VECTORAL PUBLISHING

Date of publication 10/01/2022, date of current version 10/01/2022.

VECTORAL manuscript Identifier 11.01-5/VECTORAL.2022.PUB

OPTIMIZING SERVICE FUNCTION CHAINING (SFC) FOR LATENCY-SENSITIVE APPLICATIONS IN SOFTWARE-DEFINED WIDE AREA NETWORKS (SD-WAN)

ARUNKUMAR VELAYUTHAM[1]

[1]Cloud Software Development Engineer and Technical Lead at Intel, Arizona, USA

Corresponding author: Velayutham, A.

ABSTRACT Service Function Chaining (SFC) in Software-Defined Wide Area Networks (SD-WAN) enables the arrangement of network services in a predefined sequence, such as firewalls, intrusion detection systems (IDS), and load balancers. However, latency-sensitive applications like Voice over IP (VoIP) and video conferencing demand stringent low-latency guarantees that are often compromised by the additional delay introduced by chaining multiple services. This paper examines advanced methods for optimizing SFC in SD-WAN to reduce latency, focusing on key strategies such as service function placement at network edges, dynamic and adaptive service orchestration, minimizing Virtual Network Function (VNF) processing overheads, and leveraging network function virtualization (NFV) and Software-Defined Networking (SDN) principles to enable real-time traffic optimization. The study further explores traffic steering mechanisms, Quality of Service (QoS) enforcement, and the use of hardware acceleration techniques like DPDK and SR-IOV to improve performance. The proposed solutions are designed to significantly reduce delays introduced by SFC while maintaining the security and efficiency necessary for real-time applications in SD-WAN environments.

INDEX TERMS latency optimization, network function virtualization, QoS enforcement, SD-WAN, service function chaining, traffic steering, VNF processing

I. INTRODUCTION

Latency in network technology refers to the delay between the time a piece of information is sent from the source and the time it is received at its destination. This delay is crucial in applications where real-time data transmission is important, such as video surveillance, online gaming, and live video streaming. Latency is typically measured in units of time, often in seconds or milliseconds. The exact measurement of latency can be challenging, primarily due to the difficulty of perfectly synchronizing the clocks between the source and the destination devices. In practice, several techniques can be employed to approximate latency, one of which is using a timestamp overlay on video frames to estimate the time delay from capture to rendering (Bhamare et al., 2016; Zu et al., 2019).

Latency is influenced by various components, each contributing to the overall delay experienced in data transmission. One primary component is transmission latency, which represents the time required to transmit all bits of a data packet onto the transmission medium. This can be mathematically expressed as $T_{\text{tx}} = \frac{L}{R}$, where L is the size of the data packet in bits, and R is the transmission rate of the communication link, measured in bits per second (bps). Larger data packets or slower transmission rates result in higher transmission latency. Another key component is propagation latency, which is the time taken for the signal to travel from the source to the destination (Zou et al., 2018). This is determined by the physical distance between the two points and the speed of signal propagation in the medium, typically close to the speed of light. The propagation latency is given by $T_{\text{prop}} = \frac{d}{v}$, where d is the distance between the source and the destination, and v is the speed at which the signal propagates, which can vary depending on the medium, such as fiber optic cables or electrical wires.

In addition to transmission and propagation latency, processing latency also plays a critical role in the total delay. Processing latency refers to the time required for devices, such as routers and switches, to examine and forward data packets.

The time taken for such operations can vary depending on the complexity of the network equipment and the algorithms used for routing and switching decisions. High-performance routers generally have lower processing latencies due to their more efficient packet handling capabilities. Finally, queuing latency is the time that a packet spends waiting in a queue before being transmitted. In congested networks, queuing latency can increase significantly, as multiple packets may need to be processed in sequence, resulting in higher delays.

The overall end-to-end latency in a network system can be considered the sum of these individual components: $T_{\text{total}} = T_{\text{tx}} + T_{\text{prop}} + T_{\text{proc}} + T_{\text{queue}}$. Each component contributes to the delay experienced in sending data from one point to another. For a video surveillance system, this total latency would encompass the time from when an image is captured by the camera, processed by the system, transmitted over the network, and finally rendered on a display device. Given the potential for delay at each stage, understanding and minimizing these factors is crucial for maintaining high-quality video monitoring, especially in situations where real-time response is required, such as security and surveillance applications. For instance, if a camera captures an image frame, the processing at the camera, the transmission over the network to a monitoring station, and the rendering of the image all introduce delays that collectively determine the latency experienced by the user (Guo et al., 2016).

One practical method for estimating end-to-end latency in such systems is through timestamp overlay techniques. By superimposing a timestamp on each video frame at the time of capture, it becomes possible to measure the time difference between when the frame is displayed on a monitoring device and when it was initially captured. This approach provides an approximate measure of the system's latency, allowing for analysis and optimization. Although it may not perfectly account for every source of delay in cases where there is slight clock drift between devices, it offers a useful approximation of performance.

Real-time communication applications have become essential in many industries, including education, healthcare, gaming, and industrial automation. These applications enable users to exchange data, audio, video, or control signals instantly, with minimal delay between transmission and reception. The success of real-time communication (RTC) applications hinges on the performance of their underlying components and system architectures . To function effectively, they rely on carefully designed systems that prioritize low latency, high availability, and efficient data transmission. Understanding the components and architectures that power RTC applications helps explain how they achieve these objectives.

At the heart of most real-time communication applications are several fundamental components, including clients, servers, network protocols, and media processing systems. Clients refer to the user-facing devices or applications that send and receive data in real time. These include computers, smartphones, or specialized hardware, such as video conferencing systems or gaming consoles. On the other end, servers act as intermediaries, facilitating the communication between clients. Depending on the application's architecture, servers can perform different roles. For example, in peer-to-peer communication systems, the server is used only to establish the initial connection between clients, after which data flows directly between them. In contrast, in client-server architectures, the server is heavily involved in managing all data transmissions between clients.

Network protocols play a critical role in ensuring data is transferred efficiently and with minimal delay. Most real-time communication applications use either Transmission Control Protocol (TCP) or User Datagram Protocol (UDP) for data transmission. TCP is a connection-oriented protocol that ensures reliable data transfer by using acknowledgments and retransmissions in case of packet loss. While TCP guarantees that all packets arrive in order, its overhead can lead to increased latency, making it less suitable for certain real-time applications, such as live video or gaming, where speed is prioritized over perfect reliability. In such cases, UDP is preferred because it is a connectionless protocol that prioritizes speed by sending packets without waiting for acknowledgments or retransmissions. Although UDP can result in some packet loss, the reduced latency makes it ideal for applications like live video streaming or online gaming, where minor data loss is often imperceptible to the user (Guo et al., 2016; Hantouti et al., 2018).

Another crucial component of real-time communication systems is the media processing system, which is responsible for encoding, decoding, compressing, and decompressing audio and video data. In video conferencing and live streaming applications, this is important because the raw media data is often too large to transmit over the network without optimization. Codecs are the algorithms used for this purpose. Some popular codecs include H.264 for video and Opus for audio. These codecs compress data while maintaining acceptable quality, ensuring that real-time media can be streamed with minimal latency even on networks with limited bandwidth. Additionally, echo cancellation, noise suppression, and bandwidth estimation mechanisms are integrated into media processing systems to enhance the quality of real-time audio and video communications.

System architectures for real-time communication applications can vary depending on the use case, with two primary models being peer-to-peer (P2P) and client-server architectures. In a peer-to-peer architecture, clients connect directly to one another to exchange data without routing it through a central server (Jani, 2021). This architecture is commonly used in applications such as VoIP and file-sharing systems because it minimizes the number of intermediate hops between clients, reducing latency. The challenge in P2P systems is the network address translation (NAT) traversal, which can complicate establishing direct connections between clients behind firewalls or routers. To address this, techniques like STUN (Session Traversal Utilities for NAT) and TURN (Traversal Using Relays around NAT) are used to help peers

Category	Examples
Video Conferencing	Zoom, Microsoft Teams, Google Meet
Voice over IP (VoIP)	Skype, WhatsApp, Google Voice
Online Gaming	Fortnite, Call of Duty, League of Legends
Live Video Streaming	Twitch, YouTube Live, Facebook Live
Remote Control and Teleoperation Systems	Drone control, robotic surgery systems
Virtual and Augmented Reality	VR/AR collaborative environments
Financial Trading Platforms	Stock trading, cryptocurrency exchanges
Instant Messaging Platforms	Slack, Discord, WhatsApp
Telemedicine and Remote Diagnostics	Real-time doctor consultations
Industrial Automation and Monitoring Systems	SCADA systems in manufacturing
Real-Time Collaboration Software	Google Docs, Miro
Emergency Communication Systems	911 services, disaster response systems
Interactive Online Learning	Real-time lectures, virtual classrooms
Remote Desktop Applications	TeamViewer, AnyDesk
Smart Home Control Systems	Real-time control of IoT devices

TABLE 1. Common Real-Time Communication Applications

establish direct connections or, in cases where NAT traversal fails, route traffic through a relay server (Leivadeas et al., 2020).

In contrast, client-server architectures centralize communication through a server, which coordinates and relays data between clients. This model is often used in video conferencing platforms and online multiplayer games, where the server manages all communication, ensuring that every client receives the necessary data. Although this introduces an additional hop in the communication process, client-server architectures offer better scalability and control, allowing for features such as centralized recording, authentication, and load balancing. In large-scale real-time applications, client-server architectures can leverage cloud-based infrastructure to dynamically scale resources based on demand, providing resilience and ensuring that latency remains low even with increasing user loads.

To mitigate the latency introduced by server involvement, some real-time communication applications implement hybrid architectures. These combine elements of both peer-to-peer and client-server models. For instance, in video conferencing applications, a server may handle signaling and manage the initial connection setup, while the media streams flow directly between clients in a peer-to-peer fashion. If direct peer-to-peer communication is not feasible due to NAT restrictions or poor network conditions, the media stream can be relayed through the server. This flexibility helps balance the efficiency of peer-to-peer communication with the robustness of a centralized server architecture (Pei et al., 2018).

At the protocol level, many real-time communication applications leverage WebRTC (Web Real-Time Communication), an open standard that enables peer-to-peer communication directly between browsers and mobile applications. WebRTC supports both audio and video communications and is widely used in modern video conferencing and instant messaging applications. One of WebRTC's key features is its ability to handle NAT traversal automatically using STUN and TURN, making it easier to establish peer-to-peer connections across a wide range of network configurations. Additionally, WebRTC provides built-in encryption, ensuring that all communications are secure, which is critical for protecting user privacy in real-time applications.

For real-time communication applications to function efficiently, they must also handle the challenge of jitter, or the variation in packet arrival times. Jitter can be caused by network congestion, leading to inconsistent delays that degrade the quality of communication. To counteract jitter, RTC applications often implement jitter buffers, which temporarily store incoming packets and reorder them before playback. While jitter buffers introduce a small amount of delay, they smooth out packet arrival times and improve the user experience by ensuring that media streams play back without interruptions.

In addition to jitter management, packet loss is another issue that real-time communication systems must address. When data packets are lost during transmission, the quality of the communication can suffer, especially in video and audio streaming. Some RTC applications use Forward Error Correction (FEC) techniques, which add redundant data to the transmission, allowing the receiver to recover lost packets without needing retransmission. FEC is useful in environments where retransmission would cause unacceptable delays, such as live video broadcasting or online gaming.

In network systems, latency is a critical factor that determines the performance of real-time communication applications such as video conferencing, online gaming, and telemedicine. Latency refers to the time delay between the transmission of a data packet from the source and its reception at the destination. Various technical parameters influence this delay, including physical distance, transmission protocols, packet processing, queuing mechanisms, and hardware limitations. In this detailed discussion, we explore the main parameters that affect latency, with a focus on their technical characteristics and how they contribute to overall delay.

Propagation delay is the time taken for a signal to travel from the sender to the receiver through a communication medium. This delay is a function of the distance between

the source and the destination and the speed at which signals travel through the medium. In most cases, the signal speed is close to the speed of light in a vacuum, approximately 3×10^8 m/s, but this speed can be slower in materials such as fiber optic cables or copper wires. Propagation delay can be expressed mathematically as

$$T_{\text{prop}} = \frac{d}{v}$$

where d is the distance between the sender and the receiver, and v is the propagation speed in the medium. For example, in fiber optic cables, the speed of light is reduced by approximately 30%, meaning the effective propagation speed is closer to 2×10^8 m/s. In long-distance communications, such as transcontinental connections, this delay can be significant. For instance, transmitting data over a 5000 km fiber link would incur a propagation delay of roughly 25 ms. As a result, propagation delay becomes relevant in global networks, such as satellite communications, where the large distances introduce considerable latency.

Transmission delay refers to the time required to push all the bits of a packet onto the transmission medium. This delay depends on the size of the data packet (denoted as L) and the bandwidth of the communication link (denoted as R). The transmission delay can be expressed by the formula

$$T_{\text{tx}} = \frac{L}{R}$$

where L is the size of the packet in bits, and R is the link's transmission rate, measured in bits per second (bps). Larger packets or slower transmission rates result in higher transmission delay. For example, transmitting a 1 MB packet (8×10^6 bits) over a 100 Mbps link would take

$$T_{\text{tx}} = \frac{8 \times 10^6}{100 \times 10^6} = 0.08\,\text{seconds} = 80\,\text{ms}$$

Transmission delay is important in scenarios where large data payloads are involved, such as streaming high-definition video. Reducing the packet size or increasing the bandwidth can help mitigate this form of latency.

Processing delay occurs when intermediate network devices, such as routers and switches, analyze and forward data packets. This involves checking the packet header, determining the routing path, and possibly performing additional tasks, such as error checking or encryption. The time required for processing depends on the complexity of the routing algorithm and the performance of the hardware involved. Modern routers use cut-through switching, which reduces processing delay by forwarding packets as soon as the destination address is determined, without waiting for the entire packet to be received. However, more sophisticated routers may engage in store-and-forward switching, which processes the entire packet before forwarding, resulting in higher latency.

