Article

Not peer-reviewed version

A Performance Evaluation for Software Defined Networks with P4

Omesh A Fernando *, Hannan Xiao, Joseph Spring, Xianhui Che

Posted Date: 30 April 2025

doi: 10.20944/preprints202504.2530.v1

Keywords: Software Defined Networking (SDN); Programming Protocol independent Packet Processing (P4); ONOS; Mininet

 doi:10.20944/preprints202504.2530.v1

Article

A Performance Evaluation for Software Defined Networks with P4

Omesh A Fernando [1], Hannan Xiao [2], Joseph Spring[1], and Xianhui Che [3,*]

[1] Department of Computer Science, University of Hertfordshire, Hatfield AL10 9AB, UK; w.k.fernando@herts.ac.uk (O.A.F.); j.spring@herts.ac.uk (J.S.)
[2] King's College London; hannan.xiao@kcl.ac.uk
[3] Queen Mary University of London
* Correspondence: w.k.fernando@herts.ac.uk

Abstract: Software Defined Networking (SDN) and Programming Protocol-independent Packet Processors (P4) have been attracting attention from both industry and academia as a means of achieving programmability in the network at different layers. The two emerging technologies have in the main, been researched individually in the literature, however, research evaluating the performance of SDN with P4 is limited. As the number of connected users grows and communication continues to evolve, increasing complexity and issues in heterogeneity relating to applications, increase the challenge for service providers to provide resilient connections with faster processing of packets increases. The application of SDN, with its ease of management through centralised programmable control logic, attracts attention from both academia and industry as a means of improving performance. This research has been conducted to explore the hypothesis that combining programmability at both the control plane and data plane, namely *SDN+P4*, will provide a platform with faster packet processing. In this research, we design a system platform to investigate the network and applications performance in *SDN+P4*, in comparison to *SDN+Open vSwitch*. Topologies considered were the multi-path, grid and transit-stub network models with hosts placed at leaf nodes to emulate a small, an intermediate, and a core network. The mininet emulation results demonstrate that across all case studies, parallel processing of P4 has provided more queues for traffic to be processed with the utilisation of a flexible parser. With the evolution of the internet and the heterogeneity of applications, *SDN+P4* will, we believe, provide a more resilient and stringent service to all use cases of 5G and beyond.

Keywords: Software Defined Networking (SDN); Programming Protocol independent Packet Processing (P4); ONOS; Mininet

1. Introduction

Software Defined Networking (SDN) [1,2] allows for the decoupling of the control plane from the data forwarding plane facilitating an improved performance via programmable network management. The flexibility obtained enables virtualisation, and centralised control for the network, with routing tables being generated and updated centrally by an OpenFlow controller. The separation of the control plane and data plane is made possible through a clearly defined interface which connects the switches to the network controller. This interface employs an Application Programming Interface (API), which allows the controller to exert direct control over the data plane elements (networked devices). The protocol OpenFlow [3,4], is a notable example of such an API.

An OpenFlow forwarding rule consists of a 'match' and 'action'. The 'match' involves the matching of packet header fields such as source and destination fields and an 'action' to be performed such as forward or drop a packet. This function is referred to as 'match+action' [5]. The controller installs these flow entries in the Flowtables of the SDN enabled devices. If a matching flow entry is found against a packet in the data flow, the predefined action (pass, drop) for that entry are performed on the matched packet. If no match is found, the packet is forwarded to the network controller as a

 doi:10.20944/preprints202504.2530.v1

Packet-in Message. The controller is responsible to determine how the packet should be handled by returning this specific packet to the switch and stating which port it should be forwarded to (Packet-out message).

OpenFlow was first introduced with 12 'match' header fields [6], and now stands at 44 header fields, as described in the OpenFlow specification 1.5.1 [4]. These are anticipated to grow exponentially in the foreseeable future, in response to the evolution and heterogeneity of the internet. This could potentially be a problem as the application of new header field extensions cannot be achieved when using a stringent-fixed parser at run time, whilst the switches are actually processing packets.

An alternative to extending the OpenFlow specification is to employ a novel parser that can carry out 'match+action' in parallel allowing for programmability in the data forwarding plane without having to modify the OpenFlow specification. In our abstract model, switches forward packets via a programmable parser followed by multiple stages of match+action, arranged in series, parallel or a combination of both. The forwarding model is controlled by two types of operations: Configure and Populate. Configure operation which is composed of Parse graph, Control Program, Table configuration, and Action set, program the parser, arrange the order of match+action, and specify the order of fields processed by each stage. Populate operation, add or remove entries to the match+action tables that were specified during configuration. Such a parser can be updated, extended, and modified without interrupting packet processing and forwarding, thereby enabling the integration of heterogeneous applications. A protocol that can be employed to carry out 'match+action' in parallel is the Programming Protocol Independent Packet Processing (P4) [6]. Parallel processing can be achieved in P4 by employing one of the following.

1. Multiple Processing Cores: Some network devices have multiple CPU cores or processing units. P4 programs can be designed to distribute packet processing tasks across these cores to achieve parallelism. For example, one core might handle packet parsing, while another core handles packet forwarding.
2. Parallel Pipelines: P4 allows the definition of multiple packet processing stages within a pipeline. These stages can be designed to operate in parallel, with each stage processing packets independently. For instance, one stage might perform access control checks while another stage performs packet routing.
3. Hardware Offload: In some cases, P4 programs can be used to offload certain packet processing tasks to specialized hardware accelerators or programmable ASICs. These hardware components can operate in parallel with the CPU, further increasing packet processing efficiency.
4. Load Balancing: P4 can be used to implement load balancing mechanisms, where incoming packets are distributed across multiple processing units or network paths, enabling parallel processing of packets.

P4 has three main properties: re-configurability, protocol independence, and target independence allowing the network administrator to determine the functions and capabilities of a switch rather than adhering to vendor specification [7]. Utilising a common open interface, the administrator can leverage P4 to program a flexible parser to match new header fields as opposed to working with a fixed parser in OpenFlow which would require a specification update in order to process new header fields.

The extensive research carried out in [8,9] indicate that the majority of future network traffic is expected to be the result of using hand-held smart devices. It is claimed that the increased traffic due to hand-held devices has resulted in approximately 52% of all consumer Internet traffic [9]. The results from the study [10] reports that mobile traffic for 2024 reached 1.3 zettabytes, supporting the conjectures from earlier studies. The increased load of end-user IP traffic threatens to lead to network congestion, resulting in a reduced Quality of Service (QoS) in terms of, for example, jitter, packet loss, and throughput [11] with service providers trying to maintain an acceptable level of service. It is interesting to note a challenge presented in [12], which suggests that the majority of the delay in data communication occurs at the core of the network due to an exponential growth in periodic updates [13,14] used to maintain the network state.

 doi:10.20944/preprints202504.2530.v1

Control and data plane programmable functions motivate us to investigate how this emerging approach can improve the performance of a network by processing packets faster. It is, we feel, crucial to investigate whether and how the extent to which a solution with programmability in both the control plane and forwarding plane will provide an improved core network to accommodate stringent performance and flexibility as network traffic continues to increase.

We hypothesise that a combination of data plane and control plane programmability (*SDN+P4*) will elevate the performance of the network, to accommodate stringent performance requirements for future applications in comparison to 'Open vSwitch (OvS) [15] coupled with control plane programmability' (*SDN+OvS*). In this paper, we explore *SDN+OvS* and *SDN+P4* environments for different topologies comparing the performance of each. Various types of traffic were transmitted to test the performance of the *SDN+OvS* and *SDN+P4*.

1.1. Related Work

Research in [16,17] discussed technological enhancements for networks highlighting technologies such as OpenFlow, P4, the Data Plane Development Kit and Click-based solutions. It was established that the protocol independent nature of the high-level language P4 [6] together with independence from the underlying hardware and header limitation benefits the network, increasing performance through the faster processing of packets.

The utilisation of the language P4 has been presented for many applications. [7,18–24]. Research in [7] applied the programmability of P4 to create an application capable of handling data centre traffic. In [18] a congestion control mechanism has been used together with P4 and in [19], mechanisms for packet processing capabilities through P4 have been utilised with a Robust Header Compression (ROHC) scheme for improving performance. Further examples include a tool for developing and evaluating data plane applications [20], Service Function Chaining on P4 devices in [21] and in [22], an extension to the application of Open vSwitch, to act as a hypervisor switch. A recent study by [24] proposed a classification employing data plane programming to better manage QoS by embedding Service Level Objectives into packets.

A study conducted in [25,26] employed an identical topology to that initially presented in [27] which highlights the validity of the respective initial work. The work presented by [25] employed a Transit Autonomous system setup with dynamic path computation based on real-time network statistics, enabling improved QoS through the use of a centralised controller. The study [26] focused on economic factors from a hypothetical migration towards OpenFlow for Network Service Providers (NSP) but did not evaluate network performance. In [28,29] a methodology to orchestrate a dynamic end-to-end (E2E) path between transit stub domains was presented, with the idea of SDN successfully managing traffic between NSPs. How an SDN controller should be placed between NSPs was discussed in [30] as a means of improving performance. This was achieved through the 'correct' and optimal placement of SDN controllers at various topological locations. A similar approach has been employed in [31] where a second controller was placed to mitigate traffic overload in the core network. In [32] it was suggested that in order to achieve better performance, a minimum of 20% of the nodes in the core network should operate as SDN controllers with control plane programmability. In [33,34] the authors utilised control plane functions in an SDN controller to improve performance. In addition, in [35] the use of a high-level specification template for management patterns was utilised to improve performance in SDN networks.

Various researchers have utilised the P4 language to improve the performance of a particular case study or an instance. To the best of our knowledge research aimed at improving the performance of a network through the utilisation of programmability at the control plane and data forwarding plane (*SDN+P4*) have been limited to improving performance for a specific case study only. This research is aimed at extending the work presented in [36] where the notion of evaluating SDN by the application of *SDN+P4* was initially tested.

 doi:10.20944/preprints202504.2530.v1

1.2. Contributions

This paper makes the following contribution to improving network performance by utilising the faster packet processing of P4. In particular, we have:

- Investigated the overhead created due to the slow path utilisation of OvS and the performance variation in comparison to a P4 target switch. With more and more packets requiring processing using a controller, a network model that utilises a slow path approach such as OvS will potentially lead to an exponential growth in traffic congestion.
- Evaluated the performance of networks when *SDN+P4* is employed rather than *SDN+OvS*. The evolution of 5G and beyond has led to the need to evaluate methods for reducing the delay at the core. Initialising programmability in the network has been shown to increase performance at the core. To the best of our knowledge, current literature does not evaluate the performance of the network when the control plane and data plane programmability (*SDN+P4*) is employed in comparison to the control plane (SDN+OvS) programmability.
- For a time-sensitive application with minimal latency requirements, reducing the delay at the core can be essential. For example, Vehicle-to-Vehicle (V2V) and Ultra Reliable Low Latency Communication (URLLC) applications. A solution, that processes packets in parallel as opposed to sequential processing in OvS, has been considered in this research and its effect on performance in applications. Our research has established that with the initialisation of *SDN+P4* with parallel processing of packets, various applications have better performance in comparison to applications run over *SDN+OvS*.
- Evaluated the quality of applications and the effect *SDN+P4* had on the network traffic of ICMP, TCP, UDP, SIP and CDN in comparison to *SDN+OvS*. The statistics such as increased bps and throughput, reduced delay jitter, packet loss, delay and buffering time have led to a higher quality of performance at the receiver's end

1.3. Structure of the Paper

The rest of the paper is organised as follows. System platforms employed for this research are discussed at Section II outlining the platforms and programs used. This is followed by the experimental design in Section III. Results and analysis for individual applications are presented in Section IV followed by results for the simultaneous execution of applications in Section V. The overall results obtained from our research are discussed in Section VI. Concluding observations are presented in Section VII.

