Bulletin of Electrical Engineering and Informatics
Vol. 14, No. 3, June 2025, pp. 1779~1793
ISSN: 2302-9285, DOI: 10.11591/eei.v14i3.9063

Comparative performance analysis of software-defined networking vs conventional IP networks using IGP protocols

Santiago Nicolas Viuche, **Edith Paola Estupiñán Cuesta, Juan Carlos Martínez Quintero**
Telecommunications Engineering Program, Universidad Militar Nueva Granada, Bogota, Colombia

Article Info

Article history:

Received Jul 27, 2024
Revised Nov 23, 2024
Accepted Dec 25, 2024

Keywords:

iPerf
Jitter
OpenDayLight
Ping
Round-trip time
Routing
Software defined network

ABSTRACT

The exponential growth of users in data networks presents significant challenges in terms of availability and traffic management. The advent of software-defined networking (SDN) technology offers new opportunities for enhancing performance and reducing operational costs. This article compares traditional data networks using conventional routing protocols like OSPF with SDN networks. An evaluation scenario was designed to assess the performance of conventional data networks configured with OSPF against those implemented with SDN using OpenFlow. Performance tests were conducted with various packet sizes, evaluating round-trip time (RTT) and jitter metrics using GNS3 and Mininet software to simulate conventional and SDN networks, respectively. The results demonstrated superior performance in SDN, with shorter transmission times; RTT values reached a maximum of 0.18 ms for packets ranging from 32 to 512 bytes, and jitter values remained below 1 ms. Furthermore, a routing analysis highlighted the need for specifying path redundancy in SDN environments via simulation scripts, a limitation not observed in conventional networks. This emphasizes the importance of addressing this issue when deploying SDN in production environments.

Corresponding Author:

Edith Paola Estupiñán Cuesta
Telecommunications Engineering Program, Universidad Militar Nueva Granada
Bogotá, Colombia
Email: edith.estupinan@unimilitar.edu.co

1. INTRODUCTION

The exponential growth in the number of users in data networks has led to significant challenges, particularly in terms of availability, scalability, and traffic management. Traditional networks, which rely on conventional routing protocols such as OSPF, face difficulties in meeting these increasing demands, especially in large-scale infrastructures where configuration and maintenance become increasingly complex [1]. Legacy networks have played a fundamental role in the development of telecommunications infrastructure, but their management has proven tedious, requiring lengthy configurations and increasing use of dedicated devices [2]. As a result of these limitations, software-defined network (SDN) have emerged as an alternative offering greater flexibility and centralized control [3]-[7].

Previous studies have investigated various aspects of the transition to SDN. For instance [8], analyzed the impact of SDN controller deployment in legacy networks, proposing models to optimize their placement. The study discusses the minimum number of controllers required for an efficient transition, proposes an analytical model, and conducts experiments on the quantity, and optimal locations of the controllers. An optimization model is presented to address the controller placement problem during the transition, with three alternative policies for selecting nodes to host controllers. The impacts of these policies on controller load balancing during the transition are examined. The reliability of SDN was evaluated in comparison to traditional networks, revealing that while SDN offers greater flexibility, it does not necessarily improve operational reliability [9]. A model based on link failures is proposed, finding that although SDN networks offer advantages in other aspects, they do not significantly improve operational reliability compared to legacy networks. Approaches to enhance reliability, such as considering control convergence and rapid fault detection, are suggested, providing valuable insights for network operators considering migration to SDN. Routing optimization algorithms in hybrid SDN networks were examined, demonstrating improvements in link utilization [10]. The study proposes considering path cardinality constraints in routing optimization, formulating the problem as a mixed integer nonlinear programming (MINLP) model, an approximation algorithm called the H-permissible paths routing scheme (HPRS) is presented, which selects a specific number of permissible paths for flow routing. The results show that HPRS outperforms other routing algorithms in minimizing maximum link utilization (MLU) and reducing flow entries. In terms of performance metrics, in [11] compared various SDN controllers, concluding that OpenDayLight performed better in terms of round-trip time (RTT) and jitter. OpenDayLight showed better results in RTT in 7 of 8 changes, while RYU demonstrated better results in jitter in half of the variations and in Throughput in six of the eight variations. This study showed the initial results for the choose of the OpenDayLight controller in this research. Finally, the k-LB problem is addressed, and an algorithm is proposed to solve it, demonstrating its effectiveness in experiments with different network topologies [12]. Results show that the algorithm outperforms to others in terms of performance and stability, approaching the optimal solution in terms of link utilization.

The research preview evidence studies over choice of controllers and analysis of their operation, definition and optimization of algorithms to improve routing processes are evidenced but not provide evidence of a comparative analysis over performance metrics like RTT and jitter between conventional and SDN networks, nor do they assess the impact of packet size on the operation of these two types of networks. Additionally, the previous studies do not address redundancy analysis in the context of logical or physical link failures, highlighting an open and active area of research in the field of networking. In accordance with the above, this study first analyzes the performance in terms of latency (RTT) and variability (jitter) in both conventional and SDN networks with varying packet sizes. Second, it evaluates redundancy in SDN and conventional networks offering novel perspective on managing link failures in SDN and comparing the response of traditional networks. The rest of the article is organized as follows: section 2 principles SDN, section 3 details the experimental method and results, and section 4 presents conclusions along with suggestions for future research.

2. SOFTWARE-DEFINED NETWORK PRINCIPLES

The architecture of conventional networks exhibits a dependency on control and forwarding functions, provided by the integration into a single network node between the control plane (control layer) and the data plane (infrastructure layer), as depicted in Figure 1 [9]. Additionally, the management plane (application layer) is included, further constraining system centralization, impacting its flexibility and scalability [13]. The architecture of SDN is built upon the principle of centralization by decoupling the control plane from the management plane, driving intelligent global connections, low loss, and cloud abstraction [13]-[15]. This is achieved because the control plane is comprised of a component called the controller, which dictates how packets are managed in the data plane corresponding to the network components where packet forwarding occurs [2], as illustrated in Figure 2.

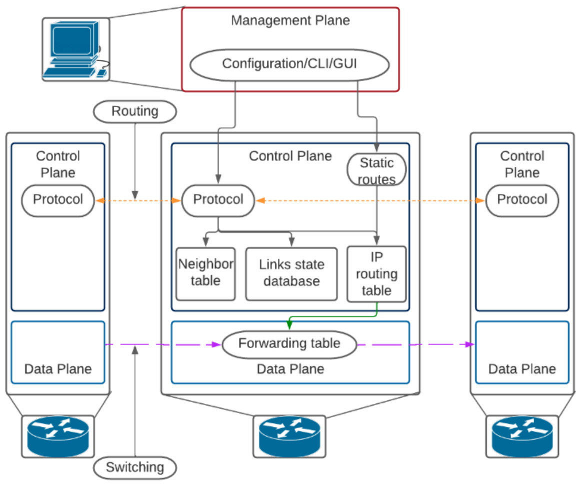


Figure 1. Conventional network architecture, based on [16]-[20]

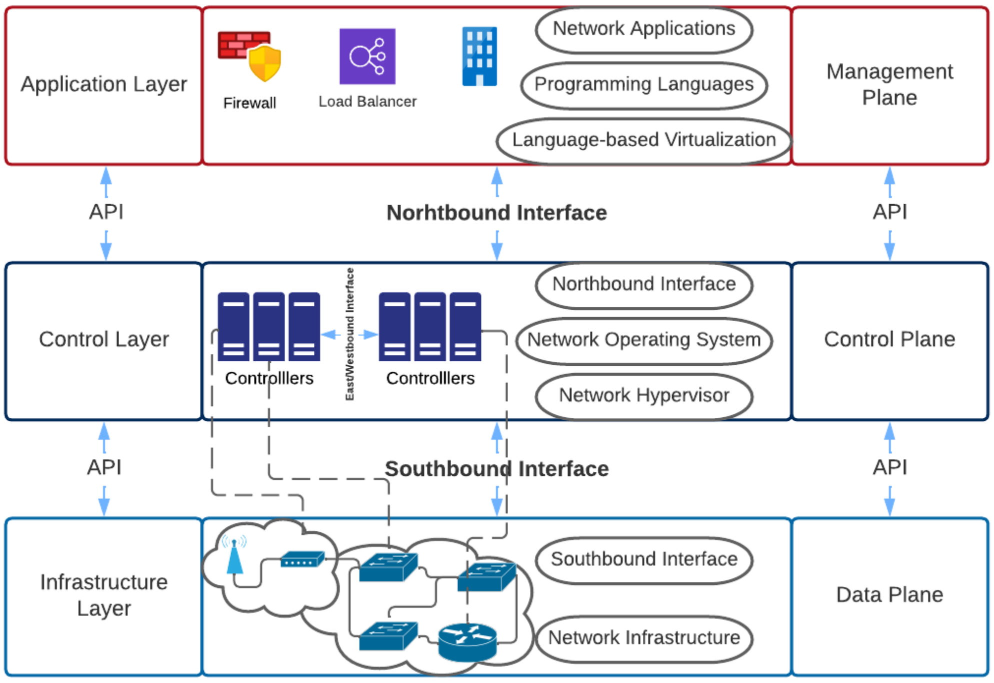


Figure 2. SDN network architecture, based on [13], [21], [22]

From the higher-level, management, or application plane, network instructions and management are provided, indicating corresponding routing rules [21], [23]. Communication between the planes occurs through 3 types of APIs (northbound, southbound, and east/westbound), the first two are relevant for this research. The northbound API facilitates communication between the management plane and the control plane, potentially involving third-party applications and enabling network management and rule-setting for the control plane [21], [23], [24]. A southbound API enables communication between the control plane (controller) and the data plane, providing routing rules [21], [24], the most popular API of this kind for SDN networks is OpenFlow, standardized by the open networking foundation [25], [26]. Figure 3 illustrates the most well-known and used APIs in the SDN environment [25], [27].