Processing delays are generally low, in the order of microseconds, but they can accumulate as packets traverse multiple routers in large-scale networks. The performance of the processing unit, the efficiency of the routing protocols (e.g., OSPF, BGP), and the packet inspection mechanisms (e.g., firewall rules, deep packet inspection) all influence the processing delay.

Queuing delay arises when data packets wait in a queue before being transmitted through the network. This occurs when the arrival rate of packets exceeds the transmission capacity of the network device, such as a router or switch. Queuing delays are highly variable and depend on the traffic load, network congestion, and queuing discipline employed by the network devices. In heavily congested networks, queuing delays can be significant, leading to noticeable increases in latency.

The total queuing delay can be modeled using Little's Law, which relates the average number of packets in the queue (N) to the average arrival rate (λ) and the average waiting time in the queue (W):

$$N = \lambda \times W$$

This equation suggests that, as the traffic load increases, the queuing delay will grow linearly. To manage queuing delay, many networks implement Quality of Service (QoS) mechanisms that prioritize certain types of traffic (e.g., voice or video) to ensure that latency-sensitive applications are less affected by congestion. Common queuing mechanisms include First In, First Out (FIFO), Priority Queuing (PQ), and Weighted Fair Queuing (WFQ), each offering different trade-offs between fairness and performance under load.

Packet loss can occur due to network congestion, faulty hardware, or errors introduced during transmission. When a packet is lost, a retransmission is required, introducing additional delay. Retransmission delay is problematic in real-time applications, as it can disrupt the flow of communication and degrade the user experience. For example, in a video conferencing application, packet loss could result in noticeable glitches in the audio or video stream. In protocols such as TCP, retransmissions are handled automatically through acknowledgments and timeouts. However, this process introduces delays, as the sender must wait for confirmation that the packet was successfully received before sending the next one (Sun et al., 2020; Zamani & Sharifian, 2018). This delay increases as the number of retransmissions grows in networks with high packet loss rates.

On the other hand, protocols like UDP, which is often used in real-time applications such as video streaming or online gaming, do not implement retransmissions, allowing for faster transmission but at the cost of potential packet loss. In some cases, applications implement Forward Error Correction (FEC) to recover from packet loss without requiring retransmission. FEC works by sending redundant data along with the original data, allowing the receiver to reconstruct lost packets if enough of the redundant information is received. While FEC introduces additional overhead, it

Parameter	Formula	Example Values	Description
Propagation Delay	$T_{\text{prop}} = \frac{d}{v}$	$d = 5000\,\text{km}, v = 2 \times 10^8\,\text{m/s}$	Propagation delay is the time taken for a signal to travel from the sender to the receiver. The speed of light is slower in mediums like fiber optic cables. For example, transmitting data over a 5000 km link would introduce a delay of 25 ms.
Transmission Delay	$T_{\text{tx}} = \frac{L}{R}$	$L = 1\,\text{MB}, R = 100\,\text{Mbps}$	Transmission delay refers to the time required to push all the bits of a packet onto the transmission medium. Larger packets or slower transmission rates result in higher delays. For instance, a 1 MB packet over a 100 Mbps link would take 80 ms.
Processing Delay	–	–	Processing delay is the time required by intermediate network devices, such as routers, to analyze and forward data packets. It depends on hardware efficiency and protocol complexity. Processing delays are typically in the microsecond range but accumulate across multiple hops.
Queuing Delay	$N = \lambda \times W$	$N = 10, \lambda = 2\,\text{packets/sec}, W = 5\,\text{sec}$	Queuing delay occurs when packets wait in line before being transmitted through the network. It can be modeled by Little's Law, which suggests that delays increase with higher traffic loads and longer wait times.

TABLE 2. Parameters influencing latency in real-time communication applications

helps reduce the impact of packet loss on latency-sensitive applications.

The performance of the hardware used in network communication also affects latency. Routers, switches, and other network devices must process and forward packets quickly to minimize delay. The speed and capacity of the hardware directly influence how fast packets can be processed and transmitted. For example, routers with higher processing power can handle more packets simultaneously, reducing queuing and processing delays. Similarly, switches with high port speeds and low internal latency contribute to lower transmission delays.

Congestion occurs when the volume of data being transmitted exceeds the capacity of the network, leading to packet loss, increased queuing delays, and slower transmission speeds. In congested networks, packets may be dropped, requiring retransmission, or may experience long queuing delays as they wait for transmission. Managing network congestion is crucial for reducing latency, and various techniques such as traffic shaping, load balancing, and congestion control algorithms (e.g., TCP congestion control) are employed to mitigate its effects.

With the increasing reliance on real-time communication applications such as VoIP and video conferencing, network infrastructures must adapt to meet the strict latency requirements these services demand. Real-time applications are sensitive to even small delays, with noticeable degradation in performance as latency increases. In wide area networks (WANs), where traffic travels across multiple network segments, latency becomes a critical factor affecting the quality of service.

Traditional WAN architectures have relied on centralized data centers to handle network functions like security enforcement, traffic optimization, and packet inspection. This centralized model forces traffic to travel long distances to reach these data centers, resulting in significant propagation delays. These delays negatively impact latency-sensitive applications, which require near-instantaneous transmission and reception. Additionally, each network service applied—such as firewalling, deep packet inspection, or encryption—introduces further processing delays, adding to the overall latency.

Software-Defined Wide Area Networks (SD-WAN) have emerged to address some of these limitations by offering centralized control and dynamic, application-aware routing. SD-WAN optimizes the traffic flow based on real-time network conditions. However, as SD-WAN deployments begin to incorporate Service Function Chaining (SFC), new latency challenges arise. SFC allows for the sequential application of multiple Virtual Network Functions (VNFs), such as security, optimization, and monitoring services, along the data path. While SFC enables the flexible delivery of services, each function introduces additional processing delays, which can significantly impact real-time applications.

The key issue with SFC is that each hop in the service chain—whether it be firewalls, intrusion detection systems, or traffic shaping—adds to the cumulative latency. For applications like VoIP and video conferencing, these delays result in degraded service quality, such as audio and video lag, jitter, and increased packet loss. The sequential nature of SFC, where each service must be applied in order, compounds this issue. Even in SD-WAN architectures, where routing is optimized for performance, the additional service processing introduces delays that real-time applications cannot tolerate.

Latency-sensitive applications require very low end-to-end latency for acceptable performance. Services like online gaming, financial trading, and telemedicine have strict performance requirements, where even small delays can lead to poor outcomes. Introducing multiple service hops in the data path adds further challenges. While the functions in the

chain—such as firewalls or traffic optimization—are necessary for network security and efficiency, they conflict with the need for minimal latency. This creates a fundamental tension between applying critical network services and maintaining low-latency performance.

Besides processing delays from SFC, other factors contribute to overall latency in SD-WAN environments. These include propagation delays due to geographic distance, transmission delays related to available bandwidth, and queuing delays at network nodes. As the demand for real-time applications grows, managing these various sources of delay becomes increasingly complex. Although SD-WAN offers improvements over traditional WAN models, the additional delays introduced by SFC pose a critical problem for applications that depend on minimal latency to function correctly.

II. BACKGROUND

Software-Defined Wide Area Network (SD-WAN) is a transformative approach in modern networking, leveraging the principles of Software-Defined Networking (SDN) to abstract the control plane from the physical infrastructure. By decoupling network management from underlying hardware, SD-WAN enables centralized control, policy enforcement, and advanced traffic management. It intelligently directs traffic across various transport networks—such as Multiprotocol Label Switching (MPLS), broadband internet, and LTE/5G—according to predefined policies and real-time conditions. This programmability enhances the efficiency of bandwidth utilization and significantly improves both the performance of applications and the overall cost-effectiveness compared to traditional WAN architectures.

The SD-WAN architecture is designed around several core functionalities. First, it provides centralized management through a controller, which enables network administrators to deploy and modify policies globally while receiving real-time feedback on network performance. Second, it supports dynamic path selection, whereby the system continuously assesses the health of the available transport links and selects optimal paths for data transmission based on performance metrics, such as bandwidth, latency, and packet loss. These capabilities ensure that SD-WAN delivers a higher degree of reliability and quality of service (QoS) for critical and real-time applications.

A key strength of SD-WAN lies in its ability to handle traffic dynamically. By prioritizing traffic based on the nature of applications and their requirements—such as prioritizing business-critical applications over less sensitive traffic—SD-WAN ensures that mission-critical services receive the bandwidth and network conditions necessary to operate effectively. This capability is advantageous for latency-sensitive applications, which demand low-latency, high-performance connections. In this context, SD-WAN's ability to steer traffic over the least congested and lowest-latency links enhances application performance and minimizes the risk of service disruption, a critical consideration in fields such as telemedicine, financial services, and cloud-based enterprise applications.

The use of multiple transport technologies provides another layer of flexibility and cost management. MPLS, often favored for its guaranteed QoS, tends to be more expensive compared to broadband or cellular connectivity. SD-WAN allows organizations to strike a balance between high-performance MPLS and more cost-effective broadband or 4G/5G links, thereby optimizing operational expenses without sacrificing service quality. Moreover, SD-WAN's ability to rapidly scale and adapt to changing business needs makes it an appealing choice for organizations seeking to enhance their agility while maintaining strict control over network performance and security.

Service Function Chaining (SFC) is an essential concept in modern SD-WAN architectures, enabling dynamic and flexible application of network services to specific traffic flows. SFC facilitates the steering of data packets through a sequence of network services, often implemented as Virtualized Network Functions (VNFs), which can include stateful firewalls, intrusion detection systems (IDS), WAN optimization, and load balancers. In a traditional network, these services would often be deployed as dedicated hardware devices, but the virtualized nature of VNFs allows for greater flexibility and efficiency, as well as the potential for automation and scalability.

Each VNF in an SFC performs a specific function, and the chain is constructed according to the needs of the traffic being handled. For instance, a packet may pass through a firewall for security inspection, followed by a WAN optimizer to ensure efficient bandwidth use, and finally through a load balancer to distribute traffic evenly across servers. This sequence is defined by policy and can vary depending on the type of traffic and the required service level agreements (SLAs). The programmability of SFC means that network administrators can dynamically adjust the chain of services in real-time based on changing network conditions or application requirements.

While the flexibility offered by SFC is a significant advantage, there are inherent challenges concerning the introduction of latency. Every VNF in the chain introduces processing delays, as each service inspects and potentially modifies the data packets before passing them on to the next service in the chain. This cumulative latency can become problematic, especially for latency-sensitive applications such as VoIP, online gaming, or real-time video streaming. As the number of VNFs in the chain increases, so does the overall delay, which may degrade the performance of these applications.

Optimizing the placement and order of VNFs within an SFC is therefore a critical task in SD-WAN environments when dealing with latency-sensitive applications. Proper optimization can help mitigate the latency introduced by each service in the chain, balancing the need for robust security and traffic management with the performance requirements of the underlying applications. Techniques such as VNF placement algorithms and intelligent traffic steering can help minimize these delays, ensuring that services are applied

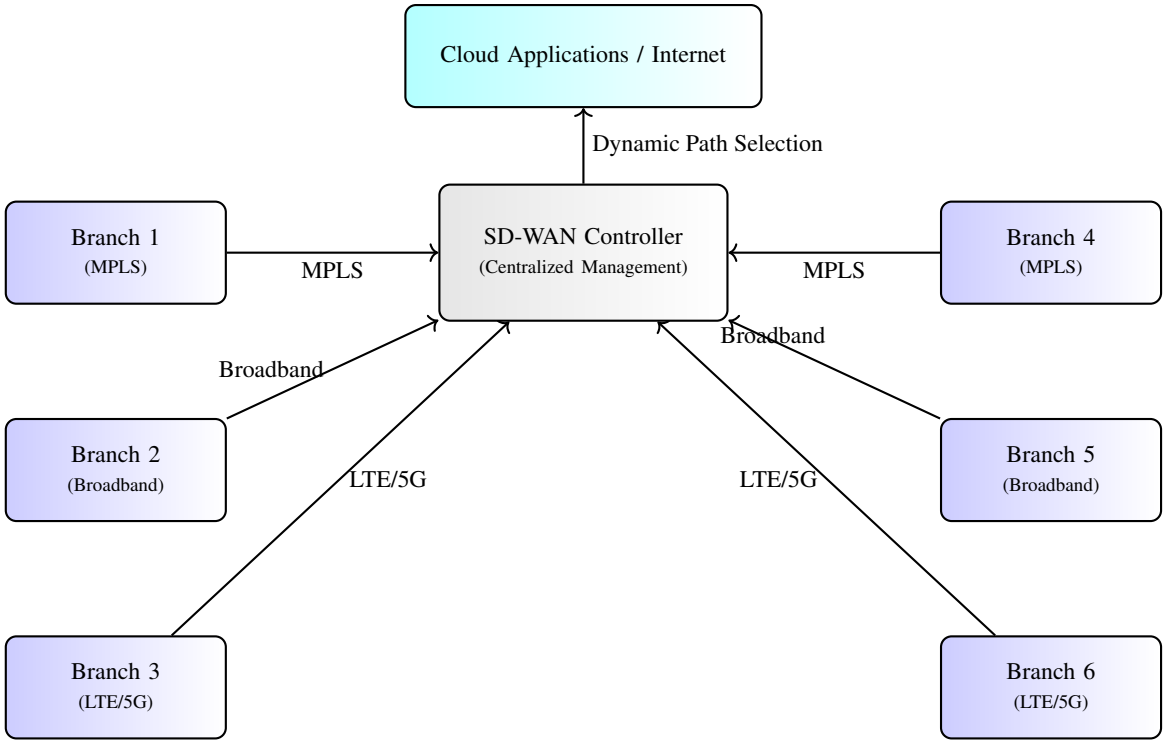

FIGURE 1. SD-WAN Architecture Overview: Centralized Management with Multiple Transport Technologies (MPLS, Broadband, LTE/5G) and Cloud/Internet Integration.

efficiently while adhering to the performance thresholds required by the end users.