2. System Platforms

To explore the research questions, a network emulation was built on Ubuntu 18.04LTS running on VMWare with a Core i7 CPU with 3.40GHz together with an Open Networking Operating System (ONOS) [37]. Mininet [38] was utilised to emulate the networks, as shown in Figure 1.

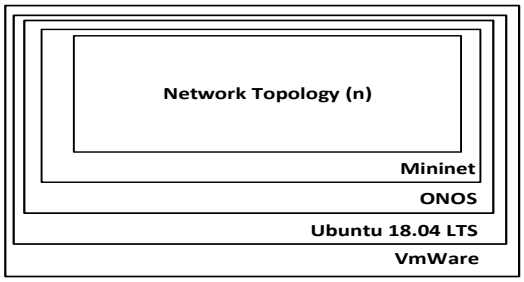


Figure 1. System Platform.

2.1. SDN Platform

ONOS is an open-source SDN network operating system built to provide a platform for service provider networks [37]. ONOS facilitates the control plane for network components such as switches,

 doi:10.20944/preprints202504.2530.v1

links or software [39]. Being highly reliable, resilient, and scalable makes ONOS, we believe the best open-source option [40,41] available for building and catering for SDN networks. A real-time GUI provides a global view of the network that aids administrators to monitor and manage resources accordingly. ONOS provides two APIs: the southbound API interacts with devices and the northbound API offers services to applications. The controller inside ONOS can push a switch configuration towards a forwarding device. We chose ONOS as a platform in this research for the above features and its ability to maintain a network's state without compromising performance.

2.2. Mininet

Mininet [38], developed by Stanford University, is an open-source, lightweight and easy-to-deploy network emulator that provides a programmable interface to launch and initialise networks either in wireless or wired mode. Mininet has the ability to initialise a large network with multiple hosts and switches on a physical host. Mininet supports the emulation of OpenFlow enabled switches such as OvS for this research. OvS coupled with an ONOS controller was used to emulate the *SDN+OvS* environment for this research.

2.3. P4 Switch

Upon selecting ONOS for the SDN platform, we chose the bmv2 switch [42] as our P4 software switch. A bmv2 switch can configure a custom parser with ingress and egress pipelines working in parallel to perform match+action. The program utilised is independent of the target switch design and can be employed to represent different switch designs should the need arise. The P4 software switch, bmv2 was instantiated without using specific metadata or specific port forwarding to specific traffic, to ensure fairness between the OvS switch and the bmv2 switch. The bmv2 switch coupled with the ONOS controller was employed to emulate the *SDN+P4* environment in this research.

SDN+P4 will enable programmability in both the control and data forwarding plane. Unlike the previous instance (*SDN+OvS*), *SDN+P4* has the capability to modify the switch configuration such as a flexible parser, 'match+action' which can operate in parallel or series or include metadata and buffer, using a JavaScript Open Notation (JSON) obtained from a P4 program [6]. By utilising a P4 switch, a fixed parser in OvS has been replaced with a flexible parser, which can process packets more efficiently and effectively.

3. Experimental Design

3.1. Network Topologies

Figure 2 illustrates the three topologies that we employed. Topology I is a multi-path topology that extends the spine-leaf [43] topology used by data centres. Multi-path topology was employed to test the effectiveness of SDN+OvS and SDN+P4 to a popular topology employed to simulate a data centre network. Gradual increment of nodes from [36] and ease of programming also contributed towards the rationality for the multi-path topology. Topology II was employed to emulate an environment with a simple-grid topology and a similar topology has been presented in [34,44,45]. Further extension of the multi-path topology with more nodes to the network and the popularity of similar topologies being employed for SDN research were the rationale towards employing the simple grid topology. Topology III emulates the transit-stub network model also known as the Internet topology initially presented in [27] and in subsequent works [25,26,46–52]. Extensive research conducted towards an SDN based internet, and the popularity of similar topologies being employed for such research were the rationale towards employing the internet topology.

 doi:10.20944/preprints202504.2530.v1

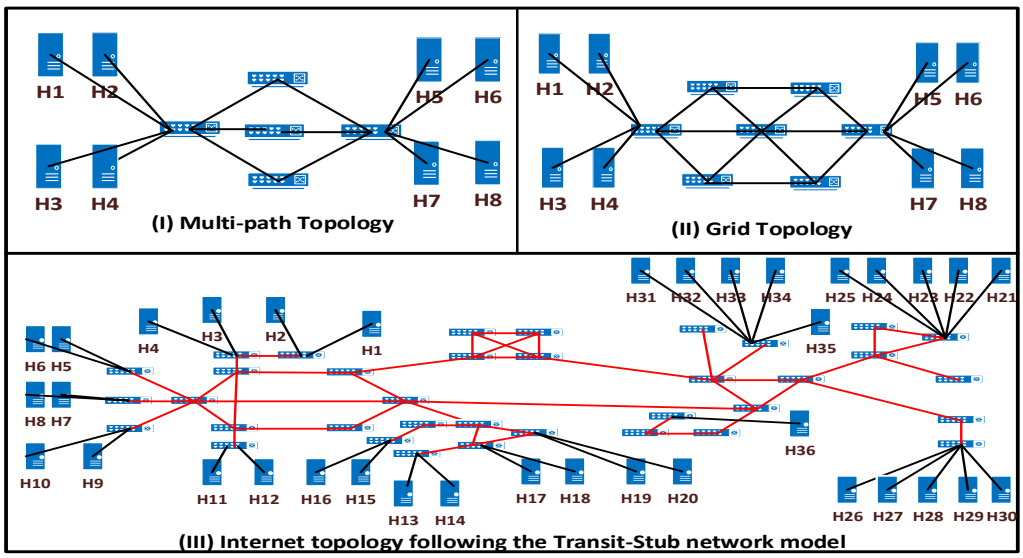


Figure 2. Topologies employed for this research, (I) multi-path topology (II) grid topology and a (III) Transit stub network model. (H denotes Host)

3.2. Traffic Design

In order to mimic the Internet as closely as possible, traffic types such as Internet Control Message Protocol (ICMP), Transmission Control Protocol (TCP) [53], User Datagram Protocol (UDP) [54] and Content Delivery Network (CDN) were chosen. The traffic types were selected based on research presented in [55], where the results focused on popular downloaded traffic. ICMP traffic is widely used as a measurement tool for troubleshooting, control and error messaging services. TCP can be considered as one of the main protocols of the IP suite. TCP is considered as one of the most trustworthy and reliable protocols utilised by applications on the internet. UDP is widely used to send messages with a minimum protocol mechanism. Based on the research presented in [56] and the popularity of UDP applications, UDP traffic was included. A recent study by [57], confirms the network traffic trends for 2024, indicating that TCP and UDP constitute the majority of Internet traffic. Finally with the popularity and demand of the CDN as highlighted in [8,58,59], a case study emulating the same has been included. The study [59], published in 2025 reports that an average user consumes approximately 6GB of CDN data per day. The above distinct traffic types were examined both on an individual basis and mixed basis to test and evaluate the performance of the network for both *SDN+OvS* and *SDN+P4*.

The involved experiments were conducted in two stages: Tier 1, in which only one type of traffic at a time was run over the network, ICMP, TCP, UDP or CDN, each forming a single case study and Tier 2, in which all four types of traffic were run simultaneously over topologies II and III.

3.3. Tier-I—Single Type of Traffic Run

Table 1 shows the configuration of simple types of traffic run in this Tier.

Case Study 1—ICMP

A custom Python script was utilised to send ICMP traffic between the designated hosts simultaneously to emulate an unpredictable traffic load in the network. ICMP traffic was generated using the ping command. A total of 1000 ICMP sequenced packets were saved for calculations for multi-path and grid topologies and 2000 sequenced packets for the Internet topology each with a packet size of 1500 bytes and a data rate of 12 Kbps. Data was saved for both *SDN+OvS* and *SDN+P4* in each of the three topologies. The default ping command encompasses the aforementioned packet size and the data rate. Hence, the rationality in choosing these configurations. The number of ICMP packets transferred for multi-path and grid topologies was chosen for ease of calculation. Due to the size of the network, the number of ICMP packets saved was doubled for the Internet topology.

 doi:10.20944/preprints202504.2530.v1

Table 1. Configuration of traffic. Hosts are shown in Figure 2.

Traffic	Multi-Path Topology and Grid Topology	Internet Topology
	Client → Server	Client → Server
ICMP	H1 → H5, H2 → H6, H3 → H7, H4 → H8	H1 → H21, H5 → H22, H9 → H23, H13 → H24, H17 → H25
TCP	H1 → H5, H2 → H6, H3 → H7, H4 → H8	H2 → H26, H6 → H27, H10 → H28, H14 → H29, H18 → H30
UDP	H1 → H5, H2 → H6, H3 → H7, H4 → H8	H3 → H31, H7 → H32, H11 → H33, H15 → H34, H19 → H35
CDN	H1 → H8, H2 → H8, H3 → H8, H4 → H8	H4 → H36, H8 → H36, H12 → H36, H16 → H36, H20 → H36

Case Study 2—TCP Traffic

In order to establish a connection with the server and the client and to transmit data between the client and the server, iPerf[60] v2.0 was used. iPerf's capabilities and functions were thoroughly examined and presented in [55]. Given the stability of v2.0 over v3.0 in iPerf, the former was selected for the experiments. For multi-path and grid topologies, data with a file size limitation of 1GB encapsulated, as TCP was chosen. For the Internet topology, a data burst for 1200s was chosen to identify which network abstraction was able to process the largest amount of data. For ease of emulation and calculation, a download with a file size of 1GB was chosen for multi-path and grid topologies. The 1GB file size served as an example to a scenario where a user intends to download a file of a fixed size from a File Transfer Protocol (FTP) server. For a scenario where the user is engaged in a TCP transmission with a time constraint (i.e., Secure Shell (SSH) or Telnet), a time limitation of 1200s was employed for the Internet topology. This was also applied with respect to the size of the network and for extending the download time, and the amount of data.

Case Study 3—UDP Traffic

For this case study, iPerf was utilised to stream UDP traffic. Similar to case study 2, we used of 1GB traffic for topologies I and II with a data burst of 1200s for topology III. The iPerf traffic was initiated with a bound of 12MBps between the client and the server for both UDP and TCP due to resource limitations in the system platform. Similar to the TCP case study, a download with a file size of 1GB was chosen for Topologies I and II. This case study setting was designed with the intent of emulating a user downloading multimedia content, which will be bound by a file size. For topology III, a UDP download was conducted for 1200s in the aim of emulating a user engaged in online gaming or video conference. A time constraint of 1200s was also applied with respect to the size of the network and for extending the download time and the amount of data. Measurements such as throughput, delay jitter, and transmission completion time were collected for both *SDN+OvS* and *SDN+P4* for each of the three topologies.