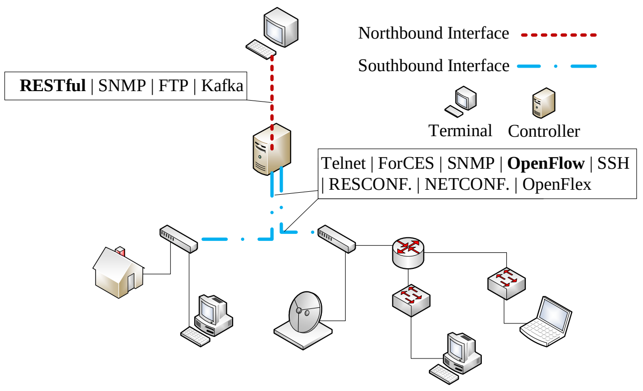

Northbound Interface
Southbound Interface
RESTful | SNMP | FTP | Kafka
Terminal
Controller
Telnet | ForCES | SNMP | OpenFlow | SSH | RESCONF. | NETCONF. | OpenFlex

Figure 3. SDN SBI and NBI API protocols

2.1. Routing metrics on conventional network

The protocols used by conventional IP networks rely on metrics that determine the shortest path to a remote network. Among the metrics used are hops for RIP; bandwidth, delay, MTU, and reliability for EIGRP; and cost for OSPF. RIP was developed by xerox network systems (XNS) and standardized in RFC 1058 [28]-[30]. EIGRP, like RIP, is a distance vector protocol developed by Cisco and standardized in RFC 7868 [31]. OSPF is a link-state protocol developed by the IETF and openly standardized in RFC 2328; the metric OSPF works with is the cost based on the bandwidth of its interfaces to reach the destination network (1), with an inversely proportional relationship between higher bandwidth and lower cost [32]. For the evaluation of performance at the time level, metrics such as RTT and Jitter stand out, among others. The first can be calculated using (2) with the ping tool and is described in RFC 6349, which refers to the total RTT [33]. For jitter, its calculation is based on the real-time transport protocol (RTP), as described in RFC 1889 [34], and corresponds to the statistical variation of packet arrival times; (3) is used to calculate of the difference (D) between the arrival times (R) and RTP timestamp (S) of packets i and j. For Jitter, it is continuously calculated upon receiving data packets (i) from the source, using the difference (D) and the previous packet ($i-1$), resulting in (4). Where J represents the jitter value, $D(i-1,i)$ is the difference between arrival times and RTP timestamp of the previous and current packets. The gain parameter 1/16 provides a good noise reduction ratio.

$$OSPF\ Cost = \frac{10^8}{Bandwidth} \tag{1}$$

$$RTT = \frac{Total\ RTTs\ during\ transfer}{Transfer\ duration\ in\ sec} ms \tag{2}$$

$$D(i,j) = (R_j - R_i) - (S_j - S_i) \tag{3}$$

$$J = J + \frac{|D(i-1,i)| - J}{16} \tag{4}$$

3. METHOD

This study defined four stages for the development of the research, as illustrated in Figure 4. The first stage involved the selection of software tools and versions used in the project, with the primary references for tool selection being [35]. The second stage focused on the design and implementation of the network environment, where use cases were applied to emulate both conventional and SDN networks; in SDN was used OpenDayLight controller. The third stage consisted of performance testing, generating ICMP, UDP, and TCP traffic through the simulated environments with the objective of measuring RTT and jitter under varying packet sizes. Finally, the fourth stage presents an analysis of redundancy links in both SDN and traditional networks.

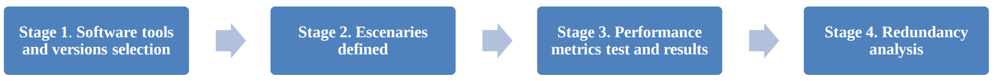

Stage 1. Software tools and versions selection
Stage 2. Escenaries defined
Stage 3. Performance metrics test and results
Stage 4. Redundancy analysis

Figure 4. Method proposed

3.1. Stage 1 software tools and versions selection

The software tools used for the experiment are presented in Table 1, and include Mininet for the SDN environment, GNS3 for the conventional network, and Wireshark for traffic analysis. The version of OpenFlow chosen was 1.3, as it is integrated with Wireshark and serves as the southbound API (SBI) for the SDN environment. The installation of Mininet was performed on Ubuntu 22.04 operating system by extracting the repositories through Github. After installing Mininet, the performed the integration with OpenDayLight version 0.3.0 Lithium, extracted from the official site.

Table 1. Software specifications used

Software	Version	Programming language	Programming language version
Mininet	2.3.1B4	Python	3.8.10
GNS3	2.2.43		3.10.11
OpenFlow	1.3	N/A	N/A
iPerf	iPerf3	C	N/A
OpenDayLight	0.3.0 Lithium	Java	8

3.2. Stage 2 escenaries defined

The experimental process began by configuring the topologies in GNS3 for conventional networks, as show Figure 5, and Mininet for SDN networks, as shown in Figure 6. Both networks were set up with a bandwidth of 155.52 Mbps over link. Wireshark was used to capture and analyze the test data, while OpenFlow 1.3 was chosen as the API for SDN controller interaction. The configuration SDN topology over mininet take reference [26].

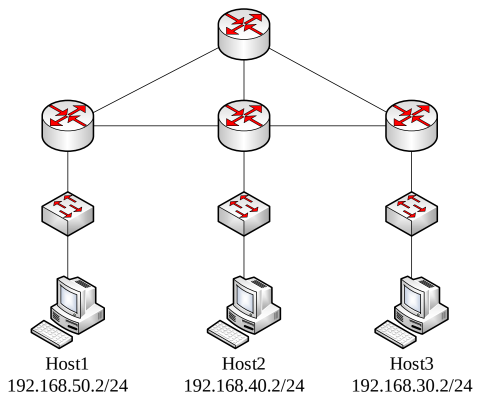


Figure 5. Proposed topology

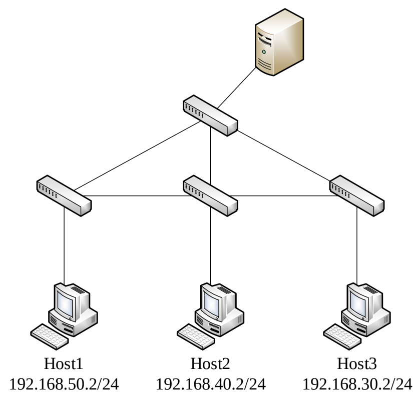


Figure 6. Proposed SDN topology

3.3. Performance metric test and results

Performance analysis was conducted in each network environment by executing 10 tests for each of the 10 packet size variations respectively (32 bytes, 64 bytes, 128 bytes, 256 bytes, 512 bytes, 1024 bytes, 1500 bytes, 2048 bytes, 4096 bytes, and 8192 bytes). 500 samples were extracted per test, resulting in a total of 5000 samples per link test.

The RTT metric was evaluated by sending ICMP traffic using the ping tool, with tests performed between Host 1 as the sender and Host 2 and Host 3 as receivers. Additional tests were carried out with Host 2 as the sender and Host 3 as the receiver. A total of 300 RTT tests were performed, yielding 150,000 samples. For jitter, UDP traffic was sent using the iPerf3 software, with each network component set up as either a server or client to ensure bidirectional traffic flow. Table 2 presents the setup for RTT and jitter testing.

Table 2. Proposed scenarios to RTT and jitter evaluation

Link	RTT/tool	Jitter/software
Host 1-Host 2	Ping	IPerf, software used for this metric
Host 1-Host 3	Ping	
Host 2-Host 1	N/A	
Host 2-Host 3	Ping	
Host 3-Host 1	N/A	
Host 3-Host 2	N/A	

For jitter testing, 6 packet transmission scenarios were defined to ensure a comprehensive evaluation, and 600 tests (100 per scenario) were conducted, resulting in 300,000 total samples for this metric. Upon completion of data collection for both metrics, a total of 900 tests were performed, yielding 450,000 samples. Table 3 summarizes the testing and sampling process for each metric. Throughout the testing process, collected data for both RTT and jitter were analyzed to determine the network's performance under different packet size variations. Inconsistent results due to high CPU load in the GNS3 environment led to the exclusion of some samples. These anomalies were primarily observed during higher packet size tests (4096 bytes and 8192 bytes), and as a result, they were discarded to maintain the reliability of the analysis.