Latency-sensitive applications impose strict requirements on network performance, demanding rapid packet transmission with minimal delays to ensure a seamless user experience. Examples of such applications include VoIP, video conferencing, online gaming, and real-time financial trading platforms, all of which depend on the timely delivery of data. For instance, in VoIP applications, end-to-end latency must remain below 150 milliseconds to maintain acceptable voice quality. Video conferencing applications may tolerate slightly higher latencies but are sensitive to jitter (the variability in packet arrival times) and packet loss, both of which can result in poor video and audio quality.

The performance degradation of latency-sensitive applications can manifest in several ways. In VoIP, excessive latency can cause noticeable delays in conversation, resulting in interruptions and difficulty in maintaining natural dialogue. Similarly, video conferencing applications may experience frozen video, out-of-sync audio, or reduced video quality due to high jitter or packet loss. For applications that involve real-time interactions, such as online gaming or financial trading, even small delays can lead to significant consequences, affecting gameplay or leading to financial losses due to delayed transactions.

In the context of SD-WAN environments that employ Service Function Chaining, managing latency becomes even more critical. The added processing time introduced by each VNF, while essential for security, optimization, and traffic management, can exacerbate latency issues. As such, it is imperative to minimize the number of service functions a packet must traverse or to optimize the VNF deployment to reduce their processing overhead. Additionally, intelligent traffic routing, based on real-time network conditions, can help ensure that traffic is steered through the least congested and lowest-latency paths.

Another important consideration for latency-sensitive applications is the network's ability to maintain consistent performance under fluctuating conditions. Factors such as link congestion, packet loss, and fluctuating bandwidth can dramatically affect real-time applications, causing jitter or packet reordering. SD-WAN addresses these challenges by constantly monitoring the health of available paths and dynamically adjusting the routing of traffic to avoid congested or unreliable links. By doing so, SD-WAN ensures that latency-sensitive applications receive the network resources necessary to maintain high performance.

However, the challenge of meeting the stringent requirements of latency-sensitive applications in SD-WAN environments is not limited to routing decisions alone. It also extends to the optimization of the underlying infrastructure. For example, VNFs used in SFC must be highly optimized for performance, with minimal processing delays. Moreover, strategies such as VNF offloading to specialized hardware (e.g., network processors or smart NICs) can help reduce the latency introduced by software-based network functions, providing a balance between the need for dynamic service chaining and the performance requirements of real-time ap-

plications.

III. CHALLENGES IN SFC FOR SD-WAN

VNF processing overheads in SFC for SD-WAN stem from the fundamental architectural differences between traditional hardware-based network appliances and the software-based nature of VNFs. VNFs are deployed on commodity hardware, which introduces inherent inefficiencies due to virtualization layers. These layers, typically involving hypervisors or container runtimes, create a more complex interaction between the hardware and the VNF compared to traditional, dedicated hardware appliances. The hypervisor or container runtime must manage resource allocation across multiple VNFs or applications, leading to a reliance on context switching within the operating system. In a high-throughput network environment where multiple VNFs are chained together, context switching can result in considerable processing overhead. Each time the CPU switches between different processes or VNFs, it must store the current state of the executing VNF, load the state of the new VNF, and resume processing. This switching increases processing latency and reduces the overall throughput of the system, as the CPU is effectively spending time managing transitions rather than executing tasks.

In addition, the I/O bottlenecks that arise in virtualized environments can further degrade VNF performance. Network traffic typically passes through several layers of software abstraction before being processed by a VNF. For example, traffic entering a virtualized environment may first be handled by a virtual switch or network interface controller (NIC), which is controlled by the hypervisor. The traffic is then passed to the appropriate VNF for processing. Each layer introduces additional latency, as the data must traverse from one layer to the next. This is evident when VNFs are deployed across different virtual machines (VMs) or containers, as traffic must be routed between these isolated environments via the hypervisor. This routing is often managed by the virtualized network stack, which adds further delay due to the increased complexity of managing multiple isolated environments on the same hardware (Leivadeas et al., 2020; Pei et al., 2018).

CPU limitations also play a significant role in VNF processing overheads, especially when the VNFs are tasked with resource-intensive operations such as deep packet inspection (DPI), encryption, or traffic analysis. These operations are computationally expensive and can saturate the CPU if the hardware is not optimized for such tasks. In traditional network appliances, specialized hardware such as application-specific integrated circuits (ASICs) or network processors are used to offload these tasks from the CPU, allowing for faster and more efficient processing. In contrast, VNFs running on general-purpose CPUs must rely solely on software-based processing, which is inherently less efficient. This leads to increased processing times and higher CPU utilization, further contributing to the overall latency of the service chain.

The problem of I/O bottlenecks becomes even more pronounced when considering the need for efficient packet processing. Each packet entering the system must be classified, processed, and forwarded to the next VNF in the service chain. This processing often involves multiple operations, such as filtering, load balancing, encryption, or DPI, all of which require significant computational resources. The process of forwarding packets between VNFs also introduces overhead, as the packets must be queued, scheduled, and transmitted between different VMs or containers. The complexity of this process increases with the length of the service chain, as each additional VNF adds another layer of processing and forwarding, further increasing the overall latency.

Path selection and traffic steering in SD-WAN environments introduce additional challenges when combined with SFC. SD-WAN systems are designed to dynamically select the best path for traffic based on real-time network conditions, such as latency, jitter, and packet loss. However, when service function chaining is implemented, the presence of VNFs in the path selection process complicates the situation. Each VNF introduces its own processing delays, and these delays can vary significantly depending on the type of VNF, the load on the system, and the nature of the traffic. For example, a VNF performing encryption may introduce significantly more latency than a VNF performing basic packet forwarding or filtering. As a result, the SD-WAN controller must account for both the network-induced latency (e.g., the delay caused by routing traffic across a congested link) and the service-induced latency (e.g., the delay introduced by the VNFs themselves).

The dynamic nature of SD-WAN further complicates traffic steering decisions. In traditional network environments, traffic steering decisions are often made based on static configurations or predefined policies. However, in an SD-WAN environment, these decisions are made dynamically based on real-time data collected from the network. This means that the SD-WAN controller must continually monitor network conditions, such as link latency, jitter, and packet loss, as well as the performance of the VNFs in the service chain. This real-time monitoring requires the collection and processing of a large amount of data, which introduces its own overhead and complexity. Furthermore, the controller must make traffic steering decisions in real-time, often within milliseconds, to ensure that latency-sensitive traffic is routed over the most optimal path.

Jitter, or the variability in packet delay, presents a significant challenge for latency-sensitive applications in the context of SD-WAN with SFC. Jitter can be introduced by variations in VNF processing times, as well as by fluctuations in network conditions. For example, a VNF may experience an increase in processing time if it becomes overloaded with traffic or if the underlying hardware becomes saturated. Similarly, network conditions may fluctuate due to congestion, link failures, or other factors. These variations can lead to inconsistent packet delays, which are problematic for applications such as VoIP, video conferencing, or online gaming, where consistent packet timing is crucial for main-

Technical Challenge	Cause	Impact
VNF Processing Overheads	VNFs are hosted on generic hardware and use hypervisors or containers, leading to overheads from context switching, I/O bottlenecks, and CPU limitations.	Increased delay for latency-sensitive applications, as each VNF in the chain processes traffic and introduces additional processing time.
Path Selection and Traffic Steering	Multiple transport networks in SD-WAN environments with dynamic path selection can be affected by VNF processing in SFC.	Service chains introduce additional delays, impacting the selection of the most optimal path and potentially increasing latency, packet loss, or jitter.
Service Placement and Proximity	Centralized hosting of VNFs in data centers increases round-trip times, especially for geographically distributed applications.	Long-distance traffic transport increases latency, resulting in slower performance for applications that require fast processing.

TABLE 3. Challenges in SFC for SD-WAN

taining performance. The SD-WAN controller must therefore not only minimize average latency but also reduce jitter by selecting paths and VNFs that introduce minimal variability in processing times and transmission delays.

Service placement and proximity are also critical factors in the performance of SFC within SD-WAN. In traditional architectures, VNFs are often deployed in centralized data centers, which can introduce significant delays due to the physical distance between the data center and the traffic source or destination. For example, if a user in one geographic region is accessing a service hosted in a data center located in a different region, the traffic must traverse long distances to reach the VNFs in the service chain, be processed, and then continue to the destination. This introduces considerable round-trip delay, which can degrade the performance of latency-sensitive applications.

The problem of service placement is further complicated by the need to balance the proximity of VNFs with the computational resources required to support them. Deploying VNFs closer to the edge of the network, near the users and data sources, can reduce latency by shortening the distance that traffic must travel. However, edge deployments often come with limited computational resources, which may not be sufficient for resource-intensive VNFs such as those performing DPI, encryption, or large-scale traffic analysis. As a result, there is a trade-off between proximity and performance: deploying VNFs closer to the edge can reduce latency, but it may also limit the types of services that can be provided due to resource constraints.

Additionally, centralized service placement can result in inefficient traffic routing. In many cases, traffic must be backhauled to a central location for processing, even if more direct routing paths are available. This backhauling introduces unnecessary delays and consumes additional bandwidth, further degrading network performance. For example, traffic originating in one region may need to be routed to a centralized data center in another region for processing by a VNF, only to be routed back to the destination in the original region. This inefficient routing adds to the overall latency of the service chain and can significantly degrade the performance of latency-sensitive applications (Silvestro, 2019).

The cumulative impact of these challenges is significant, as they all contribute to the overall performance degradation of SFC within SD-WAN environments, especially for applications requiring real-time or near-real-time communication. The complexity of managing these factors increases as the scale and scope of the network grow, requiring intricate optimization strategies and real-time decision-making processes to minimize latency and maximize network performance.

IV. METHODS TO OPTIMIZE SFC FOR LOW-LATENCY APPLICATIONS

A. INTELLIGENT SERVICE FUNCTION PLACEMENT

Minimizing latency in Service Function Chains (SFCs) for real-time applications requires a strategic placement of Virtual Network Functions (VNFs) within the network infrastructure. VNFs, being software-based network functions such as firewalls, load balancers, or Network Address Translators (NATs), can be placed at various points in the network to optimize performance. Given that latency is a critical performance metric for many real-time applications—such as video conferencing, online gaming, or industrial automation—efficient placement of VNFs becomes paramount. One of the most effective strategies to reduce latency involves placing VNFs in geographically proximate locations to end users. This can be accomplished dynamically using Software-Defined Wide Area Networks (SD-WANs), where an SD-WAN controller manages the optimal VNF placement at edge nodes close to users. This approach not only minimizes the delay caused by data transmission over long distances but also enhances the network's agility and responsiveness (Trajkovska et al., 2017).

Edge computing, which pushes processing tasks closer to the user, plays a significant role in reducing geographic latency. Placing VNFs that are highly sensitive to delay—such as firewalls, NATs, or traffic optimizers—at edge nodes reduces the round-trip time (RTT) for data packets. RTT is a crucial factor for real-time applications, where even millisecond-level delays can degrade the user experience. For instance, in applications such as video streaming or online gaming, high RTT can lead to buffering or lag, which negatively impacts the performance of these services. By strategically placing latency-sensitive VNFs at the edge, close to both the data source and the destination, the network can process traffic more efficiently, leading to a smoother and

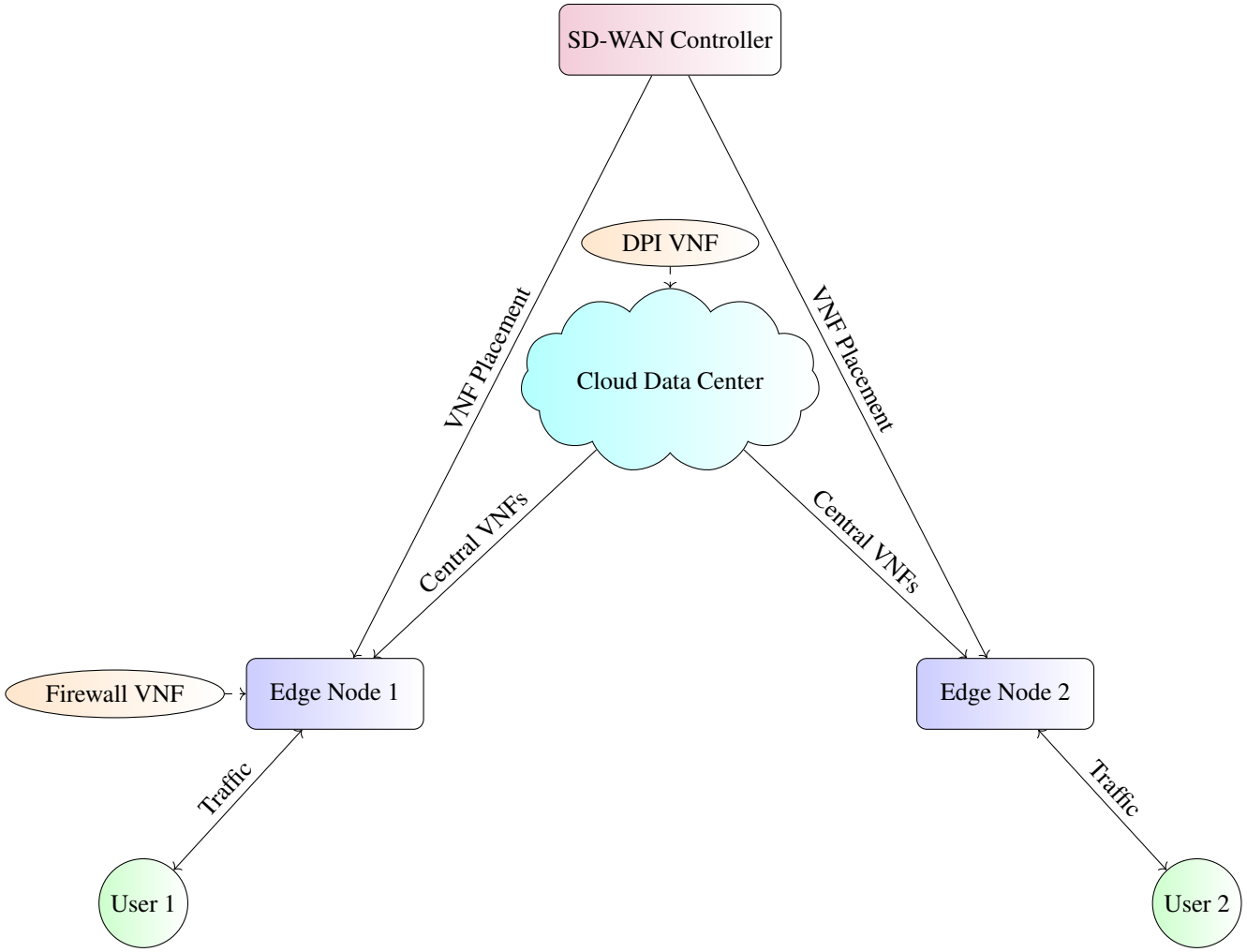

FIGURE 2. Hybrid VNF Placement using SD-WAN for Real-Time Applications. Latency-sensitive VNFs are placed at edge nodes closer to users for minimal delay, while non-latency-sensitive VNFs are processed in centralized cloud data centers. The SD-WAN controller dynamically manages VNF placement based on network conditions and latency requirements.

more reliable user experience.