Case Study 4—Content Delivery Network

An emulation was designed to explore video traffic with live stream, using VLC-player [61] to send videos with the quality of 1080p and H.264 compression due to the demand and the popularity for high-quality video streaming. For multi-path and grid topologies, a video of 600s long was chosen, and for the Internet topology, a video of similar quality was chosen with a 1200s duration. The length of the two video files (600s and 1200s) was used for the ease of emulation and calculation purposes. As the Internet topology was larger in size as opposed to multi-path and grid topologies, a larger video file was employed. The Internet topology emulated a higher number of hosts and switches involved. In order to establish an emulation reinforcing a live video stream, a Python script was utilised that determined transfers between multiple clients with a server.

 doi:10.20944/preprints202504.2530.v1

3.4. Tier-II—Multiple Types of Traffic Running Simultaneously

Networks will carry different types of traffic, executing simultaneously. Tier II involves two case studies.

Case Study 5—Simultaneous Traffic in Grid Topology

We attempted to saturate the links by launching multiple applications simultaneously using a custom Python script that was applied to all of the traffic in Table 1. In order to make it easier to evaluate the performance, the client and server configurations were kept the same for this case study as they were for the individual case studies. This allows us to easily observe the variance in performance between individual execution and simultaneous execution. The distinction of traffic types and their usage remains the same with the addition of VoIP traffic. VoIP traffic was not presented individually since the results between *SDN+OvS* and *SDN+P4* had less significance due to a smaller volume of traffic involved in the communication.

Case Study 6—Simultaneous Traffic in the Internet Topology

In this case study, we launched all of the traffic simultaneously in the Internet topology according to Table 1, in order to closely mimic the way that the Internet operates, where multiple clients and servers are sending and receiving traffic. By stressing the network with a large amount of traffic and applications, we are able to collect valuable data on key performance metrics such as delay, delay jitter, packet loss, bps, and throughput. This data is collected on the client side.

4. Results and Analysis of Tier-I Single Type of Traffic Run

In this section, we present and analyse the results of the experiments for Tier-I, a single type of traffic run. Data for both *SDN+OvS* and *SDN+P4* will be presented under each case study followed by their analysis.

4.1. Case Study 1—ICMP

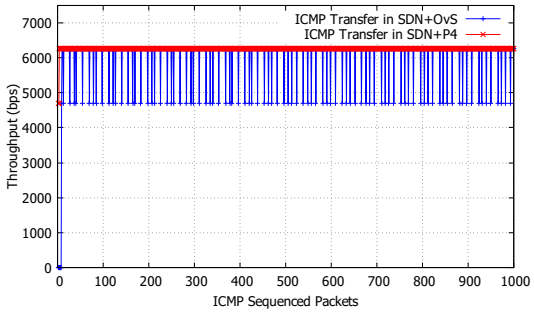
The ICMP traffic for the configuration given in Table 1 commenced at the same time. At the client end, Wireshark collected 1000 ICMP packets. Following the completion of the transfer, the average throughput for each connection was recorded.

Figure 3 presents the ICMP throughput for *SDN+OvS* networking and *SDN+P4* networking, over the three topologies given in Figure 1. For multi-path and grid topologies, the *SDN+P4* networking has maintained a steady stream of ICMP traffic with a constant throughput of approximately 6300bps until the transmission ends (see the red line in Figures 3a and 3b). The *SDN+OvS* emulation recorded recurring drops in throughput from approximately 6300bps to 4800bps (see the blue lines in Figure 3a,b). The same patterns occurred in Figure 3c, the transit-stub network model, with values of approximately 7800bps for *SDN+P4* and recurring drops in throughout to approximately 6200bps for *SDN+OvS*. We accumulated results over ten executions of experiments to inspect the average values of throughput. However, in Figure 3c, the throughput has registered an aperiodic drop for *SDN+OvS* as depicted by blue lines. This can be due to one of the two following factors, processing requirements or congestion. When faced with processing requirements or congestion, SDN networks tend to face performance degradation and cause packet loss. The same has also been observed by [62].

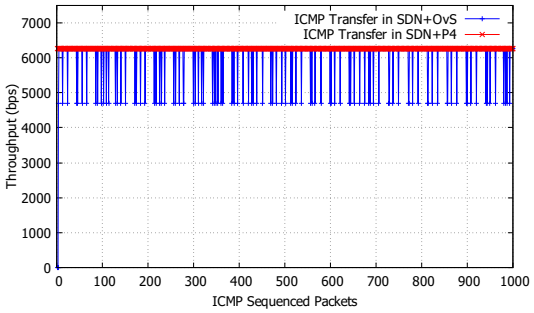
We investigated the throughput drop for the *SDN+OvS* experiments. Figure 4 shows the throughput drops at 12s and 292s (Figure 4a,b) together with the Wireshark captures Figure 4c,d at those times for multi-path topology. Out of the two controller states, reactive and proactive, the reactive state was chosen to avoid misconfigurations during the installation of FlowRules using the Reactive Forwarding application in the ONOS controller. Given that the controller was in a reactive state, PACKET_IN and PACKET_OUT control messages played a crucial role in maintaining ARP/IP address mapping information [63]. The drop in throughput for *SDN+OvS* resulted from the ARP and LLDP packets (observed in Figure 4c,d) being routed towards the slow path of the OvS, detouring from the caching layer. A detour occurs when the OvS sends a PACKET_IN message to the SDN controller to resolve the headers. Until a response arrives as a form of PACKET_OUT from the SDN controller, the packet will

 doi:10.20944/preprints202504.2530.v1

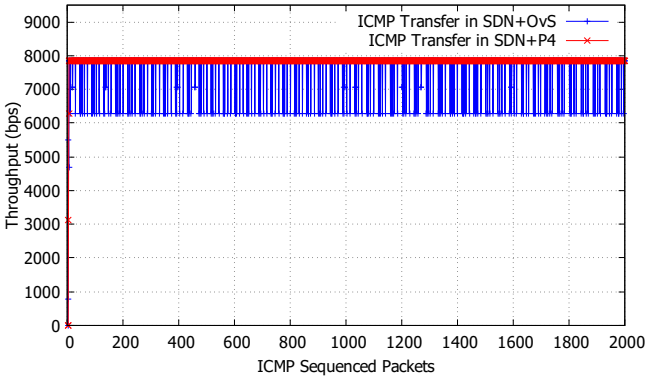
remain in the switch buffer. Deviating from the caching layer or fast path can create an overhead [64] in the network which leads to a deterioration in performance. Research in [13,65,66] established that ARP messages can create a bottleneck in the network due to high volume and frequency [67–69]. Therefore, deviating from a fast path in the OvS architecture can create a substantial overhead in the network. Processing requirements from the CPU towards the ARP/LLDP packets can also affect the processing of application centric packets, ultimately resulting in a poor quality of service. Since each emulated host represents a networked host with Ubuntu 18.04, the default ARP update time is 60s. Hence, a periodic ARP update will be generated automatically. The results in Figures 3 and 4 demonstrate how the network performance has been affected in the SDN+OvS architecture through a periodic ARP updates.



(**a**) ICMP transfer of 1000 packets in multi-path topology

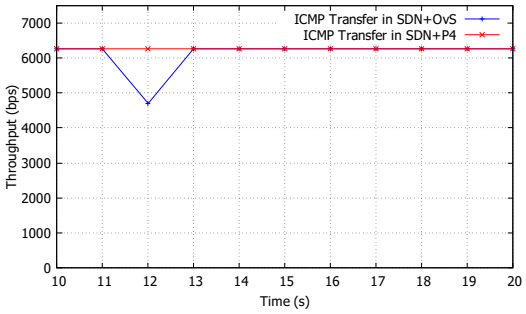


(**b**) ICMP transfer of 1000 packets in grid topology

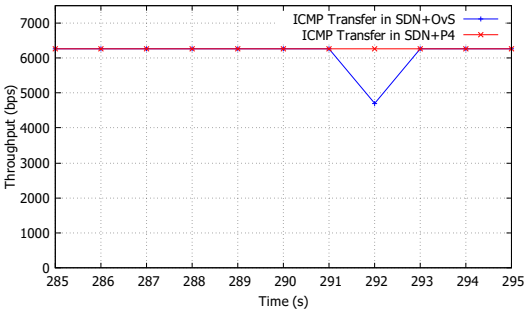


(**c**) ICMP transfer of 2000 packets in transit-stub network model

Figure 3. Throughput of ICMP transfer in case study 1



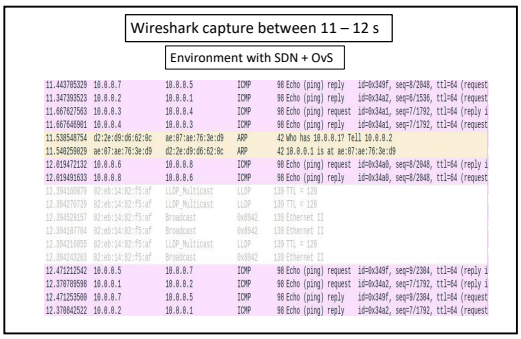
(**a**) ICMP throughput at 10-20s



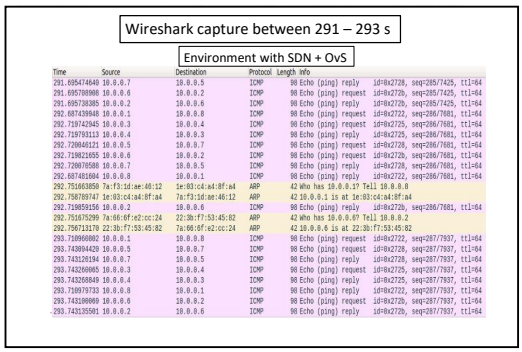
(**b**) ICMP throughput at 285-295s

Figure 4. *Cont.*

 doi:10.20944/preprints202504.2530.v1



(**c**) Wireshark capture of ARP and LLDP messages at 11-12s



(**d**) Wireshark capture of ARP and LLDP messages at 291-293s

Figure 4. ICMP data capture at 10-20s and 285-295s in multi-path topology

In the *SDN+P4* experiments, the P4 switch processed headers in parallel, thus the LLDP and ARP packets did not interfere with ICMP packets in the pipeline. The *SDN+P4* environment, therefore, did not experience a throughput drop (as shown by the red line in Figure 3). Due to parallel processing for headers and packets, the bmv2 switch was able to process packets faster, reducing delay in the core network. Given the periodic nature of the ARP packets (60s by default), the exponential growth associated with a higher number of connected devices and ARP storms caused by network outages, a bottleneck may occur if the packets are not processed faster. Hence utilising *SDN+P4* will help in meeting some stringent requirements for future applications given its ability to process packets faster for both application centric and network centric traffic.

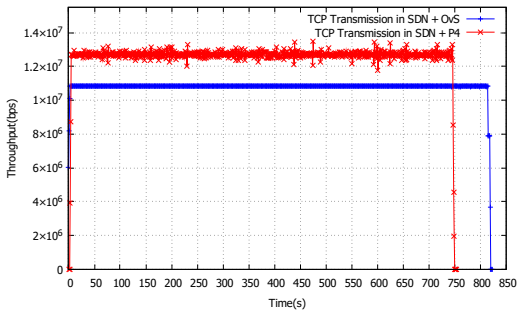
4.2. Case Study 2—TCP

TCP traffic was generated via a file download of 1GB using iPerf for multi-path and grid topologies. For the Internet topology, a transmission of TCP for 1200s was employed. See Figure 1 and Table 1 for respective topologies and traffic configuration. The results for the experiments are shown in Figure 5 with a summary of the results given in Table 2.

