology as the network load increases. The major benefits of deploying SDN are scalability, reliability, and flexibility.

The proposed survey is different from the existing surveys as the comparative analysis is done on the existing work. The comparison of the existing survey with the proposed survey considered various parameters such as SDN components, SDN issues, network virtualization, and SDN deployment in various real-time applications. The proposed survey is divided into three parts. The first part of the survey discussed the history and architecture of SDN in detail. The SDN history is analyzed based on year-wise improvement and innovation in SDN components. Next, the architecture components of SDN such as SDN switches, controllers, interfaces, emulation, and simulation tools are analyzed. The second part of the survey discussed real-time deployment applications of SDN. The deployment applications considered in the survey are SD-DC, SD-FC/EC, SD-IoT, SDWSNs, and SD-WBAN. The comparative analysis of the existing survey on SDN applications is performed based on problem formulation, techniques proposed, and the parameters that improved the overall network performance. Finally, the last part of the survey discussed the open issues and research challenges in the SDN ecosystem with respect to smart applications.

In the future, the time-sensitive networks (TSN) configuration in the SDN-based control plane for data-driven real-time industrial IoT applications will be explored. For delay-constraint applications, an efficient reconfiguration scheme will be proposed for real-time traffic flows and resilient scheduling of the GCL on the OF switches.

ACKNOWLEDGMENT

This work is partially supported through Startup Fund provided by the Durham University, UK.

DATA AVAILABILITY STATEMENT

Data sharing not applicable to this article as no datasets were generated or analysed during the current study.

ORCID

Gagangeet Singh Aujla https://orcid.org/0000-0002-2870-8938
Neeraj Kumar https://orcid.org/0000-0002-3020-3947

REFERENCES

1. Kaur K, Garg S, Aujla GS, Kumar N, Rodrigues JJPC, Guizani M. Edge computing in the industrial internet of things environment: Software-defined-networks-based edge-cloud interplay. *IEEE Commun Mag*. 2018;56(2):44-51. https://doi.org/10.1109/MCOM.2018.1700622
2. Habeeb F, Alwasel K, Noor A, et al. Dynamic bandwidth slicing for time-critical iot data streams in the edge-cloud continuum. *IEEE Trans Ind Inform*. 2022.
3. Cisco global cloud index: Forecast and methodology, 2015-2020 white paper. Accessed on: March 2017.
4. Whitney J, Delforge P. Data center efficiency assessment–scaling up energy efficiency across the data center industry: Evaluating key drivers and barriers. 14–08, Rep. IP NRDC and Anthesis; 2014.
5. Singh M, Aujla GS, Singh A, Kumar N, Garg S. Deep-learning-based blockchain framework for secure software-defined industrial networks. *IEEE Trans Ind Inform*. 2020;17(1):606-616.
6. Nunes BAA, Mendonca M, Nguyen X-N, Obraczka K, Turletti T. A survey of software-defined networking: Past, present, and future of programmable networks. *IEEE Commun Surv Tutorials*. 2014;16(3):1617-1634.
7. Hu F, Hao Q, Bao K. A survey on software-defined network and openflow: From concept to implementation. *IEEE Commun Surv Tutorials*. 2014;16(4):2181-2206.
8. Lara A, Kolasani A, Ramamurthy B. Network innovation using openflow: A survey. *IEEE Commun Surv Tutorials*. 2014;16(1):493-512.
9. Akyildiz IF, Lee A, Wang P, Luo M, Chou W. A roadmap for traffic engineering in SDN-openflow networks. *Comput Netw*. 2014;71:1-30.
10. Kreutz D, Ramos FernandoMV, Verissimo PE, Rothenberg CE, Azodolmolky S, Uhlig S. Software-defined networking: A comprehensive survey. *Proc IEEE*. 2015;103(1):14-76.
11. Xia W, Wen Y, Foh CH, Niyato D, Xie H. A survey on software-defined networking. *IEEE Commun Surv Tutorials*. 2015;17(1):27-51.
12. Blenk A, Basta A, Reisslein M, Kellerer W. Survey on network virtualization hypervisors for software defined networking. *IEEE Commun Surv Tutorials*. 2015;18(1):655-685.