Table 3. Testing and sampling for RTT and jitter

Metric	Traffic	Number of test	Number of samples
RTT	ICMP	300	150000
Jitter	UDP	600	300000

3.3.1. Round-trip time performance test results

The results of the RTT tests, as summarized in Table 4, show that as the packet size increases, the RTT times also increase, particularly in conventional networks. In SDN, this increase is minimal, demonstrating its superior performance in terms of latency. This behavior aligns with previous findings [8], [10], [11], which emphasized that SDN’s centralized architecture allows for more efficient routing, reducing the time required for packet processing. In contrast, conventional networks using OSPF require more time to calculate the shortest path due to their distributed nature, which results in longer RTT times as packet sizes grow.

Table 4. RTT measurement results-average

Test (bytes)	RTT average (ms)					
	H1-H2		H1-H3		H2-H3	
	Conv.	SDN	Conv.	SDN	Conv.	SDN
32	39.53	0.21	57.50	0.18	40.17	0.18
64	38.42	0.22	58.01	0.18	39.09	0.18
128	39.19	0.22	57.16	0.18	39.70	0.20
256	39.55	0.22	59.10	0.18	39.45	0.21
512	39.45	0.22	59.17	0.18	39.68	0.21
1024	40.13	0.22	59.00	0.19	41.78	0.21
1500	60.69	0.29	80.66	0.24	60.80	0.29
2048	60.76	0.32	81.55	0.24	62.40	0.28
4096	81.45	0.49	102.24	0.44	82.11	0.49
8192	143.69	1.24	166.02	1.19	145.65	1.23

It can be observed that in both types of networks, there was an increase in RTT time as the packet size increased. For the conventional network environment, the results are shown in Figure 7. From the results obtained in the conventional network, as shown in Figure 7 and Table 5, an increase in RTT was observed for each packet size variation when testing from H1 to H3. Although the proportional increase in RTT with increasing packet size was observed, the best time was 38.42 ms in the 64 bytes test from Host 1 to Host 2, while the highest time was for the 8192 bytes test from H1 to H3, with 166.02 ms. The final RTT results in the SDN environment are depicted in Figure 8, showing a notable difference between the highest SDN results, with measurements below 1.50 ms compared to measurements above 150 ms in the conventional network.

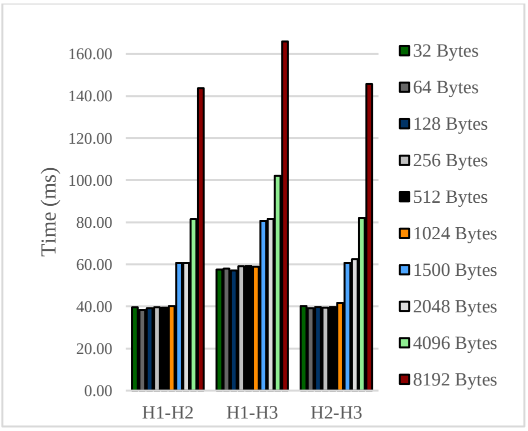


Figure 7. Average RTT in conventional network

Table 5. Jitter test average 1

Test (bytes)	Jitter average (ms)					
	H1-H2		H1-H3		H2-H3	
	Conv.	SDN	Conv.	SDN	Conv.	SDN
32	20.36	0.004	20.31	0.005	15.23	0.004
64	23.36	0.004	18.93	0.004	47.50	0.003
128	23.26	0.005	18.04	0.007	23.84	0.006
256	20.84	0.010	34.83	0.012	11.87	0.005
512	29.90	0.006	29.25	0.010	12.79	0.014
1024	58.63	0.015	66.16	0.012	41.43	0.016
1500	51.34	0.030	74.57	0.018	51.41	0.024
2048	72.84	0.027	80.16	0.017	64.53	0.023
4096	204.97	0.022	139.97	0.015	147.17	0.019
8192	378.47	0.015	354.61	0.013	306.24	0.012

The low RTT values in SDN (ranging from 0.18 ms to 1.24 ms across different tests) highlight the network's ability to handle real-time traffic more efficiently compared to conventional networks, which exhibited RTT values above 100 ms in larger packet sizes. This difference is particularly relevant for latency-sensitive applications such as real-time video streaming or VoIP, where consistent low-latency performance is crucial. However, it is important to note that SDN's reliance on centralized control can also be a limitation in larger, more complex networks, where the controller may become a bottleneck if not properly optimized.

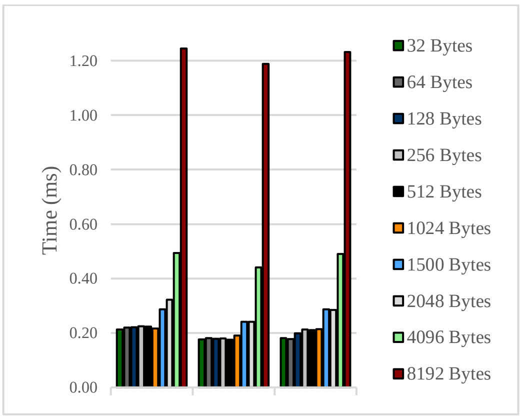


Figure 8. Average RTT in SDN

3.3.2. Jiter performance test results

The jitter results, presented in Tables 5 and 6, indicate a much more stable performance in SDN networks compared to conventional networks. As observed, jitter values in SDN remained below 10 ms in all tests, even with larger packet sizes, whereas conventional networks saw jitter values exceeding 400 ms with 8192-byte packets. This confirms that SDN provides a more reliable experience in environments where traffic stability is critical, such as in multimedia communications or online gaming, where high jitter can cause noticeable disruptions. From the collected data, it can be observed, with the assistance of Figure 9, that the jitter behavior in the conventional network environment is similar to that observed in the RTT test. It is evident that the jitter time increased directly with the packet size worked on.

Table 6. Jitter test average 2

Test (bytes)	Jitter average (ms)					
	H1-H2		H1-H3		H2-H3	
	Conv.	SDN	Conv.	SDN	Conv.	SDN
32	25.76	0.004	14.11	0.004	50.79	0.018
64	14.65	0.003	21.29	0.004	18.14	0.004
128	16.41	0.005	18.20	0.004	38.89	0.012
256	25.14	0.005	37.10	0.006	46.63	0.018
512	44.18	0.009	42.07	0.012	34.23	0.010
1024	49.07	0.011	43.39	0.012	44.60	0.011
1500	45.50	0.017	81.14	0.017	32.67	0.012
2048	109.40	0.015	138.14	0.014	100.81	0.012
4096	170.48	0.014	136.47	0.015	189.53	0.019
8192	468.04	0.012	429.90	0.015	438.13	0.018

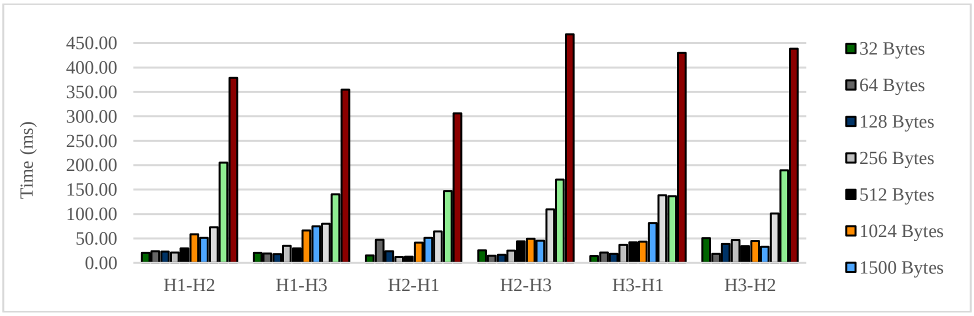


Figure 9. Average jitter in conventional network

The highest jitter values were obtained for packet size tests of 8192 bytes, with the highest result being in the H2-H3 test with 468.04 ms. The lowest results in this environment were 18.04 ms and 18.93 ms for packet sizes of 128 bytes and 64 bytes for H1-H3, respectively; 15.23 ms, 11.87 ms, and 12.79 ms for packet sizes of 32 bytes, 256 bytes, and 512 bytes in H2-H1; 14.65 ms and 16.41 ms in the tests of 64 bytes and 128 bytes for H2-H3; 14.11 ms and 18.20 ms for tests of 32 bytes and 128 bytes in H3-H1. The best result in the

H3-H2 test was with 64 bytes, with an average of 18.14 ms. The lowest jitter values were observed in tests between H1 and H3, while the highest value was generated in the H3-H2 test. The final averages of the jitter tests for the SDN network are shown in Figure 10, where no averages exceeded 10 ms, and in 5 out of the 6 tests conducted between hosts, there were spikes in times when working with 1500 bytes, presenting 0.030 ms in H1-H2, 0.018 ms in H1-H3, 0.024 ms in H2-H1, 0.017 ms in H2-H3, and H3-H1. In the H3-H2 test, the peak time was 0.019 with 4096 bytes.