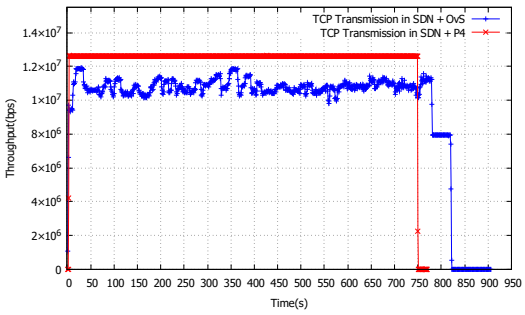
In *SDN+P4* (red line) TCP has achieved a higher and more constant throughput in comparison to *SDN+OvS* (blue line). In Figure 5a,b, the fixed sized TCP download completed earlier for SDN+P4 in approximately 750s as opposed to approximately 820s for *SDN+OvS*. In Figure 5b,(blue line) experienced fluctuations of throughput. TCP traffic, by design, requires acknowledgements. As the data packets and the acknowledgements traverse in the same route (as observed by accessing the ONOS GUI), the throughput fluctuates due to the saturation of links. It is also noteworthy that for both *SDN+P4* and *SDN+OvS*, TCP traffic traversed in the same route for the grid topology. However, the *SDN+P4* didn't record a fluctuation of throughput due to faster processing capability. For multi-path topology, (Figure 5a) a fluctuation of throughput was not recorded in the *SDN+OvS* due to the small size of the network and fewer nodes with processing requirements. For the Internet topology (Figure 5c), similar results were observed as per multi-path topology. This is because the links in the Internet topology were not saturated with network traffic.

Figure 5c presents the transmission of TCP traffic in the Internet topology. In contrast to a fixed download size (Figures 5a,b), the TCP traffic file downloading from iPerf was run with a time constraint of 1200s. Again, *SDN+P4* (red) achieves a higher throughput than that achieved with *SDN+OvS* (blue), demonstrating that P4 switches process packet forwarding in *SDN+P4* faster than OvS switches in *SDN+OvS*.

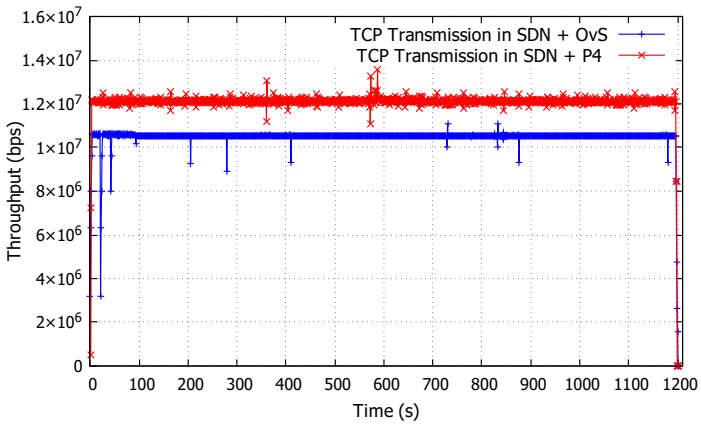
 doi:10.20944/preprints202504.2530.v1



(**e**) TCP transfer of 1GB data using iPerf in multi-path topology



(**f**) TCP transfer of 1GB data using iPerf in grid topology



(**g**) TCP transfer for 1200s using iPerf in the Internet topology

Figure 5. Throughput of TCP data transfer in case study 2

Table 2 summarises the performance metrics in terms of throughput, packet loss, Syn packet delay, delay, amount of data transmitted and transmission time for this group of experiments, averaged over the connections for each of the topologies given in Table 1. In all counts, *SDN+P4* has shown significant improvement over *SDN+OvS*. Throughput has shown improvements in all three topologies, by 9%, 11% and 12% respectively. Syn delay has also been reduced for SDN+P4, by 77%, 94% and 64%, respectively, for the three topologies. The TCP Syn Delay [70] has been calculated by determining which switch architecture can complete the TCP handshake more efficiently and effectively to establish a TCP connection. Given the nature of the TCP traffic, SYN packets play a crucial rule in establishing a TCP connection. As shown in Table 2, *SDN+P4* has spent the least amount of time in synchronising the stateful connection between the respective client and server. Having achieved a higher throughput for all cases in *SDN+P4*, total transmission time has been reduced by 8% and 9%, respectively, for multi-path and grid topologies, with a 12% increase on the total data transmitted for the Internet topology. The application of *SDN+P4* has improved the performance of the network significantly.

 doi:10.20944/preprints202504.2530.v1

a Network performance of TCP transfer in multi-path topology

	Multi-path Topology—Single Type of Traffic (TCP)		
	SDN+OvS	*SDN+P4*	Improvement
Throughput	1.09×10^7 bps	1.19×10^7 bps	9.17%
Packet Loss	4%	0.75%	(-81.25%)
Delay	0.0041 s	0.0035 s	(-14.6%)
Syn Delay	0.43 ms	0.096 ms	(-77.6%)
Data transmitted	1 GB	1 GB	N/A
Total transmission time	822.8 s	752.1 s	(-8.5%)

b Network performance of TCP transfer in grid topology

	Grid Topology—Single Type of Traffic (TCP)		
	SDN+OvS	*SDN+P4*	Improvement
Throughput	1.05×10^7 bps	1.175×10^7 bps	11.9%
Packet Loss	5.9%	3.9%	(-33.8%)
Delay	0.06 s	0.03 s	(-50%)
Syn Delay	0.94 ms	0.05 ms	(-94.68%)
Data transmitted	1 GB	1 GB	N/A
Total transmission time	820.2 s	746.1 s	(-9.0%)

c Network performance of TCP transfer in Internet topology

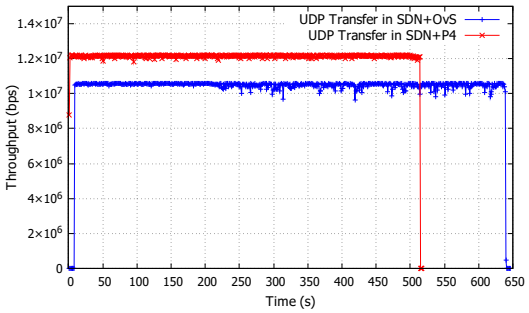
	Internet Topology—Single Type of Traffic (TCP)		
	SDN+OvS	*SDN+P4*	Improvement
Throughput	1.04×10^7bps	1.169×10^7bps	12.4%
Packet Loss	1.8%	3.7%	(-51.4%)
Delay	2.87 s	0.94 s	(-67.24%)
Syn Delay	3.89 ms	1.4 ms	(-64.01%)
Data transmitted	1.0796 GB	1.2106 GB	12.13%
Total transmission time	1200 s	1200 s	N/A

Table 2. Network performance for case study 2—TCP traffic

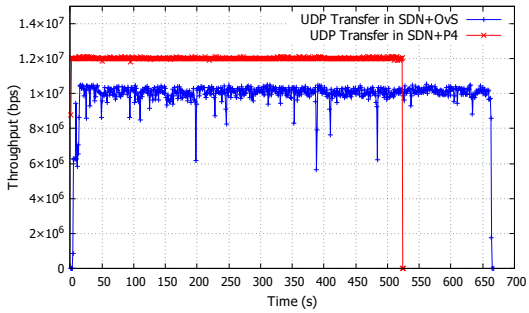
4.3. Case Study 3—UDP

In case study 3, UDP traffic was generated via file downloads using iPerf. Clients downloaded a file of 1GB from respective servers for multi-path and grid topologies, and considered a UDP transmission of 1200s for the Internet topology. The results of the experiments are presented in Figure 6 together with a summary of the results given in Table 3.

For all three experiments, Figures 6a, 6b and 6c show that *SDN+P4* achieved a higher and constant throughput in comparison to *SDN+OvS*. The initial delay in *SDN+OvS* was longer than for *SDN+P4*, because the route discovery delay is significant in *SDN+OvS* but less so with *SDN+P4*. With a successful population of FlowRules, consuming a greater time for route discovery, bottlenecks can result in real-time applications that involve a demanding service such as, for example, Vehicle-to-Everything (V2X).



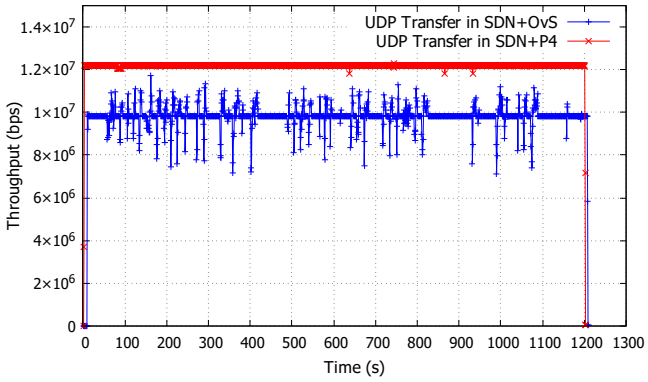
(**a**) UDP transfer of 1GB data using iPerf in multi-path topology



(**b**) UDP transfer of 1GB data using iPerf in grid topology

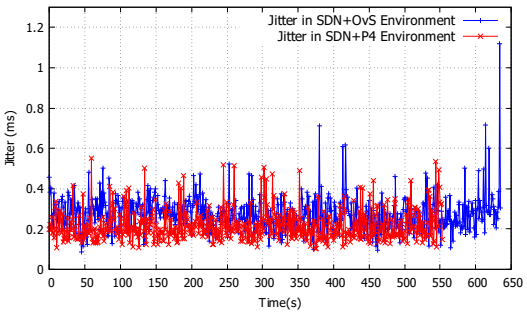
Figure 6. *Cont.*

 doi:10.20944/preprints202504.2530.v1

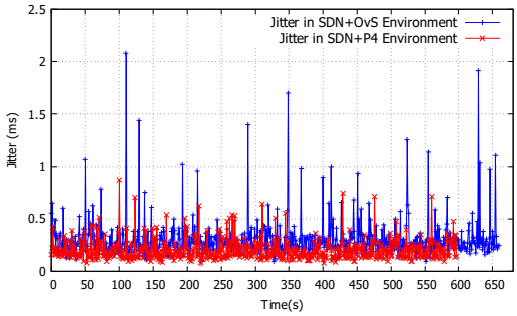


(**c**) UDP transfer for 1200s using iPerf in the Internet topology

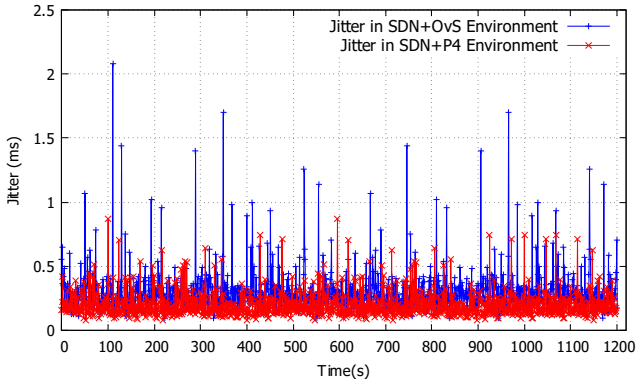
Figure 6. Throughput of UDP data transfer in case study 3



(**d**) Jitter experienced for UDP Transmission in multi-path topology



(**e**) Jitter experienced for UDP Transmission in grid topology



(**f**) Jitter experienced for UDP Transmission in the Internet topology

Figure 7. Jitter of UDP data transmission in case study 3

In Figure 6a, the fixed size UDP download completed earlier for *SDN+P4* (in approximately 510s (Figure 6a), than for *SDN+OvS* which took approximately 640s, (Figure 6b). In both *SDN+P4* and *SDN+OvS*, traffic traversed the same route for topologies I and II. In contrast to Figure 6a,b, the UDP traffic file download in iPerf ran for a fixed 1200s in the Internet topology. As shown in Figure 6c, *SDN+P4* (red) achieved a higher throughput than *SDN+OvS* (blue). In all cases the P4 switches process packet forwarding in *SDN+P4* faster than non-P4 switches in *SDN+OvS*.