13. Mijumbi R, Serrat J, Gorricho J-L, Bouten N, De Turck F, Boutaba R. Network function virtualization: State-of-the-art and research challenges. *IEEE Commun Surv Tutorials*. 2016;18(1):236-262.
14. Trois C, Del Fabro MD, de Bona LuisCE, Martinello M. A survey on SDN programming languages: Toward a taxonomy. *IEEE Commun Surv Tutorials*. 2016;18(4):2687-2712.
15. Nguyen X-N, Saucez D, Barakat C, Turletti T. Rules placement problem in openflow networks: a survey. *IEEE Commun Surv Tutorials*. 2016;18(2):1273-1286.
16. Guck JW, Van Bemten A, Reisslein M, Kellerer W. Unicast QoS routing algorithms for SDN: A comprehensive survey and performance evaluation. *IEEE Commun Surv Tutorials*. 2018;20(1):388-415.
17. Alvizu R, Maier G, Kukreja N, et al. Comprehensive survey on T-SDN: Software-defined networking for transport networks. *IEEE Commun Surv Tutorials*. 2017;19(4):2232-2283.
18. Huang T, Yu FR, Zhang C, Liu J, Zhang J, Liu Y. A survey on large-scale software defined networking (SDN) testbeds: Approaches and challenges. *IEEE Commun Surv Tutorials*. 2017;19(2):891-917.
19. Baktir AC, Ozgovde A, Ersoy C. How can edge computing benefit from software-defined networking: a survey, use cases, and future directions. *IEEE Commun Surv Tutorials*. 2017;19(4):2359-2391.
20. Yan Q, Yu FR, Gong Q, Li J. Software-defined networking (SDN) and distributed denial of service (DDoS) attacks in cloud computing environments: A survey, some research issues, and challenges. *IEEE Commun Surv Tutorials*. 2016;18(1):602-622.
21. Scott-Hayward S, Natarajan S, Sezer S. A survey of security in software defined networks. *IEEE Commun Surv Tutorials*. 2016;18(1): 623-654.
22. Rawat DB, Reddy SR. Software defined networking architecture, security and energy efficiency: A survey. *Environment*. 2017;3(5):6.
23. Khan S, Gani A, Wahab AWA, Guizani M, Khan MK. Topology discovery in software defined networks: Threats, taxonomy, and state-of-the-art. *IEEE Commun Surv Tutorials*. 2017;19(1):303-324.
24. Molina E, Jacob E. Software-defined networking in cyber-physical systems: A survey. *Comput Electr Eng*. 2018;66:407-419. https://doi.org/10.1016/j.compeleceng.2017.05.013
25. Zhao Y, Li Y, Zhang X, Geng G, Zhang W, Sun Y. A survey of networking applications applying the software defined networking concept based on machine learning. *IEEE Access*. 2019;7:95,397-95,417. https://doi.org/10.1109/ACCESS.2019.2928564
26. Rafique W, Qi L, Yaqoob I, Imran M, Rasool RU, Dou W. Complementing iot services through software defined networking and edge computing: A comprehensive survey. *IEEE Commun Surv Tutorials*. 2020;22(3):1761-1804. https://doi.org/10.1109/COMST.2020.2997475