A hybrid approach, combining both edge and cloud resources, can further optimize VNF placement. This method allows for the distribution of VNFs based on their latency requirements. Latency-sensitive VNFs are placed at edge locations to process traffic with minimal delay, while less latency-critical VNFs, such as those involved in deep packet inspection or non-real-time analytics, can be placed in centralized cloud data centers. This division balances the need for low latency with the efficient utilization of resources, as cloud data centers offer scalable compute and storage capabilities. In many cases, cloud resources are more cost-effective for handling high volumes of traffic that do not require immediate processing. Thus, offloading certain VNFs to the cloud while retaining latency-critical VNFs at the edge achieves a balance between performance and resource efficiency (Sun et al., 2020).

One of the central challenges in VNF placement involves determining the optimal locations for these functions within the network. Latency is not the only factor to consider; other variables such as bandwidth, processing capacity, and network congestion also play significant roles. A well-designed VNF placement strategy must account for these constraints in addition to latency. For instance, placing VNFs at an edge node may minimize latency, but if that node becomes congested with traffic or lacks the necessary compute resources to handle the VNF, the overall network performance could degrade. Therefore, the SD-WAN controller must have a comprehensive view of network conditions to dynamically adjust the placement of VNFs in real-time. This dynamic adaptability is a core advantage of SD-WAN technology, which provides centralized control and monitoring of the entire network infrastructure.

The SD-WAN controller's role is crucial in managing VNF placement for SFC optimization. It monitors real-time network metrics such as link latency, jitter, and packet loss, and adjusts the placement of VNFs accordingly. For instance, if a particular edge node experiences increased congestion

or processing delays, the SD-WAN controller can migrate the VNF to another edge node with lower latency or higher capacity. This dynamic adjustment ensures that the VNFs are always optimally placed to meet the demands of real-time applications. Moreover, SD-WAN technology enables multi-path routing, where traffic is distributed across multiple network paths to avoid congestion and reduce latency. This feature further enhances the flexibility and reliability of VNF placement strategies.

The placement of latency-sensitive VNFs also depends on the architecture of the underlying network. In many cases, edge nodes may be located in close proximity to access points, such as mobile base stations or local gateways, which provide an optimal location for hosting these functions. However, in more distributed network architectures, where edge nodes are sparse or located at significant distances from end users, the benefits of edge placement may be reduced. In such scenarios, a hybrid approach involving both edge and cloud resources becomes even more critical. For example, in rural or remote areas where edge nodes are less available, cloud data centers located in regional hubs may need to handle a larger portion of the network traffic. The SD-WAN controller can dynamically adjust the VNF placement based on the availability of edge and cloud resources, ensuring that the network continues to deliver low-latency services even in less densely populated areas.

The hybrid approach also enables better scalability and cost efficiency. Hosting all VNFs at the edge can become prohibitively expensive due to the higher costs associated with deploying and maintaining edge infrastructure. Centralized cloud data centers, by contrast, offer economies of scale and can handle large amounts of data traffic more cost-effectively. By offloading less latency-sensitive VNFs to the cloud, network operators can reduce the operational costs associated with edge computing, while still meeting the latency requirements of real-time applications through edge-hosted VNFs. This approach also allows for more efficient use of network resources, as VNFs can be dynamically allocated based on demand. During periods of low traffic, less critical VNFs can be consolidated into cloud data centers, freeing up edge resources for latency-sensitive applications (Li et al., 2019; Zhang et al., 2017).

In addition to optimizing latency and resource utilization, VNF placement strategies must also consider security and resilience. Many real-time applications require strict security measures to protect sensitive data, and VNFs such as firewalls and intrusion detection systems play a key role in safeguarding these applications. Placing security VNFs closer to the edge, where data enters and exits the network, can enhance the overall security posture by reducing the exposure of sensitive data as it traverses the network. Additionally, this approach can help mitigate distributed denial of service (DDoS) attacks or other malicious traffic, as threats can be detected and neutralized closer to their source.

However, placing VNFs at the edge also introduces certain risks with regard to resilience. Edge nodes are typically smaller and less robust than centralized cloud data centers, making them more vulnerable to hardware failures, power outages, or other disruptions. Therefore, it is essential to build redundancy into the VNF placement strategy. This can be achieved through techniques such as VNF replication, where multiple instances of a VNF are deployed across different edge nodes. In the event of a failure at one node, the SD-WAN controller can seamlessly reroute traffic to another node hosting a replica of the VNF, ensuring continuous service availability. Additionally, the use of containerized VNFs, which can be easily deployed and migrated across different nodes, enhances the flexibility and resilience of the network.

Moreover, the performance of VNFs is highly dependent on the underlying infrastructure, including both hardware and software components. Edge nodes equipped with powerful processors, high-speed memory, and efficient storage systems can execute VNFs with minimal delay, but this hardware can be expensive to deploy at scale. On the software side, the orchestration of VNFs is typically handled by a management framework such as NFV MANO (Management and Orchestration), which oversees the lifecycle of VNFs, including their deployment, scaling, and migration. Integrating NFV MANO with SD-WAN controllers provides a comprehensive solution for optimizing VNF placement, as it allows for seamless coordination between network infrastructure and application requirements.

The evolution of network architectures, such as the introduction of 5G, further amplifies the importance of efficient VNF placement. 5G networks, with their ultra-low latency and high throughput capabilities, will support a wide range of real-time applications, from autonomous vehicles to augmented reality. These applications require VNFs to be placed as close to the end user as possible to meet the stringent latency requirements of 5G. The deployment of Multi-access Edge Computing (MEC) in 5G networks provides an ideal platform for hosting VNFs at the edge, allowing for ultra-fast processing of data traffic. SD-WAN controllers, in combination with NFV MANO, can dynamically manage VNF placement in 5G networks, ensuring that real-time applications consistently meet their performance requirements.

B. DYNAMIC SFC ORCHESTRATION

Dynamic Service Function Chaining (SFC) orchestration offers a flexible and adaptive approach to handling network traffic, overcoming the limitations of static configurations by responding to real-time network conditions. Traditional static SFC configurations predefine a set path for traffic to follow through various network functions, such as firewalls, load balancers, and intrusion detection systems. While this approach works under predictable network conditions, it is suboptimal in dynamic environments where network traffic patterns, congestion levels, and the performance of virtualized network functions (VNFs) fluctuate. Static configurations can lead to inefficiencies such as prolonged latency, suboptimal resource utilization, and potential service degra-

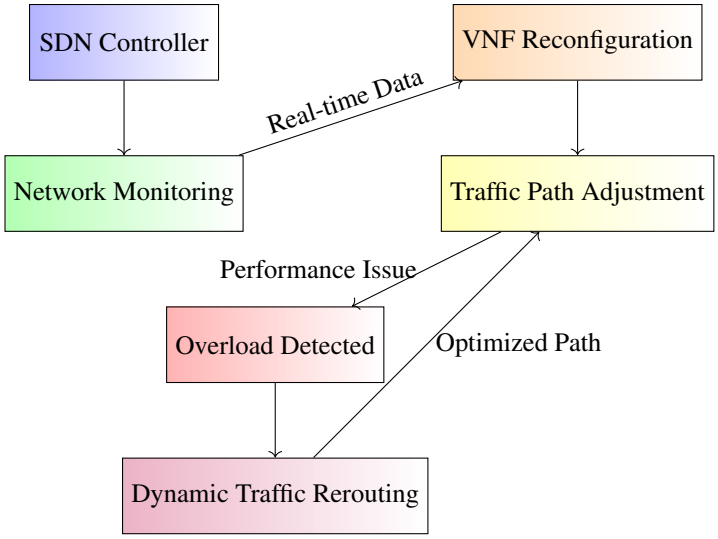

FIGURE 3. Dynamic SFC Orchestration Process. The SDN controller monitors network performance, adjusts traffic paths, and reconfigures VNFs based on real-time data. When overload or congestion is detected, traffic is dynamically rerouted to optimize performance.

dation, especially in latency-sensitive applications. Dynamic SFC orchestration addresses these challenges by adjusting the service chain in real time, leveraging continuous monitoring of network and VNF performance data (Pei et al., 2018; Silvestro, 2019).

Software-Defined Networking (SDN) controllers play a pivotal role in dynamic SFC orchestration by managing both network traffic and VNFs. These controllers utilize real-time data on performance metrics like latency, jitter, and packet loss to make informed decisions about rerouting traffic or bypassing non-essential services within the service chain. This capability is critical for optimizing the performance of applications where low latency and high reliability are required. The dynamic nature of this orchestration allows the service chain to adapt to varying conditions, ensuring efficient traffic paths, better resource allocation, and sustained service quality.

One of the primary advantages of dynamic SFC orchestration is its ability to handle VNF overload and network congestion in real time. VNFs, which are virtualized implementations of traditional network functions, run on commodity hardware and are subject to resource limitations such as CPU, memory, and bandwidth. In scenarios where a VNF experiences high CPU load or reaches a performance bottleneck, traffic passing through the VNF can experience delays, leading to increased latency and jitter. SDN controllers, which continuously monitor the performance of these VNFs, can detect such conditions and take corrective action by rerouting traffic to alternate VNFs that are not under stress. This ensures that traffic is processed efficiently and that the overall quality of service (QoS) is maintained.

In addition to rerouting traffic to mitigate congestion, dynamic SFC orchestration can also bypass certain VNFs when they are deemed non-essential for specific traffic flows. For instance, not all types of traffic require the same level of security or quality checks. A video streaming service may not need to pass through a deep packet inspection (DPI) VNF, which is crucial for detecting security threats in web traffic but unnecessary for certain multimedia streams. By dynamically bypassing VNFs that are irrelevant to the current traffic type, the SDN controller can reduce processing overhead and further minimize latency.

Another important factor influencing dynamic SFC orchestration is the constant fluctuation in network conditions. Changes in link bandwidth, network congestion, or fluctuating user demand can all impact the performance of service chains. In response to these changes, dynamic orchestration mechanisms can adjust the paths that traffic takes through the network. For example, if a particular link experiences congestion, the SDN controller can reroute traffic through an alternative path with more available bandwidth, reducing latency and packet loss. By continuously monitoring network conditions and adjusting service chains in real time, dynamic SFC orchestration ensures that network resources are used optimally and that service level agreements (SLAs) are met.

A key enabler of dynamic SFC orchestration is the integration of telemetry and analytics within the SDN architecture. Telemetry tools collect real-time performance data from the network and VNFs, providing the SDN controller with the necessary insights to make informed orchestration decisions. These tools measure key performance indicators (KPIs) such as throughput, packet loss, latency, jitter, and the utilization of computational resources on VNFs. Using this data, the SDN controller can apply advanced analytics and machine learning algorithms to predict potential performance degradation and proactively adjust the service chain before issues arise. For example, machine learning models can analyze historical traffic patterns to predict periods of high demand and preemptively reroute traffic to avoid overloading certain network segments or VNFs.

Dynamic SFC orchestration also benefits from the programmability of SDN, which decouples the control plane from the data plane in network devices. This separation allows the SDN controller to have a global view of the network and its resources, enabling more intelligent and efficient traffic management. In static configurations, service chains are often hardcoded into the network, making it difficult to adjust them in response to changing conditions. With dynamic orchestration, the SDN controller can reprogram the network on the fly, introducing new paths or VNFs into the service chain without requiring manual intervention. This programmability not only enhances the flexibility of the network but also reduces the operational complexity of managing service chains.

The use of Network Function Virtualization (NFV) in conjunction with dynamic SFC orchestration further enhances the adaptability of the network. NFV allows network functions to be virtualized and deployed on general-purpose hardware, making it easier to scale and manage network functions in response to changing demands. For example, if the SDN controller detects an increase in traffic that requires additional processing capacity, it can instantiate new VNFs on the fly, adding them to the service chain dynamically. This capability ensures that the network can scale elastically to meet user demand, providing a more efficient and cost-effective solution compared to traditional hardware-based network appliances.