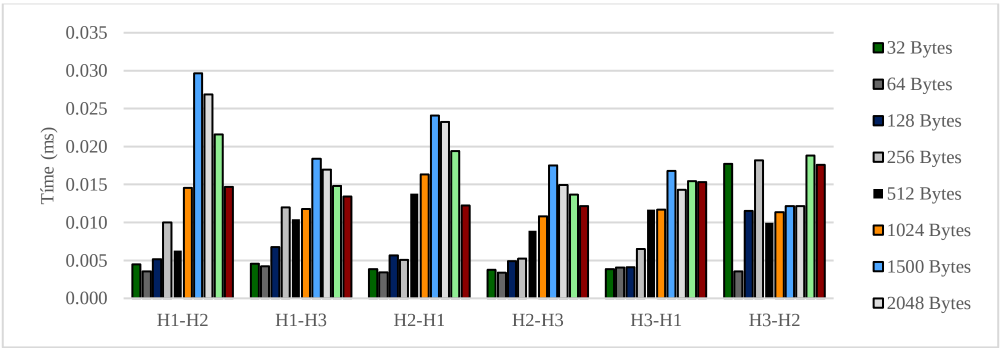


Figure 10. Average jitter in SDN

In general, the best time was observed when working with 64 bytes for all tests conducted, with values of 0.004 ms and 0.003 ms. In the network tests between H1-H2, H1-H3, H2-H1, and H2-H3, there was a trend of increasing jitter average between 32 bytes and 1500 bytes. In the remaining 3 packet size variations, a decrease in jitter time was observed, which did not occur in the conventional environment tests. As mentioned earlier, for the transmission of 64-bytes packets, this variable yielded the best results in the conventional network, a characteristic that was observed in each of the SDN environment tests. Except for the H3-H2 tests, the 128-bytes packet was the second-best performing variable overall for the SDN environment. This aspect, in the conventional network, resulted in the second-best outcome in the H2-H1 and H3-H1 tests, with the lowest values in H1-H3 and H2-H3 tests.

These findings support the idea that the programmability of SDN allows for better traffic management and queue handling, reducing the variability in packet arrival times, which is a common issue in conventional networks relying on traditional routing protocols like OSPF. For applications requiring low jitter, SDN is clearly the best option.

3.4. Redundancy analysis over software-defined network routing versus conventional networks

3.4.1. Link redundancy analysis in software-defined networks

Once the connection between all network hosts was validated using ping, the highlighted interface in Figure 11 was shutdown from the OpenDayLight platform environment. The link failure is triggered using the command "link s3 s2 down" via the Mininet terminal. Figure 12 confirms this process by indicating the interface that has been shut down and the corresponding command.

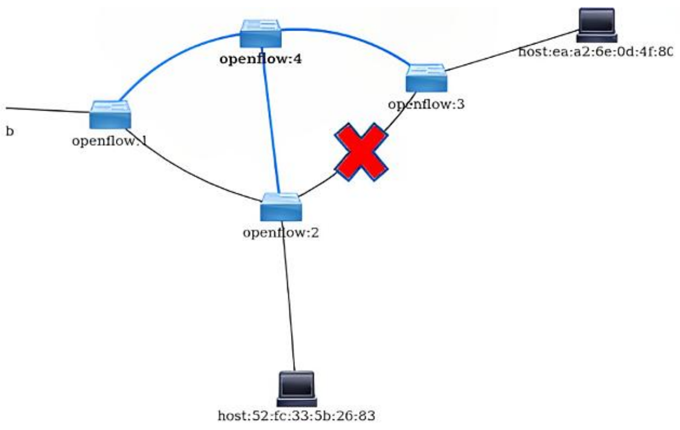
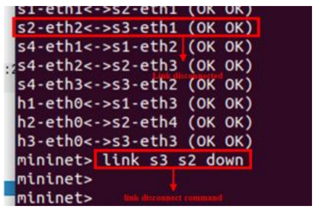
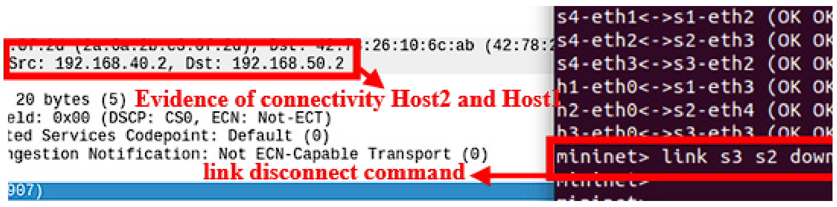


Figure 11. Remove first link between Host 3-Host 2

Following the interface shutdown, the ongoing transmission of ICMP messages from Host 3 to Host 2 is evident in Figure 13. The path redundancy process obtained is illustrated in Figure 14, corresponding to the second path configured in the SDN network topology script.



```
s1-eth1<->s2-eth1 (OK OK)
s2-eth2<->s3-eth1 (OK OK)
s4-eth1<->s1-eth2 (OK OK)
s4-eth2<->s2-eth3 (OK OK)
s4-eth3<->s3-eth2 (OK OK)
h1-eth0<->s1-eth3 (OK OK)
h2-eth0<->s2-eth4 (OK OK)
h3-eth0<->s3-eth3 (OK OK)
mininet> link s3 s2 down
mininet>
mininet>
```



```
Src: 192.168.40.2, Dst: 192.168.50.2          :26:10:6c:ab (42:78:2
 20 bytes (5)
eld: 0x00 (DSCP: CS0, ECN: Not-ECT)
ted Services Codepoint: Default (0)
ngestion Notification: Not ECN-Capable Transport (0)
907)
```

```
s4-eth1<->s1-eth2 (OK OK
s4-eth2<->s2-eth3 (OK OK
s4-eth3<->s3-eth2 (OK OK
h1-eth0<->s1-eth3 (OK OK
h2-eth0<->s2-eth4 (OK OK
h3-eth0<->s3-eth3 (OK OK
mininet> link s3 s2 dow
mininet>
```

Figure 12. Test interface shutdown

Figure 13. ICMP traffic between test devices

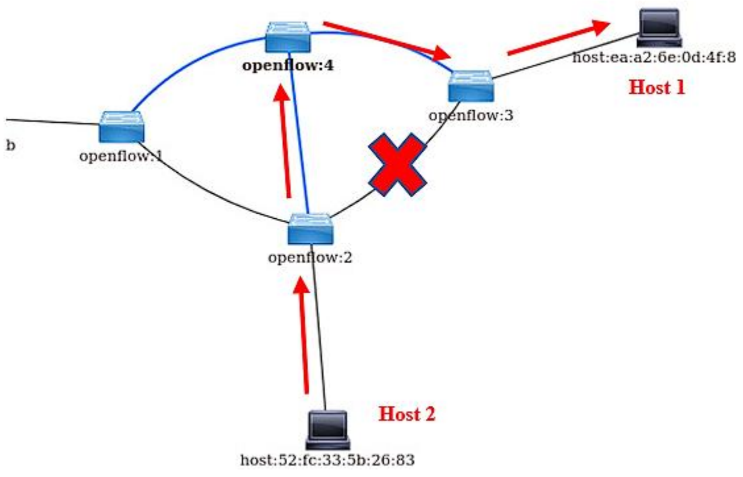


Figure 14. New route after disconnection between switch3-switch2

As a final validation, the interface between SwitchOF4 and SwitchOF2 was shutdown, as shown in Figure 15, resulting in Host 2 being unreachable by Host 3. This demonstrates the necessity of configuring redundancy in the network through the topology script in order to add the new route Host 2 → SwOF3 → SwOF4 → SwOF1 → SwOF2 → Host 1. From the controller, the disconnection event is analyzed using Wireshark, revealing the lack of communication between Host 1 and Host 2, as shown in Figure 16.