27. Haji SH, Zeebaree SR, Saeed RH, et al. Comparison of software defined networking with traditional networking. *Asian J Res Comput Sci*. 2021;9(2). https://doi.org/10.9734/ajrcos/2021/v9i230216
28. Pusuluri R. YA. Software-defined networking and architecture of iot with security, challenges and applications: A survey. *Advances in Smart System Technologies. Advances in Intelligent Systems and Computing*. Singapore: Springer; 2021. https://doi.org/10.1007/978-981-15-5029-443
29. Bhardwaj S, Panda SN. Performance evaluation using ryu sdn controller in software-defined networking environment. *Wirel Pers Commun*. 2022;122(1):701-723.
30. Ma X, Liao L, Li Z, Lai RX, Zhang M. Applying federated learning in software-defined networks: A survey. *Symmetry*. 2022;14(2):195.
31. Liu F, Kibalya G, Santhosh Kumar SVN, Zhang P. Challenges of traditional networks and development of programmable networks. *Software defined internet of everything*: Springer; 2022:37-61.
32. Kibalya G, Gorricho J-L, Zhang P, et al. A reinforcement learning approach for virtual network function chaining and sharing in softwarized networks. *IEEE Trans Netw Serv Manag*. 2022.
33. Zhang P, Liu F, Vashisht S, Singh Mann R. A robust network measurement and feature selection strategy for software-defined edge computing environment. *Trans Emerg Telecommun Technol*. 2021;32(6):e4003.
34. Ren X, Aujla GS, Jindal A, Batth RS, Zhang P. Adaptive recovery mechanism for sdn controllers in edge-cloud supported fintech applications. *IEEE Internet Things J*. 2021.
35. Zhang P, Wang C, Kumar N, Liu L. Space-air-ground integrated multi-domain network resource orchestration based on virtual network architecture: A drl method. *IEEE Trans Intell Transp Syst*. 2021.
36. Aujla GS, Garg S, Kaur K, Sikdar B. *Software defined internet of everything*: Springer; 2022.
37. Smith JM, Farber DJ, Gunter CA, Nettles SM, Feldmeier DC, Sincoskie WD. *Switchware: accelerating network evolution (white paper)*; 1996.
38. Alexander DS, Arbaugh WA, Hicks MW, et al. The switchware active network architecture. *IEEE Netw*. 1998;12(3):29-36.
39. Kohler E, Morris R, Chen B, Jannotti J, Kaashoek MF. The click modular router. *ACM Trans Comput Syst (TOCS)*. 2000;18(3):263-297.
40. Handley M, Hodson O, Kohler E. XORP: An open platform for network research. *ACM SIGCOMM Comput Commun Rev*. 2003;33(1): 53-57.
41. Quagga routing software suite.
42. The BIRD internet routing daemon.
43. Van der Merwe JE, Rooney S, Leslie L, Crosby S. The tempest-a practical framework for network programmability. *IEEE Netw*. 1998; 12(3):20-28.
44. Feamster N, Balakrishnan H, Rexford J, Shaikh A, van der Merwe J. The case for separating routing from routers. In: Acm sigcomm workshop on future directions in network architecture, FDNA '04; New York, NY, USA; 2004:5-12.