For case study 3, Figure 7a–c present jitter for UDP packets. As observable by the Figure 7a,b, jitter in *SDN+OvS* (blue lines) have recorded a lower jitter than that of collated in *SDN+P4*. Albeit being recorded for a fraction of a few seconds in a quasi-periodic trend, an argument can be presented that these should not contribute towards an application that requires stringent jitter constraints. The primary reason for this quasi-periodic pattern of higher jitter is the higher throughput that was recorded

 doi:10.20944/preprints202504.2530.v1

by *SDN+P4*. This has been corroborated by the authors at [71]. In Table 3a, *SDN+OvS* recorded an average jitter of 0.298ms while *SDN+P4* produced an average of 0.231 ms using the Equation (1).

$$\frac{1}{M}\sum_{f=1}^{M}\sum_{i=1}^{N_f} J_{fi}/N_f \tag{1}$$

Here M represents the number of flows in the network, N_f denotes the number of packets in flow f and J_{fi} represents the jitter in packet i of flow f. With the application of Equation (1) and the analysis of jitter, *SDN+P4* recorded a reduction in jitter of 22% in comparison to *SDN+OvS*.

a Network performance of UDP transfer in simple multi-path topology

	Multi-path Topology—Single Type of Traffic (UDP)		
	SDN+OvS	*SDN+P4*	Improvement
Throughput	1.06*10^7bps	1.2*10^7bps	13%
Packet Loss	22%	9%	(-59%)
Delay	1.9519s	0.3221s	(-83%)
Jitter	0.298 ms	0.231 ms	(-22%)
Data transmitted	1GB	1GB	N/A
Total transmission time	640s	514s	(-14%)

b Network performance of UDP transfer in simple-grid topology

	Grid Topology—Single Type of Traffic (UDP)		
	SDN+OvS	*SDN+P4*	Improvement
Throughput	1.04*10^7bps	1.2*10^7bps	15%
Packet Loss	37%	7%	(-81%)
Delay	1.9344s	0.3744s	(-80%)
Jitter	0.371 ms	0.204 ms	(-45%)
Data transmitted	1GB	1GB	N/A
Total transmission time	670s	524s	(-17%)

c Network performance of UDP transfer in the Internet topology

	Internet Topology—Single Type of Traffic (UDP)		
	SDN+OvS	*SDN+P4*	Improvement
Throughput	0.98*10^7bps	1.2*10^7bps	22%
Packet Loss	39%	8%	(-79%)
Delay	1.9905s	1.0432s	(-60%)
Jitter	0.301 ms	0.2007 ms	(-33.3%)
Data transmitted	1.4723GB	1.6178GB	10%
Total transmission time	1200s	1200s	N/A

Table 3. Network performance for case study 3—UDP traffic

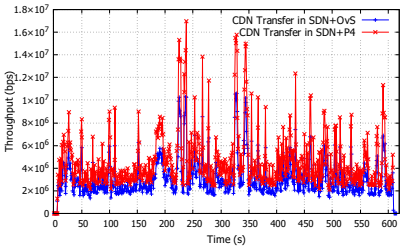
Table 3 summarises the performance metrics in terms of throughput, packet loss, delay, amount of data transmitted and transmission time for this group of experiments, averaged over the respective connections given in Table 1. In all cases, *SDN+P4* has demonstrated a significant improvement in network related statistics in comparison to *SDN+OvS*. For example, the grid topology has demonstrated a 13% improvement of throughput in *SDN+P4* in comparison to *SDN+OvS*. *SDN+P4* has reduced the packet loss from 22% to 9%, a reduction of 59%. Recorded delay for UDP traffic has reduced from 1.95s for *SDN+OvS* to 0.32s for *SDN+P4*, a reduction of 83%. Lastly, the total transmission time has reduced by 14% due to the faster download speed achieved by *SDN+P4* over that achieved with *SDN+OvS*. Although the above results for each case study were conducted on software switches, the experiments can be conducted on hardware switches to further support our hypothesis with improved results [21].

4.4. Case Study 4- Content Delivery Network

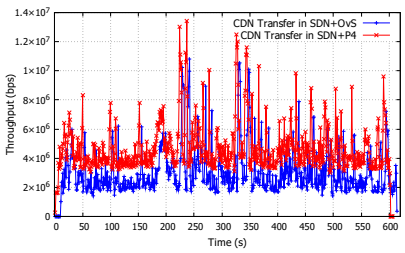
In this case study, we used VLC-player for our experiments for its ability to program video streaming via a Python script and collect data for video traffic and its lightweight deployment. Figure 8 presents the captured data of a video stream for all three topologies.

 doi:10.20944/preprints202504.2530.v1

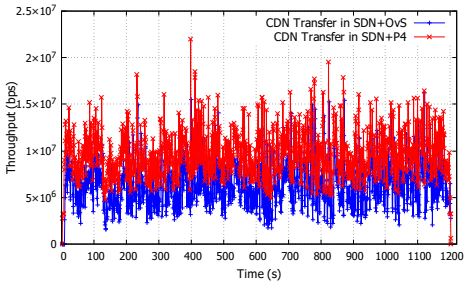
Figure 8a–c show that for each of the three topologies, the video for *SDN+P4* had a higher throughput than with *SDN+OvS*. Table 4 shows further that, throughput has improved by 60% in the multi-path topology, 77% in the grid topology and 60% in the Internet topology for *SDN+P4* in comparison to *SDN+OvS*. The high throughput achieved with *SDN+P4*, resulted in an increased amount of data being transferred between hosts. As shown in Table 4, the amount of data transmitted improved by 46% in the multi-path topology, 50% in the grid topology and 43% in the Internet topology for *SDN+P4* in comparison to *SDN+OvS*. A notable delay of initiating the video stream can be observed in Figure 8a–c for *SDN+OvS* (blue line). This again is due to the delay caused in route discovery, resulting in a delayed start for the video with later termination for *SDN+OvS*. This can be seen by observing the averaged buffering delay illustrated in Table 4. Buffering delay in *SDN+P4* has been reduced by 69%, 84% and 83% for topologies I, II and III respectively in comparison to *SDN+OvS*. Figures 9a–c represent the observed jitter during the CDN transmission. Equation (1) was used once more to calculate the averaged values for the jitter. As shown in Table 4, jitter in *SDN+P4* has been reduced by 35%, 41% and 23% respectively in comparison to *SDN+OvS* for all three topologies. In our experiments, multi-path and grid topologies streamed a video for 600s, whilst the Internet topology streamed a video for 1200s. Although the streaming times are different, the performance gain remained constant across all three topologies.



(**a**) Live video stream in multi-path topology for 600s

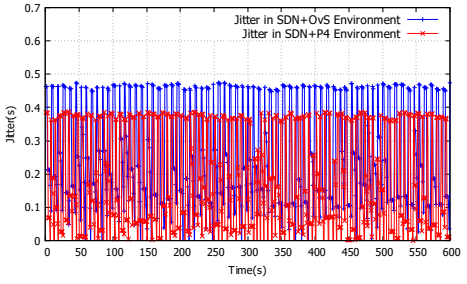


(**b**) Live video stream in grid topology for 600s

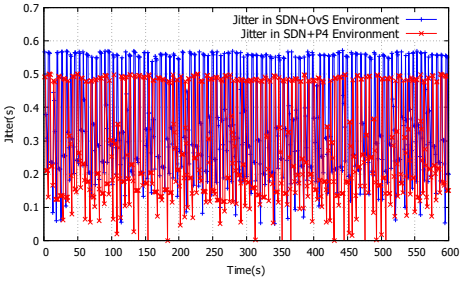


(**c**) Live video stream in the Internet topology for 1200s

Figure 8. Throughput of video content transfer in case study 4



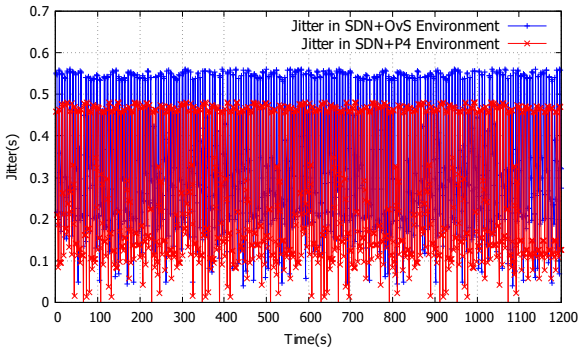
(**a**) Jitter experienced in CDN Transmission in multi-path topology



(**b**) Jitter experienced in CDN Transmission in grid topology

Figure 9. *Cont.*

 doi:10.20944/preprints202504.2530.v1



(**c**) Jitter experienced in CDN Transmission in the Internet topology

Figure 9. Jitter of CDN data transmission in case study 4

a Network performance of CDN transfer in multi-path topology

	Multi-path Topology—Single Type of Traffic (CDN)		
	SDN+OvS	SDN+P4	Improvement
Throughput	1.0*10^{6}bps	1.6*10^{6}bps	60%
Data transmitted	667.63MB	973.61MB	46%
Buffering delay	8.672s	2.7s	(-69%)
Jitter	0.255ms	0.164ms	(-35.6%)

b Network performance of CDN transfer in grid topology

	Grid Topology—Single Type of Traffic (CDN)		
	SDN+OvS	SDN+P4	Improvement
Throughput	0.9*10^{6}bps	1.6*10^{6}bps	77%
Data transmitted	647.63MB	973.61MB	50%
Buffering delay	16.9s	2.7s	(-84%)
Jitter	0.351ms	0.207ms	(-41%)

c Network performance of CDN transfer in the Internet topology

	Internet Topology—Single Type of Traffic (CDN)		
	SDN+OvS	SDN+P4	Improvement
Throughput	1.0*10^{6}bps	1.6*10^{6}bps	60%
Data transmitted	1609.17MB	2308.88MB	43%
Buffering delay	12.94s	2.17s	(-83%)
Jitter	0.342ms	0.261ms	(-41%)

Table 4. Network performance for case study 4—CDN traffic

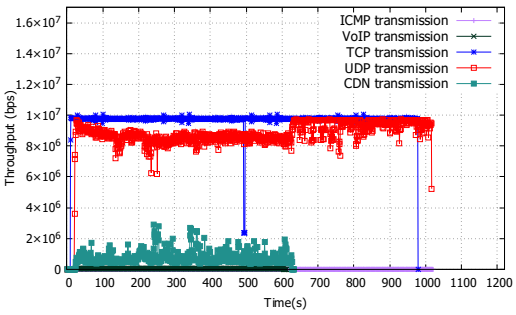
5. Results and Analysis of Tier-II—Multiple Types of Traffic Running Simultaneously

Having investigated the performance of single types of traffic running over different network topologies for *SDN+P4* and *SDN+OvS*, we now study the performance of mixed types of traffic running simultaneously over grid topology and the Internet topology.