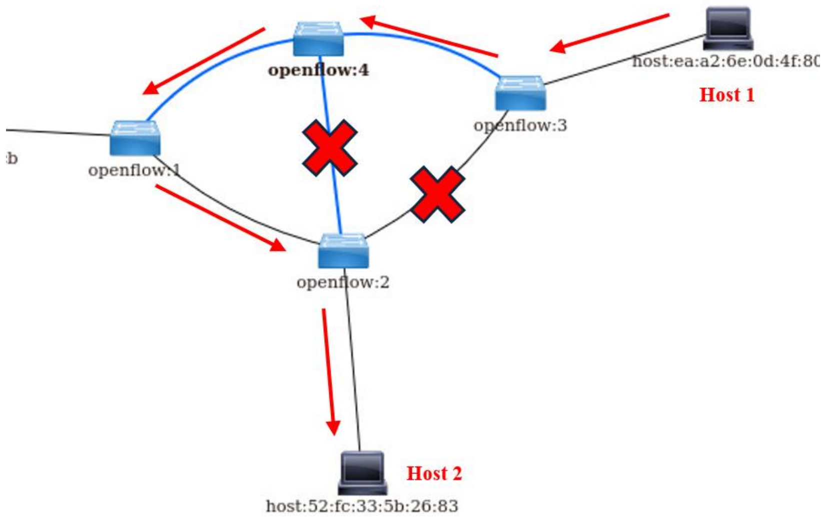
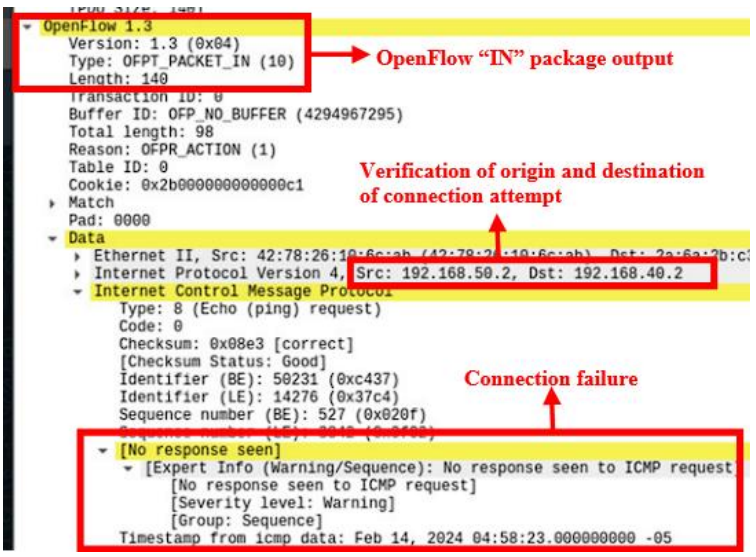


Figure 15. Elimination of second link between Host 1 and Host 2

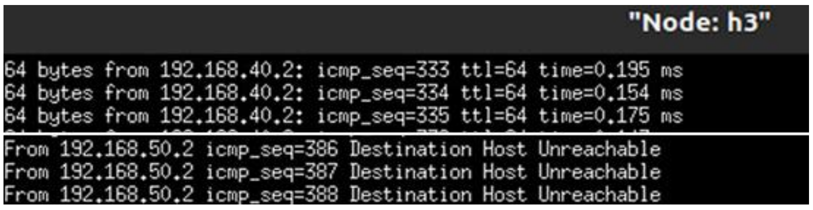


```
▾ OpenFlow 1.3
      Version: 1.3 (0x04)
      Type: OFPT_PACKET_IN (10)
      Length: 140
      Transaction ID: 0
      Buffer ID: OFP_NO_BUFFER (4294967295)
      Total length: 98
      Reason: OFPR_ACTION (1)
      Table ID: 0
      Cookie: 0x2b000000000000c1
    ▸ Match
      Pad: 0000
    ▾ Data
      ▸ Ethernet II, Src: 42:78:26:10:6c:ab (42:78:26:10:6c:ab), Dst: 2a:6a:2b:c3
      ▸ Internet Protocol Version 4, Src: 192.168.50.2, Dst: 192.168.40.2
      ▾ Internet Control Message Protocol
          Type: 8 (Echo (ping) request)
          Code: 0
          Checksum: 0x08e3 [correct]
          [Checksum Status: Good]
          Identifier (BE): 50231 (0xc437)
          Identifier (LE): 14276 (0x37c4)
          Sequence number (BE): 527 (0x020f)
        ▾ [No response seen]
          ▾ [Expert Info (Warning/Sequence): No response seen to ICMP request]
              [No response seen to ICMP request]
              [Severity level: Warning]
              [Group: Sequence]
          Timestamp from icmp data: Feb 14, 2024 04:58:23.000000000 -05
```

OpenFlow "IN" package output
Verification of origin and destination of connection attempt
Connection failure

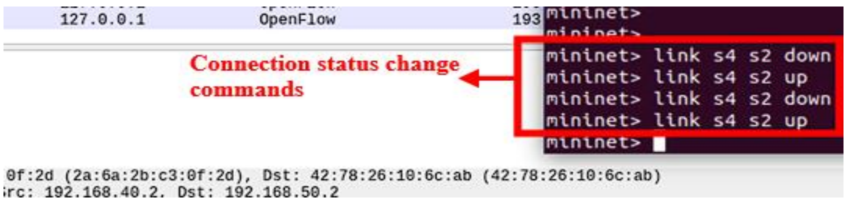
Figure 16. Host disconnection detection from controller

The results from the SDN routing analysis indicate that although centralized control offers flexibility, specific scripts must be implemented to ensure redundancy in case of failures. This is one of the main differences from conventional networks, where protocols like OSPF dynamically manage route recovery. The lack of automation in SDN can be a limitation unless routing configurations are optimized. Figure 17 indicates the inability to connect between Host 1 and Host 2, while Figure 18 shows the command to turn off and on the interface. In this last disconnection event, no redundancy was generated in the network because the path illustrated in Figure 15 is not specified in the topology script.



```
"Node: h3"
64 bytes from 192.168.40.2: icmp_seq=333 ttl=64 time=0.195 ms
64 bytes from 192.168.40.2: icmp_seq=334 ttl=64 time=0.154 ms
64 bytes from 192.168.40.2: icmp_seq=335 ttl=64 time=0.175 ms
From 192.168.50.2 icmp_seq=386 Destination Host Unreachable
From 192.168.50.2 icmp_seq=387 Destination Host Unreachable
From 192.168.50.2 icmp_seq=388 Destination Host Unreachable
```

Figure 17. Connection drop between Host 3 and Host 2



```
127.0.0.1            OpenFlow            193
mininet>
mininet>
mininet> link s4 s2 down
mininet> link s4 s2 up
mininet> link s4 s2 down
mininet> link s4 s2 up
mininet>
0f:2d (2a:6a:2b:c3:0f:2d), Dst: 42:78:26:10:6c:ab (42:78:26:10:6c:ab)
rc: 192.168.40.2. Dst: 192.168.50.2
```

Figure 18. Switching test interfaces shut on and shut off

3.4.2. Link redundancy analysis in conventional networks

After conducting the redundancy analysis at the link level in SDN networks, the same procedure was applied to conventional networks. Figure 19 depicts the scenario implemented for the conventional network in GNS3 using OSPF with IPv4 addressing scheme. Once the configuration using the OSPFv2 protocol is completed, the routing tables of the 4 routers involved are verified, as shown in Figures 20 to 23.

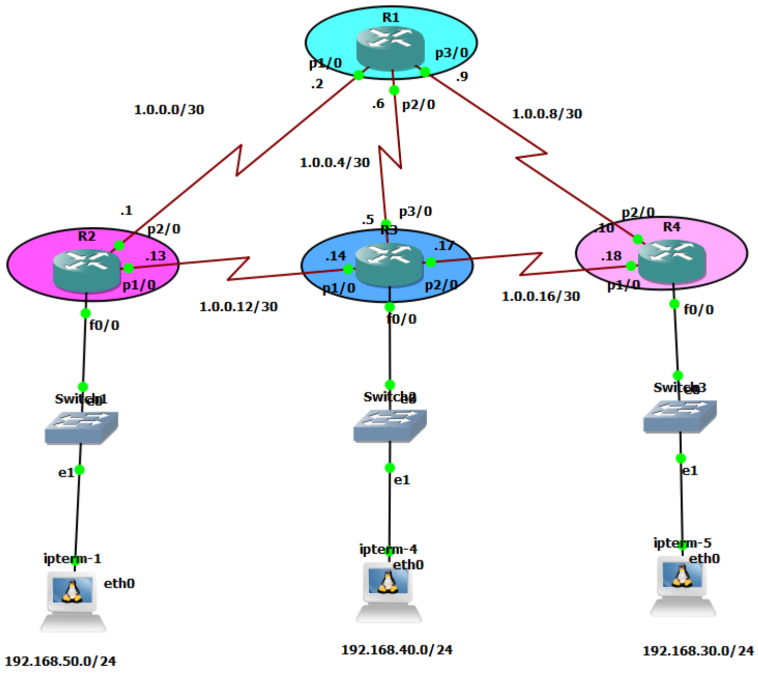
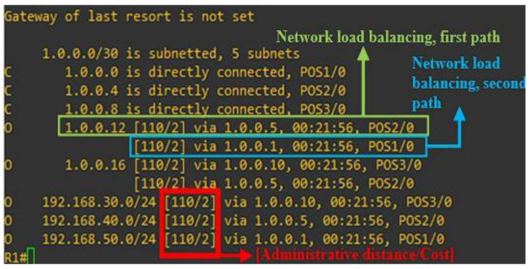