45. Rexford J, Greenberg A, Hjalmtysson G, et al. Network-wide decision making: Toward a wafer-thin control plane. In: Proc. hotnets; 2004:59-64.
46. Greenberg A, Hjalmtysson G, Maltz DA, et al. A clean slate 4D approach to network control and management. *ACM SIGCOMM Comput Commun Rev*. 2005;35(5):41-54.
47. Chen X, Zhong Y, Jukan A. Multipath routing in path computation element (PCE): Protocol extensions and implementation. In: 8th Conference on Network and Optical Communications (NOC); 2013:75-82.
48. Casado M, Freedman MJ, Pettit J, Luo J, McKeown N, Shenker S. Ethane: Taking control of the enterprise. In: Acm sigcomm Computer Communication Review, Vol. 37; 2007:1-12.
49. Gude N, Koponen T, Pettit J, Pfaff B, Casado M, McKeown N, Shenker S. NOX: towards an operating system for networks. *ACM SIGCOMM Comput Commun Rev*. 2008;38(3):105-110.
50. Aujla GS, Chaudhary R, Kumar N, Kumar R, Rodrigues JJPC. An ensembled scheme for qos-aware traffic flow management in software defined networks. In: 2018 IEEE International Conference on Communications (ICC). IEEE; 2018:1-7.
51. Singh A, Aujla GS, Garg S, Kaddoum G, Singh G. Deep-learning-based sdn model for internet of things: An incremental tensor train approach. *IEEE Internet Things J*. 2019;7(7):6302-6311.
52. Aujla GS, Singh A, Kumar N. Adaptflow: Adaptive flow forwarding scheme for software-defined industrial networks. *IEEE Internet Things J*. 2019;7(7):5843-5851.
53. Jindal A, Aujla GS, Kumar N. Survivor: A blockchain based edge-as-a-service framework for secure energy trading in sdn-enabled vehicle-to-grid environment. *Comput Netw*. 2019;153:36-48.
54. Aujla GS, Singh M, Bose A, Kumar N, Han G, Buyya R. Blocksdn: Blockchain-as-a-service for software defined networking in smart city applications. *IEEE Netw*. 2020;34(2):83-91.
55. Singh A, Batra S, Aujla GS, Kumar N, Yang LT. Bloomstore: dynamic bloom-filter-based secure rule-space management scheme in sdn. *IEEE Trans Ind Inform*. 2020;16(10):6252-6262.
56. Wang H, Srivastava A, Xu L, Hong S, Gu G. Bring your own controller: Enabling tenant-defined SDN apps in IaaS clouds. In: IEE Infocom; 2017:1-9.
57. Singh A, Aujla GS, Bali RS. Intent-based network for data dissemination in software-defined vehicular edge computing. *IEEE Trans Intell Transp Syst*. 2020;22(8):5310-5318.
58. Chaudhary R, Kumar N. Loads: Load optimization and anomaly detection scheme for software-defined networks. *IEEE Trans Veh Technol*. 2019;68(12):12329-12344.
59. Wen Z, Garg S, Aujla GS, et al. Running industrial workflow applications in a software-defined multicloud environment using green energy aware scheduling algorithm. *IEEE Trans Ind Inform*. 2020;17(8):5645-5656.
60. Aujla GS, Jindal A, Kumar N, Singh M. SDN-based data center energy management system using RES and electric vehicles. In: Ieee Global Communications Conference (GLOBECOM); 2016:1-6.
61. Aujla GS, Kumar N. SDN-based energy management scheme for sustainability of data centers: An analysis on renewable energy sources and electric vehicles participation. *J Parallel Distrib Comput*. 2018;117:228-245. https://doi.org/10.1016/j.jpdc.2017.07.002
62. Yu B, Han Y, Wen X, Chen X, Xu Z. An energy-aware algorithm for optimizing resource allocation in software defined network. In: IEEE Global Communications Conference (GLOBECOM); 2016:1-7.
63. Aujla GS, Kumar N. MEnSuS: An efficient scheme for energy management with sustainability of cloud data centers in edge-cloud environment. *Futur Gener Comput Syst*. 2018;86:1279-1300. https://doi.org/10.1016/j.future.2017.09.066
64. Zeng D, Yang G, Gu L, Guo S, Yao H. Joint optimization on switch activation and flow routing towards energy efficient software defined data center networks. In: IEEE International Conference on Communications (ICC); 2016:1-6.
65. Son J, Dastjerdi AV, Calheiros RN, Buyya R. Sla-aware and energy-efficient dynamic overbooking in SDN-based cloud data centers. *IEEE Trans Sustain Comput*. 2017;2(2):76-89.
66. Amokrane A, Langar R, Boutaba R, Pujolle G. Flow-based management for energy efficient campus networks. *IEEE Trans Netw Serv Manag*. 2015;12(4):565-579.
67. Chaudhary R, Kumar N. Enflow: An energy-efficient fast flow forwarding scheme for software-defined networks. *IEEE Trans Intell Transp Syst*. 2020;22(8):5293-5309.
68. Amarasinghe H, Jarray A, Karmouch A. Fault-tolerant IaaS management for networked cloud infrastructure with SDN. In: IEEE International Conference on Communications (ICC); 2017:1-7.
69. Abdelmoniem AM, Bensaou B, Abu AJ. SICC: SDN-based incast congestion control for data centers. In: IEEE International Conference on Communications (ICC); 2017:1-6.
70. Liu J, Liu J, Xie R. Reliability-based controller placement algorithm in software defined networking. *Comput Sci Inform Syst*. 2016;00:14-14.
71. Li H, Li P, Guo S, Nayak A. Byzantine-resilient secure software-defined networks with multiple controllers in cloud. *IEEE Trans Cloud Comput*. 2014;2(4):436-447.
72. Astaneh SA, Heydari SS. Optimization of SDN flow operations in multi-failure restoration scenarios. *IEEE Trans Netw Serv Manag*. 2016;13(3):421-432.
73. Xie J, Guo D, Zhu X, Ren B, Chen H. Minimal fault-tolerant coverage of controllers in IaaS datacenters. *IEEE Trans Serv Comput*. 2017;18:1128.