One challenge associated with dynamic SFC orchestration is ensuring that reconfigurations do not introduce new inefficiencies or unintended consequences. Frequent changes to the service chain, especially in large-scale networks, can result in oscillations or instability if not managed carefully. For instance, rapidly switching traffic between VNFs or network paths can lead to increased jitter or out-of-order packet delivery, which may degrade the performance of certain applications. To mitigate this risk, SDN controllers often implement decision-making algorithms that consider not only current network conditions but also the potential impact of reconfiguration on the overall system stability. These algorithms aim to strike a balance between optimizing performance and minimizing disruptions to the network.

Security considerations also play a significant role in dynamic SFC orchestration. As service chains are reconfigured dynamically, it is essential to ensure that security policies are consistently applied across the network. For example, if traffic is rerouted to an alternative VNF, the same security checks and protections must be in place to prevent vulnerabilities from being introduced into the network. To address this concern, SDN controllers can enforce security policies programmatically, ensuring that traffic passing through the network adheres to predefined security rules regardless of the path it takes. Moreover, dynamic SFC orchestration can enhance security by detecting and responding to security threats in real time. If a VNF responsible for intrusion detection identifies malicious activity, the SDN controller can dynamically reconfigure the service chain to route traffic through additional security functions, providing an adaptive and responsive security posture (Toumi et al., 2020).

The implementation of dynamic SFC orchestration also requires careful consideration of the underlying infrastructure and the performance characteristics of the VNFs themselves. VNFs may have different resource requirements depending on their function, and the hardware on which they run can impact their performance. For example, a firewall VNF may require significant CPU resources to inspect traffic, while a load balancer VNF may be more dependent on network bandwidth. When orchestrating a dynamic service chain, the SDN controller must take into account the specific resource needs of each VNF and allocate resources accordingly. This requires close integration between the SDN controller and the NFV infrastructure, ensuring that VNFs are placed on hardware that can meet their performance requirements.

C. VNF OPTIMIZATION WITH HARDWARE ACCELERATION

VNF (Virtual Network Function) optimization through hardware acceleration represents a critical advancement in network function virtualization (NFV) architecture, aimed at mitigating the inherent performance overhead introduced by software-based processing in virtualized environments. As VNFs are designed to perform specific network functions, such as firewalling, load balancing, and packet inspection, the efficiency and speed with which they process packets is paramount for achieving the desired throughput and minimizing latency. Traditional virtualized environments that rely on general-purpose processors and virtualized network I/O tend to introduce bottlenecks due to the additional layers of abstraction and context switching between the virtualized network stack and the underlying hardware. Techniques such as DPDK (Data Plane Development Kit) and SR-IOV (Single Root I/O Virtualization) provide effective solutions to address these performance challenges by enabling hardware acceleration, thus enabling VNFs to function with near-native performance.

DPDK is a set of libraries and drivers that facilitate high-performance packet processing by allowing VNFs to bypass the traditional kernel network stack. In a conventional system, network packets are typically processed by the Linux kernel, which involves a considerable amount of overhead in terms of system calls, context switches, and interrupt handling. This overhead is especially pronounced in high-throughput environments where VNFs need to process large volumes of packets per second. By leveraging DPDK, VNFs can directly interact with the NIC (Network Interface Card), thus eliminating the need for the packets to traverse the kernel network stack. This direct interaction drastically reduces latency and enables higher packet throughput.

The core mechanism behind DPDK's efficiency lies in its ability to provide user-space poll mode drivers (PMDs) that directly access the NIC without requiring kernel intervention. In contrast to traditional interrupt-based processing, where the kernel would be interrupted by the NIC to signal the

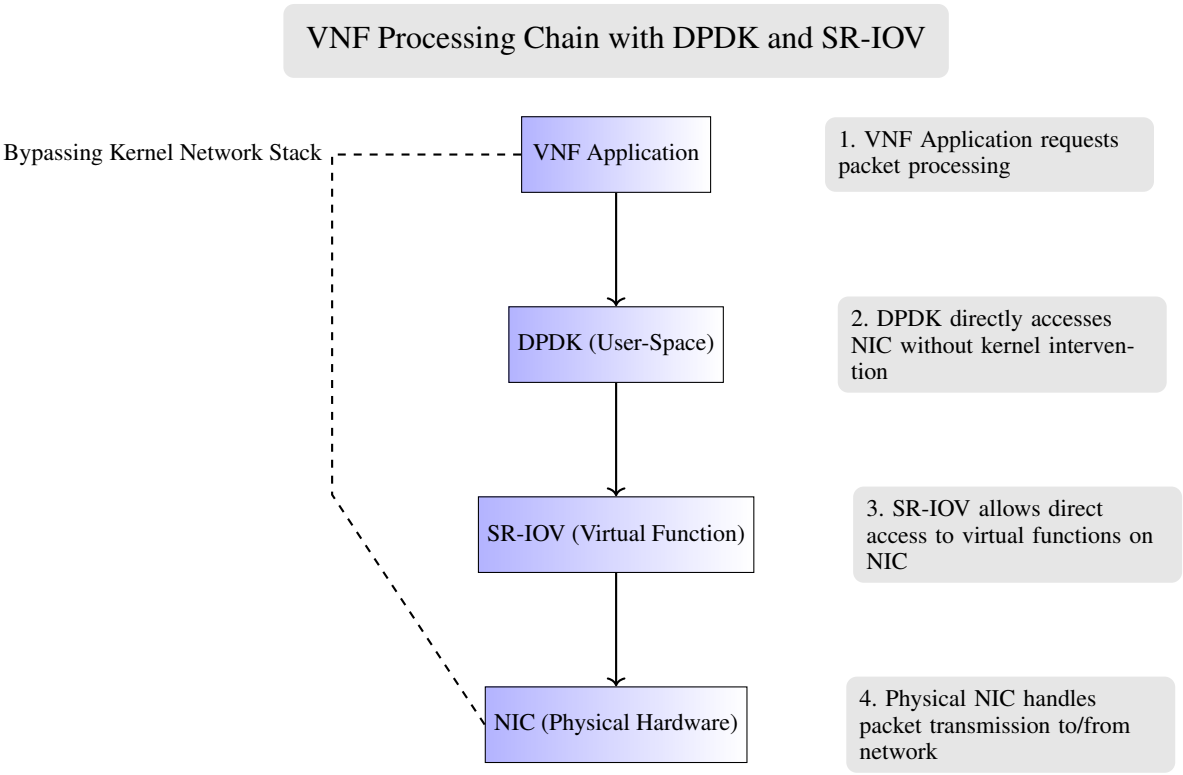

FIGURE 4. Optimized VNF Packet Processing using DPDK and SR-IOV

arrival of new packets, DPDK operates in a polling mode, wherein the application continually polls the NIC for new packets. Although polling can increase CPU utilization, it provides much lower latency and a predictable processing pattern, which is essential for applications requiring high-performance, real-time packet processing. Furthermore, DPDK's use of huge pages and memory pre-allocation improves memory access times, as these mechanisms minimize the number of page faults and memory fragmentation during runtime.

DPDK is highly advantageous in use cases such as firewalls, packet inspection, and network address translation (NAT), where low latency and high throughput are critical. One of the key benefits of DPDK is its flexibility, as it can run on various hardware platforms, including x86 architectures, ARM, and PowerPC, allowing for broad applicability across different types of VNF deployments. Additionally, DPDK supports multi-core architectures, enabling applications to take full advantage of modern multi-threaded processors, thus further enhancing packet processing capabilities.

While DPDK offers a significant performance boost for VNFs, it is important to recognize that the overhead associated with virtualization still exists in scenarios where multiple VNFs are chained together to perform complex service functions. This overhead can be reduced by integrating DPDK with SR-IOV, another hardware acceleration technique designed to minimize the performance penalties introduced by virtualization. SR-IOV allows VNFs to bypass the hypervisor's virtual switch entirely by providing direct access to the physical NIC, effectively eliminating the need for software-based packet switching between virtual machines (VMs).

SR-IOV achieves this by partitioning a single physical NIC into multiple virtual functions (VFs), each of which can be assigned directly to a virtual machine. These virtual functions operate independently, allowing each VM to access the NIC as though it were a dedicated resource, thus significantly reducing I/O latency and improving packet processing performance. By leveraging SR-IOV, VNFs can achieve performance that is closer to bare-metal implementations, as the overhead associated with packet copying, interrupt handling, and context switching is greatly diminished.

When combined, DPDK and SR-IOV create a highly optimized environment for VNF execution. DPDK provides high-performance packet processing by allowing VNFs to bypass the kernel, while SR-IOV reduces the virtualization overhead by enabling direct access to physical hardware resources. This combination ensures that VNFs can process packets at near-native speeds, making it possible to handle the high throughput and low-latency requirements of modern telecommunications networks, such as those found in 5G environments.

In a typical VNF deployment, multiple VNFs are often chained together to form a service function chain (SFC), wherein each VNF performs a specific role in the overall processing of network traffic. For example, a common SFC

might consist of a firewall, followed by a load balancer, and then a deep packet inspection (DPI) function. Without hardware acceleration, the processing of each packet would involve significant overhead due to the context switches between VNFs and the underlying hardware, resulting in increased latency and reduced throughput. By utilizing DPDK and SR-IOV, each VNF in the chain can operate more efficiently, thus reducing the overall processing time for packets traversing the chain.

DPDK and SR-IOV not only optimize packet processing performance but also provide enhanced resource utilization in NFV environments. By enabling VNFs to offload packet processing tasks to dedicated hardware resources, the CPU is freed up to handle other tasks, such as control plane operations and management functions. This improved resource utilization is critical in large-scale NFV deployments, where multiple VNFs may need to coexist on the same physical host. Furthermore, the reduced overhead associated with DPDK and SR-IOV allows service providers to deploy more VNFs per host, thus increasing the overall density of VNFs in the network while maintaining high performance.

Another aspect to consider when implementing DPDK and SR-IOV in VNF environments is the impact on scalability and flexibility. While SR-IOV provides direct access to hardware, it introduces a degree of hardware dependency, as the number of virtual functions that can be created is limited by the capabilities of the physical NIC. This limitation can pose challenges in environments where dynamic scalability is required, as it may not be possible to create additional virtual functions on demand. In contrast, DPDK offers more flexibility in terms of scaling, as it is a purely software-based solution that can be deployed on a wide range of hardware platforms without requiring specialized NICs. However, the performance benefits of SR-IOV are more pronounced in environments where low-latency, high-throughput processing is critical, such as in core network functions like packet gateways and evolved packet cores (EPCs).

In addition to scalability concerns, the use of DPDK and SR-IOV also introduces complexities in terms of management and orchestration. Both technologies require careful configuration and tuning to achieve optimal performance, and this process can be non-trivial, especially in large-scale deployments where multiple VNFs are running simultaneously. For example, DPDK requires fine-tuning of the CPU core allocation, memory management, and NIC configuration to ensure that packet processing is evenly distributed across the available resources. Similarly, SR-IOV requires precise mapping of virtual functions to VNFs to avoid resource contention and ensure that each VNF has sufficient access to the NIC. These configuration challenges necessitate advanced orchestration tools that can automate the deployment and management of VNFs while taking into account the specific requirements of DPDK and SR-IOV.

Security is another consideration when deploying DPDK and SR-IOV in VNF environments. The direct access to hardware resources provided by SR-IOV, while beneficial for performance, also increases the attack surface, as VMs have direct access to the physical NIC. This can potentially expose the system to security vulnerabilities if the VNFs are not properly isolated from one another. To mitigate these risks, strict access control policies must be implemented, and network traffic should be monitored for any suspicious activity. Additionally, the use of trusted platform modules (TPMs) and secure boot mechanisms can help ensure that the VNFs running on the host are not compromised.

D. QOS-AWARE TRAFFIC STEERING AND PRIORITIZATION

Quality of Service (QoS)-aware traffic steering and prioritization within Software-Defined Wide Area Networks (SD-WAN) is a sophisticated mechanism aimed at ensuring efficient network traffic management in scenarios where multiple types of traffic with varying requirements co-exist. The core of QoS-aware traffic steering revolves around the classification, prioritization, and routing of traffic in a manner that aligns with the service-level agreements (SLAs) and application-specific needs. Through SD-WAN, this process can be automated and dynamically adjusted, ensuring that critical, latency-sensitive traffic, such as real-time applications like Voice over IP (VoIP) and video conferencing, is prioritized over less critical, best-effort traffic. The integration of Service Function Chaining (SFC) further enhances this capability by allowing the precise orchestration of network functions and traffic paths.

In the context of QoS policies, the fundamental aim is to differentiate between various types of traffic based on their sensitivity to network conditions like latency, jitter, and packet loss. Real-time applications, such as VoIP and video conferencing, have strict latency requirements, as even slight delays can degrade the user experience. Therefore, these applications are typically assigned higher priorities within the network, ensuring that they are processed with minimal delay. SD-WAN controllers can classify traffic based on several parameters, such as application type, port numbers, or even deeper inspection methods that analyze traffic patterns. This classification enables the network to make informed decisions on how to handle the traffic flows.

Once traffic is classified, it can be assigned a priority level corresponding to its QoS requirements. High-priority traffic, such as VoIP, can be steered along paths that are specifically optimized for low latency and minimal jitter. This is achieved through dynamic path selection algorithms that constantly monitor network conditions and reroute traffic as needed. In contrast, traffic that is less sensitive to latency, such as bulk file transfers or web browsing, can be routed through paths that might not offer the lowest latency but are sufficient for the QoS requirements of such applications.