Figure 19. Conventional network using GNS3

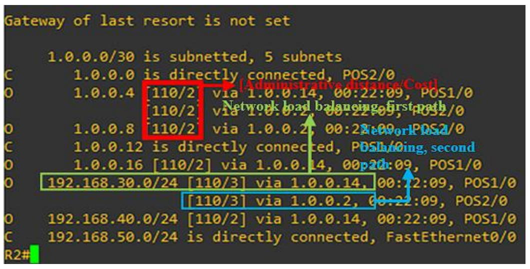


Network load balancing, first path
Network load balancing, second path
[Administrative distance/Cost]

```
Gateway of last resort is not set

     1.0.0.0/30 is subnetted, 5 subnets
C       1.0.0.0 is directly connected, POS1/0
C       1.0.0.4 is directly connected, POS2/0
C       1.0.0.8 is directly connected, POS3/0
O       1.0.0.12 [110/2] via 1.0.0.5, 00:21:56, POS2/0
                 [110/2] via 1.0.0.1, 00:21:56, POS1/0
O       1.0.0.16 [110/2] via 1.0.0.10, 00:21:56, POS3/0
                 [110/2] via 1.0.0.5, 00:21:56, POS2/0
O    192.168.30.0/24 [110/2] via 1.0.0.10, 00:21:56, POS3/0
O    192.168.40.0/24 [110/2] via 1.0.0.5, 00:21:56, POS2/0
O    192.168.50.0/24 [110/2] via 1.0.0.1, 00:21:56, POS1/0
R1#
```

Figure 20. Routing table R1

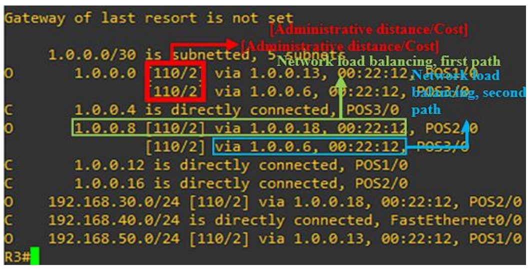


[Administrative distance/Cost]
Network load balancing, first path
Network load balancing, second path

```
Gateway of last resort is not set

     1.0.0.0/30 is subnetted, 5 subnets
C       1.0.0.0 is directly connected, POS2/0
O       1.0.0.4 [110/2] via 1.0.0.14, 00:22:09, POS1/0
                [110/2] via 1.0.0.2, 00:22:09, POS2/0
O       1.0.0.8 [110/2] via 1.0.0.2, 00:22:09, POS2/0
C       1.0.0.12 is directly connected, POS1/0
O       1.0.0.16 [110/2] via 1.0.0.14, 00:22:09, POS1/0
O    192.168.30.0/24 [110/3] via 1.0.0.14, 00:22:09, POS1/0
                     [110/3] via 1.0.0.2, 00:22:09, POS2/0
O    192.168.40.0/24 [110/2] via 1.0.0.14, 00:22:09, POS1/0
C    192.168.50.0/24 is directly connected, FastEthernet0/0
R2#
```

Figure 21. Routing table R2

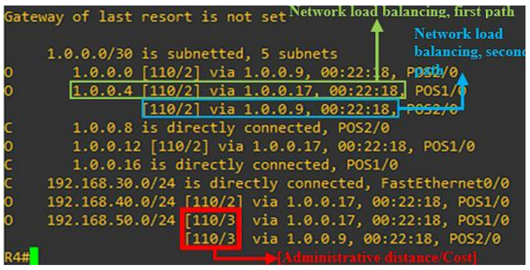


[Administrative distance/Cost]
[Administrative distance/Cost]
Network load balancing, first path
Network load balancing, second path

```
Gateway of last resort is not set

     1.0.0.0/30 is subnetted, 5 subnets
O       1.0.0.0 [110/2] via 1.0.0.13, 00:22:12, POS1/0
                [110/2] via 1.0.0.6, 00:22:12, POS3/0
C       1.0.0.4 is directly connected, POS3/0
O       1.0.0.8 [110/2] via 1.0.0.18, 00:22:12, POS2/0
                [110/2] via 1.0.0.6, 00:22:12, POS3/0
C       1.0.0.12 is directly connected, POS1/0
C       1.0.0.16 is directly connected, POS2/0
O    192.168.30.0/24 [110/2] via 1.0.0.18, 00:22:12, POS2/0
C    192.168.40.0/24 is directly connected, FastEthernet0/0
O    192.168.50.0/24 [110/2] via 1.0.0.13, 00:22:12, POS1/0
R3#
```

Figure 22. Routing table R3



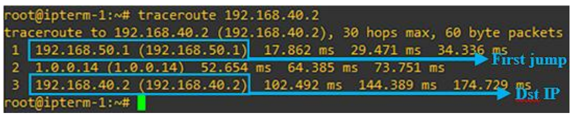
Network load balancing, first path
Network load balancing, second path
[Administrative distance/Cost]

```
Gateway of last resort is not set

     1.0.0.0/30 is subnetted, 5 subnets
O       1.0.0.0 [110/2] via 1.0.0.9, 00:22:18, POS2/0
O       1.0.0.4 [110/2] via 1.0.0.17, 00:22:18, POS1/0
                [110/2] via 1.0.0.9, 00:22:18, POS2/0
C       1.0.0.8 is directly connected, POS2/0
O       1.0.0.12 [110/2] via 1.0.0.17, 00:22:18, POS1/0
C       1.0.0.16 is directly connected, POS1/0
C    192.168.30.0/24 is directly connected, FastEthernet0/0
O    192.168.40.0/24 [110/2] via 1.0.0.17, 00:22:18, POS1/0
O    192.168.50.0/24 [110/3] via 1.0.0.17, 00:22:18, POS1/0
                     [110/3] via 1.0.0.9, 00:22:18, POS2/0
R4#
```

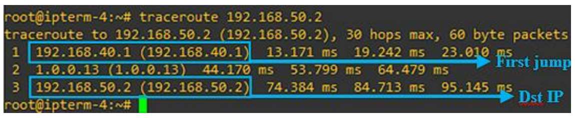
Figure 23. Routing table R4

The routing test conducted in the conventional network environment involved sending ICMP packets from Host 1 to Host 2, validating the preliminary paths chosen by OSPF, as indicated in Figures 24 and 25 after executing the "traceroute" command. In order to validate the redundancy of the OSPF protocol, the same failure was simulated in the previously simulated network in SDN, by shutting down the interface connecting Router 2 with Router 3 (P1/0). The update of the defined route from Host 1 to Host 2 is evidenced in Figure 26 compared to the first route corresponding to Host 1→ Router 2 → Router 1 (1.0.0.2) → Router 3 (1.0.0.5) → Host 2 (192.168.40.2), as indicated in Figure 27. The new route from Host 1 to Host 2 is displayed from the console of the first component in Figure 28 and is represented in Figure 29.



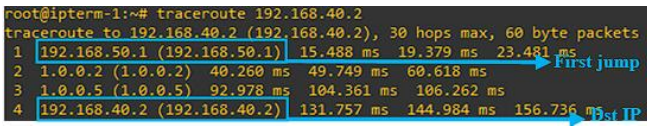
```
root@ipterm-1:~# traceroute 192.168.40.2
traceroute to 192.168.40.2 (192.168.40.2), 30 hops max, 60 byte packets
 1  192.168.50.1 (192.168.50.1)  17.862 ms  29.471 ms  34.336 ms
 2  1.0.0.14 (1.0.0.14)  52.654 ms  64.385 ms  73.751 ms
 3  192.168.40.2 (192.168.40.2)  102.492 ms  144.389 ms  174.729 ms
root@ipterm-1:~#
```

Figure 24. Traceroute H1-H2



```
root@ipterm-4:~# traceroute 192.168.50.2
traceroute to 192.168.50.2 (192.168.50.2), 30 hops max, 60 byte packets
 1  192.168.40.1 (192.168.40.1)  13.171 ms  19.242 ms  23.010 ms
 2  1.0.0.13 (1.0.0.13)  44.170 ms  53.799 ms  64.479 ms
 3  192.168.50.2 (192.168.50.2)  74.384 ms  84.713 ms  95.145 ms
root@ipterm-4:~#
```

Figure 25. Traceroute H2-H1



```
root@ipterm-1:~# traceroute 192.168.40.2
traceroute to 192.168.40.2 (192.168.40.2), 30 hops max, 60 byte packets
 1  192.168.50.1 (192.168.50.1)  15.488 ms  19.379 ms  23.481 ms
 2  1.0.0.2 (1.0.0.2)  40.260 ms  49.749 ms  60.618 ms
 3  1.0.0.5 (1.0.0.5)  92.978 ms  104.361 ms  106.262 ms
 4  192.168.40.2 (192.168.40.2)  131.757 ms  144.984 ms  156.736 ms
```

Figure 26. Traceroute H1-H2

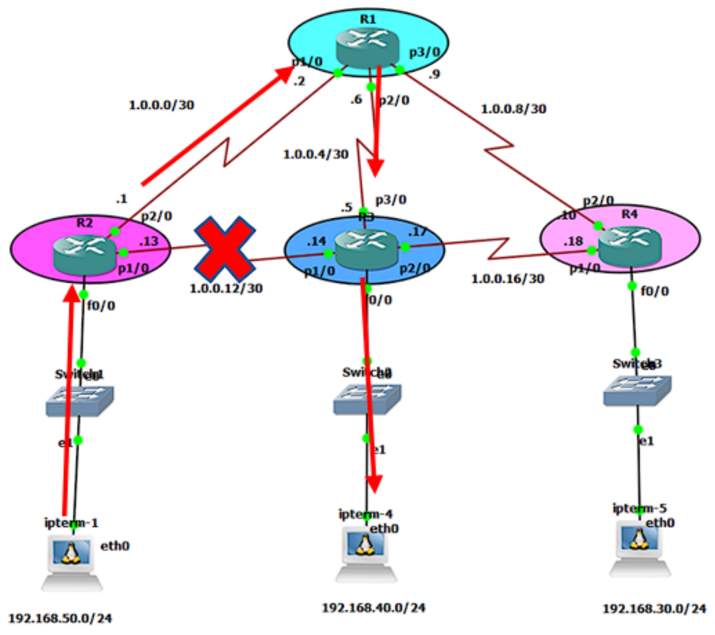
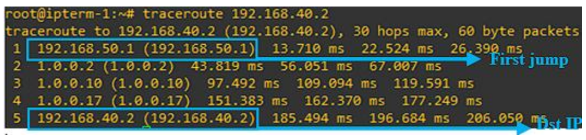