74. Chaudhary R, Kumar N. Parc: placement availability resilient controller scheme for software-defined datacenters. *IEEE Trans Veh Technol*. 2020;69(8):8985-9001.
75. Son J, Dastjerdi AV, Calheiros RN, Ji X, Yoon Y, Buyya R. Cloudsimsdn: Modeling and simulation of software-defined cloud data centers. In: 15th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID); 2015:475-484.
76. Li F, Cao J, Wang X, Sun Y. A QoS guaranteed technique for cloud applications based on software defined networking. *IEEE access*. 2017;5:21,229-21,241.
77. de Souza FR, Miers CC, Fiorese A, Koslovski GP. QoS-aware virtual infrastructures allocation on SDN-based clouds. In: 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID); 2017:120-129.
78. Yuan H, Bi J, Zhang J, Tan W, Huang K. Workload-aware revenue maximization in SDN-enabled data center. In: IEEE 10th International Conference on Cloud Computing (CLOUD); 2017:18-25.
79. Li W, Qi H, Li K, Stojmenovic I, Lan J. Joint optimization of bandwidth for provider and delay for user in software defined data centers. *IEEE Trans Cloud Comput*. 2017;5(2):331-343.
80. Zhang L, Guo S, Yang Y, Liu D, Liu R. RE-FPR: flow preemption routing scheme with redundancy elimination in software defined data center networks. *Sustain Comput: Inform Syst*. 2018;18:14-24.
81. Amiri M, Sobhani A, Al Osman H, Shirmohammadi S. SDN-enabled game-aware routing for cloud gaming datacenter network. *IEEE Access*. 2017;5:18,633-18,645.
82. Chiu C-H, Singh DK, Wang Q, Lee K, Park S-J. Minimal coflow routing and scheduling in openflow-based cloud storage area networks. In: IEEE 10th International Conference on Cloud Computing (CLOUD); 2017:222-229.
83. Caria M, Jukan A, Hoffmann M. SDN partitioning: A centralized control plane for distributed routing protocols. *IEEE Trans Netw Serv Manag*. 2016;13(3):381-393.
84. Wu Y, Zhang Z, Wu C, Guo C, Li Z, Lau FrancisCM. Orchestrating bulk data transfers across geo-distributed datacenters. *IEEE Trans Cloud Comput*. 2017;5(1):112-125.
85. Li D, Shang Y, He W, Chen C. EXR: greening data center network with software defined exclusive routing. *IEEE Trans Comput*. 2015; 64(9):2534-2544.
86. Chaudhary R, Aujla GS, Kumar N, Rodrigues JJPC. Optimized big data management across multi-cloud data centers: Software-defined-network-based analysis. *IEEE Commun Mag*. 2018;56(2):118-126.
87. Aujla GS, Chaudhary R, Kumar N, Rodrigues JoelJPC, Vinel A. Data offloading in 5g-enabled software-defined vehicular networks: A stackelberg-game-based approach. *IEEE Commun Mag*. 2017;55(8):100-108.
88. Yan Q, Yu FR. Distributed denial of service attacks in software-defined networking with cloud computing. *IEEE Commun Mag*. 2015; 53(4):52-59.
89. Lorenz C, Hock D, Scherer J, et al. An SDN/NFV-enabled enterprise network architecture offering fine-grained security policy enforcement. *IEEE Commun Mag*. 2017;55(3):217-223.
90. Giotis K, Argyropoulos C, Androulidakis G, Kalogeras D, Maglaris V. Combining openflow and sflow for an effective and scalable anomaly detection and mitigation mechanism on SDN environments. *Comput Netw*. 2014;62:122-136.
91. Monshizadeh M, Khatri V, Kantola R. Detection as a service: An SDN application. In: 19th International Conference on Advanced Communication Technology (ICACT); 2017:285-290.
92. Shamseddine M, Itani W, Kayssi A, Chehab A. Virtualized network views for localizing misbehaving sources in SDN data planes. In: IEEE International Conference on Communications (ICC); 2017:1-7.
93. Pisharody S, Natarajan J, Chowdhary A, Alshalan A, Huang D. Brew: A security policy analysis framework for distributed SDN-based cloud environments. *IEEE Trans Dependable Secure Comput*. 2017;16(6):1011-1025.
94. Sendi AS, Jarraya Y, Pourzandi M, Cheriet M. Efficient provisioning of security service function chaining using network security defense patterns. *IEEE Trans Serv Comput*. 2016;12:534.
95. Feng H, Llorca J, Tulino AM, Raz D, Molisch AF. Approximation algorithms for the NFV service distribution problem. In: IEEE Infocom; 2017:1-9.
96. Zhao Z, Hong F, Li R. SDN based VxLAN optimization in cloud computing networks. *IEEE Access*. 2017;5:23,312-23,319.
97. Noghani KA, Benet CH, Kassler A, Marotta A, Jestin P, Srivastava VV. Automating ethernet VPN deployment in SDN-based data centers. In: Fourth International Conference on Software Defined Systems (SDS); 2017:61-66.
98. Caixinha D, Kathiravelu P, Veiga L. ViTeNA: An SDN-based virtual network embedding algorithm for multi-tenant data centers. In: IEEE 15th International Symposium on Network Computing and Applications (NCA); 2016:140-147.
99. Wang H, Li Y, Zhang Y, Jin D. Virtual machine migration planning in software-defined networks. In: IEEE Conference on Computer Communications (INFOCOM); 2015:487-495.
100. Cziva R, Jouet S, Stapleton D, Tso FP, Pezaros DP. SDN-based virtual machine management for cloud data centers. *IEEE Trans Netw Serv Manag*. 2016;13(2):212-225.
101. Singh A, Bali RS, Aujla GS. Prospective on technical considerations for edge–cloud cooperation using software-defined networking. *Software defined internet of everything*: Springer; 2022:147-176.
102. Aujla GS, Kumar N, Garg S, Kaur K, Ranjan R. Edcsus: Sustainable edge data centers as a service in sdn-enabled vehicular environment. *IEEE Trans Sustain Comput*. 2019;7(2):263-276.
103. Al Faruque MA, Vatanparvar K. Energy management-as-a-service over fog computing platform. *IEEE Internet Things J*. 2016;3(2): 161-169.