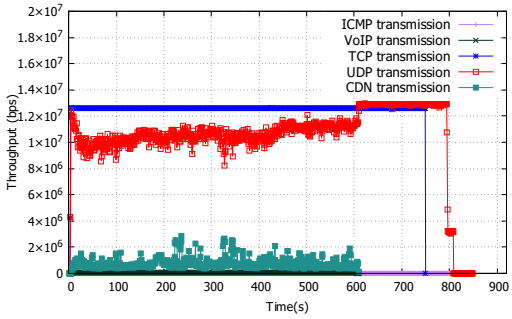
5.1. Case Study 5—Mixed Type of Traffic over Grid Topology

Five types of traffic from Table 1, ICMP, VoIP, TCP, UDP and CDN, were run simultaneously. The performance of each type of traffic is presented in Table 5. Table 5a (ICMP traffic) reports an improvement of 6.7% in throughput for *SDN+P4* in comparison to the throughput achieved by *SDN+OvS*. Table 5b (TCP traffic) records an improvement of 12% in throughput from 1.05*10^{7}bps for *SDN+OvS* to 1.18*10^7 bps for *SDN+P4*. These are represented in Figure 10a,b by the blue line. Packet loss, packet delay, synchronisation delay and total transmission time achieved significant reductions for *SDN+P4* over *SDN+OvS* of 36%, 6%, 77% and 18% respectively.

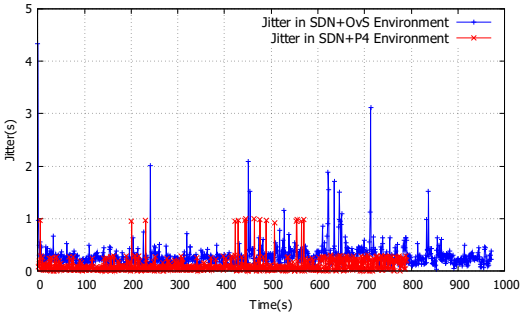
Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 30 April 2025 doi:10.20944/preprints202504.2530.v1



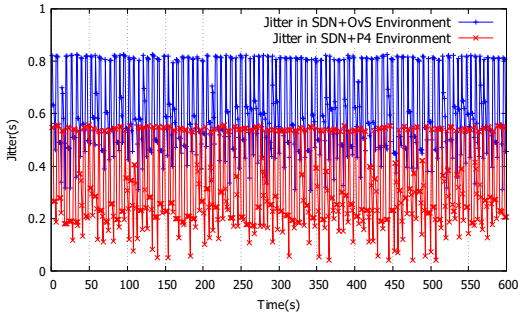
(**d**) Throughput for mixed traffic for *SDN+OvS*



(**e**) Throughput for mixed traffic for *SDN+P4*

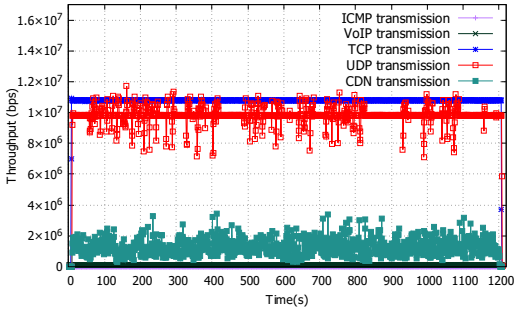


(**f**) Jitter in UDP Transmission for mixed traffic

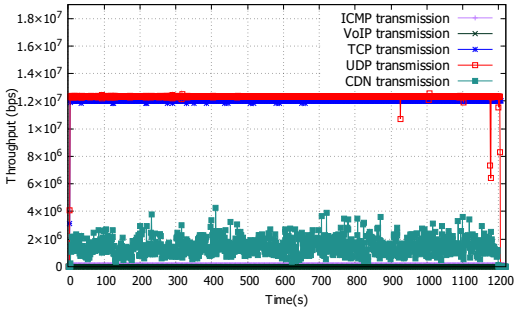


(**g**) Jitter in CDN Transmission for mixed traffic

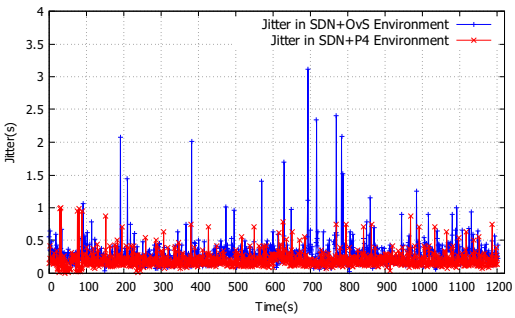
Figure 10. Data for mixed traffic over the simple-grid topology—case study 5



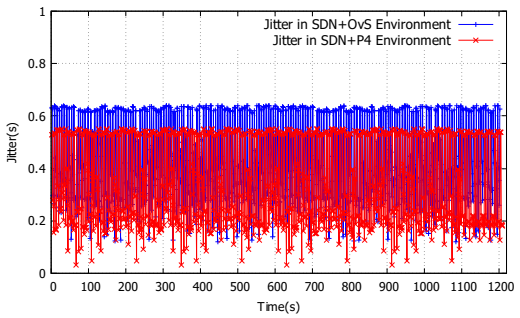
(**a**) Mixed traffic throughput for *SDN+OvS*



(**b**) Mixed traffic throughput for *SDN+P4*



(**c**) Jitter experienced in UDP Transmission while simultaneous applications occupy the network space



(**d**) Jitter experienced in CDN Transmission while simultaneous applications occupy the network space

Figure 11. Data of mixed traffic over the Internet topology —case study 6

Table 5c (UDP traffic) reports an increase in UDP throughput of 43% (from 7*10^6bps in *SDN+OvS* to 1.0*10^7bps in *SDN+P4*) as illustrated in Figures 10a,b by the red line. It was also observed that *SDN+P4* completed the download of 1GB of UDP traffic in 800s in contrast to *SDN+OvS* which took 1020 seconds. Packet loss, delay and jitter were each observed to have reduced values for *SDN+P4*

 doi:10.20944/preprints202504.2530.v1

in comparison to those obtained using *SDN+OvS*, these being 73%, 78% and 48% respectively. The reduction in jitter for UDP packets is shown in Figure 10c.

a Performance of ICMP traffic in case study 5

	Grid Topology—Mixed Traffic (ICMP)		
	SDN+OvS	*SDN + P4*	Improvements
Throughput	5.85*10^3 bps	6.3*10^3 bps	6.7%

b Performance of TCP traffic in case study 5

	Grid Topology—Mixed Traffic (TCP)		
	SDN + OvS	SDN + P4	Improvement
Throughput	1.05*10^7bps	1.18*10^7bps	12%
Packet Loss	7.9%	5%	(-36%)
Delay	3.1 s	2.9s	(-6%)
Syn Delay	2.6 ms	0.6 ms	(-77%)
Data transmitted	1 GB	1 GB	N/A
Total transmission time	982.1 s	807 s	(-18%)

c Performance of UDP traffic in case study 5

	Grid Topology—Mixed Traffic (UDP)		
	SDN+OvS	*SDN+P4*	Improvement
Throughput	7*10^6bps	1*10^7bps	43%
Packet Loss	38%	10%	(-73%)
Delay	1.9157s	0.4064s	(-78%)
Jitter	0.480 ms	0.247 ms	(-48%)
Total transmission time	1020s	800s	(-21%)

d Performance of CDN traffic in case study 5

	Grid Topology—Mixed Traffic (CDN)		
	SDN+OvS	SDN+P4	Improvement
Throughput	3.8*10^5bps	4*10^5bps	5%
Data transmitted	588.41 MB	817.54 MB	38%
Buffering delay	3.11s	1.91s	(-38%)
Jitter	0.351ms	0.33ms	(-5%)

Table 5. Network Performance of for case study 5

Table 5d (CDN traffic) records an increase of 5% in throughput from 3.8*10^5bps for *SDN+OvS* to 4*10^5bps for *SDN+P4*. This is illustrated by the green line in Figures 10a,b. Higher throughput resulted in *SDN+P4* transferring more data (817.54MB) in comparison to *SDN+OvS* (588.41MB), an improvement of 38% for the same video file. Buffer delay and jitter in *SDN+P4* resulted in significant reductions of 38% and 5% respectively, in contrast to *SDN+OvS*. The reduction in jitter is shown in Figure 10d.

The difference in throughput has an effect on the quality of the video. Figure 12a shows a comparison for the 34^{th} frame captured during the experiments for both *SDN+OvS* and *SDN+P4*. The same frame was sent from the server side, with the client side in *SDN+OvS* displaying a distorted frame in comparison to that received by the client with *SDN+P4*.

 doi:10.20944/preprints202504.2530.v1

Grid Topology – Mixed Traffic (CDN)				
SDN+OvS			*SDN+P4*	
Server		Client	Server	Client
				
Observing time	90.0s	90.0s	90.0s	90.0s
Sequence No of the Frame observed	34th	34th	34th	34th

(a)

Internet Topology – Mixed Traffic (CDN)				
SDN+OvS			*SDN+P4*	
Server		Client	Server	Client
				
Observing time	90.0s	90.0s	90.0s	90.0s
Sequence No of the Frame observed	34th	33rd	34th	34th

(b)

Figure 12. Comparison of CDN frames for both *SDN+OvS* and *SDN+P4* for case studies 5 and 6.

5.2. Case Study 6—Simultaneous Run over the Internet Topology

Table **??** presents the data collected during this case study. In Table **??**a, the ICMPthroughput shows has seen an improvement of 6.7% for *SDN+P4* in comparison to *SDN+OvS*.

Table **??**b (TCP traffic) records an increase in TCP throughput of 19% from $0.99*10^7$bps for *SDN+OvS* to $1.18*10^7$bps for *SDN+P4*. These are illustrated by the blue lines in Figure 11a,b. Packet loss, packet delay and synchronisation delay for *SDN+P4* reflect significant reductions in comparison to that achieved using *SDN+OvS* of 75%, 65% and 45% respectively. The data transmitted achieved a 23% improvement from 0.82GB for *SDN+OvS* to 1.01GB with *SDN+P4*.

Table **??**c (UDP traffic) records an increase in UDP throughput of 33% from $9*10^6$bps for *SDN+OvS* to $1.2*10^7$bps for *SDN+P4*. These are illustrated by the red lines in Figure 11a,b. Packet loss, delay and jitter in *SDN+P4* reflect significant reductions in comparison to that achieved using *SDN+OvS* of 69%, 58% and 31% respectively. A reduction in jitter for UDP packets is shown in Figure 11c. The higher speed, reduced delay, jitter and packet loss in *SDN+P4* result in an increase in data transmission (1.602GB) for *SDN+P4* in comparison to (1.456GB) for *SDN+OvS*.

Table **??**d (CDN traffic) shows an increase in throughput of 44% from $0.97*10^6$bps in *SDN+OvS* to $1.4*10^6$bps in *SDN+P4*. This is illustrated by the green line in Figure 11a,b. Higher throughput resulted in *SDN+P4* transferring more data (1451.3MB) in comparison to *SDN+OvS* (1183.4MB), an improvement of 23% for the same video file. Buffer delay and jitter in *SDN+P4* resulted in significant reductions of 61% and 21% respectively, in comparison to *SDN+OvS*. The reduction in jitter is shown in Figure 11d.