Figure 27. New best route between H1 and H2



```
root@ipterm-1:~# traceroute 192.168.40.2
traceroute to 192.168.40.2 (192.168.40.2), 30 hops max, 60 byte packets
 1  192.168.50.1 (192.168.50.1)  13.710 ms  22.524 ms  26.390 ms
 2  1.0.0.2 (1.0.0.2)  43.819 ms  56.051 ms  67.007 ms
 3  1.0.0.10 (1.0.0.10)  97.492 ms  109.094 ms  119.591 ms
 4  1.0.0.17 (1.0.0.17)  151.383 ms  162.370 ms  177.249 ms
 5  192.168.40.2 (192.168.40.2)  185.494 ms  196.684 ms  206.050 ms
```

Figure 28. New traceroute H1-H2

The cost for the new route is 4, after adding the link corresponding to interface P1/0 of Router 4. After conducting the routing tests in the network environments, using OSPF and OpenFlow protocols, it was evident that a conventional network updates the shortest possible routes when a link failure occurs, with the help of dynamically emitted protocol messages. In the case of SDNs, at the software level, it was evident that it is necessary to establish possible routes between end devices in the own script for simulation to establish redundancy in the network. The SDN routing analysis demonstrated that while SDN offers significant flexibility through centralized control, there are limitations when it comes to automatic redundancy. In our tests, redundancy had to be manually configured via scripts, whereas conventional networks using OSPF were able to dynamically manage route recovery when a link failure occurred. This highlights a key area where SDN currently falls short: its ability to autonomously manage failures and reroute traffic without human intervention. Future work should focus on developing more advanced algorithms for automatic failover and route recovery in SDN environments. While OpenFlow provides a flexible foundation, its current implementation still requires significant manual intervention to configure redundancy, limiting its effectiveness in dynamic or large-scale production environments. Research into controller-based algorithms that can dynamically adjust routes and provide automatic failover would address this gap and improve SDN’s overall reliability in real-world applications.

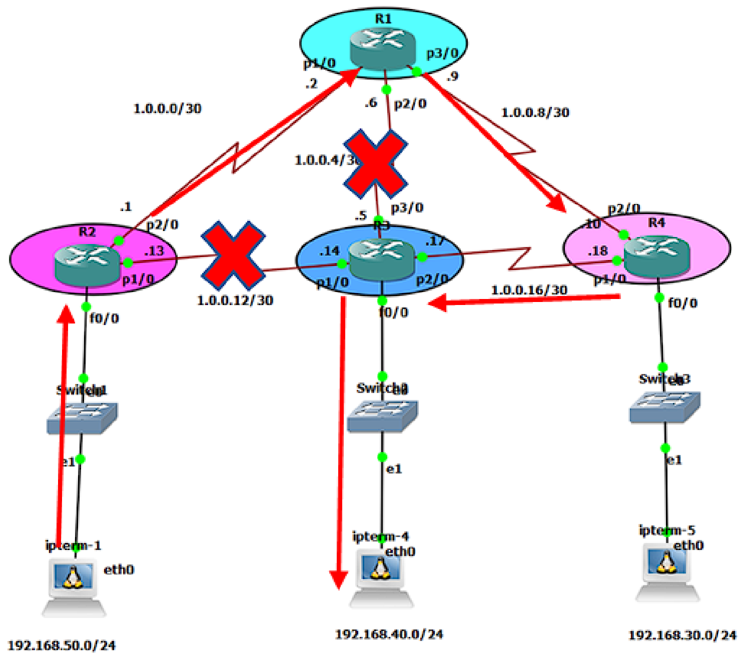


Figure 29. New best route between H1 and H2

4. CONCLUSION

This study provides a comprehensive analysis of the performance differences between conventional IP networks and SDN, focusing on RTT and jitter metrics. The findings demonstrate that SDN offers superior performance, particularly with smaller packet sizes, due to its centralized control and programmability. This makes SDN well-suited for environments requiring low latency and high reliability, such as real-time communications and cloud-based services.

In addition to confirming SDN's advantages in these key metrics, the study highlights the importance of redundancy configurations. SDN improves network responsiveness and enhances recovery during link failures, making it a robust solution for architectures requiring fault tolerance. However, the limitations of this study must be acknowledged, as the simulations were conducted in controlled, emulated environments. Future work should focus on testing SDN in real-world scenarios to assess scalability and stability.

Several avenues for future research have emerged from this study. Comparative evaluations of SDN controllers like ONOS, Ryu, and Floodlight under different traffic loads and topologies would provide insights into how controller architecture affects performance. Additionally, future work should explore more scalable and automated redundancy and failover solutions to improve dynamic route management and fault tolerance in SDN networks. Finally, the centralized nature of SDN offers potential for enhancing network security through real-time threat detection and mitigation, a promising area for further investigation. Other performance metrics, such as throughput, packet loss, and energy efficiency, should also be examined to gain a more comprehensive understanding of SDN's impact on network performance.

FUNDING INFORMATION

This work was developed within the GISSIC research group, as a product derived from the Maxwell research seedbed, endorsed by the vice-rectorate for research at the Universidad Militar Nueva Granada.

AUTHOR CONTRIBUTIONS STATEMENT

This journal uses the Contributor Roles Taxonomy (CRediT) to recognize individual author contributions, reduce authorship disputes, and facilitate collaboration.

Name of Author	C	M	So	Va	Fo	I	R	D	O	E	Vi	Su	P	Fu
Santiago Nicolas Viuche	✓		✓	✓	✓	✓	✓		✓		✓			✓
Edith Paola Estupiñan Cuesta		✓		✓	✓	✓		✓	✓	✓		✓	✓	✓
Juan Carlos Martinez Quintero		✓		✓	✓	✓		✓		✓		✓	✓	✓

C	: **C**onceptualization	I	: **I**nvestigation	Vi	: **Vi**sualization
M	: **M**ethodology	R	: **R**esources	Su	: **Su**pervision
So	: **So**ftware	D	: **D**ata Curation	P	: **P**roject administration
Va	: **Va**lidation	O	: Writing - **O**riginal Draft	Fu	: **Fu**nding acquisition
Fo	: **Fo**rmal analysis	E	: Writing - Review & **E**diting		

CONFLICT OF INTEREST STATEMENT

Authors state no conflict of interest.

DATA AVAILABILITY

The data that support the findings of this study are available from the corresponding author, [EPEC], upon reasonable request.

REFERENCES

[1] S. Kemp, “Digital 2024: 5 billion social media users-we are social UK,” 2024. [Online]. Available: https://wearesocial.com/uk/blog/2024/01/digital-2024-5-billion-social-media-users/. (Date accessed: Feb. 03, 2024).

[2] B. Dawadi, A. Thapa, R. Guragain, D. Karki, S. Upadhaya, and S. Joshi, “Routing Performance Evaluation of a Multi-Domain Hybrid SDN for Its Implementation in Carrier Grade ISP Networks,” *Applied System Innovation*, vol. 4, no. 3, p. 46, Jul. 2021, doi: 10.3390/asi4030046.

[3] S. Vissicchio, L. Vanbever, and O. Bonaventure, “Opportunities and research challenges of hybrid software defined networks,” *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 2, pp. 70–75, Apr. 2014, doi: 10.1145/2602204.2602216.

[4] Y. Guo *et al.*, “SOTE: Traffic engineering in hybrid software defined networks,” *Computer Networks*, vol. 154, pp. 60–72, May 2019, doi: 10.1016/j.comnet.2019.03.008.

[5] C. Chaudet and Y. Haddad, “Wireless Software Defined Networks: Challenges and opportunities,” in *2013 IEEE International Conference on Microwaves, Communications, Antennas and Electronic Systems (COMCAS 2013)*, Oct. 2013, pp. 1–5, doi: 10.1109/COMCAS.2013.6685237.

[6] A. Drescher, “A Survey of Software-Defined Wireless Networks,” *Washiongton University in St.Louis*, pp. 1–15, 2014.

[7] J. F. G. Orrego and J. P. U. Duque, “Throughput and delay evaluation framework integrating SDN and IEEE 802.11s WMN,” in *2017 IEEE 9th Latin-American Conference on Communications (LATINCOM)*, Nov. 2017, pp. 1–6, doi: 10.1109/LATINCOM.2017.8240186.