SD-WAN solutions, by leveraging the principles of traffic steering and prioritization, are capable of dynamically selecting the optimal path for each flow. These decisions can be based on real-time network analytics, such as link congestion, latency measurements, packet loss, and the availability

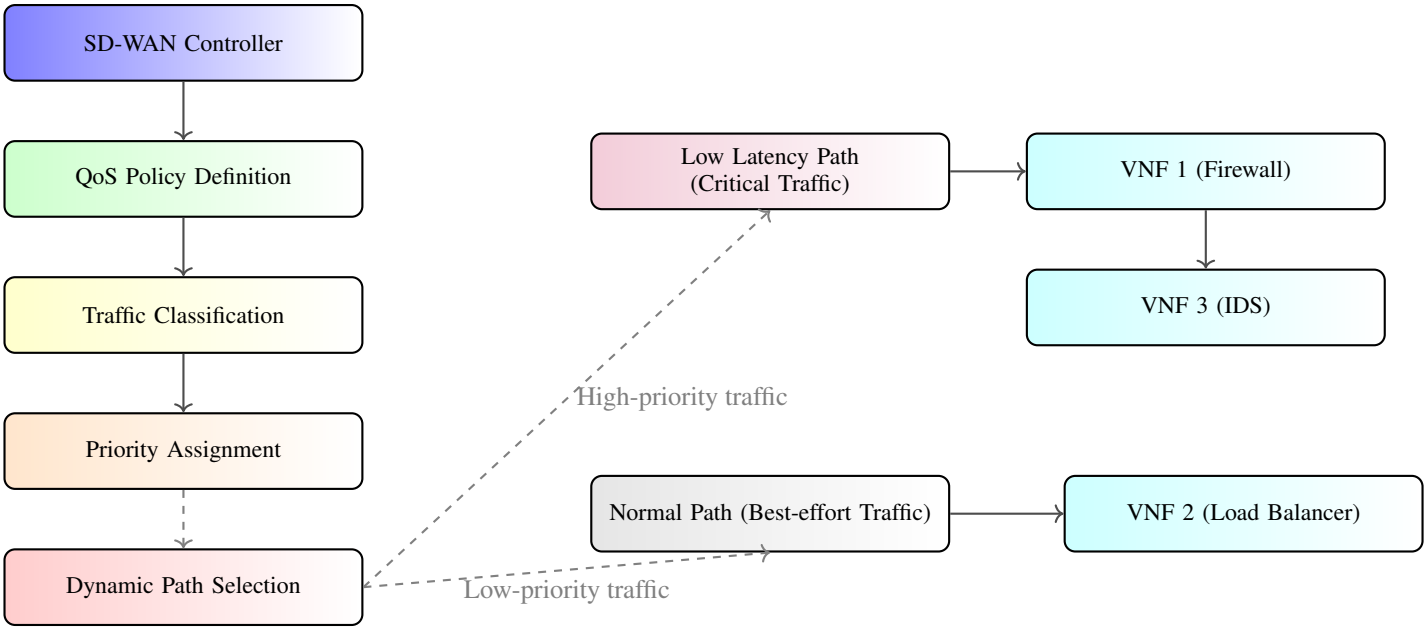

FIGURE 5. QoS-Aware Traffic Steering and Prioritization in SD-WAN

of virtualized network functions (VNFs). For instance, in a multi-site enterprise network, an SD-WAN controller might choose to steer VoIP traffic through a direct, low-latency link between two data centers while routing less critical traffic over a higher-latency, internet-based VPN tunnel. This type of dynamic path optimization is beneficial in hybrid WAN environments, where multiple types of connections—such as MPLS, broadband, and LTE—are available.

The concept of SFC plays a significant role in QoS-aware traffic steering. Service Function Chaining allows for the creation of logical service chains, wherein traffic is routed through a series of VNFs before reaching its destination. These VNFs can include network services such as firewalls, intrusion detection systems, and load balancers. In a QoS-aware framework, the selection of the service chain path can be influenced by the QoS requirements of the traffic. For example, latency-sensitive traffic might be routed through a simplified service chain with fewer VNFs to minimize processing delays, while less critical traffic might traverse a more complex service chain that includes additional network functions.

Packet tagging mechanisms are another essential component of QoS-aware traffic steering. By tagging packets with specific QoS labels, such as Differentiated Services Code Point (DSCP) values, the network can apply different handling policies to individual packets. This allows for granular control over traffic prioritization within the service chain. For instance, packets marked with high-priority DSCP values might be assigned to queues that ensure low-latency forwarding, while packets with lower-priority tags might be queued in best-effort buffers. Packet tagging thus enables differentiated treatment of traffic flows, ensuring that the most critical traffic receives the resources it needs, even when network congestion occurs.

One of the key challenges in QoS-aware traffic steering is the dynamic nature of modern networks. Network conditions can change rapidly due to factors such as link failures, congestion, or fluctuating bandwidth availability. To address this, SD-WAN controllers typically employ continuous network monitoring and real-time telemetry to adjust traffic steering decisions on the fly. This is often achieved through the use of algorithms that predict potential network bottlenecks or performance degradation and reroute traffic proactively. For instance, if a particular VNF experiences high processing delays due to a sudden increase in load, the SD-WAN controller can redirect high-priority traffic to an alternative service chain that offers better performance (Trajkovska et al., 2017; Zu et al., 2019).

The use of machine learning (ML) and artificial intelligence (AI) techniques has also been explored in the context of QoS-aware traffic steering. By analyzing historical network performance data and traffic patterns, ML models can predict future network behavior and make proactive adjustments to traffic routing. For example, an AI-driven SD-WAN solution might learn that certain links become congested during peak hours and automatically adjust its path selection algorithms to avoid these links for high-priority traffic. These predictive capabilities enhance the overall efficiency of the network and help ensure that QoS requirements are consistently met, even under varying network conditions.

In addition to dynamic path selection and service chaining, QoS-aware traffic steering also involves sophisticated traffic shaping and policing mechanisms. Traffic shaping involves regulating the flow of traffic to ensure that it conforms to predefined QoS policies. For example, in a scenario where bandwidth is limited, traffic shaping might be used to ensure

that high-priority traffic is allocated the necessary bandwidth, while lower-priority traffic is rate-limited to prevent it from consuming excessive resources. Traffic policing, on the other hand, involves enforcing QoS policies by dropping or marking packets that exceed the allowed traffic limits. This is important in scenarios where network resources are constrained, and it is necessary to ensure that high-priority traffic is not adversely affected by lower-priority flows.

Another critical aspect of QoS-aware traffic steering is the integration of network slicing, a technique commonly associated with 5G networks. Network slicing allows the creation of virtualized, isolated network segments, each with its own QoS characteristics. By leveraging network slicing, SD-WAN solutions can allocate dedicated resources to specific types of traffic, ensuring that QoS requirements are met even in highly congested networks. For example, a slice dedicated to real-time video conferencing might be provisioned with guaranteed low latency and high bandwidth, while another slice used for general internet access might have less stringent QoS requirements. This approach allows for the fine-tuning of network resources to match the needs of different applications, enhancing the overall efficiency and performance of the network.

The success of QoS-aware traffic steering also relies on the interoperability of SD-WAN solutions with existing network infrastructure and protocols. For instance, QoS policies defined within the SD-WAN controller must be compatible with the underlying routing protocols, such as BGP or OSPF, as well as with network devices that enforce QoS, such as routers and switches. This requires careful coordination between the SD-WAN controller and the underlying network devices to ensure that QoS policies are consistently applied across the network. Additionally, in environments where SD-WAN coexists with traditional MPLS networks, it is essential to ensure that QoS policies are harmonized across both types of networks to avoid inconsistencies in traffic handling.

Security considerations also play an essential role in QoS-aware traffic steering. In modern network environments, where cyber threats are constantly evolving, it is crucial to ensure that high-priority traffic is not only delivered with minimal latency but also processed securely. This requires the integration of security functions within the service chain, such as firewalls and intrusion prevention systems (IPS). However, these security functions can introduce additional latency, creating a trade-off between security and performance. To mitigate this, SD-WAN controllers can be configured to bypass certain security functions for trusted, high-priority traffic, or to route such traffic through optimized security paths that minimize latency. This approach ensures that QoS requirements are met without compromising the security of the network.

E. EDGE COMPUTING INTEGRATION

Edge computing represents a paradigm shift in the design and deployment of network architectures in the context of service function chaining (SFC) and software-defined wide-area networks (SD-WAN). By moving critical processing tasks closer to the end-user, edge computing addresses some of the inherent challenges associated with centralized data center architectures. In SFC, which relies on the sequential processing of traffic through various service functions (e.g., firewalls, load balancers, intrusion detection systems), the introduction of edge computing enables significant reductions in latency, increases in bandwidth efficiency, and enhanced reliability of real-time applications.

In traditional architectures, network functions were typically housed in centralized data centers, meaning that traffic from end users would often need to traverse the entire wide-area network (WAN) to reach these processing points. This introduced substantial latency for latency-sensitive applications such as video streaming, gaming, and IoT-driven automation systems. Edge computing mitigates this by hosting virtual network functions (VNFs) at edge nodes, strategically located closer to end users, allowing traffic to be processed locally rather than traveling back and forth between remote data centers. This decentralization of service functions results in faster processing times and improved overall network performance (Wang et al., 2018).

One of the key advantages of integrating edge computing with SFC is the potential for dramatic latency reduction. For real-time applications, where every millisecond counts, routing traffic to a centralized data center can significantly impact performance. By deploying VNFs at edge nodes, service functions such as firewalls, traffic optimizers, and intrusion detection systems (IDS) can be executed locally, minimizing the physical distance data packets must travel. This localized processing, combined with the potential for faster hardware and optimized software environments at the edge, enables faster responses to user requests and more efficient handling of high-throughput data streams.

Another dimension where edge computing enhances SFC is in its ability to handle network congestion and link failures dynamically. Traditional centralized architectures often rely on static routing paths, making them more susceptible to performance degradation in the event of congestion or failure at specific network nodes. Edge computing introduces the concept of dynamic traffic routing, where traffic can be redirected to alternative edge nodes with available capacity in real-time. This ensures greater service continuity and reduced downtime, even when specific links experience high traffic loads or fail entirely. By distributing VNFs across multiple edge locations, service providers can implement more resilient SFCs, providing fault tolerance through alternative processing routes.

In addition to improving latency and fault tolerance, edge computing also allows for more efficient resource utilization in SD-WAN environments. Traditional WAN architectures often experience inefficiencies in resource allocation, as traffic from disparate sources must funnel through centralized nodes, which may lead to bottlenecks or underutilization of network resources. Edge nodes, by contrast, distribute the processing load more evenly across the network. This

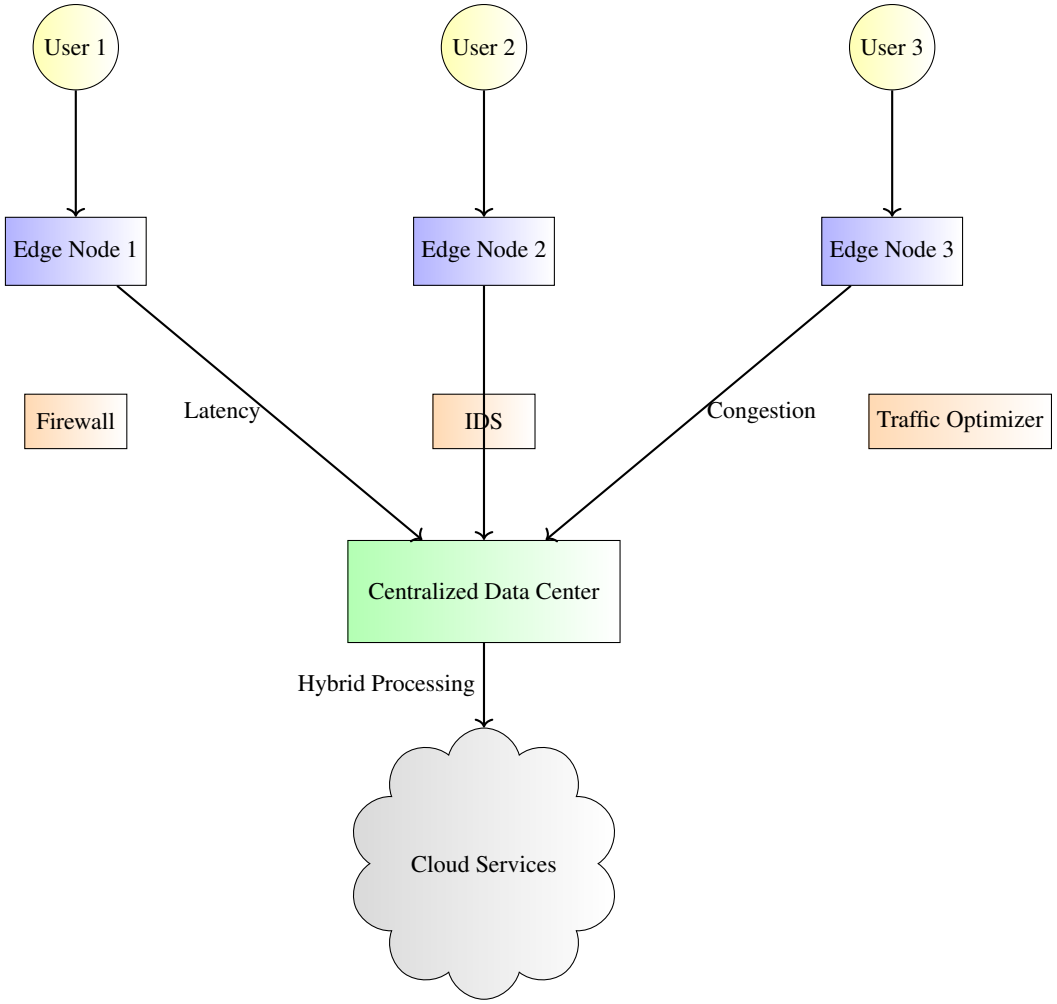

FIGURE 6. Edge Computing Integration in Service Function Chaining (SFC) within an SD-WAN Environment. Real-time applications from end-users are processed by edge nodes hosting VNFs (firewall, IDS, traffic optimizer), reducing latency and improving network performance. In case of high resource demand or non-critical tasks, traffic is sent to the centralized data center, which can offload to the cloud for hybrid processing.

distribution enables more optimal use of available computing and network resources, reducing the likelihood of any single node becoming overloaded, while also improving overall network scalability.