104. Aujla GS, Kumar N, Zomaya AY, Ranjan R. Optimal decision making for big data processing at edge-cloud environment: An SDN perspective. *IEEE Trans Ind Inform*. 2018;14(2):778-789.
105. Sarkar S, Chatterjee S, Misra S. Assessment of the suitability of fog computing in the context of internet of things. *IEEE Trans Cloud Comput*. 2015;1:46-59.
106. Gupta H, Vahid Dastjerdi A, Ghosh SK, Buyya R. ifogsim: A toolkit for modeling and simulation of resource management techniques in the internet of things, edge and fog computing environments. *Softw Pract Experience*. 2017;47(9):1275-1296.
107. Yin B, Shen W, Cheng Y, Cai LX, Li Q. Distributed resource sharing in fog-assisted big data streaming. In: IEEE International Conference on Communications (ICC). IEEE; 2017:1-6.
108. Aliyu SO, Chen F, He Y, Yang H. A game-theoretic based qos-aware capacity management for real-time edgeiot applications. In: IEEE International Conference on Software Quality, Reliability and Security (QRS); 2017:386-397.
109. Ruia A, Casey CJ, Saha S, Sprintson A. Flowcache: A cache-based approach for improving sdn scalability. In: IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS); 2016:610-615.
110. Chaudhary R, Kumar N, Zeadally S. Network service chaining in fog and cloud computing for the 5G environment: Data management and security challenges. *IEEE Commun Mag*. 2017;55(11):114-122.
111. Sharma PK, Chen M-Y, Park JH. A software defined fog node based distributed blockchain cloud architecture for IoT. *IEEE Access*. 2018;6:115-124.
112. Stojmenovic I, Wen S. The fog computing paradigm: Scenarios and security issues. In: Federated Conference on Computer Science and Information Systems (FEDCSIS); 2014:1-8.
113. Tseng Y, Nait-Abdesselam F, Khokhar A. SENAD: Securing network application deployment in software defined networks. In: IEEE International Conference on Communications (ICC); 2018:1-6.
114. Bruschi R, Davoli F, Lago P, Pajo JF. A scalable SDN slicing scheme for multi-domain fog/cloud services. In: IEEE Conference on Network Softwarization (NETSOFT); 2017:1-6.
115. Li K, Nabrzyski J. Virtual machine placement in cloudlet mesh with network topology reconfigurability. In: IEEE 6th International Conference on Cloud Networking (CLOUDNET); 2017:1-7.
116. Liang K, Zhao L, Chu X, Chen H-H. An integrated architecture for software defined and virtualized radio access networks with fog computing. *IEEE Netw*. 2017;31(1):80-87.
117. Dominicini CK, Vassoler GL, Meneses LF, Villaca RS, Ribeiro MR, Martinello M. Virtphy: Fully programmable NFV orchestration architecture for edge data centers. *IEEE Trans Netw Serv Manag*. 2017;14(4):817-830.
118. Bruschi R, Davoli F, Lago P, Lombardo A, Lombardo C, Rametta C, Schembra G. An SDN/NFV platform for personal cloud services. *IEEE Trans Netw Serv Manag*. 2017;14(4):1143-1156.
119. Hu L, Wang J, Song E, Ksentini A, Hossain MA, Rawashdeh M. SDN-SPS: Semi-physical simulation for software-defined networks. *IEEE Sensors J*. 2016;16(20):7355-7363.
120. Wen S, Huang C, Chen X, Ma J, Xiong N, Li Z. Energy-efficient and delay-aware distributed routing with cooperative transmission for internet of things. *J Parallel Distrib Comput*. 2018;118:46-56.
121. Wang C-H, Kuo J-J, Yang D-N, Chen W-T. Green software-defined internet of things for big data processing in mobile edge networks. In: IEEE International Conference on Communications (ICC); 2018:1-7.
122. Bendouda D, Rachedi A, Haffaf H. An hybrid and proactive architecture based on SDN for internet of things. In: 13th International Wireless Communications and Mobile Computing Conference (IWCMC); 2017:951-956.
123. Petroulakis NE, Spanoudakis G, Askoxylakis IG. Fault tolerance using an SDN pattern framework. In: IEEE Global Communications Conference (GLOBECOM); 2017:1-6.
124. Mohan PM, Truong-Huu T, Gurusamy M. Primary-backup controller mapping for byzantine fault tolerance in software defined networks. In: Globecom 2017-2017 IEEE Global Communications Conference; 2017:1-7.
125. Munir MS, Abedin SF, Alam MGR, Tran NH, Hong CS. Intelligent service fulfillment for software defined networks in smart city. In: 2018 International Conference on Information Networking (ICOIN); 2018:516-521.
126. Bouten N, Mijumbi R, Serrat J, Famaey J, Latré S, De Turck F. Semantically enhanced mapping algorithm for affinity constrained service function chain requests. *IEEE Trans Netw Serv Manag*. 2017;14(2):317-331.
127. Cerroni W, Buratti C, Cerboni S, et al. Intent-based management and orchestration of heterogeneous openflow/IoT SDN domains. In: IEEE Conference on Network Softwarization (NETSOFT); 2017:1-9.
128. Banaie F, Yaghmaee MH, Hosseini SA. SDN-based scheduling strategy on load balancing of virtual sensor resources in sensor-cloud. In: 8th International Symposium on Telecommunications (ist); 2016:666-671.
129. Wu D, Arkhipov DI, Asmare E, Qin Z, McCann JA. UbiFlow: Mobility management in urban-scale software defined IoT. In: IEEE Conference on Computer Communications (infocom); 2015:208-216.
130. Lee M-C, Sheu J-P. An efficient routing algorithm based on segment routing in software-defined networking. *Comput Netw*. 2016;103:44-55.
131. Kharkongor C, Chithralekha T, Varghese R. A SDN controller with energy efficient routing in the internet of things (IoT). *Procedia Comput Sci*. 2016;89:218-227.
132. Lin C, Wang K, Deng G. A QoS-aware routing in SDN hybrid networks. *Procedia Computer Sci*. 2017;110:242-249.
133. De Assis MV, Hamamoto AH, Abrao T, Proença ML. A game theoretical based system using holt-winters and genetic algorithm with fuzzy logic for DoS/DDoS mitigation on SDN networks. *IEEE Access*. 2017;5:9485-9496.