The difference in throughput has directly affected the quality of the video. Figure 12b gives a comparison for the 34^{th} frame captured during the experiments for both *SDN+OvS* and *SDN+P4*. The same frame was sent from the server side. At the client side, *SDN+OvS* displayed the previous frame (33^{rd}) in comparison to that received by the client via *SDN+P4* (34^{th}).

6. Discussion

In the context of this research, we explored the performance of SDN with *OvS* (*SDN+OvS*) and SDN with *P4* (*SDN+P4*). We have seen that *SDN+P4* outperforms *SDN+OvS* for each of our case studies using different types of traffic and topologies.

We investigated the overhead created due to the slow path utilisation of OvS and the performance variation in comparison to a P4 target switch. With the evolution of the Internet and the increased number of connected devices, networks will face congestion. With more and more packets requiring processing using a controller, a network model that utilises a slow path approach such as OvS will

 doi:10.20944/preprints202504.2530.v1

potentially lead to an exponential growth in traffic congestion. During this research, we have saturated the links in our elected network (Sections 5.1 and 5.2) to test how the network will cope with an increased load in traffic. Our results suggest that the network can maintain a gain in performance whilst links are saturated with traffic for the *SDN+P4* environment in comparison to that experienced with *SDN+OvS*.

The evolution of 5G and beyond has led to the need to evaluate methods for reducing the delay at the core. Initialising programmability in the network has been shown to increase performance at the core. To the best of our knowledge, current literature does not evaluate the performance of the network when the control plane and data plane programmability (*SDN+P4*) is employed in comparison to the control plane (SDN+OvS) programmability.

a Performance of ICMP traffic in case study 6

	Internet Topology—Mixed Traffic (ICMP)		
	SDN+OvS	*SDN+P4*	Improvements
Throughput	6.9*10^3bps	7.8*10^3bps	6.7%

b Performance of TCP traffic in case study 6

	Internet Topology—Mixed Traffic (TCP)		
	SDN+OvS	*SDN+P4*	Improvement
Throughput	0.99*10^7bps	1.18*10^7bps	19%
Packet Loss	0.4%	0.1%	(-75%)
Delay	3.2s	1.1s	(-65%)
Syn Delay	3.84ms	2.1ms	(-45%)
Data transmitted	0.82GB	1.01GB	23%

c Performance of UDP traffic in case study 6

	Internet Topology—Mixed Traffic (UDP)		
	SDN+OvS	*SDN+P4*	Improvement
Throughput	9*10^6bps	1.2*10^7bps	33%
Packet Loss	36%	11%	(-69%)
Delay	2.516s	1.034s	(-58%)
Jitter	0.292ms	0.201ms	(-31%)
Data transmitted	1.456GB	1.602GB	10%

d Performance of CDN traffic in case study 6

	Internet Topology—Mixed Traffic (CDN)		
	SDN+OvS	*SDN+P4*	Improvement
Throughput	0.97*10^6bps	1.4*10^6bps	44%
Data transmitted	1183.4MB	1451.3MB	23%
Buffering delay	20s	7.8s	(-61%)
Jitter	0.422ms	0.331ms	(-21%)

Table 6. Network performance for case study 6

We evaluated the performance of applications in Sections 4 and 5, where *SDN+P4* has been employed in contrast to *SDN+OvS*. For a time-sensitive application with minimal latency requirements, reducing the delay at the core can be essential. For example, for Vehicle-to-Vehicle (V2V) and Ultra Reliable Low Latency Communication (URLLC) applications. A solution, that processes packets in parallel as opposed to sequential processing in OvS, has been considered in this research and its effect on performance in applications. Our research has established that with the initialisation of *SDN+P4* with parallel processing of packets, various applications have achieved an improvement in performance in comparison to applications run over *SDN+OvS*. We also evaluated the quality of applications that occurred due to faster processing achieved with *SDN+P4* in comparison to *SDN+OvS*. The statistics such as increased bps and throughput, reduced delay jitter, packet loss, delay and buffering time have led to a higher quality of application at the receiver's end. Improvement of the quality of the video at the receiver end for a video streaming application (Figure 12) has also been presented.

 doi:10.20944/preprints202504.2530.v1

We investigated whether the type of traffic has a bearing on performance for *SDN+P4* in comparison to *SDN+OvS*. Given that the majority of network traffic will be accumulated by multimedia applications in the future, a fair switching mechanism that enhances the performance of such applications will provide a beneficial factor for service providers. As the majority of the downloaded traffic in a modern context consists of UDP traffic (multimedia applications) and the demand for such applications is expected to grow exponentially in the future, the employment of *SDN+P4* is a viable solution for service providers looking to provide an improved service. Protocols such as Google Quick UDP Internet Connections (GQUIC) in mobile telecommunication will also benefit from the employment of *SDN+P4*. The employment of *SDN+P4* will also benefit non-media applications where a connection-less protocol is employed by a service provider. For example, communication between a Radio Device and eNB, Multi-access Edge Clouds (MEC) or between Radio Access Networks (RAN) and Fog nodes [56] each can benefit through the use of *SDN+P4*. To meet the requirements for faster convergence and faster processing of packets, *SDN+P4* will, we believe, serve as an inventive or improved solution.

7. Conclusions

The above research has answered the research questions given at the beginning of this study. We conclude that the application of *SDN+P4* has enabled networks to improve their performance for different topologies and traffic. The results from our experiments show an improvement for, for example, delay and packet loss which have reduced significantly with *SDN+P4*, whilst throughput has increased for all case studies. *SDN+P4* has improved performance in the networks through the use of parallel processing in P4, which has complemented the standardised SDN architecture. The results for all case studies indicate that *SDN+P4* is a promising alternative to *SDN+OvS* providing a resilient approach for future networking.

Through a combination of SDN and P4 (control plane programmability with data plane programmability), we have established that it is possible to provide an improved service to clients using *SDN+P4* rather than with *SDN+OvS*. Across all case studies, parallel processing in P4 has provided an increase in available queues for processing traffic with the utilisation of a flexible parser. With the evolution of the internet and the heterogeneity of the applications being used, *SDN+P4* will, we believe, provide an improved service to all use cases for 5G and beyond. Based on the results collated from this study, we implemented an *SDN+P4* environment in our 5G testbed paper [72].

References

1. Kirkpatrick, K. Software-defined networking. *Communications of the ACM* **2013**, *56*, 16–19.
2. Kreutz, D.; Ramos, F.M.; Verissimo, P.E.; Rothenberg, C.E.; Azodolmolky, S.; Uhlig, S. Software-defined networking: A comprehensive survey. *Proceedings of the IEEE* **2014**, *103*, 14–76.
3. McKeown, N.; Anderson, T.; Balakrishnan, H.; Parulkar, G.; Peterson, L.; Rexford, J.; Shenker, S.; Turner, J. OpenFlow: Enabling innovation in campus networks. *ACM SIGCOMM Computer Communication Review* **2008**, *38*, 69–74.
4. Nygren, A.; Pfaff, B.; Lantz, B.; Heller, B.; Barker, C.; Beckmann, C.; Cohn, D.; Malek, D.; Talayco, D.; Erickson, D.; et al. Openflow switch specification version 1.5. 1. *Open Networking Foundation, Tech. Rep* **2015**.
5. Isolani, P.H.; Wickboldt, J.A.; Both, C.B.; Rochol, J.; Granville, L.Z. Interactive monitoring, visualization, and configuration of OpenFlow-based SDN. In Proceedings of the 2015 IFIP/IEEE International Symposium on Integrated Network Management (IM). IEEE, 2015, pp. 207–215.
6. Bosshart, P.; Daly, D.; Gibb, G.; Izzard, M.; McKeown, N.; Rexford, J.; Schlesinger, C.; Talayco, D.; Vahdat, A.; Varghese, G.; et al. P4: Programming protocol-independent packet processors. *ACM SIGCOMM Computer Communication Review* **2014**, *44*, 87–95.
7. Sivaraman, A.; Kim, C.; Krishnamoorthy, R.; Dixit, A.; Budiu, M. Dc. p4: Programming the forwarding plane of a data-center switch. In Proceedings of the Proceedings of the 1st ACM SIGCOMM Symposium on Software Defined Networking Research, 2015, pp. 1–8.
8. Index, C.V.N. Forecast and methodology, 2016–2021. *White Paper, June* **2017**.
9. Cisco, V. The zettabyte era: Trends and analysis. Updated (07/06/2017), 2017.

 doi:10.20944/preprints202504.2530.v1

10. Ericsson. Mobile Data Traffic Forecast—Ericsson Mobility Report, 2024. Accessed: 2025-04-22.
11. Sugeng, W.; Istiyanto, J.E.; Mustofa, K.; Ashari, A. The impact of QoS changes towards network performance. *International Journal of Computer Networks and Communications Security* **2015**, *3*, 48–53.
12. Monserrat, J.F.; Mange, G.; Braun, V.; Tullberg, H.; Zimmermann, G.; Bulakci, Ö. METIS research advances towards the 5G mobile and wireless system definition. *EURASIP Journal on Wireless Communications and Networking* **2015**, *2015*, 53.
13. Boteanu, V.; Bagheri, H.; Pels, M. Minimizing ARP traffic in the AMS-IX switching platform using OpenFlow. *Cited on* **2013**, p. 7.
14. Mahmood, A.; Yuksel, M. Resource Sharing on the Internet: A Comprehensive Survey on ISP Peering. *ACM Computing Surveys* **2025**.
15. Production Quality, Multilayer Open Virtual Switch.
16. Cerović, D.; Del Piccolo, V.; Amamou, A.; Haddadou, K.; Pujolle, G. Fast packet processing: A survey. *IEEE Communications Surveys & Tutorials* **2018**, *20*, 3645–3676.
17. Hauser, F.; Häberle, M.; Merling, D.; Lindner, S.; Gurevich, V.; Zeiger, F.; Frank, R.; Menth, M. A survey on data plane programming with p4: Fundamentals, advances, and applied research. *Journal of Network and Computer Applications* **2023**, *212*, 103561.
18. Sharma, N.K.; Kaufmann, A.; Anderson, T.; Krishnamurthy, A.; Nelson, J.; Peter, S. Evaluating the power of flexible packet processing for network resource allocation. In Proceedings of the 14th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 17), 2017, pp. 67–82.
19. da Silva, J.S.; Boyer, F.R.; Chiquette, L.O.; Langlois, J.P. Extern objects in P4: An ROHC header compression scheme case study. In Proceedings of the 2018 4th IEEE Conference on Network Softwarization and Workshops (NetSoft). IEEE, 2018, pp. 517–522.
20. Wang, H.; Soulé, R.; Dang, H.T.; Lee, K.S.; Shrivastav, V.; Foster, N.; Weatherspoon, H. P4fpga: A rapid prototyping framework for p4. In Proceedings of the Proceedings of the Symposium on SDN Research, 2017, pp. 122–135.
21. Chen, X.; Zhang, D.; Wang, X.; Zhu, K.; Zhou, H. P4sc: Towards high-performance service function chain implementation on the p4-capable device. In Proceedings of the 2019 IFIP/IEEE Symposium on Integrated Network and Service Management (IM). IEEE, 2019, pp. 1–9.
22. Shahbaz, M.; Choi, S.; Pfaff, B.; Kim, C.; Feamster, N.; McKeown, N.; Rexford, J. Pisces: A programmable, protocol-independent software switch. In Proceedings of the Proceedings of the 2016 ACM SIGCOMM Conference, 2016, pp. 525–538.
23. Patra, P.G.K.; Cesen, F.E.R.; Mejia, J.S.; Feferman, D.L.; Csikor, L.; Rothenberg, C.E.; Pongracz, G. Toward a Sweet Spot of Data Plane Programmability, Portability, and Performance: On the Scalability of Multi-Architecture P4 Pipelines. *IEEE Journal on Selected Areas in Communications* **2018**, *36*, 2603–2611.
24. Saqib, M.; Elbiaze, H.; Glitho, R.; Ghamri-Doudane, Y. An Intelligent & Programmable Data Plane for QoS-Aware Packet Processing. *IEEE Transactions on Machine Learning in Communications and Networking* **2024**.
25. Fares, O.; Dandoush, A.; Aitsaadi, N. Sdn-based platform enabling intelligent routing within transit autonomous system networks. In Proceedings of the 2022 IEEE 19th Annual Consumer Communications & Networking Conference (CCNC). IEEE, 2022, pp. 909–912.
26. Bagci, K.T.; Tekalp, A.M. SDN-enabled distributed open exchange: Dynamic QoS-path optimization in multi-operator services. *Computer Networks* **2019**, *162*, 106845.
27. Calvert, K.L.; Doar, M.B.; Zegura, E.W. Modeling internet topology. *IEEE Communications magazine* **1997**, *35*, 160–163.
28. Kotronis, V.; Gämperli, A.; Dimitropoulos, X. Routing centralization across domains via SDN: A model and emulation framework for BGP evolution. *Computer Networks* **2015**, *92*, 227–239.
29. Kotronis, V.; Klöti, R.; Rost, M.; Georgopoulos, P.; Ager, B.; Schmid, S.; Dimitropoulos, X. Stitching inter-domain paths over IXPs. In Proceedings of the Proceedings of the Symposium on SDN Research, 2016, pp. 1–12.
30. Heller, B.; Sherwood, R.; McKeown, N. The controller placement problem. *ACM SIGCOMM Computer Communication Review* **2012**, *42*, 473–478.
31. Fonseca, P.; Bennesby, R.; Mota, E.; Passito, A. A replication component for resilient OpenFlow-based networking. In Proceedings of the 2012 IEEE Network operations and management symposium. IEEE, 2012, pp. 933–939.