[8] D. F. T. Pontes, M. F. Caetano, G. P. R. Filho, L. Z. Granville, and M. A. Marotta, “On the Transition of Legacy Networks to SDN-An Analysis on the Impact of Deployment Time, Number, and Location of Controllers,” in *Proceedings of the IM 2021-2021 IFIP/IEEE International Symposium on Integrated Network Management*, 2021, pp. 367–375.

[9] Y. Al Mtawa, A. Haque, and H. Lutfiyya, “Migrating from Legacy to Software Defined Networks: A Network Reliability Perspective,” *IEEE Transactions on Reliability*, vol. 70, no. 4, pp. 1525–1541, Dec. 2021, doi: 10.1109/TR.2021.3066526.

[10] Y. Guo, H. Luo, Z. Wang, X. Yin, and J. Wu, “Routing optimization with path cardinality constraints in a hybrid SDN,” *Computer Communications*, vol. 165, pp. 112–121, Jan. 2021, doi: 10.1016/j.comcom.2020.11.004.

[11] S. Viuche, E. Estupiñán, and J. Martínez, “Performance Analysis of SDN Controllers using RTT, Jitter and Throughput Metrics,” *Libro de memorias*, vol. 14, pp.808-835, Nov. 2023.

[12] Y. Cheng, H. Zhou, X. Gao, J. Zheng, and G. Chen, “Optimizing incremental SDN upgrades for load balancing in ISP networks,” *Theoretical Computer Science*, vol. 962, p. 113927, Jun. 2023, doi: 10.1016/j.tcs.2023.113927.

[13] M. Paliwal and K. K. Nagwanshi, “Effective Flow Table Space Management Using Policy-Based Routing Approach in Hybrid SDN Network,” *IEEE Access*, vol. 10, pp. 59806–59820, 2022, doi: 10.1109/ACCESS.2022.3180333.

[14] D. Kreutz, F. M. V. Ramos, P. E. Verissimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, “Software-defined networking: A comprehensive survey,” *Proceedings of the IEEE*, vol. 103, no. 1, pp. 14–76, Jan. 2015, doi: 10.1109/JPROC.2014.2371999.

[15] F. D. O. Silva, J. H. D. S. Pereira, P. F. Rosa, and S. T. Kofuji, “Enabling future internet architecture research and experimentation by using software-defined networking,” in *Proceedings-European Workshop on Software Defined Networks, EWSDN 2012*, Oct. 2012, pp. 73–78, doi: 10.1109/EWSDN.2012.24.

[16] T. Abhishek, “Management, Control and Data plane-Cisco Community,” *Cisco*, 2019. https://community.cisco.com/t5/switching/management-control-and-data-plane/td-p/2803553 (accessed May 03, 2024).

[17] F. B. Carvalho, R. A. Ferreira, I. Cunha, M. A. M. Vieira, and M. K. Ramanathan, “State Disaggregation for Dynamic Scaling of Network Functions,” *IEEE/ACM Transactions on Networking*, vol. 32, no. 1, pp. 81–95, Feb. 2024, doi: 10.1109/TNET.2023.3282562.

[18] C. Carthern, W. Wilson, and N. Rivera, “Cisco Networks: Engineers’ Handbook of Routing, Switching, and Security with IOS, NX-OS, and ASA, Second Edition,” in *Cisco Networks: Engineers’ Handbook of Routing, Switching, and Security with IOS, NX-OS, and ASA, Second Edition*, 2nd ed., Apress Berkeley, CA, 2021, pp. 1–1073, doi: 10.1007/978-1-4842-6672-4.

[19] M. Lyu, H. H. Gharakheili, C. Russell, and V. Sivaraman, “Hierarchical Anomaly-Based Detection of Distributed DNS Attacks on Enterprise Networks,” *IEEE Transactions on Network and Service Management*, vol. 18, no. 1, pp. 1031–1048, Mar. 2021, doi: 10.1109/TNSM.2021.3050091.

[20] M. D and A. Durresi, "A survey: Control plane scalability issues and approaches in Software-Defined Networking (SDN)," *Computer Networks*, vol. 112, pp. 279–293, 2017, doi: 10.1016/j.comnet.2016.11.017.
[21] M. Bano, A. Qayyum, R. N. Bin Rais, and S. S. A. Gilani, "Soft-Mesh: A Robust Routing Architecture for Hybrid SDN and Wireless Mesh Networks," *IEEE Access*, vol. 9, pp. 87715–87730, 2021, doi: 10.1109/ACCESS.2021.3089020.
[22] E. Haleplidis, K. Pentikousis, S. Denazis, J. H. Salim, D. Meyer, and O. Koufopavlou, "Software-defined networking (SDN): Layers and Architecture Terminology," *RFC*, vol. 7426, pp. 1–35, Jan. 2015, doi: 10.17487/rfc7426.
[23] J. E. C. Guevara and C. A. C. Fajardo, "Arquitectura y funcionamiento de redes definidas por software (SDN)," Repositorio Universidad Distrital Francisco José de Caldas, 2022.
[24] S. Das, D. Talayco, and R. Sherwood, "Chapter 17 - Software-Defined Networking and OpenFlow," in *Handbook of Fiber Optic Data Communication (Fourth Edition)*, Fourth Edition., C. DeCusatis, Ed., Oxford: Academic Press, 2013, pp. 427–445, doi: 10.1016/B978-0-12-401673-6.00017-9.
[25] Z. Latif, K. Sharif, F. Li, M. M. Karim, S. Biswas, and Y. Wang, "A comprehensive survey of interface protocols for software defined networks," *Journal of Network and Computer Applications*, vol. 156, p. 102563, Apr. 2020, doi: 10.1016/j.jnca.2020.102563.
[26] Open Networking Foundation, "SDN Migration Considerations and Use Cases," ONF Whitepapaer, 2014. [Online]. Available: https://opennetworking.org/wp-content/uploads/2014/10/sb-sdn-migration-use-cases.pdf, (accessed: Feb. 03, 2024).
[27] S. Badotra and S. N. Panda, "Software-Defined Networking: A Novel Approach to Networks," *Handbook of Computer Networks and Cyber Security*, Springer, Cham, 2020, doi: 10.1007/978-3-030-22277-2_13.
[28] D. Liu, B. Barber, and L. DiGrande, "CHAPTER 5 - Routing Protocols: RIP, RIPv2, IGRP, EIGRP, OSPF," in *Cisco CCNA/CCENT Exam 640-802, 640-822, 640-816 Preparation Kit*, Boston: Syngress, 2009, pp. 169–196, doi: 10.1016/B978-1-59749-306-2.00009-9.
[29] P. Singh and T. Sood, "A Comparative Analytical Survey on RIP, EIGRP and OSPF," in *2023 2nd Edition of IEEE Delhi Section Flagship Conference (DELCON)*, Rajpura, India, 2023, pp. 1-6, doi: 10.1109/DELCON57910.2023.10127233.
[30] C.L. Hedrick, "RFC 1058 Routing Information Protocol," RFC Editor, Jun. 1988, doi:10.17487/RFC1058.
[31] D. Savage, J. Ng, S. Moore, D. Slice, P. Paluch, and R. White, "RFC 7868 Cisco's Enhanced Interior Gateway Routing Protocol (EIGRP)," RFC Editor, May 2016, doi:10.17487/RFC7868.
[32] J. Moy, "STD 54 RFC 2328 OSPF Version 2," RFC Editor, Apr. 1998, doi: 10.17487/rfc2328.
[33] B. Constantine, G. Forget, R. Geib, and R. Schrage, "RFC 6349 Framework for TCP Throughput Testing," RFC Editor, Aug. 2011, doi: 10.17487/rfc6349.
[34] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, "RFC 1889 RTP: A Transport Protocol for Real-Time Applications," RFC Editor, Jan. 1996, doi: 10.17487/rfc1889.
[35] L. Shengjin, W. Rongtao, and L. Junjie, "Power-grid services oriented coordinated and unified control technology of IP and optical networks," in *2018 IEEE 4th International Conference on Computer and Communications, ICCC 2018*, Dec. 2018, pp. 885–888, doi: 10.1109/CompComm.2018.8781003.

BIOGRAPHIES OF AUTHORS

Santiago Nicolas Viuche is a Telecommunications Engineer from Universidad Militar Nueva Granada. During his major, he joined the GISSIC research group (Maxwell hotbed of research), which led him to learn about some research methods. He can be contacted at email: est.santiago.viuche@unimilitar.edu.co.

Edith Paola Estupiñán Cuesta received the M.Sc. degree in Electronic Engineering from Pontifical Xavierian University in 2013. She is currently working at Universidad Militar Nueva Granada. Her research interests include mobile network, traffic analysis, and data and management network. She can be contacted at email: edith.estupinan@unimilitar.edu.co.

Juan Carlos Martínez Quintero received the M.Sc. degree in Autonomous Systems of Production from in Universidad Tecnologica de Pereria in 2013. He is currently profesor at Universidad Militar Nueva Granada. His research interests include mobile networks, SDR, communication systems, and digital signal processing. He can be contacted at email: juan.martinezq@unimilitar.edu.co.