134. Ozcelik M, Chalabianloo N, Gur G. Software-defined edge defense against IoT-based DDoS. In: IEEE International Conference on Computer and Information Technology (CIT); 2017:308-313.
135. Phan TV, Bao NK, Park M. A novel hybrid flow-based handler with DoS attacks in software-defined networking. In: Intl IEEE conferences on ubiquitous intelligence & computing, advanced and trusted computing, scalable computing and communications, cloud and big data computing, internet of people, and smart world congress (uic/atc/scalcom/cbdcom/iop/smartworld); 2016:350-357.
136. Dao N-N, Kim J, Park M, Cho S. Adaptive suspicious prevention for defending DoS attacks in SDN-based convergent networks. *PloS one*. 2016;11(8):e0160375.
137. Blenk A, Basta A, Zerwas J, Reisslein M, Kellerer W. Control plane latency with SDN network hypervisors: The cost of virtualization. *IEEE Trans Netw Serv Manag*. 2016;13(3):366-380.
138. Al-Janabi TA, Al-Raweshidy HS. Efficient whale optimisation algorithm-based SDN clustering for IoT focused on node density. In: 16th annual mediterranean ad hoc networking workshop (med-hoc-net); 2017:1-6.
139. Wang R, Zhang Z, Zhang Z, Jia Z. ETMRM: An energy-efficient trust management and routing mechanism for SDWSNs. *Comput Netw*. 2018;139:119-135.
140. Zhao Y, Yan B, Zhang J. Software defined passive optical networks with energy-efficient control strategy. *Optik-Int J Light Electron Optics*. 2016;127(23):11211-11219.
141. Chaudhary R, Kumar N. Secgreen: Secrecy ensured power optimization scheme for software-defined connected iov. *IEEE Trans Mob Comput*. 2021.
142. Shi H-L, Hou KM, Zhou H-Y, Liu X. Energy efficient and fault tolerant multicore wireless sensor network: E^{2}mwsn. In: 7th international conference on wireless communications, networking and mobile computing (wicom); 2011:1-4.
143. Aly WHF. A novel fault tolerance mechanism for software defined networking. In: European Modelling Symposium (EMS); 2018. https://doi.org/10.1109/EMS.2017.47
144. Pan T, Guo X, Zhang C, Meng W, Liu B. ALFE: A replacement policy to cache elephant flows in the presence of mice flooding. In: IEEE International Conference on Communications (ICC); 2012:2961-2965.
145. Kitsuwan N, Oki E. Analysis of flows reduction scheme by adopting two MPLS tags in software-defined network. In: IEEE conference on standards for communications and networking (cscn); 2016:1-6.
146. Challa R, Lee Y, Choo H. Intelligent eviction strategy for efficient flow table management in openflow switches. In: IEEE Netsoft Conference and Workshops (NETSOFT); 2016:312-318.
147. Han Z, Li Y, Li J. A novel routing algorithm for IoT cloud based on hash offset tree. *Futur Gener Comput Syst*. 2018;86:456.
148. Gharaibeh A, Salahuddin MA, Hussini SJ, Khreishah A, Khalil I, Guizani M, Al-Fuqaha A. Smart cities: A survey on data management, security, and enabling technologies. *IEEE Commun Surv Tutorials*. 2017;19(4):2456-2501.
149. Gulati A, Aujla GS, Chaudhary R, Kumar N, Obaidat M, Benslimane A. Dilse: Lattice-based secure and dependable data dissemination scheme for social internet of vehicles. *IEEE Trans Dependable Secure Comput*. 2019;18:2520.
150. Moyano RF, Cambronero DF, Triana LB. A user-centric SDN management architecture for NFV-based residential networks. *Comput Stand Interfaces*. 2017;54:279-292.
151. Arezoumand S, Bannazadeh H, Leon-Garcia A. Hyperexchange: A protocol-agnostic exchange fabric enabling peering of virtual networks. In: IFIP/IEEE Symposium on Integrated Network and Service Management (IM); 2017:204-212.