The integration of edge computing with SFC also has significant implications for security, which is a primary concern in modern networks. By deploying security functions, such as firewalls and IDS, closer to the network edge, potential threats can be detected and mitigated much earlier in the traffic flow. This pre-emptive processing of security-related functions at the edge reduces the risk of compromised traffic reaching central data centers or critical core network components. Moreover, edge computing provides greater flexibility for service providers to implement security policies that are more tailored to the specific requirements of local environments or user groups. This localized approach to security can be more effective in addressing region-specific threats or attacks, as well as providing a quicker response to potential breaches.

From an architectural perspective, the integration of edge computing into SD-WAN environments necessitates careful planning in terms of both hardware and software design. Edge nodes require adequate computing power, storage capacity, and network connectivity to host and execute VNFs effectively. However, edge nodes typically have more constrained resources compared to centralized data centers, so the VNFs deployed at the edge must be lightweight and optimized for resource efficiency. This may involve using containerized network functions (CNFs) instead of traditional VNFs, as CNFs are generally more resource-efficient and can be deployed more quickly in edge environments.

In addition to hardware considerations, the software infrastructure supporting edge computing in SFC must be highly dynamic and programmable. Software-defined networking (SDN) plays a critical role in enabling this flexibility, as it allows for the centralized management and orchestration of network functions while maintaining the ability to dynamically reconfigure network paths in response to changing traffic conditions. SDN controllers can be used to manage the placement of VNFs across edge nodes, ensuring that

service chains are deployed in the most efficient and optimal manner. By combining SDN with edge computing, network operators can create highly adaptable and scalable network environments that can quickly respond to both user demands and network conditions.

The scalability of edge computing in SFC also hinges on the ability to seamlessly integrate with cloud infrastructures. Many VNFs may still need to be deployed in centralized data centers or cloud environments when processing-intensive tasks are required or when edge nodes lack sufficient resources. Hybrid architectures, which combine both edge computing and cloud-based processing, provide a solution to this challenge. In such architectures, less latency-sensitive or more resource-intensive VNFs can be processed in the cloud, while critical, latency-sensitive VNFs are handled at the edge. This hybrid approach allows for a more efficient allocation of network resources and ensures that the benefits of edge computing are fully realized without sacrificing the flexibility and power of cloud-based processing.

Furthermore, edge computing facilitates the advancement of emerging technologies, such as machine learning (ML) and artificial intelligence (AI), in SFC. AI-driven analytics and ML algorithms can be applied at the edge to analyze network traffic patterns in real-time, predict potential congestion points, and optimize service function placement dynamically. By processing these analytics locally at edge nodes, the system can achieve lower response times for decision-making processes, enabling more proactive network management. This capability is useful for applications such as autonomous driving, augmented reality, and industrial automation, where rapid decision-making is essential.

The integration of edge computing in SFC also brings forth challenges that must be addressed for optimal performance. One of the primary concerns is the management and orchestration of distributed edge nodes in large-scale networks. As the number of edge nodes increases, so too does the complexity of coordinating VNFs across multiple locations. Effective management frameworks must be developed to handle this distribution, ensuring that service chains remain consistent and reliable, even as VNFs are deployed across a wide range of edge nodes. Standardized interfaces and protocols for VNF orchestration are critical to addressing this complexity, enabling seamless communication between centralized controllers and distributed edge nodes.

Another challenge lies in ensuring consistency and synchronization across the distributed edge environment. As VNFs are executed across multiple edge locations, there is a risk of state inconsistency between VNFs that depend on shared state information. This can be especially problematic in applications requiring real-time synchronization of data, such as collaborative IoT systems or distributed databases. Solutions such as distributed consensus algorithms, state replication techniques, and eventual consistency models can help mitigate these risks, but they must be carefully implemented to avoid negatively impacting network performance.

F. AUTOMATION AND SDN-BASED SFC MANAGEMENT

Automation in SDN-based Service Function Chaining (SFC) management plays a pivotal role in modern networking environments by leveraging the programmability and centralized control offered by Software-Defined Networking (SDN). This paradigm allows for the dynamic orchestration of network functions and service chains, where virtualized network functions (VNFs) can be strategically placed and managed in response to evolving network demands. Central to this is the SDN controller, which maintains an overarching view of the network and can adjust resources and paths according to real-time network conditions, optimizing performance for applications that require precise and responsive network behavior.

In traditional networking environments, managing the placement and execution of services often involves manual intervention or static policies that fail to account for the fluidity of network conditions. With SDN, the controller can monitor the state of the network in real-time and apply automation to dynamically adjust service chains based on traffic patterns, service demands, and application performance requirements. This enables the network to respond quickly and efficiently to changes in conditions, such as congestion, latency variations, or fluctuations in bandwidth demand, ensuring that services continue to meet their performance objectives (Xu, 2020).

A significant advantage of SDN-based automation is the ability to adjust service chains dynamically. In traditional static architectures, service function chaining is rigid, requiring pre-configured paths through which traffic flows, regardless of real-time network conditions. This static nature can lead to inefficiencies, such as suboptimal routing, increased latency, or wasted resources. SDN-based automation addresses these challenges by using the centralized controller to reconfigure the service chain dynamically. The SDN controller collects network telemetry and performance metrics in real-time, analyzing this data to determine the best paths for traffic to flow through the service chain. For example, in the case of a spike in traffic to a latency-sensitive application, the SDN controller can reroute traffic through VNFs located closer to the source of the demand, reducing latency and ensuring that the application continues to perform optimally.

Service Function Chaining (SFC) involves the sequencing of VNFs that are applied to traffic flows, such as firewalls, load balancers, intrusion detection systems, and more. With SDN, automation can enhance the SFC management by dynamically provisioning these functions and placing them in the most appropriate locations within the network topology. The SDN controller, having a global view of the network, can determine the most optimal placement of VNFs based on factors such as current traffic loads, network topology, and performance requirements of the services. Automation here ensures that VNFs are instantiated and deployed only where and when they are needed, thus optimizing resource utilization. Additionally, by leveraging automation, the SDN controller can also scale VNFs horizontally in response to increasing traffic or resource demands, enabling seamless elasticity in service delivery.

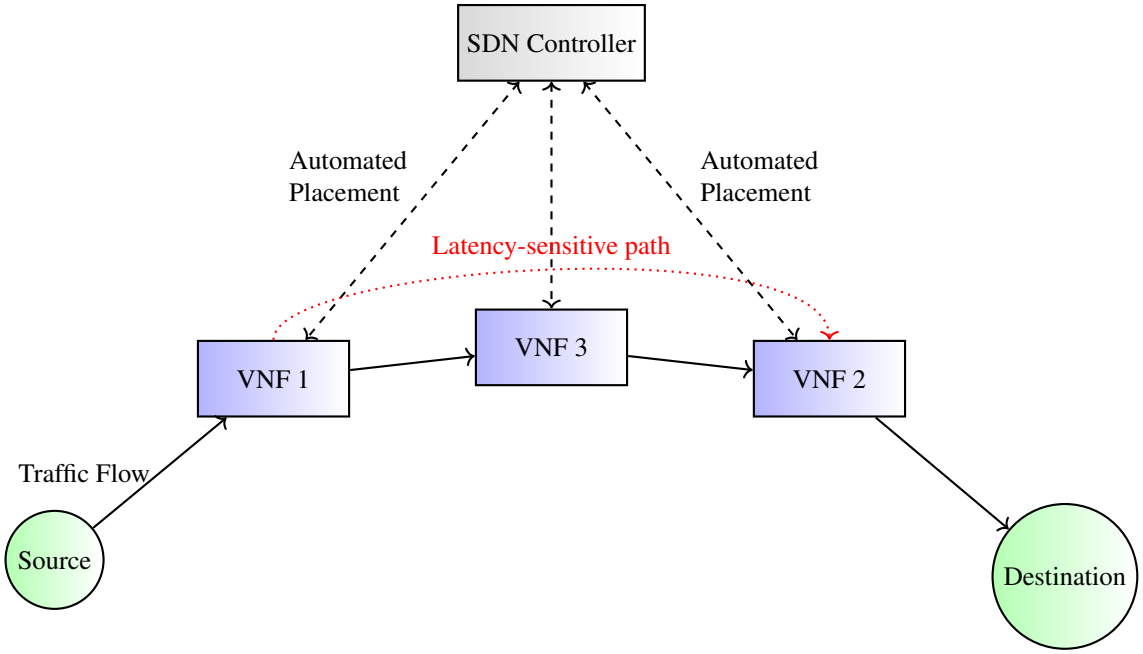

FIGURE 7. SDN-Based SFC Management and Automation Process

The SDN controller's capability to automate the placement of VNFs is crucial in ensuring that latency-sensitive services, such as voice over IP (VoIP), video conferencing, or real-time gaming, are processed at locations in the network where latency can be minimized. Latency-sensitive applications typically have stringent performance requirements, and any delay in processing can lead to degraded user experience. By using automation, SDN-based SFC management can constantly monitor network latency and adjust the placement of VNFs accordingly. For instance, if a VNF providing video compression for a video streaming service is experiencing increased latency due to a congested network link, the SDN controller can migrate that VNF to a different location that is closer to the user or has lower latency. This dynamic placement minimizes the end-to-end delay and ensures that the service continues to operate within acceptable latency bounds.

Automation in SDN-based SFC management also enables the prioritization of traffic, ensuring that critical or time-sensitive traffic is handled with the least possible delay. Real-time monitoring allows the SDN controller to identify the types of traffic traversing the network, and automation can be used to apply appropriate Quality of Service (QoS) policies. For example, high-priority traffic, such as emergency voice calls or financial transactions, can be routed through low-latency paths and prioritized over less critical traffic, such as bulk data transfers or file downloads. This real-time traffic engineering ensures that critical services are not impacted by congestion or suboptimal routing decisions.

Another key aspect of SDN-based automation in SFC management is the ability to optimize the network based on application performance metrics. Modern applications often require specific network conditions, such as a minimum guaranteed bandwidth or low jitter, to perform optimally. By integrating automation with application performance monitoring, the SDN controller can adjust the network and service chain in real-time to meet these requirements. For example, in the case of an application that requires a high-throughput network, the SDN controller can dynamically allocate additional bandwidth to the application's service chain or reroute traffic through high-capacity links to avoid congestion. Similarly, for an application sensitive to jitter, the SDN controller can adjust the routing of traffic to minimize variations in packet delivery times, ensuring a smooth and consistent user experience.

The dynamic nature of SDN-based SFC management also enhances the security and reliability of the network. Automation can be used to detect anomalies or security threats in real-time and adjust the service chain to mitigate these threats. For instance, if a security breach is detected on a specific path, the SDN controller can automatically reroute traffic away from that path, apply additional security functions, such as intrusion detection or encryption, or isolate the compromised segment of the network. This automated response reduces the time to mitigate security incidents and minimizes the impact on the network and services. Additionally, automation allows for continuous monitoring of the health of VNFs and service chains, ensuring that any failures or performance degradations are detected and addressed promptly.

The integration of automation in SDN-based SFC management also facilitates efficient resource utilization and energy savings. In traditional networking environments, VNFs and service chains are often overprovisioned to handle peak

loads, leading to wasted resources and higher operational costs. With SDN automation, the controller can dynamically allocate resources based on real-time demand, reducing overprovisioning and optimizing the use of network infrastructure. For example, during periods of low traffic, the SDN controller can consolidate VNFs, decommissioning unused or underutilized instances to free up resources and save energy. Conversely, during periods of high traffic, the controller can instantiate additional VNFs or allocate more bandwidth to meet the increased demand, ensuring that services continue to perform optimally without overburdening the network.

SDN-based automation also simplifies the management and deployment of service chains, reducing the complexity and operational overhead associated with traditional network management. In traditional environments, managing SFC involves configuring multiple devices and services manually, which is time-consuming and prone to errors. With SDN, the centralized controller can automate much of this process, allowing network administrators to define high-level policies and service chain requirements, which the controller then translates into low-level configurations. This not only reduces the potential for human error but also accelerates the deployment of new services and the modification of existing service chains.

A critical enabler of SDN-based SFC management is the use of intent-based networking (IBN) principles, where the network administrator specifies the desired outcomes or intent, and the SDN controller automatically configures the network to achieve those outcomes. For example, an administrator might define a policy that prioritizes traffic from a specific application or user group, or ensures that a certain level of performance is maintained for a critical service. The SDN controller, through automation, enforces these policies by dynamically adjusting the service chain and network resources to meet the specified intent. This approach simplifies network management and allows for more agile and responsive networks that can adapt to changing business needs and user demands.

The programmability of SDN further enhances the flexibility and customization of automation in SFC management. Network administrators can define custom automation workflows and policies that align with the specific requirements of their network and applications. For example, an organization might develop a custom policy that automatically scales up the number of VNFs in response to a specific type of traffic or service demand, or that automatically applies a specific set of security functions when certain types of traffic are detected. This level of customization allows organizations to tailor their network automation strategies to their unique needs, maximizing the benefits of SDN-based SFC management.

V. CONCLUSION

The increasing demand for real-time communication applications, such as Voice over Internet Protocol (VoIP) and video conferencing, has placed significant pressure on network infrastructures to meet the stringent latency requirements that these services necessitate. Traditional Wide Area Network (WAN) architectures, which typically relied on static service deployments in centralized data centers, have proven insufficient in addressing the performance needs of modern latency-sensitive applications. This inadequacy has prompted the development of new architectures like Software-Defined Wide Area Network (SD-WAN), which incorporates programmability and dynamic control to optimize routing and service delivery. However, even with SD-WAN, the introduction of Service Function Chaining (SFC) poses challenges for applications that require low latency. SFC involves processing traffic through a sequence of Virtual Network Functions (VNFs), and each additional service hop in the chain introduces potential delays, which can degrade the performance of latency-sensitive applications. This necessitates the exploration of methods to minimize the latency introduced by SFC in SD-WAN environments.