 doi:10.20944/preprints202504.2530.v1

32. Hock, D.; Hartmann, M.; Gebert, S.; Jarschel, M.; Zinner, T.; Tran-Gia, P. Pareto-optimal resilient controller placement in SDN-based core networks. In Proceedings of the Proceedings of the 2013 25th International Teletraffic Congress (ITC). IEEE, 2013, pp. 1–9.
33. Machado, C.C.; Granville, L.Z.; Schaeffer-Filho, A. ANSwer: Combining NFV and SDN features for network resilience strategies. In Proceedings of the 2016 IEEE Symposium on Computers and Communication (ISCC). IEEE, 2016, pp. 391–396.
34. Zhang, X.; Wei, K.; Guo, L.; Hou, W.; Wu, J. SDN-based resilience solutions for smart grids. In Proceedings of the 2016 International Conference on Software Networking (ICSN). IEEE, 2016, pp. 1–5.
35. Smith, P.; Schaeffer-Filho, A.; Hutchison, D.; Mauthe, A. Management patterns: SDN-enabled network resilience management. In Proceedings of the 2014 IEEE Network Operations and Management Symposium (NOMS). IEEE, 2014, pp. 1–9.
36. Fernando, O.; Xiao, H.; Che, X. Evaluation of Underlying Switching Mechanism for Future Networks with P4 and SDN (Workshop Paper). In Proceedings of the International Conference on Collaborative Computing: Networking, Applications and Worksharing. Springer, 2019, pp. 549–568.
37. Sanvito, D.; Moro, D.; Gulli, M.; Filippini, I.; Capone, A.; Campanella, A. ONOS Intent Monitor and Reroute service: Enabling plug&play routing logic. In Proceedings of the 2018 4th IEEE Conference on Network Softwarization and Workshops (NetSoft). IEEE, 2018, pp. 272–276.
38. Team, M.
39. onos. ONOS Project.
40. Ruchel, L.V.; Turchetti, R.C.; de Camargo, E.T. Evaluation of the robustness of SDN controllers ONOS and ODL. *Computer Networks* **2022**, *219*, 109403.
41. Mamushiane, L.; Shozi, T. A QoS-based evaluation of SDN controllers: ONOS and OpenDayLight. In Proceedings of the 2021 IST-Africa Conference (IST-Africa). IEEE, 2021, pp. 1–10.
42. Consortium, P.L.; et al. Behavioral model (bmv2). *URL: https://github. com/p4lang/behavioral-model [cited 2020-01-21]* **2018**.
43. Alizadeh, M.; Edsall, T. On the data path performance of leaf-spine datacenter fabrics. In Proceedings of the 2013 IEEE 21st annual symposium on high-performance interconnects. IEEE, 2013, pp. 71–74.
44. Al-Rubaye, S.; Kadhum, E.; Ni, Q.; Anpalagan, A. Industrial internet of things driven by SDN platform for smart grid resiliency. *IEEE Internet of Things Journal* **2017**, *6*, 267–277.
45. Josbert, N.N.; Ping, W.; Wei, M.; Li, Y. Industrial networks driven by SDN technology for dynamic fast resilience. *Information* **2021**, *12*, 420.
46. Zegura, E.W.; Calvert, K.L.; Donahoo, M.J. A quantitative comparison of graph-based models for Internet topology. *IEEE/ACM Transactions on networking* **1997**, *5*, 770–783.
47. Pastor-Satorras, R.; Vespignani, A. *Evolution and structure of the Internet: A statistical physics approach*; Cambridge University Press, 2007.
48. Khandaker, F.; Oteafy, S.; Hassanein, H.S.; Farahat, H. A functional taxonomy of caching schemes: Towards guided designs in information-centric networks. *Computer Networks* **2019**, *165*, 106937.
49. Yang, B.; Chai, W.K.; Xu, Z.; Katsaros, K.V.; Pavlou, G. Cost-efficient NFV-enabled mobile edge-cloud for low latency mobile applications. *IEEE Transactions on Network and Service Management* **2018**, *15*, 475–488.
50. Matos, F.; Matos, A.; Simoes, P.; Monteiro, E. Provisioning of Inter-Domain QoS-Aware Services. *Journal of Computer Science and Technology* **2015**, *30*, 404–420.
51. Fernandes, S. Methods and Techniques for Measurements in the Internet. In *Performance Evaluation for Network Services, Systems and Protocols*; Springer, 2017; pp. 45–73.
52. Chen, G.; Fan, Z.; Li, X. Modelling the complex Internet topology. In *Complex Dynamics in Communication Networks*; Springer, 2005; pp. 213–234.
53. Postel, J.; et al. Transmission control protocol **1981**.
54. Postel, J. RFC0768: User Datagram Protocol, 1980.
55. Yildirim, E.; Suslu, I.H.; Kosar, T. Which network measurement tool is right for you? a multidimensional comparison study. In Proceedings of the 2008 9th IEEE/ACM International Conference on Grid Computing. IEEE, 2008, pp. 266–275.
56. Balevi, E.; Gitlin, R.D. Unsupervised machine learning in 5G networks for low latency communications. In Proceedings of the 2017 IEEE 36th International Performance Computing and Communications Conference (IPCCC). IEEE, 2017, pp. 1–2.
57. Enea. TCP vs UDP—The Fight for Space on a Shared Network, 2024. Accessed: 2025-04-22.

 doi:10.20944/preprints202504.2530.v1

58. Saroiu, S.; Gummadi, K.P.; Dunn, R.J.; Gribble, S.D.; Levy, H.M. An analysis of internet content delivery systems. *ACM SIGOPS Operating Systems Review* **2002**, *36*, 315–327.
59. Networks, A. AppLogic Networks Unveils 2025 Global Internet Phenomena Report: Key Insights into Internet Traffic Trends, 2025. Accessed: 2025-04-22.
60. GUEANT, V. iPerf—The ultimate speed test tool for TCP, UDP and SCTPTest the limits of your network + Internet neutrality test.
61. VideoLAN. VLC media player for Ubuntu.
62. Rahman, M.; Yaakob, N.; Amir, A.; Ahmad, R.; Yoon, S.; Abd Halim, A. Performance analysis of congestion control mechanism in software defined network (SDN). In Proceedings of the MATEC Web of Conferences. EDP Sciences, 2017, Vol. 140, p. 01033.
63. Bifulco, R.; Boite, J.; Bouet, M.; Schneider, F. Improving sdn with inspired switches. In Proceedings of the Proceedings of the Symposium on SDN Research, 2016, pp. 1–12.
64. Aliyu, A.L.; Bull, P.; Abdallah, A. Performance implication and analysis of the OpenFlow SDN protocol. In Proceedings of the 2017 31st International Conference on Advanced Information Networking and Applications Workshops (WAINA). IEEE, 2017, pp. 391–396.
65. Antichi, G.; Castro, I.; Chiesa, M.; Fernandes, E.L.; Lapeyrade, R.; Kopp, D.; Han, J.H.; Bruyere, M.; Dietzel, C.; Gusat, M.; et al. Endeavour: A scalable sdn architecture for real-world ixps. *IEEE Journal on Selected Areas in Communications* **2017**, *35*, 2553–2562.
66. Giotsas, V.; Dietzel, C.; Smaragdakis, G.; Feldmann, A.; Berger, A.; Aben, E. Detecting peering infrastructure outages in the wild. In Proceedings of the Proceedings of the conference of the ACM special interest group on data communication, 2017, pp. 446–459.
67. Sharma, S.; Staessens, D.; Colle, D.; Pickavet, M.; Demeester, P. Enabling fast failure recovery in OpenFlow networks. In Proceedings of the 2011 8th International Workshop on the Design of Reliable Communication Networks (DRCN). IEEE, 2011, pp. 164–171.
68. Awobuluyi, O. Periodic control update overheads in OpenFlow-based enterprise networks. In Proceedings of the 2014 IEEE 28th International Conference on Advanced Information Networking and Applications. IEEE, 2014, pp. 390–396.
69. Groma, M.; Boros, T.; Helebrandt, P. Scalable Cache-Based Address Resolution Protocol Handling in Software-Defined Networks. In Proceedings of the 2019 XXVII International Conference on Information, Communication and Automation Technologies (ICAT). IEEE, 2019, pp. 1–6.
70. Lai, Y.C.; Ali, A.; Hossain, M.S.; Lin, Y.D. Performance modeling and analysis of TCP and UDP flows over software defined networks. *Journal of Network and Computer Applications* **2019**, *130*, 76–88.
71. Emma, D.; Loreto, S.; Pescapé, A.; Ventre, G. Measuring SCTP throughput and jitter over heterogeneous networks. In Proceedings of the 20th International Conference on Advanced Information Networking and Applications-Volume 1 (AINA'06). IEEE, 2006, Vol. 2, pp. 5–pp.
72. Fernando, O.A.; Xiao, H.; Spring, J. Developing a Testbed with P4 to Generate Datasets for the Analysis of 5G-MEC Security. In Proceedings of the 2022 IEEE Wireless Communications and Networking Conference (WCNC). IEEE, 2022, pp. 2256–2261.