152. Drutskoy D, Keller E, Rexford J. Scalable network virtualization in software-defined networks. *IEEE Internet Comput*. 2013;17(2):20-27.
153. Duan Q, Ansari N, Toy M, et al. Software-defined network virtualization: an architectural framework for integrating sdn and nfv for service provisioning in future networks. *IEEE Netw*. 2016;30(5):10-16.
154. Aujla GS, Chaudhary R, Kaur K, Garg S, Kumar N, Rajan R. SAFE: SDN assisted framework for edge-cloud interplay in secure healthcare ecosystem. *IEEE Trans Indust Inform*. 2018;15:469. https://doi.org/10.1109/TII.2018.2866917
155. Li S, Hu D, Fang W, Zhu Z. Source routing with protocol-oblivious forwarding (POF) to enable efficient e-health data transfers. In: IEEE International Conference on Communications (ICC); 2016:1-6.
156. Lin K, Xia F, Wang W, Tian D, Song J. System design for big data application in emotion-aware healthcare. *IEEE Access*. 2016;4:6901-6909.
157. Salahuddin MA, Al-Fuqaha A, Guizani M, Shuaib K, Sallabi F. Softwarization of internet of things infrastructure for secure and smart healthcare. arXiv preprint arXiv:1805.11011; 2018.
158. Al Shayokh M, Abeshu A, Satrya GB, Nugroho MA. Efficient and secure data delivery in software defined WBAN for virtual hospital. In: International conference on control, electronics, renewable energy and communications (ICCEREC); 2016:12-16.
159. Jiang S, Duan M, Wang L. Toward privacy-preserving symptoms matching in SDN-based mobile healthcare social networks. *IEEE Internet Things J*. 2018;5:1379.
160. Meng W, Choo K-KR, Furnell S, Vasilakos AV, Probst CW. Towards bayesian-based trust management for insider attacks in healthcare software-defined networks. *IEEE Trans Netw Serv Manag*. 2018;15(2):761-773.
161. Chaudhary R, Jindal A, Aujla GS, Kumar N, Das AK, Saxena N. LSCSH: Lattice-based secure cryptosystem for smart healthcare in smart cities environment. *IEEE Commun Mag*. 2018;56(4):24-32.
162. Varadharajan V, Tupakula U, Karmakar K. Secure monitoring of patients with wandering behavior in hospital environments. *IEEE Access*. 2018;6:11,523-11,533.
163. Li T-M, Liao C-C, Cho H-H, Chien W-C, Lai CF, Chao H-C. An e-healthcare sensor network load-balancing scheme using SDN-SFC. In: IEEE 19th International Conference on e-Health Networking, Applications and Services (HEALTHCOM); 2017:1-4.

164. Muhammad G, Alhamid MF, Alsulaiman M, Gupta B. Edge computing with cloud for voice disorder assessment and treatment. *IEEE Commun Mag*. 2018;56(4):60-65.
165. Rego A, Garcia L, Sendra S, Lloret J. Software defined network-based control system for an efficient traffic management for emergency situations in smart cities. *Futur Gener Comput Syst*. 2018;88:243.
166. Junior W, Silva B, Dias K. A systematic mapping study on mobility mechanisms for cloud service provisioning in mobile cloud ecosystems. *Comput Electr Eng*. 2018;16:256.
167. Silva F, Castillo-Lema J, Neto A, Silva F, Rosa P. Software defined ehealth networking towards a truly mobile and reliable system. In: IEEE 16th International Conference on e-Health Networking, Applications and Services (HEALTHCOM); 2014:560-564.

How to cite this article: Chaudhary R, Aujla GS, Kumar N, Chouhan PK. A comprehensive survey on software-defined networking for smart communities. *Int J Commun Syst*. 2025;38(1):e5296. doi:10.1002/dac.5296