SD-WAN represents an evolution in WAN architecture that separates network control from underlying hardware, using principles derived from Software-Defined Networking (SDN). This separation allows for centralized management and policy enforcement, coupled with real-time traffic engineering capabilities. Unlike traditional WAN architectures, SD-WAN can intelligently route traffic over multiple transport networks—such as MPLS, broadband, and LTE/5G—based on network conditions and predefined policies. One of the key advantages of SD-WAN is its ability to dynamically steer traffic, enabling the efficient utilization of bandwidth and improving application performance, especially for latency-sensitive applications. For real-time services like VoIP and video conferencing, SD-WAN's programmability allows traffic to be routed over the least congested and lowest-latency paths, providing a higher level of performance than traditional WAN models.

Service Function Chaining (SFC) is an integral aspect of SD-WAN, enabling the application of a series of network services in a specific sequence. These services, often virtualized as VNFs, include firewalls, WAN optimizers, intrusion detection systems (IDS), and load balancers, among others. The flexibility of SFC allows network operators to customize the service path for different traffic flows, applying the necessary functions based on the specific requirements of the application. However, the key drawback of SFC lies in the latency introduced by each VNF in the chain. As packets pass through each VNF, processing delays accumulate impacting the performance of latency-sensitive applications. For example, VoIP applications typically require end-to-end latency below 150 milliseconds to maintain acceptable voice quality. The additional delays introduced by multiple service hops can push the latency beyond acceptable thresholds, resulting in degraded user experiences, including jitter, packet loss, and out-of-order packet delivery.

Addressing the latency challenges posed by SFC in SD-WAN environments requires an understanding of the technical challenges that contribute to increased delay. One of the primary factors is the processing overhead associated

with VNFs. VNFs are software-based implementations of network services, typically hosted on general-purpose hardware through virtualization technologies. While this provides flexibility, VNFs suffer from performance limitations, including context switching, I/O bottlenecks, and CPU constraints, all of which contribute to increased processing time. For latency-sensitive applications, these overheads can significantly increase the overall delay of the service chain. Optimizing VNF performance through techniques such as hardware acceleration can help mitigate these issues. For instance, technologies like Data Plane Development Kit (DPDK) and Single Root I/O Virtualization (SR-IOV) allow VNFs to bypass traditional network stacks and access hardware resources directly, improving packet processing times (Hu & Li, 2018; Yang et al., 2016).

Another significant challenge in SFC for SD-WAN is the selection of optimal paths and the steering of traffic across multiple transport networks. SD-WAN environments typically support dynamic path selection, allowing traffic to be routed based on current network conditions, such as latency, packet loss, and jitter. However, when SFC is applied, each VNF in the chain can influence traffic steering decisions. Traffic steering algorithms must account for the delays introduced by each service in the chain and dynamically select the optimal path to minimize both network-induced and service-induced latencies. This requires real-time monitoring of both the network and the performance of VNFs to enable intelligent routing decisions that prioritize low-latency flows.

Service placement and proximity also play a critical role in reducing latency for latency-sensitive applications. Traditional architectures often deploy VNFs in centralized data centers, which may require traffic to traverse long distances before being processed. The physical distance between the source, the service chain, and the destination introduces significant round-trip delays, especially for applications that operate across multiple geographic regions. Edge computing, which involves distributing service functions closer to end-users, offers a solution to this problem. By deploying VNFs at edge nodes, traffic can be processed locally, reducing the need for long-haul transport and minimizing round-trip time. Integrating edge computing into SD-WAN environments allows for the deployment of critical services, such as firewalls and traffic optimizers, at the network edge, thereby reducing latency for real-time applications.

Several optimization techniques can be employed to address the latency issues in SFC for SD-WAN. One of the most effective methods is intelligent service function placement. By dynamically placing VNFs at locations that minimize latency, such as edge nodes closer to the users, SD-WAN controllers can significantly reduce geographic delays. A hybrid approach that combines edge and cloud resources can be effective, where latency-sensitive VNFs are hosted at the edge, and less critical functions are processed in centralized data centers. This approach balances the need for low-latency with the efficient utilization of resources, ensuring that critical services are delivered with minimal delay.

Dynamic SFC orchestration is another method that can reduce latency by adjusting the service chain in real-time based on current network and service performance. Static configurations of SFC can lead to inefficiencies, as they do not account for changing network conditions or fluctuating loads on VNFs. By leveraging SDN controllers, which monitor both network and VNF performance, service chains can be reconfigured dynamically to ensure optimal performance for latency-sensitive applications. For example, if a VNF in the chain experiences high load or network congestion, traffic can be rerouted to an alternative VNF or non-essential services can be bypassed to reduce latency.

VNF optimization through hardware acceleration techniques, such as DPDK and SR-IOV, can further enhance the performance of SFC in SD-WAN environments. DPDK allows VNFs to process packets more efficiently by bypassing the kernel network stack and interacting directly with the Network Interface Card (NIC), while SR-IOV enables VNFs to access physical hardware resources without the overhead of virtualization. These technologies can significantly reduce the time required for packet processing within the service chain, achieving near-native performance and improving the overall latency profile of the network.

In addition to hardware optimization, Quality of Service (QoS)-aware traffic steering can be employed to prioritize latency-sensitive traffic over less critical flows. By classifying traffic based on application type and assigning higher priority to real-time applications, SD-WAN controllers can ensure that these flows are routed through the optimal service chain paths. QoS policies can also help avoid congested links and high-latency VNFs, ensuring that critical traffic is processed with minimal delay. Packet tagging mechanisms further enhance this by allowing for granular control over how traffic is handled within the service chain, ensuring that the most critical traffic receives the highest priority.

Finally, edge computing integration plays a pivotal role in reducing latency for SFC in SD-WAN environments. By deploying service functions at the edge, closer to the end-users, SD-WAN can significantly reduce the delays associated with long-distance traffic traversal. Hosting critical services such as firewalls, traffic optimizers, and IDS at edge nodes allows for faster response times in the event of network congestion or link failures, as traffic can be dynamically rerouted to alternative edge nodes with available capacity. This ensures that real-time applications continue to perform optimally even in the face of network disruptions.

Automation and SDN-based SFC management further enhance the ability to optimize latency in SD-WAN environments. SDN controllers provide centralized control over the entire network, enabling automated adjustments to the service chain based on real-time conditions. By combining automation with real-time monitoring, SDN controllers can dynamically place VNFs, ensuring that latency-sensitive services are always processed at the most optimal locations. This enables fast decision-making and allows the network to quickly respond to changes in traffic patterns or service

demand, further reducing the latency experienced by real-time applications.

Hardware acceleration requires specialized hardware components and driver support, limiting its applicability across heterogeneous environments where varying types of hardware are deployed. Moreover, the integration of such technologies necessitates a higher level of expertise, adding to the operational burden for network operators. As a result, the benefits of these optimizations may not be universally achievable across all SD-WAN implementations in distributed or multi-vendor networks where uniform hardware support cannot be guaranteed. This presents a significant challenge in realizing consistent performance improvements at scale.

Another limitation stems from the dynamic nature of traffic patterns and network conditions in SD-WAN environments. The research primarily focuses on optimizing service placement and traffic steering based on real-time performance data. However, real-time monitoring and dynamic reconfiguration of SFC require constant communication between SDN controllers, VNFs, and network infrastructure, which can introduce additional processing and signaling overheads. This reliance on continuous monitoring poses a challenge in highly dynamic environments, where frequent reconfigurations may cause service interruptions or introduce momentary delays. Additionally, the computational load on SDN controllers, responsible for orchestrating these changes, can become a bottleneck when managing large-scale networks with numerous VNFs and complex service chains. Therefore, while dynamic orchestration holds promise for optimizing latency, the practical implementation of such mechanisms may introduce new inefficiencies, potentially counteracting the intended performance gains.

Edge computing introduces challenges related to resource allocation and management. Edge nodes typically have limited computational, storage, and networking resources compared to centralized data centers, which could constrain the number of VNFs that can be deployed at the edge. Moreover, distributing services across numerous edge locations complicates the management of VNF states and consistency, especially for services that require synchronized states across multiple nodes.

VECTORAL PUBLISHING POLICY

VECTORAL maintains a strict policy requiring authors to submit only novel, original work that has not been published previously or concurrently submitted for publication elsewhere. When submitting a manuscript, authors must provide a comprehensive disclosure of all prior publications and ongoing submissions. VECTORAL prohibits the publication of preliminary or incomplete results. It is the responsibility of the submitting author to secure the agreement of all co-authors and obtain any necessary permissions from employers or sponsors prior to article submission. The VECTORAL takes a firm stance against honorary or courtesy authorship and strongly encourages authors to reference only directly relevant previous work. Proper citation practices are a fundamental obligation of the authors. VECTORAL does not publish conference records or proceedings.

VECOTORAL PUBLICATION PRINCIPLES

Authors should consider the following points:

1) To be considered for publication, technical papers must contribute to the advancement of knowledge in their field and acknowledge relevant existing research.
2) The length of a submitted paper should be proportionate to the significance or complexity of the research. For instance, a straightforward extension of previously published work may not warrant publication or could be adequately presented in a concise format.
3) Authors must demonstrate the scientific and technical value of their work to both peer reviewers and editors. The burden of proof is higher when presenting extraordinary or unexpected findings.
4) To facilitate scientific progress through replication, papers submitted for publication must provide sufficient information to enable readers to conduct similar experiments or calculations and reproduce the reported results. While not every detail needs to be disclosed, a paper must contain new, usable, and thoroughly described information.
5) Papers that discuss ongoing research or announce the most recent technical achievements may be suitable for presentation at a professional conference but may not be appropriate for publication.

References

Bhamare, D., Jain, R., Samaka, M., & Erbad, A. (2016). A survey on service function chaining. *Journal of Network and Computer Applications*, *75*, 138–155.

Guo, L., Pang, J., & Walid, A. (2016). Dynamic service function chaining in sdn-enabled networks with middleboxes. *2016 IEEE 24th International Conference on Network Protocols (ICNP)*, 1–10.

Hantouti, H., Benamar, N., Taleb, T., & Laghrissi, A. (2018). Traffic steering for service function chaining. *IEEE Communications Surveys & Tutorials*, *21*(1), 487–507.

Hu, Y., & Li, T. (2018). Enabling efficient network service function chain deployment on heterogeneous server platform. *2018 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, 27–39.

Jani, Y. (2021). The role of sql and nosql databases in modern data architectures. *International Journal of Core Engineering & Management*, *6*(12), 61–67.

Leivadeas, A., Falkner, M., & Pitaev, N. (2020). Analyzing service chaining of virtualized network functions with sr-iov. *2020 IEEE 21st International Conference on High Performance Switching and Routing (HPSR)*, 1–6.

Li, G., Zhou, H., Feng, B., Zhang, Y., & Yu, S. (2019). Efficient provision of service function chains in overlay networks using reinforcement learning. *IEEE Transactions on Cloud Computing*, *10*(1), 383–395.

Pei, J., Hong, P., Xue, K., & Li, D. (2018). Resource aware routing for service function chains in sdn and nfv-enabled network. *IEEE Transactions on Services Computing*, *14*(4), 985–997.

Silvestro, A. (2019). *Architectural support for implementing service function chains in the internet* [Doctoral dissertation, Dissertation, Göttingen, Georg-August Universität, 2018].

Sun, G., Zhou, R., Sun, J., Yu, H., & Vasilakos, A. V. (2020). Energy-efficient provisioning for service function chains to support delay-sensitive applications in network function virtualization. *IEEE Internet of Things Journal*, *7*(7), 6116–6131.

Toumi, N., Bernier, O., Meddour, D.-E., & Ksentini, A. (2020). Towards cross-domain service function chain orchestration. *GLOBECOM 2020-2020 IEEE Global Communications Conference*, 1–5.

Trajkovska, I., Kourtis, M.-A., Sakkas, C., Baudinot, D., Silva, J., Harsh, P., Xylouris, G., Bohnert, T. M., & Koumaras, H. (2017). Sdn-based service function chaining mechanism and service prototype implementation in nfv scenario. *Computer Standards & Interfaces*, *54*, 247–265.

Wang, J., Qi, H., Li, K., & Zhou, X. (2018). Prsfc-iot: A performance and resource aware orchestration system of service function chaining for internet of things. *IEEE Internet of Things Journal*, *5*(3), 1400–1410.

Xu, Y. (2020). A study on efficient service function chain placement in network function virtualization environment.

Yang, K., Zhang, H., & Hong, P. (2016). Energy-aware service function placement for service function chaining in data centers. *2016 IEEE Global Communications Conference (GLOBECOM)*, 1–6.

Zamani, A., & Sharifian, S. (2018). A novel approach for service function chain (sfc) mapping with multiple sfc instances in a fog-to-cloud computing system. *2018 4th Iranian Conference on Signal Processing and Intelligent Systems (ICSPIS)*, 48–52.

Zhang, Y., Anwer, B., Gopalakrishnan, V., Han, B., Reich, J., Shaikh, A., & Zhang, Z.-L. (2017). Parabox: Exploiting parallelism for virtual network functions in service chaining. *Proceedings of the Symposium on SDN Research*, 143–149.

Zou, D., Huang, Z., Yuan, B., Chen, H., & Jin, H. (2018). Solving anomalies in nfv-sdn based service function chaining composition for iot network. *IEEE Access*, *6*, 62286–62295.

Zu, J., Hu, G., Wu, Y., Shao, D., & Yan, J. (2019). Resource aware chaining and adaptive capacity scaling for service function chains in distributed cloud network. *IEEE Access*, *7*, 157707–157723.